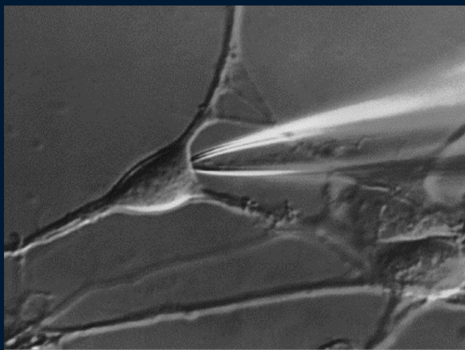
Tutorial 2.73

Patchmaster

Tutorial



HEKA Elektronik	Phone	+49 (0) 6325 / 95 53-0
Dr. Schulze GmbH	Fax	+49 (0) 6325 / 95 53-50
Wiesenstrasse 71	Web Site	www.heka.com
D-67466 Lambrecht/Pfalz	Email	sales@heka.com
Germany		support@heka.com

HEKA Electronics Inc.	Phone	+1 902 624 0606
643 Highway #14	Fax	+1 902 624 0310
R.R. #2	Web Site	www.heka.com
Chester, NS B0J 1J0	Email	nasales@heka.com
Canada		support@heka.com

HEKA Instruments Inc.	Phone	+1 516 882-1155
2128 Bellmore Avenue	Fax	+1 516 467-3125
Bellmore, New York 11710-5606	Web Site	www.heka.com
USA	Email	ussales@heka.com
		support@heka.com

Title Page: Cultivated Nerve Cell, soma contacted by a patch pipette; Courtesy of Prof. Sakmann, Planck-Institut fr medizinische Forschung, Heidelberg, Germany

COPATU/6

Contents

Part I

PATCHMASTER for Beginners

1. The First Experiment

This chapter will guide you briefly through the main features of the PATCHMASTER program and should take you a maximum of about 2 hours to read it. It briefly describes how a very simple first experiment with PATCHMASTER could look like. Of course, you will not have to do a *real* experiment, instead you should use the model circuit to *simulate* the conditions of a patch-clamp recording. The reader should not worry about options that are unclear, because more detailed descriptions of all of the mentioned steps are to follow. This section is designed for users that can't wait to get something done with PATCHMASTER. The basic requirements for starting the program and for performing a simple experiment are outlined. For more detailed descriptions of the features, refer to the PATCHMASTER reference manual.

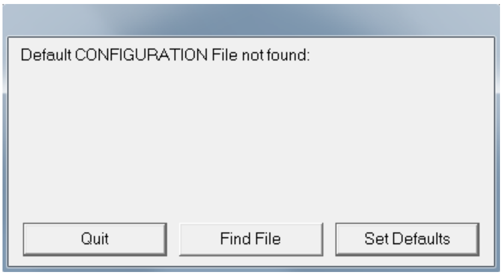
In the following it is assumed that the hard- and software have already been set up correctly. Please refer to the corresponding manuals for the hard- and software installation. It is also assumed that we start from the scratch, so we have a fresh and native installation of the PATCHMASTER software. Further, the following instructions refer only to the operation of an EPC 10 amplifier. If you plan to use the EPC 10 Double or Triple you should also first read the chapter `Amplifier Window` in the PATCHMASTER reference manual to get an idea of the basic amplifier operation.

1.1 Configuration

Your very first steps are:

- Turn on the amplifier.
- Turn on the computer.
- Start PATCHMASTER.

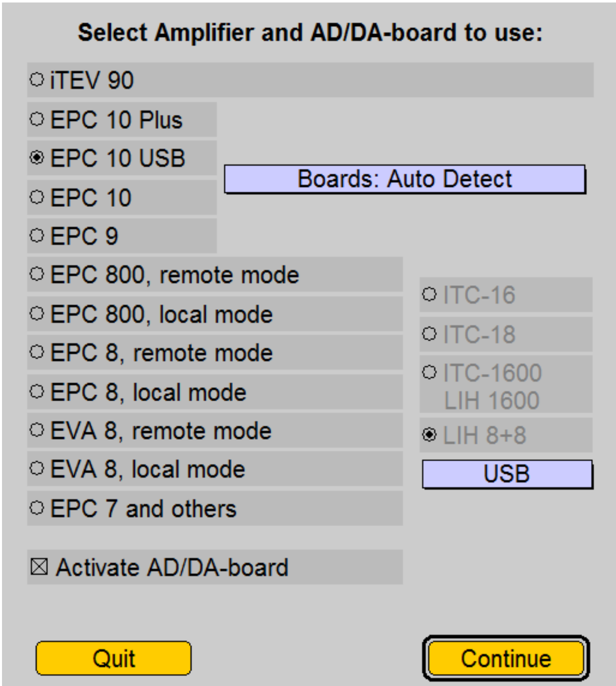
Assuming that it is the first time you run the PATCHMASTER software after the installation the following window appears:



At this point you do not have a so-called ”Configuration File” (i.e., a file with the extension `*.set` that contains all of your individual program settings). PATCHMASTER asks you what to do next:

- Quit: The PATCHMASTER software will be closed.
- Find File: If you have already a configuration file (e.g. `Patchmaster.set`) but it is e.g. not stored in the ”Patchmaster” folder or the file name is different, select *Find File* for selecting the appropriate folder where the file is stored in.
- Set Defaults: If you do not have a configuration file at all, which is the case right now, select *Set Defaults*.

After selecting *Set Defaults* PATCHMASTER will generate the default settings and comes up with a new dialog window:



Now you have to select your appropriate amplifier and interface in the offered list. It might be also necessary to distinguish between built-in PCI cards and USB connections.

There are two major settings you have to do:

- Amplifier Selection: Select the amplifier you are working with. If it is not in the list, select the EPC 7.
- AD/DA Board Selection: If your amplifier is not an EPC 10 (nor EPC 10 Double, Triple or Quadro) you will also have to define the AD/DA converter you use (ITC-16, ITC-18, or LIH 1600) on the

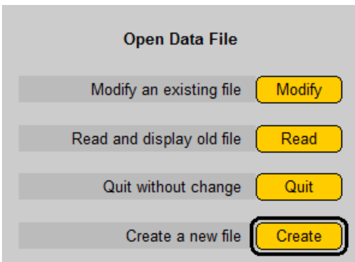
right side. In case you are running PATCHMASTER with multiple PCI boards, the active one has to be specified as well. If you are using an USB adapter card (e.g. USB-16 or USB-18) please press the *USB* button below the interface selection field to activate the USB connection. In this example, you cannot select an AD/DA-board (the selections are disabled), since the EPC 10 uses its built-in AD/DA converter.

The number of amplifier boards will be detected automatically when *Activate AD/DA-board* is active.

After you made your selection and pressed the *Continue* button the hardware will be initialized and PATCHMASTER will now look for file paths and the default files in the "PatchMaster" folder inside the "HEKA" folder.

If PATCHMASTER cannot find e.g. your `*.pgf` file, it will write a message into the Notebook window and will create a default file (`DefPgf_v9.pgf`). There may be other paths missing and PATCHMASTER 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 1.1.2 on page 9).

A new dialog window appears asking whether you wish to create a new experiment or just want to analyze some data:



There are four possibilities:

- Modify: Opens an existing experiment for modification, i.e. you can delete or add further experimental data to a file.

- Read: Opens and displays an existing experiment. The file will be write protected, so that modification (or loss) of data is prevented.
- Quit: Cancels the dialog.
- Create: Allows you to create a new experiment file.

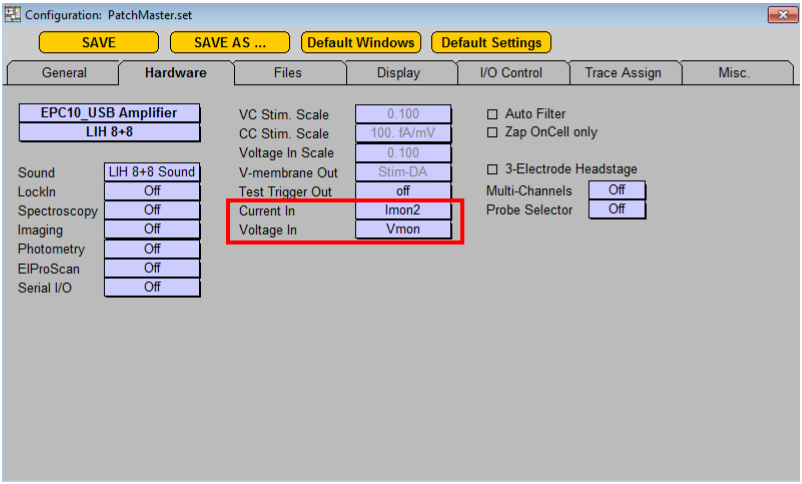
Select the *Create* option to start with a new experiment. You can call the file whatever you like, e.g. `Tutorial.dat`.

> ***Note:*** *A* PATCHMASTER *experiment consists of at least 3 files, the raw data (`*.dat`), the pulse protocols used (`*.pgf`) and a file that contains the amplifier settings and structure of your experiment (`*.pul`). You do not 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". For more information, see* PATCHMASTER *reference manual,* ***`Appendix I: File Overview`****.*

1.1.1 Hardware

PATCHMASTER will open some windows: the most obvious one is the Oscilloscope window. We will deal with that window 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.

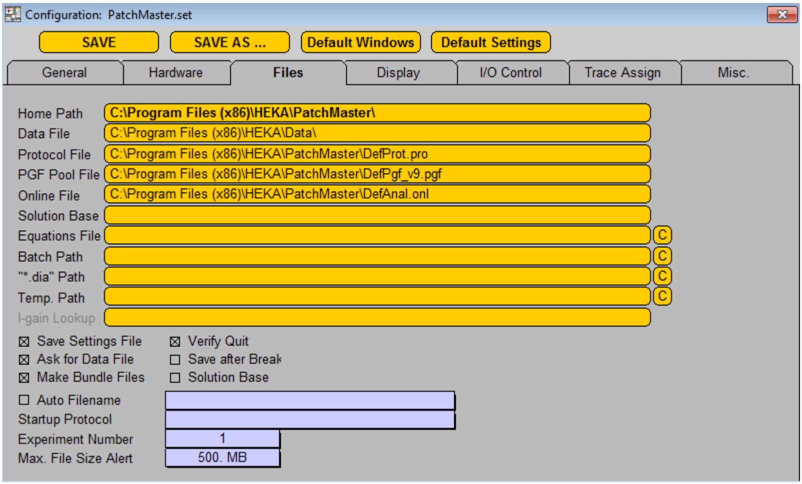
To open it, select `Configuration` from the drop-down menu `Windows` and select the *Hardware* tab.



When you are using e.g. an EPC 10 Single amplifier the default channels for current and voltage are named *Imon2* and *Vmon*, respectively. We have to know this later on when we create a stimulation sequence in the Pulse Generator.

1.1.2 Files and Paths

In order to tell PATCHMASTER where to look for the relevant files and where to store your data, you need to set up the paths and files. Therefore, we select the *Files* tab to customize the paths for the PATCHMASTER files.

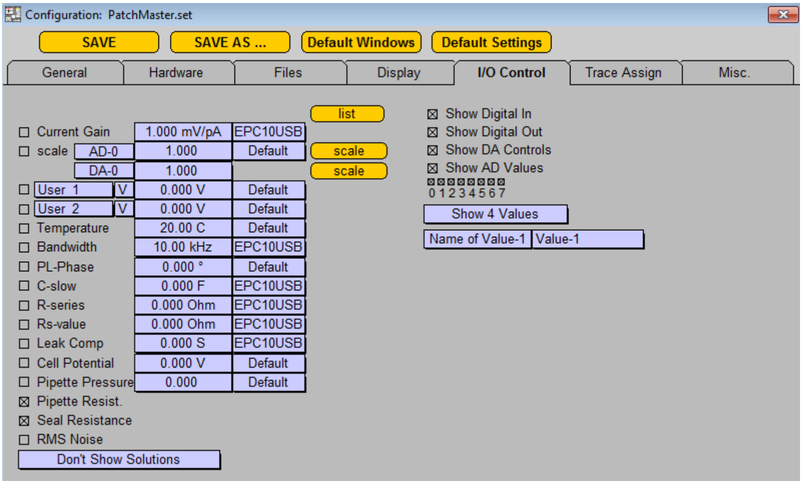


Usually, if you install PATCHMASTER with its default settings, you can leave these entries untouched.

In case you install into other directories, though, please adapt the paths, e.g., *Home Path*, *Data File*, *Protocol File*, *PGF Pool File etc.* according to your local installation settings.

1.1.3 I/O Control

So far, we specified the most important settings. In the *I/O Control* tab of the Configuration window there is a list of further values that are acquired and stored together with the experiment. These parameters can be input via different means:



- Either sampled through a free AD channel (*Source* = AD-0... AD-4)
- Derived directly from the amplifier (*Source* = EPC) like the settings of the *C-slow* compensation or
- Typed in by the user during the running experiment (*Source* = Default).

The checkboxes in the parameter list left to each parameter determine whether the parameter is displayed in the I/O-Control window.

> ***Note:*** *The checkboxes to the left of each parameter only define if the corresponding setting will be visible within the* PATCHMASTER *session. Irregardless of this setting,* PATCHMASTER *will always store every parameter during data acqui-*

sition. These data can be viewed in the Parameter *window (see 1.5.4 on page 36).*

1.1.4 Saving the Configuration

Finally, you can save the configuration:

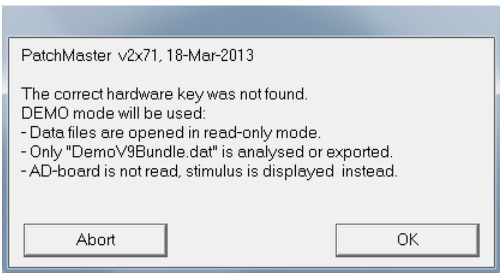
- Click *SAVE* if you want to save the file under the default name `Patchmaster.set` or
- Click *SAVE AS* if you want to save the file under another name. This is simply for your personal use, since PATCHMASTER will always start with the default file `Patchmaster.set`.

1.1.5 Troubleshooting

We also want to describe some initial error messages and their significance.

- *"The correct hardware key was not found."*

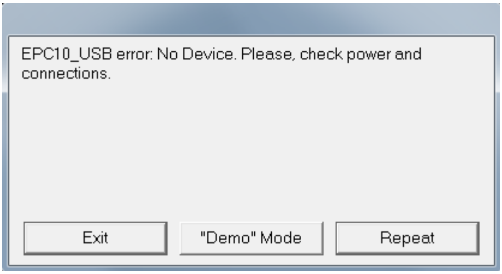
  You may not have the hardware key correctly installed. PATCHMASTER will continue to run in *Demo* mode, with a stimulus simulation of the AD board if you press the *OK* button. For installation of the hardware key, we refer to the `Installation Guide`.



  For further information about the *Demo* we refer to `Demo mode`, 1.10.1 on page 53.

- *"EPC10_USB error. No Device. Please, check power and connections."*

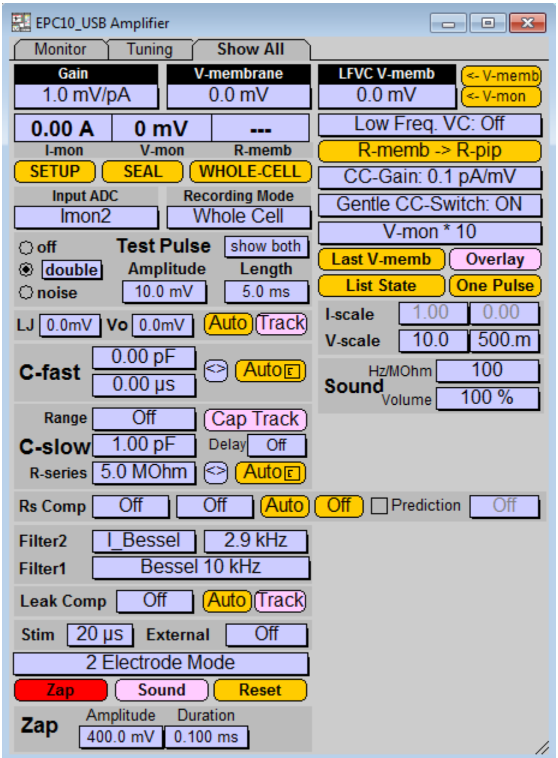
  You may not have connected any AD/DA hardware. PATCHMASTER will recognize this and will ask you how to proceed:

  Note: *Depending on your hardware the content of the error message can vary.*

  – Exit: PATCHMASTER will be closed.
  – "Demo" Mode: PATCHMASTER will run in the *Stand-alone* mode (see 1.10.2 on page 54).
  – Repeat: If you just forgot to turn on the power of the EPC 10, do so and select *Repeat*.

1.2 Controlling the Amplifier



The EPC 10 Amplifier window provides the amplifier control functions when an EPC 10 amplifier is used (the picture is for an EPC 10 Single). More detailed descriptions of the functions of the EPC 10 versions and their control windows are given in the corresponding amplifier manuals.

1.2.1 Basic Protocols

At the top left side of the EPC 10 Amplifier window, you can find three yellow buttons. These predefined protocols are essential for the patch-clamp procedure.

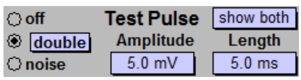
SETUP: Resets all parameters (with the exception of LJ and V_0), and sets the *Test Pulse*, the *Recording Mode*, the *Gain* and the start values for *C-slow* and *R-series* determination.

SEAL: Switches the *Gain* range to a typical setting for a cell-attached patch recording. Further, the *Recording Mode* is set and an *Auto C-fast* compensation is performed.

WHOLE-CELL: Switches the *Gain* range to a typical setting for a whole-cell recording, sets initial *C-slow* estimates, and invokes an *Auto C-slow* compensation.

The command lines for the operation of the protocols can be found in the Protocol Editor window. They can be modified and other commands can be recorded using the *Macro Recording* function (see Protocol Editor window). Since the Protocol Editor (see `Automating the Data Acquisition`, 1.7 on page 45) allows for more complex automation, we will not discuss protocols further at this point.

1.2.2 Test Pulse



The *Test Pulse* is applied to the pipette whenever you activate the amplifier by bringing the Amplifier window to the front. *Test Pulses* are added to the holding potential and applied to the pipette; the current responses are sampled and displayed. *Test Pulses* are applied at maximal rates depending on the pulse length specified.

The *Test Pulse* is defined in two different windows:

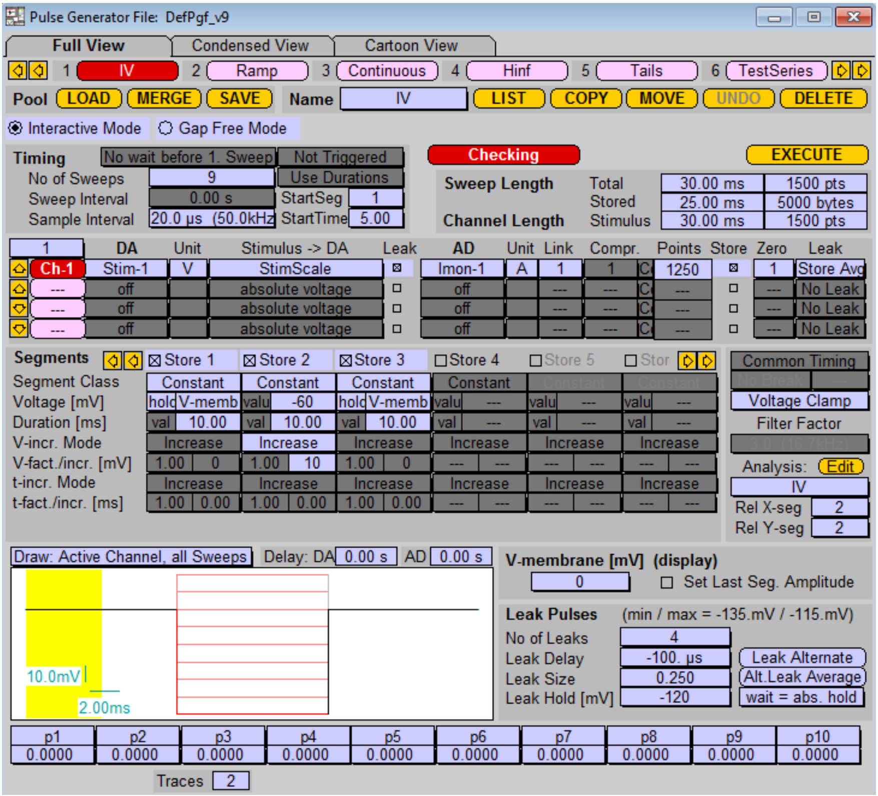
- In the Amplifier window you set the parameters for the *Test Pulse*, like *Amplitude*, *Duration* and pulse type. "Current" or "Voltage" means that the current *Trace* or the voltage *Trace* is displayed in the Oscilloscope, respectively, every time the *Test Pulse* runs. While "show both" displays the current and the voltage *Trace* simultaneously. Use this if you frequently want to apply *Test Pulses* in the *Current Clamp* configuration.

- In the Configuration window (*Misc.* tab) you can set whether the *Test Pulse* shall be scaled.

1.3 Setting up a simple Pulse Sequence

The PATCHMASTER software allows you to create stimulation sequences that range from simple rectangular pulses to highly complicated stimulation patterns. The stimulus templates are edited in the Pulse Generator window.

To open it, select Pulse Generator from the `Windows` drop-down menu.

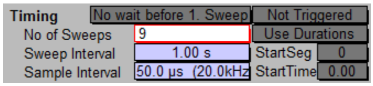


A stimulation sequence consists of an arbitrary number of pulse *Segments* that have *Constant, Ramp, Continuous, Square, Sinusoidal* or *Chirp* design. The file `DefPgf_v9.pgf`, distributed with the software release, is usually installed into the "Patchmaster" folder inside the "HEKA" folder and contains several pulse protocols which are a good starting point to create your own ones.

Click on a free position in the PGF pool (pink button). If there is no free position, click the right arrow unless you reach the end of the pool. PATCHMASTER will ask you for a new entry name. We will call our new sequence "SPS" (Simple Pulse Sequence).

1.3.1 Setting up the Timing

We want to create a PGF sequence that gives us a current-voltage relationship. The response to 9 depolarizing pulses in steps of 10 mV given at an interval of 1 s has to be studied.



In the *Timing* section set *No of Sweeps* to "9" and the *Sweep Interval* to "1". Choose the *Sample Interval*: here 50 μs. To edit the fields, double-click in the corresponding field and enter the number.

Usually PATCHMASTER will wait the time defined in *Sweep Interval* before starting the pulse sequence. However, right now we want the sequence to start immediately after activating it, so please select *No wait before 1. Sweep* from the selection field next to *Timing*. Also the option *Start Segment* should be switched off (select "0").

1.3.2 Defining the Segments

The section *Segments* of the Pulse Generator defines the actual pulse sequence to be applied. It will consist of three parts:

1. Holding the cell at a defined holding potential.

2. Depolarizing step.

3. Holding the cell at a defined holding potential.

The individual parts of the pulse protocol are called *Segments*. At the beginning, the protocol has only one *Segment* of 10 ms duration. To add the additional two *Segments*, mark the *Store* option for the second and the third *Segment*. The result should look like the following:

Segments		Store 1		Store 2		Store 3
Segment Class	Constant		Constant		Constant	
Voltage [mV]	valu	0	valu	0	valu	0
Duration [ms]	val	10.00	val	10.00	val	10.00
V-incr. Mode	Increase		Increase		Increase	
V-fact./incr. [mV]	1.00	0	1.00	0	1.00	0
t-incr. Mode	Increase		Increase		Increase	
t-fact./incr. [ms]	1.00	0.00	1.00	0.00	1.00	0.00

Although you can edit the *Segments* in any order, it is often advisable to start by defining the length of the individual *Segments*. Since we want to give all three *Segments* the same length, we can use the *PGF parameter* function.

At the bottom of the window you will find this row:

p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

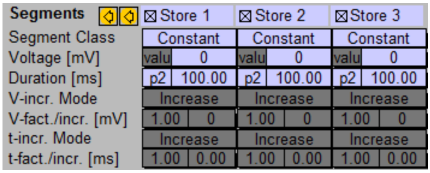
"p1" to "p10" are called "PGF parameters". You can use them as variables in the *Segment* settings for *Voltage* or *Duration*. This allows you to change multiple settings with changing only one parameter. Proceed as follows:

1. Select "p2" instead of "val" from the drop-down menu right before the *Duration* value in the first *Segment*.

2. Click on the number under the "p2" entry in the PGF parameters row and enter "0.1".

3. Now you can see that the value in the *Segments* section has changed.

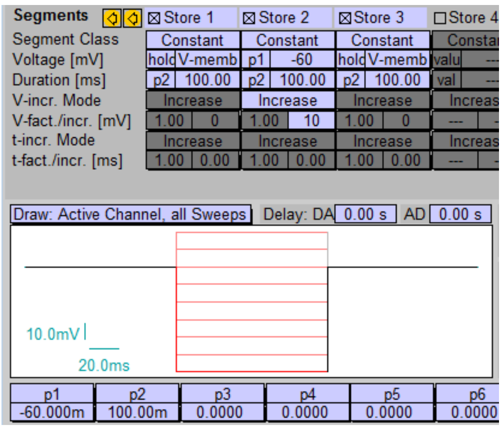
4. Choose "p2" for the other two *Segments* too. All *Segments* are set to "p2 = 100" now.



The first and last *Segment* should be at the holding potential, so select "holding" instead of "value" from the drop-down menu right before the voltage value. The value in voltage changes to "V-memb" (i.e. the actual pipette holding potential at the time of executing the protocol).

Change the value in the second *Segment* to "p1" and set the *PGF parameter* "p1" to -0.06.

Then set the *V-incr. [mV]* field to "10". This will instruct PATCHMASTER to jump to -60 mV when it first executes the protocol and then always increment this *Segment* by 10 mV for the following 8 repeats (-50, -40, -30, . . . , +20 mV). The *Segments* and their preview should look like the following:

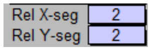


If a segment is set to "V-memb", PATCHMASTER will fill in the actual holding voltage at time of data acquisition. In the sequence cartoon of the Pulse Generator the *Segments* are filled in with the value entered under

V-membrane [mV] (display). Thus, to make the cartoon look realistic, you may want to enter a typical holding voltage (e.g. to -80 mV) into the field.

> ***Note:*** *This value does not affect your measurement – it is only used for previewing the Sweep!*

1.3.3 Defining the Segment for Online Analysis



Maybe you wondered why one segment is drawn in red color in the preview while the rest is black. PATCHMASTER can perform an Online Analysis whenever you run or replay an experiment. 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 is done separately for the *Segment* that delivers the X- and the Y-value. Set both values to "2".

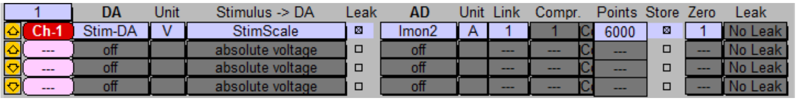
Your later analysis will of course not be restricted to the *Segments* you define here. In the *Analysis Functions* section of the Online Analysis window you can set a positive or negative *Segment Offset* that will be added to the *Relevant Segment*, thus allowing you to analyze other *Segments*. For more information, see `Defining the Analysis Functions`, 1.6.3 on page 40.

1.3.4 Setting the Output Channel and the AD Input

In the next step we define the AD and DA channels to be used for stimulation and acquisition of data in the sections DA channels and AD channels. For the EPC 10, some of these channels are predefined:

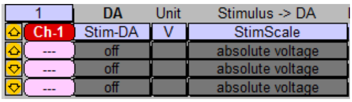
- The voltage stimulus is always expected to go via *Stim-DA* (V-membrane Out).
- The current input is sampled via *Imon2* (Current In).
- The voltage is sampled from *Vmon* (Voltage In).

> ***Note:*** *The* EPC 10 *has 4 DA output channels (0...3) and 8 AD input channels (0...7). For the* EPC 10 *Single, the channels DA-0...2 and AD-0...4 are available. For the* EPC 10 *Double, the channels DA-0...1 and AD-0...2 are available. For the* EPC 10 *Triple, the channels DA-0 and AD-0 are available. The other channels are internally hardwired to the current and voltage output of the respective amplifiers.*

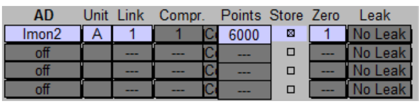


In the rows *Ch-1...* you set the parameters for each channel. The default channel is "1", the other channels (Channels = 2...) may be used to simultaneously record other data such as the potential, an amperometric signal, or a fluorescence.

DA and AD settings are independent from each other. Their reference is only given by the variable *Link* in the AD settings! So to prevent confusion here, we will split the above picture for a closer look.



The DA section on the left allows you to set the properties of the DA output, e.g., the stimulus signal. Note that the expression "channel" is used exclusively for the DA stimulus output! The output via *Stim-DA* and the *Unit* "V" are the default entries for the EPC 10.



The AD section on the right allows you to set the properties of the AD input, e.g., the acquired data. The input via *Imon2* and the *Unit* "A" are the default entries for EPC 10. The variable *Link* defines with which DA stimulation this AD input is associated, in our case to channel 1.

Remember that this *Link* variable allows you to associate several AD inputs to the same DA stimulation! The rationale behind the *Link* variable is that during analysis one has to know which stimulus was applied for a given data *Trace*.

The option *Store* will make sure that the acquired data can be stored to disk. For some protocols it might not be required to save the data (test PGF etc.), so you can disable this feature in these cases.

1.3.5 Other Settings in the Pulse Generator

There are a few more options in the right part of the Pulse Generator window that did not have to be changed in our case. Nevertheless, it is still important to know what they do: the setting *Voltage Clamp* will restrict the execution of the pulse protocol to the voltage-clamp modes only. Thus, PATCHMASTER will refuse to start this sequence if you are in the current-clamp mode and instead will produce an error message.

> ***Note:*** *A given pulse protocol only makes sense for Voltage- or Current-Clamp conditions, never for both modes. The option Any Mode in the* Pulse Generator *window is only there for special applications like photometry. If you want to be able to run a Current-Clamp sequence while you are in a Voltage-Clamp mode, you should create a protocol that switches to the Current-Clamp mode, and associate it with the pulse protocol.*

The section *Sweep/Channel Length* gives you some important information about the pulse protocol.

Sweep Length	Total	300.0 ms	6000 pts
	Stored	300.0 ms	12000 bytes
Channel Length	Stimulus	300.0 ms	6000 pts

Sweep Length: Maximal possible length of a *Sweep*, determined by the timing settings.

- Total: Denotes the total time needed for one *Sweep* of the given sequence in ms and points.
- Stored: Denotes the total time stored for one *Sweep* of the given sequence in ms and bytes. *Total* and *Stored* durations may be different when a *Start Seg.* and *Start Time* were set or when conditioning segments were used (e.g. segments with the *Store* button off).

Channel Length: Length of the actual DA stimulation. This can be shorter than the *Sweep Length*, e.g., a short trigger pulse.

- Stimulus: Denotes the time for the *Stimulus* signal in ms and points.

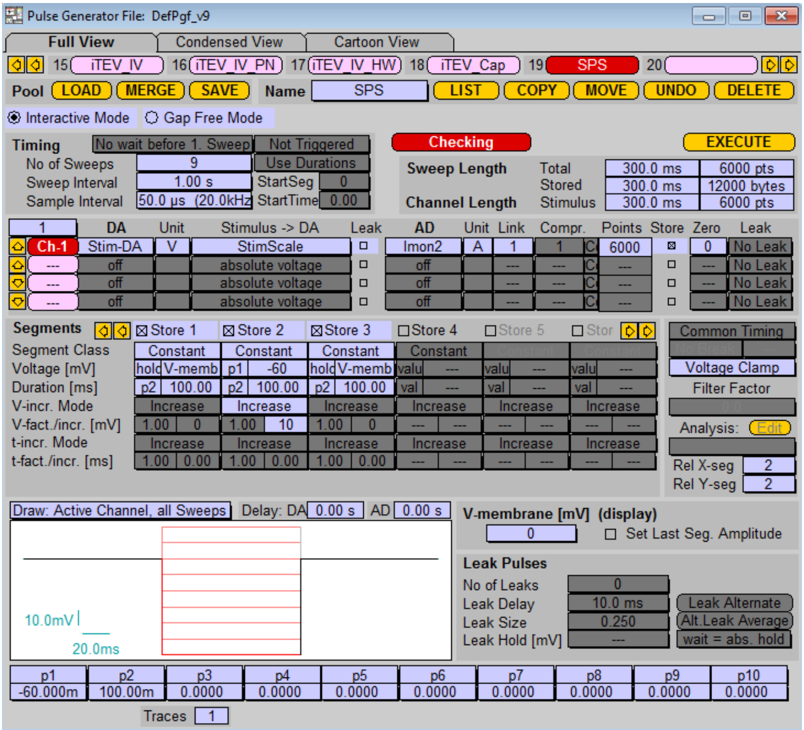
For our example, the value *Total* as the total length of stimulation calculates like this: Each *Sweep* has a duration of 100 + 100 + 100 = 300 ms sampled at an interval of 50 μs or a frequency of 20 kHz. This makes a total number of 6000 data points.

If, for example, there was a StartSegment 1 and a StartTime 5 ms, the first 5 ms (or 100 data points) would not be saved. But since we want to store the whole *Sweep* to disk, we did not specify a StartSegment.

> ***Note:*** PATCHMASTER *allows you per default a maximum of 5 channels with 262144 points each. These parameters can be adjusted in the* CONFIGURATION *window, provided your computer has enough RAM.*

This new, modified Pulse Generator file should now be stored to disk by clicking on "SAVE" and entering a name. The default file extension is `*.pgf`. Note that on program start PATCHMASTER will always load the file defined in the Configuration window; the default is DefPgf_v9.pgf. If another 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.

The resulting Pulse Generator window should look like this:

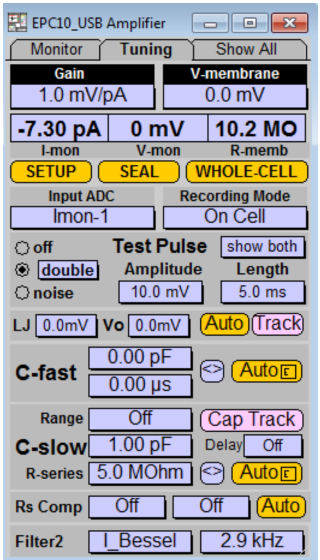


1.4 Starting the Experiment

Now it is time for the experiment. Therefore, attach the model circuit to the headstage.

1.4.1 Patching a Cell

Switch the model circuit into the "10 MOhm" setting to simulate a 10 MΩ pipette that is open to the bath solution.



Hit the space bar in the main dialog to activate the Amplifier window - if the Amplifier window is not in front, hit the space bar twice, the space bar toggles between the Control window and the Amplifier window.

As long as the Amplifier window is on top, the program will generate *Test Pulses* according to the settings in the *Test Pulse* section. A double pulse of 5 mV *Amplitude* and a *Length* of 5 ms per pulse will be output. The sampled current responses will be shown in the Oscilloscope window. The resistance of the pipette is calculated from the responses and displayed in the *R-memb* field.

Besides the fast *Test Pulses* (single or double) you can select the third entry in the *Test Pulse* pop-up list, which requires to specify a sequence from the Pulse Generator File. Instead of the fast *Test Pulses*, this sequence is then repeated continuously providing an alternative and quite flexible *Test Pulse* mode.

> ***Note:*** *The currently measured resistance of the pipette is always called R-memb because the program cannot distinguish between an open and a sealed pipette. As long as the pipette is open to the bath, R-memb corresponds to the pipette resistance.*

The command potential is controlled by the program via the control *V-membrane*. This variable always displays the physiological membrane potential, i.e., the *Recording Mode* is already taken into account reverting the polarity of the applied potential in *On Cell* and *Inside Out* modes.

> ***Note:*** *Most functions, such as canceling the offset current, setting the amplifier Gain or the holding potential, etc. should be obvious, but make sure that the Recording Mode is always set properly, because this setting will automatically determine the actual polarity of the voltage at the patch pipette!*

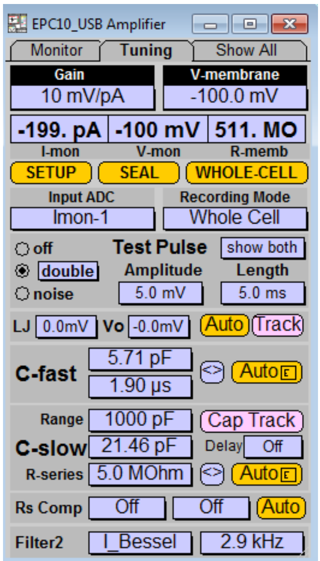
You can correct pipette offset potentials by adjusting the *Vo* value or you can alternatively click on the *Auto Vo* button to let PATCHMASTER do this correction automatically for you. The same is done by calling the protocol "SETUP", in this case, PATCHMASTER will also adjust the amplifier *Gain* and the *Test Pulse*. When the pipette potential is adjusted and you are ready to form a seal, store the value of the pipette resistance - which is the actual *R-memb* value that will be overwritten after forming the seal. This is done by clicking on *R-memb* → *R-pip*. This value is not changed any further, unless you click on *R-memb* → *R-pip* again.

> ***Note:*** *R-memb is updated as long as the Test Pulses are active, i.e. every time the* Amplifier *window is in front, and stored*

> *as variable Seal Resistance with every acquired Sweep (see* Parameters *window). The pipette resistance will be stored together with every acquired Sweep.*

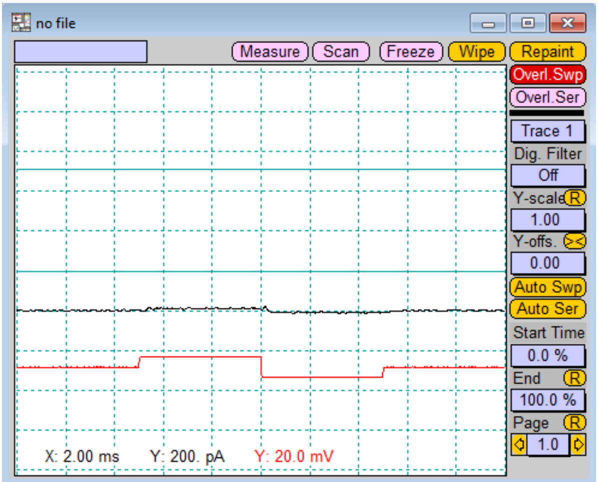
Now, simulate a pipette sealed to the membrane by switching the model circuit into the middle position. If you have an EPC 10, make an automatic fast capacitance cancellation by clicking on the *Auto C-fast* or "SEAL" protocol button. Otherwise, compensate your amplifier for the pipette capacitance of about 6 pF.

To break into the cell, set the switch of the model circuit to its bottom or "0.5 GOhm" position. If you have an EPC 10 make an automatic slow capacitance cancellation by clicking on the *Auto C-slow* or "WHOLE-CELL" protocol button. Otherwise, compensate your amplifier for the cell capacitance of about 20 pF. Watch the *R-memb* display that now shows "500 M" instead of "10 M". Change the pipette holding potential to -100 mV by either entering the value into the *V-membrane* field or use the left and right cursor keys. Now we are ready to run the pulse protocol we defined before.



1.4.2 Setting up the Display

Bring the Oscilloscope window to the front. Make sure that the button *Store* is highlighted in the Control window. If the Control window is not open yet, open it via the `Windows` menu. If the *Store* button is not highlighted PATCHMASTER will show the data but not write them to disk! If you did not create a file yet, PATCHMASTER will ask you to do this now.



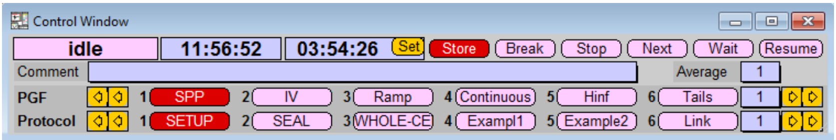
The bottom of the Oscilloscope window shows the buttons used to control the execution of sequences or protocols.

To see all *Sweeps* from one *Series*, activate the *Overl.Sweep* button in the Oscilloscope window; otherwise, the display will be erased before every *Sweep*.

Before we execute the "SPS" sequence (or "Series" in PATCHMASTER terminology, which describes a number of individual *Sweeps* based on the same Pulse Generator protocol) we will set up the display. Usually you can use the default settings of a new PATCHMASTER installation, but let us have a look at the `Display` menu. The following options should be activated: *Auto Show, Show Zero Line, Dimmed Overlay, Overlay Traces, Overlay Sweeps, Labeling → Grid + Labels.*

1.4.3 Starting the Sequence from the Control Window

In the Control window you can see two rows with either PGFs (e.g. "SPS") or protocols (e.g. "SETUP", "SEAL", "WHOLE-CELL").

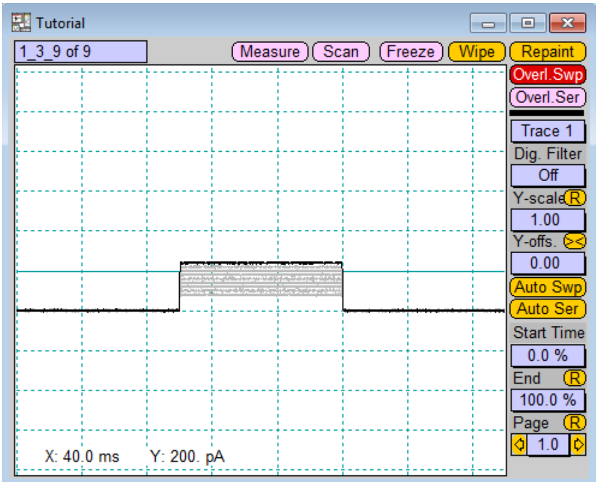


To start data acquisition directly, click on the "SPS" button or type "1" into the blue entry field.

> ***Note:*** *The numbering of PGFs or protocols might be different in your Control window. Either you scroll to the according position via the scroll arrows or you change the position in the Pulse Generator or Protocol Editor directly.*

1.4.4 Displaying the Data

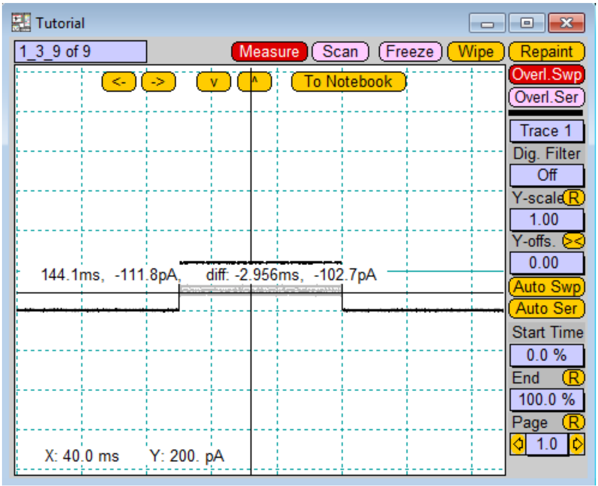
The pulse pattern we defined above is output via the specified DA channel and the response is shown in the Oscilloscope window. The last *Sweep* of the Series is shown in black color, the other *Sweeps* are gray since we activated *Dimmed Overlay*. The grid is drawn in green color and scaling values are given in the lower left side of the Oscilloscope.



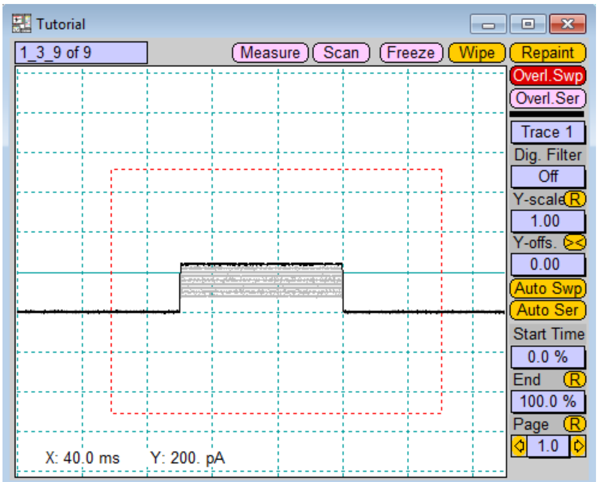
1.4.5 Changing the Display Settings

In case you want to have a closer look at your displayed data, you have various possibilities to change the display settings.

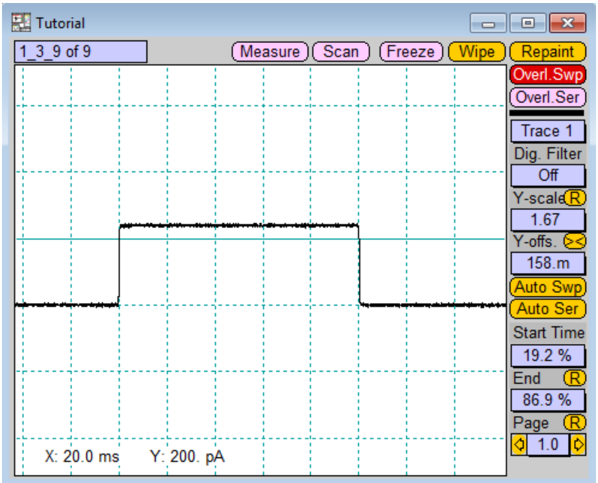
For example, you can do a quick check-up on the measured values. When you click on the button *Measure*, a cursor with two connected lines will be displayed in the Oscilloscope window. The data of the actual point will be displayed, and you can copy them into the Notebook via the *To Notebook* option.



In case the data is too small on your display, you can use the "lasso-ing" function. Start in the top left corner and press the left mouse button. Pull the opening red square to the appropriate size and release the mouse button.



The marked area will be set to fill the Oscilloscope screen. Note that the scaling has to be done for each *Trace* separately, even when you have selected the *Overlay* option! So the result looks like the following (only one *Sweep* is shown):



By using the yellow *Reset* buttons on the right ridge of the Oscilloscope window all changes of the Y- and X-axis scaling can be reset.

1.5 Handling of the Data

1.5.1 Saving the Data

To write the recorded data to disk, select `File` → `Update File` or close the experiment with `File` → `Close`. The latter will automatically store all files associated with the experiment.

To create a new file for data acquisition, select `File` → `New...`. PATCHMASTER will close the running experiment and open a new, empty one.

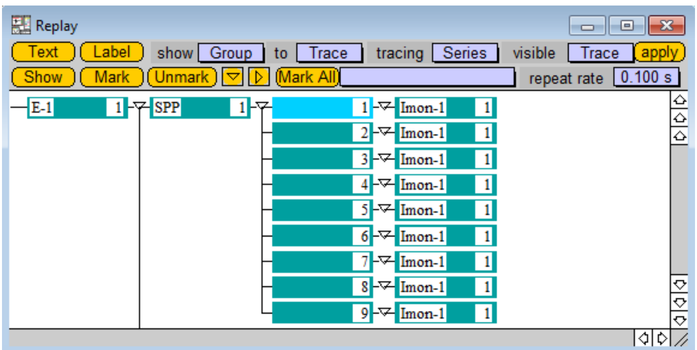
Remember: Recorded data can only be saved and/or replayed if the *Store* button was active during acquisition!

1.5.2 Replaying the Data

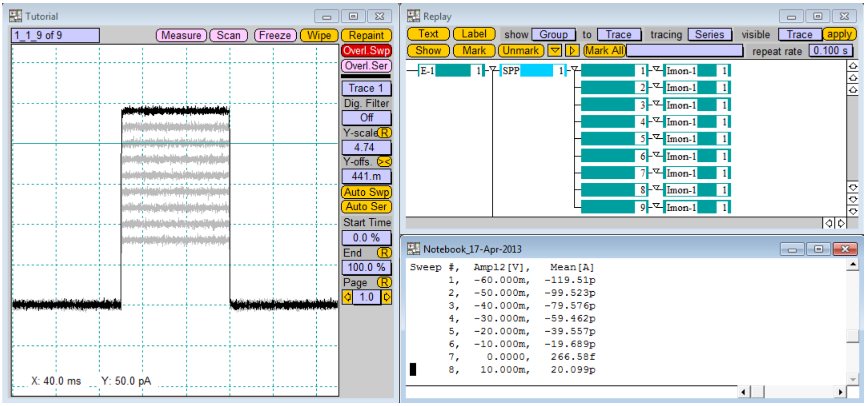
If – and only then! – the *Store* button was active, the structure of the stored data will be shown in the data tree of the Replay window. This is also the basis for the replay of data.

To open the Replay window select `Replay` in the `Windows` menu.

Double-click the "SPS" entry to replay the just recorded sequence; double-click a single *Sweep* to inspect 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 or double-click on the currently active *Group*, *Series*, *Sweep* or *Trace*, it will be displayed in the Oscilloscope. This may look like the following example:



While replaying the data, the Online Analysis will be calculated. For more information on the analysis options, see `Analyzing the Results`, 1.6 on page 38.

The `Replay` menu provides functions for modification of the tree entries. E.g., a single *Sweep*, a *Series*, or a whole group of *Series* can be removed by marking the item and then selecting *Replay* → *Delete*.

1.5.3 Exporting Data

You can export the data into various other file types via the export options in the `Replay` menu. PATCHMASTER cannot only export to plain text files, but also to formats for software like Igor Pro or MatLab.

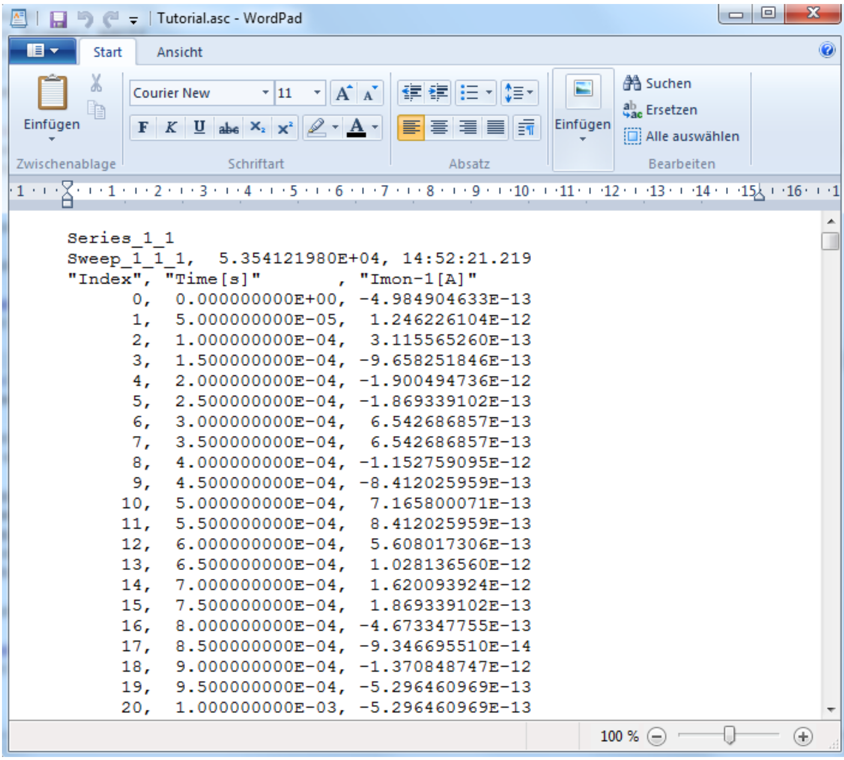
For example: to export raw data to comma-separated ASCII format, you would have to set the following options:

- Export Format: ASCII
- Export Mode: Traces

- ASCII option: Comma-separated and the linefeed type that would fit your operating system.

Select *Export* to export the data as it is displayed in the Oscilloscope or *Export Full Sweep* to export the data independent of the Oscilloscope settings. Then you are asked for a file name. The pre-set data extension for the output is `*.asc` (as for ASCII).

The resulting file would look like this in a ASCII viewer, e.g., WordPad:

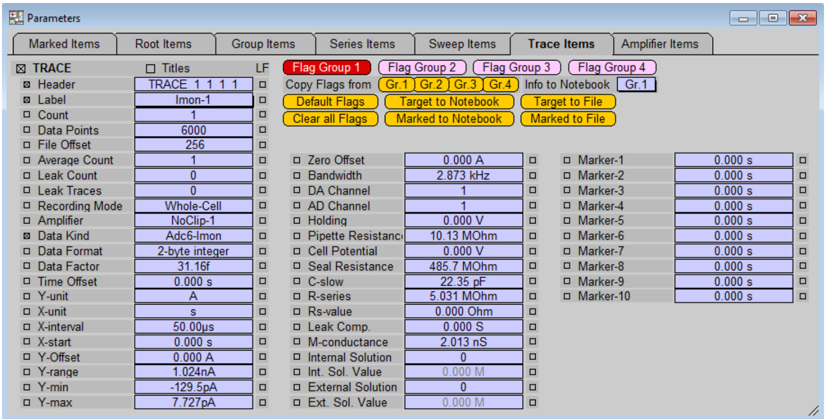



```
Series_1_1
Sweep_1_1_1,  5.354121980E+04, 14:52:21.219
"Index", "Time[s]"        , "Imon-1[A]"
      0,  0.000000000E+00, -4.984904633E-13
      1,  5.000000000E-05,  1.246226104E-12
      2,  1.000000000E-04,  3.115565260E-13
      3,  1.500000000E-04, -9.658251846E-13
      4,  2.000000000E-04, -1.900494736E-12
      5,  2.500000000E-04, -1.869339102E-13
      6,  3.000000000E-04,  6.542686857E-13
      7,  3.500000000E-04,  6.542686857E-13
      8,  4.000000000E-04, -1.152759095E-12
      9,  4.500000000E-04, -8.412025959E-13
     10,  5.000000000E-04,  7.165800071E-13
     11,  5.500000000E-04,  8.412025959E-13
     12,  6.000000000E-04,  5.608017306E-13
     13,  6.500000000E-04,  1.028136560E-12
     14,  7.000000000E-04,  1.620093924E-12
     15,  7.500000000E-04,  1.869339102E-13
     16,  8.000000000E-04, -4.673347755E-13
     17,  8.500000000E-04, -9.346695510E-14
     18,  9.000000000E-04, -1.370848747E-12
     19,  9.500000000E-04, -5.296460969E-13
     20,  1.000000000E-03, -5.296460969E-13
```

1.5.4 Exporting Parameters

For long series of data, you may want to get an overview of the settings and parameters with which the data were acquired.

For this, open the Parameters window (`Windows` menu). Here, all information concerning replayed or actual data is displayed. Via the checkboxes (flag options) you can select information that you want to export to the Notebook window or to a file.



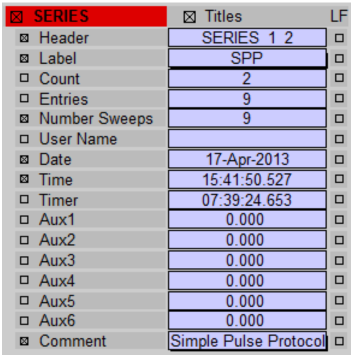
The window is structured as follows:

- The parameters are stored according to their data tree affiliation (*Root*, *Group*, *Series*, *Sweep*, *Trace* or *Amplifier* tabs).
- In the *Marked Items* tab all parameters with active checkboxes are displayed when the corresponding main checkbox (e.g. *TRACE*) is active, too.
- On the top middle side you can find the flag management and the export features.
- On the left, middle and right part of the window you can find the data tree entries.

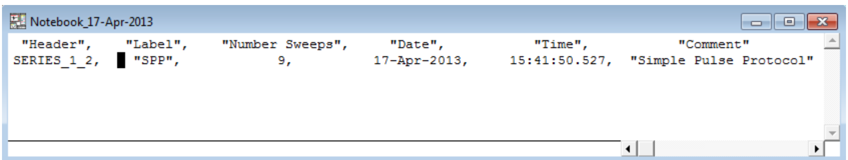
Select the data you want to export by checking the small checkboxes in front of the relevant entries. Note, that you have to check the main checkboxes, e.g., *ROOT*, *SERIES* etc., to export the other parameters of that group!

If you want to export the parameter names together with their corresponding values, check the option *Titles* above the parameter values. Otherwise, only the values will be exported. Then click on *Target to Notebook* to export the parameters into the Notebook window.

For a setting like the this:



where the parameters of the *Series* - that is the target in the Replay window – are exported with *Header*, *Label*, *Number Sweeps*, *Date*, *Time* and *Comment* (and *Titles* option), the result in the Notebook window may look like this:

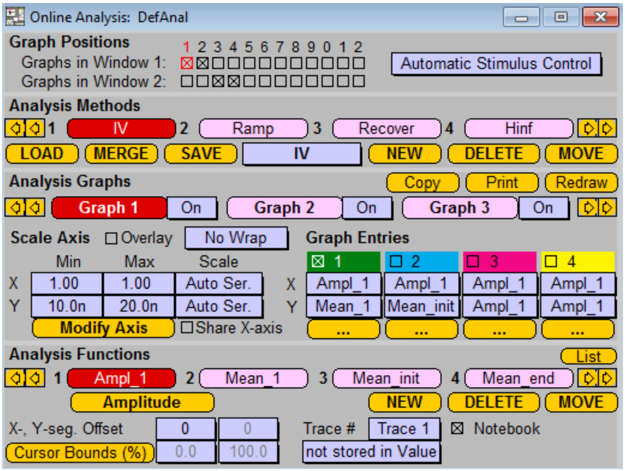


When you click on *Target to File*, the program will ask for a file name. The pre-set data extension for the output is `*.asc` (as for ASCII).

1.6 Analyzing the Results

1.6.1 Using the Online Analysis

The Online Analysis allows you to immediately calculate and display data that are based on the acquired *Traces*, thus giving you a fast overview over your results.



The highlighted *Analysis Method* is the one that will automatically be executed when you acquire or replay data.

PATCHMASTER can show such analysis results as columns in the Notebook window or plot them in the Online Analysis windows 1 or 2 after or during execution of a *Series* (based on the settings made in the various controls inside this window).

The Online Analysis is structured as follows:

1. Based on incoming data, a number of Analysis Functions are defined.
2. These functions produce analysis results based on the relevant *Segments* of the sequence.

3. These results are then displayed in the Notebook (if the *Notebook* option is checked) and/or shown in an *Online Graph* inside either Online Window 1 or Online Window 2.

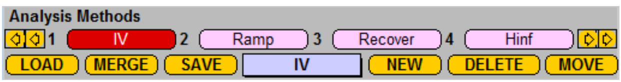
Elements of a graph are *Graph Entries*, i.e. couples of analysis results to be used as X- and Y-reference. Up to 4 *Graph Entries* fit into one graph; multiple graphs fit into Online Window 1 or 2.

The entire setting of the Online Analysis is called *Analysis Method*. An arbitrary number of such *Analysis Methods* can be saved in Online Analysis files (`*.onl`).

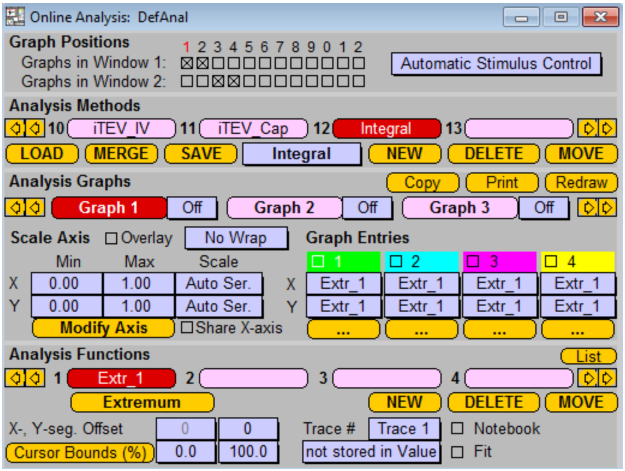
Thus, the first thing to do is to define *Analysis Functions*. Only then, the respective analysis results are placed as *Graph Entries* in graphs and windows.

1.6.2 Entering a New Analysis Method

Usually, you can set up a new *Analysis Method* by copying the data from one method to the other. However, for the purpose of this tutorial we will start from scratch.



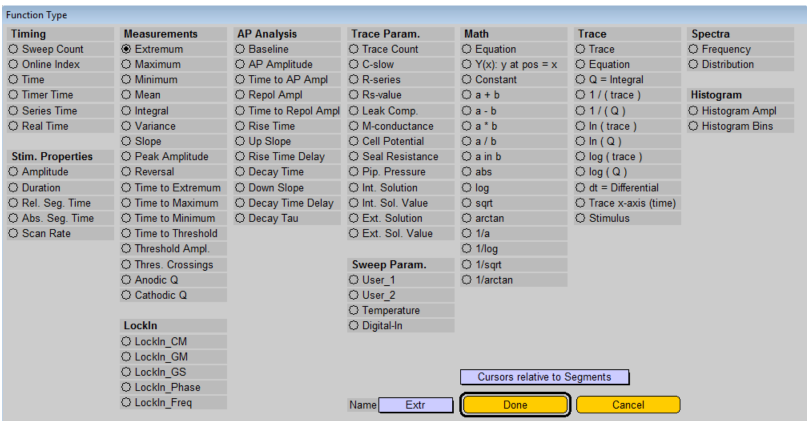
Click on *New* and enter the name "Integral" for the new *Analysis Method*.



It will be created and placed on the next free entry number.

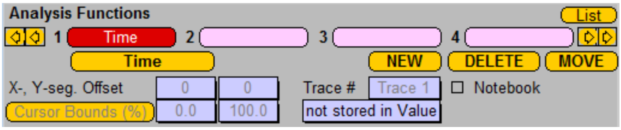
1.6.3 Defining the Analysis Functions

Extr_1 is given as default *Analysis Function*. Now, we need to customize our method. Click on *Extremum* to open the Function Type dialog.



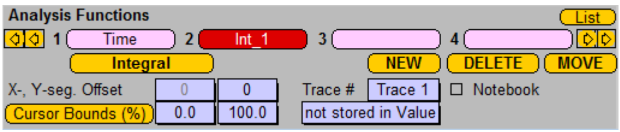
Choose the entry *Time* and click *Done*.

The first entry in the section *Analysis Functions* of the Online Analysis window has changed to *Time*. Later we will use this result for an X-axis variable.



Now, we need some other function to provide a variable for the Y-axis. Click on *New* to set up a new function, choose *Integral* from the Function Type dialog and click *Done*. As you might have seen there are more options for the *Integral* function available in the Function Type dialog which will be neglected now. Further information about these options can be found in the PATCHMASTER reference manual.

The new *Analysis Function* is now called "Int_1". The "1" means that the integral of *Trace 1* is calculated (see *Trace #*).



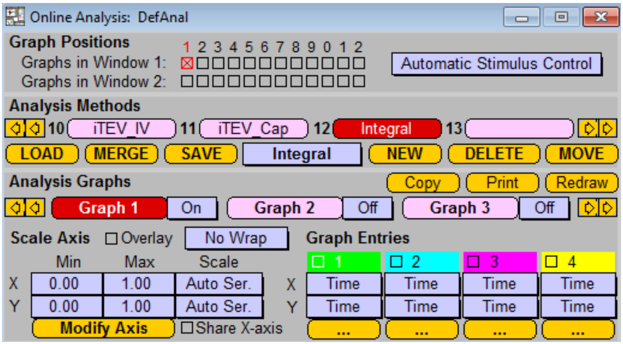
We also want both results to be copied into the Notebook, so check the *Notebook* option for both.

Remember that all analyses will be performed on the *Relevant Segment* of the sequence as it is set in the Pulse Generator window (see `Defining the Segments for Online Analysis`, 1.3.3 on page 20).

Also note that the order in which the data is displayed in the Notebook window later on depends on the order of the functions in the *Analysis Function* section. This means, if you prefer a certain order, you have to select them accordingly at the very start or use the *Move* function.

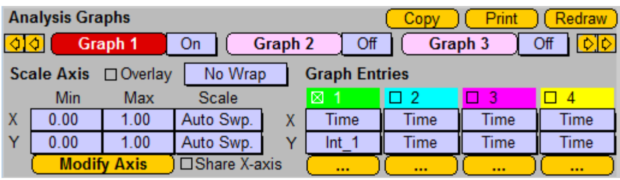
1.6.4 Setting up the Analysis Graph

To set up the graph, you first have to define in the *Graph Positions* section in which Online Window (1 or 2) the graph shall be displayed. In general it is possible to activate up to 16 *Analysis Graphs*. Please activate the checkbox "1" in *Graphs in Window 1* of the *Graph Positions* section. Further, we disable all *Analysis Graphs* except *Graph 1*.

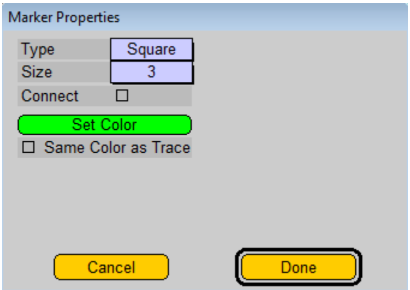


Then we have to define the *Graph Entries* for *Graph 1*. Up to four *Graph Entries* can be in one graph, but we need only one entry here.

To define the entry, check the first entry (light green in our example) and then choose the X- and the Y- axis. The scaling of the axes of the graphs displayed later on in the Online Window can be set in the *Scale Axis* section.



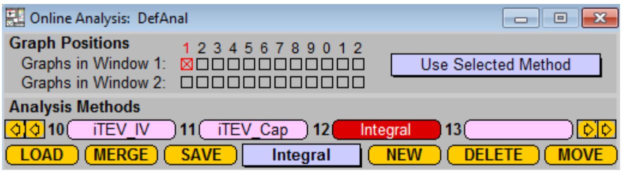
The color of the graph field will be the display color in the future graph. Light green is not very handy – let us change this to dark blue. For this, click on ... to open the following window:



Change the color by clicking on the *Color* button and choosing from the possible colors. Click *Done* to save your selection.

1.6.5 Performing an Online Analysis

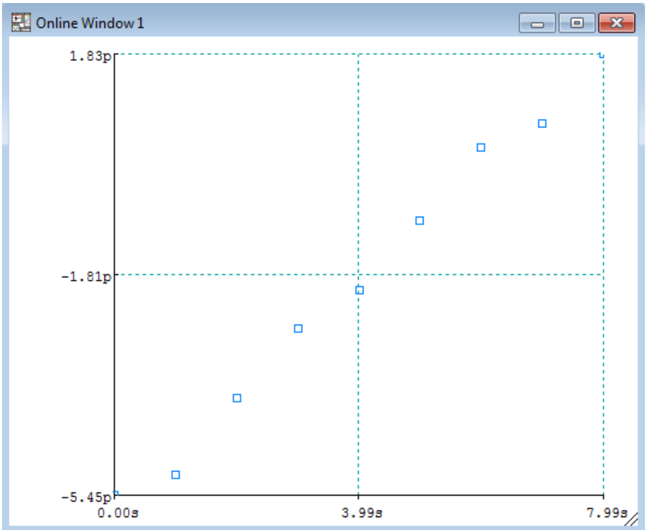
1. Make sure that the new method "Integral" is highlighted. It is also recommended to set *Automatic Stimulus Control* to *Use Selected Method* when a specific analysis of the data should be performed.



2. Open the Online Window 1 by selecting `Windows` → `Online Window 1`.

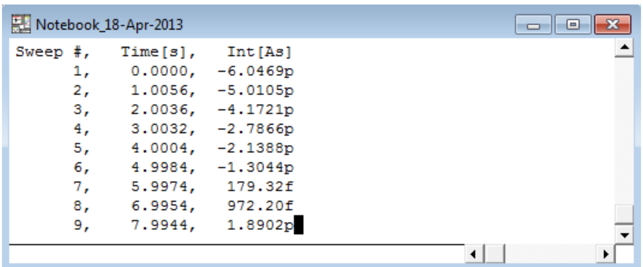
3. Acquire data or replay data by double-clicking on the *Series* in the Replay window. The analysis results will be displayed in the Online Window 1 and in the Notebook window.

The result in the Online Window should look like the following:



It is of course possible to refine the displaying of the data in the Online window. Therefore, select the *Modify Axis* option.

If you bring the Notebook window to the front you should see something like this:


Notebook_18-Apr-2013

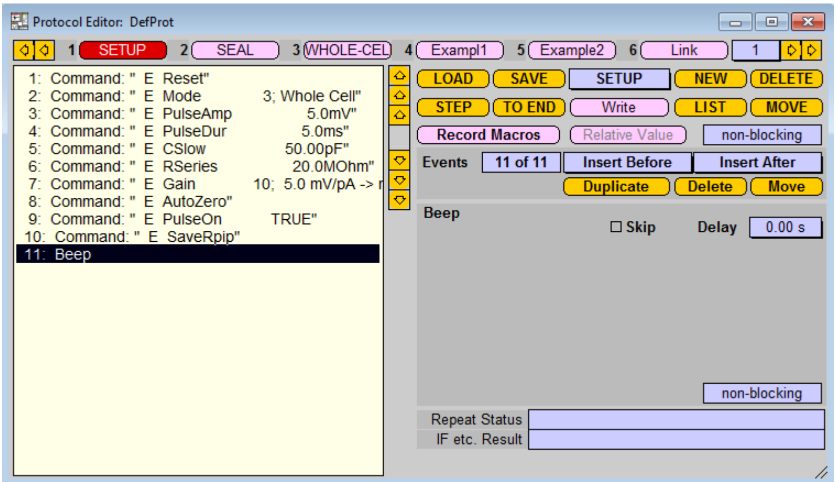
Sweep #,	Time[s],	Int[As]
1,	0.0000,	-6.0469p
2,	1.0056,	-5.0105p
3,	2.0036,	-4.1721p
4,	3.0032,	-2.7866p
5,	4.0004,	-2.1388p
6,	4.9984,	-1.3044p
7,	5.9974,	179.32f
8,	6.9954,	972.20f
9,	7.9944,	1.8902p

In case not all data are listed here, check the *Notebook* checkbox for each function!

1.7 Automating the Data Acquisition

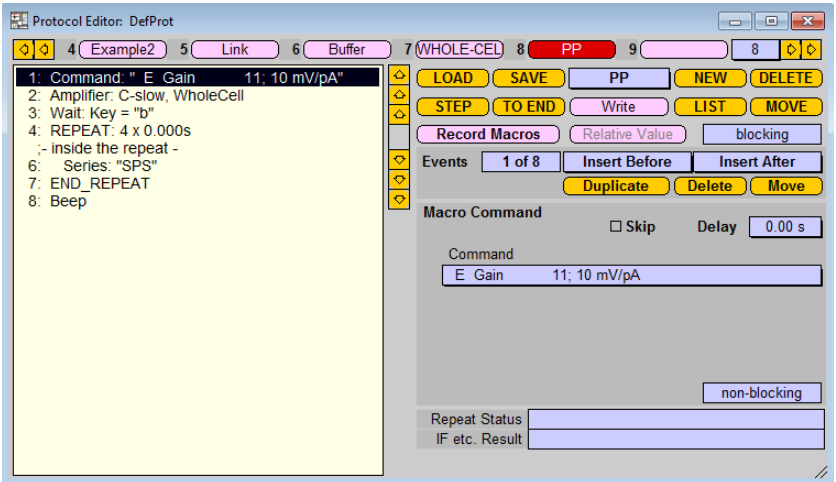
1.7.1 The Protocol Editor

As you have seen in the beginning the protocols are stored and edited in the Protocol Editor (e.g. "SETUP", "SEAL", "WHOLE-CELL"). The Protocol Editor can assemble complex experimental arrangements by combining PGF-templates with other operations (e.g. breaks, IF-THEN loops, setting changes). This window is the heart of the PATCHMASTER software concerning the automation of experiments.



Note that in the Protocol Editor window there are two different kinds of pools: A protocol pool and an event pool. The sequence pool on top is the protocol pool. Here, you can find all protocols that have been set up until now. The event pool can be accessed via *Insert Before* or *Insert After* buttons.

The protocol ("PP") that we use in this tutorial looks like the following:



As you can see, each entry (event) has its own index number. You can use these numbers if you want to move an entry to another position.

After the index number, the event name is displayed, e.g., *Amplifier*. When you click on an event, the corresponding input fields will be opened on the right.

1. This is a so-called *Macro Command*. This command sets the *Gain* value in the Amplifier window, in our case to 10 mV/pA. Although there are some amplifier settings that can be set in the Protocol Editor (see below), others have to be set via a *Macro Command*. You can find out these commands by recording a macro (use *Record Macros*) and then analyzing the macro content [`Macro Command` → `"E Gain 11"`].

2. Nothing more is done than to set the *Recording Mode* to *Whole Cell* and to mark *Auto C-slow* correction [`Amplifier event` → `Recording Mode: Whole Cell; Auto C-slow`].

3. This is followed by a *Wait* event. This event is useful if you want to be alerted during the protocol execution, perhaps because you want to change some external settings before the actual data acquisition

takes place. Only when you press the key B, the protocol execution will proceed [`Wait event` → `Wait type: Key "b"`].

4. Up to now, these settings could have been set manually by the user. However, the following event *Repeat* is the start of a loop, in this case with *Repeat Counts* "4". This way, the loop will be repeated four times [`REPEAT event` → `Repeat Counts: 4`].

5. "– inside the repeat –" is a text entered in the so-called *Annotation* event [`Annotation event` → `Annotation: - inside the repeat -`].

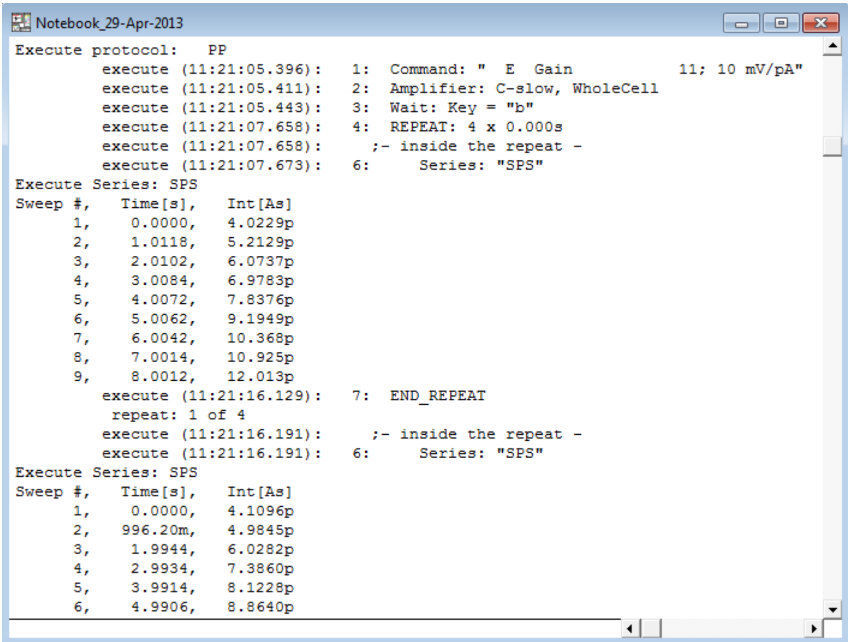
6. *Series* shows that the *Acquire Series* event is called, in this case our PGF Sequence "SPS". This starts the data acquisition. Note that you can directly open the PGF template from the event menu to edit this sequence [`Acquire Series event` → `Sequence: SPS`].

7. The next event is *END_REPEAT*, which marks the end of the loop. This event is automatically inserted when you insert a *REPEAT* event.

8. When the loop has finished, the *Beep* event is called [`Beep event`].

To start the "PP" protocol, click on the appropriate button in the *Protocol* row of the Control window.

Since in our example the option *Write* is activated, the respective event will be written into the Notebook; the latter should read like the following:



Just as in normal data acquisitions, the data will be automatically analyzed by the activated *Analysis Method*. But be aware that as long as the PGF is executed, you cannot change entries in the *Online Analysis* window. If you try it anyway, the message *"This online function is not allowed while acquiring"* will be displayed in the Notebook window.

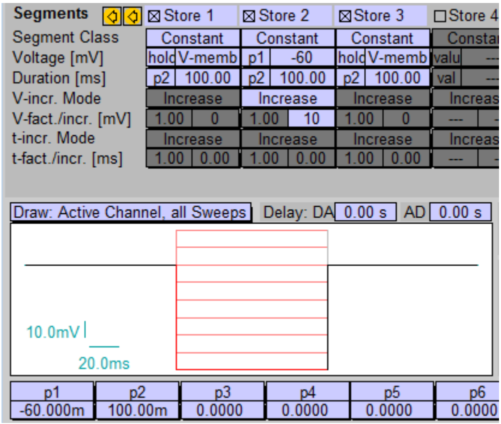
The same holds for the Protocol Editor itself – if you click on an entry in the event list during execution, the message *"Cannot run: protocol is already executing"* will be displayed in the Notebook window.

In this case, you have to click *Stop* or *Break* in the Control window to halt the protocol execution. Then you are able to modify parameters again.

1.7.2 Changing PGF Parameters via a Protocol

You also have the possibility to manipulate the *PGF Parameters* of the Pulse Generator window in a protocol.

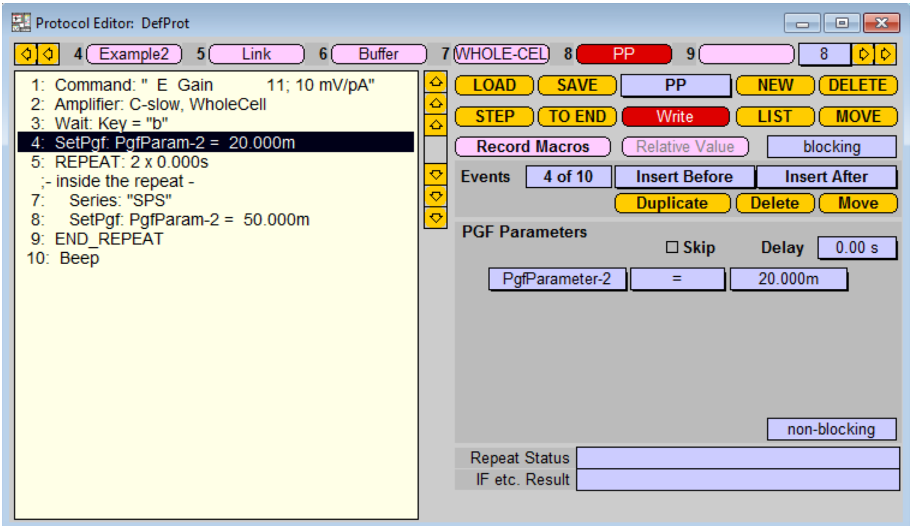
Remember: We used the *PGF parameters* "p1" and "p2" in our segment definitions.



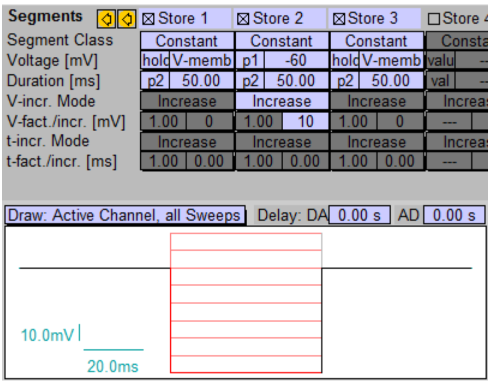
With the event *PGF Parameters* you can set a value for the parameter. Do as follows:

1. Mark line "3" in your event list and *Insert After* the event *PGF Parameters*. Set "p2" to "20m".
2. Then mark the new line "7" and *Insert After* the event *PGF Parameters* again. Here, set "p2" to "50m".
3. Decrease the number of repeats from "4" to "2".

This way, you will get a first execution of "SPP" with a lower "p2", and a second execution with a higher "p2".



When the protocol has been executed, you can see the resulting PGF values also in the Pulse Generator window.



> ***Note:*** *When you execute a Series the next time, the value 50.000m will be used as start value for "p2"!*

1.8 Customizing the Front-End

1.8.1 Customizing the Keys

In PATCHMASTER, all key commands are saved in the file `PatchMaster.key` and will be read at program start. In case the file `PatchMaster.key` is not available at program start, no key commands are available!

Please take also in consideration that you can customize all commands, so the settings in your working version of PATCHMASTER might differ from these default settings.

To display the key assignments in the various windows, choose `Help` → `Show Keys`.

To list the keys in the Notebook, choose `Help` → `List Keys`.

To save the keys, choose `Help` → `Save Keys`.

The keys are saved in the file `PatchMaster.key`. Old keyboard assignments will be automatically saved with an incrementing extension, e.g., `*.k00, *.k01, *.k02`....

You can freely customize the key commands by

- editing the keys via the dialog control and saving them or by
- directly modifying the key file, e.g., in a text editor.

1.8.2 Customizing the Windows

To modify dialog and control items in the PATCHMASTER user interface, you have to select `Enable Icon Configuration` from the `Windows` menu and then press certain controls, depending on your intended action and your operating system (MS Windows, Mac OS). For further information, please refer to `Modifying Dialogs and Controls` in the PATCHMASTER reference manual.

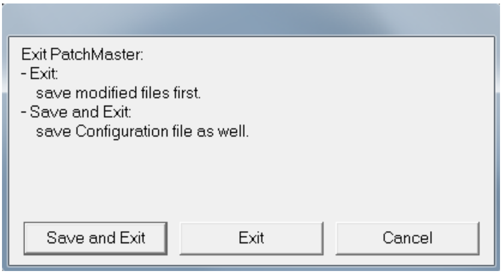
1.9 Closing PATCHMASTER

To exit from PATCHMASTER, do this:

Choose *Quit* from the drop-down menu `File` or press CTRL + Q.

Press CMD + Q.

The following window will appear:

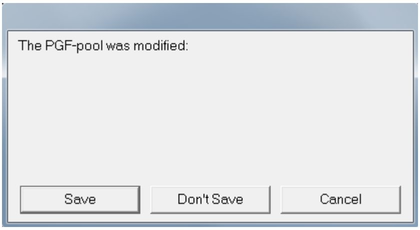


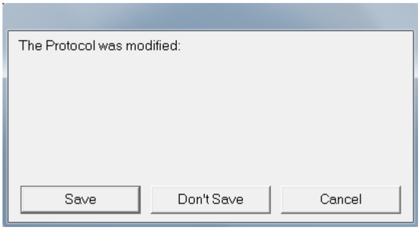
You have three possibilities:

- Save + Exit: Saves data files and configuration and quits the program. At least the first few times of running PATCHMASTER, 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, which you don't want to modify anymore, you can safely ignore this question.
- Exit: Saves data files and quits the program.
- Cancel: Aborts the exit process, you return to the program. This is the right button if you accidentally pressed the shortcut combination for exiting.

As you can see, data files will always be saved. If you lose data files, you might verify if you checked the option *Store* in the Pulse Generator or the option *Store* in the Control window during your experiment.

If you changed the pools in the Protocol Editor or the Pulse Generator, you will be asked independently if you want to save them.





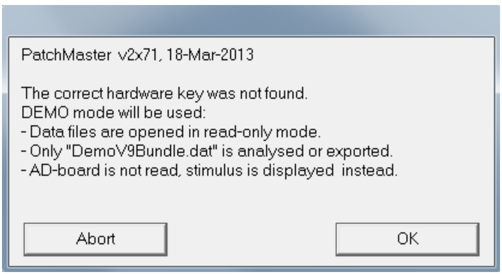
The default is *Save* for each, so just press RETURN twice to save them and exit the program.

1.10 Using the Software without Connected AD/DA Hardware

In general you have two possibilities for using the software without connected amplifier hardware: the *Demo* mode and the *Stand-alone* mode.

1.10.1 Demo mode

When no dongle is connected to the computer, the software will start in the true Demo mode. In this case, you will see the following warning:



Be aware of the messages written in the dialog window:

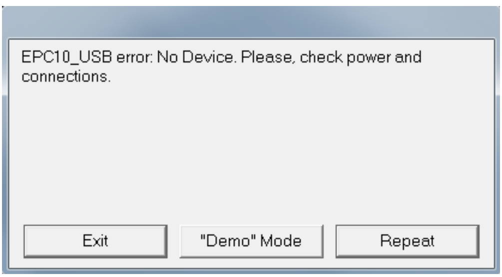
- Data files are opened in read-only mode!
- Only data from the demo data file `DemoV9Bundle.dat` can be analyzed or exported!
- AD-board is not initialized and the stimulus is displayed instead!

The *Demo* mode can be used to inspect data on an extra computer that is not connected to the setup and to evaluate the software package. In this mode, only the demo data file `DemoV9Bundle.dat` can be analyzed and exported.

Note that any output that is created is taken as input; i.e., if a *Stim.Scaling* of 0.1 is selected in the Configuration window. The system now behaves as if an amplifier is connected with a pipette having a resistance of 10 MΩ.

1.10.2 Stand-alone mode

When a dongle but no hardware is connected to the computer PATCHMASTER will come up with the message:



Click on *Demo* mode to start the software in the *Stand-alone* mode.

The *Stand-alone* mode can be used to inspect and analyze data on an extra computer that is not connected to the setup. In this mode, data can be edited and saved to disk.

Any output that is created is taken as input; i.e., if a *Stim.Scaling* of 0.1 is selected in the Configuration. The system now behaves as if an amplifier is connected with a pipette having a resistance of 10 MΩ.

> ***Note:*** *Unlike the Demo mode the Stand-alone mode needs a valid dongle.*

2. PATCHMASTER for PULSE Users

This section is intended for those users familiar with the PULSE acquisition software. We will quickly summarize the most relevant differences between PULSE and PATCHMASTER. For a detailed description of all functions, please refer to the relevant chapters of the PATCHMASTER reference manual.

2.1 General

Although in some respects PATCHMASTER looks similar to PULSE, it is a completely new program; following an improved strategy for process handling and programming. Therefore, PULSE users may miss some sometimes typical PULSE behavior of the program until it becomes clear what the benefits of the new features are. Obviously, a substantially increased realm of functions and flexibility comes at some price. In most cases, this price is to set some definitions before using PATCHMASTER in order to customize the program according to the individual needs. Thus, this tutorial tries to explain why some things are different although the old way worked very well.

There are three major changes that have several consequences:

- Support of multiple output and input channels.
- Partial support of parallel task processing, e.g. type text while performing acquisition.
- Removal of implicit functions and full capabilities for task automation.

The increased number of channels requires specifying which channels are used, displayed, analyzed etc. If only one or two channels are to be used (like in PULSE), not too many things have to be adjusted.

PATCHMASTER tries to provide at least partial support of parallel task processing. This is useful if one wants to edit some text or change windows while the program is doing something else.

Since PATCHMASTER is a program with full automation capability, all functions have to be capable of being called by the Protocol Editor. Therefore, all implicit functions and key assignments (as partially used in PULSE) had to be removed. Key assignments are now only done via direct links to buttons in windows and entries in drop-down menus. All these assignments are stored in `PatchMaster.key`. With the default settings of this file, as supplied with the software, most keys behave like they did in PULSE. Some examples:

Pipette pressure In PULSE there were implicit key assignments for setting pipette pressure (e.g., S=suction, P=positive pressure, etc.). They have been removed in PATCHMASTER. To achieve a similar behavior in PATCHMASTER the user can create protocols increasing or decreasing the voltage of the DA channel connected to the pipette pressure controller. The execution of the protocol (corresponding protocol buttons) can be assigned to keys.

Here an example: Let's assume that the pipette pressure controller is connected to DA-2. Open the Protocol Editor (F9) and create a new protocol. We also open the I/O Control window to monitor the DA-2 voltage (`Windows menu` → `I/O Control`). Now, there are two possible ways to define a protocol (button) for increasing your pipette pressure:

1. Setting *Events* by macro recording:
   - Start a macro recording by pressing the "Recording macro" button in the Protocol Editor.
   - Press the button "Relative Value" in the Protocol Editor.
   - Enter "0.1" in the *DA-Channel* field of *DA-2* in the I/O Control window.

2. Setting *Events* manually:
   - Use the *Insert After* button to insert the "Set Value" event in the Protocol Editor.
   - There, set "Value-1" to "0.1" V and select *add to item "I Dac2"*.

To assign the new protocol (button) to a key you have to select `Enable Icon Configuration` in the `Windows` menu. After that you press CTRL+left-click on the protocol button to open the Icon Configuration dialog. Select the field *Key* and press a character on your keyboard (e.g. "p"). Each time you press now either the protocol button of our new protocol or P the "Pressure" is increased. The same can be done for the "Sunction" where we subtract the same voltage from DA-2.

The new protocols and key assignment have to be saved in your default Protocol Editor file and key file. Before selecting a new key please check for double key assignments.

Store pipette resistance In PULSE the value of *SealResistance* could be stored into the variable *PipetteResistance* by typing W. Now this function is accomplished by a new button in the Amplifier Window (*R-memb* → *R-pip*). The key W can now be assigned to this field in order to obtain the same behavior as in PULSE (this is indeed done in the default `Patchmaster.key` settings).

2.2 Major Changes

Patchmaster is a multi-channel acquisition program. Pulse could stimulate 1 DA channel and could take data via 2 AD channels. In order to support trigger pulses, up to three separate trigger channels could be defined. These trigger channels, however, could only be used for very simple pulse paradigms. Patchmaster supports a (theoretically) unlimited number of input and output channels. Currently 16 input and output channels are supported by Patchmaster. This number, however, is limited by the hardware used. Given an Epc 10, there are 4 DA channels (1 stim-out and 3 free DA channels on the front panel) plus 16 digital output channels (three are available at the front panel). The number of input channels corresponds to the number of available AD channels, i.e. in most cases 8, plus the number of digital inputs (16 digital inputs).

This extension of the number of channels required a substantial redesign of pulse generation and data acquisition. As a result, independent pulse patterns can be output via the selected DA channels. Since there are no separate trigger channels anymore, short pulses to be used as trigger signals have to be designed with a regular output channel. In order to simplify synchronization of parallel output channels and for subsequent analysis purposes it can be helpful to generate such output patterns with a fixed pulse segment paradigm. For this purpose, the PulseGenerator of Patchmaster offers the feature *Common Timing*. When this option is selected, the durations only of the segments in the first stimulation channel can be altered – the corresponding segments of all other channels will be treated in the same way.

For later analysis, AD channels have to be logically linked to DA channels. For this purpose, for each input channel such a link has to be specified.

Due to the increased number of DA and AD channels, at several places in the program the channels of interest have to be selected explicitly (e.g., in the Display and Online Analysis).

Input channels do not need to have identical *Sampling Intervals*. During data acquisition, all input channels are read with an identical *Sampling Interval*. For storage of the data, however, these input channels can be compressed. A compression factor and mode have to be specified for each channel. In addition, using the feature of *Virtual Traces*, derivative input channels can be generated from other channels.

Zeroline subtraction can be performed on any segment. Unlike in PULSE, zeroline subtraction can be performed based on any segment to be specified for each input channel.

Leak handling. Leak pulses can be generated for individual output channels. Turn *Leak* on, and specify leak parameters as usual. In the linked AD channel, *Leak* also has to be turned on to tell the program to acquire leak signals. In addition, it has to be specified whether and how these leak pulses are to be stored (no storage, store average – like in PULSE, or store all individual leak responses).

Start segment. Since there are no separate trigger channels anymore, PATCHMASTER does not need the implicit assumption that data storage starts after the first trigger. Instead, data storage starts at the specified *StartSegment* and *StartTime*.

What's new in Segments? The segments are arranged a little differently to PULSE. Major changes are that values for *Voltage* and *Duration* can be replaced by global variables (p1, ..., p10). In addition, the *Logarithmic Increment* mode can be specified for each incrementing variable separately.

There are no *Conditioning* segments anymore. Instead, for each segment it can be specified whether or not it is to be stored (see `Non-stored segments in the Pulse Generator`, 5 on page 73). *Store* = "off" largely behaves like the previous *Conditioning Segment*. The implementation and details, however, are different. Like in PULSE, for *Store* = "off" segments no P/n leak pulses are generated. Unlike in PULSE, the *Store* = "off" segments are always explicitly output and sampled. Only after sampling the corresponding data are removed from the *Traces*. The big advantage is that the durations of the *Store* = "off" segments are now precise like all other segments. In addition, non-constant segments can be used (e.g. conditioning stimulation with a sine wave). For long *Conditioning* segments huge data arrays may be required. Therefore, the user has to make sure to set the maximal number of sample points accordingly (in Configuration window, just limited by the amount of available RAM).

Global parameters. As already mentioned, in addition to *Holding* and specific *Values*, *Amplitude* and *Duration* of segments can be filled in with global parameters (p1, . . . , p10). At the time of execution, these parameters are filled into the pulse patterns. This makes it much easier changing many segments at once when these have identical parameters (e.g. a train of pulses to a given *Voltage* = p1). For further reference please refer to `Global Variables in PATCHMASTER`, 4 on page 69.

Sequence Timing. PULSE users may wonder where they could find the sequence timing parameters *Linked Sequence* and *Repeat*. The sequence timing has been removed from the Pulse Generator in PATCHMASTER. Instead, a much more flexible sequence timing is now provided within the Protocol Editor.

Amplifier adjustments, updates. Like sequence timing, amplifier adjustments such as updates of *C-fast*, *C-slow*, *G-series* etc., have been removed from the Pulse Generator and now have to be called from the Protocol Editor.

Protocol Editor. The Protocol Editor is completely new. It allows for a versatile definition of complex protocols (see PATCHMASTER reference manual).

Display. The display has not changed very much. Since there are now many *Traces* to be shown simultaneously, scaling parameters, colors etc., have to be specified for each *Trace* separately. Several features were previously supported as buttons in the `Oscilloscope` window. These functions have now been moved to the `Display` drop-down menu (e.g. `Subtract Zero Offset`). Key assignments to the `Display` menu entries, e.g. `Show Leak Traces`, have to be defined individually by the user.

Pipette pressure. Pipette pressure is not supported as an implicit output channel anymore. One now has to use the I/O Control Window for setting the corresponding pressure.

Online Analysis. The Online Analysis functions have been redesigned completely. As a result, the analysis has become much more flexible and powerful. The immediate consequence for PULSE users is that there is no default analysis anymore. Thus, without definition of online functions (see PATCHMASTER reference manual), there will be no Online Analysis. The major improvements with respect to PULSE are:

- Arbitrary number of analyses and a much greater set of analysis functions.
- Generation of derivative data.
- Analysis results do not have to be shown in the Notebook window.
- Analysis results can be plotted in multiple graphs placed in up to two windows (Online Window 1 & 2).
- An unlimited number of analysis protocols can be stored.
- Online Analysis can be directly triggered by the incoming data (Analysis method can be specified in the Pulse Generator).

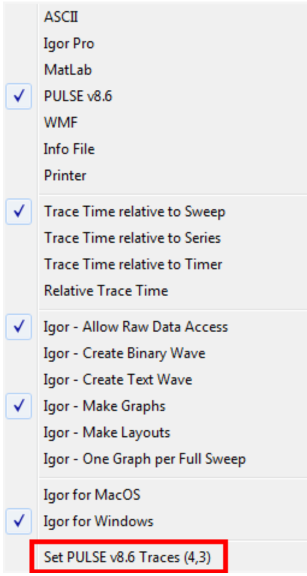
3. Export of Patchmaster Data in PULSE 8.6 Format

The PATCHMASTER data format has been significantly extended compared to the PULSE data format. Therefore, conversion of PATCHMASTER data into PULSE data format only works for data that have been acquired following the more restrictive conventions of PULSE.

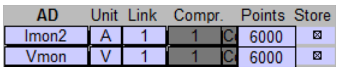
3.1 Export Rules

If you plan to export data into *PULSE v8.6* format, e.g., in order to analyze the data with PULSEFIT, PULSETOOLS or PULSESIM, please make sure that the acquisition of PATCHMASTER data follows the rules below:

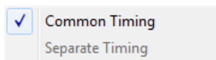
Number of Traces: The maximal number of *Traces* which can be exported at once can not exceed "2". Which *Traces* are exported as 1^{st} and 2^{nd} *Trace* can be specified in the dialog "SetPULSE v8.6 Traces (1,2)" which can be selected from the `Export Format` drop down menu list.



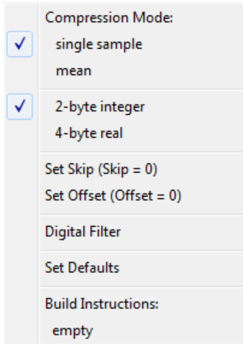
Compression Factor: The 1^{st} *Trace* and 2^{nd} *Trace* must have the same number of data points. Do not choose different data compression factors for the first and second *Trace* in PATCHMASTER.



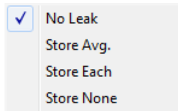
Common Timing: The number of segments must be the same for both *Traces*. Do not activate *Separate Timing* in the segments settings.



Data Format: The data should be stored as 16-bit integers. In PATCHMASTER you can choose the data format. Therefore, make sure that *2-byte integer* is selected in the compression section of the channel settings.



Leak Traces: If individual leak *Traces* are stored in PATCHMASTER, only the averaged leak *Trace* will be exported together with the data *Trace*. Individual leak *Traces* can be exported separately if selected as *Trace* for export directly in PATCHMASTER.



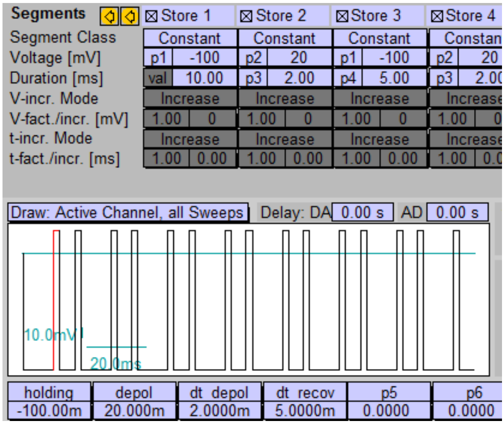
4. Global Variables in PATCHMASTER

There are two types of global variables in PATCHMASTER that are accessible by the user – *PGF Parameters* and *Values*. We will shortly outline how these parameters can be used, which features they enable, and how they can facilitate various tasks.

4.1 PGF Parameters

These Parameters (p1, . . . , p10) are stored in a PGF pool file. They can be edited in the Pulse Generator or can be set via the Protocol Editor. These parameters are used to facilitate input for *Durations* or *Voltages* in segments of the stimulation sequence. E.g., if a stimulation sequence consists of a train of pulses to the same *Voltage*, this *Voltage* could be specified via a parameter. Editing of this parameter will then be of effect for many segments without extra editing of segments. In the Pulse Generator the names for the parameters can be specified in order to remember what they are supposed to be used for.

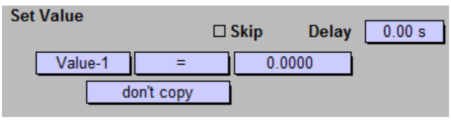
The example below shows how a train of pulses can be specified with parameters. Note that parameters to be used in the Pulse Generator have to be given in SI units, i.e. in volts and s (not mV and ms).



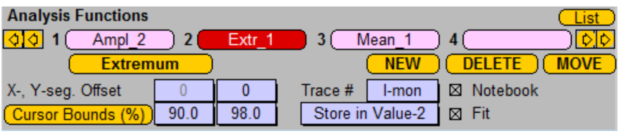
4.2 Values

Values-1 ... 16 are used for calculations in the Protocol Editor and in the Online Analysis. In addition, they are used to exchange information between these two program modules. By setting *Values* in the Online Analysis and by reading *Values* in the Protocol Editor (in a conditional *Event* "IF ... THEN"), the protocol can respond to an analysis result. In addition, *Values* can be set in the Protocol Editor.

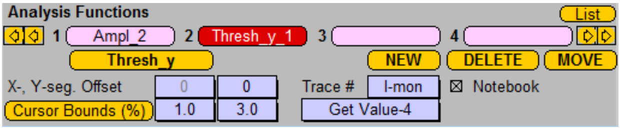
Set a *Value* in the Protocol Editor:



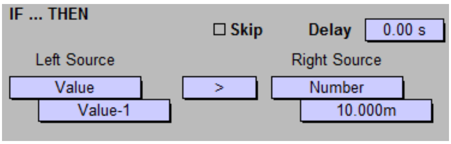
Set a *Value* in the Online Analysis (e.g. store an analysis result for later use in a calculation such as normalization or background subtraction):



Read a *Value* in the Online Analysis (e.g. to read a previously stored analysis result and use it for another calculation):



Read a *Value* in a conditional *Event* of the Protocol Editor (e.g. read a previously stored analysis result and compare it to a specified *Value* (here: 10 mV) to make a decision in an experiment such as *Break* or *Continue* with execution of a specified pulse protocol):



5. Non-stored segments in the Pulse Generator

On top of each segment column a checkbox defines whether or not this segment is going to be *stored*. The purpose of this feature is to save space when long conditioning intervals have to be introduced between important data and when the data of such *Conditioning* segments are not of interest. PATCHMASTER will output the template for such segments, as it will for all other segments (i.e. it would generate *Constant, Continuous, Ramp, Sine* or *Square* segments as specified). Data are also sampled during these periods, but they are removed from the *Traces* prior to storage. Thus, in order to have *Non-Stored* segments very long, one has to make sure PATCHMASTER has enough memory (see `Configuration Window` in the PATCHMASTER reference manual).

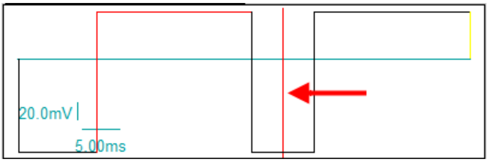
5.1 Using non-stored Segments

A *Non-Stored* segment can be a *Relevant X-segment*; it cannot be a *Relevant Y-segment* because there is no data obtained during that segment that could be analyzed later on. In addition, a *Non-Stored* segment must not be the *Start Segment*.

An Example: Here is an example for a template in which segment # 4 is not stored. This segment is a *Conditioning* segment with voltage -100 mV. Its duration in the first *Sweep* is zero, then it is incremented using the *t*Fact./Incr. Mode* (factor of 2 and an increment of 50 ms, thus yielding durations in ms of 0, 50, 100, 200, 400...). This protocol could, for example be used to measure the time course of recovery from inactivation that had occurred during the first depolarization (segment # 2).

Segments	⊠ Store 1	⊠ Store 2	⊠ Store 3	□ Store 4	⊠ Store 5	⊠ Stor
Segment Class	Constant	Constant	Constant	Constant	Constant	Constant
Voltage [mV]	valu -100	valu 50	valu -100	valu -100	valu -100	valu 50
Duration [ms]	val 10.00	val 20.00	val 4.00	val 0.00	val 4.00	val 20.00
V-incr. Mode	Increase	Increase	Increase	Increase	Increase	Increase
V-fact./incr. [mV]	1.00 0	1.00 0	1.00 0	1.00 0	1.00 0	1.00 0
t-incr. Mode	Increase	Increase	Increase	Increase	Increase	Increase
t-fact./incr. [ms]	1.00 0.00	1.00 0.00	1.00 0.00	2.00 50.00	1.00 0.00	1.00 0.00

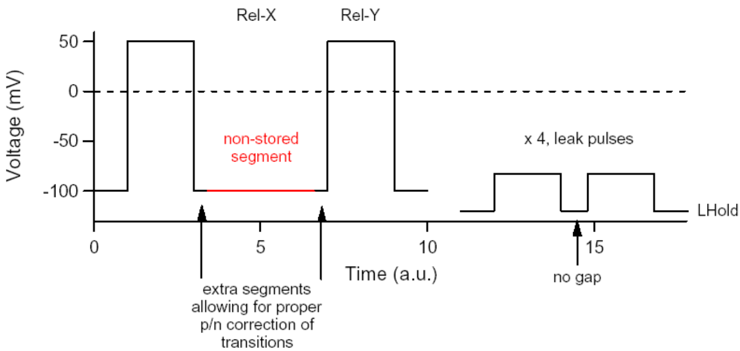
In the template cartoon *Non-Stored* segments are shown as vertical lines as illustrated below.



Advantage 1. A major advantage of *Non-Stored* segments is to save time during the compilation of leak responses for those segments that are not important for later analysis. This is because *Non-Stored* segments are **NOT** considered in the leak pulse templates. Thus, for long conditioning times leak pulses are now much shorter and allow for shorter repetition intervals.

Advantage 2. In addition, in some cases holding cells at or around the leak holding potential may be stressful. Therefore, minimizing the time used for leak pulses may increase overall stability of the recording configuration.

The immediate drawback of having *Non-Stored* segments eliminated from the leak templates is that there is no P/n correction possible at the transitions to and from such segments. Therefore, if proper P/n correction in these regions is essential, one must ”sandwich” *Non-Stored* segments by *Constant* segments of the same voltage. In the example above these are segments # 3 and # 5. The duration of these segments should be such that transients after voltage steps fully relax. For the example shown this means that the actual conditioning time increases by the duration of segment 3 and segment 5 (4 + 4 = 8 ms). Thus, during final analysis of the recovery time course, one has to add these 8 ms to the sequence of durations, yielding 8, 58, 108, 208, 408 ms,. . . .



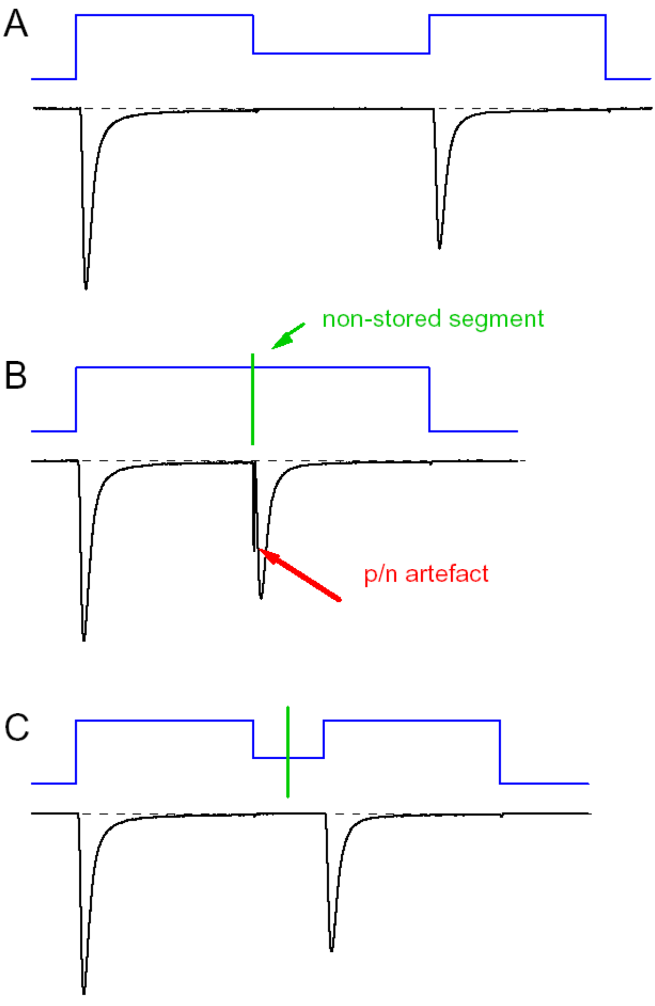
The latter is easily achieved in the Online Analysis by defining a *Constant* of 8 ms to be added to the incrementing durations of segment # 4. Alternatively, durations of segments # 3 and # 5 can be added explicitly.

The examples shown below illustrate the problem described above.

In panel A a pulse protocol of 5 segments elicits sodium currents at -20 mV to cause full inactivation. In segment # 3 at -80 mV channels are partially recovered from inactivation, assayed in segment # 4 by another depolarization.

In panel B segment # 3 is replaced by a *Non-Stored* segment (green vertical line). As a result, there are no transitions between segments 2-3 and 3-4 in the P/n pulse and, thus, the capacitive currents are not properly corrected (red arrow).

In panel C this problem is remedied by sandwiching the *Non-Stored* segment by short segments of the same voltage yielding proper P/n correction.



Advantage 3. The relevant (stored) segments are displayed in an aligned fashion in the Oscilloscope window.

6. Ramp Protocols

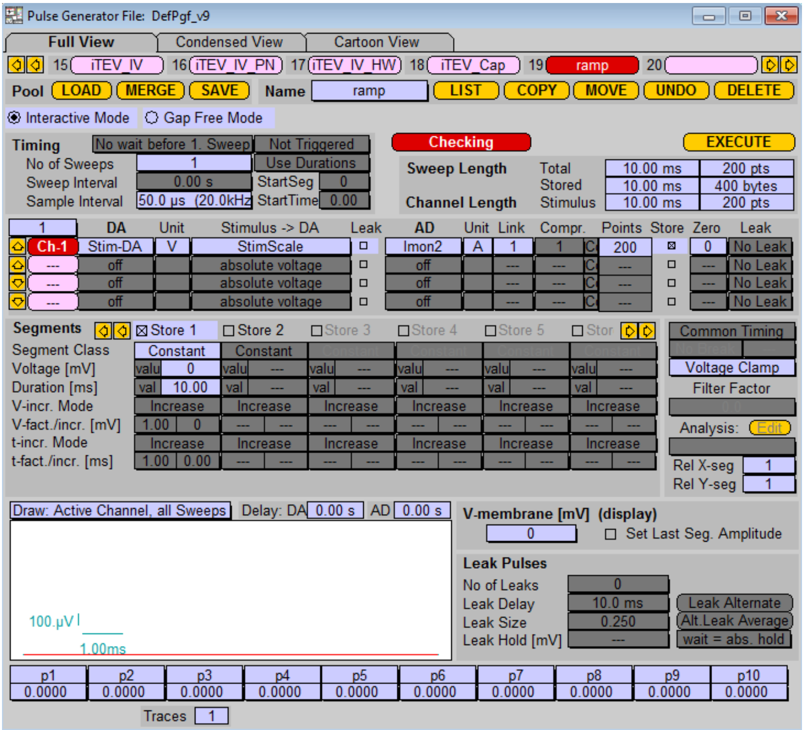
The Pulse Generator of PATCHMASTER can handle several segment types, such as *Constant*, *Continuous*, *Sine*, *Square* segments and *Ramp* segments. *Ramp* protocols are often used for a fast characterization of membrane conductances. In such a *Ramp* protocol, the holding potential is continuously changed from a given start potential to an end potential. The corresponding current is recorded and plotted over the holding potential - a typical current-voltage relationship or "IV plot".

Let us start from the very beginning and first create a template for this kind of stimulation. So, please open the Pulse Generator dialog of PATCHMASTER and create a new sequence with the name "ramp".

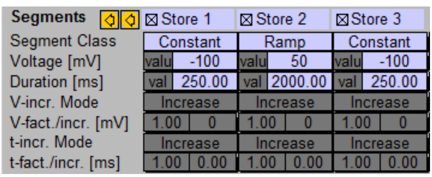
6.1 The Pulse Generator Dialog

Before we proceed, we should check some important settings: the *No of Sweeps* is set to "1". This is fine. But the *Sample Interval* is set to 50 μs. For our "ramp" PGF, a *Sample Interval* of 200 μs is sufficient. This corresponds to a sampling rate of 5 kHz.

The other settings are ok and we can now design the stimulation pattern in the *Segments* section of the dialog. One segment of type *Constant* already exists. This can be used to set the start potential for our "ramp" PGF. Let's set the *Voltage* of this segment to "-100 mV". Now we have to create two other segments, one of type *Ramp* and another *Constant* segment.

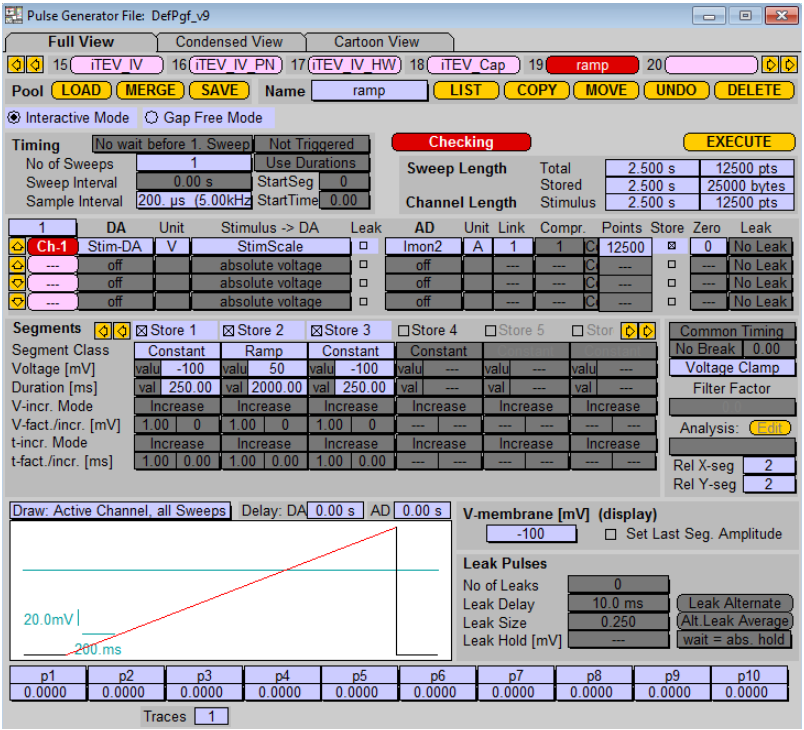


In the *Ramp* segment, we set the *Voltage* to "50 mV". This is the end potential of the *Ramp*. The last segment is used to jump back to "-100 mV". Finally, we increase the duration of the 3 segments. The duration of the *Constant* segments is set to 250 ms and the duration of the *Ramp* is set to 2000 ms.



For the analysis, it is important to let PATCHMASTER know, what segment should be analyzed by the Online Analysis. In our protocol, this is the second segment. Therefore, we set *Rel X seg* and *Rel Y seg* to "2".

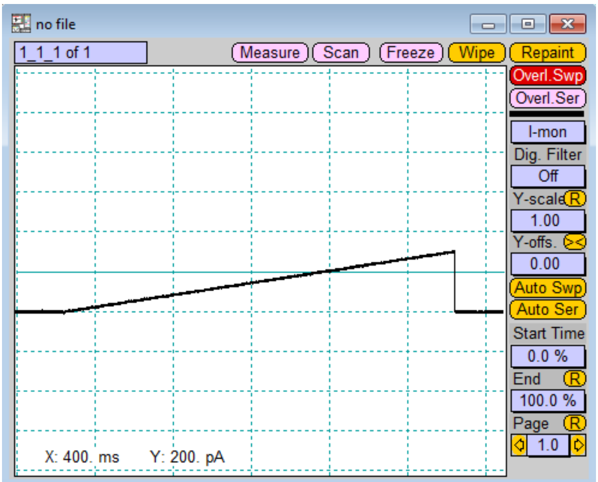
Our final PGF template will look like this:



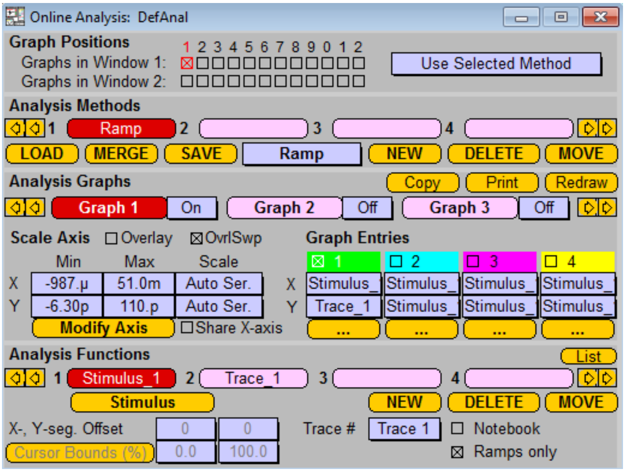
6.2 Running the Ramp Protocol

For testing our new protocol we connect the model circuit to the headstage of the EPC 10 amplifier. Switch the model circuit to the 10 MΩ position and click on "SETUP" in the amplifier dialog of PATCHMASTER. Then switch to the middle position and click on "SEAL". Finally, go to the 0.5 GΩ setting and click on "WHOLE-CELL".

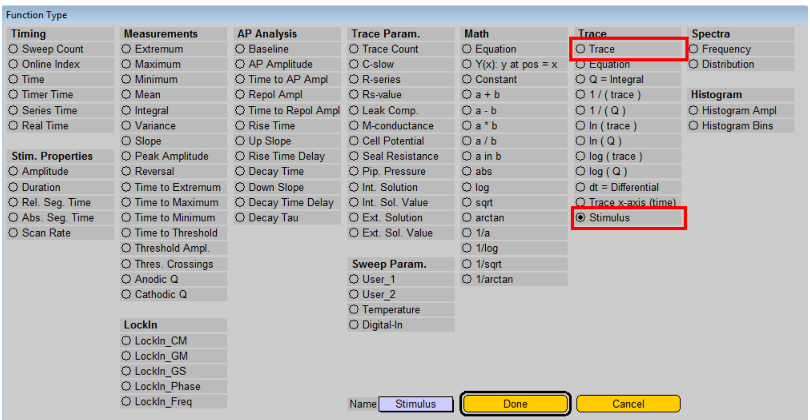
If the Control window is not yet opened, then bring it to front by selecting it from the Windows menu and run the sequence by clicking on the "ramp" button.



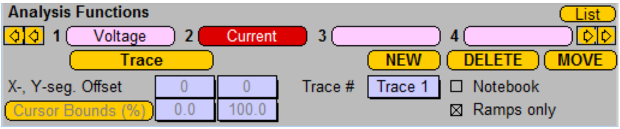
In the Oscilloscope the current response to our ramp sequence is displayed, but the current is plotted over the time, not over the ramp potential. Here, we need the assistance of the Online Analysis. Push the F7 key to bring the Online Analysis dialog to front.



Click on an empty button in the *Analysis Methods* section of the dialog in order to create a new and empty *Analysis Method* named "Ramp". The goal is to plot the current over the ramp potential. So we need two *Analysis Functions*: the current *Trace* and the stimulus information. These functions are listed in the Analysis Functions dialog, which can be opened by clicking on a button in the *Analysis Functions* section of the Online Analysis dialog.



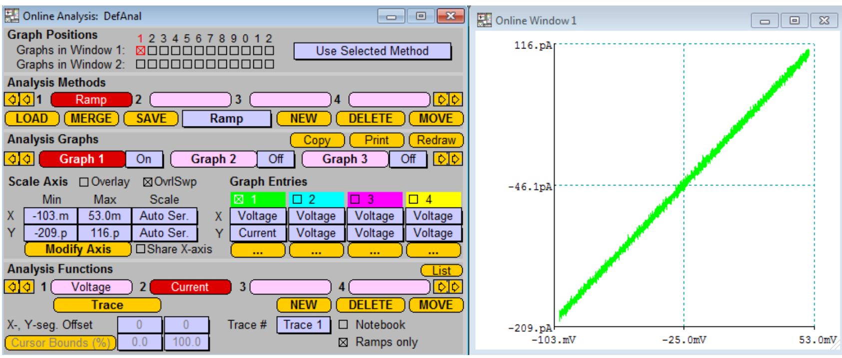
Select the *Trace* function from the list. At the bottom of the dialog, you can name this function. "Current" is a good choice for our data. Then close the dialog.



Please note, that parameters can be set for most *Analysis Functions*. For our *Trace* function, we have to specify which *Trace* should be analyzed. In our PGF sequence, we have only 1 *Trace*, consequently "Trace #" should be set to "Trace 1". It is also a good idea to check the *Ramps only* box. Later on we will see why we should do so.

The same has to be done for creating the stimulus *Analysis Function*. We select the *Stimulus* from the *Analysis Functions* list, rename it into "Voltage", close the dialog and also check *Ramps only* for this function.

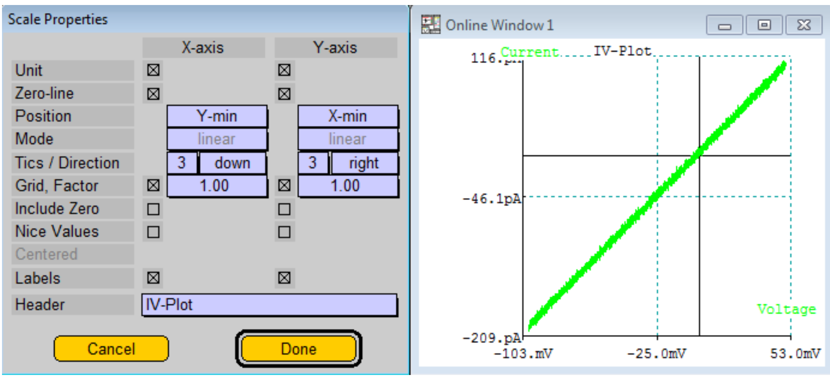
These two functions can now be used in the *Analysis Graphs* section of the Online Analysis dialog.



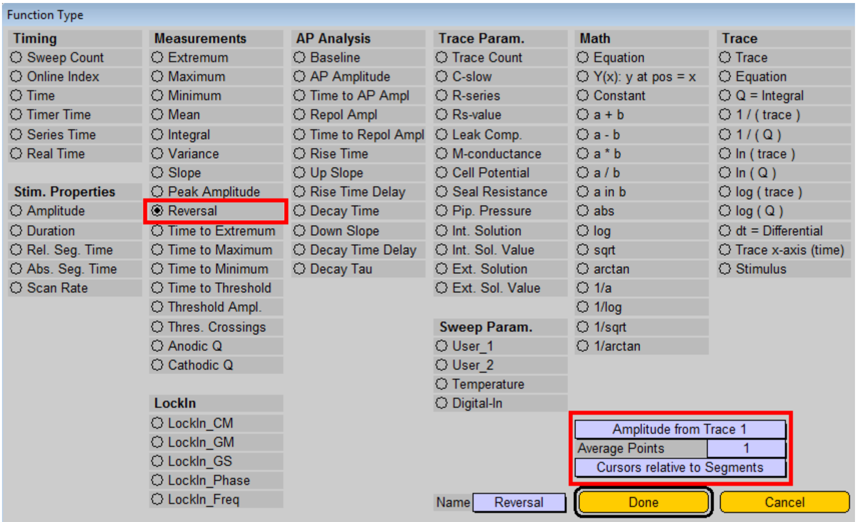
For each graph, 4 *Graph Entries* are available. We only need the first one and we set X to "Voltage" and Y to "Current". Finally, we have to decide if the graph should be plotted to Online Window 1 or to Online Window 2. Let us activate the first checkbox in *Graphs in Window 1* in the *Graph Positions* section. This enables the display of *Graph 1* in the Online Window 1. Of course, we should now open the `Online window 1`, otherwise we will not see the result of our efforts (`Windows menu` → `Online Window 1`).

The next time we execute the "ramp" sequence or replay the already recorded data the current voltage relationship of data is displayed in the Online Window 1.

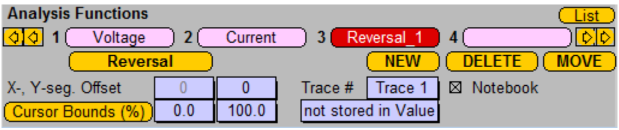
It would be quite nice to add labels to the graph. This can be done by clicking on the *Modify Axis* button in the Online Analysis dialog. In the Scale Properties dialog, we can enter a header for the graph. If *Labels* is checked, then the name of the displayed *Analysis Function* is plotted in the graph. And finally, we can check the *Zero-line* boxes, in order to make the zero lines visible in our IV-plot.



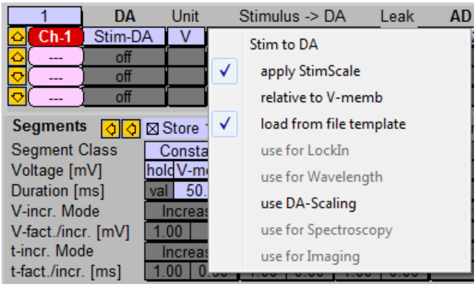
An important parameter of such an IV is the reversal potential. This reversal potential can be calculated by the *Analysis Function Reversal*. Please note, that you have to specify in the Analysis Function dialog, if the potential information should be read from another *Trace* (e.g. an recorded voltage *Trace*) or from the theoretical ramp potential.



After specifying the *Reversal* potential function in the Function Type dialog you should activate the checkbox *Notebook* to get the values of the reversal potential displayed in the Notebook.



7. Using a Recorded Waveform as Stimulus



The DAC-stimulus template output by PATCHMASTER can either be computed by the program or loaded from a file by activating *load from file template* in the *Stimulus → DA* section in the channel settings of the Pulse Generator of PATCHMASTER. This way, one can stimulate any complex pulse pattern that PATCHMASTER otherwise could not calculate. Even a prerecorded voltage *Trace* such as an action potential can be used for stimulation.

7.1 Rules

There are the following things to consider when using the *load from file template* feature:

1. Location of the template file: The template must be in a file in the folder where the `*.pgf` files are. One can also put the files into a sub-folder inside the folder where the `*.pgf` files are. In this case, the folder name must be the same as the name of the stimulus.

2. Name convention: The file names of the templates define how the templates are used. PATCHMASTER offers the following options:

   (a) One template file per DA channel should be common for all *Sweeps* of the *Series*: For this option, the name of the template would be..."[stimulus name] _[channel number].tpl". E.g., if the stimulus name is "IV", then PATCHMASTER looks for the template file "IV_1.tpl" to be used as the template file for all *Sweeps* of the first (1.) DA-channel.

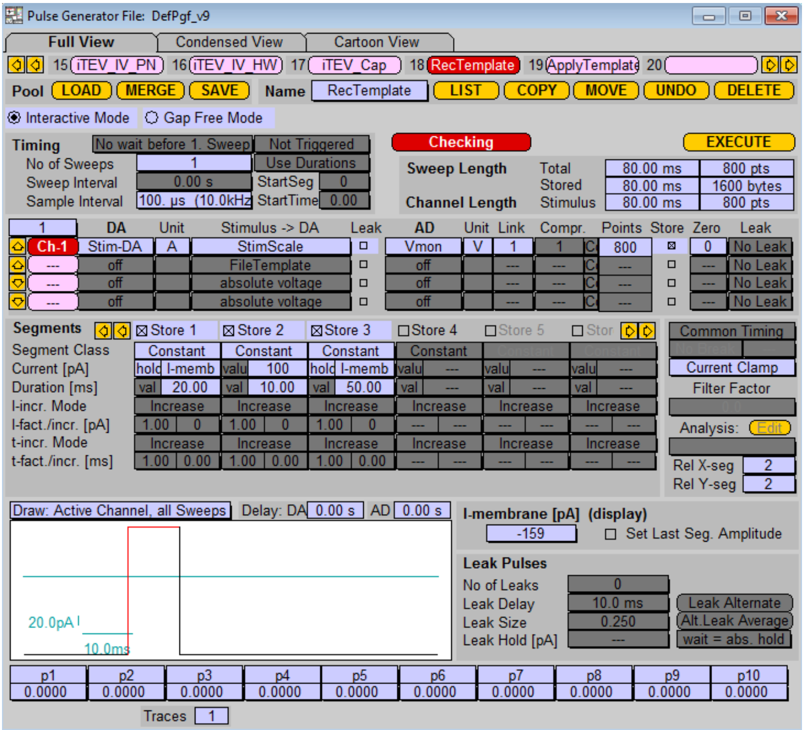
(b) Different template files per DA channel and Sweep: For this option, the name of the template would be..."[stimulus name]_[sweep index]_[channel number].tpl". E.g., if the stimulus name is "IV", then PATCHMASTER looks for the template file "IV_1_1" to be used as template for the first *Sweep* of channel one, "IV_2_1" to be used as template for the second *Sweep* of channel one etc. Please note, that the location of the template files is in the "[stimulus name]" folder inside the folder where the PGF file is originated.

3. Data format: **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".

7.2 An Example

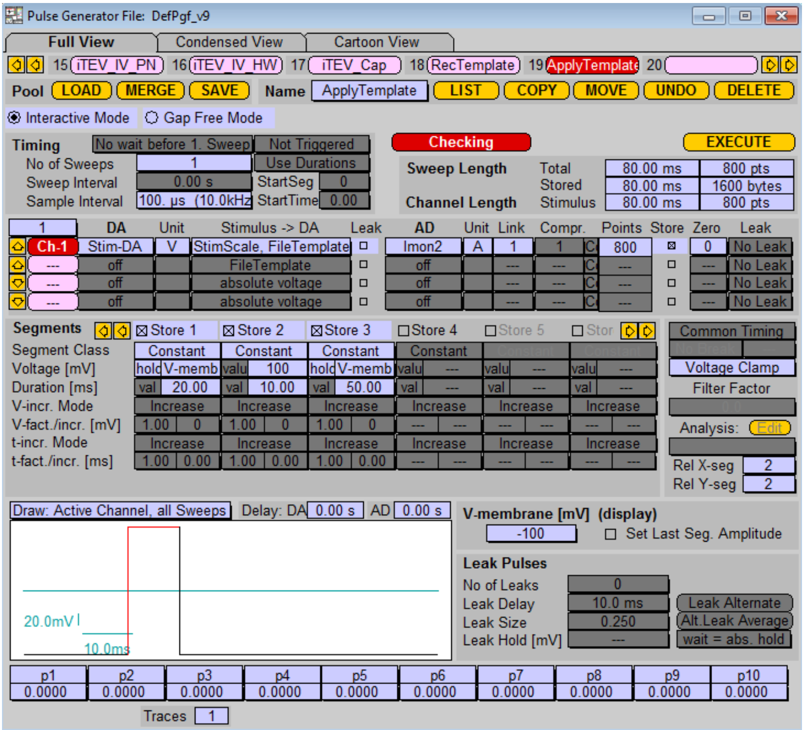
In the following we will demonstrate how the "File Template" feature is applied to stimulate with a prerecorded pulse pattern. One can easily test this procedure using the model circuit:

1. **PGF Series for recording the template:** In order to record an "Action Potential" we generate a simple Pulse Generator *Series* named "RecTemplate" with one *Sweep* per *Series* (No of Sweeps = 1) and three *Constant* segments (Duration: 20, 10, and 50 ms). Change the mode of the PGF from *Voltage Clamp* to *Current Clamp*. The first and third segment we set to holding current (*holding*) and in the second segment we inject some current into the cell (Imem, 100 pA, Imem). The *Sample Interval* should be 100 μs and one input channel has to be acquired (AD = Vmon).

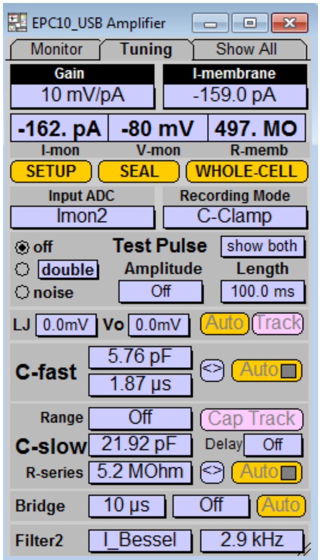


2. **PGF Series for applying the template:** The name of the PGF *Series* and the template that is used by the *Series* must have the same base name. We therefore create a *Series* with name "ApplyTemplate" by duplicating the *Series* "RecTemplate" using the

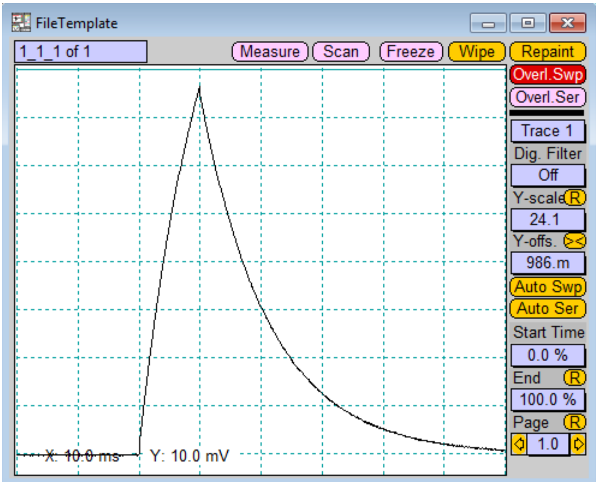
Copy function. In the *Stimulus → DA* section we select *load from file template* and adjust other parameters (e.g. *Voltage Clamp* mode, sample from the current monitor, Imon2,...).



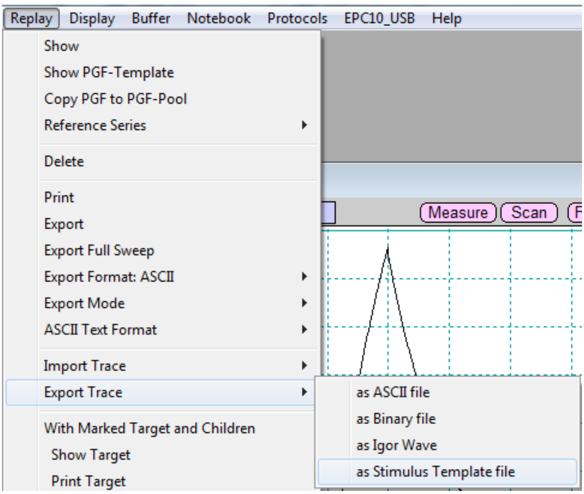
3. **Setting up the model circuit and amplifier measuring mode:** If the the model cell is used, then first establish a whole cell recording situation by putting the model cell in the 500 MΩ position and choose the appropriate amplifier settings. E.g. use a holding potential of "-80 mV" and switch to *Current Clamp* mode in the Amplifier window.



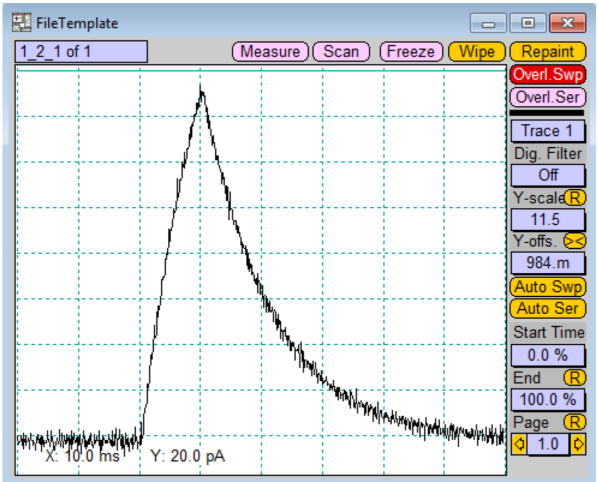
4. **Record the template:** In the Control window, execute the "RecTemplate" stimulus (press the "RecTemplate" button). The *Store* button must be on in the Oscilloscope, otherwise the *Sweep* will not be stored. The voltage response with its corresponding "Action Potential" shape should be seen. Let's assume that the response is about " 80 mV" in amplitude.



5. **Export the template file:** Select the *Trace* to be exported in the *Replay* window and select `Export Trace` → `As Stimulus Template` from the `Replay` menu. A file selector will pop up. Store the template to disk into the folder where the PATCHMASTER `*.pgf` files are located as `ApplyTemplate_1.tpl`.



6. **Apply template:** Switch the recording mode in the Amplifier window from *Current Clamp* to *Voltage Clamp* mode. Finally, execute the "ApplyTemplate" stimulus in the Control window. The file template is read and used as the template, and one should see the corresponding current response.



Alternatively, third party programs such as `Igor Pro` can be used for generating the stimulus template file.

Part II

PATCHMASTER for Advanced Users

8. Chart Recording

In this chapter we explain in detail how PATCHMASTER is configurated and used as a chart recorder. If your not interested in reading the details but are eager to start immediately with predefined settings you can download the corresponding demo configuration from our support homepage (http://www.heka.com/support/tuto.html). Inside the downloaded folder you will find the different PATCHMASTER files and a document sheet giving you instructions how to set up PATCHMASTER (for details see also 8.4 on page 102).

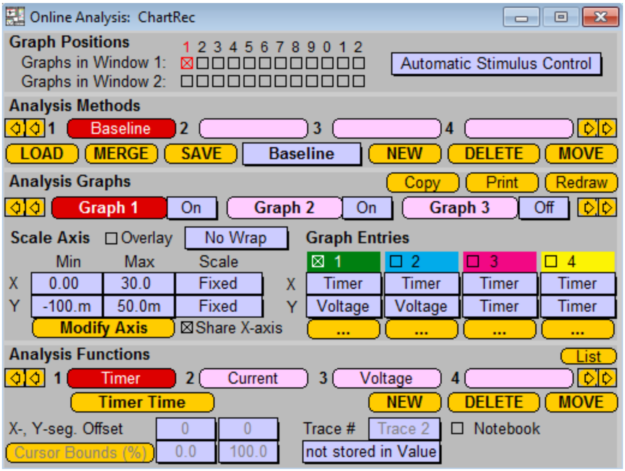
The Online Analysis and the Online Window of PATCHMASTER are the essential parts of the chart recording function. For a proper operation during the experiment the user should consider the following points:

- A fixed set of parameters has to be displayed in one Online Window.
- The display parameters of the Online Window should be kept constant in all *Analysis Methods* used during the experiment.
- The set of *Analysis Functions* must be contained in each used *Analysis Method*.

If an *Analysis Function* can not be calculated due to e.g. missing data, no entry to the online display is added.

8.1 Settings

8.1.1 Online Analysis



In the Online Analysis the following settings should be made:

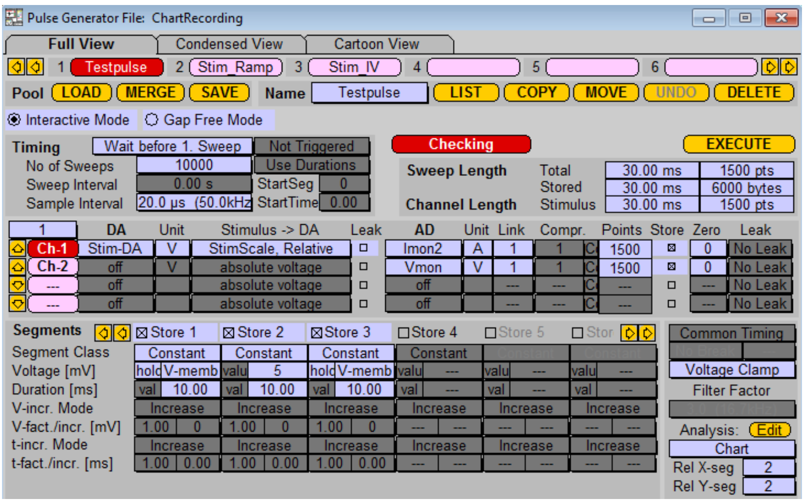
- Wipe behavior: In order to prevent a clearing of the display in the Online Window, you should check the option *Overlay* in the Online Analysis window. A *Wipe* can be performed manually by clicking on the *Wipe* button in the Oscilloscope window or automatically via a command implemented in the protocol of the Protocol Editor.

- Wrap behavior:
  - In case you want to have a display of the whole experiment chart in the *Online Window* at all times, deselect the *Wrap* option and use *Auto Scale* for the X-axis.
  - A more typical display method during chart recording is the wrapping at specified time intervals. Select the *Wrap* option and set the X-axis scaling to fixed and specify the time interval (e.g. 30 s).

- Common X-axis Scaling: Usually all graphs in a chart recording have its entries plotted against the time. In case you want to change the scaling properties of the X-axis it would be tedious to do this in each individual graph. To circumvent this you can choose the option *Share X-axis* that copies all X-axis-related scaling properties of the active graph to all other graphs of the same *Online Window*. Hence, changes in the scaling have to be done in one graph only.

- Analysis Functions: A standard set of *Analysis Functions* should be listed at the beginning of the function selector at constant index numbers. E.g. the *TimerTime* should be assigned to *Analysis Function* #1, current to #2, voltage to #3.... So that all functions that are common to all *Analysis Methods* and a subject to be used for the chart display are listed in the beginning.

8.1.2 Stimulation Sequences



When setting up the stimulation sequences the following issues should be followed:

- You should use the identical order of acquired *Traces* in all stimulation sequences that are used during one experiment in order to allow replay of data with one *Analysis Method*.
- You may use the *Trace Assignment* to assign a *Trace* count (index) to an individual *Trace*. This is a way to overrule the order in the AD channel section. The *Trace* count can be assigned in the Configuration window (*Trace Assign* tab).
- Specify the *Analysis Method*, if different methods are used for different stimulation sequences.

8.2 Replay of the Whole Experiment

When replaying the whole experiment you have to address the following points:

- Activate the *Share X-axis* option and set the following:
  - Disable the *Wrap* option.
  - Set the X-axis scaling to *Auto after each Sweep*.
- In order to disable the *Automatic Stimulus Control* for the Online Analysis you must select *Use Selected Method* as online mode (this can be done automatically at the end of the experiment from within the protocol).
- Replay the whole experiment (automatic replay at the end of the protocol can be implemented).

8.3 Experiment Control via a Protocol

The whole experimental control can be implemented in a protocol. The protocol can be used e.g.

- to configure the program for the individual display and analysis needs.
- to react to user input during the experiment.
- to react on analysis results or signals coming from peripheral devices.
- to automate tasks at the end of the experiment such as updating the file or replaying the whole experiment.

  Note: *Another advantage of the protocol control is that you can switch between different stimulation sequences during the experiment without wiping the display of online data which have been recorded before.*

Now we describe parts of the predefined "Chart" protocol which comes along the demo configuration package (`ChartRecording.pro`).

8.3.1 Prefix

Main Settings: First, we wipe all data in the Online Windows and the Oscilloscope. After that we create a new *Group* which will help us to organize the acquired data and facilitates the replaying of all data at the end of the protocol. Since we do not want to wipe all the display of the Online Analysis when a new pulse sequence is started, we set *Wipe=OFF* in the *Acquire* event. Next, we reset the *Timer* time and then we set the Online Analysis behavior to *Automatic Stimulus Control*.

```
Command          ( 0.000s)  : "  O  Wipe"
File             ( 0.000s)  : NewGroup
Acquire          ( 0.000s)  : Wipe=OFF,
SetOsci          ( 0.000s)  : Timer,
Online           ( 0.000s)  : Auto
```

8.3.2 Main Loop

GOTO_MARK: At the beginning of the main loop we define a *GOTO_MARK* as reference point for returning back into the main loop.

```
GOTO_MARK        ( 0.000s)  : "Chart"
```

Repeat Each Sweep: With the *Acquire Each Sweep* event the main loop is defined. In a defined interval (e.g. every 1 second) one *Sweep* of a test *Series* is executed. In our example we execute a test pulse together with a command pulse for excitation of a fluorescence dye.

```
REPEAT          ( 0.000s)  : sweeps 1.000s
    ...
  Sweep         ( 0.000s)  : "TestPulse", "", ""
    ...
END_REPEAT
```

Conditional Statement: Within the *Acquire Each Sweep* event we place a conditional statement. In our example the "IF" statement is used to start a special action upon a key stroke.

If key Help (F1) is hit, we jump to the *GOTO_MARK* "Stimulus_1" which is located in the special actions section. Key F2 jumps to "Stimulus_2". Key F12 jumps to the mark "End" to enter the postfix section of the protocol to terminate the experiment.

```
    IF                  ( 0.000s):  Key = Help
      ClearKey          ( 0.000s)
      GOTO              ( 0.000s): "Stimulus_1"
   ELSIF                ( 0.000s):  Key = F2
      ClearKey          ( 0.000s)
      GOTO              ( 0.000s): "Stimulus_2"
   ELSIF                ( 0.000s):  Key = F12
      ClearKey          ( 0.000s)
      GOTO              ( 0.000s): "End"
   END_IF
```

Special Actions: Upon jumping to one of the *GOTO_MARKs* the acquisition of a *Series* is executed. In our example when jumping to

"Stimulus_1" the *Series* "Stim_Ramp" is executed and then we return to the *GOTO_MARK* "Chart" to reenter the experimental main loop.

```
;Special Actions
GOTO\_MARK              ( 0.000s): "Stimulus_1"
Series                  ( 0.000s): "Stim_Ramp", "", ""
GOTO                    ( 0.000s): "Chart"
GOTO_MARK               ( 0.000s): "Stimulus_2"
Series                  ( 0.000s): "Stim_IV", "", ""
GOTO                    ( 0.000s): "Chart"
GOTO_MARK               ( 0.000s): "End"
```

8.3.3 Postfix

Redraw all Online Analysis results: At the end of the experiment we would like to display the online data of the whole experiment in the Online Window 2. Therefore, we select the *Analysis Method Whole Results* and replay all results of the present *Group*.

```
;Change X-axis scaling and redraw Online Analysis
Online                  ( 0.000s): "Whole Results"
Replay                  ( 0.000s): Group
```

8.4 Chart Recording: An Example

The following example is configured for working with an EPC 10 Single patch-clamp amplifier. In case you are using another amplifier (e.g. EPC 9 or EPC 10 Double or Triple) please double check the assignment of the acquisition channels in the example PGF file.

8.4.1 Getting Started

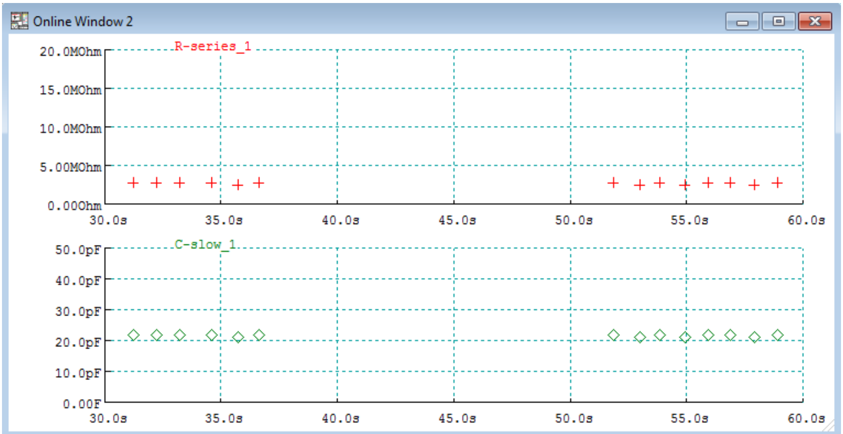
- Please connect the model cell to your amplifier.
- Start the PATCHMASTER software.
- Load the following example files:
  - ChartRecording.pgf
  - ChartRecording.onl
  - ChartRecording.pro
- Open (create) a new data file.
- Bring the model cell into the whole-cell configuration. To do so:
  - Switch the model cell to the 10 MΩ position and press the "SETUP" button in the Amplifier window.
  - Then switch the model cell to the middle position and press "SEAL".
  - Finally, switch the model cell to the 0.5 GΩ position and press "WHOLE-CELL".
- Now apply some typical holding potential. Let's say "-70 mV".

Now we are ready to start the experiment.

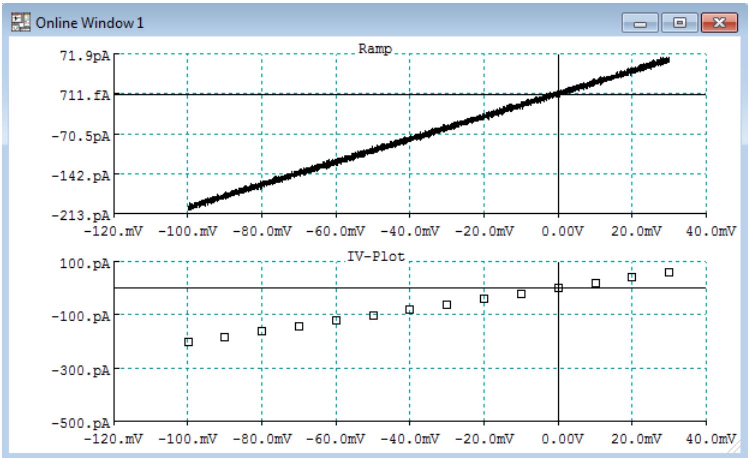
8.4.2 The Chart Recording

During the example recording you should execute or note the following points:

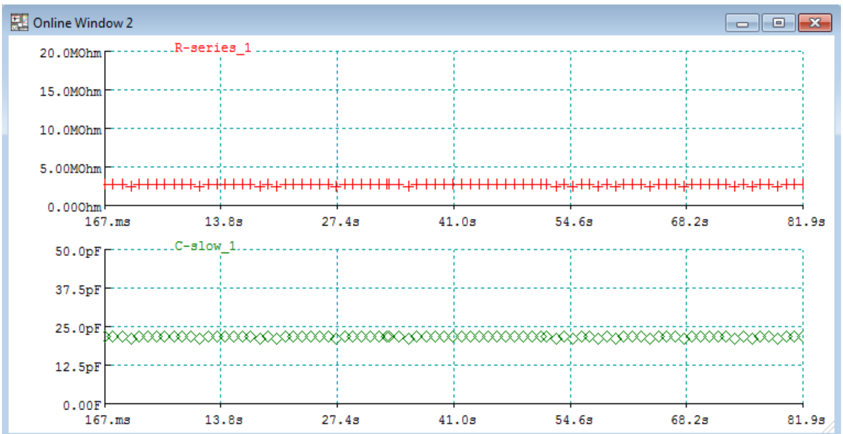
- You will start the chart recording by executing the protocol "Chart". To do so, select the protocol in the Control window.
- Now you will see *R-series* and *C-slow* values displayed in the two graphs of Online Window 2.



- When the *Timer* time crossed a multiple of 30 seconds the chart will be wrapped and wiped.
- You can play around with it and e.g. execute some other sequences (in this example we have prepared the protocol and a PGF for two different stimulation sequences). You can execute the PGF sequences "Stim_Ramp" and "Stim_IV" by pressing the keys F1 or F2 on your keyboard. Either you will see a ramp or an IV-relation in the graphs of Online Window 1. After the "Stim_Ramp or "Stim_IV" sequence is finished the protocol will automatically return to the "Testpulse" sequence.



- If you would like to terminate the experiment press the key F12. Then, automatically all *R-series* and *C-slow* values of the present (last) *Group* will be replayed and displayed in Online Window 2.



When starting the next experiment a new *Group* will automatically created.

9. Using the EPC 10 in Current Clamp Mode

9.1 Introduction

The current clamp mode of the EPC 10 features, in contrast to the EPC 7/8/9 series patch clamp amplifier, a voltage follower circuit.

9.1.1 The Gains in Current Clamp

CC-Gain: The EPC 10 features four current clamp gain (*CC-Gain*) ranges. The *CC-Gain* acts like a stimulus scaling, converting the voltage from the DA output of the interface to a current stimulus in the amplifier:

- 0.1 pA/mV (± 1 nA range)
- 1.0 pA/mV (± 10 nA range at 100 MΩ input resistance)
- 10.0 pA/mV (± 100 nA range at 10 MΩ input resistance)
- 100.0 pA/mV (± 1 μA range at 1 MΩ input resistance)

The gain range can be pre-selected while the amplifier operates in voltage clamp mode (typically *On Cell* or *Whole Cell*).

In the *CC-Gain* ranges of 0.1 and 1.0 pA/mV the amplifier operates in the medium gain range. This means that the current will be injected via the 500 MΩ resistor in the headstage.

In the *CC-Gain* ranges of 10.0 and 100.0 pA/mV the amplifier operates in the low gain range. This means that the current will be injected via the 5.1 MΩ resistor in the headstage.

The *CC-Gain* ranges 10.0 and 100.0 pA/mV are typically used with input impedances of up to 10 MΩ. This would typically correspond to a connection of an open patch pipette to the amplifier.

For larger input impedances (100 MΩ and more) the lower *CC-Gain* ranges have to be used for optimal performance.

In general one should try to operate the amplifier in the lowest *CC-Gain* range.

Current Gain: The *Gain* control in the Amplifier window sets the input *Current Gain*. In current clamp mode this means the gain used when reading back the current stimulus.

9.1.2 How much current can be injected?

The amount of current injected into the cell is limited by:

- the input impedance connected to the probe input
- the feedback resistor in the headstage
- the *CC-Gain*
- the usable compliance voltage

We will formulate the question in the way that we ask for the maximal input impedance that would limit the current injection at the maximal current in the respective *CC-Gain* range.

For example, let's assume that we work in the *1.0 pA/mV* range. Nominally the maximal current in this range is 10 nA. Since this *CC-Gain* range uses the 500 MΩ resistor in the headstage for current injection, we know that at this resistor 5 V (= 10 nA * 500 MΩ) will drop if 10 nA are passing. The EPC 10 amplifier can use a compliance voltage which is 1 V larger than the voltage dropping at the injection resistor or a maximum of 10 V. In our case we could use 6 V compliance, 5 V would drop at the injection resistor, remaining 1 V to drive current through the input impedance. Hence, the input impedance is limited by:

$$R = \frac{1\ V}{10\ nA} = 100\ M\Omega \tag{9.1}$$

In case the cell has a larger input resistance (e.g. 200 MΩ), then only 5 nA could be injected.

If the input impedance is less than 100 MΩ, than one can increase the current injection capabilities by choosing the next higher *CC-Gain* range.

Now, let's assume that we work in the *10.0 pA/mV* range. Nominally the maximal current in this range is 100 nA. Since this *CC-Gain* range uses the 5.1 MOhm resistor in the headstage for current injection, we know that at this resistor 0.51 V (= 100 nA * 5.1 MΩ) will drop if 100 nA are passing. The EPC 10 amplifier can use a compliance voltage which is 1 V larger than the voltage dropping at the injection resistor or a maximum of 10 V. In our case we could use 1.51 V compliance, 0.51 V would drop at the injection resistor, remaining 1 V to drive current through the input impedance. Hence, the input impedance is limited by:

$$R = \frac{1\ V}{100\ nA} = 10\ M\Omega \tag{9.2}$$

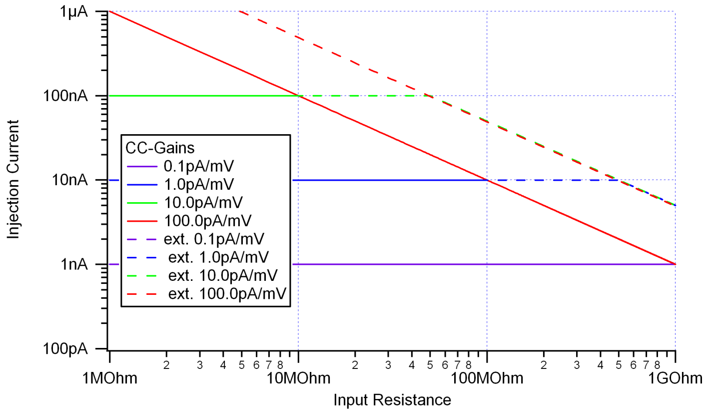
The *CC-gain* range of 100 pA/mV should be used with the EPC 10 not featuring the *Extended Stimulus Range* (up to revision S) with input impedances less than 10 MΩ only.

In case the EPC 10 amplifier is revision T or later, than it can be used with *Extended Stimulus Range* mode turned on (see `EPC 10` menu). In this mode the amplifier can use a compliance voltage 5 V larger than the voltage dropping at the injection resistor or a maximum of 10 V. This increases the current injection capabilities with low input impedances.

Let's assume that we work in the *100.0 pA/mV* range. Nominally the maximal current in this range is 1000 nA. Since this *CC-gain* range uses the 5.1 MΩ resistor in the headstage for current injection, we know that at this resistor 5.1 V (= 1000 nA * 5.1 MΩ) will drop if 1000 nA are passing. The EPC 10 amplifier with *Extended Stimulus Range* can use a compliance voltage which is 5 V larger than the voltage dropping at the injection resistor or a maximum of about 10 V. In our case we could use the full 10.1 V compliance, 5.1 V would drop at the injection resistor, remaining 5 V to drive current through the input impedance. Hence, the input impedance is limited by:

$$R = \frac{5\ V}{1000\ nA} = 5\ M\Omega \tag{9.3}$$

The following graph illustrates how much current can be injected in the different *CC-Gain* ranges in dependence of the input resistance. The dashed lines show the current injection capabilities of amplifiers with *Extended Stimulus Range*.



9.1.3 Bridge Balance

The idea of a balanced bridge in current clamp mode is to be able to inject current through the resistance of the pipette while still being able to measure the true membrane voltage. With other words, the *Bridge* compensates the voltage drop over the resistance of the pipette (series resistance) in current clamp mode.

When the amplifier is switched from voltage clamp to current clamp mode, the *R-series* control turns in the *Bridge* control. In order to turn the *Bridge* on, set the time constant to the smallest value (here 10 μs) and set the percentage control to 100 %. Then, the *Bridge* compensates exactly the value of *R-series* that was calculated during *C-slow* compensation and is displayed in the *C-slow* section.

> ***Note:*** *In case the time constant of the Bridge control is set to a slower value (e.g. 100 µs) then voltage transients are visible the injected current steps to another value and it is difficult to distinguish transients originating from uncompensated fast capacitance and transients due to slow bridge compensation.*

With the *Bridge* the following maximal access resistances can be compensated:

- 0.1 pA/mV and 1.0 pA/mV CC-Gain: 500 MΩ
- 10 pA/mV and 100 pA/mV CC-Gain: 5 MΩ

9.1.4 C-fast Compensation

The *C-fast* compensation in current clamp mode works as in voltage clamp mode. The automatic *C-fast* compensation however is disabled. The *C-fast* setting greatly influences the stability of the amplifier. Thus, *C-fast* is reduced internally by 0.5 pF in current clamp mode (and reset to its full value after switching back to a voltage clamp mode).

9.1.5 C-slow Compensation

The *C-slow* compensation is automatically turned OFF when switching to current clamp mode.

The *C-slow* compensation can be turned on in current clamp mode to inject in addition to a commanded current pulse a capacitance loading transient as set by *C-slow* and *R-series* parameters into the cell. This results in faster increase of the voltage signal.

> ***Note:*** *Using C-slow compensation in current clamp mode requires careful adjustment of C-fast and C-slow values and also*

very stable recording conditions. Slight maladjustment or overcompensation might result in oscillations, which might kill the cell.

9.1.6 Filtering in Current Clamp Mode

For EPC 10 up to revision S:

- *Filter 1* is set to 10 kHz (fixed).
- *Filter 2* sets the bandwidth of the current monitor signal.
- The voltage input is filtered with a time constant given by the access resistance times the uncompensated *C-fast* value (typically 5 μs = 10 MΩ * 0.5 pF).

For EPC 10 revision T and later:

- *Filter 1* is used to filter the current signal and is set to 10 kHz or 30 kHz (fixed).
- *Filter 2* can be used to either filter the voltage or the current signal. When setting *Filter 2* to *V_Bessel* the voltage signal can be low passed filtered before acquisition.

9.2 Examples

The following examples have been performed with an EPC 10 patch clamp amplifier and PATCHMASTER software version 2x73.

9.2.1 The Model Circuit

The subsequent examples are just for demonstration. We test the procedures with a very special model circuit containing:

- 50 MΩ parallel to 4.7 pF: simulating a high resistance pipette in the bath with stray capacity.
- 4.7 pF only: simulating a clogged electrode or an ideal *On Cell* configuration, stray capacitance only.
- 4.7 pF stray capacitance in parallel to 100 MΩ (access resistance) in series with (500 MΩ parallel to 47 pF): simulating an impaled cell or a whole cell configuration with a very high access resistance and stray capacitance of the electrode.

9.2.2 PATCHMASTER Configuration

For manual bridge balance adjustments and capacitance compensation in current clamp mode it is useful to be able to scale the display of the *Test Pulse*. This option has to be enabled in the Configuration window. The scaling and offset values for the current and voltage *Trace* of the *Test Pulse* can be entered in the right part of the Amplifier window. Please select the tab *Show All* for easy access to those controls.

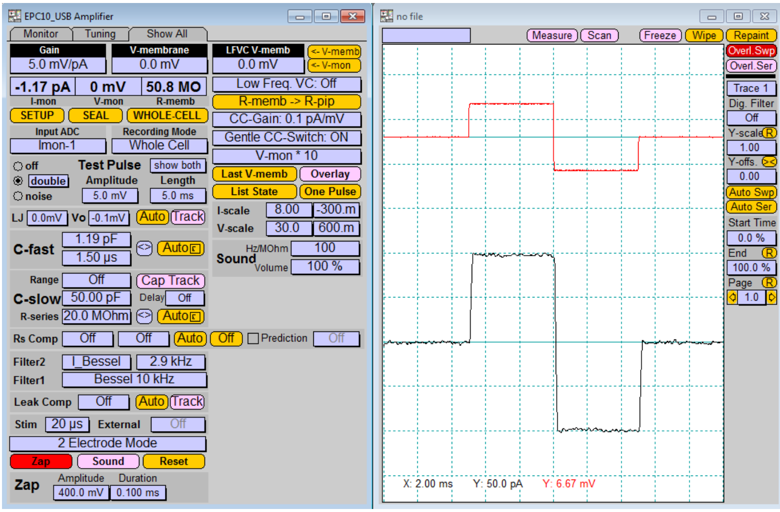
9.2.3 1. Intracellular voltage recording using sharp electrodes

In our example the sharp electrode has a resistance of about 50 MΩ and is sharp enough to impale a cell.

We connect the model cell described above to the EPC 10 headstage and switch to the 50 MΩ position (electrode in the bath).

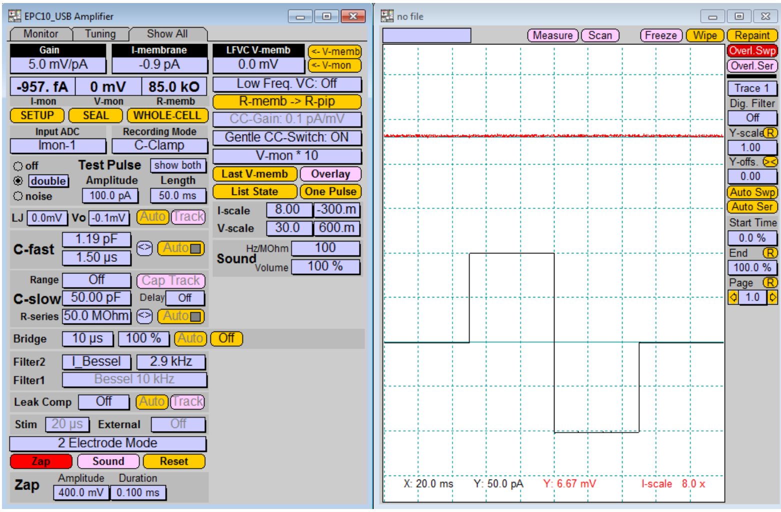
Next, we press the *SETUP* button in the Amplifier window to reset the amplifier. We have a *Test Pulse* on and read a pipette resistance of 50.5 MΩ in the `R-memb` field. The *V-scale* and *I-scale* of the *Test Pulse* has been increased for better visualization of the pulses.

Before switching into the current clamp/bridge mode and balance the bridge we do an *Auto C-fast* compensation.



To facilitate the compensation in the current clamp/bridge mode, we first type 50.5 MΩ in the *R-series* field and then select *C-Clamp* as *Recording Mode*.

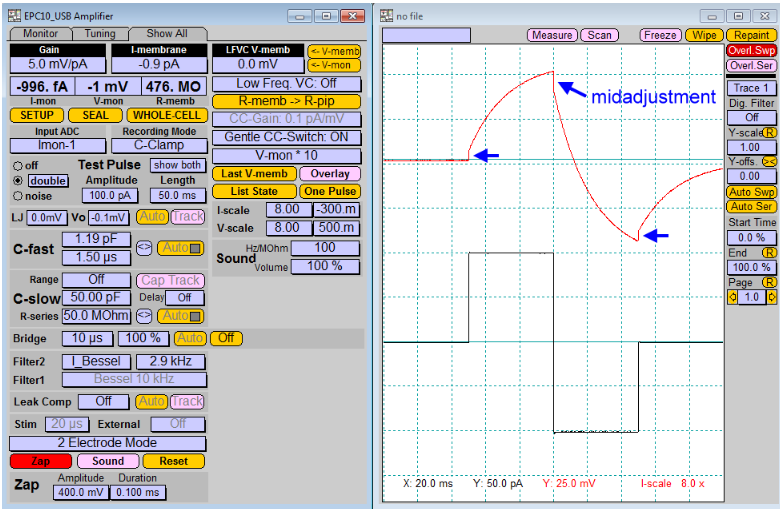
After switching in current clamp mode the first time you will have to set the *Bridge* to 10 μs and 100 % and adjust the *Test Pulse* amplitude and duration. These settings will be stored and automatically set when switching to current clamp again.



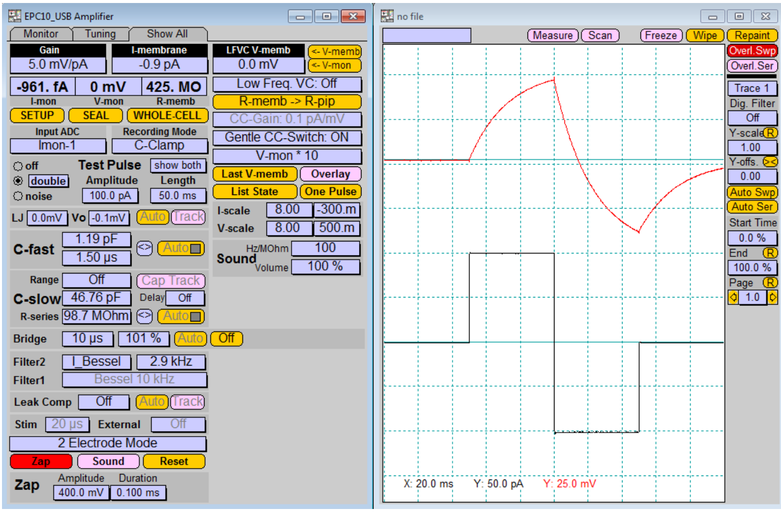
We see that the *Bridge* is very well balanced. No voltage steps and no cpacitive transients due to the current injection are visible. When *C-fast* is not compensated very well some capacitive transients due to the stray capacitance of the pipette may be visible. In current clamp mode these transients can be manually compensated by adjusting the *C-fast* controls. Do increase the total compensated capacitance value very slowly until the *Trace* is flat.

> ***Important note:*** *Overcompensation of the C-fast value easily result in oscillation of the amplifier! Please try to compensate the capacitance not faster than in steps of 0.1 pF in order to stop increasing the compensation before the amplifier starts oscillating.*

Now we will impale the cell.



Since the series resistance has changed while penetrating the cell (simulated by increasing the access resistance in the third position of the model cell to 100 MΩ), we clearly see fast voltage steps following the onset of the current pulses. Now we have to readjust the bridge balance by adjusting the *R-series* value.



Note: *R-series has been increased from 50 MΩ to 98.7 MΩ.*

9.2.4 2. Automation of Bridge Balancing

Automation of bridge balancing can be performed by switching to voltage clamp mode, measure and compensate the parameters *R-series* and *C-fast* respectively and then switch back to current clamp mode and set the bridge balance to the measured values. This can be done as well with the electrode in the bath and with the cell impaled or in whole-cell configuration.

9.2.4.1 Balancing the bridge while the electrode is in the bath

```
Amplifier ( 0.000s): OnCell
Command ( 0.000s): "E CSlowRange 0"
Command ( 0.000s): "E PulseOn True"
Command ( 0.000s): "E SaveRPip"
Amplifier ( 0.000s): C-fast
Value ( 0.000s): Value-1 = "I PipetteResistance"
Value ( 0.000s): Value-1 MUL 1.0000 , copy to "E RSeries"
Amplifier ( 0.000s): CClamp
Value ( 0.000s): Value-1 = 1.0000, copy to "E RSComp"
Command ( 0.000s): "E PulseOn True"
Switch ( 0.000s): "Amplifier"
```

9.2.4.2 Balancing the bridge during intracellular recording

```
Amplifier ( 0.000s): C-slow, WholeCell
Amplifier ( 0.000s): CClamp
Command ( 0.000s): "E RSComp 100% "
Command ( 0.000s): "E PulseAmp 1000"
Command ( 0.000s): "E PulseDur 50"
Command ( 0.000s): "E PulseOn True"
Switch ( 0.000s): "Amplifier"
```

9.2.5 3. Current clamp recordings with high resistance patch pipettes

Patch electrodes with high resistances may be used to reduce intracellular washout.

The whole-cell configuration will be established as in a conventional patch clamp experiment in voltage clamp mode. For a detailed description please refer to the PATCHMASTER Tutorial `The First Experiment`.

From the holding potential in whole-cell voltage clamp mode we switch to current clamp mode. If *Gentle CC-Switch* is selected, the holding current will be automatically adjusted such that the cell is held at the membrane potential that was set in voltage clamp mode.

We can now turn on the *Test Pulse* and the *Bridge* compensation and set the percentage value to 100 %. Then we continue with adjustment of the bridge mode as described before (9.2.3 on page 114).

9.3 Related Topics

9.3.1 Automatic stimulus adjustment to elicit APs

We setup a PGF sequence for current clamp, in which we control the amplitude of the stimulation via a *PGF parameter*.

In the Online Analysis we analyze the peak voltage during the stimulation.

A protocol contains a loop controlling the stepwise increase of the stimulus amplitude. In case the voltage is below a threshold (for firing action potential), we increase the amplitude (increase the value of the *PGF parameter*). If the voltage crosses the threshold, we know that the stimulus has elicited an action potential. Hence, the loop in the protocol is stopped and we continue with the experiment, e.g. applying a train of stimuli with the optimal stimulus amplitude.

In the following we show the protocol, we have written for this application:

```
SetOsci           ( 0.000s): WipeOnline, Tr(N)= 1111111111111111
SetPgf            ( 0.000s): PgfParam-10 =  0.0000
REPEAT            ( 0.000s): inf 1.000s
   Series            ( 0.000s): "StimTest", "", ""
   IF                ( 0.000s):  Online-2 >  10.000m
      BREAK             ( 0.000s): repeat
   END_IF
   SetPgf            ( 0.000s): PgfParam-10 INC  10.000m
END_REPEAT
Series            ( 0.000s): "Stim", "", ""
```

Comments:

- Online function 1 returns the amplitude of the stimulus.
- Online function 2 returns the voltage peak.

> ***Note:*** *A similar protocol can be found in the demo configuration* `Automatic Stimulus Adjustment in Current Clamp Mode` *on our homepage (http://www.heka.com/support/tuto.html).*

9.3.2 Adjusting the holding current to keep the cell at its resting potential

9.3.2.1 The Low Frequency Voltage Clamp (LFVC)

This mode is hardware implemented and therefore independent from the data acquisition sequence. It is always active when turned on.

- Go to current clamp mode.
- Set the resting membrane potential for *LFVC*:
  - use the *V-memb* button to copy the last membrane potential from the voltage clamp mode.
  - use the *V-mon* button to copy the current membrane potential value (corresponding to *I-holding* currently set).
  - or enter the potential manually
- Turn *LFVC* on and set an appropriate time constant of the *LFVC* mode.
- Use a stimulus *relative to V-memb* in the PGF sequence.

9.3.2.2 Adjustment of the holding current via a protocol

In case the hardware implementation of this mode can not be used, the adjustment of the holding current can be implemented in a protocol.

- First, we require a measurement of the membrane potential. This can be either done via the Online Analysis or you directly read this value from the corresponding control of the Amplifier window (use the *SetValue* command).
- Then you compare this value to the set membrane potential and decide if the holding current should be increased or decreased.

With such a loop the holding current can be adjusted with a time resolution of about one per second.

9.3.3 Automatic oscillation detection

An automatic oscillation detection can be programmed on basis of a variance measurement and included in the main experimental protocol.

Measure variance of the voltage *Trace* (in current clamp) or current *Trace* (in voltage clamp) in the baseline (e.g. first segment) of a *Test Pulse*.

At the beginning of the experiment, record typical variance values and then tune/teach the system when it should detect an oscillation. This can be done by setting a threshold for the variance result. In case the threshold is crossed, a parameter that causes the oscillation can be automatically adjusted. E.g. in voltage clamp reduce the percentage of *Rs Comp*, or in current clamp reduce the *C-fast* value.

10. Rapid Mode Switching during Acquisition

One method to study ionic currents after or during physiological stimuli is to switch rapidly from one recording mode to another. Let us explain the intention of this new feature with an example:

In current clamp mode we elicit an action potential. In order to study e.g. the ionic currents that contribute to the repolarization phase or to the delay after depolarization (DAD) we want to switch at time t from current clamp mode to voltage clamp.

10.1 General Prerequisites

In the following section we describe which parameters have to be set for mode switching during sweep acquisition.

10.1.1 Amplifier Settings

Current Gain: When *Rapid Mode Switching* is used, the *Current Gain* in the Amplifier window must be in the medium range.

CC-Gain: *Rapid Mode Switching* can be used in the following *CC-Gain* ranges:

- 0.1 pA/mV = ± 1 nA range
- 1.0 pA/mV = ± 10 nA range

Leak Comp.: Must be turned off.

LFVC: Must be turned off.

10.1.2 PGF

Sampling Frequency: Sampling Frequency should be as fast as possible 200 kHz. One should reduce the number of samples by compression.

Stim-DA: One DA must be the stimulus for the amplifier *Stim-DAC*. During the phase in current clamp mode it defines the current output, during the voltage clamp phase the voltage output. Scaling of pA to mV is:

- 1 mV/pA when switching from VC to CC
- 1 pA/mV when switching from CC to VC

Build Instruction: There are 2 key words to be entered in the build instruction of the channel (recall: must be in the part after the ";"):

- VC-Switch=[segment number] → switch to VC mode.
- CC-Switch=[segment number] → switch to CC mode.

The position of the switching is at the beginning of the specified segment.

Link: The *Link* channel of the channel with the *Switch=* build instruction must point to the DA-channel with the stimulus for the amplifier Stim-Dac.

Digital Out: One DA must be set to *Dig-out (word)*. It is used to send the commands to the EPC 10.

10.1.3 Preadjustment of the holding current/voltage after the switch

The holding current/voltage after the switch can be set via the *PGF parameter* 1. In case you do not want to set it to a user defined value, you can e.g. execute first a recording without a mode switch and measure the holding current/potential at desired time of the switch. This value can then be assigned to the *PGF parameter* 1.

10.2 Examples

10.2.1 Settings

We provide an example configuration `Rapid Mode Switching` which contains a set of Pulse Generator sequence, Online Analysis and protocols. You can download this example from the tutorial section of the PATCHMASTER software on our web site (http://www.heka.com/support/tuto.html).

The following examples have been performed with an EPC 10 patch clamp amplifier and PATCHMASTER software version 2x71.

We test the following procedures with our standard MC9/10 model circuit.

10.2.1.1 PGF Sequences

In our example we have set up 4 different Pulse Generator sequences:

1. "VC → CC_Switch"
2. "CC → VC_Switch"
3. "VC → CC_Test"
4. "CC → VC_Test"

VC → CC_Switch / CC → VC_Switch:

These sequences are used to switch the recording mode while acquiring data. All sequences have in common:

- Sample Interval: 200 kHz
- DA channel 1: *Stim-DA*
- DA channel 2: *Dig-out (word)*
- AD channel 1: *Imon-2*
- AD channel 2: *Vmon*

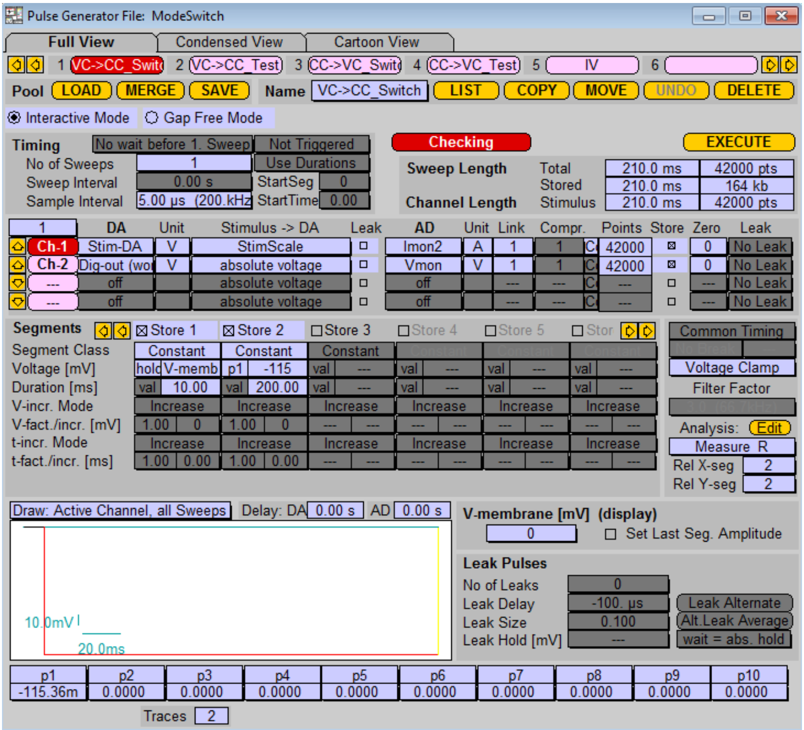
- Built Instruction on AD channel 1:
  - ";CC-Switch=2": In case of switch to current clamp mode at the beginning of the second segment.
  - ";VC-Switch=4": In case of switching to voltage clamp mode at the beginning of fourth segment.

VC → CC_Switch This PGF sequence consists of 2 segments:

- Segment 1: Baseline segment; the time of switch can be adjusted with the duration of this segment.
- Segment 2: Switching segment; at the beginning of this segment the *Recording Mode* is switched from voltage clamp to current clamp. The amplitude of the holding current after the switch is set via the *PGF parameter* 1.

The relevant segment is the 2^{nd} segment and we use the Online Analysis *Measure R*.

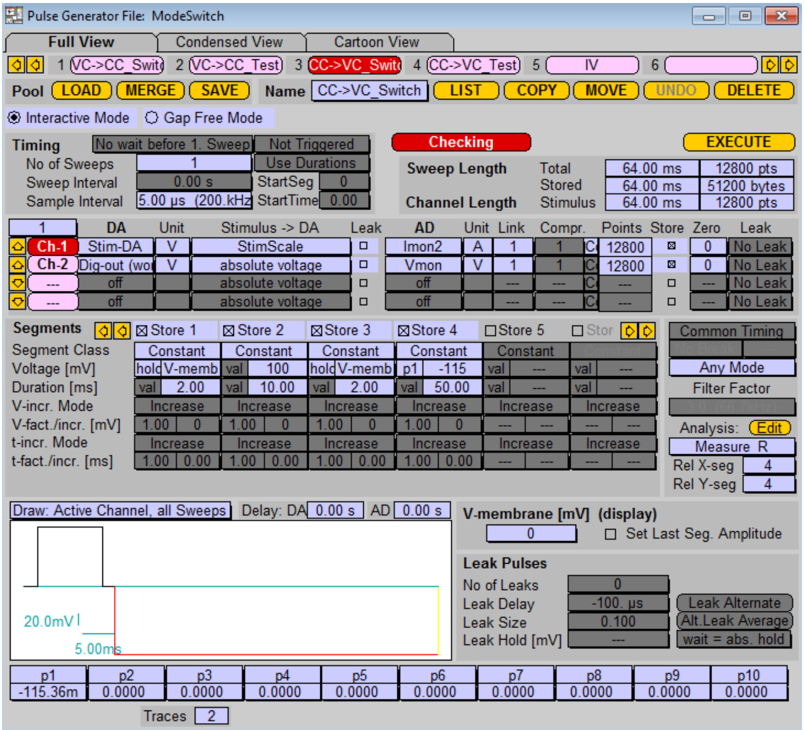
The sequences *CC → VC_Test* and *VC → CC_Test* are used to measure the holding potential and holding current that should be applied after switching from one to the other mode.



CC → VC_Switch This PGF sequence consists of 4 segments:

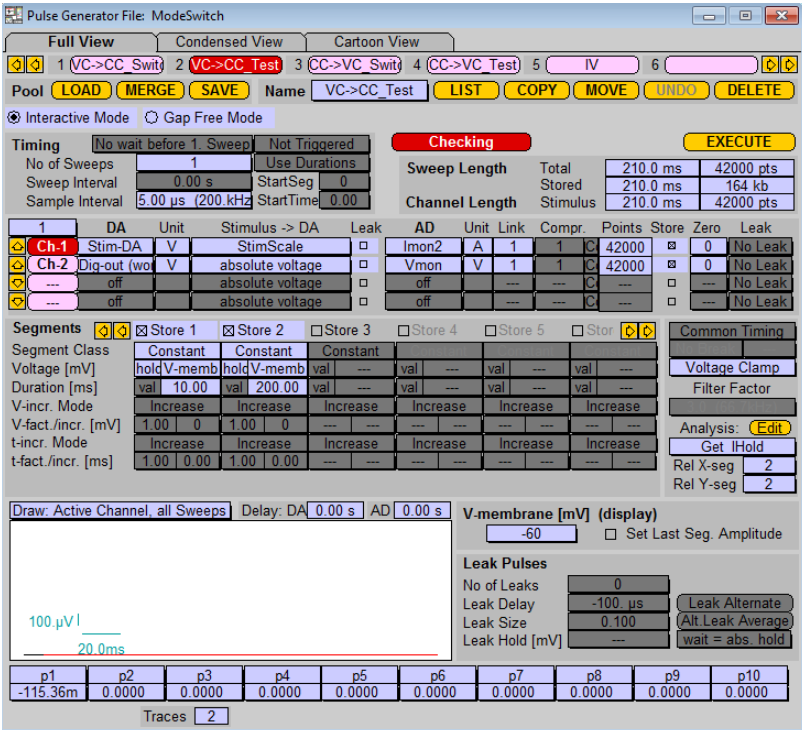
- Segment 1: Baseline segment.
- Segment 2: Current injection; the potential at the model circuit is increasing.
- Segment 3: Baseline segment; discharging period of the model circuit; the potential is decreasing. The time of switch can be adjusted with the duration of this segment.
- Segment 4: Switching segment; at the beginning of this segment the mode is switched from current clamp to voltage clamp. The amplitude of the holding potential after the switch is set via the *PGF parameter* 1.

The relevant segment is the 4^{th} segment and we use the Online Analysis *Measure R*.



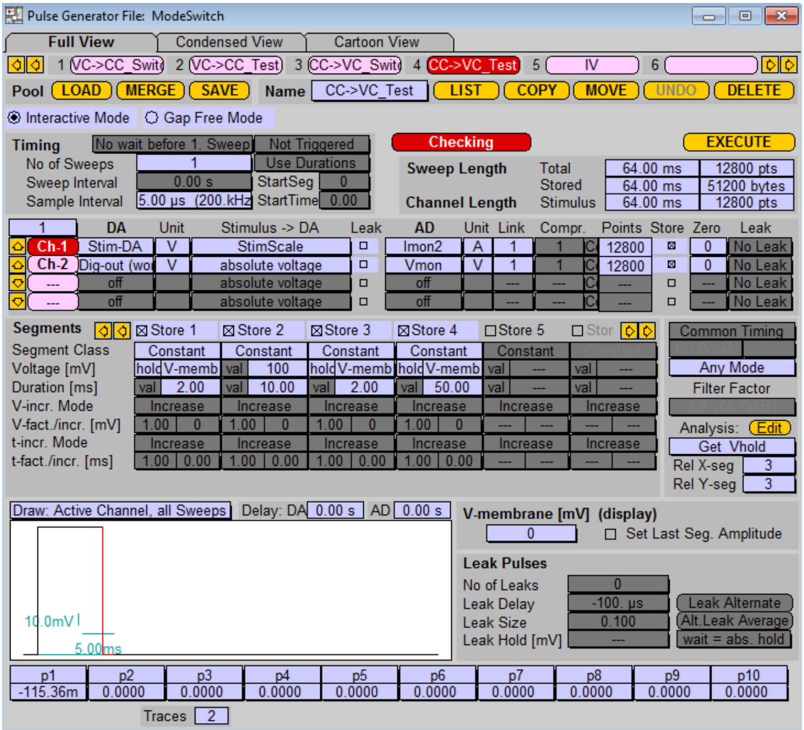
VC → CC_Test This sequence is identical to the sequence "VC → CC_Switch" except the following:

- No entry in the build instruction of the first AD channel (no switch command).
- Segment "2" is also a baseline segment with holding potential as amplitude (same as segment "1").
- We use the Online Analysis *Get Ihold* to measure the current at the holding potential.



CC → VC_Test This sequence is identical to the sequence "CC → VC_Switch" except the following:

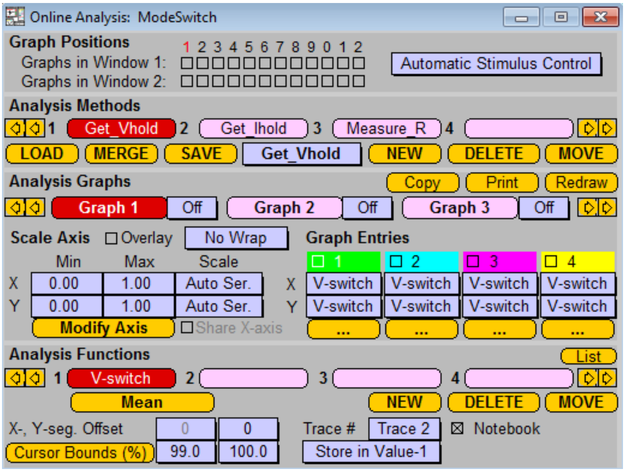
- No entry in the build instruction of the first AD channel (no switch command).
- Segment "4" is also a baseline segment with holding potential as amplitude (same as segment "3").
- The relevant segment is segment number "3" and we use the Online Analysis *Get Vhold* to measure the potential at the end of segment number "3".


Pulse Generator File: ModeSwitch
Full View
Condensed View
Cartoon View
1 VC->CC_Swit
2 VC->CC_Test
3 CC->VC_Swit
4 CC->VC_Test
5 IV
6
Pool LOAD MERGE SAVE
Name CC->VC_Test
LIST COPY MOVE UNDO DELETE
Interactive Mode
Gap Free Mode
Timing
No wait before 1. Sweep
Not Triggered
No of Sweeps 1
Use Durations
Sweep Interval 0.00 s
StartSeg 0
Sample Interval 5.00 µs (200.kHz)
StartTime 0.00
Checking
EXECUTE
Sweep Length
Channel Length
Total 64.00 ms 12800 pts
Stored 64.00 ms 51200 bytes
Stimulus 64.00 ms 12800 pts
DA Unit Stimulus -> DA Leak AD Unit Link Compr. Points Store Zero Leak
Ch-1 Stim-DA V StimScale Imon2 A 1 1 12800 0 No Leak
Ch-2 Dig-out (wo V absolute voltage Vmon V 1 1 12800 0 No Leak
Segments
Store 1 Store 2 Store 3 Store 4 Store 5
Segment Class
Voltage [mV]
Duration [ms]
V-incr. Mode
V-fact./incr. [mV]
t-incr. Mode
t-fact./incr. [ms]
Constant
hold V-memb
val 100
val 2.00
val 10.00
val 50.00
Increase
Common Timing
Any Mode
Filter Factor
Analysis: Edit
Get Vhold
Rel X-seg 3
Rel Y-seg 3
Draw: Active Channel, all Sweeps
Delay: DA 0.00 s
AD 0.00 s
10.0mV
5.00ms
V-membrane [mV] (display)
0
Set Last Seg. Amplitude
Leak Pulses
No of Leaks 0
Leak Delay -100. µs
Leak Size 0.100
Leak Hold [mV] ---
Leak Alternate
Alt.Leak Average
wait = abs. hold
p1 -115.36m
p2 0.0000
p3 0.0000
p4 0.0000
p5 0.0000
p6 0.0000
p7 0.0000
p8 0.0000
p9 0.0000
p10 0.0000
Traces 2

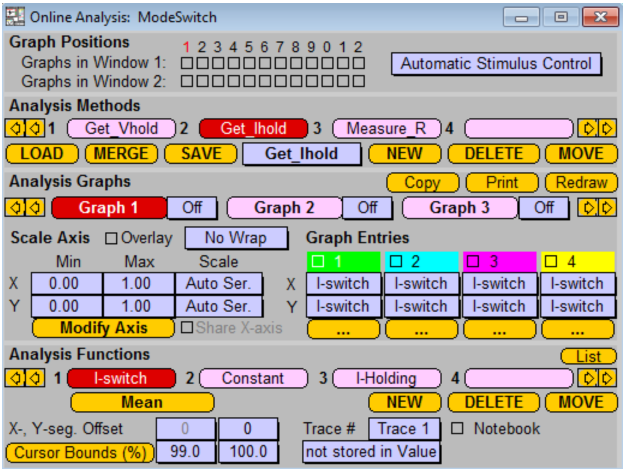
10.2.1.2 Online Analysis

We have set up three different *Online Analysis* methods:

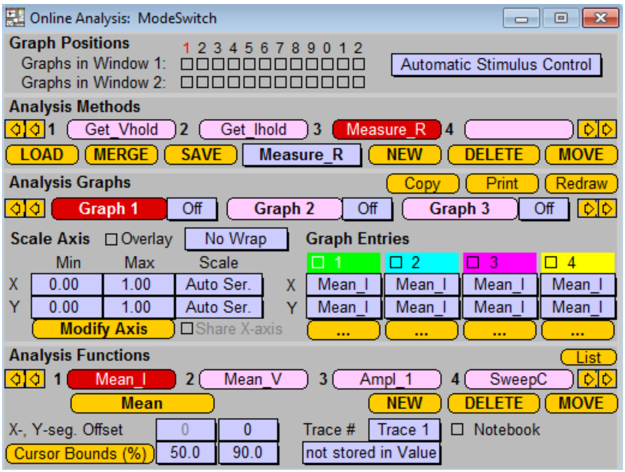
1. "GetVhold": Measures the holding potential that should be applied after the switch.



2. "GetIhold": Measures the holding current that should be applied after the switch.



3. "Measure_R": Measures the current and voltage after the mode switching. This method is also used to check for correct current and voltage scaling.



10.2.1.3 Protocols

We have set up one protocol that performs a switch from current clamp to voltage clamp mode ("TestCC-VC") and another that performs a switch from voltage clamp to current clamp mode ("TestVC-CC").

The protocols are configured in a way that one can repetitively execute them without changing the recording mode in the amplifier window. When starting the protocols, the amplifier should be in the whole-cell recording mode.

TestCC-VC:

```
Online           ( 0.000s): Auto
Amplifier        ( 0.000s): WholeCell
Command          ( 0.000s): "E FIlter2 10.0"
Amplifier        ( 0.000s): C-slow
Command          ( 0.000s): "E CSlowRange 0"
Command          ( 0.000s): "E GentleSwitch 0"
Command          ( 0.000s): "E CCGain 0"
Amplifier        ( 0.000s): CClamp
; Turn on Bridge Mode
Command          ( 0.000s): "E RSMode 2"
Command          ( 0.000s): "E RSComp 100"
; Measure potential at time of switch
Series           ( 0.000s): "CC->VC_Test", "", ""
; Configure switch PGF sequence
SetPgf           ( 0.000s): PgfParam-1 = Value-1
#SetPgf           ( 0.000s): PgfParam-1 =  0.0000
; Execute switch sequence
Series           ( 0.000s): "CC->VC_Switch", "", ""
```

TestVC-CC:

```
Online           ( 0.000s): Auto
Amplifier        ( 0.000s): Vh= -60.000mV, WholeCell
Command          ( 0.000s): "E FIlter2 10.0"
Amplifier        ( 0.000s): C-slow
Command          ( 0.000s): "E CSlowRange 0"
Command          ( 0.000s): "E GentleSwitch 0"
Command          ( 1.000s): "E CCGain 0"
```

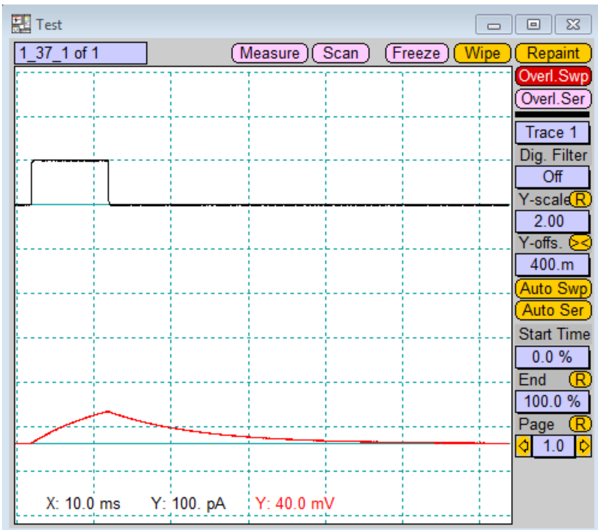
```
; Get Holding current
Series           ( 0.000s): "VC->CC_Test", "", ""
; Configure switch PGF sequence
SetPgf           ( 0.000s): PgfParam-1 = Value-1
#SetPgf           ( 0.000s): PgfParam-1 =  0.0000
Series           ( 0.000s): "VC->CC_Switch", "", ""
Amplifier        ( 0.000s): WholeCell
```

10.2.2 Switching from Current Clamp Mode to Voltage Clamp

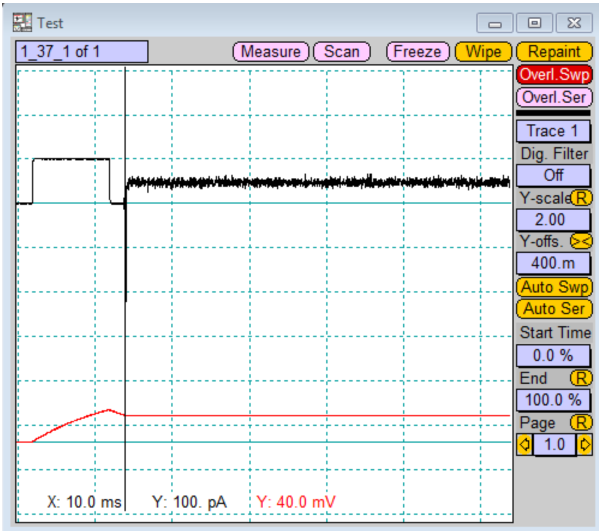
Please open a data file and establish the whole-cell configuration with the MC9 or MC10 model circuit.

Then execute the protocol "TestCC-VC".

Two *Series* with one *Sweep* will be acquired. The first *Series* is used to record in current clamp mode and measure the potential at the time when switching to voltage clamp mode.



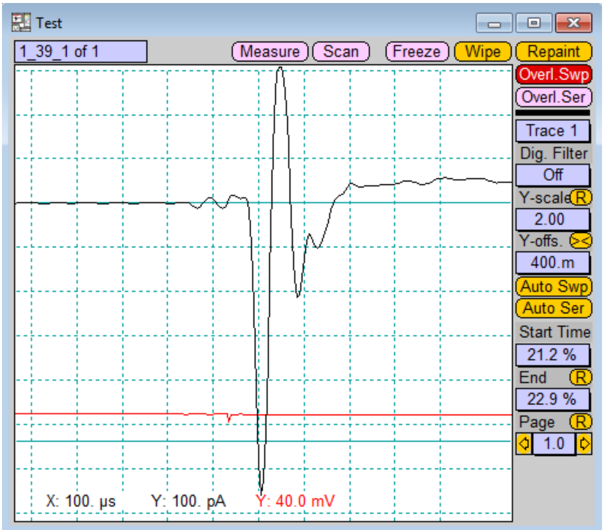
This potential value is copied to the *PGF parameter 1* that defines the holding voltage after the switch (in the sequence "*CC_VC_Switch*"). Then the switch sequence is executed.



As you can see from the data above, the potential is nicely kept at level before the switch.

The calculated membrane resistance after the switch is about 505 MΩ. This proofs that the scaling of the current and voltage display is correct.

Now, we have closer look for the switching artefact:



As you can see the maximum duration of the switching artefact is less than 300 μs.

> ***Note:*** *The duration of the switching artefact depends on the sampling frequency since the multiple switching commands that have to be executed for switching from one to the other mode, are executed consecutively with each sample.*

The potential at the cell is kept constant during the switch.

In a second run of the protocol we want to switch to 0 volts after the switch and measure the discharging current of the model cell. To do this, we have to change the protocol slightly. Please skip the line:

```
SetPgf           ( 0.000s): PgfParam-1 = Value-1
```

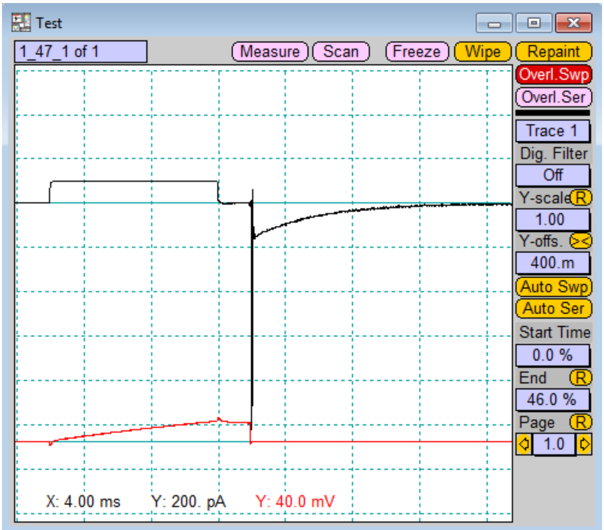
and un-skip the line:

```
#SetPgf          ( 0.000s): PgfParam-1 =  0.0000
```

Then, execute the protocol `TestCC-VC` again.

In the second series you will see a fast current transient that recovers to zero. The time constant of this current decay is given by R-series * R-membrane / (R-series + R-membrane) * C_M. With the standard MC10 model circuit the time constant for current decay is about 0.1 ms. This might be too fast to nicely separate from the switching artefact.

We, therefore, have used a slightly modified model cell with R-series = 100 MΩ and C_M = 47 pF, resulting in a time constant of 3.9 ms.

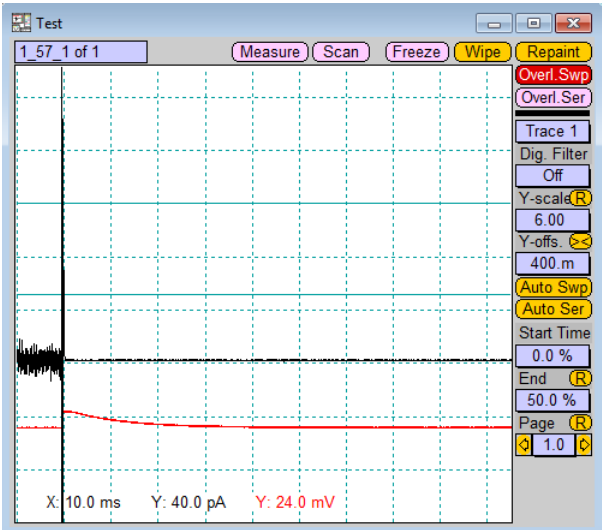


10.2.3 Switching from Voltage Clamp Mode to Current Clamp

- Please establish the whole-cell configuration with the MC9 or MC10 model circuit.
- Execute the protocol `TestVC-CC`.

Two *Series* with one *Sweep* will be acquired. The first *Series* is used to record in voltage clamp mode the current at the holding potential.

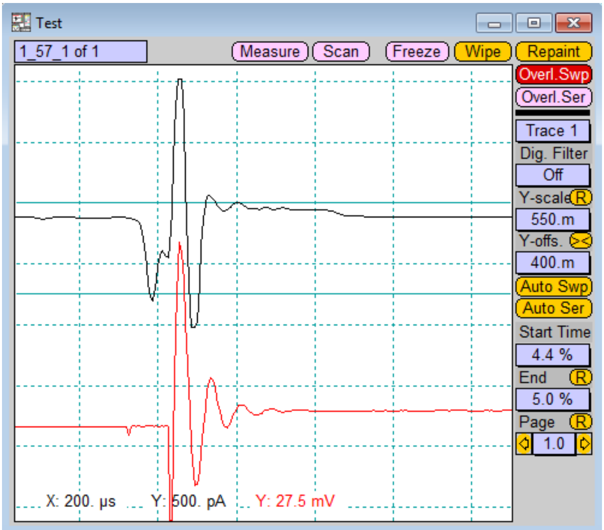
This potential value is copied to the *PGF parameter* 1 that defines the holding current after the switch (in the sequence "*VC_CC_Switch*"). Then the switch sequence is executed.



The current after the switch is kept at the level before the switch.

The voltage approaches its final value after the mode switch with a time constant: R-membrane * C_M.

Now, we have a closer look for the switching artefact:



The overall duration of the initial switching artefact is about 500 μs.

Here, you also have the option to switch to 0 pA after the switch and measure the discharging voltage of the model cell by skipping event line "11" and activating event line "12".

11. Fluorescence Measurements

11.1 Introduction

In the following tutorial we will describe a whole-cell experiment with loading a ratiometric fluorescence dye (e.g. Fura-2) through the patch pipette into the cell.

Before we start the detailed description how to set-up and configure PATCHMASTER, we will first outline the different phases of the experiment.

1. Approach of the cell with the patch pipette and formation of a seal.
2. During seal formation usually some fluorescence dye diffuses into the bath solution. Therefore, after seal formation we will detect a declining fluorescence signal since the dye molecules are diffusing in the bath solution away from the detection area. Due to dilution effects the fluorescence signal will decrease in the following tens of seconds. In this phase of the experiment, we have to wait for a stable fluorescence background, originating from auto fluorescence of the cell and dye in the tip of the patch pipette. To reduce the contribution of fluorescence from dye molecules in the patch pipette you should restrict the field of fluorescence excitation and detection to the area of the cell. When background fluorescence has been stabilized we will record this fluorescence as background fluorescence for later correction of the fluorescence ratio.
3. Establishment of the whole-cell configuration (break through) and recording of dye loading into the cell. During this phase of the experiment we start the fluorescence ratio measurement. We can follow the dye loading level and get first measurements of a fluorescence ratio.

In general, we will have to control the fluorescence excitation light source and record the fluorescence signal from the fluorescence detector. In addition, there can be some corrections and calculations be performed to get the corrected ratio of the fluorescence or even the free Ca^{2+} concentration.

11.2 Setting up PATCHMASTER

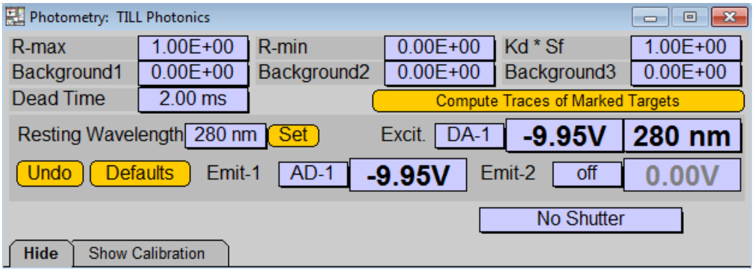
In the following chapter we describe all tasks which should be done before starting the final experiment. Our description outlines using a photometry setup which accepts an analog voltage for wavelength control and provides an analog voltage (0 to 10 V) proportional to the fluorescence signal. Usually, such system consists of a monochrometer for fluorescence excitation and a photomultiplier or photo diode as fluorescence detector. Such a system can be controlled by PATCHMASTER using the `T.I.L.L. Photometry Extension`.

All PGF sequences, protocols and *Online Analysis* explained in detail in the following tutorial can also be downloaded from our homepage: http://www.heka.com/support/tuto.html.

11.2.1 Configuration

11.2.1.1 Activate the Photometry Extension

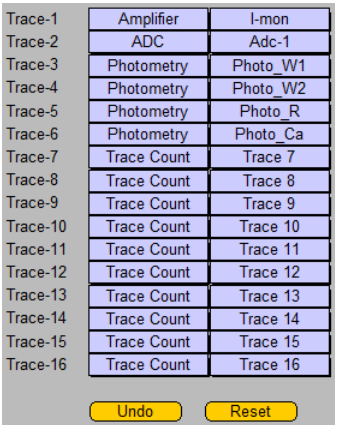
After starting PATCHMASTER open the Configuration window (`Windows` → `Configurations`) and select the `T.I.L.L. Photometry Extension` in the "Hardware" tab (see also PATCHMASTER reference manual). Thereafter, please open the Photometry dialog by selecting `Windows` → `Photometry`.



In the Photometry dialog you can set the basic settings for your photometry device. Please remember the DA and AD channel for *Excitation* and *Emission* which will be needed when creating the PGF. For the calibration of the monochrometer we refer to the PATCHMASTER reference manual and the product manual of the monochrometer. Once the monochrometer is calibrated you can specify the excitation wavelength in PATCHMASTER directly. Conversion to the corresponding command voltage will be done by the `Photometry Extension`.

11.2.1.2 Trace Assignment

Next, we want to use the *Trace Assignment* option to label each of the acquired *Traces* regarding their origin with individual labels. Therefore we select the "Trace Assign" tab in the Configuration window and assign the first 6 *Traces*:



Now, we have to save all the settings and restart PATCHMASTER.

11.2.2 Creating the PGF sequences

For our experiment we need three PGF sequences:

TestFura: A sequence which applies a test pulse and records one pair of fluorescence values. You will use this sequence e.g. during and shortly after seal formation for measuring the decline of the background fluorescence.

RatioFura: This sequence applies a defined holding potential and records one pair of fluorescence values. In addition the calcium concentration is calculated and stored in a separate *Trace*. This sequence will be used when recording a baseline.

Ca_Entry: A depolarizing pulse is given and the corresponding changes in fluorescence, respectively calcium concentration, is recorded.

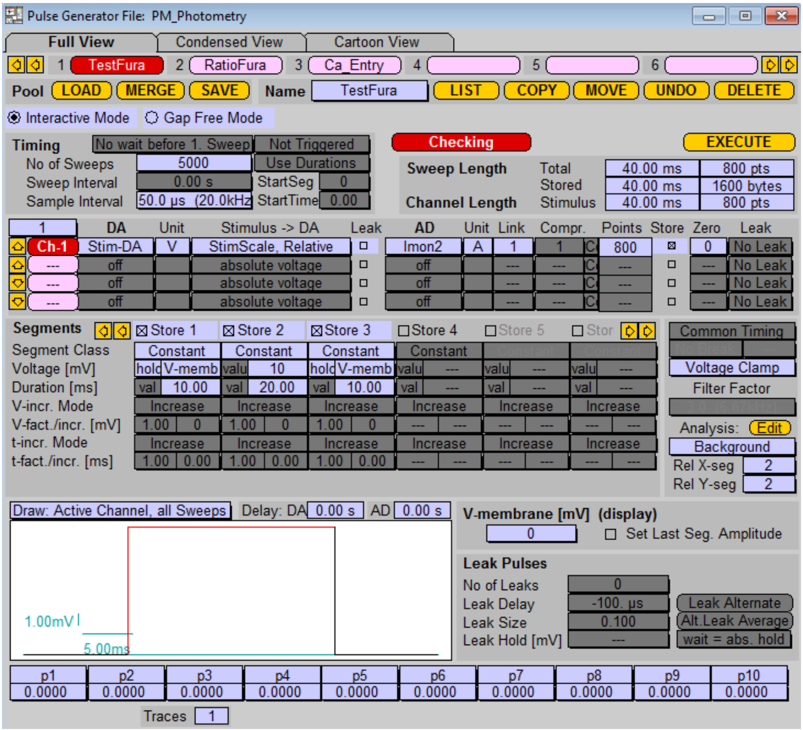
11.2.2.1 The "TestFura" sequence

Open the Pulse Generator (`Windows` → `Pulse Generator`) and load the default protocol pool (`DefPgf_v9.pgf`). Create a new PGF by clicking into an empty field or modify an existing PGF. Save the new Pulse Generator file under a new name. The name of the first PGF we want to create is "TestFura".

Please make the following settings while the first DA channel is selected in your PGF:

- Disable *Wait before 1. Sweep*.
- Enter a large *No of Sweeps* (e.g. "5000").
- Set *Sample Interval* to 50 μs.
- The *Stim-DA* should be set to optionStimScale (`Stimulus` → `DA`).
- In addition activate *relative to V-memb* (`Stimulus` → `DA`).
- Select *Imon2* as an AD channel.
- Create three *Constant* segments with 10, 20 and 10 ms *Duration*.
- Set the *Voltage* to: *holding*, 10 mV and *holding*.
- Enter "Background" in the *Analysis Method* field.
- Set *Rel X- and Y-seg* to "2".

Your PGF for the 1^{st} channel will look like this:

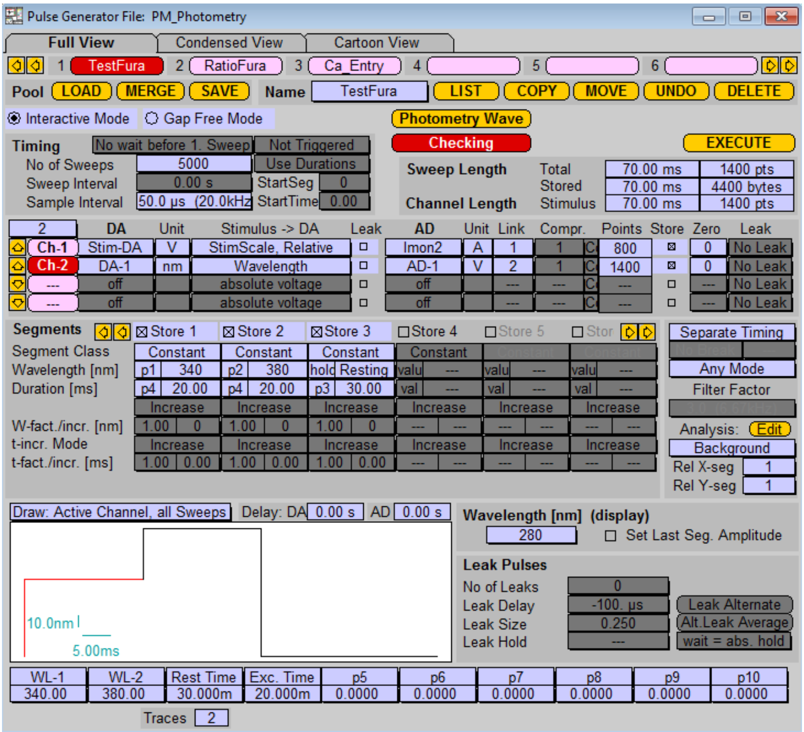


We use the second channel for controlling the fluorescence excitation device and record the fluorescence signal from the detector of the photometry system. The following settings should be made while the second DA channel is selected:

- Set *use for Wavelength* in `Stimulus` → `DA` for *DA-1* which is the excitation channel set in the photometry configuration (see 11.2.1.1 on page 141).
- Set AD-1 as emission channel as specified in the photometry configuration (see 11.2.1.1 on page 141).
- Set *Link* to "2".

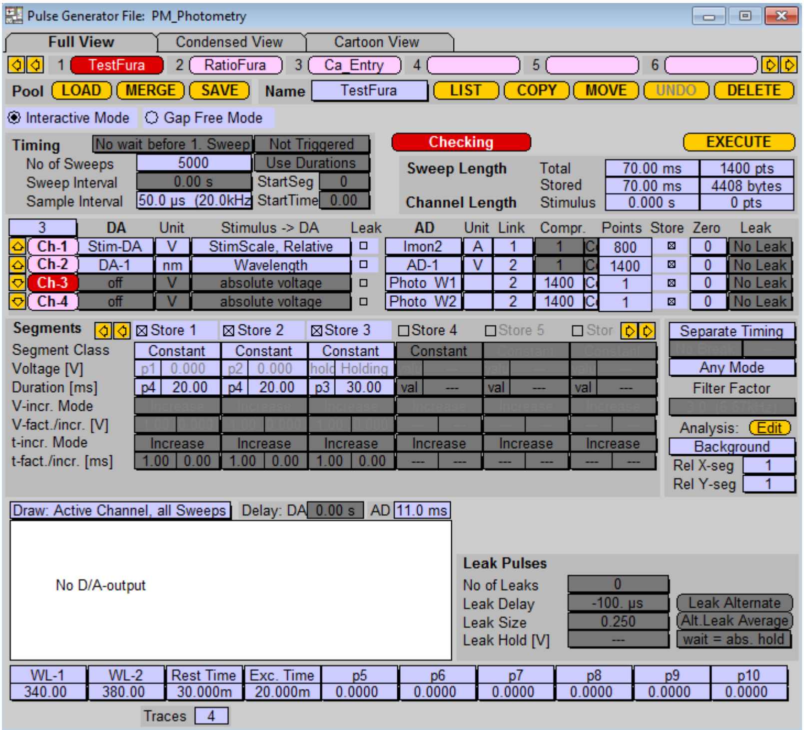
- Activate *Separate Timing*.
- Select *Any Mode* instead of *Voltage Clamp*.
- Set the *Rel Y- and X-seg* to "1".
- Set the *Wavelength* of the segments to: p1 – p2 – *holding*.
- Set the *Duration* of the segments to: p4 – p4 – p3.
- Rename the *PGF Parameters* p1 – p4 to "W1", "W2", "Rest Time" and "Exc. Time", respectively.
- Enter "340", "380", "30m" and "20m" for these four *PGF Parameters*.

The PGF for the 2nd channel will look like this:



Now we need two more AD channels for the recording of the fluorescence signal:

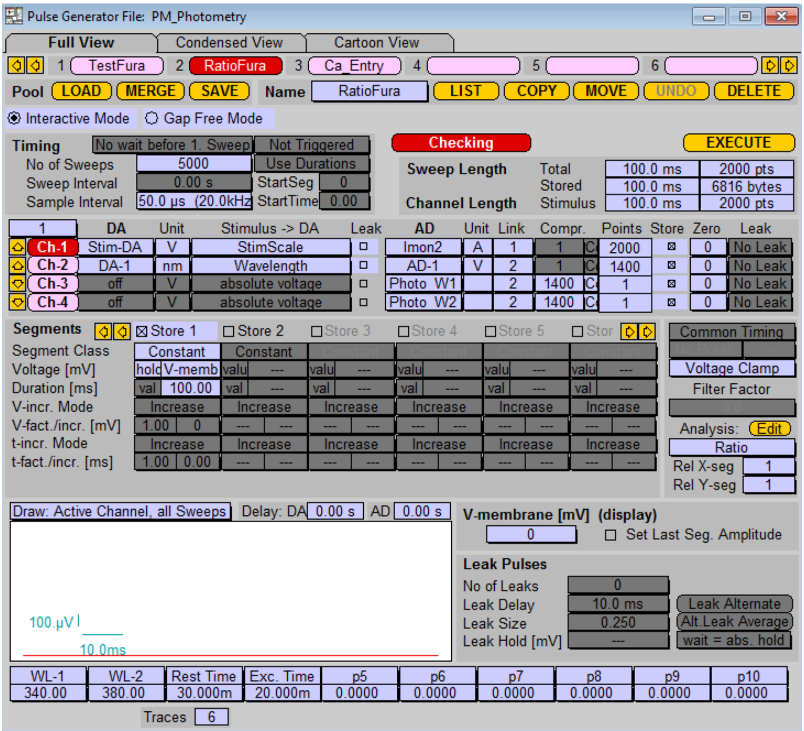
- Set the AD input for channel "3" to *Photo_W1* (first wavelength).
- Set the AD input for channel "4" to *Photo_W2* (second wavelength).
- Set *Link* for both AD channels to "2" (corresponding emission channel).



11.2.2.2 The "RatioFura" sequence

The "RatioFura" sequence can be derived from the "TestFura" sequence. Please copy the "TestFura" sequence and name it "RatioFura". For channel "1" we will do the following modifications:

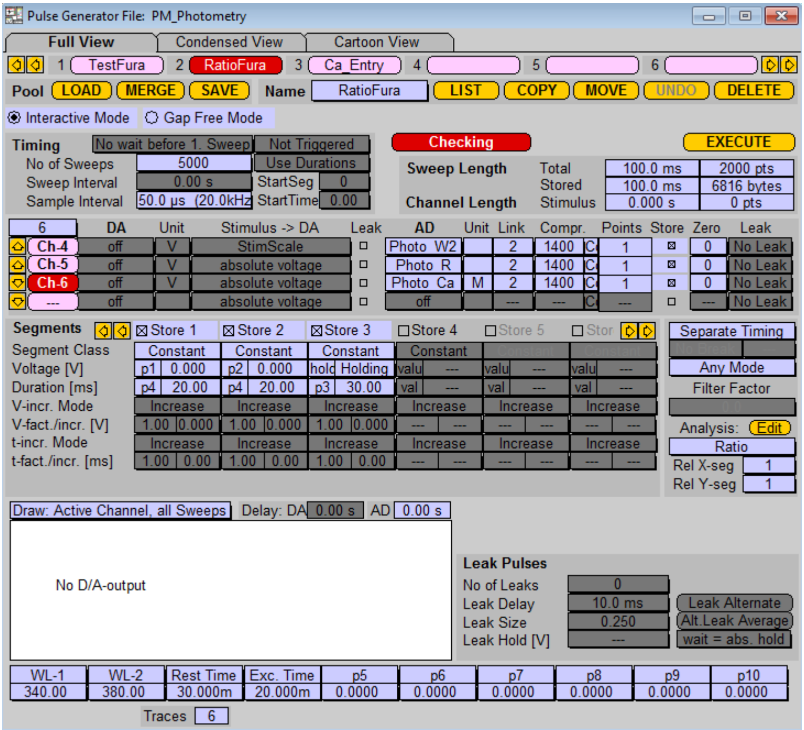
1. Change *Rel X- and Y-seg* to "1".
2. Change the Online Analysis method from "Background" to "Ratio".
3. Delete segments number "2" and "3".
4. Set the *Duration* of segment "1" to 100 ms.
5. Deselect *relative to V-memb* in `Stimulus` → `DA` for *Stim-DA*.



Next, we add two more AD channels to the PGF sequence:

- Add *Photo_R* for the ratio of the wavelength 1 and 2 for channel "5".
- Add *Photo_Ca* for the calcium concentration for channel "6".
- Set *Link* of both AD channels to "2".

Now we have 6 AD input channels for our photometry experiment:

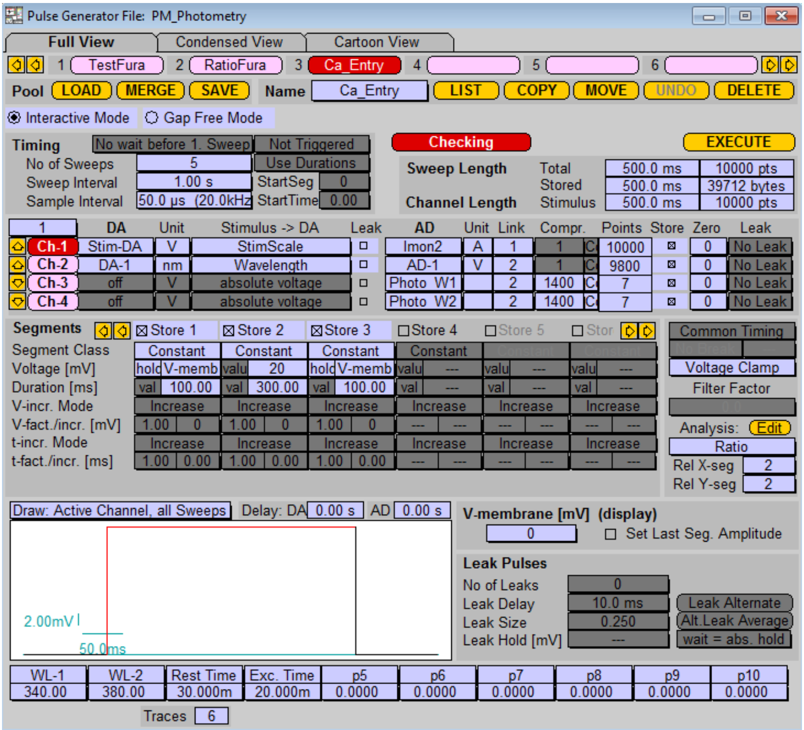


11.2.2.3 The "Ca_Entry" sequence

Our last PGF we have to create is the "Ca_Entry" sequence. Copy the "RatioFura" sequence and select again the first DA channel to do the following modifications:

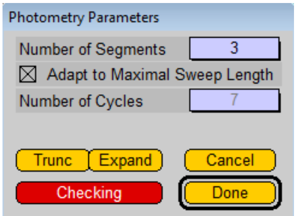
- Set *No of Sweeps* to e.g. "5".
- Set the *Sweep Interval* to 1 s.
- Enable three *Constant* segments.
- Set the *Voltage* to: holding – 20 mV – holding.
- Set the *Duration* to: 100 ms – 300 ms – 100 ms.
- Set the *Rel X- and Y-seg* to "2".

Our PGF for the first channel will look like this:

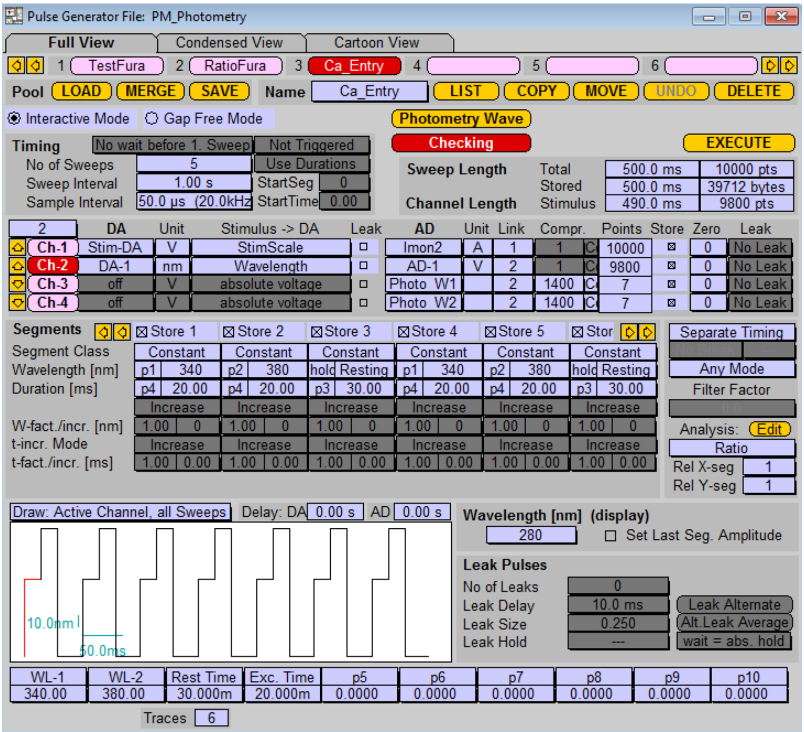


Next, we select DA channel "2". There, we open the Photometry Parameters by clicking on the yellow *Photometry Wave* button. In the Photometry Parameters window we check for:

- *Number of Segments* = "3".
- *Adapt to Maximal Sweep Length* is checked.



We press the *Expand* button to get as many photometry cycles in our *Sweep* as possible and leave the Photometry Parameters dialog by clicking on the *Done* button. Thereafter, we will see in the number of segments and in the cartoon that we now have seven photometry cycles in one *Sweep*:



11.2.3 Creating the Online Analysis

For creating our Online Analysis we start from the scratch. Open the the default `DefAnal.onl` file and delete all *Analysis Methods*. Save the modified `DefAnal.onl` file with a new name. In the next steps we will add three new Online Analysis methods:

Background: This method will be used to analyze the fluorescence values and store them in a parameter values.

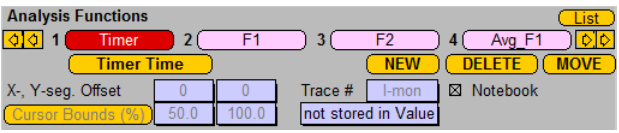
Ratio: This method will be used for the fluorescence ratio analysis during the experiment.

Replay: This method will be used to replay all results of the experiment in the Online window after our protocol ended.

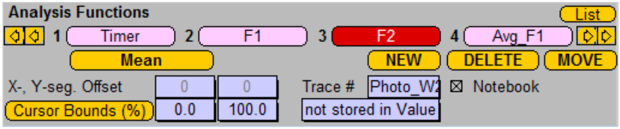
11.2.3.1 The "Background" Analysis Method

In the upper part of the Online Analysis window, we create a new *Analysis Method* and name it "Background". In the *Analysis Functions* we have to configure five *Analysis Functions*.

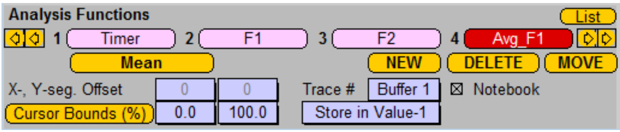
The first function we need is the *Timer Time*. We name this function just "Timer". We will use this function to calculate the X-value for the online display.



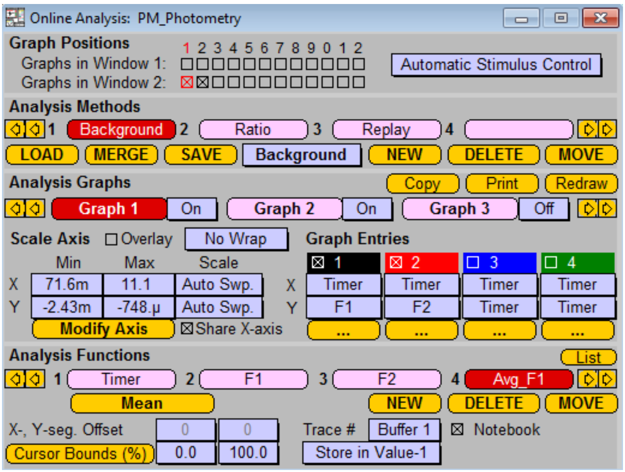
For the analysis of the fluorescence values we create two similar functions. First, we select the function *Mean* from the *Measurements* functions in the Function Type dialog. We name this function "F1" and confirm it by clickimg the *Done* button. In the *Analysis Functions* section we have to set the *Trace #* to "Photo_W1" since the fluorescence signal of our first wavelength is recorded in this assigned *Trace*. Thereafter, we repeat these steps but name the function "F2" and select "Photo_W2".



In the last two functions of the "Background" analysis we will calculate the mean of the *Traces* accumulated in the buffers (for details see 11.2.4.1 on page 156). Therefore, we select a *Mean* function from the Function Type dialog and name it "Avg_F1". In the *Analysis Functions* section we select "Buffer 1" (*Trace #*) and *Store in Value-1*. For "Avg_F2", we repeat these settings but select "Buffer 2" and *Store in Value-2*.



We display "F1" and "F2" versus "Timer" in two *Graph Entries* in *Graph 1* and "Avg_F1" and "Avg_F2" in two *Graph Entries* in *Graph 2* in the Online Window 2. The scaling for the Y- and X-axis is set to *Auto after each Sweep*.



11.2.3.2 The "Ratio" Analysis Method

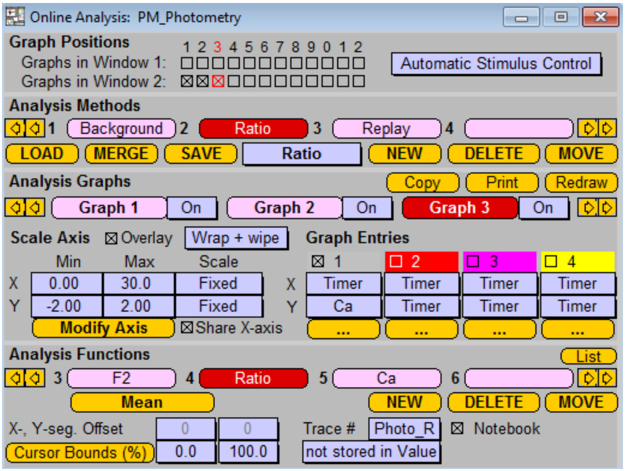
For creating the *Analysis Method* "Ratio" we copy the content of the *Analysis Method* "Background". Now, we substitute the *Analysis Functions* "Avg_1" and "Avg_2" with two new functions, one calculating the fluorescence ratio ("Ratio") and one calculating the calcium concentration ("Ca"). Both functions are *Mean* functions whereas "Ratio" refers to *Trace #* "Photo_R" and "Ca" refers to optionTrace # "Photo_Ca".

For showing the results in the Online Window 2 we activate three graphs:

- Graph 1: 2 *Graph Entries*
  - Timer vs. F1
  - Timer vs. F2
- Graph 2: Timer vs. Ratio
- Graph 3: Timer vs. Ca

For all graphs the following settings should be made in addition:

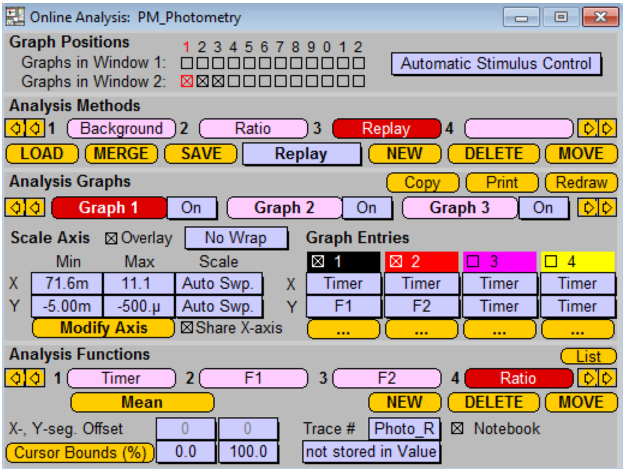
- Set fixed X- and Y-values.
- Enable *Share X-axis*.
- Enable *Wrap + wipe*.
- Enable *Overlay*.



11.2.3.3 The "Replay" Analysis Method

The last *Analysis Method*, called "Replay", is an identical copy of the "Ratio" *Analysis Method*. There we make just some minor modifications which enables us to replay our whole data at the end of our experiment (for details see 11.2.4.2 on page 159). The modifications are:

- Change the X- and Y-axis scaling from *Fixed* to *Auto after each Sweep*.
- Disable *Wrap + wipe* and set *No Wrap*.



11.2.4 Creating Protocols in the Protocol Editor

For creating the protocols in the Protocol Editor we start again from the scratch. Please open the default file (`DefProt.pro`). For the experimental control and automation we will add two new protocols:

Background: This protocol is important to determine the background fluorescence values.

Baseline: This protocol determines the fluorescence levels of the cells. In addition we can stimulate the cells to alter the calcium content (fluorescence levels).

11.2.4.1 The "Background" Protocol

At the beginning of the "Background" protocol we add an dialog explaining the function of the keys. We put this dialog in an *IF...THEN* event to be able to abort or to proceed the experiment.

```
;Dialog
IF                ( 0.000s): Break "Clear Background Buffer with "F10",
```

```
   Finish with "F12""
   BREAK            ( 0.000s): protocol
END_IF
```

After pressing the *Continue* button of the start dialog the protocol starts running. First, we set or reset initial parameters, e.g. we

- create a new *Group*
- wipe Oscilloscope and the Online Windows
- reset the *Timer*
- disable the storing of data
- set the *Online Analysis* mode to *Automatic Stimulus Control*

The storage function is disabled because the fluorescence values at the beginning of the experiment, before reaching the cell with the patch pipette, can be discarded.

```
;Initial settings
Acquire          ( 0.000s): Wipe=OFF,
File             ( 0.000s): NewGroup
SetOsci          ( 0.000s): Wipe, Timer, Tr(Y)= 0000000000111111
File             ( 0.000s): No_Store
Online           ( 0.000s): Auto
```

In the next step we clear our *Trace Buffers* and reset the background fluorescence values to "0".

```
;Clear buffer and background
TraceBuffer      ( 0.000s): Buffer-1, clear
TraceBuffer      ( 0.000s): Buffer-2, clear
Value            ( 0.000s): Value-1 =  0.0000, copy to "H Background1"
Value            ( 0.000s): Value-2 =  0.0000, copy to "H Background2"
```

Now we can start with acquiring fluorescence values. The *REPEAT* event starts with an *Acquire Each Sweep* event executing the "TestFura" sequence. After each *Sweep* the content of the *Traces* 3 and 4 (*Photo_W1*, *Photo_W2*) are accumulated in *Buffer* 1 and 2, respectively. Using the key F12 the repeat loop will be terminated whereas when pressing key F10 the buffer content is reset. It is recommended to reset the buffer if the fluorescence values are stable and you are ready to determine the background auto-fluorescence.

```
;Determining background
REPEAT           ( 0.000s): sweeps 1.000s
   Sweep            ( 0.000s): "TestFura", "", ""
   TraceBuffer      ( 0.000s): Trace-3 accumulate in Buffer-1, Osci, Online
   TraceBuffer      ( 0.000s): Trace-4 accumulate in Buffer-2, Osci, Online
   IF               ( 0.000s): Key = F12
      BREAK            ( 0.000s): repeat
   ELSIF            ( 0.000s): Key = F10
      TraceBuffer      ( 0.000s): Buffer-1, clear
      TraceBuffer      ( 0.000s): Buffer-2, clear
      ClearKey         ( 0.000s)
   END_IF
END_REPEAT
```

After terminating the repeat loop (F12) we enable the storage of the data and start with another *REPEAT* event. Inside the repeat loop we execute the "TestFura" sequence once and label the stored series ("Background").

```
;Store one sweep for background
File             ( 0.000s): Store
REPEAT           ( 0.000s): sweeps 1.000s
   Sweep            ( 0.000s): "TestFura", "Background", ""
   IF               ( 0.000s): RepeatCount >  0.0000
      BREAK            ( 0.000s): repeat
   END_IF
END_REPEAT
```

The fluorescence values which are stored now in *Value-1* and *Value-2* are copied to the photometry background values "Background1" and "Background2".

```
;Store background values in photometry configuration
Value            ( 0.000s): Value-1 copy to "H Background1"
Value            ( 0.000s): Value-2 copy to "H Background2"
```

In the last step we replace the photometry *Traces* "Photo_W1" and "Photo_W2" with the content of *Buffer-1* and *Buffer-2*.

```
;Replace last photometry traces for background
Replay           ( 0.000s): Trace
Command          ( 0.000s): "R ScrollDown"
Command          ( 0.000s): "R ScrollDown"
TraceBuffer      ( 0.000s): Buffer-1, Replace-3
Command          ( 0.000s): "R ScrollDown"
TraceBuffer      ( 0.000s): Buffer-2, Replace-4
```

11.2.4.2 The "Baseline" Protocol

After measuring the auto-fluorescence and establishing the whole cell configuration we can start measuring the fluorescence intensities in the cell. In addition, we will execute our "Ca_Entry" sequence and determine the calcium concentration. For a proper calculation of the calcium concentration according to the Grynkiewicz formalism specific values have to be entered in the Photometry configuration (see PATCHMASTER reference manual, chapter 19).

First, we start again with a dialog explaining the key functions.

```
;Dialog
IF               ( 0.000s): Break "Press "1" for stimulation, "F12" for break"
   BREAK            ( 0.000s): protocol
END_IF
```

Before starting with the main repeat loop we wipe the content of the Online Windows. Then we set a *GOTO_MARK* named "Baseline". Thereafter, the *REPEAT* event starts with the "Ratio" sequence. Once more we have two key definitions: 1 for executing the "Ca_Entry" sequence in between the *Sweep* acquisition of the "RatioFura" sequence and F12 to terminate the acquisition.

```
;Main experimental loop
SetOsci          ( 0.000s): WipeOnline, Tr(N)= 1111111111111111
GOTO_MARK        ( 0.000s): "Baseline"
REPEAT           ( 0.000s): sweeps 1.000s
   Sweep            ( 0.000s): "RatioFura", "", ""
```

```
   IF                ( 0.000s): Key = "1"
      ClearKey          ( 0.000s)
      GOTO              ( 0.000s): "Stimulus"
   ELSIF             ( 0.000s): Key = F12
      ClearKey          ( 0.000s)
      GOTO              ( 0.000s): "End"
   END_IF
END_REPEAT
```

When the key 1 is used the protocol jumps to the defined *GOTO_MARK* "Stimulus", executing the "Ca_Entry" sequence. Thereafter, the protocol returns to the *GOTO_MARK* "Baseline" to continue the "RatioFura" *Sweep* acquisition.

```
;special actions
GOTO_MARK        ( 0.000s): "Stimulus"
Series           ( 0.000s): "Ca_Entry", "", ""
GOTO             ( 0.000s): "Baseline"
```

At the end of our protocol we wipe our displayed data, switch to the *Analysis Method* "Replay" and replay the results of the whole group.

```
;Replay all results
GOTO_MARK        ( 0.000s): "End"
SetOsci          ( 0.000s): Wipe, Tr(N)= 1111111111111111
Online           ( 0.000s): "Replay"
Replay           ( 0.000s): Group
```

11.3 The Experiment

This section is in principal a rough summary of your photometry experiment. Before you start the experiment you should open the Online Windows and the Control window.

First, approach the cell and form a giga-seal, as usual. When starting the protocol "Background" you should see some fluorescence signal displayed in the Online Window 2. The signal might be very close to zero since the cell is not loaded with dye yet. Wait until the fluorescence signal has stabilized. This remaining signal origins from auto-fluorescence of the cell and dye in the very tip of the patch pipette. The fluorescence values will be accumulated in the buffer and written into *Values*. At a certain point of the protocol, when the user terminates it, the *Values* will be stored as *Background* values. In addition, the content of the *Buffers* will overwrite the stored photometry *Traces* in the tree and label it "Background".

Next, we establish the whole-cell configuration, thereby loading the cell with our fluorescence dye. We start the "RatioFura" sequence which records the fluorescence signals in the cell. You will see a loading curve of the fluorescence dye. If the fluorescence signal is stable you can start evoking calcium currents with the "Ca_Entry" sequence.

12. Capacitance Measurements using the LockIn Extension

12.1 Basics

Membrane capacitance (C_m) measurements have been used extensively to study exocytosis and endocytosis in individual cells. Currently, the most popular techniques for measuring changes in C_m utilize a sinusoidal voltage stimulus and process the resulting sinusoidal current using a phase-sensitive detector or "lock-in amplifier" implemented either in hardware or software. A software-based phase sensitive detector is only a small part of the *LockIn* extension of the PATCHMASTER software. *LockIn* together with an EPC 9 and EPC 10 comprises a unique "virtual" instrument with unprecedented capabilities for performing cellular admittance measurements without the need for external filters. Admittance can be measured even when the current monitor signal is profoundly altered with the use of capacitance and/or series resistance compensation. Thus estimates of the actual values of equivalent circuit parameters (rather than just relative changes) can be generated from *bona fide* admittance measurements.

12.1.1 Sine + DC (SDC) versus Piecewise Linear technique (PL

Two different techniques are widely used for estimating changes in C_m (see Gillis, 1995 for details; see 12.7 on page 211) and are implemented as different modes by *LockIn*. Each takes a different approach towards resolving the dilemma that the use of a single sine wave stimulus provides only 2 pieces of information (magnitude and phase), whereas the equivalent circuit of a cell in the whole-cell recording configuration has 3 unknown parameters (G_s, G_m, and C_m).

In the first approach, which we call the "Piecewise Linear" (PL) technique (Neher and Marty, 1982; see 12.7 on page 211), no attempt is made to determine the actual value of the admittance or of any of the three equivalent circuit parameters. Instead, only *changes* in the parameters which result from measured changes in admittance are noted. The technique is based upon the approximation that small deviations in C_m lead to linear changes in the admittance of the equivalent circuit which are orthogonal to changes in admittance introduced by small deviations in G_m or G_s (Neher and Marty, 1982; Joshi and Fernandez, 1988; Fidler and Fernandez, 1989; see 12.7 on page 211). Thus if the output signal of the patch clamp amplifier is input to a phase-sensitive detector that is set to the appropriate phase angle, then the output of the instrument is linearly proportional to small changes in C_m, but relatively insensitive to changes in the resistive parameters. A signal is also generated from an orthogonal phase setting (the "G" signal), which reflects changes in either G_s or G_m. In the PL mode of *LockIn*, "G" is split into two signals: ΔG_s and ΔG_m. Each of these signals are scaled appropriately to reflect changes in their respective parameters.

> ***Note:*** *It is important to realize, however, that these two signals are actually differently scaled versions of the same Trace. Changes in G_s can not be distinguished from changes in G_m in the PL mode.*

In the second approach, which we call the *Sine + DC* method (SDC), the DC current, together with the real and imaginary parts of the admittance, provide the three necessary pieces of information (Lindau and Neher, 1988; see 12.7 on page 211). Since the DC current is used, the reversal potential of the membrane conductance (E_{rev}) must be known. This limitation, however, is often not very serious since errors in the assumed value of E_{rev} are not very critical under the common condition of $G_s \gg G_m$ (Gillis, 1995; see 12.7 on page 211).

Certainly the SDC method offers many advantages over the PL technique. The greatest advantage is that the SDC technique generates actual values of all three parameters rather than just relative changes. Why then is the PL technique still the most popular method for high-resolution recording of changes in C_m? One important reason is that making true admittance measurements requires one to keep track of many parameters (see `Parameters which affect admittance measurements`, 12.5.2 on page 188). On the other hand, calibration of the "C" *Trace* with the usual implementation of the PL technique requires simply offsetting the capacitance compensation circuitry by a fixed amount. This procedure automatically takes into account the amplitude of the stimulus sinusoid, the gain of the patch clamp amplifier, and other relevant factors. In addition, the required phase setting of the lock-in amplifier can be determined (albeit with potential errors; Gillis, 1995; see 12.7 on page 211) by series resistance dithering ("phase tracking", Fidler and Fernandez, 1989, see 12.7 on page 211) without a direct measurement of the phase shifts introduced by the patch clamp amplifier.

> ***Note:*** *The `LockIn Extension` uses a calculated method for setting the phase and calibrating the Traces in the PL mode that avoids the errors associated with series resistance dithering (see `Using LockIn in piecewise-linear mode`, 12.2.2 on page 175).*

One of the great advantages of the PATCHMASTER software coupled to an EPC 9 or EPC 10 patch clamp amplifier is that all of the instrumentation settings which can potentially affect an admittance measurement are "known" by the software. Thus the advantages of the SDC method can be realized with no additional record keeping burden imposed upon the user.

> ***Note:*** *The Sine + DC mode is the recommended or "standard" mode of operation of the `LockIn Extension`. The Piecewise Linear mode was included to satisfy those users who may be more comfortable with this "traditional" type of measurement while making the transition to the superior SDC method.*

Whereas the `LockIn Extension` was designed with the EPC 9 and EPC 10 in mind, the software can also be used in a straightforward manner with any patch clamp amplifier (see PATCHMASTER reference manual). Also, the discussion throughout the manual emphasizes acquisition of C_m data, however, estimates of the other two equivalent circuit parameters (G_s and G_m) accompany each C_m estimate when the SDC mode is used.

12.1.2 Applicable Sine Wave Frequencies

The range of sine wave frequencies that can be used for *LockIn* measurements depends on the equivalent circuitry resembling the measuring configuration.

Whole Cell, Frequencies up to 3 kHz: In a typical *Whole Cell* configuration sine wave frequencies in the range of 500 – 3000 Hz result in lowest noise recordings. In this frequency range the *Sine + DC* and *Piecewise Linear* method can be used.

On Cell, Frequencies up to 40 kHz: In *On Cell* configuration much higher frequencies can be used. In this configuration usually the *On Cell LockIn* method, that assumes a parallel network of a capacitor (membrane capacitance) and a resistor (membrane conductance), is used. The highest frequency that can be used with the *LockIn* depends on the specifications of the AD/DA converter board. E.g., when working with a sampling frequency of 200 kHz, a 40 kHz sine wave is resembled with 5 data points only. This is still enough and the *LockIn* method gives reliable results. A typical frequency range for *On Cell* capacitance measurements is 10 – 30 kHz.

12.1.3 Different Methods of Determining the Internal Phase Shift

When using a phase-sensitive detector for admittance measurements on an equivalent circuit it is always necessary to account for the phase shift introduced by the measuring instrument (system). This is usually done by measuring the phase of a well known component (e.g. a pure resistor or a pure capacitance). Alternatively, one can change the value of the resistor or capacitor (dithering) and adjust the phase that the change in the composite value (e.g. capacitance when resistance is dithered or conductance when the capacitance is dithered) becomes zero.

Measure the phase of a resistor (*Measured Calibration*): A pure resistor does not introduce any phase shift to the measured current when applying a sine wave voltage. That means, the measured phase shift between voltage and current origins for the measuring system itself. Please note that it is important that no remaining stray capacitances that might be associated with the resistor falsify the measurement. That becomes a very important issue at high sine wave frequencies, where the contribution of the capacitance to the overall admittance increases. At high frequencies phase determinations by measurements on capacitances is the preferable method.

Measure the phase of a capacitor: A pure capacitor introduces a 90° phase shift to the measured current. In order to determine the phase shift introduced by the measuring system, the measured phase has to be corrected by these 90°.

Capacitance Dithering: Especially at high sine wave frequencies the capacitive currents become quite large and bring the amplifier into saturation easily. In this case, introducing a small change in capacitance is often the favorable method.

Resistance Dithering: Not implemented.

Calculate the phase shift (*Calculated Calibration*): In case the measuring system is well-characterized it is possible to calculate the phase shift of the system depending on the system configuration (e.g. amplifier and filter settings). Please note that this method does account for the variability between different hardware devices of the same type. The accumulated tolerances of all relevant components might introduce a phase error of a few degrees. For a phase offset of the individual device can be corrected using the *Phase Shift* function in the *LockIn* configuration (see PATCHMASTER reference manual).

12.1.4 High time resolution measurement of C_m and low time resolution display

Different experimental conditions typically lead to different rates of C_m changes. For example, increases in C_m occur over many seconds to minutes when buffered Ca^{2+} solutions or other stimulatory substances are introduced into the cell through the patch pipette. Decreases in C_m reflecting endocytosis also often occur on a relatively slow time scale. On the other hand, high time resolution estimation of C_m is desirable to follow exocytosis immediately before and after a depolarizing pulse and is essential to record exocytosis evoked by flash photolysis of caged Ca^{2+}. Matching the acquisition rate to the expected rate of change in C_m can dramatically reduce storage requirements. In addition, the memory (RAM) that has been allocated by the PATCHMASTER software places constraints on how long high speed acquisition of C_m can occur without gaps.

> ***Note:*** *In* PATCHMASTER *all data are stored at high time resolution. The* Online Analysis *can be used to e.g. display averages of high time resolution data points versus time.*

High speed acquisition produces a C_m estimate for every sine wave cycle, i.e. typically 1000 points per second. This high rate of acquisition can be supported without interruption for the duration of a *Sweep*. The *Sweep* duration can easily be more than 60 seconds if the allocated buffer size is set appropriate (see `Memory Allocation` in the `General` tab of the Configuration window). For the display in the Oscilloscope window, typically *Continuous Redraw* mode is chosen for *Sweeps* exceeding a duration of one second. Acquisition of *Sweeps* with a duration of several seconds should be used if totally gap-free data are required. For most applications it is sufficient to repetitively acquire short *Sweeps* separated by small gaps. Here, we call this mode "low time resolution" measurements. This mode allows chart recorder style display of data derived from the *Sweeps* using the Online Analysis.

The raw data of each *Sweep* (*Imon*) must be stored in order to allow the C_m values to be recalculated offline. The high time resolution C_m values are stored e.g. as second *Trace*.

Low time resolution measurements is best performed by acquiring multiple *Sweeps* and using the Online Analysis of the PATCHMASTER software for display.

Here, data from a segment in a *Sweep* are averaged to result in one C_m point per *Sweep*. The C_m values (as well as the G_s and G_m values, if desired) are displayed by the Online Analysis. Typically, 100 sine wave cycles are output with each *Sweep* and a gap of about 100 ms occurs between *Sweeps*, resulting in an acquisition rate of about 5 Hz.

The two acquisition rates can be easily mixed during the course of an experiment. Consider the example of depolarization-evoked increases in C_m. *Sweeps* which contain depolarizing pulses and therefore contain voltage-gated currents such as I_{Ca} are displayed at high time resolution C_m in the Oscilloscope window. Before and after these *Sweeps* a series with short *Sweeps* can be acquired and displayed in the Online window. The linking and complete experiment control is performed via a protocol of the Protocol Editor.

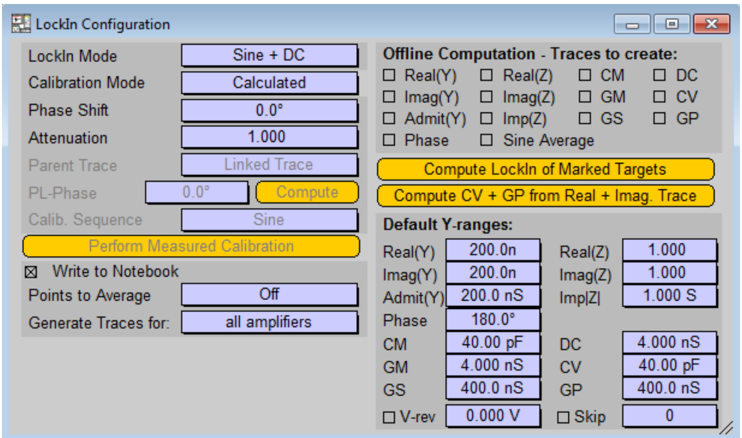
12.2 SDC versus PL mode

Here we want to compare the *Sine + DC* and the *Piecewise Linear* mode for capacitance measurements. The following description is in analogy to `Capacitance Measurements - Step by Step` (see PATCHMASTER reference manual) but provides further background information about *LockIn* measurements and differences of *Sine + DC* and *Piecewise Linear* modes.

12.2.1 Using LockIn in SDC mode

1. Activate the `LockIn Extension` of PATCHMASTER in the Configuration window that can be opened via the Windows menu. The LockIn Configuration window automatically opens.

2. Set the `LockIn Configuration` parameters as shown below:



3. Attach the MC-10 model circuit to the headstage of the EPC 10 and set the switch to the 10 MΩ position.

4. Activate the Amplifier window.

   (a) Press the *SETUP* button or perform an *Auto Vo* compensation.

   (b) Set the switch on the model circuit to the middle position.

   (c) Press the *SEAL* button or perform an *Auto C-fast* compensation.

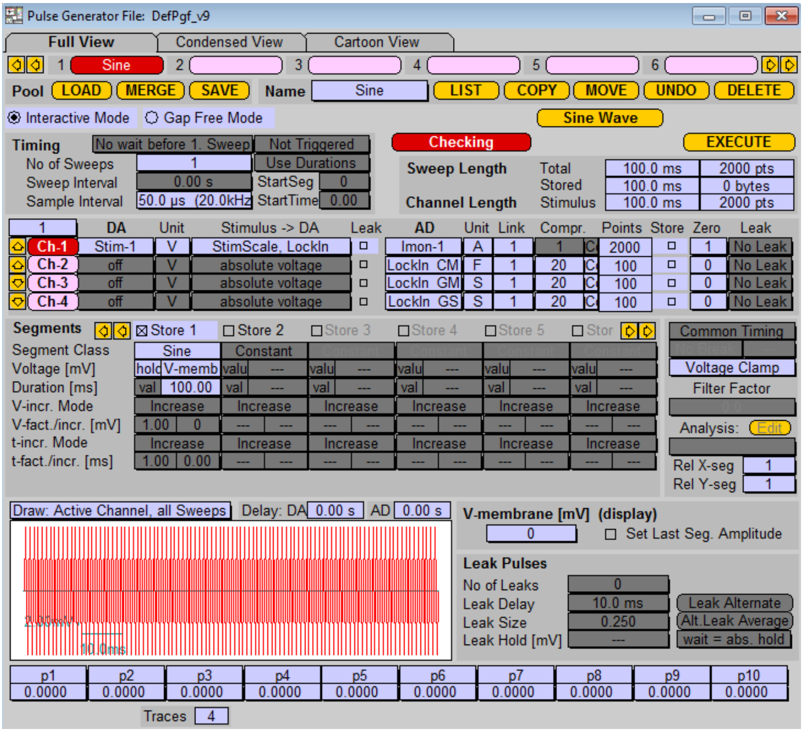
   (d) Set the switch on the model cell to the 0.5 GΩ position.

   (e) Make the following settings:

      - Set the *Gain* to *1.0 mV/pA*.
      - Set *V-membrane* to *-50 mV*.
      - Select *Imon2* for *Input ADC*.
      - Select *Whole Cell* mode in *Recording Mode*.
      - Set *Filter 1* to *Bessel 10 kHz*.
      - Set *Filter 2* to *Bessel 2.9 kHz*.
      - Set *Stim* to *20 μs*.
      - Disable *C-slow*, *Rs Comp*, *Leak Comp* and *Stim-External*

5. Make sure the Notebook window is open.

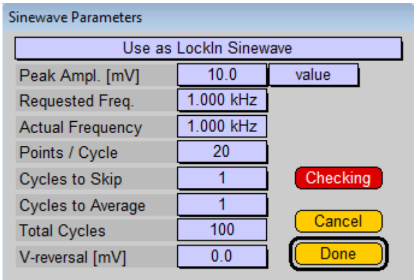
6. Create a simple sine wave *Series* in a PGF.



(a) Open the Pulse Generator window from the pull down menu (or press F8). Duplicate the following settings to create a *Series* named "Sine":

(b) Note that *Store* is deselected.

(c) Pressing the *Wave Parameter* button opens up a window, duplicate the following settings:



7. Execute the new *Sine Series* by pressing the corresponding button in the Control window. The *Series* outputs 100 cycles of a 1 kHz sine wave voltage, measures the resulting sinusoidal current, and outputs averaged estimates to the Notebook window. Typical values are:

```
Calculated LockIn Calibration:
Est. Int. Phase: 288.6   Est. Att.: 0.956
A:  67.00nS   B:  87.89nS   b:  2.007nS   Phase:  52.7
RS:  5.381MOhm   RM:  493.0MOhm   CM:  22.113pF
```

> ***Note:*** *These values are only written into the Notebook when the Write to Notebook option is enabled in the* LockIn Configuration *dialog.*

> ***Note:*** *These values were obtained with capacitance compensation "Off". Often, it is desirable to nullify the capacity current transients that accompany depolarizing pulses in order to prevent the amplifier from saturating.*

8. Select an *C-slow* range (e.g. 1000 pF) and perform an *Auto C-slow* compensation.

9. Execute the *Sine Series* again. The values output to the Notebook window are now slightly different. Typical values are:

```
A:  65.94nS   B:  90.43nS   b:  2.003nS   Phase:  53.9
RS:  5.159MOhm   RM:  494.2MOhm   CM:  22.041pF
```

Why are the values different? Note that the sinusoidal current displayed in the Oscilloscope window is very different depending on if *C-slow* compensation is *On* or *Off*. Use of *C-slow* compensation "nulls out" the bulk of the sinusoidal current just as it eliminates capacitive current transients elicited by voltage steps. However, the original "unnulled" current is needed to estimate the three equivalent circuit values. Therefore, the software adds back the nulled currents when processing the estimates (Gillis, 1995; 12.7 on page 211). *LockIn* can calculate what to add back because it always knows the *C-Slow* and *R-Series* instrument settings when an EPC 9 or EPC 10 is used. This process is generally quite accurate, however, it is not perfect. *C-slow* compensation and the model circuit may not behave ideally (neither, of course, do real cells), and small phase errors within *LockIn* can occur. Estimates generated by *LockIn* with *C-slow* compensation on can be thought of as a hybrid of estimates made by nulling current transients with *C-slow* circuitry and of estimates made with sine waves. The estimate from *C-slow* circuitry dominates the *LockIn* estimate when the bulk of the sinusoidal currents are nulled.

> ***Important note:*** *The LockIn estimates generated above are essentially identical to the C-slow and R-series settings. This situation has an important consequence: An Auto C-slow update can result in small jumps in C_m estimates generated by LockIn that have nothing to do with exocytosis or endocytosis.*

Once you are aware of this possibility, it seldom creates an actual problem. *Auto C-slow* updates should be performed during waiting periods between depolarizing pulses, **not immediately before, after or during pulses** (for example) and can be avoided entirely if C_m changes are evoked by stimulatory substances introduced into the cell through the patch pipette. This is only an issue for low time resolution acquisition.

> ***Note:*** *High time resolution measurements of changes in C_m are not affected by C-slow updates immediately before or after the Sweep.*

10. Set *C-slow* = 17.0 pF, *R-series* = 7 MΩ, Range = 1000 pF

    Execute the *Sine Series* again. Typical values sent to the Notebook window are:

    ```
    A:  66.63nS   B:  90.07nS   b:  2.004nS   Phase:  53.5
    RS:  5.204MOhm   RM:  493.9MOhm   CM:  22.175pF
    ```

    > ***Note:*** *Even though both compensated values are in error by more than 30 %, the values calculated by LockIn are within 1 % of the values obtained with correct compensation.*

12.2.2 Using LockIn in PL mode

Usually, in the *Piecewise Linear* mode the *C-slow* compensation is to eliminate the bulk of the sinusoidal current which results from the sinusoidal voltage stimulus. The residual current is processed with a phase sensitive detector set to an appropriate phase angle to produce an output signal which is linearly proportional to small changes in C_m.

The most common way to set the phase is to induce a change in series resistance and then calculate the phase setting which produces an output which is insensitive to this change ("phase tracking", Fidler and Fernandez, 1989; 12.7 on page 211). This method, however, can produce phase errors because the pipette capacitance is neglected in the analysis (Gillis, 1995; 12.7 on page 211). Calibration is usually performed by offsetting *C-slow* compensation by a defined amount (thus simulating a change in C_m) and noting the change in output of the phase sensitive detector.

LockIn avoids errors introduced by phase tracking by calculating the *PL-Phase* from *C-slow*, *G-series* and an estimate of G_m which is generated during *Auto C-slow* compensation according to:

$$PL - Phase = arctan(\frac{\omega * C-slow}{G_m}) - 2 * arctan(\frac{\omega * C-slow}{(G_m + G_s)})$$

In principle, the use of this phase leads to a ΔC_m signal which is insensitive to changes in G_s and only mildly sensitive to changes of G_m under common recording conditions (Gillis, 1995; Joshi and Fernandez, 1988; 12.7

on page 211). If a patch clamp amplifier other than an EPC 9 or EPC 10 is used, then G_m is assumed to be zero. In this case the phase leads to a ΔC_m signal which is insensitive to changes in G_m and mildly sensitive to changes of G_s (Gillis, 1995; Joshi and Fernandez, 1988; 12.7 on page 211).

> ***Note:*** *The use of a calculated phase requires that the phase shift introduced by the recording instrumentation be known. This is not a problem for LockIn, because phase calibration is an integral part of the software.*

Calibration of the output *Traces* is also automatically performed by *LockIn* using the formulas given by Neher and Marty (1982; 12.7 on page 211). Thus displacement of the *C-slow* setting is not necessary for calibration, but can be used to test the calibration and phase setting as illustrated by the following example:

1. Follow steps 1. - 6. from the example given before (12.2 on page 170). We will use the *Sine Series* for our example.

2. Select an *C-slow* range (e.g. 1000 pF) and perform an *Auto C-slow* compensation. The *C-slow* reads 22.26 pF.

3. Open the *LockIn Configuration* window and toggle the *LockIn Mode* from *Sine + DC* to *Piecewise Linear*. Type in "Sine" as the *Calib. Sequence*. Press the *Compute* button in the *PL-Phase* line. This prints the *PL-Phase* value to the Notebook window. The value should be approximately 287°.

4. Execute the *Sine Series*. In our case, the following values were printed to the Notebook:

```
Calculated LockIn Calibration:
Est. Int. Phase:  288.6   Est. Att.:  0.956
A:  766.0pS   B:  -217.0pS   Phase:  344.2
dGS:  -2.217nS  dGM:  1.186pS  dCM:  -53.46fF
```

5. In our example, we decrease the *C-slow* setting by 0.1 pF (in our case to 22.16 pF) to simulate an *increase* in C_m. Execute the "Sine" *Series* again. We got the following values:

```
Calculated LockIn Calibration:
A:  766.0pS   B:  200.5pS   Phase:  14.7
dGS:  -2.218nS  dGM:  1.187pS  dCM:  49.40fF
```

The difference in the "dCM" values is 102.86 fF, which indicates that there is about a 3 % discrepancy between the calculated calibration of the ΔC_m *Trace* and the calibration indicated by displacement of the *C-slow* setting.

Ideally, no changes in the "dGS" or "dGM" values should result from the *C-slow* offset. How do we evaluate if the actual changes suggest a problem in the setting of PL-Phase? The "Real" part is multiplied by two different scaling factors to give the ΔG_s and ΔG_m values. The "Imaginary" part is scaled to give the ΔC_m value. Note that the "Imaginary" changes much more than the "Real" upon displacement of the *C-slow* value. A phase discrepancy between the calculated *PL-Phase* and the phase suggested by *C-slow* displacement is given by: tan^{-1} (ΔRe/ΔImg).

We can also test the *PL* mode by displacing the *R-series* setting:

6. Set *R-series* to a rounded value. In our case we used 5.4 MΩ . Note that the value of *R-series* indicated in the Amplifier window is only displayed to 2 decimal places of precision. The actual value may be slightly different. Execute *Sine*. We got the following values:

```
Calculated LockIn Calibration:
A:  -1.451nS   B:  259.7pS   Phase:  169.9
dGS:  4.198nS  dGM:  -2.246nS  dCM:  63.96fF
```

7. Now set *R-series* to 5.5 MΩ and execute *Sine* again. We got:

```
Calculated LockIn Calibration:
A:  -2.680nS   B:  327.5pS   Phase:  173.0
dGS:  7.755nS  dGM:  -4.148nS  dCM:  80.68fF
```

Note that "dCM" changed by +16.72 fF whereas "dGS" changed by 3.557 nS. The displacement of *R-series* simulates a change in G_s of 3.557 nS. Note that "dGM" changed by 1.902 nS. This represents the change in G_m that would have to occur to result in the same change in admittance as the displacement in the *R-series* setting.

> ***Note:*** *In the PL mode, it is impossible to distinguish if a change in a "ΔG" Trace originates from an actual change in G_m or G_s.*

In most respects the *PL* mode can be used similarly as the *Sine + DC* mode as illustrated by the examples in the previous sections. Values labeled as "Cm", "Gm" and "Gs" should now be interpreted as "ΔCm", "ΔGm" and "ΔGs". In order for the *PL-Phase* to remain valid, it is important to perform *Auto C-slow* updates followed by execution of *Compute* periodically (say, once per minute).

> ***Important note:*** *Both of these operations can result in jumps in the online display of ΔCm which are unrelated to exocytosis or endocytosis.*

12.3 Using LockIn in On Cell mode

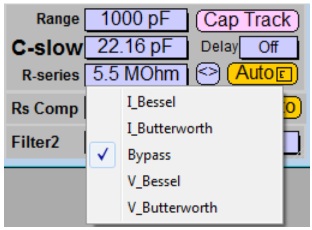
In *On Cell* mode the software lock-in returns the following parameters:

- $\mathtt{C}_m = \mathrm{Im}/(2\pi\ \mathrm{f})$
- $\mathtt{G}_m = \mathrm{Re}$
- $\mathtt{G}_s = 0$

The compensated *C-fast* capacitance will not be added back to the C_m value (see `Using LockIn in Sine + DC mode`, 12.2 on page 170).

12.3.1 Measurements using high frequency sine waves

For measurements of kinetics or small capacitance changes in *On Cell* mode it is often desired to measure with sine wave frequencies in the range of several kHz. At frequencies that are far above the cut-off frequency of *Filter 2*, it might be required to bypass *Filter 2*.

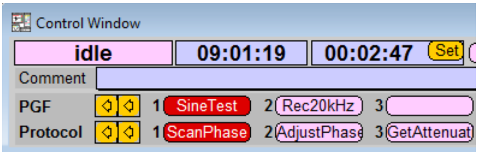


- If you use EPC 10 amplifier with revision "N" or later you can just use the *Filter 2 Bypass*, which can be selected as *Filter 2* type.
- In case of older amplifiers the current signal of the active amplifier after *Filter 1* can be recorded from the *MUX* channel. All *LockIn* results can then be derived form the *MUX* channel instead from *Imon-2*.

12.3.2 Finding the appropriate Phase and Attenuation settings

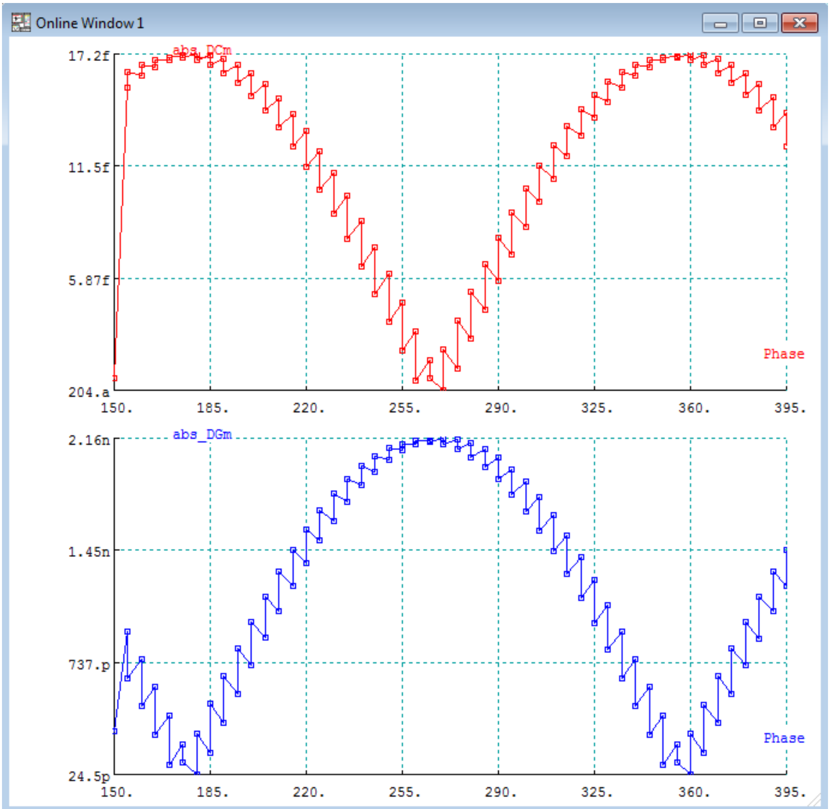
When working with sine wave frequencies larger of 10 kHz it is advisable to use the *Manual Calibration Mode* of the *LockIn* and determine the correct phase and attenuation before the measurement.

If you are in a stable *On Cell* recording configuration you can use the *C-fast* compensation to introduce small changes in the measured capacitance. This dithering of the capacitance can be used to determine the correct phase setting for the software lock-in in this recording configuration. In our example configuration we provide two protocols named *ScanPhase* and *AdjustPhase*, which both help to semi-automatically determine the phase setting.



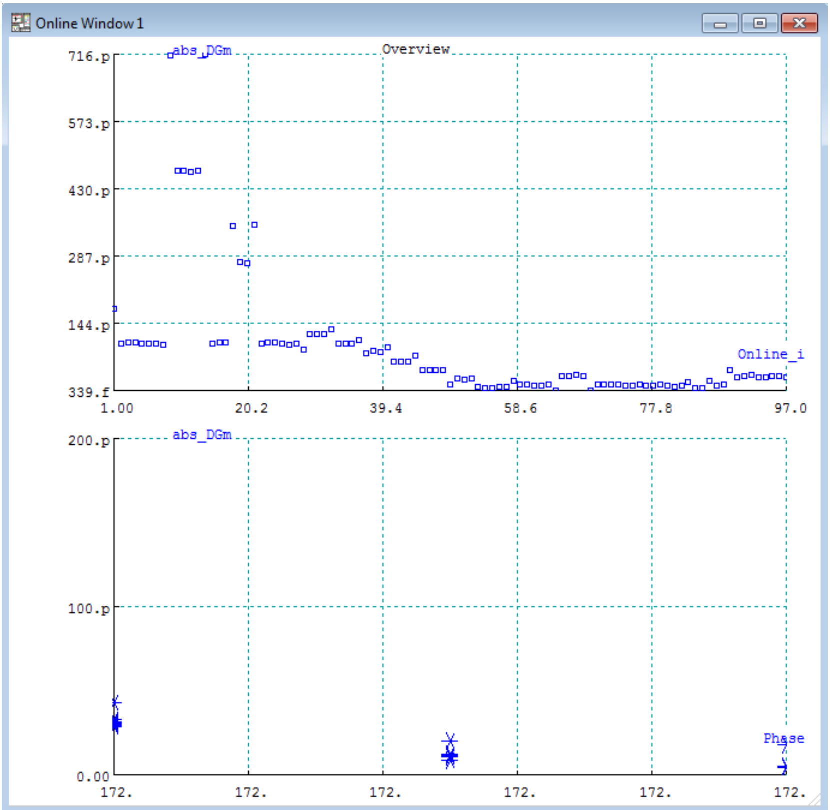
> ***Important note:*** *Before you start with further reading please go to our homepage (`http://www.heka.com/support/tuto.html`) and download the demo configuration `On Cell Cm measurements using the EPC 10`. Inside the `*.zip` file you will find an instruction how to set up* PATCHMASTER.

ScanPhase: Start the protocol "ScanPhase". From a starting phase of 150° the phase is increased in 5° steps and the ΔG_m and ΔC_m is measured upon toggling the *C-fast* compensation by 50 fF. The ΔC_m values are displayed in the upper graph and the ΔG_m values in the lower graph. The correct phase is reached if ΔC_m goes through its maximum and ΔG_m through its minimum. When the measurement passed the correct phase setting, terminate the protocol with the F12 key. The phase at minimum of ΔG_m will be entered as *Phase Shift* in the *LockIn* configuration.



In the Online Analysis window you see the results of a phase scan from 150° to 395° . The correct phase is reached if the ΔG_m becomes a minimum, in this case at around 170°.

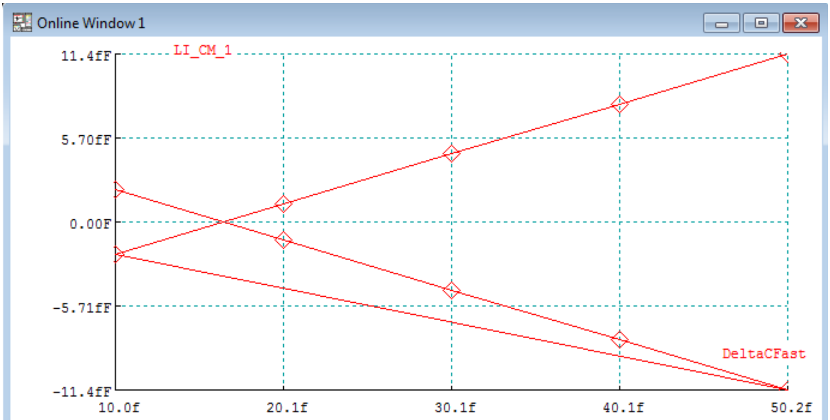
AdjustPhase: Fine tune the *Phase Shift* with the protocol "AdjustPhase". As starting phase angle the last entry in the *LockIn* configuration is prompted. Press "OK". The ΔG_m values are displayed in two graphs. The upper graph provides an overview and the lower graph displays the ΔG_m value at higher resolution. The second graph will be wiped every 20 measurements. Use the keys 1 and 2 to increment and decrement the phase by 10°, respectively. Use the keys 3 and 4 to increment and decrement the phase by 0.5°, respectively. Terminate the phase adjustment by pressing the F12 key.



In the Online Analysis window you see the results of a phase scan from 170° to 173°. The correct phase is reached if the ΔG_m becomes a minimum, in this case at around 172.5°. This phase setting should be entered in the LockIn Configuration window (*Phase Shift*).

GetAttenuation: At the the given phase angle we now calibrate the amplitude of our C_m measurements. As calibration reference we use the *C-fast* compensation of the EPC 10. Start the protocol "GetAttenuation". The

C-fast compensation will be reduced starting from its auto compensation value by 10 fF. Then the C_m is measured with the *LockIn* and plotted against the difference in *C-fast* compensation. This cycle is repeated 5 times and the slope is calculated by linear regression. The slope is the scaling factor for the amplitude (ratio of measured / expected capacitance change) and is set as *Attenuation* in the *LockIn* configuration.



In our example the `Attenuation` is calculated to be 0.3462. This means that the detected sine wave signal is attenuated at the frequency of 20 kHz by the stimulus filter and the current filter to about 34 % of its specified amplitude.

With the correct *Phase* and *Attenuation* settings the software lock-in is ready to use. In the following example we are recording the capacitance from a 0.8 pF capacitor of the EPC 10 at a sine wave frequency of 20 kHz and a sine amplitude of 200 mV. In order to reduce the noise in the C_m *Trace* we average the results of 1000 sine wave cycles.

The standard deviation of the C_m *Trace* is 2.6 aF (atto-Farad) with a time resolution of one point every 50 ms.

12.4 Performing a Measured Calibration

1. Create a basic *Series* (we will call it *Sine*) with values for *Actual Frequency* and *Points/Cycle* that you will use for all your experimental *Series*. Specify a sine segment with a duration of at least 100 ms. The *Peak Ampl.* of the sine wave should be something like 20 mV (1 mV if you are calibrating for the high gain range: $\geq$ 50 mV/pA).

2. In the EPC 9/10 Amplifier window, set *Filter 1*, *Filter 2*, *Stim* time constant, and *Gain* range to values that you will use during acquisition. *C-slow* compensation should be *Off*.

3. We have found that the 10 MΩ resistor built into the MC-10 model circuit works fine for calibration for frequencies up to about 2 kHz. Attach the MC-10 to the head stage.

4. Set the switch to the middle position and perform an *Auto C-fast* compensation. Then set the switch to the "10 M" position.

5. Perform an *Auto Voffset* compensation. Then set *V-membrane* to something like -40 mV (-1 mV if you are calibrating for the high gain range: $\geq$ 50 mV/pA).

6. Open the LockIn Calibration window from the Windows menu and set the *Calibration Mode* to *Measured*. The *LockIn Mode* should be *Sine + DC*. Make sure the *Write to Notebook* box is checked. Type in the name of your test *Series* (*Sine*) in the *Calib. Sequence* box.

7. Press the *Perform Measured Calibration* button.

8. The Phase and Attenuation that were found are printed to the Notebook window and internally stored. The measured calibration is finished.

Now you can switch the MC-10 to "0.5 G" for testing the new measured calibration and execute the *Sine Series*. The values sent to the Notebook window should be something like RS: 5.2 MΩ RM: 510 MΩ CM: 22 pF.

Note: *To compute the PL-Phase or to perform the Measured Calibration, you have to store at least one sine segment, otherwise no capacity measurement and thus no calibration is possible.*

12.5 Tips on the use of the LockIn

12.5.1 Recommended parameter settings for LockIn measurements

Frequency of the sinusoid (f_c): The C_m noise increases as f_c exceeds the "break frequency" (f_b) defined as: $[2\pi C_m/(G_s + G_m)]^{-1}$. Very low values of f_c (say, below $f_b/4$) also result in noisy C_m estimates (Gillis, 1995; 12.7 on page 211) for details. Due to the non-ideal nature of C_m estimation, different stimulus frequencies can lead to slightly different values of C_m. Therefore, once a value of f_c is selected, it is best not to change it during the course of the experiment (i.e. all *Series* used should have the same *Actual Frequency* in the LockIn Parameters window).

Number of input points per sinusoid: Generally, a value of 8 or great is advised in order to minimize aliasing noise of C_m estimates (Gillis, 1995; 12.7 on page 211). If long *Sweeps* are desired, 8 *Points / Cycle* is sufficient if *Filter 2* is set to twice the frequency of the sinusoid.

Amplitude of the sinusoid: Larger amplitudes result in lower C_m noise, but the activation of voltage-dependent conductances in excitable cells should be avoided. For excitable cells, setting *V-membr.* as hyperpolarized as the cell will tolerate allows a larger amplitude to be used. Very small amplitudes (say, below 2 mV) can cause quantization problems, particularly if the number of output *Points / Cycle* is small.

V-reversal: A value of "0.0" can be used if G_m is small and the actual reversal potential is unknown. If you expect a significant membrane conductance to be activated during the course of the experiment, set *V-reversal* to the zero current potential of the activated conductance. In principle, this value can be found during pilot experiments by:

1. Stepping the membrane potential by about +/- 20 mV about *V-membrane*

2. Measuring the steady-state currents at the two potentials and

3. Extrapolating these values to the zero current potential.

Gain of the patch clamp amplifier: In principle, the larger value of feedback resistor used in the high gain range ($\geq$ 50 mV/pA in the EPC 7/8/9/10) results in lower C_m noise. However, the improvement is often negligible for typical circuit parameters and sinusoidal frequencies (Gillis, 1995; 12.7 on page 211). High gain makes *Measured Calibration* difficult and can easily lead to amplifier saturation. Typically of gain of 1-10 mV/pA is used.

Filter 1 setting (Epc 9 and Epc 10): The 10 kHz *Bessel* filter is usually an appropriate setting.

Filter 2 setting (Epc 9 and Epc 10):. A setting of (at least) twice the stimulus frequency ensures that estimates generated for each sinusoidal cycle are independent. Higher values than this can lead to aliasing noise unless the number of input *Points / Cycle* is high. The *Bessel* filter type is preferable if *Calculated Calibration* is used because the circuitry appears to follow the theoretical characteristics better than the *Butterworth* filter type.

Stimulus filtering: It has been customary with software lock-in amplifiers to filter the stimulus D/A signal to produce a sine wave with a "smooth" appearance (i.e. remove the harmonics). However, this is not actually necessary, because the harmonics are automatically removed by the software lock-in algorithm. The 20 μs time constant filter built into the EPC 7/8/9/10 sufficiently smoothes the D/A signal to prevent amplifier saturation.

Capacitance compensation: The use of capacitance compensation is quite convenient to eliminate the current transients which result from depolarizing steps. However, there is an additional benefit. C_m noise appears to be slightly less when *C-slow* compensation in used in an EPC 9 and EPC 10, presumably because it cancels some of the noise present in the stimulus pathway (Gillis, 1995; 12.7 on page 211). Testing of *LockIn* is best performed with capacitance compensation *Off*, because *LockIn* estimates are essentially identical to *C-slow* and *R-series* values when the bulk of the sinusoidal current has been nulled out. See `Using LockIn in Sine + DC mode`, 12.2 on page 170.

Series resistance compensation: The use of series resistance compensation can actually reduce C_m noise if a value of f_c is used which approaches or exceeds the break frequency (f_b) defined above (12.5.1 on page 186). This is because the circuitry will boost the amplitude of the stimulus sinusoid to partially compensate for the drop in voltage across G_s. Since a component of the current monitor signal is fed back to the stimulus, however, the C_m *Trace* can become noisier if a high percentage of *R-series* compensation is used. The effect of series resistance compensation on the current signal is accounted for by *LockIn* when an EPC 9 or EPC 10 is used. This correction, however, is inexact – particularly for large fractional compensation. Perhaps it is best that series resistance compensation only be used when it is needed.

12.5.2 Parameters which affect admittance measurements

1. Scaling parameters
   - Gain (trans-impedance) of the patch clamp amplifier
   - Amplitude of sine wave stimulus
2. "Critical parameters" which determine phase shifts (delays) and attenuation introduced by the patch clamp amplifier and the sine wave generator
   - Number of *Points / Cycle* output by the DAC in generating the sine wave
   - Sine wave frequency
   - Stimulus filtering (EPC 7/8/9/10: $t_s = 2\ \mu s$ or $20\ \mu s$)
   - Low-Pass filters present in the current monitor pathway. For the EPC 9 and EPC 10, this consists of:

   - Filter 1 (F_1 = 10 kHz *Bessel*, 30 kHz *Bessel*, *HQ* 30 kHz, or 100 kHz *Bessel*)
   - Filter 2 (F_2 = Bandwidth + *Bessel* or *Butterworth*)
- Feedback resistor of the patch clamp amplifier head stage (for the EPC 9 and EPC 10: R_f = 5 MΩ for low gain range; 500 MΩ for medium range; 50 GΩ for high gain range)

3. "Compensation" parameters
   - Capacitance compensation (*C-slow, G-series*)
   - Series resistance compensation. *LockIn* only supports this for EPC 9 and EPC 10 amplifiers.
   - Leak subtraction (software or hardware). *LockIn* does not support hardware leak subtraction.

12.5.3 Miscellaneous Tips

Calibrate the patch clamp amplifier gain with an external resistor: The automatic calibration of the EPC 9/10 performed by the internal calibration procedure is not exact. The gain can be fine tuned with an external resistor using a new feature of the calibration routine implemented in the PATCHMASTER software. The 10 MΩ resistor in the MC-10 model circuit can be used since it is accurate to within 1 %. The corrections are made in an *External Gain Calibration* table in the `Test and Calibrate` pull-down menu of the `EPC 10` menu and result in a change in the `SCALE.EPC` file. The updated file must be copied to the PATCHMASTER folder.

Digital filtering of high time resolution C_m values: High time resolution estimation of C_m occurs at the maximum rate that generates independent values – one point per cycle. Estimates generated at this rate, however, tend to be noisy. Often the user is willing to trade off some of the time resolution for lower noise estimates. This can be done by filtering the data using the digital filter in the Oscilloscope window. With the digital filter a specific *Trace* can be filtered for display and export.

12.6 Examples

12.6.1 Recording depolarization-evoked increases in Cm

The general design of the setup is that a protocol controls the experimental run. The protocol calls PGF sequences and the PGF sequences call the corresponding *Online Analysis* methods. The *Analysis Methods* finally, draw the results into the online graphs.

> ***Note:*** *The subsequent demo configuration can be downloaded from our homepage:* *http://www.heka.com/support/tuto.html*. *Inside the* ***.zip** *file you will find an instruction how to set up* PATCHMASTER.

12.6.1.1 The PGF Sequences "Sine" and "Depol"

The intention of our experiment is to measure LockIn parameters continuously over time with the capability to stimulate the cells with depolarizing pulses to provoke capacitance changes. For this purpose we will create two PGF sequences.

Sine

We create a new PGF sequence, name it "Sine" and make the following main settings:

- No of Sweeps: 1000
- Sample Interval: 50 μs
- DA channel: Stim-DA – StimScale, LockIn
- AD channels:
  1. Imon2
  2. LockIn_CM

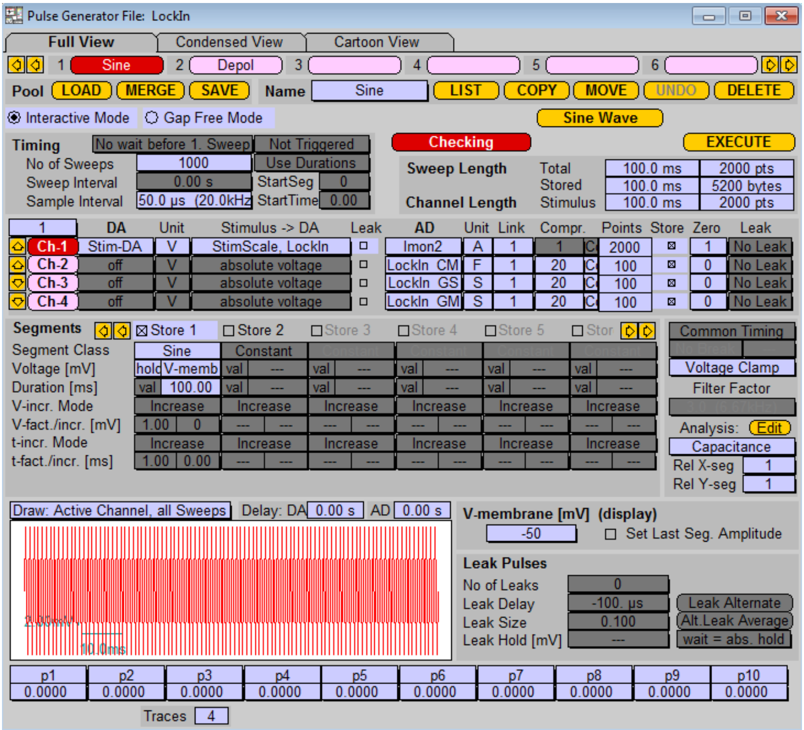
3. LockIn_GS
4. LockIn_GM

- Segments:

1. Sine – holding, 100 ms

- Analysis Method: Capacitance

The screenshot below gives you an overview of the created "Sine" PGF sequence:

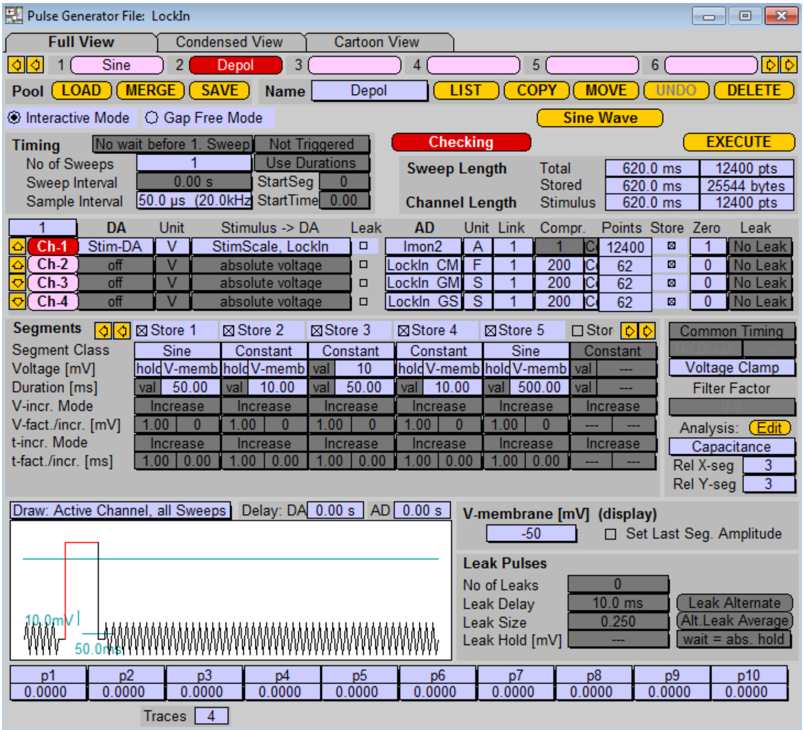


Depol

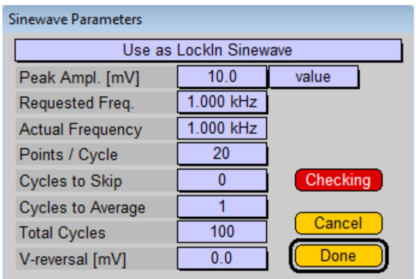
We create a new PGF sequence, name it "Depol" and make the following main settings:

- No of Sweeps: 1
- Sample Interval: 50 μs
- DA channel: Stim-DA – StimScale, LockIn
- AD channels:
  1. Imon2
  2. LockIn_CM
  3. LockIn_GS
  4. LockIn_GM
- Segments:
  1. Sine – holding, 50 ms
  2. Constant – holding, 10 ms
  3. Constant – +10 mV, 50 ms
  4. Constant – holding, 10 ms
  5. Sine – holding, 500 ms
- Analysis Method: Capacitance

The screenshot below gives you an overview of the created "Depol" PGF sequence:



Make sure that all *Wave Parameter* settings are identical for "Sine" and "Depol":



In our example we set the sine wave frequency to 1 kHz. This and of course other parameters can be modified by the user according to experimental needs.

12.6.1.2 The LockIn Protocol

We setup a protocol based on the "Chart" protocol used in the chapter `Chart Recording` and rename it into "LockIn".

```
IF: Break "Press "F1" to run a "ramp" or "F2" to run an "IV" sequence"
   BREAK: protocol
END_IF
;Main Settings
Command: "  O  Wipe"
File: NewGroup
Acquire: Wipe=OFF,
SetOsci: Timer, Tr(Y)= 1111111111111111
Online: Auto
; Main experimental loop
GOTO_MARK: "Chart"
REPEAT: sweeps 1.000s
   Amplifier: C-slow
   Sweep: "Testpulse"
   IF: Key = Help
      ClearKey
      GOTO: "Stimulus_1"
   ELSIF: Key = F2
      ClearKey
      GOTO: "Stimulus_2"
   ELSIF: Key = F12
      ClearKey
      GOTO: "End"
   END_IF
END_REPEAT
;Special actions
GOTO_MARK: "Stimulus_1"
Series: "Stim_Ramp"
GOTO: "Chart"
GOTO_MARK: "Stimulus_2"
Series: "Stim_IV"
GOTO: "Chart"
GOTO_MARK: "End"
;Change X-axis scaling and redraw Online Analysis
Online: "Whole Results"
Replay: Group
```

For a general description please read the chapter `Chart Recording`. Here we focus on the differences.

At the beginning of the experiment we change the text of the *IF...THEN* event. Furthermore we add an *Auto C-Slow* compensation.

```
IF              ( 0.000s): Break "Press "F1" to run "Depol".
                           "F12" ends the protocol."
   BREAK           ( 0.000s): protocol
END_IF
;Main Settings
Command         ( 0.000s): "  0  Wipe"
File            ( 0.000s): NewGroup
Acquire         ( 0.000s): Wipe=OFF,
SetOsci         ( 0.000s): Timer, Tr(Y)= 1111111111111111
Online          ( 0.000s): Auto
Amplifier       ( 0.000s): C-slow
```

In the main experimental loop we will execute a *Sweep* of the sequence "Sine" every second. The *GOTO_MARK* is renamed into "Cap".

```
; Main experimental loop
GOTO_MARK       ( 0.000s): "Cap"
REPEAT          ( 0.000s): sweeps 1.000s
   Amplifier       ( 0.000s): C-slow
   Sweep           ( 0.000s): "Sine", "", ""
```

If key F1 (Help) is hit, we jump to the *GOTO_Mark* "Stimulus_1" which is located in the special actions section. Key F12 jumps to the *GOTO_MARK* "End" to enter the postfix section of the protocol to terminate the experiment.

```
IF              ( 0.000s): Key = Help
      ClearKey        ( 0.000s)
      GOTO            ( 0.000s): "Stimulus_1"
   ELSIF           ( 0.000s): Key = F12
      ClearKey        ( 0.000s)
      GOTO            ( 0.000s): "End"
   END_IF
```

Upon jumping to one of the *GOTO_MARKs* the acquisition of a *Series* is executed. In our example when jumping to "Stimulus_1" the *Series* "Depol" is executed and then we return to the mark "Cap" to reenter the experimental main loop.

```
;Special actions
GOTO_MARK       ( 0.000s): "Stimulus_1"
```

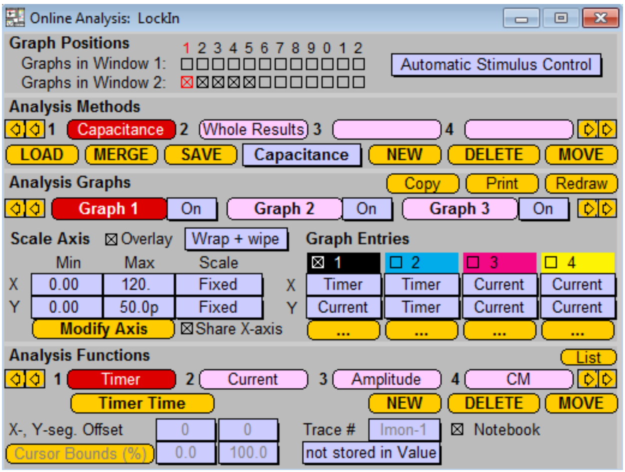
```
Series          ( 0.000s): "Depol", "", ""
GOTO            ( 0.000s): "Cap"
GOTO_MARK       ( 0.000s): "End"
```

At the end of the experiment we would like to display the online data of the whole experiment in the Online Window 2. Therefore, we change the X-axis scaling of the *Analysis Method* and turn off the *Wrap* option. Then, we replay the whole experiment.

```
Online          ( 0.000s): "Whole Results"
Replay          ( 0.000s): Group
```

12.6.1.3 The Online Analysis

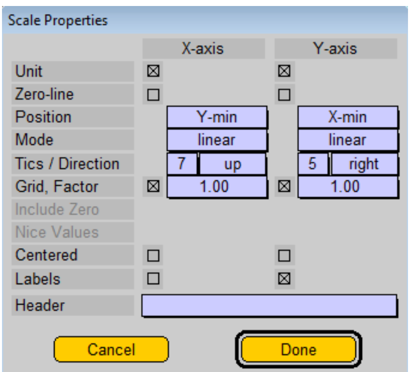
The *Analysis Method* is based on the "Baseline" method described in the `Chart Recording` tutorial. We rename into "Capacitance" and enable five *Analysis Graphs* in Online Window 2.



1. Graph:
    - Timer vs. Current (Mean)
2. Graph:
    - Timer vs. Amplitude
3. Graph:
    - Timer vs. C_M
4. Graph:
    - Timer vs. G_S
5. Graph:
    - Timer vs. G_M

For all graphs we have the following settings:

- Overlay
- Wrap + wipe
- Fixed Y- and X-axis scaling
- Share X-axis
- Scaling Properties:



For the "Whole Results" *Analysis Method* copy the *Analysis Method* "Capacitance" and make the following modifications:

- No Wrap
- X-axis scaling: Auto after a Series

12.6.2 Depolarizing pulses given at regular intervals

With the type of acquisition illustrated in `Recording depolarization-evoked increases in Cm`, 12.6.1 on page 190, depolarizing pulses are triggered at will by a user keystroke. It is often desirable, however, to have the pulses triggered automatically at regular intervals (e.g. once per minute) without relying on a manual input. One way to do this is to reduce the *No of Sweeps* in the "Sine" sequence to a value that *No of Sweeps* * *Duration* in the *Acquire Each Sweep* event are about the inter pulse duration.

For example, we execute the "Sine" sequence once a second. With *No of Sweeps* set to "9", we will yield about 9 seconds duration between two executions of the sequence "Depol". Since the execution of "Depol" itself takes also between 0.1 and 1 second, we come up with an overall repetition rate of about once per ten seconds.

Another method is to control the inter pulse duration from within the protocol. For example you can insert a section (If Timer MOD 10 seconds) as follows in the main loop:

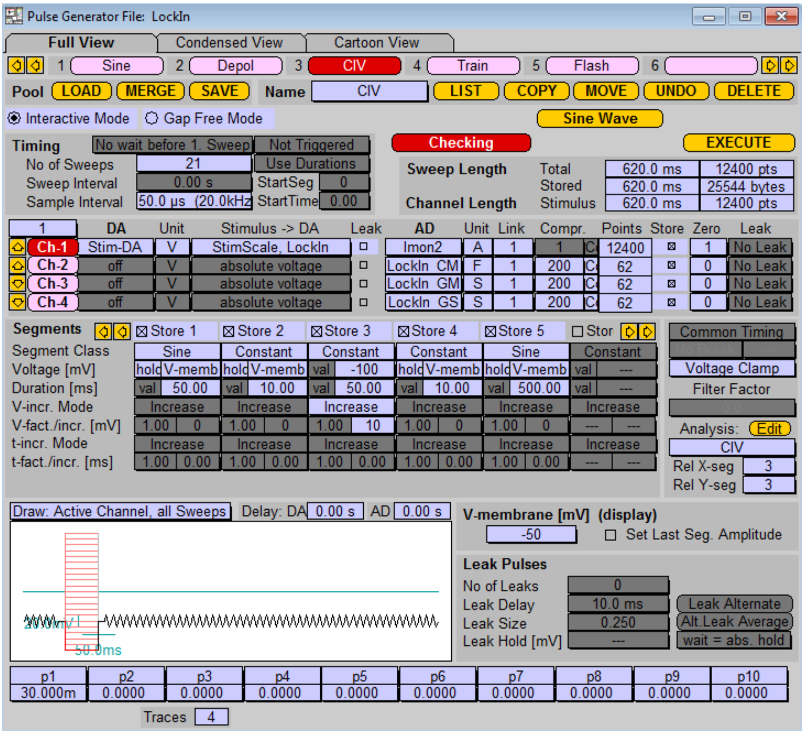
```
IF                ( 0.000s): Key = Help
      ClearKey          ( 0.000s)
      GOTO              ( 0.000s): "Stimulus_1"
   ELSIF             ( 0.000s): Timer MOD  10.000
      GOTO              ( 0.000s): "Stimulus_1"
   ELSIF             ( 0.000s): Key = F12
      ClearKey          ( 0.000s)
      GOTO              ( 0.000s): "End"
   END_IF
```

The *Timer* function can be selected from the parameter list at the *Left Source* of the *IF* statement.

12.6.3 Generating C-I-V curves

In a next step we would like to add some advanced *Analysis Methods* to our procedure. Up to now we are plotting *Current*, *Amplitude*, C_M, G_M and G_S versus time in the Online Window 2. Instead of executing a single depolarizing pulse we would like to generate a plot of ΔC_M versus current and current versus voltage (an IV curve with capacitance measurements).

First, we create a PGF sequence "CIV" based on the sequence "Depol". We use the *V-incr* to depolarize to different potentials. We adjust the *No of Sweeps* and all durations to fit our experimental design. Last, we change the name of the assigned Online Analysis to "CIV".



On basis of the *Analysis Method* "Capacitance" we create a new one with the name "CIV". Now we have to add three additional *Analysis Functions*:

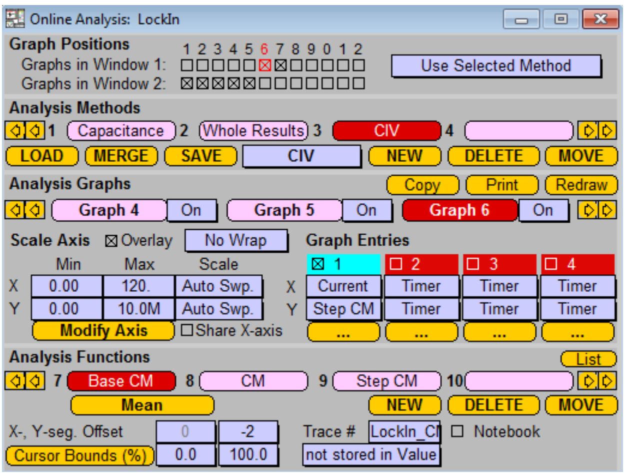
1. Base CM
2. CM
3. Step CM

Base CM and *CM* are two functions of the type *Mean*. They work on the second *Trace* (*LockIn_CM Trace*). *Base C_M* measures the mean of C_M before the stimulus (segment offset "-2") and C_m the one after the stimulus (segment offset +2). *Step C_M* is of the function type *a-b* and is used to subtract *Base C_M* from C_M.

Now we create two additional graphs that we place in Online Window 1:

- Graph 6 shows *Step CM* versus *Current*.
- Graph 7 shows *Current* versus *Amplitude*.

For both graphs we check *Overlay* and use *Auto Scaling* on both axis.



We also create a new protocol on basis of the "Capacitance" protocol and name it "CIV". In the section "Special actions" we add two *Online* events to wipe the graphs 6 and 7 and execute the *Series* "CIV" afterwards. Then the protocol will return to the main loop.

```
;Special actions
GOTO_MARK        ( 0.000s): "Stimulus_1"
Online           ( 0.000s): Wipe-6
Online           ( 0.000s): Wipe-7
Series           ( 0.000s): "CIV", "", ""
GOTO             ( 0.000s): "Cap"
GOTO_MARK        ( 0.000s): "End"
```

In case we want to generate slightly different graphs, e.g. a plot of Δ C_M versus duration, then just create the appropriate *Analysis Functions* and create an additional graph or replace the X or Y identifiers in previously designed graphs.

12.6.4 Trains of depolarizing pulses

Here we see how to create trains of depolarizing pulses with short interpulse intervals to look at depolarization-induced changes in C_M.

Very short interpulse intervals can be created by having a train correspond to a single *Sweep*. In this case *Constant* segments which produce depolarizing pulses are separated by *Sine* segments at *V-membr*. The maximum allowable number of points per *Sweep* limits the number of pulses that can be applied without a gap. However, this limit is very high and might not be a real limit for most applications.

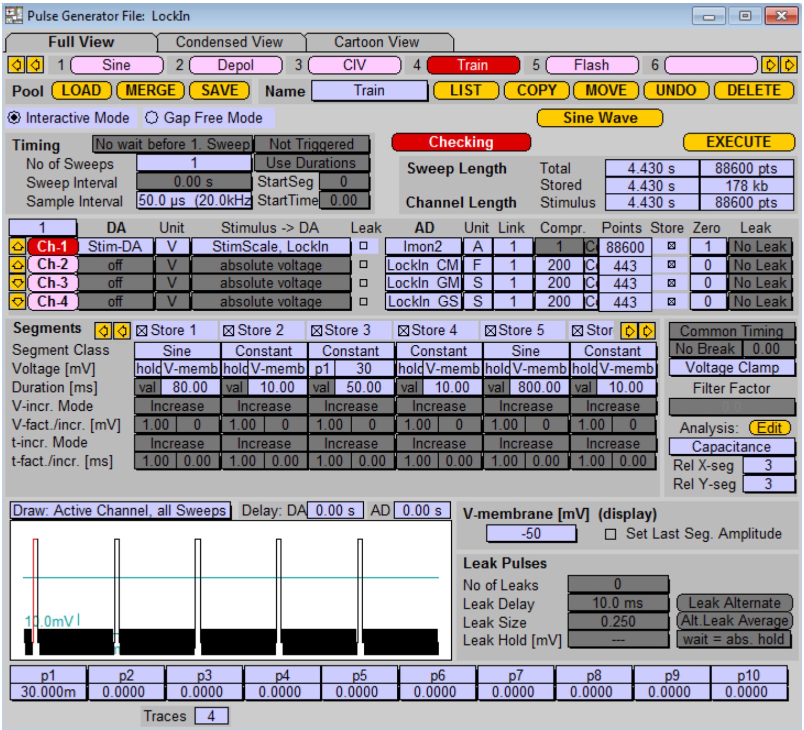
> ***Important note:*** *Before starting to work with long Sweeps in the* Pulse Generator *you should try to increase the Max. Sample Points in the* Configuration*. Usually you can easily increase this value by a factor of 10 compared to the default settings. This allows you to create Sweeps of several seconds length, sufficient for most applications.*

The *PGF parameters* and the functions *Duplicate...* let the user comfortable edit a *Sweep* with multiple stimulation pulses.

The other approach is to use multiple *Sweeps* to produce a train. Whereas a virtually unlimited number of pulses can be given in a train, "gaps" in the C_M record occur between pulses and the timing accuracy is given by the operating system. To prevent these "gaps" we recommend to use the *Gap Free Mode* in Pulse Generator instead of the *Interactive Mode* (for details see PATCHMASTER reference manual).

12.6.4.1 Train of Pulses within a Sweep

We create a sequence with a train of pulses on basis of the "Depol" sequence. First, duplicate this sequence and name it "Train". Now we adjust the timing of an individual pulse for our requirements, e.g. changing the *Duration* of segment "3" and "5" and setting the *Voltage* for segment "3". Now we use the function *Duplicate...* for generation of the train. Select this function from the first segment of your template, here segment number "2". Specify how many (here "4") and how often (here "4") to duplicate segments. The result is a train with 5 pulses. You might now jump to the last segment to enter a longer duration at the end.



Please note that it might be useful to use a *PGF Parameter* to set the *Duration* or *Amplitudes*. This facilitates changing the train parameters.

The sequence "Train" should be triggered from within the protocol (e.g. "Capacitance " protocol) as we did with the "Depol" sequence before.

> ***Note:*** *The offline analysis of the step responses is conveniently done with the feature Cyclic Analysis in* FITMASTER.

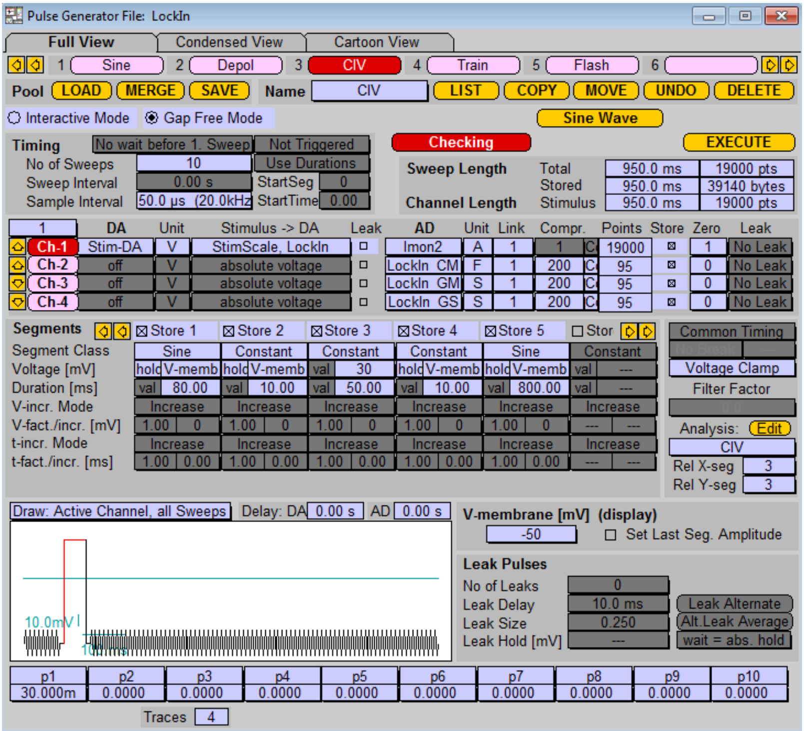
During acquisition the *Traces* are shown in the Oscilloscope. You may think about other ways of presentation and analysis on level of a *Trace*.

E.g. plot the integrals of C_M versus integral of current.

12.6.4.2 Train formed by several Sweeps

Another approach is to use the *No of Sweeps* to create a *Series* (train) of *Sweeps*. In this case please use the *Gap free Mode* in the PGF.

The experimental design is similar to the one described for generation of C-I-V curves (see 12.6.3 on page 199):



12.6.5 Flash Photolysis of Caged Ca^{2+}

Flash photolysis of caged Ca^{2+} to trigger rapid changes in C_M presents an interesting example. Here, it is desirable for the software to trigger the flash lamp at a specific time in the *Sweep* via a digital pulse output from one of the D/A outputs. It is also useful to concurrently sample a fluorescent signal which indicates the Ca^{2+} time course immediately after the flash. Finally, in our example we will also have the software control a monochromator through another D/A output to provide the dual wavelength excitation of the Ca^{2+}-sensitive dye. One should be warned that this example is quite complicated because of the demanding nature of the application. On the positive side, however, the example illustrates a few advanced topics in the use of LockIn with the PATCHMASTER software.

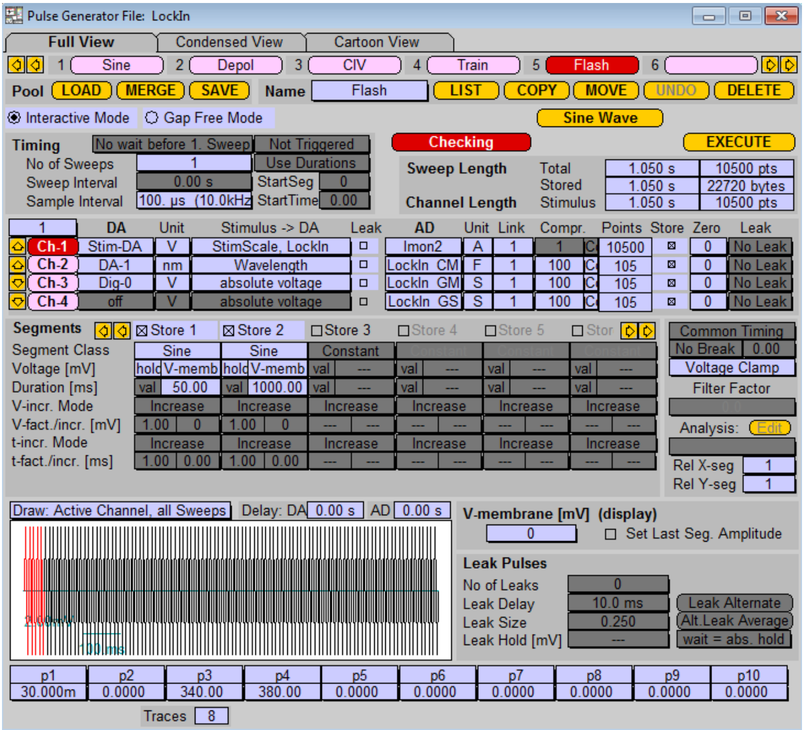
> ***Important note:*** *Please always make sure in advance that the Max. Sample Points are set to a value sufficient for your applications.*

When setting up a sequence for flash photolysis you should follow the strategy outlined below.

Channel 1 (use for capacitance):

- Select *Stim-DA*.
- Use two segments, one for baseline recording of C_M and the second segment to sample the response.
- AD-channel 1: Select *Imon2*.
- AD-channel 2: Select *LockIn_CM*.
- AD-channel 3: Select *LockIn_GM*.
- AD-channel 4: Select *LockIn_GS*.

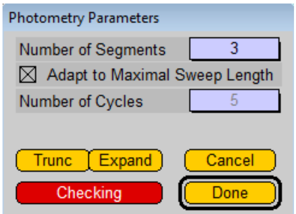
Set the *Sinewave Parameters* now.


Pulse Generator File: LockIn
Full View
Condensed View
Cartoon View
1 Sine
2 Depol
3 CIV
4 Train
5 Flash
6
Pool
LOAD
MERGE
SAVE
Name
Flash
LIST
COPY
MOVE
UNDO
DELETE
Interactive Mode
Gap Free Mode
Sine Wave
Timing
No wait before 1. Sweep
Not Triggered
Checking
EXECUTE
No of Sweeps
1
Use Durations
Sweep Interval
0.00 s
StartSeg
0
Sample Interval
100. µs (10.0kHz)
StartTime
0.00
Sweep Length
Channel Length
Total
Stored
Stimulus
1.050 s
10500 pts
1.050 s
22720 bytes
1.050 s
10500 pts
DA
Unit
Stimulus -> DA
Leak
AD
Unit
Link
Compr.
Points
Store
Zero
Leak
Ch-1
Stim-DA
V
StimScale, LockIn
Imon2
A
1
1
10500
0
No Leak
Ch-2
DA-1
nm
Wavelength
LockIn CM
F
1
100
105
0
No Leak
Ch-3
Dig-0
V
absolute voltage
LockIn GM
S
1
100
105
0
No Leak
Ch-4
off
V
absolute voltage
LockIn GS
S
1
100
105
0
No Leak
Segments
Store 1
Store 2
Store 3
Store 4
Store 5
Stor
Common Timing
Segment Class
Sine
Sine
Constant
No Break
0.00
Voltage [mV]
hold
V-memb
hold
V-memb
val
Voltage Clamp
Duration [ms]
val
50.00
val
1000.00
val
Filter Factor
V-incr. Mode
Increase
V-fact./incr. [mV]
1.00
0
1.00
0
Analysis:
Edit
t-incr. Mode
Increase
t-fact./incr. [ms]
1.00
0.00
1.00
0.00
Rel X-seg
1
Rel Y-seg
1
Draw: Active Channel, all Sweeps
Delay: DA
0.00 s
AD
0.00 s
V-membrane [mV] (display)
0
Set Last Seg. Amplitude
Leak Pulses
No of Leaks
0
Leak Delay
10.0 ms
Leak Alternate
Leak Size
0.250
Alt.Leak Average
Leak Hold [mV]
---
wait = abs. hold
100 ms
p1
30.000m
p2
0.0000
p3
340.00
p4
380.00
p5
0.0000
p6
0.0000
p7
0.0000
p8
0.0000
p9
0.0000
p10
0.0000
Traces
8

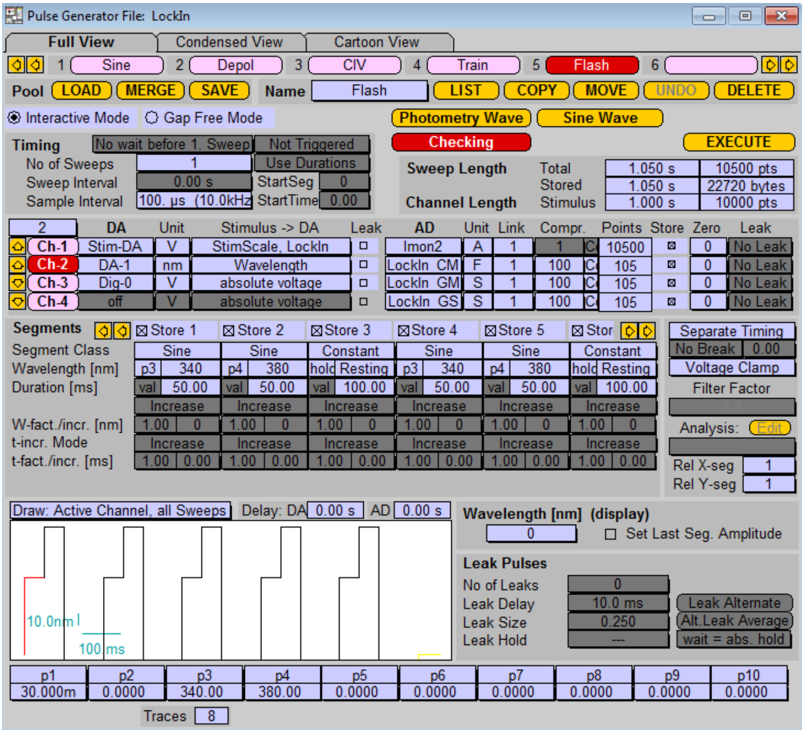
DA-Channel 2 (use for photometry):

First, activate the `Photometry Extension` in the hardware tab of the Configuration.

- Select *DA-1*.
- Use separate timing.
- Build a template with 3 segments (use PGF parameters).
- Set the *Photometry Wave* parameters as follows:
  - Set *Number of Segments* to "3".
  - Enable *Adapt to Maximal Sweep Length*.
  - Press the *Expand* button.
  - Press the *Done* button.



- AD-channel 5: Select `AD-1` (raw signal, probably compressed).
- AD-channel 6: Select `Photo_W1`.
- AD-channel 7: Select `Photo_W2`.
- AD-channel 8: Select `Photo_R`.

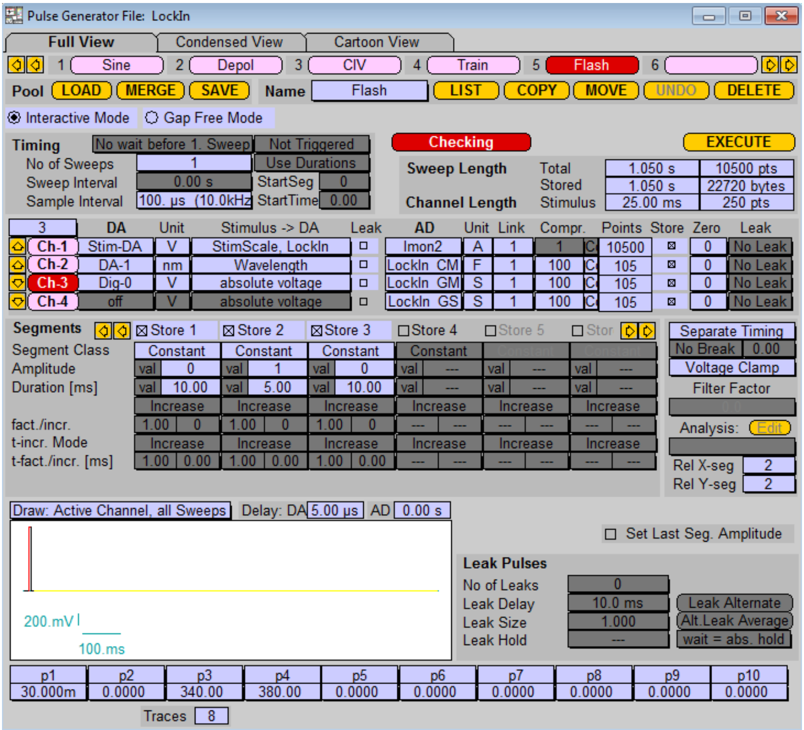


2	DA	Unit	Stimulus -> DA	Leak	AD	Unit	Link	Compr.	Points	Store	Zero	Leak
Ch-1	Stim-DA	V	StimScale, LockIn	☐	Imon2	A	1	1	10500	☒	0	No Leak
Ch-2	DA-1	nm	Wavelength	☐	LockIn CM	F	1	100	105	☒	0	No Leak
Ch-3	Dig-0	V	absolute voltage	☐	LockIn GM	S	1	100	105	☒	0	No Leak
Ch-4	off	V	absolute voltage	☐	LockIn GS	S	1	100	105	☒	0	No Leak

Segments	Store 1	Store 2	Store 3	Store 4	Store 5	Stor
Segment Class	Sine	Sine	Constant	Sine	Sine	Constant
Wavelength [nm]	p3 340	p4 380	hold Resting	p3 340	p4 380	hold Resting
Duration [ms]	val 50.00	val 50.00	val 100.00	val 50.00	val 50.00	val 100.00
	Increase	Increase	Increase	Increase	Increase	Increase
W-fact./incr. [nm]	1.00 0	1.00 0	1.00 0	1.00 0	1.00 0	1.00 0
t-incr. Mode	Increase	Increase	Increase	Increase	Increase	Increase
t-fact./incr. [ms]	1.00 0.00	1.00 0.00	1.00 0.00	1.00 0.00	1.00 0.00	1.00 0.00

DA-Channel 3 (use to trigger the flash):

- Select *Dig-0*.
- Use separate timing.
- Use three segments to trigger the flash at the beginning of the second segment of DA-1.



When executing the sequence "Flash" all 8 acquired *Traces* are shown in the Oscilloscope window. If you do not want to store individual *Traces*, then deselect the *Store* checkbox in the Pulse Generator window. If you want to save them, but are not interested to see them online in the Oscilloscope window, then deselect the *Show* flag of the specific *Trace* in the `Trace Properties` dialog of the `Display` menu. Alternatively, you can use the *Display Properties* event in the Protocol Editor to select the *Traces* shown in the Oscilloscope window.

12.6.6 Nice Scaling of the Cm Trace in the Oscilloscope

Before execution of a complex sequence, you might adjust the scaling for the C_M *Trace* in a way that the baseline C_M value as estimated by the *Auto CSlow* compensation is drawn at the top of the lowest grid box in the Oscilloscope window.

The scaling of a single grid box can be set in the LockIn window. Set C_M range to 5 times the scale of a single box.

We recommend to copy the following lines into your protocol and mark this section with a *GOTO_MARK* event. At the end of this section you might jump back to where you came from with a *GOTO* event. This block can then executed from within your protocol e.g. whenever you execute the *Auto CSlow*.

```
;Select Trace 2, CM Trace
Value            ( 0.000s): Value-4 =  1.0000, copy to "O DispTrace"
;Set Display Scaling to 1.0
Value            ( 0.000s): Value-4 =  1.0000, copy to "O YScale"
;Read CSlow value
Value            ( 0.000s): Value-1 = "E CSlow"
;get CM Y Range from LockIn configuration
Value            ( 0.000s): Value-2 = "L YRangeCM"
;get calculate 0.8 times the full range
Value            ( 0.000s): Value-2 MUL  800.00m
;add to CSlow = Cm amplitude to shift baseline
Value            ( 0.000s): Value-1 INC  Value-2
;divide by YRangeCM to get number of boxes for offset
Value            ( 0.000s): Value-1 DIV "L YRangeCM"
;multiply with -1 (neg. offset), and 0.5 (bug-correction factor)
Value            ( 0.000s): Value-1 MUL -500.00m, copy to "O YOffset"
```

> ***Note:*** *Please make sure that the index of your C_M Trace matches the Trace number set in "O DispTrace" (line 2). Note that the Trace index in the* Oscilloscope *starts with "0". Hence, Trace 2 has the index "1" etc.*

12.7 References

Fidler, N., and Fernandez, J. M., 1989, Phase tracking: an improved phase detection technique for cell membrane capacitance measurements, *Biophys. J.* 56: 1153-1162.

Gillis, K. D., 1995, Techniques for membrane capacitance measurements. Single Channel Recording, 2nd. Ed. B. Sakmann and E. Neher, Eds.. Plenum, New York.

Gillis, K. D., 2000, Admittance-based measurement of membrane capacitance using the EPC-9 patch-clamp amplifier, *Pfluegers Arch. - Eur. J. Physiol.* 439: 655-664.

Horrigan, F.T., and Bookman, R.J., 1994, Releasable pools and the kinetics of exocytosis in adrenal chromaffin cells, *Neuron* 13: 1119-1129.

Joshi, C., and Fernandez, J. M., 1988, Capacitance measurements: an analysis of the phase detector technique used to study exocytosis and endocytosis, *Biophys. J.* 53: 885-892.

Lindau, M., and Neher, E., 1988, Patch-clamp techniques for time-resolved capacitance measurements in single cells. *Pfluegers Arch.* 411: 137-146.

Neef, A., Heinemann, C., Moser, T., 2007, Measurements of membrane patch capacitance using a software-based lock-in system, *Pfluegers Arch. - Eur. J. Physiol.* 454: 335-344.

Neher, E., and Marty, A., 1982, Discrete changes of cell membrane capacitance observed under conditions of enhanced secretion in bovine adrenal chromaffin cells, *Proc. Natl. Acad. Sci. USA.* 79: 6712-6716.

13. Using the Spectroscopy Extension

13.1 Example Measurement with the MC-10 Model Circuit

13.1.1 Considerations about Sampling and Analysis Frequencies

We are interested in the frequency range up to 10 kHz. This should be sufficient to characterize the whole-cell mode in our model cell "MC-10".

In order to have enough filtering elements in our measurement circuitry we use the 20 μs stimulus filter (10 kHz cut-off) and a 10 kHz bandwidth of the current filters of the EPC 10.

A factor of "20" between the cut-off and the sampling frequency should be more than sufficient to prevent aliasing artifacts. In case we would analyze the complete chirp, we would get results up to 100 kHz. However, the signal attenuation far above the cut-off frequency in the system is so large that the noise becomes dominant. We therefore analyze only frequencies up to 12.5 kHz and set the *Min. Points / Cycle* in the Chirpwave Parameter parameter window to "16".

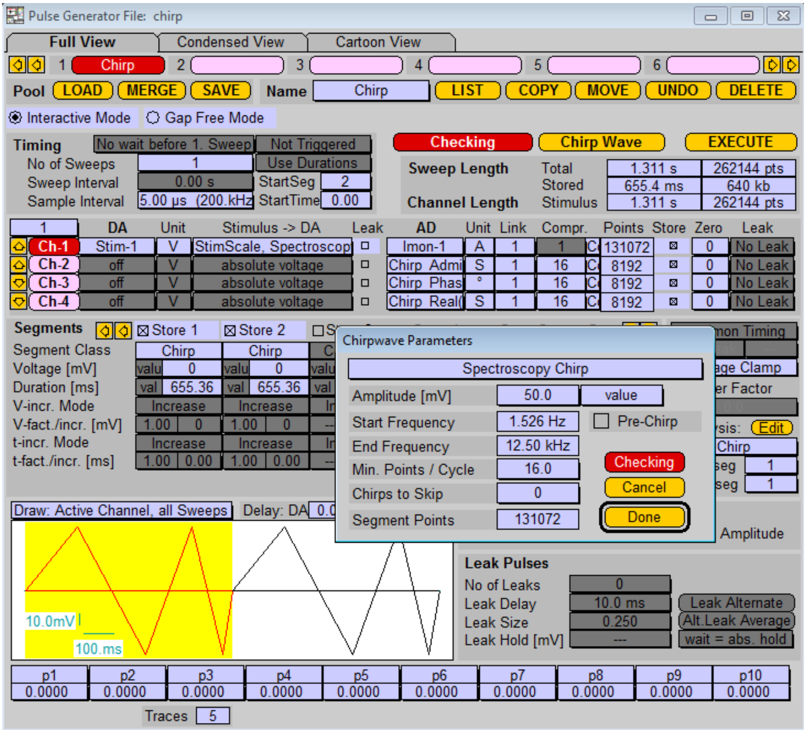
The length of the chirp stimulus determines the lowest frequency. In our example we use a duration for the chirp segment of 655.36 ms. This sets the *Start Frequency* of the chirp to 1.525 Hz.

We made a summary below of all necessary settings for the new PGF named "Chirp" you have to create:

- Number of Sweeps: 1
- Sample Interval: 200 kHz
- Start Segment: 2
- DA channel: One DA channel for stimulation (depends on your hardware). Activate *Use for Spectroscopy*.
- AD channels:
  1. Imon-1
  2. Chirp_Admit(Y) (compression 16)
  3. Chirp_Phase (compression 16)
  4. Chirp_Real(Y) (compression 16)
  5. Chirp_Imag(Y) (compression 16)
- Segments: 2 *Chirp* segments (∼655.36 ms in our example)

The *Chirpwave Parameters* are:

- Spectroscopy Chirp: On
- Start Frequency: around 1 Hz
- End Frequency: 12.5 kHz
- Min. Points / Cycle: 16
- Chirps to Skip: 0
- Segment Points: ∼131072

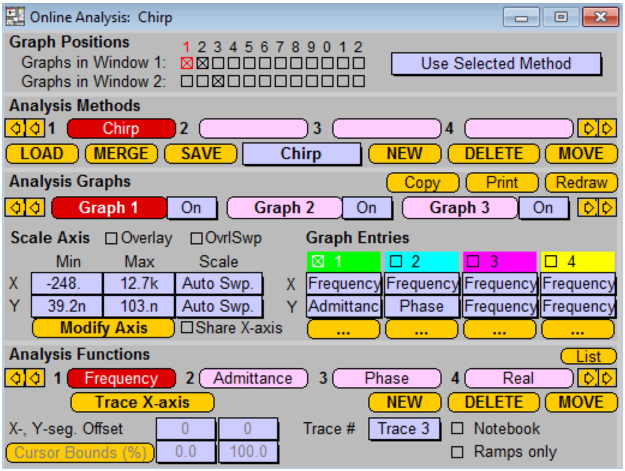


13.1.2 Reference Measurement via 10 MΩ Resistor

After setting up your PGF you have to do some more adjustments:

- *Correction Mode* in the Spectroscopy window is set to *None*.
- Amplifier *Gain* is set to 1.0 mV/pA.
- *Stim Filter* is set to 20μs.
- Set `Filter 1` to 30 kHz and `Filter 2` to 10 kHz (Bessel).
- Switch the model cell in the middle position and compensate *C-fast*.
- Switch back to the 10 MΩ position.

Before we execute our new PGF we have to define an Online Analysis:

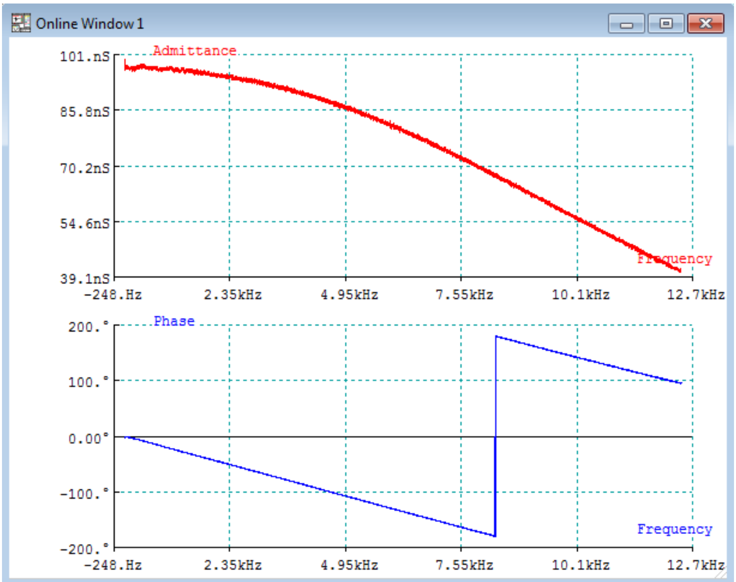


- Open the Online Analysis window.
- Create a new *Analysis Methods* by clicking into an empty field.
- Name the new *Analysis Methods* e.g. "Chirp".
- Activate *Graph* 1, 2 and 3 (*Analysis Graphs*).
- Activate *Graph Positions*: *Graph* 1 and 2 in Online window 1, *Graph* 3 in Online window 2.
- Select *Use Selected Method*.
- *Analysis Functions*:

  1. Frequency: Select *Trace x-axis (time)* of *Trace* # 3. Here, the X-axis of the *Trace* is not time but frequency!
  2. Admittance: Select *Trace* of *Trace* # 2.
  3. Phase: Select *Trace* of *Trace* # 3.
  4. Real: Select *Trace* of *Trace* # 4 (for later purposes).
  5. Imag: Select *Trace* of *Trace* # 5 (for later purposes).

- *Analysis Graphs*:
  1. X = *Frequency*, Y = *Admittance*; *Auto Sweep* scaling.
  2. X = *Frequency*, Y = *Phase*; *Auto Sweep* and *Fixed* scaling.
  3. X = *Real(Y)*, Y = *Imag(Y)*; *Auto Sweep* scaling (for later purposes).
- Open Online window 1.
- Execute the chirp acquisition ("Chirp" PGF).

The following picture shows a typical result of *Admittance* and *Phase* versus *Frequency*.



The admittance trace starts with about 100 nS (corresponds 10 MΩ) and drops with increasing frequency. At about 10 kHz the admittance is reduced by a factor of 2 (∼50 nS). This reduction is due to the two filters in the pathway. The stimulus filter reduces the signal at its cut-off frequency by 0.7 and the same does the current filter. Hence 0.7 * 0.7 = 0.5!

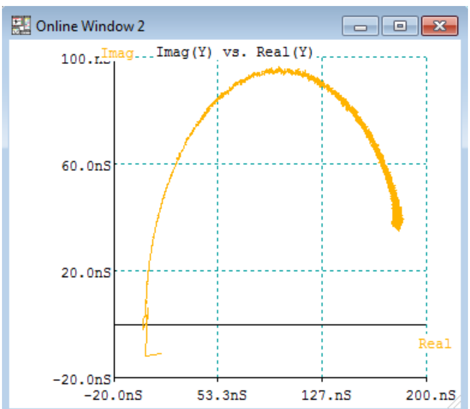
We take this recording as reference to correct all further acquisitions to this spectrum:

- Select the *I-mon* trace (Trace 1) in the Replay window.
- Select *Add Trace* from the `Buffer` menu (Trace 1 is copied into *Buffer 1*.)
- Set the *Correction Mode* in the Spectroscopy window to *Buffer 1*.
- Activate the *Reference Element Correction* and either choose *Measured Resistance* or use *Given Resistance* with the value of 10 MΩ.

13.1.3 Test Measurement on Whole-Cell Model

- Switch the model circuit to whole cell position (0.5 G)
- Open Online window 2.
- Execute the chirp acquisition (”Chirp” PGF).

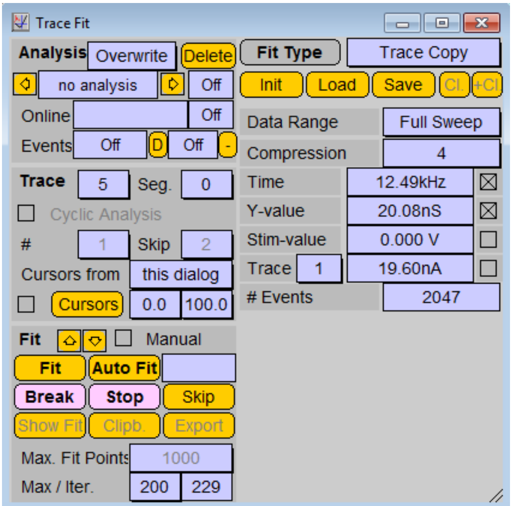
A plot of the imaginary part of the *Admittance* (Img(Y)) versus the real part of the *Admittance* (Real(Y)) is shown in the following screenshot. This plot fully describes the frequency response of the model circuit.



13.1.4 Fitting of the Results in FITMASTER

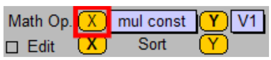
We want to use FITMASTER and its *TraceCopy* function to transfer the Real(Y) und Imag(Y) traces to the `Series Fit` level. For the analysis of the acquired data we proceed with the following steps:

- Start FITMASTER.
- Select the appropriate hardware.
- Open the data file.
- Select a sweep in the Replay window.
- Open the Trace Fit window (`Windows` menu).
- Select *Tracy Copy* as the *Fit Type*.
- Set the following parameters in the Trace Fit window:
  - Trace: Here you need first *Trace 4*, then *Trace 5*.
  - Data Range: Full Sweep
  - Compression: Set it to 4 otherwise you might get an error message concerning exceeding the maximum number of events which is per default 1024 per analysis.
  - Time: Activate the checkbox to get the values into *Series Fit*.
  - Y-Value: Activate the checkbox to get the values into *Series Fit*.

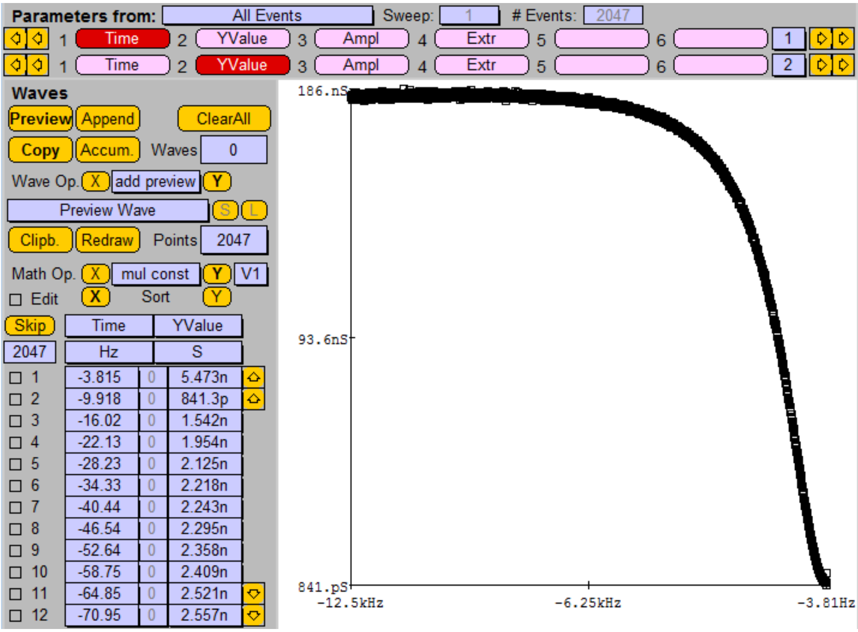


Plot the *Real(Y)* (trace 4) against the negative frequency axis:

- Select *Trace 4* in the Trace Fit window.
- Press the *Auto Fit* button.
- Series Fit: Open the Series Fit window in the `Windows` menu.
  - Select *Time* (here: frequency) for X-axis.
  - Select *YValue* for Y-axis.
  - Press the *Preview* button for displaying the graph.
  - Select *mul const* in *Math Op.* for a multiplication with a constant value.

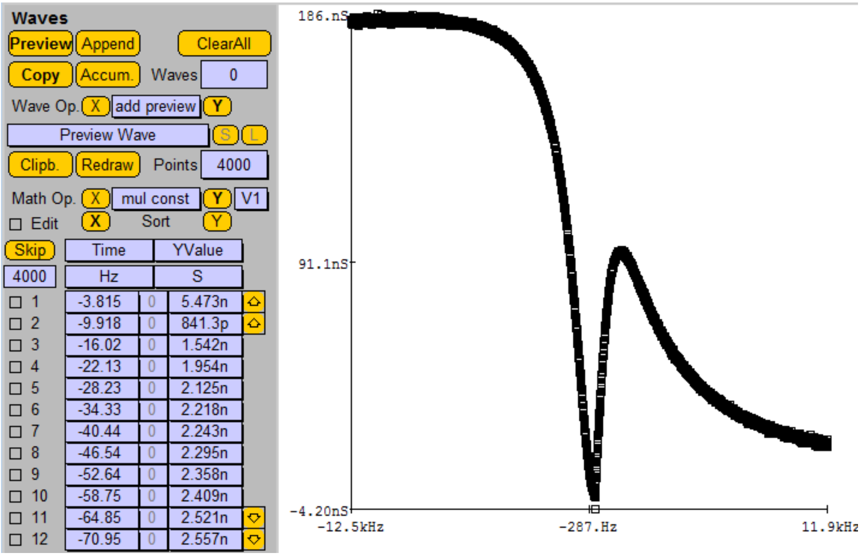


  - Press the *X* beneath *Math Op.* and enter "-1". This value will now muliplied with all X-axis values leading to the negative frequency axis.

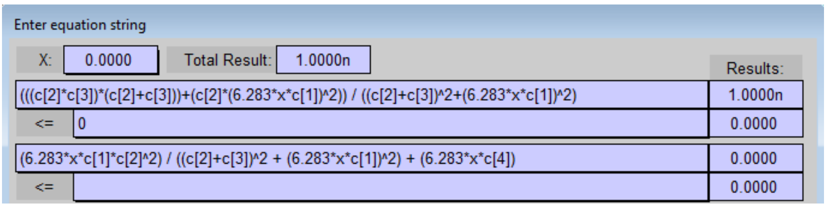


Append the *Imag(Y)* (trace 5) versus the positive frequency axis:

- Select *Trace 5* in the Trace Fit window.
- Press the *Auto Fit* button.
- Series Fit: Open the Series Fit window in the `Windows` menu.
  - Select *Time* (here: frequency) for X-axis.
  - Select *YValue* for Y-axis.
  - Press the *Append* button.



Now we can use a *Parsed Equation* as a fit function and enter a function Real(f) and Imag(f) for negative and positive frequencies respectively. We have also included a pure stray capacitance in the theoretical calculations.



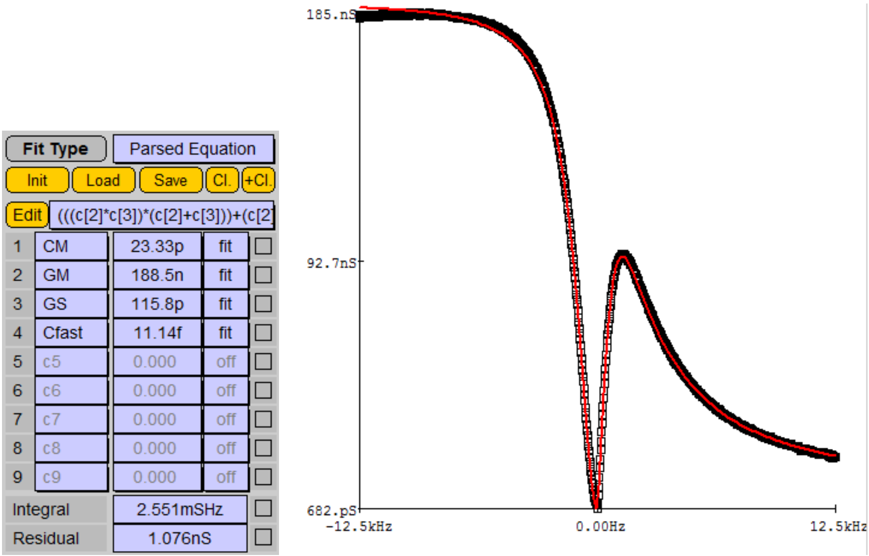
- Open the Series Fit window.
- Choose *Parsed Equation* in the *Fit Type* selection.
- Press on the *Edit* button to enter the given equation string (above). Further explanations how to save and load equations are given in the FITMASTER manual.
- Press the *Fit* button.

Note: *We advice to enter some realistic start parameters for a successful fitting.*

Note: *In case you do not see the Fit press the Show Fit button.*

We get a fit with the following parameters:

C_m = 23.33 pF, G_m = 188.5 nS, G_s = 115.8 nS, remaining C-fast = 11.14 fF.



14. Controlling PATCHMASTER

14.1 Controlling PATCHMASTER from another Program

PATCHMASTER 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 the EPC 9 and EPC 10 to be controlled over a network, even one with different platforms, such as Windows, OS/2, Mac OS, or workstations. Thus, it is now possible to control the amplifiers from computers running a multitasking operating system which can create and read shared files (e.g., Windows XP, Windows 7, Mac OS X).

Controlling PATCHMASTER from another program is possible by communicating via two ASCII-files. The user writes the commands to one file (the "command" file) and PATCHMASTER 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. PATCHMASTER will access that file with read and shared permission only. The reverse is used on the second file, the "response" file: PATCHMASTER will have write and sharing permission, and the user program read permission only (plus sharing permission, of course). These two files must be placed in the folder specified as "Batch Path" in the Configuration window of PATCHMASTER.

The first line in the "command" file must contain one positive number (as ASCII, e.g., "+1234"). This "command index" is interpreted by PATCHMASTER as follows:

- If this number is zero or negative, PATCHMASTER does not execute the commands in the "command" file.
- If the number is larger than zero, PATCHMASTER will execute the instructions immediately. PATCHMASTER will write that number to the "response" file to flag execution once all commands have been executed.
- To prevent PATCHMASTER from executing the instructions more than once, PATCHMASTER 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.
- PATCHMASTER writes the responses to the "response" file. In the first line of that file, PATCHMASTER 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 "`E9Batch.In`" and the name of the "response" file must be "`E9Batch.Out`". Both files will be inside the folder specified as the "Batch Path" in the Configuration window of PATCHMASTER.
- A text string must be set within double quotes, when it contains non-alphanumeric characters (e.g. commas, colons, blank spaces, etc.). A file name 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 "Patchmaster" folder named `"E9Batch.In"`. It has to keep this file open with "write" and "shared" access permission.

2. PATCHMASTER is then started with the *Enable Batch Control* option active. PATCHMASTER will open the file `"E9Batch.In"` with "read" and "shared" access permission. PATCHMASTER will now create the `"E9Batch.Out"` file with "write" and "shared" access permission.

3. PATCHMASTER will immediately execute the commands in the command file, provided that the "command index" is larger than zero. PATCHMASTER will also write 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, PATCHMASTER will scan the command file and execute the commands, if the "command index" changed.

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 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 are sent to the "command" file, then continues with step 5.

Polling Mode

There is a possibility of loosing commands in situations when the two programs generate intensive traffic. E.g., one application may overwrite its most recently written command before the other application had time to read it.

To avoid overwriting commands, one can switch to *Polling* mode. In this mode, one application is the talker (in PATCHMASTER named "Sender"), the other the listener (in PATCHMASTER named "Receiver"). Only the talker is actively sending commands. The other, i.e. the listener, transfers commands back only when the talker sent a message. The listener then appends its own commands to its reply. To ensure optimal responsiveness, the talker continuously sends a "heartbeat" such that the listener can send back commands with a defined, predictable delay. The "heartbeat" itself is the command "Heartbeat" without parameters. The slave has to respond with "Heartbeat [time]". A timeout occurs, if the listener does not answer to the "heartbeat" within reasonable time. The "reasonable" time is 5 times the sleep time set by the "SetSleep" command. Its default sleep time is 0.1 second.

In *Polling* mode the reply of a received command gets the prefix "Reply_", e.g. "ExecuteProtocol" is answered with "Reply_ExecuteProtocol". This change is required to support connecting multiple instances of the "Master" applications.

To inquire, whether the target application supports polling mode, send the command "ConnectionIdentify". The answer, if it supports polling mode, should be: "ConnectionProperties Sender/Receiver Polling/Bidirectional".

If the application is not in polling mode, activate it with the command "SetSleep 0.1, Polling".

Multiple Connections Mode

Polling mode only. One can activate multiple connections, whereby one application is the *Sender* and the other applications are the *Receivers*. The *Sender* is the one polling the *Receivers* via the heartbeat process (see preceding section).

The input and output files are shared by all applications. Each connection uses its separate range in the files, i.e., it reads and writes into the files starting at an offset of [connection index] * 1 MB (1048576 bytes) for a maximal file range of 1 MB. Thus, the *Sender* ¡¿ *Receiver-1* connection uses an offset of zero, *Sender ¡¿ Receiver-2* connection uses an offset of 1 MB, etc. Each connection has its own "command index", see above.

Here is an example of such a command file:

```
1. line: "1234"
2. line: "Set E VHold -0.080"
3. line: "Get E Gain"
4. line: "MakeAnError"
```

And this would be the content of the "response" file:

```
1. line: "1234"
2. line: "Get Gain 7"
3. line: "error_not_found"
```

In *Polling* mode, the example would look like:

```
1. line: "1234"
2. line: "Heartbeat"
3. line: "Set E VHold -0.080"
4. line: "Get E Gain"
5. line: "MakeAnError"
```

And this would be the content of the response file:

```
1. line: "1234"
2. line: "Heartbeat 12:30:01.123"
2. line: "Get Gain 7"
3. line: "error_not_found"
```

14.2 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\_open\_denied':  data file cannot be opened at that time
'error\_acquiring':     command not allowed while acquiring
'error\_playback':      command not allowed while macro playback
'error\_unknown':       an unidentified error occurred
```

14.3 Implemented Commands and Messages

The two operations "Get" and "Set" use the macro language to communicate with PATCHMASTER. There are two ways to get the names of macro items:

1. The option *List All Macro Items* in the `Help` menu will generate a list of all macro items in the Notebook.
2. To obtain the name of a specific item, proceed as follows:
   - Activate the option *Enable Icon Configuration* in the `Windows` menu.
   - Click on the icon with the mouse while holding the CTRL (Windows) or COMMAND (Mac OS) key pressed.
   - The name of the macro item is now displayed in the title bar of the window.

14.3.1 Messages sent by PATCHMASTER

```
GetOfflineAnalysis
     syntax: "GetOfflineAnalysis [group],[series],[sweep],[sweep_time],
     [results]"
          parameters:
               3 integers: index of Group, Series, and Sweep
               if Sweep = 0:
                    1 string containing the analysis titles
               otherwise:
                    1 time string: Sweep time in time format [hh:mm:sss.mmm]
                    n numeric values: the Online Analysis results
                         The offline analysis of the replayed Sweep.
                         This command has to be activated by a preceding
                         "SendOfflineAnalysis [notebook|all]" command,
                         see below
```

```
GetOnlineAnalysis
     syntax: "GetOnlineAnalysis [group],[series],[sweep],[sweep_time],
     [results]"
          parameters:
               3 integers: index of Group, Series, and Sweep
               if Sweep = 0:
                    1 string containing the analysis titles
               otherwise:
                    1 time string: Sweep time in time format [hh:mm:sss.mmm]
                    n numeric values: the Online Analysis results
                         The Online Analysis of the acquired sweep.
                         This command has to be activated by a preceding
                         "SendOnlineAnalysis [notebook|all]" command,
                         see below

Started
     PATCHMASTER started communication link.
     syntax: "Started"
          parameters:
               none

Shutdown
     PATCHMASTER did shutdown.
     syntax: "Shutdown"
          parameters:
               none

Terminated
     PATCHMASTER terminated communication.
     syntax: "Terminated"
          parameters:
               none
```

When the option *Synchronize Files* is active:

```
FileClosed
     PATCHMASTER closed the data file.
     syntax: "FileClosed"
          parameters:
               none

FileOpened
     PATCHMASTER opened the file in the given access mode.
     Files can be: data file ("dat"), pgf file ("pgf"),
     Protocol file ("pro"), and Online Analysis file
     ("onl"), see file extension.
     syntax: "FileOpened [new|modify|read], ["file name"]"
          parameters:
               none

FileUpdated
     PATCHMASTER updated the data file.
     syntax: "FileUpdated"
          parameters:
               none

TargetSelected
     The user clicked on a target in the Replay window and
     the field "HandledExternally" is set in the stimulation
     record of the target.
     syntax: "TargetSelected"
          parameters:
               none
```

14.3.2 Operations

```
Acknowledged
     re-synchronize command index
     syntax: "acknowledged"
          parameters:
               none
          response:
               none

CheckSequence
     syntax: "CheckSequence [Sequence]"
          parameters:
               1 quoted string: sequence to check
          response:
               "Passed", if sequence is correct and no parameter was modified
               "Modified", if sequence is correct but a parameter was modified
               error description otherwise

CloseWindow
     syntax: "CloseWindow [list of window names]"
          parameters:
               comma separated strings: names of windows to close,
               e.g. "Oscilloscope", "Amplifier", or "Notebook",
               "All" will close all windows
          response:
               "Done"

ConnectionIdentify
     syntax: "ConnectionIdentify"
          parameters:
               none
          response:
               "ConnectionProperties" followed by a list of parameters:
                    [Sender|Receiver],
                    [Polling|Bidirectional],
                    Connections=[number of activated connections, n=1...4],
                    Target=[target index of calling application, n=1...4]

DeleteSequence
     syntax: "DeleteSequence [Sequence]"
          parameters:
               1 quoted string: sequence to delete
          response:
               "Done" or error description
```

```
DisableUserActions
     disables any user action by mouse or keyboard input
     syntax: "DisableUserActions"
          parameters:
               none
          response:
               none

Echo
     syntax: "Echo [string to send back]"
          parameters:
               1 string: the string to send back in
               multi-connection mode:
                    The target address can be specified by starting
                    the string with "&" followed by "1" to "5".
         response:
              the string to send back

EnableUserActions
     Re-enables user action by mouse or keyboard input
     syntax: "EnableUserActions"
          parameters:
               none
          response:
               none

ExecuteProtocol
     syntax: "ExecuteProtocol [Protocol]"
          parameters:
               1 quoted string: protocol to execute
          response:
               "Done" or error description

ExecuteSequence
     syntax: "ExecuteSequence [Sequence]"
          parameters:
               1 quoted string: sequence to execute
          response:
               "Done" or error description
```

```
Export
     syntax: "Export [overwrite|nooverwrite], [FileName]"
          parameters:
               two strings:
               - the string defining the file overwrite mode
               - the string defining the file name. This string
                 must be set in double quotes.
               The default [export file [data file path] is used,
               if it does not contain a path.
          response:
               "Done" or error description

Get
     syntax: "Get [string]"
          parameters:
               the string defining the macro whose text is to be sent back
          response:
               - the string "Get"
               - the name of the item from which one requested
                 the item text
               - the requested item text


GetChildren
     syntax: "GetChildren"
          parameters:
             4 integers: index of Group, Series, Sweep, and Trace
             1 integer:  target level: 0=Root up to 4=Trace
          response:
               1 integer: number of children

GetComment
     syntax: "GetComment [Group, Series, Sweep, Trace, Level]"
          parameters:
               4 integers: index of Group, Series, Sweep, and Trace
               1 integer:  target level: 0=Root up to 4=Trace
          response:
               the comment as a quoted string
```

```
GetEpcParams-?
     syntax: "GetEpcParams-[index] [param-1, param-2, param-3,]"
          parameters:
               the index of the EPC amplifier (1 to 8] followed by
               the comma separated list of the requested parameters
               - parameters can be:
               CFastError
               CFastTau
               CFastTot
               Clipping
               CSlow
               CSlowError
               CSlowRange
               F2Response
               Filter1
               Filter2
               Gain
               GLeak
               IHold
               Ljunc
               Mode
               RealGain    in Ohms (i.e., V/A)
               RsComp
               RSeries
               RsMode
               StimFilter
               VHold
               Vzero
          response:
              - the string "GetEpcParams-?" followed by
              - the comma separated list of the requested parameters.
           The total string length cannot exceed 255 characters.

GetLabel
     syntax: "GetLabel [Group, Series, Sweep, Trace, Level]"
          parameters:
               4 integers: index of Root, Group, Series, Sweep, and Trace
               1 integer:  target level: 0=Root up to 4=Trace
          response:
               the label as a quoted string
```

```
GetParameters
     syntax: "GetParameters [param-1, param-2, param-3, ... ]"
          parameters:
               the comma separated list of the requested parameters
               - parameters can be:
                    AD-0 ... AD-15
                    Amplifier = quoted string identifying the amplifier
                    Clipping
                    CurrentSweep
                    DataFile
                    Digital
                    LockIn
                    OnlineFile
                    Online-0 ... Online-[max. online result]
                    Param-0 ... Param-17:
                         Param-0  = 'I-mon'
                         Param-1  = 'V-mon'
                         Param-2  = 'C-fast'
                         Param-3  = 'C-slow'
                         Param-4  = 'R-series'
                         Param-5  = 'Leak Comp.'
                         Param-6  = 'Clipping'
                         Param-7  = 'Auto C-fast'
                         Param-8  = 'Auto C-slow'
                         Param-9  = 'Pip. Pressure'
                         Param-10 = 'Pip. Resistance'
                         Param-11 = 'Temperature'
                         Param-12 = 'Cell Potential'
                         Param-13 = 'User-1'
                         Param-14 = 'User-2'
                         Param-15 = 'Timer'
                         Param-16 = 'Seal Resistance'
                         Param-17 = 'Time', i.e., seconds since midnight
                    PgfFile
                    ProtocolFile
                    SeriesDate  = international date format: 'YYYY/MM/DD'
                    SeriesTime  = 'hh:mm:ss.mmm', e.g., '14:50:13.000'
                    SweepName   = [group count]_[series count]_[sweep count]
                    SweepTime   = 'hh:mm:ss.mmm'
                    SweepTimer  = 'hh:mm:ss.mmm'
                    Value-0 ... Value-15
          response:
          - the string "GetParameters" followed by
          - the comma separated list of the requested parameters.
          The total string length cannot exceed 255 characters.
```

```
        Each number requires 9 characters.

GetSeqBlock
    syntax: "GetSeqBlock [
             sequence,
             channel,
             segment,
             target level"
        parameters:
             4 integers:
             index of sequence
             index of channel
             index of segment
             target level: 0=Root, 1=Stimulation, 2=Channel, 3=Segment
        response:
             2 integers:
             endian type: 0=big endian (PPC), 1=little endian (Intel)
             number of bytes in block
             1 data block: im binhexed format (8-to-4 bit encoding),
             the 1. nibble starts immediately after the comma following
             "number of bytes", or error description

GetSettings
    syntax: "GetSettings"
        parameters:
             none
        response:
             list of settings parameters.

GetTarget
    syntax: "GetTarget"
        parameters:
             none
        response:
             4 integers: index of Group, Series, Sweep, and Trace
             1 integer:  target level: 0=Root up to 4=Trace

GetTime
    syntax: "GetTime"
        parameters:
             none
        response:
             "Time" plus a string representing the present time
```

```
GetVersion
     syntax: "GetVersion"
          parameters:
               none
          response:
               "Version" followed by the actual program version

HardwareAccess
     syntax: "HardwareAccess [open|close]"
          parameters:
               1 string: "open" to open connection to AD/DA board,
                         "close" to close connection to AD/DA board.
          response:
               "Done" or error description

Heartbeat
     syntax: "Heartbeat"
         parameters:
              none
         response:
              "Heartbeat" plus a string representing the present time

ListProtocols
     syntax: "ListProtocols"
          parameters:
               none
          response:
               a comma separated list of quoted strings

ListSequences
     syntax: "ListSequences"
           parameters:
                none
           response:
                a comma separated list of quoted strings

NewSequence
     syntax: "NewSequence[Source,Target,Interval,Trigger,SweepNo,SegmentNo,]"
          parameters:
               2 quoted strings: source and target sequence
               1 real:     sweep interval
               3 integers: trigger mode: 0=no trigger, 1=trigger series, etc.
                           number of sweeps
                           number of segments
               N pairs of reals:
```

```
                           segment duration
                           segment amplitude
                           optionally:
                                "Tincr:" followed by the duration increment
                                "Vincr:" followed by the amplitude increment
          response:
               "Done" or error description

OpenFile
     syntax: "OpenFile [new|modify|read], [FileName]"
          parameters:
               two strings:
                    - the string defining the file access mode
                    - the string defining the file name. The default
                      [data file path] is used, if it does not contains
                      a path.
          response:
               returns an error if it fails.

OpenOnlineFile
     syntax: "OpenOnlineFile [FileName]"
          parameters:
               one string:
               - the string defining the name of the Online file. The
                 default [online file path] is used, if it does not contain
                 a path.
          response:
               results in an error if it fails.

OpenPgfFile
     syntax: "OpenPgfFile [FileName]"
          parameters:
               one string:
               - the string defining the name of the PGF file. The
                 default [pgf file path] is used, if it does not contain
                 a path.
          response:
               results in an error if it fails.
```

```
OpenProtFile
     syntax: "OpenProtFile [FileName]"
          parameters:
               one string:
               - the string defining the name of the protocol file. The
                 default [protocol file path] is used, if it does not
           contains a path.
        response:
               returns an error, if it fails.

Query
    syntax: "Query"
         parameters:
              none
         response:
              "Query_Quitting"   -> when the program is about to quit
              "Query_Ag_WaitTrig"-> when waiting for trigger
              "Query_Acquiring"  -> when acquiring sweeps
              "Query_Aq_Waiting" -> when acquiring sweeps and wait at end
              "Query_Executing"  -> when executing a protocol
              "Query_Ex_Waiting" -> when waiting while executing a protocol
              "Query_Recording"  -> when a macro is being recorded
              "Query_Playback"   -> when a macro is being played back
              "Query_Running"    -> when running a task (e.g. replay)
              "Query_Ru_Waiting" -> when waiting while running a task
              "Query_Idle"       -> otherwise.

SelectSequence
     syntax: "SelectSequence [Sequence]"
          parameters:
               1 quoted string: sequence to select in the PGF-editor
          response:
               "Done" or error description

SendOfflineAnalysis
     Instructs PATCHMSTER to activate or deactivate sending the results
     of the offline analysis of a replayed Sweep to the connected application,
     see command "GetOfflineAnalysis" above.
     syntax: "SendOfflineAnalysis [off|notebook|all]"
          parameters:
               1 string: the sending mode: [off|notebook|all]
          response:
               returns an error, if it fails
```

```
SendOnlineAnalysis
     Instructs PATCHMASTER to activate or deactivate sending the results
     of the Online Analysis of an acquired Sweep to the connected application,
     see command "GetOnlineAnalysis" above.
     syntax: "SendOnlineAnalysis [off|notebook|all]"
          parameters:
               1 string: the sending mode: [off|notebook|all]
          response:
               returns an error, if it fails

Set:
     syntax: "Set [FORCE] [WAIT] [string]"
          parameters:
               - optional: passing FORCE will execute the command even
                           when another macro is running at that moment
               - optional: passing WAIT will wait for the termination of
                           the command before returning to the caller
               - the string to be sent to the macro interpreter
          response:
               - if FORCE was not issued, and another macro is running
                 at that moment, "Set Macro_Still_Executing" is returned
                 to signal that the command was NOT executed.
               - if the macro interpreter reports an error:
                 'error_syntax'
               - otherwise, no response

SetComment
     syntax: "SetComment [Group, Series, Sweep, Trace, Level, Label]"
          parameters:
               4 integers: index of Group, Series, Sweep, and Trace
               1 integer:  target level: 0=Root up to 4=Trace
               1 quoted string: comment
          response:
               "Done" or error description

SetLabel
     syntax: "SetLabel [Group, Series, Sweep, Trace, Level, Label]"
          parameters:
               4 integers: index of Group, Series, Sweep, and Trace
               1 integer:  target level: 0=Root up to 4=Trace
               1 quoted string: label
          response:
               "Done" or error description
```

```
SetPGF:
     syntax: "SetPGF"
             Target="name",
             Source="name", Channel=[number],Dac=[number], Adc=[number],
             Source="name", ...
      parameters:
           - Target:  the name of the target PGF-template
           - Source:  the name of the 1. source PGF-template
           - Channel: the index of the target channel
           - Dac:     DA-channel of the target channel
           - Adc:     AD-channel of the target channel
           - Source:  the name of the 2. source PGF-template...
      response:
           none

SetSeqBlock
     syntax: "SetSeqBlock [
             sequence,
             channel,
             segment,
             target level,
             endian type,
             number of bytes
             data block]"
     parameters:
          6 integers:
               index of sequence
               index of channel
               index of segment
               target level: 0=Root, 1=Stimulation, 2=Channel, 3=Segment
               endian type: 0=big endian (PPC), 1=little endian (Intel)
               number of bytes in block
          1 data block: im binhexed format (8-to-4 bit encoding),
               the 1. nibble must start immediately after the comma
               following "number of bytes".
               The data block must have the same byte format as in the
               corresponding block in the "pgf" file.
     response:
          1 integer (target level), and "Done", or
          error description

SetSleep
     syntax: "SetSleep [real: seconds]"
          parameters:
               one real: the period (in seconds) between polling the
```

```
                    input file.
          optional a string: "Polling" to switch to Polling mode or
          "Bidirectional" to switch to the Non-Polling mode.
     response:
          no response, if no mode switching occured
          sends the command back, if mode switching occured

SetTarget
    syntax: "SetTarget [Group, Series, Sweep, Trace, Level, Show,
    Analyze]"
         parameters:
              4 integers: the indexes of Group, Series, Sweep, and Trace
              1 integer: the target level: 0=Root up to 4=Trace
              2 booleans: TRUE or FALSE
                   1. boolean: Show - show the specified target.
                   2. boolean: Analyze - if TRUE, and show = TRUE,
                      analyze the selected Sweep
         response:
              "Done" or error description

SetValue
     syntax: "SetValue [Index, Value]"
          parameters:
               1 integer: the index (0...15) of the static "Value" array
               1 real: the value to be stored.
          response:
               "Done" or error description

SetWindow
     syntax: "SetWindow [window name, top, left, height, width]"
          parameters:
               1 string: name of the window, e.g. "Oscilloscope",
                         "Amplifier", or "Notebook". Use "FrameWindow"
                         to address the containing Frame Window
                         (Windows only).
               4 integers: top, left, height, width. Passing zero for a
                           parameter leaves it unaffected.
```

```
ShowWindow
     syntax: "ShowWindow [list of window names]"
          parameters:
               comma separated strings: names of the windows to show, e.g.
               "Oscilloscope", "Amplifier", or "Notebook", "All" will show
               the windows visible when "CloseWindow All" was issued.
          response:
               "Done"

SweepInfo
     syntax: "SweepInfo"
          parameters:
               none
          response:
          - the string "SweepInfo" followed by a list of parameters,
            separated by semicolons, for each Trace of the presently
            selected Sweep. The parameters are:
          - Trace index at time of acquisition ("TraceCount")
          - number of data points
          - x-interval
          - "DataFactor", i.e. factor by which the raw data are to
             be multiplied to get unity.
          - offset in bytes from beginning of data file

SweepInfoExt
     syntax: "SweepInfoExt"
          parameters:
               none
          response:
               - the string "SweepInfoExt" followed by
                 - the response of the "Query" command,
                 - the Sweep ID ([GroupIndex]_[SeriesIndex]_[SweepIndex])
                 - a semicolon,
                 - a list of parameters, separated by semicolons,
             for each Trace of the presently selected Sweep.
             The parameters are:
             - Trace index at time of acquisition ("TraceCount")
             - number of data points
             - x-interval
             - "DataFactor", i.e. factor by which the raw data are to
                be multiplied to get unity.
             - Y-range: dynamic range of y-data
             - zero data
             - offset in bytes from beginning of data file
             - interleave block size in bytes
```

```
          - interleave block skip bytes
          - data type: 0=int16, 1=int32, 2=real32, 3=real64
          - endian type: 0=big endian (PPC), 1=little endian (Intel)
          - temp file: 0=stored in data file 1=in temp file

Shutdown
     syntax: "Shutdown"
          parameters:
               none
          response:
               if PATCHMASTER is acquiring, the command is ignored and
               the string "error_acquiring" is returned.
               Otherwise, the string "Shutdown" is returned, and then
               PATCHMASTER shuts down.

SystemEvent
     syntax: "SystemEvent [command]  [parameter]"
          parameters:
               if command = "Application":
                    1 string: "ToFront" -> brings the application window
                              to the front and activates it
               if command = "Window":
                    2 strings: - window_name and action
                         - window name, e.g. "Oscilloscope"
                         - action: "Activate", "Show", or "Close".
          response:
               none

Terminate
     syntax: "Terminate"
          parameters:
               none:
          response:
               returns the string "Terminated"

WriteHeartbeat
     syntax: "WriteHearbeat" [FALSE|TRUE]"
          parameters:
               1 boolean: TRUE or FALSE
          response:
               none

WriteToNotebook
     syntax: "WriteToNotebook text"
          parameters:
```

```
            1 quoted string: the text to write
        response:
            none

WriteToTermIO
    syntax:
        "WriteToTermIO [FALSE|TRUE]"
        parameters:
            1 boolean:  TRUE or FALSE
        no response
```

14.3.3 Accessing Data directly in the Data File

One can access the data while PATCHMASTER has opened the data file. The following functions can be used:

- The name of the opened data file is obtained with "GetParameters DataFile".
- The access parameters are obtained with "SweepInfoExt". All relevant parameters are returned, such as offset into the data file, number of samples, byte format, scaling information, etc.
- Finally, one can open the data file in read-only mode, and read the data using the parameters obtained with "SweepInfoExt".

14.3.4 Handshaking

Handshaking can be implemented between PATCHMASTER and the external program. The following functions are possible:

- The selection "Write to Batch" of the "WriteValue" protocol event allows to make PATCHMASTER send a message back to the external program at a given position in an event protocol.
- *Sweep* and *Series* name of an acquisition in progress can be obtained from the "Statistics" field in the `Oscilloscope` window.

- After the "WriteValue" protocol event, one can add a "Wait Icon" event.
- Thus, PATCHMASTER will wait for the external program to issue a "Resume" command. The external program can react to the synchronization with the state of PATCHMASTER, since it was actively notified by PATCHMASTER.

If the *Synchronize Files* option in the *Batch Communication* field (`Hardware` tab of the Configuration window) is active, then PATCHMASTER will actively send messages when it opens, updates, and closes files.

14.4 Notes for Programmers

One has to diligently select when to open the message file. That file is created by PATCHMASTER, therefore, it cannot and should not be opened before PATCHMASTER 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 PATCHMASTER is located and to wait for the message file to appear in the PATCHMASTER directory. At that moment one can start PATCHMASTER and enable the option *Batch Communication Enable* in the Configuration window. The message file will then be created and the user program can now open it for reading and sharing.

The first response PATCHMASTER writes to the message file is "started". Thereafter, the user can proceed to send commands to PATCHMASTER 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 proceed as follows:

1. Write a minus sign ("-") to the first byte of the command file. This prevents PATCHMASTER 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, PATCHMAS-

TER as well as the user program, run concurrently. Thus, PATCHMASTER may attempt to read from the message file while the user program is writing to it!

2. Write the remainder 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 PATCHMASTER 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. PATCHMASTER 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.

14.5 Sample Programs

The listing below is an example on implementing the communication protocol used to control PATCHMASTER as described above. It demonstrates the functions needed to write a program to control PATCHMASTER. The code is written in Modula-2. The code can easily be understood also by readers familiar with BASIC, PASCAL, C, C++, or FORTRAN.

A working VisualBasic program including the sources can be ordered from HEKA. The name of that software package is PULSECOMMANDER.

```
MODULE UserCommands;

CONST
    CommandName = ‘E9Batch.In’;
    MessageName = ‘E9Batch.Out’;

VAR
    CommandFile     : IOFiles.FileHandleType;
    MessageFile     : IOFiles.FileHandleType;
    Signature       : INTEGER;
    LastSignature   : INTEGER;

PROCEDURE WriteError( Text : ARRAY OF CHAR );
VAR
    Work : ARRAY[0..79] OF CHAR;
BEGIN
    IOFiles.Message( IOFiles.GetError(), Work );
    TermIO.WriteString( Text );
    TermIO.WriteString( ‘ failed: ’ );
    TermIO.WriteLine( Work );
    Alert.Beep;
END WriteError;

PROCEDURE WriteToCommandFile( Text : ARRAY OF CHAR ): BOOLEAN;
VAR
    Work : ARRAY[0..79] OF CHAR;
    Count : INTEGER;
BEGIN
    IF NOT CommandFile.IsOpen THEN RETURN FALSE; END;

    IF Text[0] = 0C THEN RETURN TRUE; END;

    Work[0] := 0C;
    Decode.Integer( - ABS( Signature ), Work, 0 );
    Count := Strings.Length( Work );

    (* First, we write a negative signature to the first text line.
       This prevents to the target PATCHMASTER to read and interpret
       the content of this file. We use the actual signature already
```

```
   now, so that we can just change the sign from negative to
   positive by overwriting one character!*)

IF
    ( NOT IOBytes.SetPosition(CommandFile,IOFiles.FromStart,0) ) OR
    ( NOT IOText.Write( CommandFile, Count, ADR( Work ) ) )
THEN
    WriteError( 'Writing negative signature' );
    RETURN FALSE;
END; (* IF *)

Count := Strings.Length( Text );

(* Second, we write the command text to the second (and following) line.*)

IF NOT IOText.Write( CommandFile, Count, ADR( Text ) ) THEN
    WriteError( 'Writing to command file' );
    RETURN FALSE;
END; (* IF *)

(* Third, if the signature should be positive:
 - write the terminating empty string, then
 - overwrite the negative sign (i.e., "-") of the signature in
   the first line with a plus (i.e., "+"), thus changing the
   negative signature to positive. See above for the
   explanation *)

IF Signature >= 0 THEN
    Work[0] := 0C;
    Count := 1;

    IF NOT IOBytes.Write( CommandFile, Count, ADR( Work ) ) THEN
        WriteError( 'Writing terminator' );
        RETURN FALSE;
    END; (* IF *)

    Work[0] := '+';
    Count := 1;

    IF
        (NOT IOBytes.SetPosition(CommandFile,IOFiles.FromStart,0))
    OR
        ( NOT IOBytes.Write( CommandFile, Count, ADR( Work ) ) )
    THEN
        WriteError( 'Writing positive signature' );
```

```
        RETURN FALSE;
      END; (* IF *)
    END; (* IF *)

    TermIO.WriteLn;
    TermIO.WriteString( ‘command : [’ );
    TermIO.WriteInt( Signature, 0 );
    TermIO.WriteString( ‘]’ );
    TermIO.WriteLine( Text );

    INC( Signature );

    IF Signature < 1 THEN Signature := 1; END;

    RETURN TRUE;

END WriteToCommandFile;


PROCEDURE OpenMessageFile(): BOOLEAN;
BEGIN

    IF NOT CommandFile.IsOpen THEN RETURN FALSE; END;

    IF MessageFile.IsOpen THEN
      Alert.String( ‘Message file already open! ’ );
      RETURN FALSE;
    END; (* IF *)

    IF NOT
      IOText.Open(
      MessageName,
      IOFiles.Read + IOFiles.Shared,
      MessageFile )
    THEN
      IF IOFiles.GetError() <> IOFiles.FileNotFound THEN
        WriteError( ‘Opening message file’ );
      END; (* IF *)
      RETURN FALSE;
    END; (* IF *)

    LastSignature := -1;

    RETURN TRUE;
```

```
END OpenMessageFile;


PROCEDURE PollForCommands;
VAR
    InputString      : ARRAY[0..127] OF CHAR;
    NewSignature     : INTEGER;
    Count            : INTEGER;
    Dummy            : BOOLEAN;

BEGIN

    IF NOT CommandFile.IsOpen THEN
        RETURN;
    END; (* IF *)

    IF ( NOT MessageFile.IsOpen ) AND ( NOT OpenMessageFile() ) THEN
        RETURN;
    END; (* IF *)

    (* Read the first text line. If reading fails, or converting to
       a number fails, or the resulting number is negative, then the
       answer from the target PATCHMASTER is not ready.*)

    NewSignature := -2;
    Count := HIGH( InputString );

    IF
        ( NOT IOText.SetPosition( MessageFile, IOFiles.FromStart, 0 ) )
    OR
        ( NOT IOText.Read( MessageFile, Count, ADR( InputString ) ) )
    THEN
        RETURN;
    END; (* IF *)

    IF ( InputString[0] = 0C ) OR ( InputString[0] = '-' ) THEN
        RETURN;
    END; (* IF *)

    Count := Encode.Integer( InputString, NewSignature );

    IF ( NewSignature < 0 ) OR ( LastSignature = NewSignature ) THEN
        RETURN;
    END; (* IF *)
```

```
    TermIO.WriteString( ‘response: [’ );
    TermIO.WriteInt( NewSignature, 0 );
    TermIO.WriteString( ‘]’ );

    Count := HIGH( InputString );

    IF
        IOText.Read( MessageFile, Count, ADR( InputString ) ) AND
        ( InputString[0] <> 0C )
    THEN
        TermIO.WriteLine( InputString );
    ELSE
        (* If the "response" text is empty, then the command has been
            sucessfully processed!*)

        TermIO.WriteLine( ‘done.’ );
    END; (* IF *)

    LastSignature := NewSignature;

    IF ( NewSignature > Signature ) OR ( NewSignature = 0 ) THEN
        Signature := NewSignature + 1;
        Dummy     := WriteToCommandFile( ‘acknowledged’ );
    END; (* IF *)

END PollForCommands;


PROCEDURE Startup(): BOOLEAN;
BEGIN

    IF CommandFile.IsOpen THEN
        Alert.String( ‘Command file already open! ’ );
        RETURN FALSE;
    END; (* IF *)

    IF NOT
        IOText.Open(
        ‘E9Batch.In’,
        IOFiles.Create + IOFiles.Write + IOFiles.Shared,
        CommandFile )
    THEN
        WriteError( ‘Opening command file’ );
        RETURN FALSE;
    END; (* IF *)
```

```
    Signature := 1;
    RETURN TRUE;

END Startup;

PROCEDURE InitModule;
BEGIN

    CommandFile.IsOpen := FALSE;
    MessageFile.IsOpen := FALSE;
    Signature := 1;
    LastSignature := -1;

END InitModule;


PROCEDURE Shutdown;
VAR
    Dummy : BOOLEAN;

BEGIN

    Dummy := WriteToCommandFile( 'Terminate' );
    Dummy := IOText.Close( CommandFile );
    Dummy := IOText.Close( MessageFile );

    InitModule;
END Shutdown;

END UserCommands.
```

Index