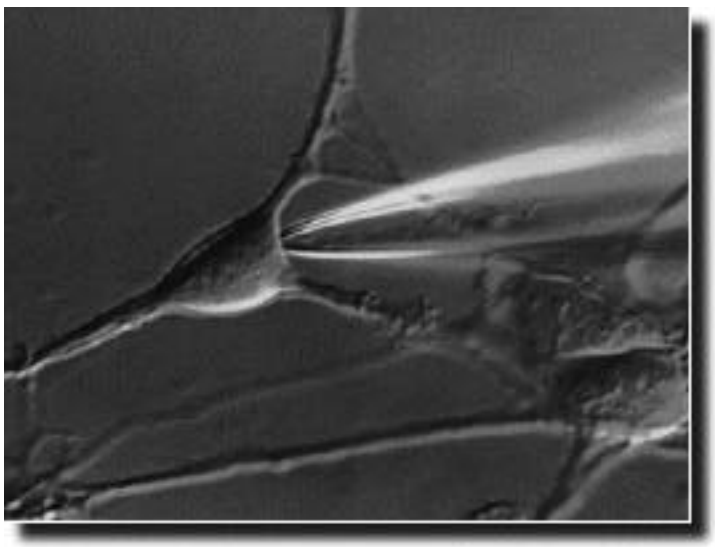
8.52

PULSE+PULSEFIT Manual



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

Table of Contents

Introduction

Scope of the Program

The *PULSE+PULSEFIT* program provides versatile tools for electrophysiological experiments. Pulse generation, data acquisition, storage, and analysis are among them. Programmers involved in development are *Hubert Affolter, Jochen Biedermann, Kevin Gillis, Stefan H. Heinemann, Michael Pusch,* and *Frank Würriehausen*. We would like to thank *Klaus Bauer, Franco Conti, Jörg Dreessen, Patrick Frisch, Toshinori Hoshi, Markus Hoth, Erwin Neher, Luis Pardo, Reinhold Penner, Nils Rennebarth, Peter Ruppersberg, Fred Sigworth, Rüdiger Steffan,* and *Walter Stühmer* for help with initial versions, stimulating input, and β-testing.

The *PULSE+PULSEFIT* package consists of two parts: *PULSE*, the main acquisition program and *PULSEFIT* to fit raw traces and derived parameters. Other programs expanding the possibilities of *PULSE+PULSEFIT* are available from HEKA:

- *PULSETOOLS* allows i.e. editing of data, special-purpose leak subtraction or non-stationary noise analysis.
- *PULSESIM*

Naming Conventions

EPC 9, EPC 9 Double, and EPC 9 Triple

Throughout the present manual we will address all three amplifiers types as “EPC9”. We will explicitly mention the particular amplifiers, where it is required.

Windows versions

The EPC9 is supported on Windows 3.1, Windows 95, Windows 98, Windows NT 3.51, Windows NT 4.0, and Windows 2000.

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

Starting PULSE+PULSEFIT

Upon clicking on *PULSE+PULSEFIT* the user will be asked to select from the two main applications (*PULSE* and *PULSEFIT*). By selecting *PULSE*, the software will be loaded and various controls will become available:

- The drop-down menus (File, Edit, Pulse, Tree, Buffer, Marks, Display, Notebook, and Amplifier, EPC7, EPC8 or EPC9).
- The *PULSE* windows (e.g., *Oscilloscope, EPC9 Amplifier, Replay, Online Analysis, Parameters, Pulse Generator, Configuration*).
- A scrolling protocol window at the bottom (*Notebook*).

Most of these windows can be iconized (close box). They can be re-opened by clicking on the icon or by selecting them in the drop-down menu **Pulse**. The *Oscilloscope* window is the master window and cannot be closed. All open windows are updated whenever the program encounters some changes; program speed can therefore be accelerated by keeping only the essential windows open.

Help

In most modes of the program help is provided by pressing 'HELP'. 'CMD' + '?' or 'CMD' + '/' (MacOS) serve the same purpose. The *Help Screen* is a text buffer, in which key words can be searched.

The option **Show Keys** in the drop-down menu **Pulse** displays the keys that are assigned to various controls of the active windows. The same can be obtained by the combination 'OPT' + 'HELP'; the labels showing the key assignments are removed by 'SHIFT' + 'OPT' + 'HELP' or if a window update is forced.

Exit and Restart

To exit from *PULSE+PULSEFIT*, select **Quit** 'CMD' + 'Q' (MacOS) or 'ALT' + 'Q' (Windows) from the drop-down menu *Files*. Acquired data are stored and files are closed before exit in any case.

Support Hotline

If you have any question, suggestion, or improvement, please contact HEKA's support team. The best way is to send us an e-mail or fax specifying:

- Your postal and e-mail address (or fax number)
- The program name:
  E9SCREEN, PULSE, PULSEFIT, etc.
- The program version number:
  v8.31, v8.50, etc.
- Your operating system and its version:
  MacOS 7.6.1, MacOS 8.5,
  Windows 98, Windows NT 4.0, etc.
- Your type of computer:
  Mac PPC 8500, Pentium II 300 MHz, etc.
- Your acquisition hardware, if applicable:
  EPC9, ITC–16, ITC–18
- Your amplifier, if applicable:
  EPC9, EPC9 Double, Axon 200B, etc.
- The serial number and version of your EPC9, if applicable:
  EPC9 single, version "920552 D".
- The questions, problems, or suggestions you have
- Under which conditions and how often the problem occurs

We will address the problem as soon as possible.

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

Installing *PULSE+PULSEFIT*

This section describes how to install the software *PULSE+PULSEFIT* on your MacOS or Windows compatible computer. *PULSE+PULSEFIT* is the acquisition and analysis software for electrophysiological experiments. *PULSE+PULSEFIT* also features functions for directly controlling HEKA's *EPC7*, *EPC8* and *EPC9* amplifier. If you purchased one of these amplifiers together with the program, please refer to the corresponding manuals. *PULSE+PULSEFIT* may also be used in combination with other amplifiers attached to an *ITC-16*, *ITC-18* or *Digidata 1200* interface board.

Actually, *PULSE+PULSEFIT* contains two applications in one executable file: *PULSE*, necessary to acquire the electrophysiological data and *PULSEFIT*, that allows you to automate the analysis of such data. The two parts of *PULSE+PULSEFIT* are protected by the *PULSE Hardware Key*. Depending on the program ordered (*PULSE* and / or *PULSEFIT*), either part may perform in full or in *Demo* mode. The demo version of *PULSE* will not actually acquire the desired AD channels. It will just display the generated stimulus. The demo mode will function even in the absence of any attached hardware. It may be used to:

1. see the actual stimulus applied to the cell
2. analyze data on a computer away from the patch-clamp setup (restricted to the files Demo.dat)
3. evaluate the PULSE+PULSEFIT software package
4. learn the software

The demo version of *PULSEFIT* is restricted to the analysis of the file **Demo.dat** which is supplied with the *PULSE+PULSEFIT* package. In addition, it will not allow to store the analysis results to disk.

The installation procedure for software and hardware is described in the separate "Installation_8x5" manual.

PULSE Tutorial: The first Experiment

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

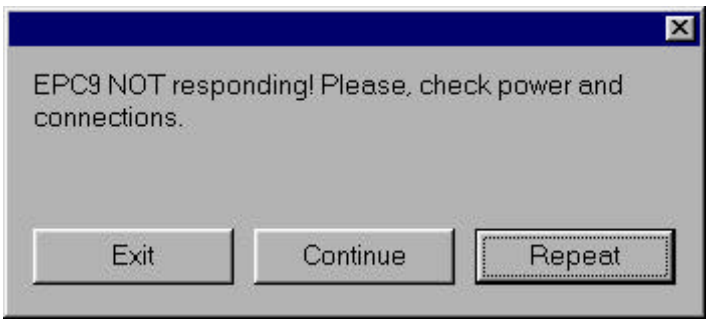
Note: In the following it is assumed that the hard- and software has already been set up correctly. Please refer to the chapter "Software Installation" for the installation of PULSE+PULSEFIT. If you plan to use the EPC9 Double or Triple you should also first read the Chapter "Setting up the EPC9" to get an idea of the basic amplifier operation.

Step 1: Starting *PULSE*

If you have a model circuit (*MC8* or *MC9*), connect it to the probe input of your amplifier via a BNC adapter and plug the black cable to the black ground connector on the probe. Turn on the amplifier, the computer and - if you don't have an *EPC9* amplifier - the AD / DA interface (*ITC-16*, *ITC-18*, or *Digidata 1200*). To start *PULSE* double-click on **Pulse.exe** (Windows) or **Pulse+PulseFit** (MacOS) which is located in the **Pulse** folder inside the **HEKA** folder. Windows users might alternatively use the **Start** button to launch *PULSE* from **Programs** → **HEKA**. Since *PULSE* and *PULSEFIT* are combined into one application, you will be asked to select which part of the program you wish to load. **PULSE** is used for data acquisition and **PulseFit** for data analysis.

When *PULSE* starts it will first look for its *Configuration File* that contains all of your individual program settings. This file is called **DefaultPulse.set** and resides in the same path as *PULSE*. If this file does not exist, *PULSE* will come up with an error message and will give you the opportunity to either locate the file (**Find File**) or to use default settings (**Use Defaults**). If you do not have a configuration file or wish to reset to the factory settings, select **Use Defaults**. In this case, *PULSE* will try to detect your hardware and generate reasonable settings.

PULSE will recognize, whether you have not connected any AD/DA hardware, and will ask you to abort (**Exit**), to continue (**Continue**), or to try again (**Repeat**). If you just forgot to turn on the power on the *EPC9* or the AD/DA converter, do so and select **Repeat**. If you don't have either of them, you still can select **Continue**, i.e., *PULSE* will continue to run in the so called *"Demo Mode"* that allows you to test the software and analyze the data. Of course, data acquisition will be disabled in *Demo Mode* but opening, saving and printing files, as well as creating and modifying pulse protocols will be possible.

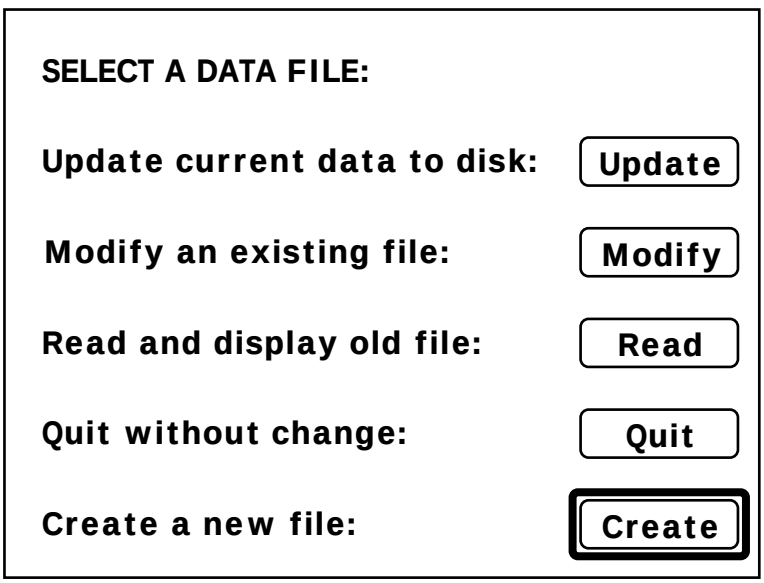


Based on your configuration file, *PULSE* will now look for file paths and a default *Pulse Generator File (PGF)*, that contains your stimulus protocols. The factory default is the file **DefPGF.pgf** in the **Data** folder inside the **HEKA** folder. If *PULSE* cannot find your PGF file, it will write a message into the *Notebook* window and will create a default file with only one entry called *"Test"*. There may be other paths missing and *PULSE* will put up an alert to that effect. You can safely ignore that error message, we will setup these paths next in the *Configuration* window (see below).

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

There are five possibilities:

- **Modify** opens an existing experiment for modification, i.e. you can delete or add further experimental data to a file.
- **Read** opens an existing experiment. The file will be write protected, so that modification (or loss) of the data is prevented.
- **Quit** cancels the dialog (or quits the program if you



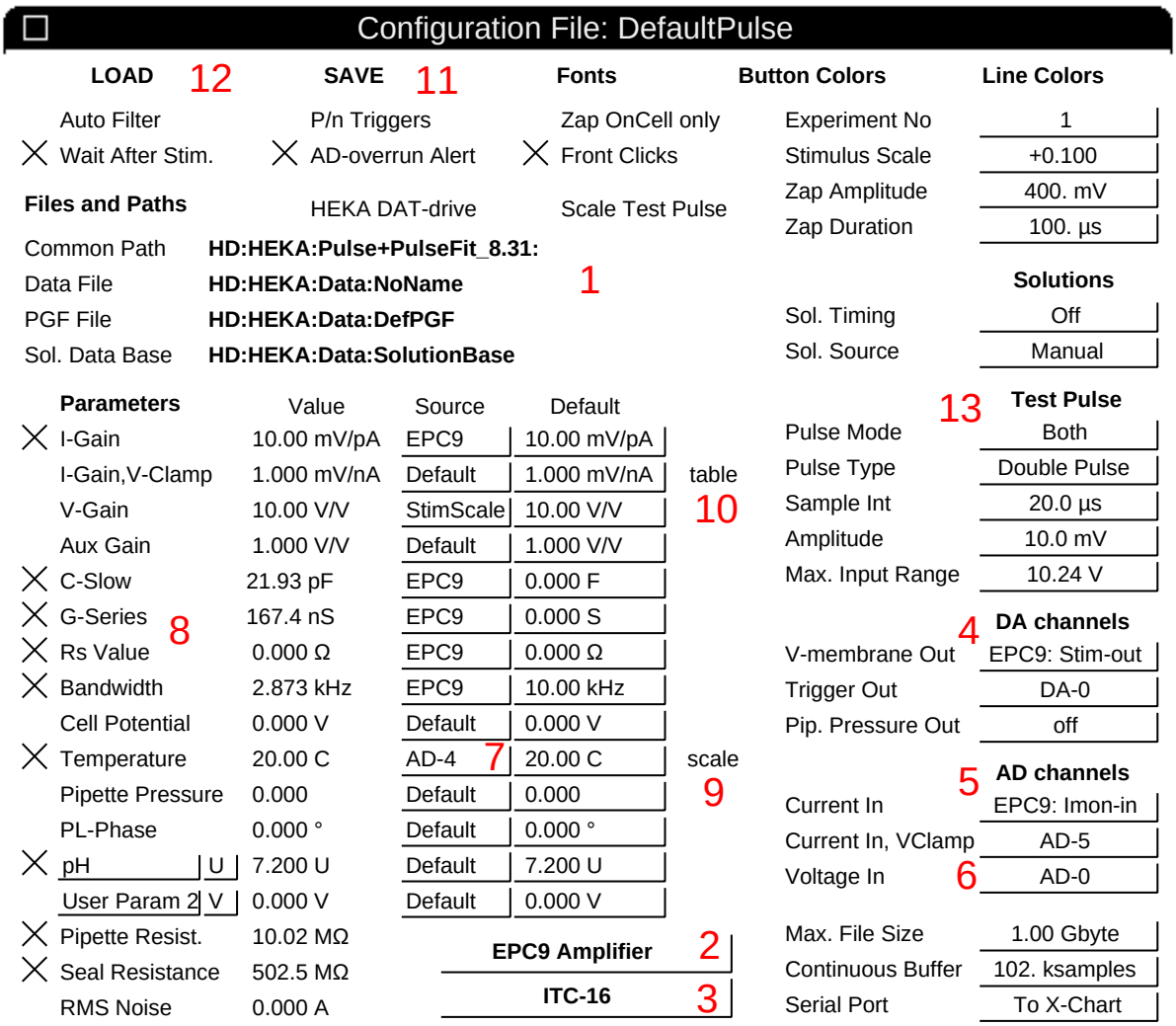
were starting *PULSEFIT*).

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

Step 2: Configuring the Hardware



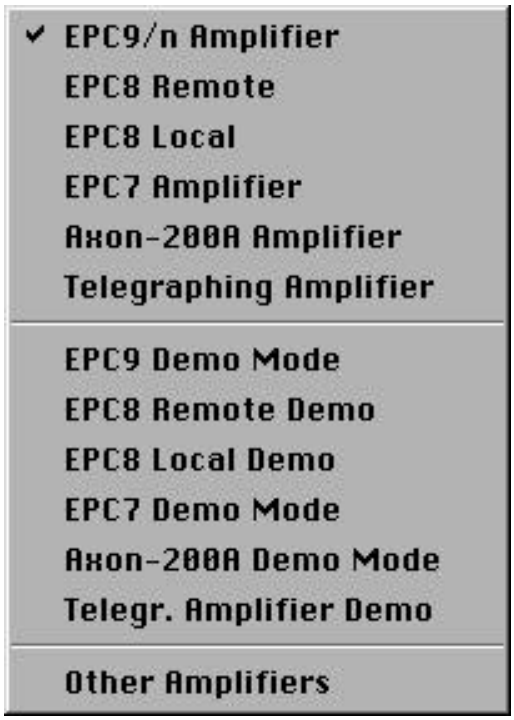
PULSE will open some windows: the most obvious ones are the *Amplifier* and the *Oscilloscope* window. We will deal with these windows soon; however, first we have to make sure that the hardware is connected properly and that the software settings meet the requirements. The most important hardware settings are defined in the *Configuration* window. Select **Configuration** from the drop-down menu **PULSE** or type F11 (MacOS) or F8 (Windows).

Paths

The Configuration window provides a variety of parameters that can be adjusted. First of all, in order to tell *PULSE* where to look for the relevant files and where to store your data, you need to set up the paths. This is done by selecting the **Common**

Path, **Data File**, **PGF File**, and **Sol. Data Base** button in the **Files and Paths** section of the *Configuration* window (1).

AD/DA Channels



PULSE has to know whether to use an *EPC9* patch-clamp amplifier or another amplifier: use the control in the lower part of the screen (2). If your particular amplifier is not in the list, select the *EPC7*, if you have no amplifier at all, use one of the *Demo* modes instead. If your amplifier is not an *EPC9* (nor *EPC9/2* or *EPC9/3*) you will also have to define the AD/DA converter you use (*ITC16*, *ITC18*, or *Digidata 1200*) in the lower popup control (3). In this example you cannot select an AD/DA-board (the selections are disabled), since the *EPC9* uses its built in AD/DA converter.

The next step will be to define the AD and DA channels to be used for stimulation and acquisition of data in the sections **DA channels** and **AD channels**. For users of the *EPC9*, some of these channels are predefined. With the *EPC9* i.e. the voltage stimulus is always expected to go via DA-3 (**V-membrane Out**, 4). The current input is sampled via AD-6 (**Current In**, 5) and the voltage is sampled from AD-0 (**Voltage In**, 6). Therefore, if you have an *EPC9*, you should connect the *VOLTAGE-MONITOR* output of the *EPC9* to the channel you defined with the **Voltage In** (6) popup by a BNC cable. In the *EPC9 Double* and the *Triple* the above connections are already hardwired inside the amplifier and therefore can not be changed.

Note: The EPC9 has 4 DA output channels (0…3) and 8 AD input channels (0…7). For the EPC9, AD channel 7 is reserved and should not be used for anything else. In addition, DA-0, DA-1, and AD 0...3 are not freely available for the EPC9 Double and DA 0...2 and AD 0...5 are not freely available for the EPC9 Triple since they are internally hardwired to the current- and voltage outputs of the respective amplifiers.

Parameter Input

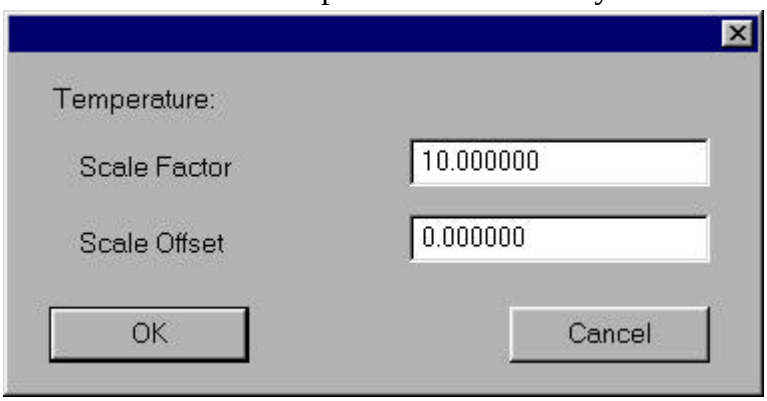
So far we specified the most important settings. On the left-hand side of the *Configuration* window - the **Parameters** section - there is a list of further values that are ac-

quired and stored together with the experiment. These *Parameters* can be input via different means: they can either be sampled through a free AD channel (**Source** = *AD-0 ... AD-7*, 7), they can be derived directly from the *EPC9* (**Source** = *EPC9*) like the settings of the *C-Slow* compensation (**C-Slow** and **G-Series**, 8) or they can be 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 *Parameters* window (see *Chapter Parameters*).

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

Go through all of these parameters and select the input source and maybe new *default* values. If an AD channel is used as input source, then the scaling can be specified. Let us suppose the *Temperature* is read from a temperature control unit via AD channel 4 (7). Let's further assume that the control unit delivers an analog signal of 100 mV/°C and 0 V at 0 °C. Then the scaling factor to be entered is *10* (1V corresponds to **10** °C) and the offset is *0*. Click the **scale** button (9) next to the *Temperature Parameter* and fill in these settings.



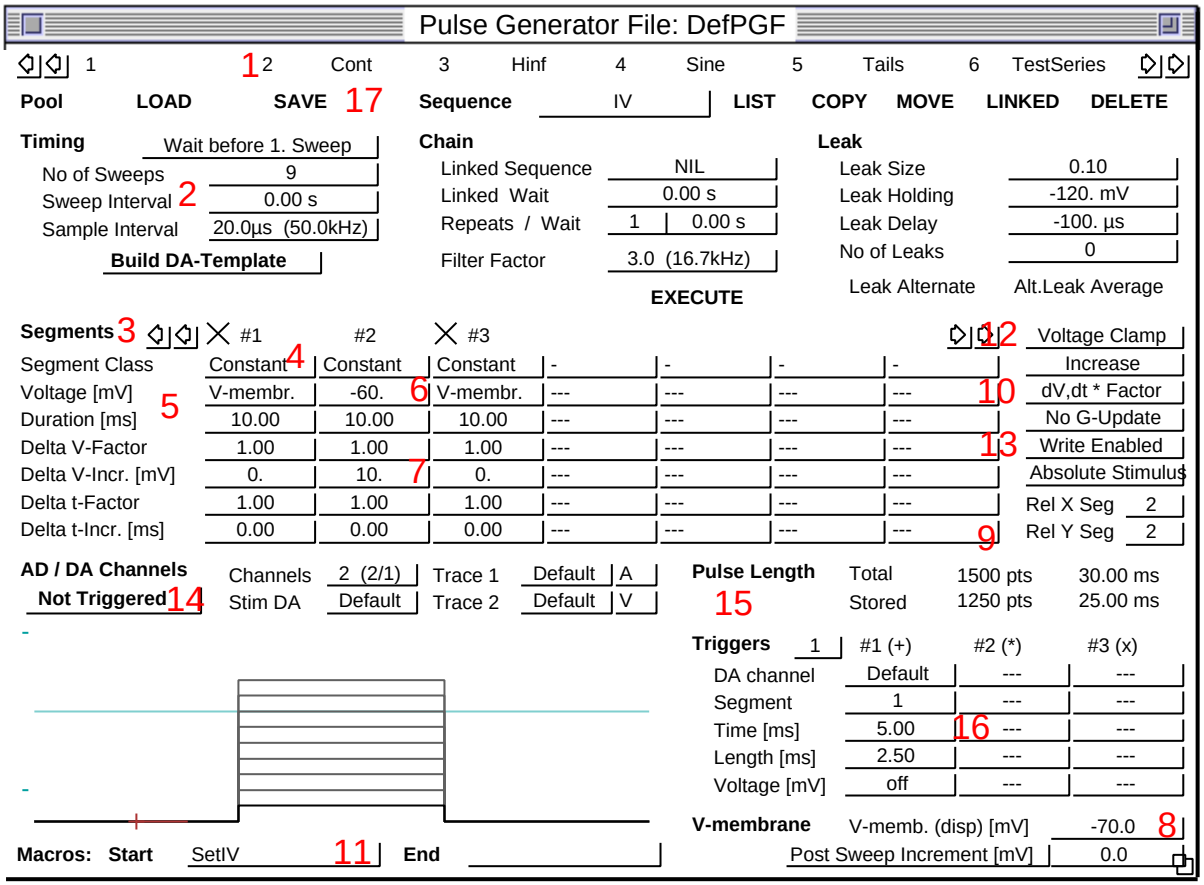
If you are sampling from a so called "*telegraphing*" amplifier, the determination of the encoded amplifier gain and filter bandwidth settings is much more involved and is done via lookup tables that are provided as ASCII files (see *Appendix IV*). You can select the corresponding gain or bandwidth table with the button **table** (10).

Test Pulse

The *Test Pulse* is applied to the pipette whenever you activate the amplifier by bringing the *Amplifier* window to the front. You can define most of its settings in the **Test Pulse** section (13) of the *Configuration* window. The amplitude, duration and pulse type are also available from the *Amplifier* window, however, the **Pulse Mode** can be only defined herein. **Current** means, that the current trace is displayed in the *Oscilloscope* every time the test pulse runs, while **Both** displays both, the current and the voltage trace. Use this later *Pulse Mode*, if you frequently want to apply test pulses in the *Current Clamp* configuration.

Finally, you can save the configuration: click the **SAVE** button (11) and provide a file name. *PULSE* allows you to store the configuration under any file name, however, when the program comes up, it will look for a file called **DefaultPulse.set**. You can load a different configuration at any time using the **LOAD** (12) button, but you should keep in mind, that many changes, e.g. those involving the redefinition of the digitizer hardware, require a restart of the program.

Step 3: Generating a Simple Pulse Protocol



The *PULSE* software allows you to stimulate your cell with pulse pattern from a basic rectangular pulse to highly complicated stimulation patterns. Stimulus templates are edited in the *Pulse Generator* window. To open it select **Pulse Generator** from the **Pulse** drop-down menu or type 'F9' (Windows) or 'F12' (MacOS). A pulse pattern (called *Sweep*) consists of an arbitrary number of pulse *Segments* that have a constant, ramp, or sinusoidal voltage. In the default *Pulse Generator File* only one sequence is created, however, the file **DefPGF.pgf** distributed with the software release and usually installed into the **Data** folder inside the **HEKA** folder contains a lot of useful pulse protocols which are a good starting point to create your own ones.

Creating a new sequence

Click a free position in the PGF pool (1). If there is no free position, click the right arrow unless you reach the end of the pool. *PULSE* will ask you for a new **Entry-Name**: type *IversusV*. The first six protocols will be directly available from the *Oscil-*

loscope window. If you want to bring this new protocol to a more handy position click the **MOVE** button and select *1* to *6*as the new position.

Timing

We want to create a protocol 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 (2) set **No of Sweeps** to *9* and the **Sweep Interval** to *1*. To edit the fields shift-click the corresponding field, i.e. click it with the mouse while keeping the 'SHIFT' key pressed. Usually *PULSE* will wait the time defined as the **Sweep Interval** before starting the pulse sequence. If you however want the sequence to start immediately after activating it, select **No wait before 1. Sweep** from the popup next to **Timing**.

Defining the Segments

The section **Segments** (3) of the *Pulse Generator* defines the actual pulse protocol to be applied. It will consist of three parts: the cell will be held at the actual holding potential in the beginning and the end of the protocol, the middle part has the depolarizing step. The individual parts of the pulse protocol are called *Segments*. At the beginning the protocol only has one segment of 5 ms duration. To add the further 2 necessary segments, click on the button labeled **Constant** (4) and select **Insert** from the popup. Repeat this step for the third segment.

Although you can edit the segments in any order, it is often advisable to start by defining the length of the individual segments. Segment 1 and 3 are to be 10 ms long, while the depolarizing pulse has a duration of 20 ms. Fill out the corresponding **Duration [ms]** fields appropriately (5).

The first and last segment should be at holding potential, so keep the corresponding checkboxes on the top set. Deselect the middle checkbox to disable it. The value in **Voltage [mV]** changes from *V-membr.* (i.e. the actual pipette holding potential at the time of executing the protocol) to *0* to reflect the fact, that *PULSE* has to set an absolute voltage (depending on the setting in 13). Change this value to *-60* (6) and set the **Delta V-incr. [mV]** field to *10* (7). this will instruct *PULSE* to jump to *-60 mV* when it first executes the protocol and then depolarize this segment by *10 mV* every other time of the 8 repeats (-50, -40, -30, ..., +20 mV).

Watch the preview in the lower left part of the window! If a segment is set to *V-membr. PULSE* will show it using the actual pipette potential (usually 0 mV if you are defining the protocol *off-line*). If you later want to apply the protocol from a hyperpolarized potential you should change the value **V-memb. (disp) [mV]** accordingly, e.g. to *-70 mV* (8). This will change the way *V-membr.* segments are displayed in the

preview. Note, this value does not affect your measurement, it is only used for previewing the sweep!

Maybe you wondered why the first segment is drawn in red color in the preview while the rest is black. *PULSE* can perform an *Online Analysis* whenever you run or replay an experiment (see below). This is done by analyzing one segment (**Rel Y Seg**, 9), 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**, 9). You can define which segment has to be analyzed by setting the so called *Relevant Segment* (9). This can be 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 *Online Analysis* window you can set a positive or negative *Segment Offset* that will be added to the relevant segment, thus allowing you to analyze also the other segments (e.g. the pre-trigger).

Other Settings in the Pulse Generator

C-Slow Update: If you are using the *EPC9* amplifier you can associate some of the automatic compensation procedures with the pulse protocol. If the option **Sweep C-Slow** (10) is activated *PULSE* will perform an estimation of the input resistance *R-Series* and membrane capacitance *C-Membrane* before every single sweep thus canceling capacitive artifacts.

Macros: Start: Before running the protocol, *PULSE* is instructed to execute a macro (**SetIVMean**, 11) that we will define in the next section. Running this macro will make sure that the *Online Analysis* window is always correctly calculating the mean current of the depolarized segment against its holding potential, when executing the protocol. At this point you should not care about the error message that tells you that the macro is not defined.

Various Settings: There are a few more options in the right and bottom part of the *Pulse Generator* window that had not to be changed in our case. Nevertheless, it is still important to know what they do: the setting **Voltage Clamp** (12) will restrict the execution of the pulse protocol to the *Voltage-Clamp* modes only. Thus, *PULSE* 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 **VC+CC Modes** in the Pulse Generator window is only there for compatibility with older versions of PULSE. If you want to be able to run a Current-Clamp sequence while you are in a Voltage-Clamp mode, you should create a macro that switches to the CC mode and associate it with the pulse protocol (**Macros: Start:**).

The option **Write Enabled** (13) will make sure, that the pulse sequence can be stored to disk. For some protocols it might not be required to save the data (such as during a pre-conditioning waiting period), so you can disable this feature in these cases.

The section **AD / DA Channels** (14) defines which channel to stimulate and from which channel(s) to acquire the data. **Stim DA** is set to *Default* which means that you are stimulating through the *EPC9* stimulus channel. The number of input **Channels** is set to 1 (*PULSE* allows you to acquire two channels simultaneously), making a total of one input and one output channel (the bracketed number next to **Channels**). The channel to be sampled (**Trace 1**) is also defined as *Default*, i.e. *PULSE* will acquire the *Current Monitor 2* in *Voltage-Clamp* and the *Voltage-Monitor* in *Current-Clamp* modes.

The section **Pulse Length** gives you some important information about the pulse protocol (15). The value **Total** is the total length of stimulation. Each sweep has a duration of 10 + 20 + 10 = *40 ms* sampled at an interval of *50 µs* or a frequency of *20 kHz* (2). This makes a total number of *800* data points (15). From these data the first 5 ms will not be stored, because *PULSE* only saves the data **after the first trigger** (16). This makes a **Stored Pulse Length** of *700* data points or *35 ms* (15). If you want to store the whole sweep to disk set the first trigger to *0 ms* or set the number of triggers to *0*. This feature is mainly thought to allow you to limit storage of pre-trigger segments.

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

This new, modified *Pulse Generator File* should now be stored to disk by clicking on **SAVE** (17). It can have any name; *PULSE* automatically adds the file extension "*.pgf*". If you want *PULSE* to come up with this *PGF-file* already loaded at the next launch, simply save the *Configuration File* before exit.

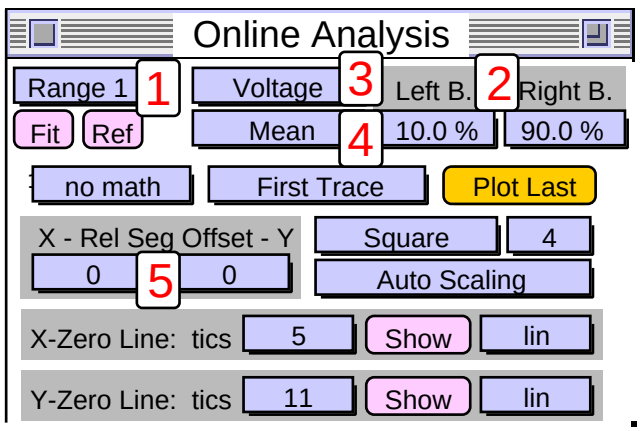
Step 4: Recording a Macro

If you run the above IV-protocol, you might want to see not only the raw current response of the cell, but also the resulting IV-curve, where the mean- or peak- current is plotted against the underlying potential. This is a typical task for the built in *Online Analysis*. To do the right kind of calculation on the right data, we will have to setup the *Online Analysis* first. Since we will often need this type of analysis it is a good idea to create a macro that does this setting up automatically in future. This macro can then be associated with the pulse protocol from above.

Macros can be generated in *PULSE* by letting the program record your commands, while you are executing them. Afterwards *PULSE* will convert these actions into a text file that you can easily edit with any ASCII editor (e.g. the Windows **NotePad** or Apple's **SimpleText**) in order to "fine tune" it. You can start the macro recorder by selecting **Start Recording** from the **EPC9 -> Macros** sub-menu.

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

PULSE can perform two calculations at a time (**Range 1** and **Range 2**). In order to define the first analysis we have to activate *Range 1* from the **Range** popup (1). Set **Left Boundary** to 10% and **Right Boundary** to 90% (2). This will restrict the analysis to the range of 10-90% of the requested relevant segment (or 2-18 ms of the depolarizing pulse). From the **Abscissa** popup (3) select *Voltage* and from the **Mode** popup (4) select *Mean*. These settings tell *PULSE* to calculate the mean current of the analyzed sweep and plot it against its holding potential. From the **Relevant Segment Offset** popup choose *0* for both settings to analyze the relevant segment we defined in the *Pulse Generator*.



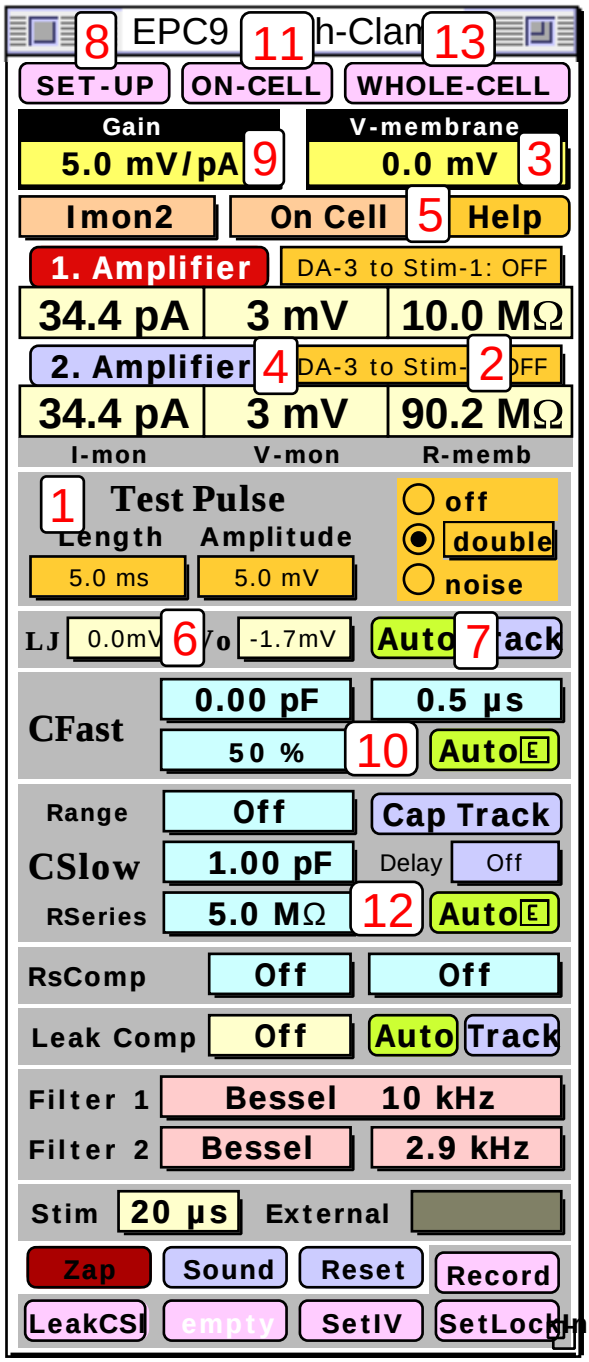
The macro is finished by now, so select **Stop Recording** from the **EPC9/Macros** sub-menu. You will be asked for an index and a name for the macro. You should not select the indices 1 through 3 because these are the predefined macros **SET-UP**, **ON-CELL**, and **WHOLE-CELL**. Use the index 4 instead. The name for the macro should be the one that you supplied within the *Pulse Generator* (*"SetIVMean"*), since macros are internally recognized upon their names rather than their index. Finally, let us have a look at the macro by selecting **List** from the **EPC9 -> Macros** sub-menu. The fourth macro should give you in the *Notebook* window something like the following:

```
4 : SetIVMean
  A  Range:            0; Range 1
  A  LeftB:           10.0%
  A  RightB:          90.0%
  A  Abscissa:         0; Voltage
  A  Mode:             5; Mean
  A  RelXSeg:          0
  A  RelYSeg:          0
  A  Scale:            1; Auto Scaling
```

Note: PULSE has a built in macro interpreter that executes command lines of the form "Window Control[: parameter; comment]". E.g., the line "A LeftB: 10%" would instruct PULSE to set the **left** boundary in the Online Analysis window to **10%**. For this tutorial it is not necessary to know all possible commands and their syntax. Please, refer to the Chapter "Macros" for further details about macros.

Step 5: Amplifier Control

The *EPC9 Patch Clamp* window provides the amplifier control functions, when an *EPC9* amplifier is used (the picture is for an *EPC9 Double)*. A more detailed tutorial describing the most common functions of the *EPC9* and its control window is given in the *Chapter "Setting up the EPC9"*. If you are using an *EPC9* you are highly recommended to read that tutorial first. An extensive description of the amplifier windows is given in the two chapters "EPC9 Amplifiers" and "EPC7, EPC8, and other Amplifiers".



If you have an amplifier other than the *EPC9* you have to make sure that:

- the command potential at the amplifier is set to zero,
- the stimulus scaling is set correctly (see *Chapter Configuration*),
- the DA channel for *Stimulus Out* is connected to the amplifier's stimulus input,
- the amplifier's current monitor is connected to the *Current In* AD channel,
- if the amplifier has telegraphing capabilities for gain and/or bandwidth, that the corresponding analog outputs are connected to the assigned AD channels. For telegraphing amplifiers, gain and bandwidth lookup tables in ASCII format can be used that translate the voltage output to the setting of the corresponding switch (see *Chapter Configuration*).

Make a Seal

Now it's time for the experiment. 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 *Oscilloscope* is not in front, hit the space bar twice, the space bar toggles between the *Oscilloscope* 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 (1): a *"double"* pulse of *5 mV* amplitude and a duration 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 (2).

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-membrane" because the program cannot distinguish between an open and a sealed pipette. As long as the pipette is open to the bath, "R-membrane" corresponds to the pipette resistance.

The command potential is controlled by the program via the control **V-membrane** (3) and displayed in (4). This variable always displays the *physiological* membrane potential, i.e., the *Recording Mode* (5) 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 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 (6) or you can alternatively click on the **Auto Vo** button (7) to let *PULSE* do this correction automatically for you. The same is done by calling the macro **SET-UP** (8), in this case, *PULSE* will also adjust the amplifier gain (9) 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 typing "W" (write). This value is not changed any further, unless you type "W" 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. The Pipette Resistance will be stored together with

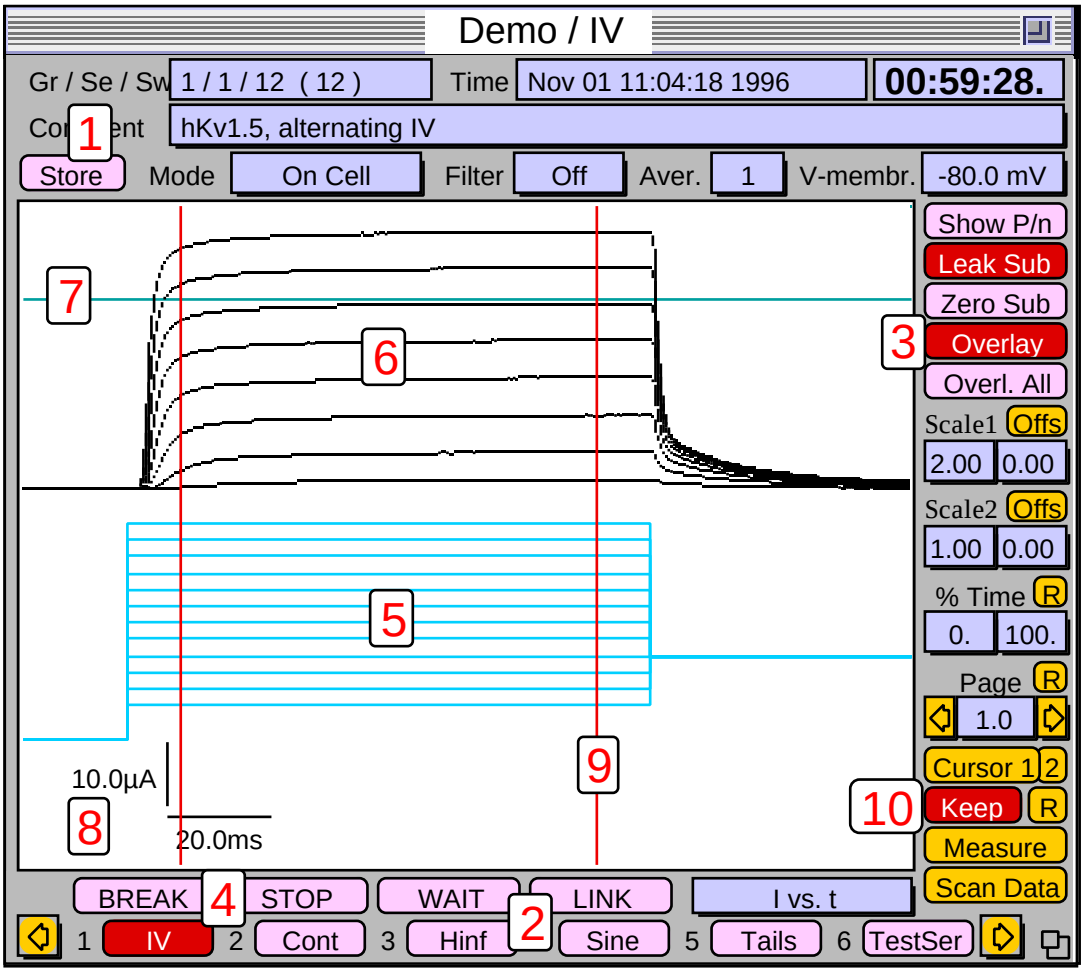
every acquired series of sweeps. This value is updated every time you type the “W” key.

Now simulate a pipette sealed to the membrane by switching the model circuit into the middle position. If you have an *EPC9* make an automatic fast capacitance cancellation by clicking on the **Auto CFast** (10) or **SET-UP** macro (11) 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 *EPC9* make an automatic slow capacitance cancellation by clicking on the **Auto CSlow** (12) or **WHOLE-CELL** macro (13) button. Otherwise compensate your amplifier for the “cell” capacitance of about 22 pF. Watch the **R-memb** display that now shows “*500 M”* instead of *“10 M”*. With the **V-membrane** control (3) change the pipette holding potential to -*70 mV*, now we are ready to run the pulse protocol we defined before.

Step 6: Run the Pulse Sequence

Bring the *Oscilloscope* window to the front. If the button **Store** (1) is not highlighted click on it to make it active. Otherwise *PULSE* will show the data but not write them to disk. If you did not create a file yet, *PULSE* will ask you to do this now.



The bottom of the *Oscilloscope* window shows the currently available *Pulse Generator Pool* (2). An individual *Sequence* can be executed by either clicking on one of the controls e.g. the **IV** button, by entering the corresponding number ('1' ... '9') or by pressing "E" (for *Execute*) for the highlighted *Sequence*. A *Sequence* with an index higher than 9 can be executed from the keyboard by typing the pound key "#" followed by the number of the sequence.

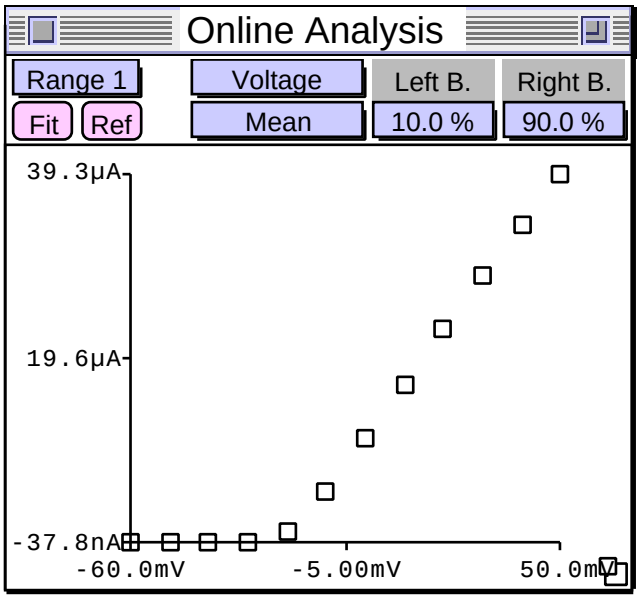
Before we execute the IV sequence (or*"Series"* in *PULSE* terminology, which describes a number of individual sweeps based upon the same *Pulse Generator Protocol*) we will set up the display: in the **Display** menu activate **Show Zero Line**, **Show Potential**, **Dimmed Overlay**, **Labelling → Labels Only** and **Display Mode → I vs. t**. To

see all sweeps from the series activate the **Overlay** button (3), otherwise, the *Oscilloscope* will be erased before every sweep. Now click on the **IV** button (4) or type "1", if *IV* is the sequence in the first position: The pulse pattern we defined above is output via the specified DA channel (blue color, 5) and the response is shown on the screen. The last sweep of the series is shown in black color, the other sweeps are gray (6) since we activated **Dimmed Overlay**.

The zero current line is drawn in green color (7) and scaling bars are given in the lower left side of the *Oscilloscope* (8). The two red lines (9) mark the range of the *Online Analysis* (see next section). You can make this range visible by turning on the **Keep** button (10).

Online Analysis

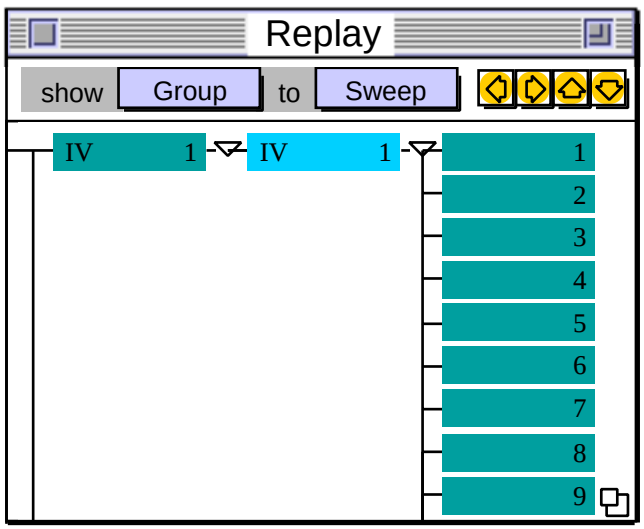
Based on the relevant segments (the depolarizing pulse) as specified in the *Pulse Generator*, a quick online analysis of the acquired (or replayed) data is performed immediately after the series has been collected entirely. The criteria for this analysis (i.e., type of analysis, time range, format of display) were specified before in the *Online Analysis* window, when we recorded the *SetIVMean* macro; now the result - an I versus V curve - is plotted in the graph inside the *Online Analysis* window. The results are also sent to the *Notebook* as a table. If you bring the *Notebook* window to the front (e.g. by selecting **Pulse** → **Notebook**), you should see something like the following:




```
Execute: IV
#   V(2)[mV]   t[ms]     i[A]
 1    -60.0    100.0  -38.289n
 2    -50.0    100.0  -36.313n
 3    -40.0    100.0  -34.633n
 4    -30.0    100.0   28.789n
 5    -20.0    100.0   1.0937µ
 6    -10.0    100.0   4.9068µ
 7      0.0    100.0   10.167µ
 8     10.0    100.0   15.724µ
 9     20.0    100.0   21.289µ
10     30.0    100.0   26.742µ
11     40.0    100.0   32.118µ
12     50.0    100.0   37.413µ
```

Storing Data

The structure of the stored data - if the **Store** button was active - will be shown in the *Data Tree* of the *Replay* window. To open it select **Pulse** → **Replay** or type 'F5' (Windows) or 'F13' (MacOS). Double-click the *"IV 1"* entry to replay the just recorded sequence; double-click a single sweep to watch it in the *Oscilloscope* window. You might use the cursor keys ('UP', 'DOWN', 'LEFT' and 'RIGHT') to walk through the data tree. If you press 'RETURN' the currently active group, series or sweep will be displayed in the *Oscilloscope* and the *Online Analysis* will be calculated.



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

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

Exit

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

User Interface

The following chapter describes the user interface of *PULSE+PULSEFIT*.

Control Types used by PULSE+PULSEFIT

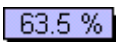
The items of the main dialog boxes and the dialog windows themselves can be changed by the user during program execution. The new layout is stored with the command **Front Dialog → Save**. It is written into the *Common Path* and the name of this file specifies the kind of window it describes (e.g, **Default.Pulse_OsciDialog** on the *MacOS* or **D-OsciP.dia** under *Windows*). Upon restart of *PULSE* these dialog files are loaded and they overwrite the default settings. If you remove them from the hard-disk, the old default settings will be used again.

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

In the following the functions of the different types of controls are explained; then it is shown how controls and windows can be modified.

Dialog Controls

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

Drag: 

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

Note1: When you enter a new value, do **never** include (or leave from the previous text) the unit character. The unit character (e.g. "m" for "meter") could conflict with the input of engineering units (e.g. "m" for "milli", see section "Numerical Input" below). Since the text is selected when the item get selected, you can just type in the new value and the old text gets replaced.

Note2: Sometimes the computer is very busy, i.e. during an Auto-CSlow compensation, and there might be not enough processing time to handle a double mouse click (the operating system will report two separate mouse clicks to the program instead). In this case you still will be able to activate a control by keeping the 'SHIFT' key pressed and then click the control once with the mouse (right mouse click under Windows).

List: 

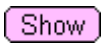
Similar in appearance to a *Drag* item. Clicking on it will pop up a menu list from which one can choose a setting.

Edit Text: 

A text string in a box with a drop shadow. Clicking on it will allow to edit the displayed string.

Button: 

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

Switch: 

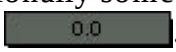
Rounded corner rectangle. Clicking on it toggles the parameter value. The switch is *On* or *Activated* if the item is highlighted 

. A *Switch* can optionally also execute some action.

Radio Button / Check Box: 

Identical to the standard dialog items. Clicking on them will toggle the respective parameters.

Framed Text / Number / Boolean: 

Simple box with optionally some text. The *Boolean* value is indicated by its color, inactive controls are gray 

.

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

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

Numerical Input: Numerical values can be entered either by dragging the value or by typing after double-click (or a 'SHIFT'-click) on the item. In the latter case the value can be entered in scientific notation (e.g., *"2.3e-3"*, *"2.3E-3"*) or in engineering format (e.g., *"2.3m"*). Numbers outside this range for engineering numbers (see table) are always displayed in scientific notation. The old value is erased as soon as the user starts to type. To preserve the old string, move the 'LEFT' or 'RIGHT' cursor first. To leave the previous value unchanged although a new one has been entered already, just clear the input by pressing 'ESC', then 'RETURN' or 'ENTER'. When you enter a new value, do **never** include (or leave from the previous text) the unit character. The unit character (e.g. "m" for "meter") could conflict with the input of the engineering units (e.g. "m" for "milli").

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

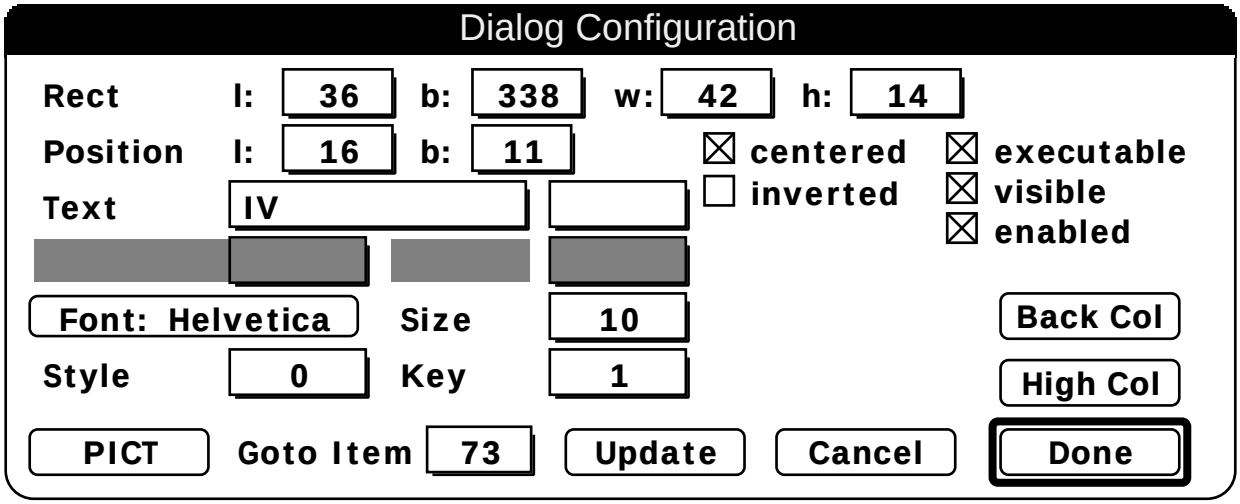
SI Units: Pulse expects most units to be SI-units, i.e., meters, seconds, amperes, volts, Hertz, etc. However, for convenience there are exceptions to that rule. In such cases the item title contains an identifier for what unit is to be used, e.g., mV (

) if a voltage is to be entered in millivolts rather than in volts.

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

Modifying the Dialogs and Controls

There's a "hidden" feature in *PULSE*: Dialog items can be modified after engaging 'CAPSLOCK' and then clicking on an item while pressing 'COMMAND' (MacOS) or 'CTRL' (Windows). This will bring up a dialog allowing to modify the item settings such as color, text font, dragging speed etc.:



Clicking on an item while pressing 'OPTION' (MacOS) or a right-click (Windows) will allow to drag and resize an item. The new item position will be ignored, if 'OPTION' is up when the mouse button is released (MacOS only). Dragging while both 'OPTION' and 'COMMAND' are down (MacOS) or a 'CTRL'-right drag (Windows), results in repositioning all items within the dragged item rectangle. The dialog is completely redrawn, after any item was modified.

The following table summarizes all actions:

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

Any item can have a character assigned to it. This enables to execute the item from the keyboard. Some dialog windows (those with a window bar) can be moved and resized as any regular window. Dialog windows can be iconized, i.e., reduced to a minimal size window. Such a window can be easily expanded to the original size (and shrunk again) by clicking in its zoom box.

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

All dialog windows can be opened, iconized, and closed using the function keys (see drop-down menu *Pulse*). The *Notebook* window can only be opened or closed but not

iconized. 'ESC' can be used to switch to the *Oscilloscope* window. In addition, 'ESC' will cause some windows to be closed (e.g., *Pulse Generator* window, *Configuration* window, *Spectra* window). Pressing the 'SPACE BAR' can be used to quickly toggle between the *Oscilloscope* window and the *Amplifier* window.

Tutorial: Changing the Size of the Oscilloscope

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

1. Increase the size of the *Oscilloscope* window. Make it at least as big that you see the "hidden" features in the bottom. If you plan to use serial communication, enable this feature in the *Configuration* window.
2. Activate the 'CAPSLOCK' key.
3. Click with the right mouse button (Windows) or while holding the 'OPTION' key down (MacOS) into the right part of the window (somewhere between the **Show P/n** and **Scan Data** button) and drag the whole group to right edge of the window. While moving you will see a gray rectangle underneath the set of controls.
4. Repeat the last step with the sequence pool and the serial button group. The buttons in the bottom of the window can not be moved together, therefore it's easier to move the group with the serial buttons below them.
5. Right click (Windows) or 'OPTION' click (MacOS) into the lower right part of the oscilloscope to resize it. If you want to resize it precisely you can also 'CTRL' click (Windows) or 'COMMAND' click (MacOS) it and enter the new size into the fields **w:** (width) and **h:** (height). If you increase the height reduce **b:** (bottom) accordingly.

Finally resize the *Oscilloscope* window, move it to the place you want to keep it and save it using the menu **Pulse → Front Dialog → Save**.

Note: There is a shortcut to quickly resize the oscilloscope window. Hold down the 'SHIFT' key while you resize the main window, and all controls will be moved automatically. To make that change permanent, save the new window organization with the menu **Pulse → Front Dialog → Save.**

Menus

This chapter describes the various drop-down menus in *PULSE+PULSEFIT*. The two parts *PULSE* and *PULSEFIT* have almost all menus in common, there is only one exception: Instead of *PULSE's* **Pulse** menu, that is used mainly for controlling data acquisition, *PULSEFIT* has a menu called **PulseFit** that allows the user to analyze the *PULSE* data. Also, the **Amplifier** menu is only available in *PULSE* since *PULSEFIT* is not able to record data. All menus get inactivated during acquisition of data. This is to prevent users to call functions which are not allowed during acquisition.

File Menu

The **File** menu has all options to handle *PULSE* experiment files. A single *PULSE "Experiment"*, that can hold a variable number of single electrophysiological experiments, consist of at least three files (see *Chapter Data Format* for a detailed description): the file with the extension ***.pgf** (= **P**ulse **G**enerator **F**ile) has the stimulus templates used (*Stim Tree*). The ***.pul** file has the complete data tree (*Pulsed Tree*), while the ***.dat** file has only the actual raw data without any timing or scaling information. In addition there might be a file ***.sol** that has the solution database (if solution timing has been activated) and ***.ana** created by *PULSEFIT* that contains the latest analysis. If the *X-CHART* extension to *PULSE* was active, there are two more files: ***.tree** with the *X-CHART* data tree and ***.grp** with the *X-CHART* data itself.

Raw data acquired by *PULSE* is only written to disk in the so called *Store* mode (i.e. if the **Store** button in the *Oscilloscope* window is engaged) and the pulse sequence executed has to defined as *write enabled* in the *Pulse Generator* window. The data are written to disk upon **completion of a sweep or a series** or during the acquisition in case of a continuous sweep. The structural information in the *Stim Tree* and *Pulsed Tree* are kept in RAM; they are stored to disk **only** when:

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

Files may be opened in three different ways:

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

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

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

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

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

Close (not available in *PULSEFIT*): Closes the actual experiment.

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

```
read-only file: "C:\HEKA\Data\Demo.dat"
length: 1231 kb; free disk space: 1428 Mb.
```

Disable Data File Caching (not available in *PULSEFIT*) **:** Normally Windows writes all data into a file cache first. Disabling this feature slows down storing of data on the benefit of more safety. Therefore, the *Data File Cache* is disabled by default - in case of a computer crash the data in the cache would be lost otherwise.

Disk Write Options: Defines when the raw data are flushed to disk:

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

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

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

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

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

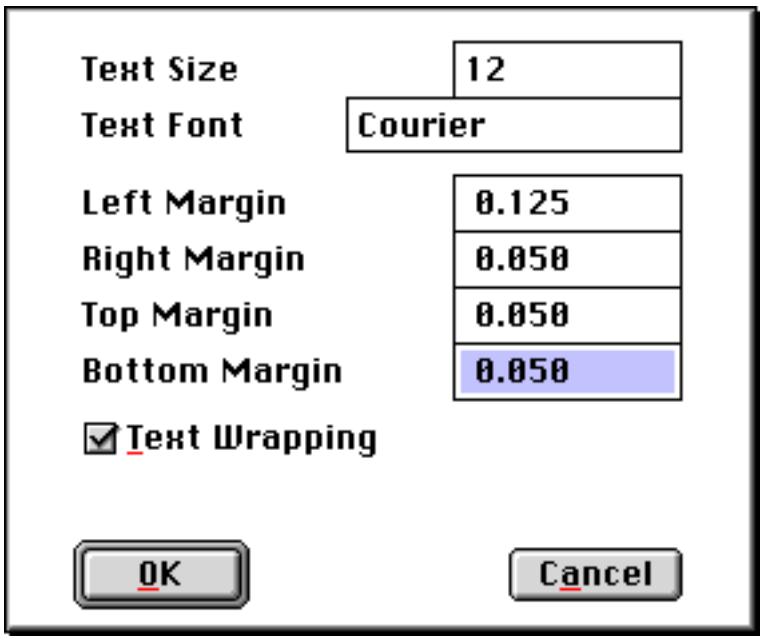
Convert To Native: Converts the raw data of the opened experiment (the ***.dat** file) into the native format of the target machine (Intel format = *Little Endian* or Motorola format = *Big Endian* or Intel format). It is usually not necessary to convert the data into the native format, since all versions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

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

Convert ST-Files... (not available in *PULSEFIT*) : Converts data files acquired by the Atari-ST data acquisition programs *E9SCREEN.PRG* and *ACQUIRE.PRG* into a *PULSE*-compatible format.

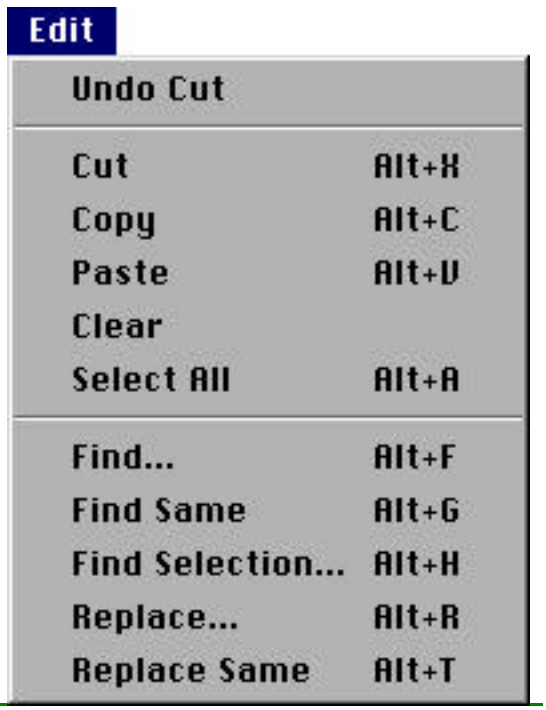
Page Setup... : Calls the *Printer/Page Setup* dialog of the operating system.

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



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

Quit: Exits *PULSE+PULSEFIT*.



Edit Menu

The *Edit* menu applies to text manipulation in the *Notebook* window (see *Chapter Notebook*). The menu is normally disabled unless the *Notebook* window is in front. The menu entries conform to the typical functions of the actual operating system (MacOS or Windows).

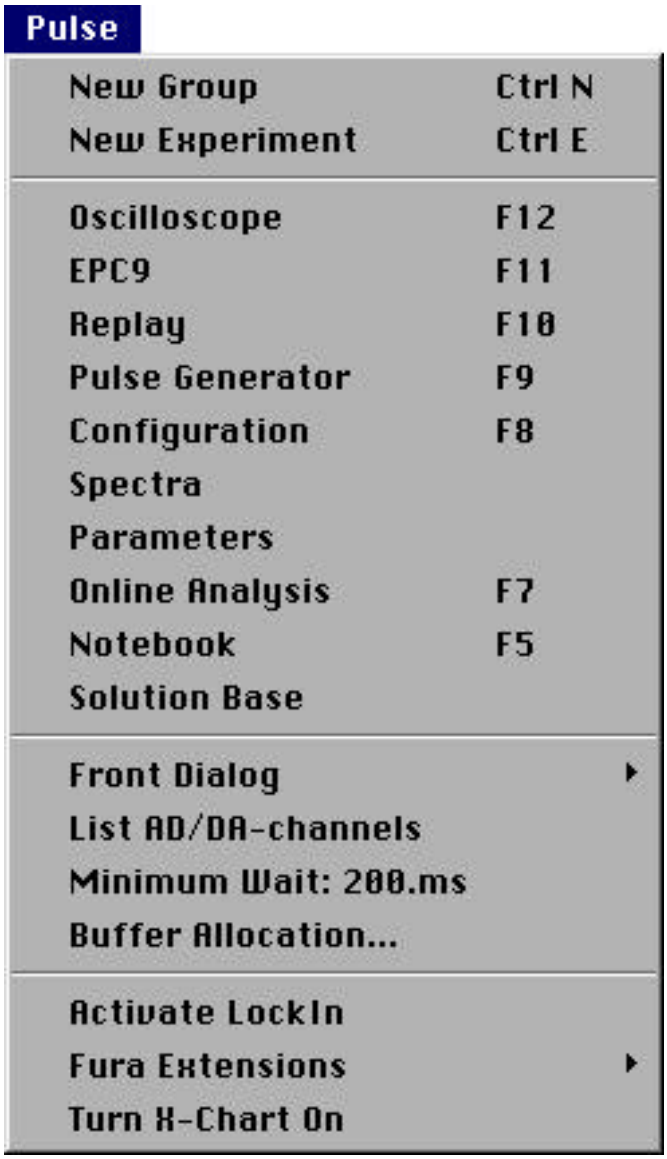
The **Cut**, **Copy**, and **Paste** commands copy the text selection to and from the clipboard.

Clear removes the text. **Select All** selects the whole notebook.

Find... , **Find Same**, **Find Selection...** , **Replace...** , and **Replace Same** perform the obvious functions.

Pulse Menu

This menu is only available in the *PULSE*. Section of *PULSE+PULSEFIT*. It contains all items used for data acquisition by *PULSE*.



New Group: Generates a new *Group* in the output *Data Tree* of the *Replay* window when a file is opened without write protection. After the addition of a new group an automatic file update is performed (see *File* Menu).

New Experiment: Generates a new *Group* (see above) and increments the internal experiment number (see *Chapter Configuration*).

Oscilloscope: Opens the *Oscilloscope* window or brings it to the front (see *Chapter Oscilloscope*).

EPC9 / EPC8 / EPC7 / Axon-200 / Amplifier: Opens the *Amplifier* window or brings it to the front. The title of the menu item reflects the amplifier set in the *Configuration*.

Replay: Selects the *Replay* window (see *Chapter Replay*).

Pulse Generator: Selects the *Pulse Generator* window (see *Chapter Pulse Generator*).

Configuration: Selects the *Configuration* window (see *Chapter Configuration*).

Spectra: Selects the *Spectra* window (see *Chapter Power Spectra*).

Parameters: Selects the *Parameters* window (see *Chapter Parameters*).

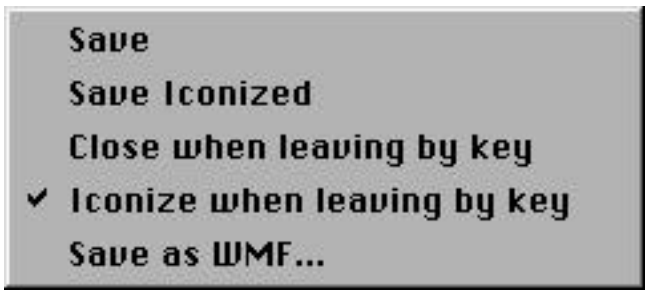
Online Analysis: Selects the *Online Analysis* window (see *Chapter Online Analysis*).

Notebook: Selects the *Notebook* window (see *Chapter Notebook*).

Solution Base: Selects the *Solution* data base (see *Chapter Solutions*).

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

- **Save**: Stores the settings of the active window to a file. E.g. the settings of the *Oscilloscope* window are written into the file **D-OsciP.dia** (Windows) or **Default.Pulse_OsciDialog** (MacOS). These files are read upon the next restart of *PULSE+PULSEFIT* and the window settings are set accordingly.



- **Save iconized**: Stores the settings to a file in iconized form (i.e., upon next restart of *PULSE+PULSEFIT* the window will appear iconized*)*.
- **Close when leaving by key**: Closes the window when the close box of the window is clicked or the dialog is exited using the 'ESC' key (note: the *Notebook* window can not be closed by key and the *Oscilloscope* window can not be closed at all).
- **Iconize when leaving by key**: Hides the window but leaves the window bar on the desktop (MacOS) or iconizes the window (Windows).
- **Save as Pict**…: Saves the window dialog as a MacOS PICT file.
- **Save as WMF**…: Saves the window dialog as a Windows metafile.

Note: If you encounter any problems with the program it is a good advice to save the corresponding front dialog (e.g. the Pulse Generator window) as a PICT or WMF file and send it together with the description of your problem to the HEKA hotline.

Pulse Help: Opens a *Help Text* window.

Show Keys: Shows the keyboard shortcuts assigned to the window controls.

Hide Keys: Hides the keyboard shortcuts assigned to the window controls.

Note: The Windows version shows the menu items ***Pulse Help***, ***Show Keys*** and ***Hide Keys*** in a separate ***Help*** menu.

List AD/DA-channels: Prints the AD/DA assignments to the *Notebook*. This is convenient to check the current hardware environment for possible collisions. A sample output could look as follows:

```
DA channels:  V-membrane DA: EPC9
                 Trigger DA: off
Pip. Pressure DA: off
    CapTrack Out - CSlow DA: 0      GSeries DA: 1
AD channels:     Current AD: EPC9
      Current In, VClamp AD: 5
                 Voltage AD: 0
Fura channels:    Filter DA: 1       Signal AD: 1
```

Minimum Wait: *Wait time* is the time in between pulses. Only during this time window *Cap. Track* and *X-CHART* functions can be executed, provided there is time available. The *Minimum Wait* time defines how much time must be available at least. An entry of *0 ms* will ensure that these functions are executed at least once in between sweeps. The drawback is that time overruns may occur when using closely spaced stimuli.

Buffer Allocation: This setting determines the total amount of RAM that *PULSE* allocates for temporary data storage. MacOS users should also refer to the section entitled *Memory Requirements of the MacOS Version*.

Activate LockIn: This activates the *LockIn* extension, which is a software implementation of a Lock-In amplifier for measuring membrane capacitance (see *LockIn Manual* for details). After activating the extension *PULSE* must be restarted. The title of the menu item changes to **LockIn Configuration** afterwards.

Fura Extensions: This activates the *FURA* extension, which controls photometric and/or fluorescence data acquisition, e.g., for measuring the intracellular calcium concentration (see *FURA Manual* for details). The *FURA* extension is a separate product available for *PULSE+PULSEFIT*, please contact HEKA for more information. At present, the following systems are supported:

- Generic: This is any monochromator that can be contrlled either via an analog voltage or a digital output. This option works with the *ITC-16*, *ITC-18* or *Digidata 1200* interface board.
- *Olympus OSP3* system
- *Olympus OSP100* system: The two supported *Olympus* systems require a *GPIB* interface board together with the corresponding drivers from *National Instruments*.

- *pti DeltaRAM*
- *pti DeltaScan:* The *pti* systems can be controlled via an analog output of the amplifier or any AD/DA converter supported by *PULSE*.
- Random Data: This option is for demo and testing purposes only.
- *Sutter Lambda-10* and *DG-4:* The *Sutter filter wheel Lambda-10 and the monochromator DG-4* can be controlled via the digital output of the *EPC9*, *ITC-16* or *ITC-18*.
- *T.I.L.L photonics:* The *T.I.L.L. photonics* system can be controlled via an analog output of the amplifier or any AD/DA converter supported by *PULSE*.

Video Extensions (Windows 95 and NT only): This activates the*Video* extension, which controls synchronous video and data acquisition, e.g., for measuring the intracellular calcium concentration (see *VideoPatch Manual* for details). The *Video* extension is a separate product available for *PULSE+PULSEFIT*. Please contact HEKA for more information. At present, the DVP-32 video processor from Instrutech Corp. is supported.

X-Chart Extension: This activates the *X-CHART* extension, which is a software implementation of a multi-channel chart recorder (see the *X-CHART Manual* for more details). The *X-CHART* extension is a separate product available for *PULSE+PULSEFIT*, please contact HEKA for more information.

PulseFit Menu

This menu is only available in the *PULSEFIT*. Section of *PULSE+PULSEFIT*. Most of the menu items are identical to the items in the **Pulse** menu, see the preceding section *PULSE Menu* for a more detailed explanation.

Oscilloscope: Selects the *Oscilloscope* window.

Replay: Selects the *Replay* window.

Pulse Generator: Selects the *Pulse Generator* window.

Configuration: Selects the *Configuration* window.

Parameters: Selects the *Parameters* window.

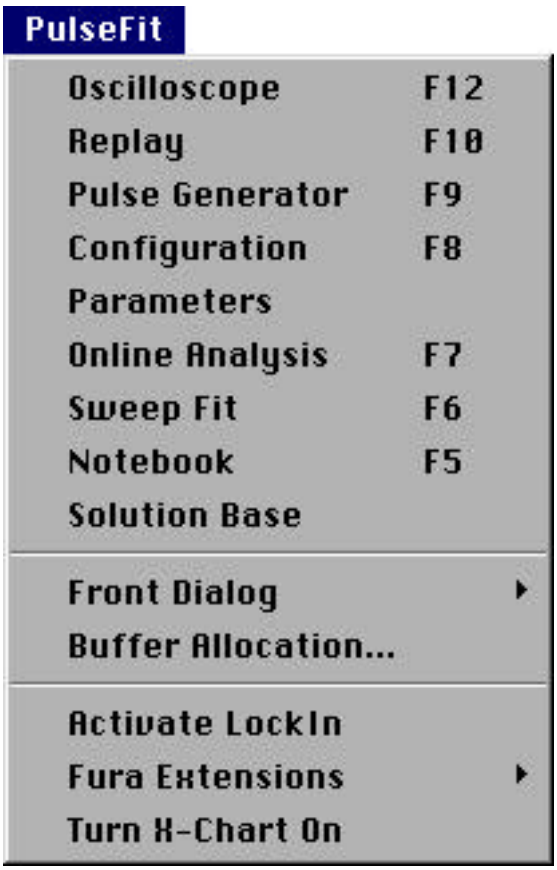
Online Analysis: Selects the *Online Analysis* window.

Sweep Fit: Selects the *Sweep Fit* window (see *Chapter Sweep Fit*).

Notebook: Selects the *Notebook* window.

Solution Base: Selects the *Solution* data base.

Front Dialog: This menu items do the same as the corresponding ones from the **Pulse** menu.



Buffer Allocation: This setting determines the total amount of RAM that *PULSEFIT* allocates for temporary data storage.

Activate Lock-In: This activates the *LockIn* extension. If you have recorded data using the *LockIn* extension to *PULSE* you must also enable the extension in the *PULSEFIT* section of the program, if you want to see the *LockIn* results.

Fura Extensions: This activates the *FURA* extension. If you have recorded data using the *FURA* extension to *PULSE* you must also enable the extension in the *PULSEFIT* section of the program, if you want to see the calcium data associated with the recorded sweeps.

Video Extensions: This activates the *Video* extension. If you have recorded data using the *Video* extension to *PULSE* you must also enable the extension in the *PULSEFIT* section of the program, if you want to see the results associated with the recorded sweeps.

X-Chart Extension: This activates the *X-CHART* extension. If you have recorded data using the *X-CHART* extension to *PULSE* you must also enable the extension in the

PULSEFIT section of the program, if you want to see the chart data associated with the recorded groups.

Tree Menu

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

Show: Displays the content of the selected target. Traces are displayed according to the settings specified in the *Oscilloscope* window. If the target for the *Show* operation is a *Group*, a *Replay Group* dialog will ask to stop or to continue with the next series. *Do All* causes to go through all series of this group; it can be aborted by pressing 'CTRL' + 'B', 'CTRL' + 'S', or mouse click on the **Break** and **Stop** buttons in the *Oscilloscope* window. A corresponding dialog will come up, if the target is the *Root* or a *Group*.

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

Export Full Sweeps: Exports the full sweep independent of what is shown in the *Oscilloscope*. However, display gain, leak-subtraction or zero-line subtraction will be applied (special case for *Igor Binary* export see below).

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

If the target is a *Series*, the results of the last *Online Analysis* of this series is shown as *Reference Analysis*. The scaling of the graph reflects the extreme values of both, the reference and actual series analysis (unless the *Fixed Scaling* of the *Online Analysis*

window option is used). The reference series can be turned off by pressing the highlighted **Ref** button in the *Online Analysis* window.

Wipe Screen: Clears the *Oscilloscope* screen.

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

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

Amplifier State: Prints the content of the *EPC9 State Record* (which reflects the hardware status for the current series) into the *Notebook*.

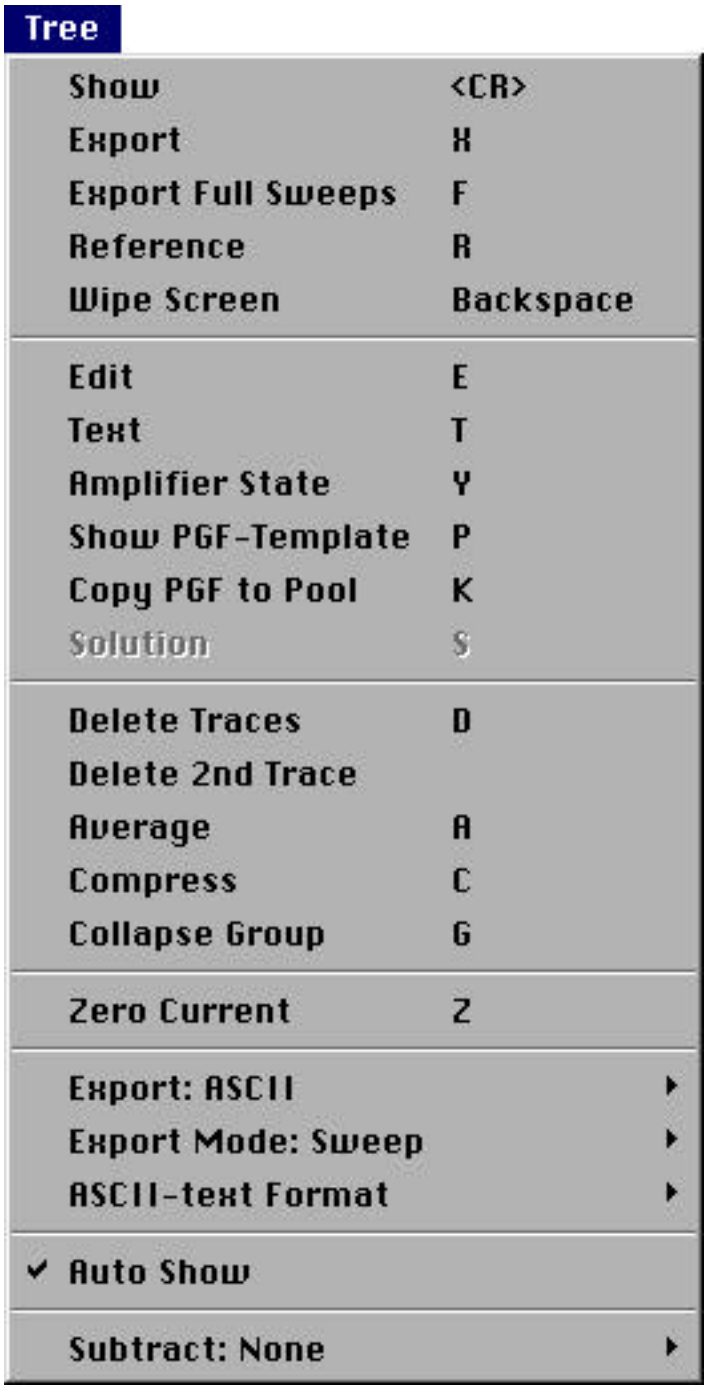
Show PGF Template: Displays the stimulus protocol of a selected sweep or series.

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

Solution: Allows editing the solution stored for the current series.

Delete Traces: Deletes all traces of the target (i.e., *1st*, *2nd*, and *Leak Trace*).

Delete 2nd Trace: Deletes the *2nd Traces* of the target.



Average: Performs averaging of target:

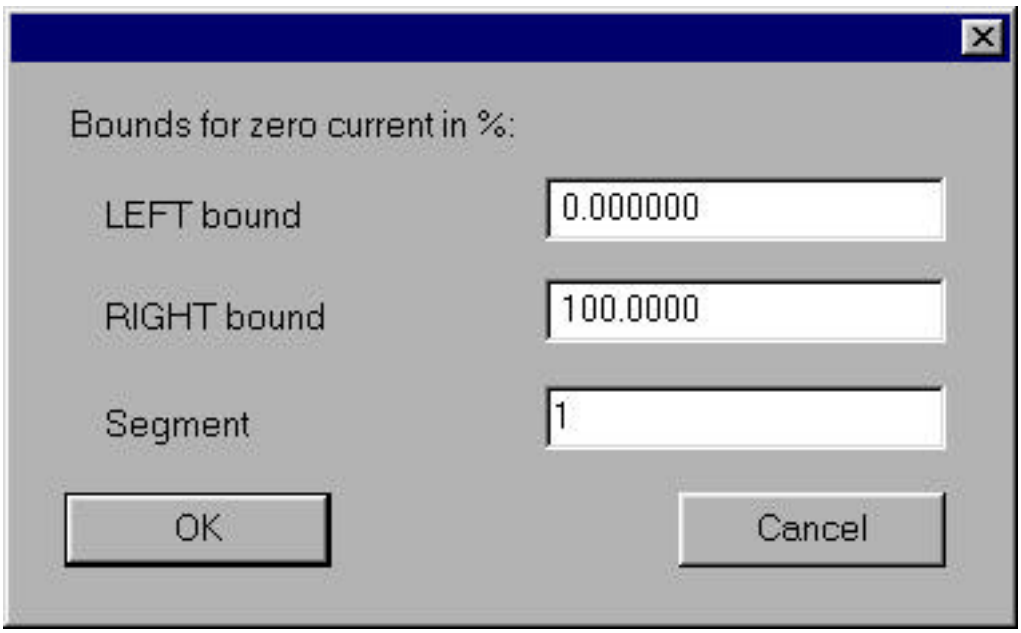
- **Series**: Averages all sweeps of a series and stores them as series with one sweep. The sweeps have to have the same length.

- **Group**: It is assumed that all series within the selected group are of the same kind. One output series is created with the sweeps being the averages of all matching sweeps within the group (e.g., all first sweeps are used to generate the new first sweep, all second sweeps generate the new second sweep).

Compress: Compresses all sweeps of the selected target by a factor of two by taking the mean of each two successive data points. A warning is given when the number of data points in a pulse segment is not a multiple of two.

Collapse Group: Allows to move the sweeps of all series in a group into the first series. This is typically used when one acquired many series with one single sweep and one wants to combine them into one series for easier online analysis. A typical situation arises when one needs the "Start Macro" to perform some action like opening perfusion valves or setting specific amplifier gains for every Sweep.

Zero Current: Recalculates the zero current within a specified section of a given pulse segment (a dialog will ask for the segment and the section). Normally the zero current is calculated automatically within the full segment. This means that artifacts in the baseline will cause the zero current to be incorrect. This feature can therefore be used to recalculate the zero current in a section of the baseline without artifact. If there is no such section left, one can edit the entry *Zero Current* in the *Tree* (see *Chapter Replay*) by taking the zero current of the next or previous sweep, for example.

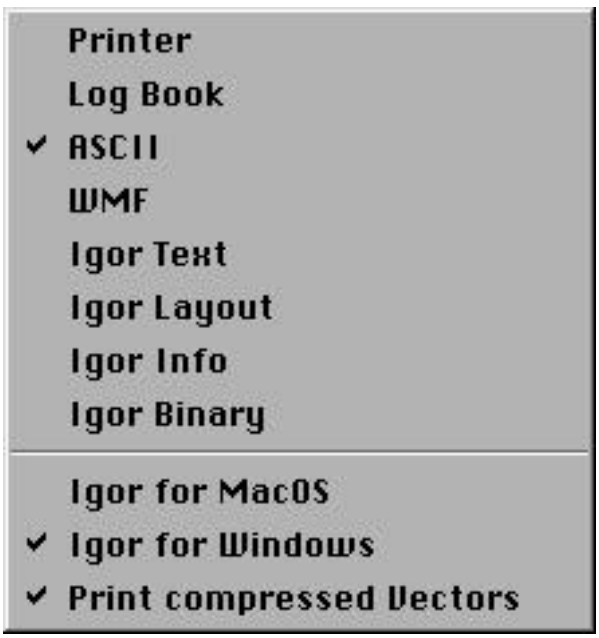


Export Format: This determines the output device and the type of output to be created. Output is generated in the way the data are displayed in the *Oscilloscope* window; e.g., when the digital filter is on, filtered data are output.

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

The following options are implemented:

- **Printer**: Direct output to a connected printer. The page setup magnification determines the line width; usually, 50% gives good results. The number of columns and rows per page have to be entered. This determines how many items are placed on a page. In any case, a form feed is output after the selected target is output. Thus, if one prints a group with 3 series to a page with 2 columns and 2 rows, three quarters of the page will be filled, then the page is released from the printer. If one wants to have individual sweeps rather than a complete series plotted in the page sections, one has to turn off *Overlay* in the *Oscilloscope* window.



- **Log Book**: The information stored in the respective branch of the *Tree* is written to an ASCII file. Each entry is identified by an ASCII string. This is intended to replace (or at least complement) a conventional notebook.
- **ASCII**: Sweeps are output as columns of ASCII numbers representing time and current (both in the scientific format). Each sweep and series starts with an identifier.

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

- **PICT**: Sweeps are exported as MacOS PICT file. Because the PICT files are limited to 32 Kbytes each file only contains a single sweep (plus leak, second trace and other options). When a series is output, the sweep files are generated automatically with the same name convention as waves for *IGOR* files: indices of "Group_Series_Sweep" are appended to the name.
- **WMF**: Sweeps are exported as Windows Meta Files.
- **Igor Text**: Export of sweeps as ASCII waves in "IGOR Text" format for the analysis and display program IGOR. Each wave is identified by indices "Group_Series_Sweep" (e.g., "Name2_4_3"). If the file name starts with a number, a "W" is placed in front of it, because in *IGOR*, waves are not allowed to start with a number. The leak currents have the identifier "_Pn" attached to the wave name. The created files have the extension "IGO" and are recognized by *IGOR*, i.e., double-click on this file will start *IGOR*, load and display the file content (not for *Sweeps*). The waves will immediately be displayed in *IGOR* only if a series or group was exported, and the *Overlay* or *Overlay All* option was selected in the *Oscilloscope* window during data export. Otherwise, the sweeps will be loaded, but must be displayed by *Igor's* "Display Wave" command. To export the stimulus pattern, select the *Show Stimulus* option in the *Display* drop-down menu. When loading *IGOR Text* output files, do not use the "General Text" import option in IGOR; always use the option *Load…IGOR Text*.
- **Igor Layout**: Export of sweeps as ASCII waves in "IGOR Text" format, arranged on an *IGOR* layout page. The sweeps are displayed in *IGOR* graphs as they appear in *PULSE* in the *Oscilloscope* window. The created files have the extension "IGL" and are recognized by *IGOR*, i.e., double-click on this file will start *IGOR*, load and display the file content.
- **Igor Info**: Export of pulse protocols as ASCII waves in "IGOR Text" format. There are two waves generated per sweep: "Amp" and "Dur", concatenated to the sweep identifier. The created files have the extension "INF".
- **Igor Binary**: Export of sweeps to *IGOR* as binary data. This function generates an *IGOR* macro which contains the instructions for IGOR on how the data are to be loaded, scaled, and displayed. It has the extension ***.IGB.** A double-click on it will make IGOR load that macro file and execute the instructions in it, importing, scaling, and displaying the data. The actual data are not really exported, when using the Exprot Full Sweep. That option will make use of the *"GBLoadWave" IGOR* extension to read the data directly from the *PULSE* raw data file, i.e., the ***.dat** file. waves. The data are converted to IGOR binary waves, when the simple *Export* option is used. But remind, even in that case, a macro file is generated and you should load the data via that file. When you want to import data from within IGOR, use the option ***Load…IGOR Text.*** to **load the macro file**. Use the option

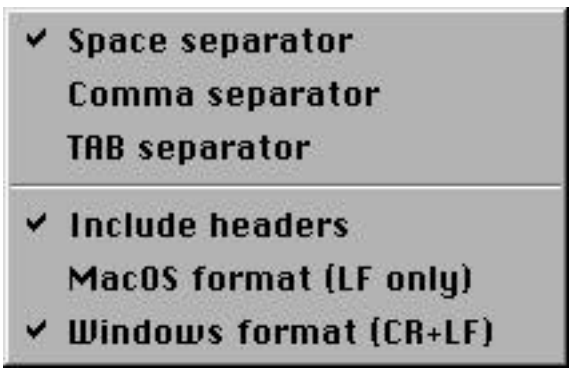
Load…IGOR Binary only when you want to explicitly load one of the generated IGOR binary waves (file extensions ***.ibw** or ***.bwav).**

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

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

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

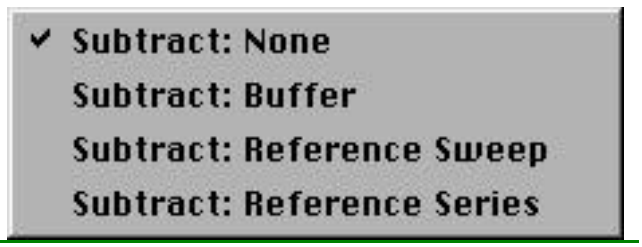
ASCII-Text Format: The ASCII export allows easy concatenation of, e.g., on-line analysis results from different sweeps and series. The resulting table can be directly imported, e.g., into *IGOR*. **ASCII-Text Format** brings up a submenu that allows to specify the type of separator used when generating ASCII tables and for selecting the format of the exported text:



- **Space/Comma/Tab Separator**: Specifies how values are separated.
- **Include Headers**: If checked, a header which specifies various parameters of the exported data will precede the actual values.
- **Mac Format**: Lines are terminated by line feeds. This generates standard MacOS text files.
- **DOS Format**: Lines are terminated by line feeds plus carriage returns. This generates text files which can be read by programs running under DOS or Windows without need for file conversion.

Auto Show: If this flag is on, sweeps are shown as soon as the corresponding sweep target is highlighted; hitting 'RETURN' is not required.

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



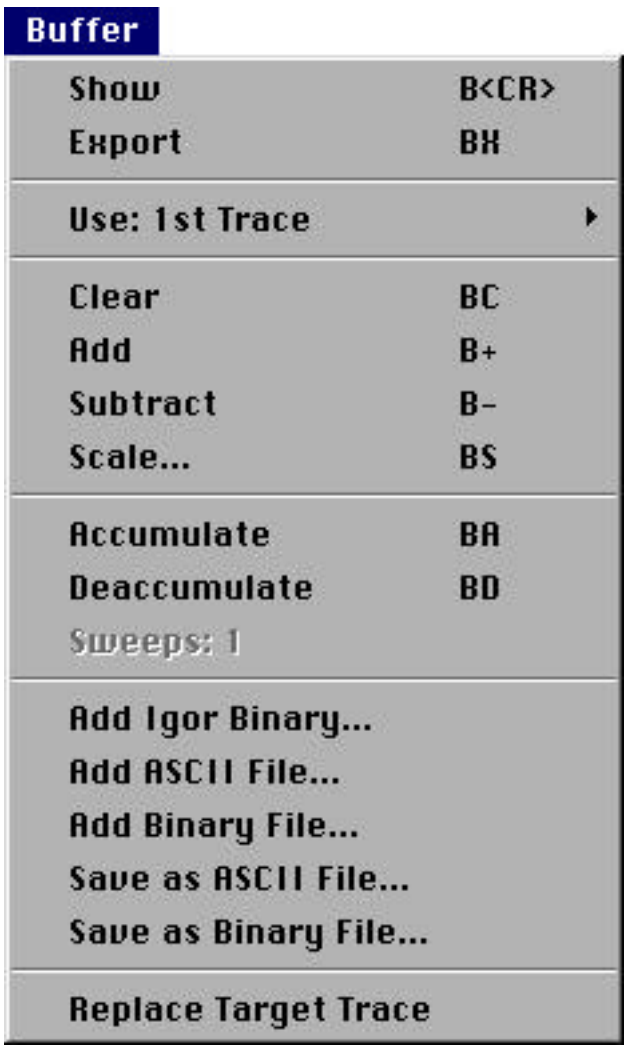
- **None**: No subtraction.
- **Buffer**: The *Buffer* content is subtracted.

- **Reference Sweep**: The *Reference Sweep* is subtracted.
- **Reference Series**: The *Reference Series* is subtracted, i.e., corresponding Sweeps from the two Series are subtracted: the 1. Sweep of the *Reference Series* is subtracted from the 1. Sweep of the selected Series in the Tree, then the 2. Sweep of the *Reference Series* is subtracted from the 2. Sweep of the selected Series, etc.

Buffer Menu

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

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



Show: Shows buffer content.

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

can be useful, e.g., to edit a recorded sweep.

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

- 1st Trace
- 2nd Trace
- P / n Leak

Clear: Sets buffer to zero.

Add: Adds sweep to buffer.

Subtract: Subtracts sweep from buffer.

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

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

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

Sweeps: Number of sweeps currently accumulated in the buffer.

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

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

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

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

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

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

Marks Menu

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

Unmark: Removes the tag from the *Tree* target.

Mark: Attaches a tag to the *Tree* target.

Mark by Name... : Marks all sweeps from a certain PGF template. You will be asked for the name of the template in the sequence pool.

Show All: Displays all marked entries.

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

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

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

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

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

Delete All Traces: Deletes all traces and sweeps of all marked entries. This option is only available when the file is opened as modifiable.

Delete All 2nd Traces: Deletes the second traces of all marked entries. Only available when the file is opened as modifiable.

Average All: Averages all sweeps in a marked series or all series in a marked group. The original sweeps will be replaced by this average. Only available when the file is opened as modifiable. It is identical to the one found in the *Tree* menu, but will operate on all marked items.

Compress All: Compresses all sweeps in a series or all series in a group by averaging two adjacent data points. Each compression reduces the data points by half. The original sweeps will be replaced by compressed sweeps. Only available when the file is opened as modifiable. It is identical to the one found in the *Tree* menu, but will operate on all marked items.

Lock-In: Export to Igor: Exports the *LockIn* results (i.e. membrane capacitance, series resistance, … calculated from sinewave segments) to an IGOR file.

Lock-In: Export as ASCII: Exports the *LockIn* results to an ASCII text file.

Display Menu

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

Show Zero Line: Draws a reference zero line.

Show Potential: Displays the stimulus template in the background.

Dimmed Overlay: Turns the *Dimmed Overlay* mode on or off.

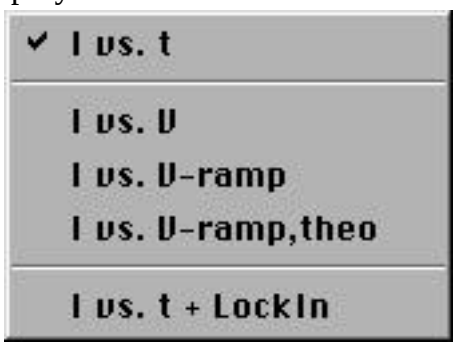
Labeling: Determines the labels in the *Oscilloscope* and *Pulse Generator* stimulus template.

- **Labels Only**: Draws calibration bars.
- **Grids + Labels**: Draws a grid and units/division.
- **Grids + Values**: Draws a labeled grid.
- **PGF-Editor Grid**: Draws a grid in the *Pulse Generator* stimulus template.

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

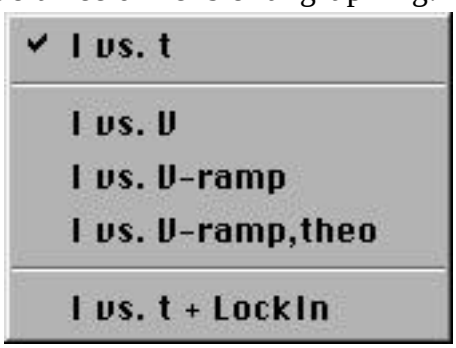
- **Off**: No background trace.
- **Reference Sweep**: Draws a trace marked as reference.
- **Running Average**: Draws the running average of acquired sweeps.
- **2nd Trace**: Draws the second trace.
- **Sweep Buffer**: Draws the buffer contents.

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



- **I vs. t**: Plots current versus time.
- **I vs. V**: Plots current versus measured voltage (i.e., 1st trace versus 2nd trace where the second trace is assumed to be the voltage trace).
- **I vs. V-ramp**: Plots current versus measured voltage during ramp segments. It is assumed that the second trace is the voltage trace.
- **I vs. V-ramp,theo**: Plots current versus theoretical ramp voltage.
- **I vs. t + LockIn**: Plots current versus time plus corresponding membrane capacitance calculated by the *LockIn* extension, if activated and the sweep contains at least one sinewave or squarewave segment.

3D Mode: Shows a submenu for selection of the pseudo three dimensional graphing.



- **3D-Graph: Enter dX and dY**: Allows to specify the horizontal and vertical offset of subsequent sweeps.
- **3D-Graph: On**: If selected, sweeps will be displayed in pseudo three dimensional mode by displaying subsequent sweeps with a horizontal and vertical offset.

Show Timer: Enables the timer in the *Oscilloscope* window. The timer is displayed in the top, right corner of the *Oscilloscope* window. The timer value at time of acquisition is stored in the series parameter block and is recalled during series replay.

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

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

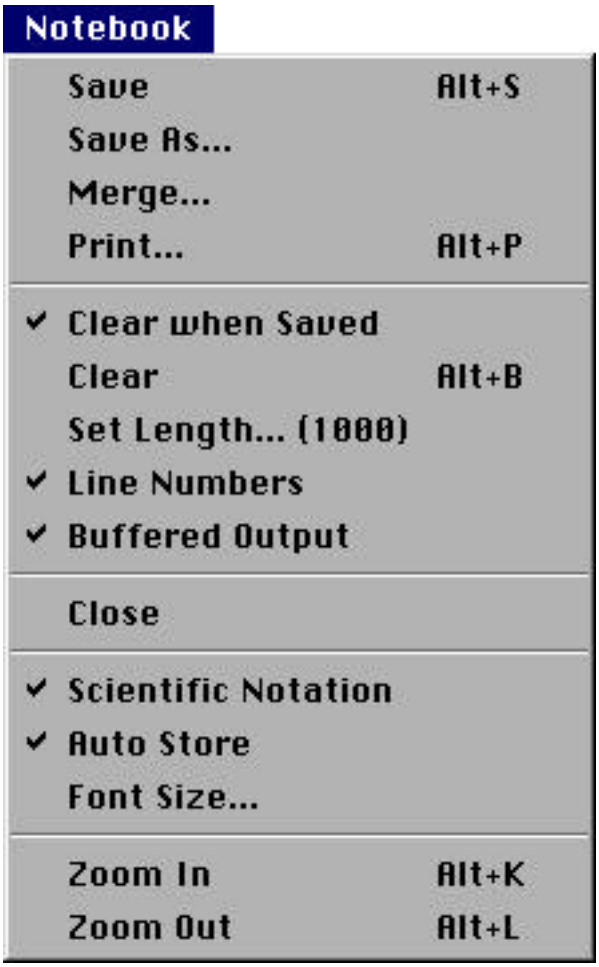
Series Info: Toggles output of series parameters to *Notebook* window.

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

Notebook Menu

The *Notebook* keeps track of information about the experiment. The ASCII-table sepa-

rator setting of the **Tree** menu is used for the *Notebook* as well. This enables to directly "cut-and-paste" to spreadsheets which require a *Tab* separator, such as Microsoft Excel. The options in the **Notebook** menu are:



Save: Saves the *Notebook* under its default name: "Notebook_Date".

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

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

Print...: Output content of *Notebook* to a printer.

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

Clear: Clears *Notebook*

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

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

Line Numbers: Shows line and column numbers.

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

Close: Closes *Notebook* window.

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

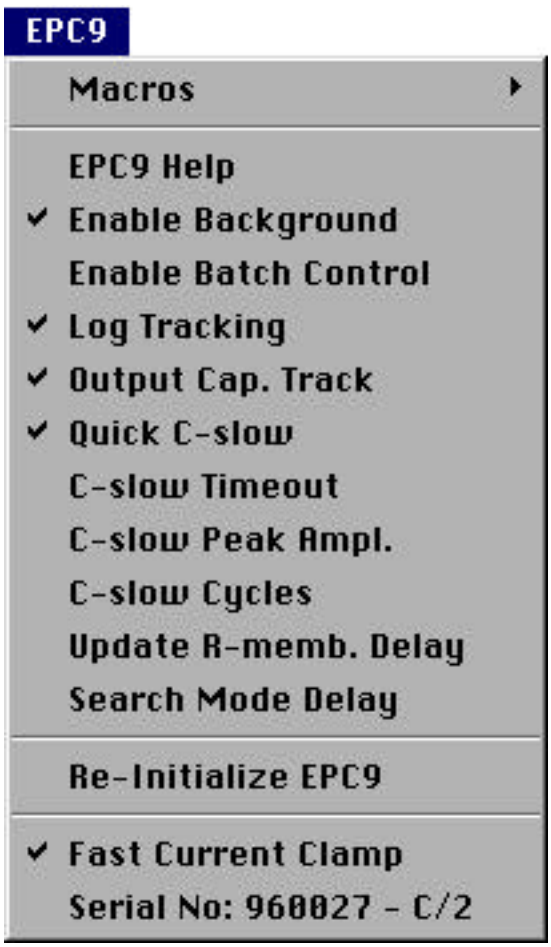
Auto Store: This option will automatically store the notebook together with the data file ([data file name].txt). Upon opening a data file, its *Notebook* file will automatically be loaded as well

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

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

Amplifier (EPC 7, EPC 8, EPC 9) Menu

The *Amplifier* drop-down menu provides functions to handle the *Amplifier* dialog, macros, and special functions (see *Chapter EPC7 Amplifier* and *Chapter EPC9 Amplifier*).



Macros: Selects a submenu for macro functions.

- **Load**…: Loads a saved macro file.
- **Save**…: Saves a macro file.
- **List**: Lists the current macros to the *Notebook* window.
- **Start Recording**: Starts macro recording.
- **Stop Recording**: Stops macro recording.
- **Execute while recording**: This option allows to record a macro without actually setting the parameters. Thus, during macro recording, commands and values are recorded, listed in the *Notebook*, and then the parameters are immediately set back to their previous value.
- **Execute**…: Executes a selected macro.

EPC 9 Help: Enters the online *Help* mode.

Enable Background: This allows to keep the test pulse running and thus *I-Pipette*, *V-Monitor*, and *R-Membrane* being updated even when *PULSE* is in the background.

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

Log Tracking: Writes the updated values of *C-slow*, *G-series*, and *G-leak* to the *Notebook* when capacitance tracking is on.

Output Cap. Track: Outputs the values of *C-slow* and *G-series* as analog voltages to DA-0 and DA-1, respectively.

Quick C-slow: Selects an alternate procedure for *Auto C-slow* compensation. It speeds up repetitive compensations, as it does not switch the gain, filter settings, etc. This option does not check as thoroughly for proper convergence and will sometimes adjust *C-slow* and *R-series* when the normal *C-slow* compensation would have returned an error. Use this option with caution.

C-slow Timeout: The time after which *PULSE* will stop trying to perform an *Auto C-slow* compensation.

C-slow Peak Amplitude: The amplitude of the voltage pulses used to compute *C-slow* and *G-series*.

C-slow Cycles: The number of voltage pulses used to compute *C-slow* and *G-series*.

Update R-memb. Delay: This option defines the delay between two measurements and updates of *I-pipette* and *R-membrane*. To prevent measuring *I-pipette* and *R-membrane*, enter a large value. The *I-pipette* and *R-membrane* computed from an acquired sweep as well as the computing of these parameters while in the amplifier window is not affected by the **Update Rmemb. Delay** value.

Search Mode Delay: The *Search Mode* (see *Chapter Operating Modes* in the *EPC9* Manual) is essentially a repetitive *Auto-*V_0 procedure at a holding potential of 0 mV. The delay specifies at what rate *Auto-*V_0 is performed.

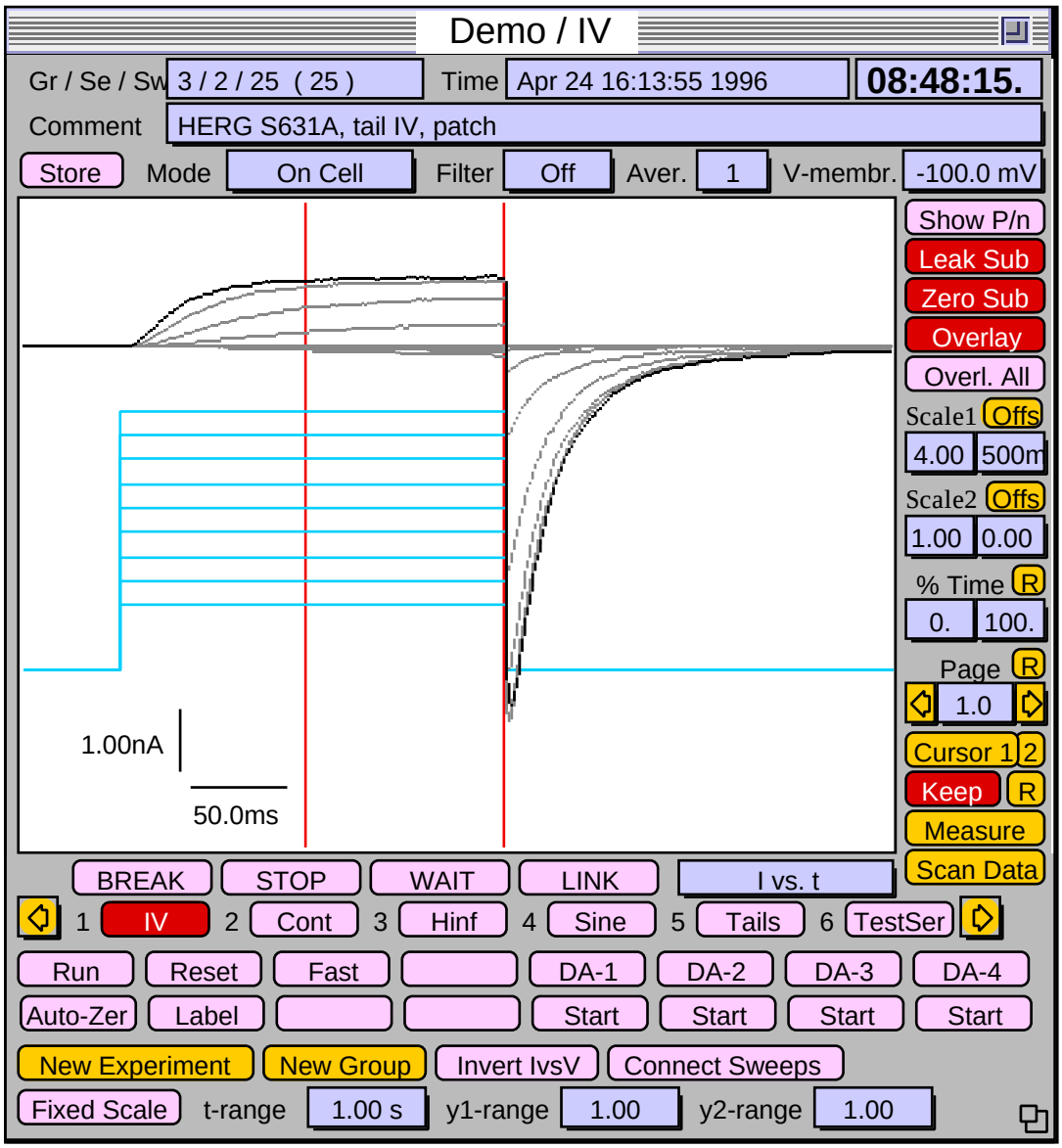
Re-Initialize EPC9: This is used to restart the AD/DA interface; e.g., in the case when *PULSE+PULSEFIT* was started with the interface being turned off.

Fast Current Clamp: This activates the *Fast Current Clamp* mode of the *EPC9* board version 'C' or later.

EPC9 Version: Tells the serial number (**960027**), board version (**C**) and number of amplifiers (**2**) of the amplifier, if an *EPC9* with board version 'C' or later is used.

Oscilloscope

The *Oscilloscope* window is mainly used for monitoring the data; controls for display scaling and data handling are provided. The title of the window contains the information on the currently active data file and the currently active series.



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

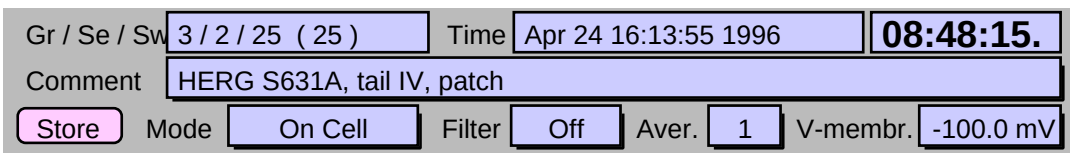
It is possible to set the display scaling in the oscilloscope window by "lasso-ing" a screen region. When the mouse button is released, the marked area will be set to fill the oscilloscope screen. Selecting while the 'OPTION' (MacOS) key is pressed will only change the display gain of the second trace (the X-scaling is not changed). If the mouse is dragged outside the active screen area, the display gains get reduced by 20%, until the mouse is moved back on the active display area. When the mouse is below or above the screen, the X-scaling is changed, and when the mouse is to the left or right of the screen, the Y-scaling is changed. To abort the dragging without changing the scaling (or restoring the previous scaling, if the mouse was dragged outside the screen), press any other modifier key (i.e., 'SHIFT', 'CONTROL', or 'COMMAND') before releasing the mouse button. Selecting the scaling region with the 'SHIFT' key down will re-size both, the first and the second trace.

There are multiple cursor shapes to accommodate different user preferences: while the "cross" cursor is displayed (when setting cursors or selecting the *Measure* option) it is possible to cycle through all available "cross" cursor shapes by pressing the 'CTRL' key.

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

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

Information about the Experiment



Gr/Se/Sw: Currently active group, series and sweep number within the *Data Tree*. The total number of sweeps per series is given in parentheses.

Time: Date and time of series (or test pulse) execution.

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

stored at the beginning of the sweep acquisition. The internal *Timer* tick corresponds to 1 ms.

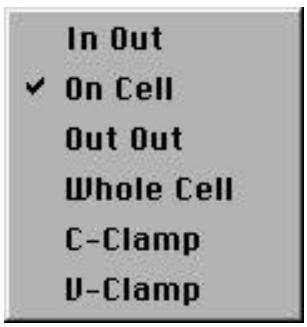
Comment: Comment to the currently active series. This field can be edited. It will result in a modification of the text of the present series, if a file was opened with write permission. This means that one has to enter a text to a series **after** it has been acquired. The new comment will be copied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

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

Note: There is no way to retrieve data that were acquired with the ***Store*** control being off. **If one is uncertain of whether to keep acquired data or not, one is advised to keep *Store* on and to remove unwanted data from the Data Tree later.**

Mode: Determines the *Recording Mode*. In *In Out* and *On Cell* mode the stimulus and the acquired data are automatically inverted. This applies to all voltages (including *V-membrane* and the pulse protocols). As for the liquid junction potential *LJ*, the polarity is always set correctly, regardless of the recording mode. Note that there is also a **Mode** control with identical function in the *Amplifier* window. For details see EPC9 Manual *Compensation Procedures*..

- **In Out** - inside-out configuration
- **On Cell** - cell-attached configuration
- **Out Out** - outside-out configuration
- **Whole Cell** - whole-cell configuration
- **C-Clamp** - current-clamp recording
- **V-Clamp** - two-electrode voltage-clamp



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

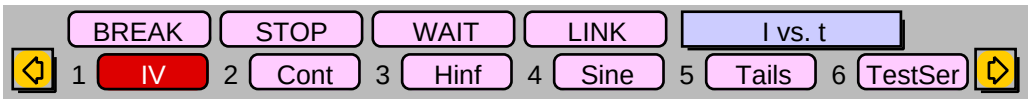
Note: This filter setting should not be confused with the analog filter settings of the EPC9.

Average: Number of averages acquired for one sweep. It is not mandatory but advised to use even numbers of averages in order to avoid possible problems with the *Alternating Leak* averaging feature (see below). Only the average is stored to disk.

V-membrane: The selected membrane potential or holding current (*I-membrane*) in current-clamp mode. This value always gives the real membrane potential (ex-

tracellular side = 0 mV), given a correct *Recording Mode* (and a correct reading of the cell potential in *On Cell* mode). The user does not need to care about sign convention.

Controlling the Pulse Generator



Sequence Pool: Pool of available pulse templates. Six template controls are displayed at a time; paging is done by clicking on the arrows at the ends of the pool section. The keys '1'…'9' are used to execute a template with that number. The number of a specific sequence with higher index than 9 can be entered after typing the pound key '#'. The highlighted sequence can alternatively be executed by typing 'E'. Note that most of the key assignments are not fixed, i.e., the user can change them by modifying and storing the dialogs. In order to avoid confusion it is not recommended to do so with the above mentioned, already assigned controls.

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

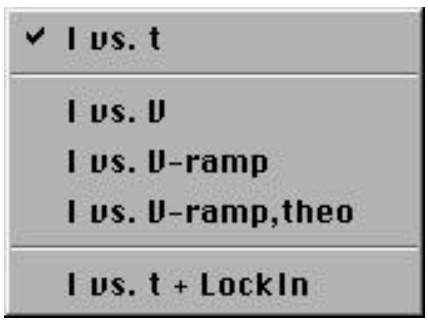
Break: **Break** is used to stop series execution. The **Break** flag is reset, when the next target is displayed or executed. If **Break** is pressed during acquisition of a sweep, this particular sweep will not be completed and the data acquired for this sweep thus far will be discarded. All previous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

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

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

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

Display Mode: Shows a popup for selecting a display mode. This popup has been added for convenience only, it shows the same options as the sub-menu **Display → Display Mode**.



- **I vs. t**: Plots current versus time.
- **I vs. V**: Plots current versus measured voltage (i.e., 1st trace versus 2nd trace where the second trace is assumed to be the voltage trace).
- **I vs. V-ramp**: Plots current versus measured voltage during ramp segments. It is assumed that the second trace is the voltage trace.
- **I vs. V-ramp, theo**: Plots current versus theoretical ramp voltage.
- **I vs. t + LockIn**: Plots current versus time plus corresponding membrane capacitance calculated by the *LockIn* extension, if activated and the sweep contains at least one sinewave or squarewave segment.

Current Corrections

Show P/n: Displays the summed and scaled leak response for each sweep in the background.

Leak Subtraction: Shows the sweeps with the leak correction applied.

Note: The raw data are stored in a leak-corrected form, i.e., with the leak-trace subtracted. The leak trace (which is also stored) may be used to reconstruct the uncorrected traces.

Zero Subtraction: Subtracts the DC leak current based on the data points corresponding to the first stored segment. This is the segment that contains the first trigger (see *Pulse Generator*). If no trigger is set, it is the entire first segment. Only the data from the beginning of the trigger till the end of that segment are used to compute the leak current. If not more than two data points are left in this segment, the *Zero Current* is set to zero, i.e., no zero subtraction is performed.

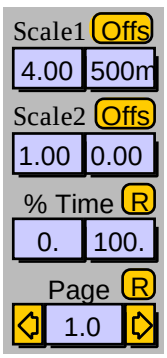
Note: The data are stored both as subtracted data and the actual P/n data. The raw (i.e. not subtracted) data can therefore always be reconstructed. The "Zero Current" is stored as entry in the "Series Record" to avoid recalculating it during data replay.

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

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

Display Scaling

Scale: For each trace (the current and the optional second trace) there is one control that determines the display scaling. The value of 1 corresponds to full scale of ±10.24 V. '+' (from the numeric keypad, take care not to have 'NUMLOCK' activated!) increases the display scaling by a factor of 2 and '-' decreases it by a factor of 2. 'OPTION' + '+' and 'OPTION' + '-' (MacOS) or 'CTRL' + '+' and 'CTRL' + '-' (Windows) do the same for *Scale 2*. .



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

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

Offs: Clicking on the **Offs** button (or '.' on the numeric keypad) automatically centers the trace on the screen. The key commands is '*' (from the numeric keypad) for the first trace, and '*' plus 'OPTION' (MacOS) or 'CTRL' (Windows) for the second trace.

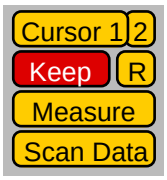
% Time: Section of the sweep to be shown on the screen in % (*Start* - *End*). The reset button sets the full sweep length (0 to 100 %).

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

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

Reset (R): This resets display gains, display offsets, start and end times, which is useful to quickly "undo" the scaling performed by the mouse selection

Cursors



Cursors: Two cursor ranges may be set independently:

- **Cursor 1**: Cursor 1
- **2**: Cursor 2

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

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

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

Measure: Allows to measure current amplitudes. These are continuously written to the *Notebook* window. Two options are available:

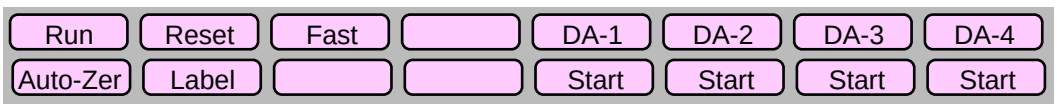
- **Measure by Dragging**: With this option the user can measure the displayed data by pointing and dragging the mouse. Two mouse-driven horizontal lines are provided to measure current differences.
- **Measure by Scanning**: This option allows to scan the data points themselves. When selected, a marker is displayed on the trace. This marker can be moved forward and backward with the cursor key, while at the same time, the data values are displayed in the *Notebook*. Every time a cursor key is

pressed, the speed with which the cursor moves over the trace is changed. Pressing twice the same cursor key increases the speed, while inverting the direction reduces the speed. The following commands are available:

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

Serial Buttons

If a serial communication has been established, the *Oscilloscope* window will show several additional buttons at the bottom (you will have to increase the size of the window to see them).

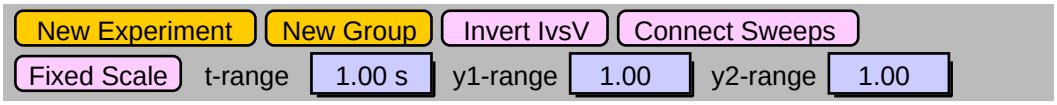


Depending on the type of protocol (**User Defined** or **To X-Chart**) the buttons will have different names. In the example below the buttons are labeled **Run**, **Reset**, ... and correspond to the *X-CHART* protocol, i.e. they represent the buttons from *X-*

CHART's Control window and allow you to "remote control" *X-CHART* running on a second computer.

Experiment Control and Display

Some features are hidden in the lower part of the *Oscilloscope* window. To see them you will have to increase the size of the window accordingly.



New Experiment: Allows to create a new *"Experiment"* target in the *Tree* window. This option has been added to allow a macro to create a new *"Experiment"*.

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

Invert IvsV: Swaps x- and y-axis, so that voltage is potted against the current.

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

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

Fixed Scale: Allows to freeze the scaling of the oscilloscope display to a given fixed range ("fixed" scaling could also be called "manual" scaling). The ranges can be entered in the fields **t-range**, **y1-range**, and **y2-range**:

- **t-range**: The width of the *Oscilloscope* in seconds.
- **y1-range**: The range to be used for the first trace, typically the current trace. E.g. a value of *"200n"* will result in a total range from -100 to +100 nA.
- **y2-range**: The range to be used for the second trace, typically the voltage trace. E.g. a value of *"20"* will result in a total range from -10 +10 V.

Changing the Size of the Oscilloscope

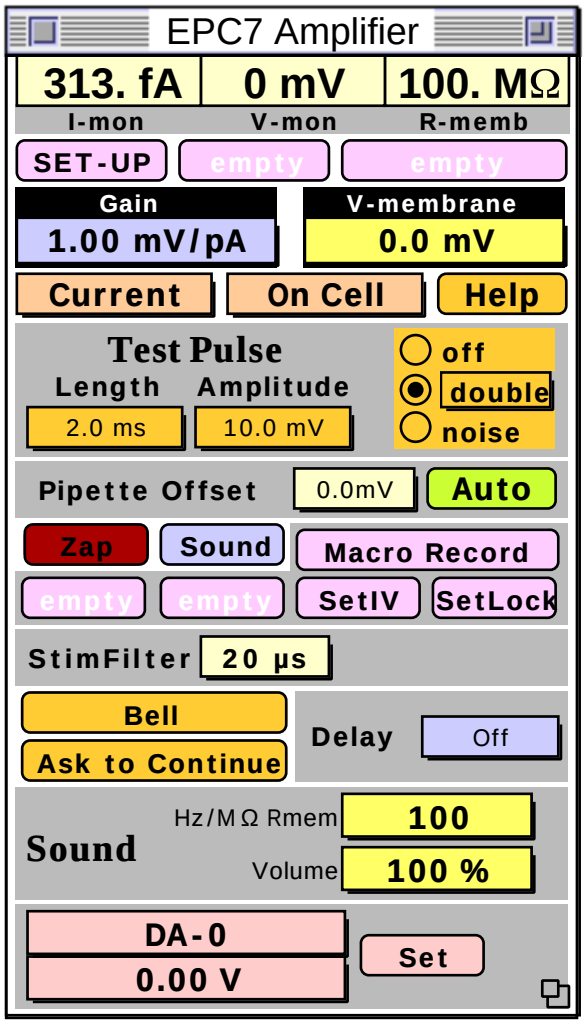
There are two ways to change the size of the *Oscilloscope* window:

1. The shortcut: Hold down the 'SHIFT' key while you resize the window, and all controls will be automatically moved down or right to make space for the *Trace* window.
2. The procedure described in the chapter *User Interface, subchapter 'Tutorial: Changing the Size of the Oscilloscope Window'*. That feature is not as easy, but it allows to define the position and graphic appearance of each single control in the window.

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

EPC 7, EPC 8, and other Amplifiers

The *Amplifier* window is used to display and adjust the settings of analog patch-clamp amplifiers (*EPC7, EPC8, Axon 200*, or any other amplifier). If an *EPC8* or a telegraphing amplifier is used (see *Chapter Configuration*), the actual gain setting of the amplifier will be read from the amplifier and displayed.

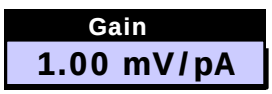


Macros: The macro buttons allow to record and play back a sequence of actions, such as activation of buttons and parameter inputs via mouse or keyboard (for more details about macros see *Chapter Macros*). The macro **SET-UP** is preset by default. It resets all parameters (with the exception of *LJ* and *Vo*), and defines the parameters of the test pulse. This macro can be redefined and other macros can be recorded using the macro record function (see below). All **empty** macros (i.e. undefined macros) are drawn in white letters.

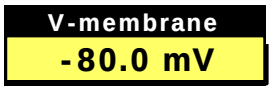
Note: The content of the macros can be written to the Notebook using the option ***Macros*** → ***List*** in the Amplifier drop-down menu.

Amplifier Gain

Gain: Shows and/or sets the gain of the connected amplifier. The gain setting can be typed directly into the field or set by the "UP" and "DOWN" arrow key from the keyboard to match the hardware setting at the amplifier. This way, *PULSE* will be able to correctly scale the recorded current traces. If an *EPC8* or any other telegraphing amplifier is used, the gain setting can be read directly from the amplifier (see *Chapter Configuration*). In the absence of gain telegraphing one can still load a lookup table (see *Appendix Lookup Tables*). In this case only the values from the lookup table will be available and one can *"scroll"* through them using the "UP" and "DOWN" arrow keys.



Holding Potential and Current



V-membrane: Sets the desired holding voltage in *Voltage Clamp* mode. The total range is ±10 V * *Stimulus Scale* (see *Chapter Configuration*) and can be set by dragging the mouse or typing. Dragging with the mouse will change the potential in 0.1 mV steps. Pressing 'CURSOR LEFT' and 'CURSOR RIGHT' changes *V-membrane* in 10 mV steps. Likewise, 'OPTION' + 'CURSOR LEFT' and 'OPTION' + 'CURSOR RIGHT' (MacOS) or 'CTRL' + 'CURSOR LEFT' and 'CTRL' + 'CURSOR RIGHT' (Windows) will change *V-membrane* in 1 mV steps. In *Current Clamp* mode, the item will display the holding current (*I-membrane*).

I-pipette, *V-monitor*, and *R-membrane* are continuously updated, even when the *EPC9* window is not in front, during the wait period between the acquisition of sweeps and series, and in current-clamp mode. *I-pipette* and *R-membrane* are also computed and updated after every acquired sweep.

Channels and Recording Mode

AD-Channel: The oscilloscope can display:

- **Voltage**: Voltage monitor.
- **Current**: Current monitor.
- **AD 0…7**: Any other AD channel.

The channels *Voltage* and *Current* correspond to the channels **Current in** and **Voltage in** that were defined in the **AD channels** section of the *Configuration* window.

Mode: Sets the *Recording Mode*. In *Inside Out* and *On Cell* mode the stimulus and the acquired data are automatically inverted. This applies to all voltages (including *V-membrane* and the pulse protocols).

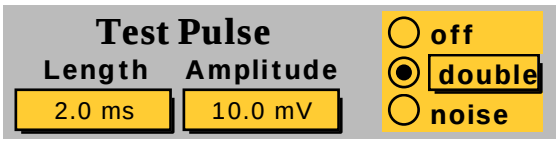
- **In Out**: inside-out configuration
- **On Cell**: cell-attached configuration
- **Out Out**: outside-out configuration
- **Whole Cell**: whole-cell configuration
- **C-Clamp**: current-clamp recording
- **V-Clamp**: voltage-clamp recording (two-electrode clamp)

When the mode *V-Clamp* is selected for operating a two-electrode voltage clamp, the title of the dialog changes from **XXX Amplifier** to **V-Clamp**.

Note: The current and the voltage will automatically be inverted in the two modes "Inside-Out" and "On-Cell". Thus, if you apply a stimulus of e.g. -60 mV, the potential difference at the pipette tip will be **+60** mV. If you are recording from excitable cells and go whole-cell in this situation you might kill the cell! Therefore, always check the mode to be ***WHOLE-CELL*** **before** going whole-cell.

Help: Displays keys assigned to controls in the *Amplifier* window.

Test Pulse



Test Pulse: Test pulses are output and the responses are sampled and displayed. There are two test pulse modes: built-in test pulses (*double* or *single*) and *Test Series* (which uses a stimulation template from the *Pulse Generator* as test pulse). The various controls for test pulse generation are as follows:

Off: Turns the test pulse off.

Double / Single / Test Series / Enter Name: The two built-in test pulses are selected from the test pulse pop-up menu: **single** or **double**. In this case mono- or bipolar pulse patterns are generated, output, and the current signal is sampled and displayed.

Note: For the built-in test pulses the display scaling and trace offset of the Oscilloscope window is not effective for the current trace (the 1st trace), because these test pulses are thought to be used for amplifier tuning, i.e., to adjust gain and capacitance cancellation to achieve a maximal dynamic range of the recorded signal. Thus, without display gain, one can easily see when the input signal saturates the AD converter. If, however, amplification is needed you should enable the setting ***Scale Test Pulse*** in the Configuration window. Display gain and offset are applied to the 2nd trace, because this trace is usually the voltage trace, which is not affected by the amplifier current gain.

The alternative to the fast built-in test pulses is to use a sequence from the pool of available sequences in the loaded *Pulse Generator* file by selecting the third popup entry (by default called *Test Series*). The menu item **Enter Name** allows you to define the pulse sequence to be used. Some restrictions apply to such a sequence:

- no continuous and conditioning segments are supported,
- linked sequences or repeats are ignored

The big advantage of these test pulses is that they provide the full flexibility in pattern generation of the *Pulse Generator*. This includes that the sampled responses are

handled like normal series and that they can be stored to disk. All display scaling features are available as well as the *Online Analysis*. It is possible to disable the output of *Online Analysis* results to the *Notebook* for these test pulses by turning the switch *Test Series Info* in the drop-down menu *Display* off.

Length / Amplitude: Duration and amplitude of built-in test pulses (*single* and *double*) can be specified in the dialog. The minimum pulse duration is 1 ms. No analysis of these pulses is provided, which makes the repetition interval quite short.

Noise: The rms noise is continuously measured and updated (in the field usually labeled **R-memb**) when the noise mode is selected with the button **Noise**. For the determination of the noise level, no pulses are output and current is sampled via the active AD-channel using the current filter settings (in case of the EPC8). It is sampled in sections of 10 times 256 points with a sample interval of 100 µs, i.e., a total length of 256 ms at 10kHz.

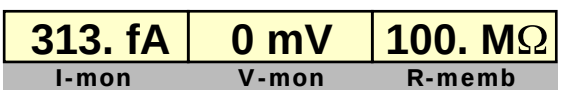
While running the test pulse certain functions can be executed by keys:

KEY	Function
W	Copy the actual value of *R-membrane* to the parameter *Pipette Resistance*
T	Toggle between single- and double pulses
N	Toggle between noise determination and the test pulse mode
U	Set the test-pulse amplitude to 10 mV
D	Set the test-pulse amplitude to 1 mV
I	Set the sample interval for the test pulse to 20 µs
SHIFT + I	Set the sample interval for the test pulse to 200 µs
O, P, SHIFT + P, S, SHIFT + S, +, -, R	Regulate the pipette pressure (see below)

Display of Current, Voltage and Resistance

I-mon: DC current monitor.

V-mon: Voltage monitor.



R-memb: The *Seal Resistance* (*R-membrane*) is determined from the current sampled during the second half of the positive and negative pulse phase (double pulses) or during the baseline and the second half of the pulse (single pulse). The variable *Pi-*

pette Resistance is also part of the *PULSE* data structure. It can be set by typing 'W' (write). This command copies the present value of *R-membrane* into *Pipette Resistance* it has to be done before approaching the cell with the pipette. *R-Membrane* can be encoded into a tone using the *Sound* feature (see below). The resistance value is not computed when using a *Test Pulse Series.*

Computing the Membrane Resistance

1. When a test pulse is applied: Membrane resistance is computed using the difference between the pipette current before and during the first test pulse amplitude.

$$R_{memb} = \frac{V_{test}}{I_{test} - I_{leak}}$$

This algorithm is quite insensitive to problems such as pipette offsets, reversal potential, and liquid junction potential.

1. When no test pulse is applied: Membrane resistance is computed by using the pipette-current only.

$$R_{memb} = \frac{V_{hold}}{I_{leak}}$$

This second algorithm is quite sensitive to many problems such as pipette offsets, reversal potential, and liquid junction potential. In most cases the membrane resistance computed by this second algorithm will differ from that computed by the first algorithm (which is more trustworthy!).

Filter

Filter: Available with an EPC8 only: It controls ("EPC8 remote") or displays ("EPC8 local") the filter settings of the build in analog Bessel filter (4-pole) for the current monitor.

Note: EPC8 only - Filter setttings above 5kHz should not be set in current clamp mode. It may cause the EPC8 to oscillate.

Stim Filter: This may be used to match the setting of the *Stimulus Filter* setting of the *Amplifier*.. The value will be stored with the data.

Various Controls

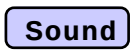
Pipette Offset: After immersing the pipette into the bath, the offset potential at the pi-

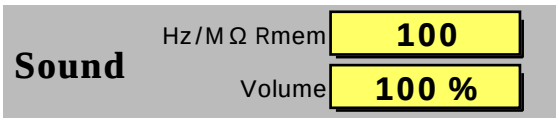
pette can be canceled automatically by pressing the **Auto** button. The pipette offset is then displayed. It can also be set manually.ote that the knob at the amplifier normally used to cancel the pipette offset should not be changed during the experiment.

Zap: A high voltage pulse is applied to the pipette in order to rupture the patch membrane. The parameters of the **Zap** pulse (duration and amplitude) can be specified in the *Configuration* window. There it can also be specified whether *Zap* is always enabled or whether it is restricted to the *On Cell* recording mode (see **Zap On Cell** only).

Note: If the user needs a more complex Zap pulse, e.g., a train of pulses, he can define and use a stimulus sequence. The stimulus can be executed from the Amplifier window by pressing the keys 1 to 9.

Sound: If this control is *On*, a sound is played with its frequency coding for *R-membrane*. The sensitivity (Hz/MΩ) and the volume (in %) of the sound encoding of *R-membrane* can be specified in the bottom part of the *Amplifier* window.





Macros

Macro Record: To start macro recording, click on the **Macro Record** button. Then, perform all desired actions (the *Notebook* window will print a protocol of the macro actions). You may record actions in the *Amplifier*, *Oscilloscope*, *Online Analysis* and *Parameters* windows. While recording a macro you have the option of actually executing all actions as they are entered or disable execution and only log the actions to the macro (see **EPC7**, **EPC8**, **Axon-200** or **Amplifier** menu **Macro** options). To specify a parameter value, enter it as usual by dragging or typing. When clicking on a macro-button you will see a dialog with the following options:

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

Do not forget to save the macros in a file on disk (you can only save all macros at once). Otherwise they will only be remembered until you leave the program. The default macro file is named **Default.EPC7_Macros** (MacOS) or **DefaultEPC7.mac** (Windows) and it will be automatically loaded next time the program is started (for more details about macros see *Chapter Macros*). The default macro file of the *EPC8* amplifier is named **Default.EPC8_Macros** (MacOS) or **DefaultEPC8.mac** (Windows)

Note: There is a limit of 40 actions per macro. To abort recording of a macro, click again on "Record" and the just recorded sequence will be lost.

A waiting period can be inserted between two functions. This may be used e.g., to wait until a cell has settled before an action is performed. The waiting period is generated by selecting a **Delay** value during a macro recording. The **Bell** and the option **Ask to Continue** can also be used for macros:

Bell: Plays the system sound. This is useful to supply feedback when playing back a macro.

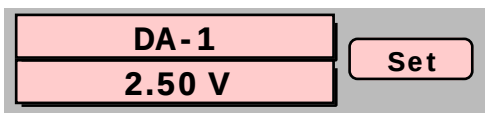
Ask to Continue: Prompts the user in an alert window to continue or abort macro execution.

Delay: Sets a delay of the specified duration (in sec.) before continuing the macro.

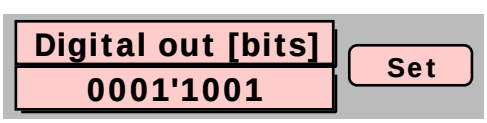
DA-Output and Digital Trigger Lines

DA-Channels / Digital Triggers: This section allows you to set the output of the DA-channels and the digital trigger lines of the *AD/DA Converter*. The digital trigger lines *Dig 0* to *Dig 3* are available through BNCs on the front side of the *ITC-16* or *ITC-18*. *Dig4* to *Dig 14* can be accessed via the DIGITAL-I/O connector on the backside of the interface. If you are using a *Digidata 1200* only *Dig0* to *Dig5* are available via BNCs on the front side. The following options are available:

- **DA-0 / DA-1 / DA-2 / DA-3**: Sets the voltage for the corresponding DA-channel. With the *Digidata 1200* board only **DA-0** is supported.



- **Digital out (word)** : Sets the digital triggers. The output is defined as digital word. E.g. the value *27* would set *Dig0, Dig1, Dig3* and *Dig4* (27 = 1 + 2 + 8 + 16) to their *"high"* state, while the rest of the triggers would be set to *"low"*.
- **Digital out (bits)** : Sets the digital triggers. The output is defined as the bit representation using zeros and ones, lowes bit to the right (the single quote <'> can be used as a separator). E.g.,



01'1001 would set: *Dig4*, *Dig3*, and *Dig0* high, all other low.

Set: This will actually output the voltage at the specified DA-channel or set the digital trigger lines.

The number of supported digital triggers depends on the selected amplifier and used AD/DA-board. The following table gives an overview of the possible combinations:

Amplifier / Digitizer	Available Triggers
EPC9 with TIB-14 ("trigger-box")	Dig0 to Dig13 = 14 lines The EPC9 should not be used without the trigger-box, see **note 1** below
EPC9 Double and Triple	Dig0 to Dig7 = 8 lines
EPC8, Remote mode Connected to ITC-16 or ITC-18	Dig2 to Dig4 = 3 lines Dig0 and Dig1 should not be used, see **note 2** below
EPC8, Local mode Connected to ITC-16 or ITC-18	Dig2 to Dig11 = 10 lines Dig0 and Dig1 should not be used, see **note 2** below
Other amplifiers Connected to ITC-16 or ITC-18	Dig0 to Dig13 = 14 lines
Other amplifiers Connected to Digidata 1200	Dig0 to Dig3 = 4 lines

Note 1: The EPC9 should not be used without the trigger-box, because the digital output lines (bit0…bit15) will get active while PULSE sends commands to the EPC9.

Note 2: One should not use bit 0 and bit 1, when an EPC8 is connected by the standard digital cable to an EPC9, ITC-16, or ITC-18 (neither in "local" nor "remote" mode, nor as "Aux"-amplifier). These digital lines control the "dithering" relays of the EPC8.

Hidden Controls

Some less often used controls are hidden to the right of the *EPC9* window. They can be accessed by clicking on the zoom box of the window:

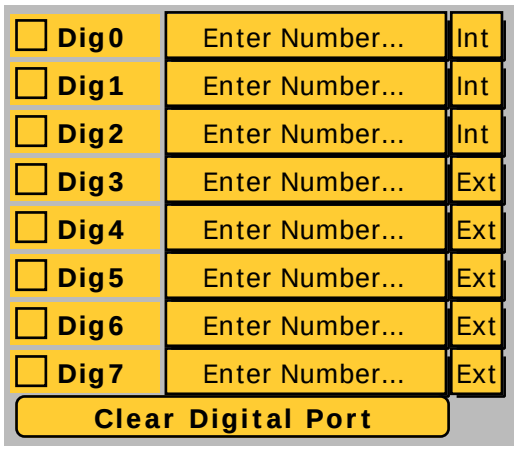
Relative Value: This button is useful when recording macros. A change of any value will be recorded as a relative change, e.g. if

you hyperpolarize the holding potential from -60 to -70 mV while recording a macro, a step of *-10 mV* will be recorded (see *Chapter Macros*).

Empty8 ... Empty20: These buttons call the macros *#8* to *#20*. As long as these macros are **empty** (i.e. undefined) they are drawn in white letters.

Digital Output Lines and the Solution Base Interface

These controls allow to control external solution changers. The number of visible buttons depends on the number of supported digital lines (called triggers in the preceding table). Each checkbox sets and clears one digital line. In addition, when the line is set high the solution selected in the pop-list to the right is set as the internal or external solution as specified in the second pop-up list.



The "Clear Digital Port" button clears (sets to low) all digital output lines.

Special Features for the EPC 8 Amplifier

The *EPC8* can be controlled by the computer (*Remote Mode*) or manually (*Local Mode*). In the first case you set gain, filter etc. within *PULSE* and are not able to change these settings at the amplifier by its hardware controls, in the latter case the settings are read by the software. The two different modes are activated in the *Configuration* window (**EPC8 Remote** or **EPC Local**). The state how the *EPC8* is controlled by the computer is indicated by the REMOTE lamp on the front panel. This LED lights as long as the *EPC8* is controlled by the computer. The following controls can be set and read by computer:

- *Gain* and *Gain Range*
- *Mode* of operation: *Test*, *Search*, *Voltage Clamp*, *Current Clamp*, *Current Clamp + Comm* and *Current Clamp + Comm Fast*
- *Filter*

Search Mode: This button indicates, that the *Search Mode* has been activated, when the *EPC8* is running in *Local* mode. In the *Remote* mode of control this button activates the *Search Mode*.

CC Fast Speed: In the *EPC8* the current-clamp circuitry has been improved to better follow rapid changes in membrane po-

tential, such as in neuronal action potentials. This *Fast Current Clamp* mode can be activated with this button, or - in *Local Mode* - this button is highlighted, when the amplifier is in *CC + Comm Fast* mode.

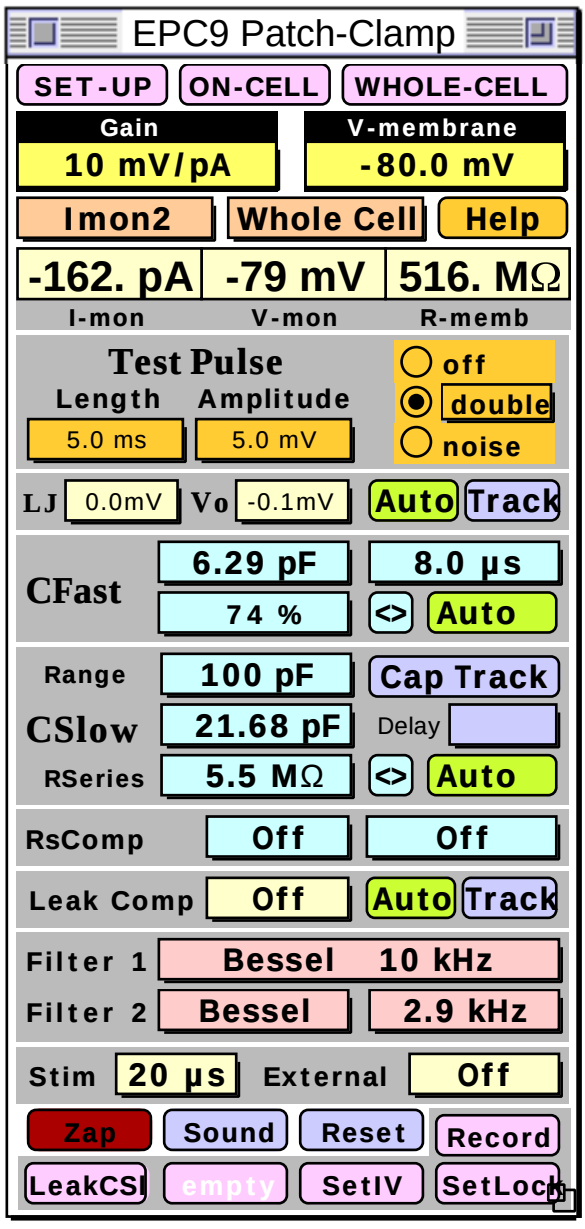
Pipette Pressure Control

For users equipped with a pipette pressure controller the following keys allow program-controlled pressure application. There are no special buttons or controls for this in the window. The corresponding voltage is output via the DA channel specified in the *Configuration* window and may be changed whenever the *Amplifier* or *Oscilloscope* windows are active.

KEY	Function
O	Apply zero pressure
P	Apply positive pressure
SHIFT + P	Set positive pressure
S	Apply negative pressure
SHIFT + S	Set negative pressure
+	Increase negative pressure
-	Decrease negative pressure
R	Reset to default values (p = 3 cm H_2O, s = 10 cm H_2O)

EPC 9 Amplifiers

This chapter describes the various controls in the *EPC9 Amplifier* window. If you want to know further details about the *function* of these controls, e.g. the different operating modes or built-in compensation procedures, please refer to the *EPC9 Manual*.



Macros: The top and the bottom of the *EPC9* window contain 7 *Macros* (actually there are 20 macros available, the rest is hidden in the right part of the window). The macro buttons allow to record and play back a sequence of actions, such as activation of buttons and parameter inputs via mouse or keyboard (for more details about macros see *Chapter Macros*). The macros **SET-UP**, **ON-CELL**, and **WHOLE-CELL** are preset. **SET-UP** resets all parameters (with the exception of *LJ* and *Vo*), and defines the parameters of the test pulse. On-Cell switches the Gain range to a typical setting for a cell-attached patch recording, and invokes an Auto C-fast compensation. 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. These macros can be redefined and other macros be recorded using the macro record function (see below). All empty macros (i.e. undefined macros) are drawn in white letters.

Note: The content of the macros can be written to the Notebook using the option Macros (List in the EPC9 drop-down menu.

Amplifier Gain

Gain: Sets the scaling of the current monitor output. The range is from 0.005 to 2000 mV/pA and can be set by dragging the mouse or pressing 'Cursor up' and 'Cursor down'. The gain setting automatically selects one of the three available current-measuring feedback resistors in the probe (5 MOhm, 500 MOhm, and 5 GOhm), corresponding to the low, medium or high gain range. The table below summarizes the main features and limitations of the gain ranges:

	Low	Medium	High
Feedback Resistor	5 MΩ	500 MΩ	50 GΩ
Gain	0.005-0.002	0.5-20	50-2000
I_{max}	± 2 µA	± 20 nA	± 200 pA
Bandwidth	100 kHz	100 kHz	60 kHz
C-slow Ranges	30 • 100 • 1000	30 • 100 • 1000	30 • 100
Current Clamp	no	yes	no
R_S-compensation	yes	yes	yes

The lowest gain range may be used for experiments in which *large* currents (i.e. up to about 2 µA) have to be measured, like bilayers, loose-patch or large cells. Capacitance compensation of up to 1 nF is available and R_S-compensation can be used for R_S values down to 10 Ω in this range.

In the medium gain range, the background noise is larger than in the high gain, but the full 100 kHz bandwidth is available, and currents of up to about 20 nA can be recorded. This range is used mainly for whole-cell recordings, and for this purpose the special features of the 1000 pF transient cancellation range (see *C-slow Ranges*), series resistance compensation, and the current-clamp modes are made available. On the other hand, the high gain range is intended for single-channel recording. It has a very low noise level, but this is obtained at the expense of a maximum current limit of about 200 pA. The maximum available bandwidth is about 60 kHz, and the special features mentioned above do not function in this range.

Slow capacitance cancellation ranges (30-100-1000 pF) can be set to any desired value. However, in the high gain range (50 GΩ resistor) the 1000 pF range will not operate. If the 1000 pF range is selected while the gain is higher than 20 mV/pA, the gain will automatically be reduced to the highest possible setting (20 mV/pA). Similarly, since the current-clamp mode is only possible in the intermediate gain

range (0.5-20 mV/pA), the gain will be reduced or increased appropriately when selecting *Current Clamp* mode from an invalid gain range.

Clipping: A blinking box labeled **Clip** in the **Gain** title shows the state of the EPC9 *Clipping* light. This LED lights whenever an amplifier saturates in the current monitor pathway. The indicator is important in voltage-clamp experiments where capacitive artifacts will be subtracted by the computer; the subtraction will work well only as long as no saturation occurs, and this indicator serves as a simple monitor of this condition. It is particularly useful since it will indicate clipping by internal amplifiers even in cases where, because of filtering, the output voltage is not saturated..

Holding Potential and Current

V-membrane / I-membrane: Sets the desired holding voltage in *Voltage Clamp* modes. The range is ±1000 mV and can be set by dragging the mouse or typing. Dragging with the mouse will change the potential in 0.1 mV steps. Pressing 'CURSOR LEFT' and 'CURSOR RIGHT' changes *V-membrane* in 10 mV steps. Likewise, 'OPTION' + 'CURSOR LEFT' and 'OPTION' + 'CURSOR RIGHT' (MacOS) or 'CTRL' + 'CURSOR LEFT' and 'CTRL' + 'CURSOR RIGHT' (Windows) will change *V-membrane* in 1 mV steps. In *Current Clamp* mode, the item will display the holding current (*I-membrane*).

I-pipette, *V-monitor*, and *R-membrane* are continuously updated, even when the *EPC9* window is not in front, during the wait period between the acquisition of sweeps and series, and in current-clamp mode. *I-pipette* and *R-membrane* are also computed and updated after every acquired sweep.

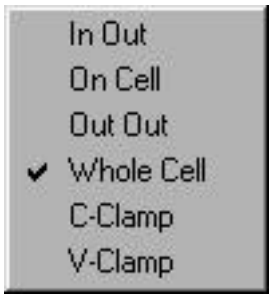
Channels and Recording Mode

AD-Channel: This control sets the analog channel shown in the oscilloscope.

- **F2-Ext**: *External Stim. Input* filtered by *Filter 2*
- **Imon-1**: *Current Monitor 1*
- **Imon-2**: *Current Monitor 2*
- **Vmon**: *Voltage Monitor*
- **AD 0…6**: This allows you to read the AD channels directly. The channel assigned to the *Voltage Monitor* of the active amplifier (see *Chapter Configuration*) will be labeled accordingly (**Voltage**). In the case of the *EPC9 Double* and *Triple* the current- and voltage outputs are hardwired with AD-Channels (**Vmon-1**, **Imon-1**, …).

The *F2-Ext* setting allows you to use *Filter 2* as a general-purpose variable filter. In this setting the *External Stim Input* becomes the filter input, and the filter output is available at *Filter 2* and is displayed in the *Oscilloscope* window.

Mode: Sets the *Recording Mode*. In *Inside Out* and *On Cell* mode the stimulus and the acquired data are automatically inverted! This applies to all voltages (including *V-membrane* and the pulse protocols). Note that there is also a *Mode* control with identical function in the *Oscilloscope* window. As for *LJ*, the polarity is always set correctly, regardless of the recording mode.



- **In Out**: inside-out configuration
- **On Cell**: cell-attached configuration
- **Out Out**: outside-out configuration
- **Whole Cell**: whole-cell configuration
- **C-Clamp**: current-clamp recording
- **V-Clamp**: voltage-clamp recording (two-electrode clamp)

Note: The current and the voltage will automatically be inverted in the two modes "Inside-Out" and "On-Cell". Thus, if you apply a stimulus of e.g. -60 mV, the potential difference at the pipette tip will be **+60** mV. If you are recording from excitable cells and go whole-cell in this situation you might kill the cell! Therefore, always check the mode to be ***WHOLE-CELL*** **before** going whole-cell.

When switching from Voltage-Clamp to Current-Clamp mode, *I-hold* will be set to inject the holding current required to keep the membrane voltage constant (*"gentle switch"*). Thus, V-mon will be near to the original *V-membrane* commanded under Voltage-Clamp. In analogy, upon returning to voltage clamp, *V-membrane* will be set according to the membrane potential under current clamp and rounded to the next mV. This feature can be turned off with a control hidden to the right of the *EPC9* window (**Gentle CC-Switch: ON/OFF**) which allows the user to have no clamping at all in CC-mode.

Note: For computation I-monitor2 is read instead of I-monitor1. This prevents possible glitches which may be caused by switching the internal multiplexer. Also, the signal on I-monitor 2 is better filtered and, thus, the signal more stable.

Help: Displays keys assigned to controls in the *EPC9* window and switches to the help modus: the next click on an EPC9-button will bring up the help window with a short description of the selected function..

Current Clamp Controls

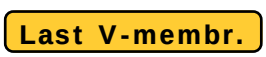
These control items are normally hidden to the right in the *EPC9* window; they can be seen by clicking on the zoom box of the *EPC9* window.

CC Fast Speed: The *EPC9* amplifiers version 'C' and later have an improved current-clamp mode with two speed settings. When *PULSE* recognizes the presence of these *EPC9* models, it adds the menu entry *CC Fast Speed* to the *EPC9* drop-down menu, and enables the *CC Fast Speed* control in the amplifier window. For details see the EPC9 Manual *Chapter Operating Modes.*.

CC Range: Allows to select the range of the current clamp stimulus. The selectable options depend on the amplifier version, only the available ones are enabled. The maximal current output in current clamp mode is 1 nA and 10 nA in the 1 pA/mV and 10 pA/mV ranges, respectively.

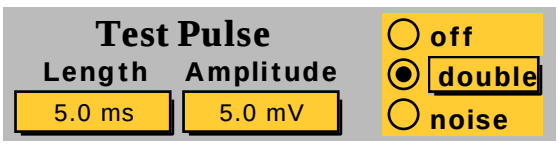
Gentle CC-Switch: Allows to select between two alternate modes of switching between voltage and current clamp and back. If the selection is set to OFF, the EPC9 will be switched to zero holding current and voltage, respectively. This could be called the "traditional switching mode", since that is the way all amplifiers beside the EPC9 have to be switched. The EPC9 allows a more sophisticated and useful mode of switching between voltage and current clamp and back. In this mode, the amplifier preloads the amplifier section to be switched to with the momentarily measured current or voltage, then switches to other mode. That way the cell will not be affected by a change in holding current or voltage. We call this mode "gentle switching" because it does not activate the cell in contrast to the "traditional" mode.

Last V-membrane: Quite often, the membrane potential under current clamp will have changed, and when returning to voltage clamp, *V-membrane* will possibly differ from that set before switching to current-clamp mode (see description of "Gentle Switch" right above). *Last V-membrane* can then be used to conveniently restore the original *V-membrane* value.



Test Pulse

Test Pulse: Test pulses are output and the responses are sampled and displayed. There are two test pulse modes: built-in test pulses (*double* or *single*) and *Test Series* (which uses a stimulation template from the *Pulse Generator* as test pulse). The various controls for test pulse generation are as follows:



Off: Turns the test pulse off

Double / Single / Test Series / Enter Name: The two built-in test pulses are selected from the test pulse pop-up menu: **single** or **double**. In this case mono- or bipolar pulse patterns are generated, output, and the current signal is sampled and displayed.

Note: For the built-in test pulses the display scaling and trace offset of the Oscilloscope window is not effective for the current trace (the 1st trace), because these test pulses are thought to be used for amplifier tuning, i.e., to adjust gain and capacitance cancellation to achieve a maximal dynamic range of the recorded signal. Thus, without display gain, one can easily see when the input signal saturates the AD converter. If, however, amplification is needed you should enable the setting ***Scale Test Pulse*** in the Configuration window. Display gain and offset are applied to the 2nd trace, because this trace is usually the voltage trace, which is not affected by the amplifier current gain.

The alternative to the fast built-in test pulses is to use a sequence from the pool of available sequences in the loaded *Pulse Generator* file by selecting the third popup entry (by default called *Test Series*). The menu item **Enter Name** allows you to define the pulse sequence to be used. Some restrictions apply to such a sequence:

- no continuous and conditioning segments are supported,
- linked sequences or repeats are ignored

The big advantage of these test pulses is that they provide the full flexibility in pattern generation of the *Pulse Generator*. This includes that the sampled responses are handled like normal series and that they can be stored to disk. All display scaling features are available as well as the *Online Analysis*. It is possible to disable the output of *Online Analysis* results to the *Notebook* for these test pulses by turning the switch *Test Series Info* in the drop-down menu *Display* off.

Length / Amplitude: Duration and amplitude of built-in test pulses (*single* and *double*) can be specified in the dialog. The minimum pulse duration is 1ms. No analysis of these pulses is provided, which makes the repetition interval quite short.

Noise: The rms noise is continuously measured and updated (in the field usually labeled **R-memb**) when the noise mode is selected with the button **Noise**. For the determination of the noise level, no pulses are output and current is sampled via the active AD-channel using the current filter settings (usually the *Current Monitor 2* output, so that the bandwidth is determined by *Filter 2*). It is sampled in sections of 10 times 256 points with a sample interval of 100 µs, i.e., a total length of 256 ms.

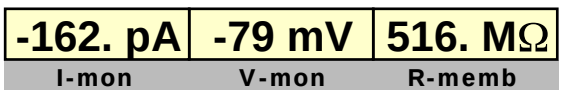
While running the test pulse certain functions can be executed by keys:

KEY	Function
W	Copy the actual value of *R-membrane* to the parameter *Pipette Resistance*
T	Toggle between single- and double pulses
N	Toggle between noise determination and the test pulse mode
U	Set the test-pulse amplitude to 10 mV
D	Set the test-pulse amplitude to 1 mV
I	Set the sample interval for the test pulse to 20 µs
SHIFT + I	Set the sample interval for the test pulse to 200 µs
O, P, SHIFT + P, S, SHIFT + S, +, -, R	Regulate the pipette pressure (see below)

Display of Current, Voltage and Resistance

I-mon: DC current monitor.

V-mon: Voltage monitor.



R-memb: The *Seal Resistance* (*R-membrane*) is determined from the current sampled during the second half of the positive and negative pulse phase (double pulses) or during the baseline and the second half of the pulse (single pulse). The variable *Pipette Resistance* is also part of the *PULSE* data structure. It can be set by typing 'W' (write). This command copies the present value of *R-membrane* into *Pipette Resistance* it has to be done before approaching the cell with the pipette. *R-Membrane* can be encoded into a tone using the *Sound* feature (see below). The resistance value is not computed when using a *Test Pulse Series*.

Computing the Membrane Resistance

1. When a test pulse is applied: Membrane resistance is computed using the difference between the pipette current before and during the first test pulse amplitude.

$$R_{memb} = \frac{V_{test}}{I_{test} - I_{leak}}$$

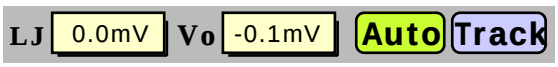
This algorithm is quite insensitive to problems such as pipette offsets, reversal potential, and liquid junction potential.

1. When no test pulse is applied: Membrane resistance is computed by using the pipette-current only.

$$R_{memb} = \frac{V_{hold}}{I_{leak}}$$

This second algorithm is quite sensitive to many problems such as pipette offsets, reversal potential, and liquid junction potential. In most cases the membrane resistance computed by this second algorithm will differ from that computed by the first algorithm (which is more trustworthy!).

Voltage Offsets



V_0 (*Pipette Offset*) : V_0 displays the offset voltage (a voltage which is added to *V-membrane* to obtain the pipette command voltage). It can be set either by the *Auto-V_0* operation or by manually dragging the mouse after clicking into the item. Furthermore, V_0 is changed automatically whenever the user changes the variable Liquid Junction *LJ*. This is necessary for *LJ* and the *Auto-V_0* operation to interact properly. It is not recommended that the user changes V_0 manually, because this interferes with the software features for automatic offset correction.

Auto-V_0: The *Auto-V_0* button calls a procedure for automatic zeroing the pipette current. Thereby, an offset voltage (V_0) to the pipette potential is systematically varied until the pipette current is zero. The range of V_0 is ± 200 mV. *Auto-V_0* is typically performed before seal formation. It works properly only when a pipette is inserted into the bath.

The *Auto-V_0* procedure interacts with the variable *LJ* to provide for online correction of liquid junction potentials and other offsets (see *EPC9* Manual *Chapter Compensation Procedures*). This requires that *V-membrane* is set to the value of *LJ* (for *On Cell* and *Inside Out* Recording Modes) or to the opposite polarity of *LJ* (for *Whole Cell* and *Outside Out* Recording Modes), before the actual zeroing operation is performed. *Auto-V_0* does this automatically and leaves *V-membrane* at that value.

Note: V_0 is not changed by the ***Reset*** function.

Track: This option implements a *Search Mode* (see *Chapter Operating Modes* in the *EPC9* Manual), which is essentially a repetitive *Auto-V_0* procedure at a holding potential of 0 mV. The rate at which *Auto-V_0* is performed is determined by the **Search Mode Delay** in the **EPC9** menu.

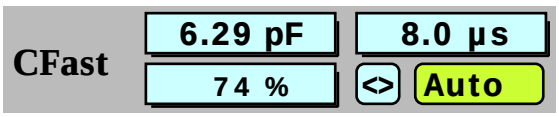
LJ (*Liquid Junction*) : *LJ* is a variable, to be set by the user, which allows to correct for liquid junction potentials and other offsets. It works in conjunction with the V_0 operation. An online correction requires an *Auto-V_0* operation to be performed before

seal formation and *LJ* to be set to an appropriate value. No correction is performed if *LJ* = 0. See *EPC9* Manual *Chapter Compensation Procedures* for more information on how to determine *LJ*.

LJ can be adjusted within ± 200 mV by dragging the mouse or typing after a double-click. Please note that *LJ* is not changed by the **Reset** function, and cannot be set by macros. This restriction is imposed to avoid unintentional offset corrections.

LJ should be 0 mV when using identical pipette and bath solutions. It may be changed to any desired value within ± 200 mV in case asymmetrical solutions are used or the bath solution is changed during an experiment. For the standard liquid junction potential correction, the polarity of the entered value should be such that it represents the potential of the bath with respect to the pipette solution. For example, if the pipette solution contains glutamate or aspartate (with chloride in the bath), then the polarity of *LJ* should be positive (+10 mV). After an *Auto-V_0* operation, *V-membrane* will be changed to -10 mV (in *Whole Cell* and *Outside Out* Recording Modes) or +10 mV (for *On Cell* and *Inside Out* Recording Modes), which corresponds to the true zero-current potential.

Fast Capacitance Compensation



CFast: This is used to cancel fast capacitive currents that charge the pipette and other stray capacitance (range: 0-15 pF). With nothing connected to the probe input, cancellation is typically obtained at a setting of 1-1.5 pF due to the residual input capacitance of the current-measuring amplifier. The compensation can be performed manually by dragging the mouse or typing. Actually, the setting consists of two components, a completely unfiltered one and a second one filtered by τ-*fast*. The program allows you to set the amplitude of the sum of the two components and the relative contribution of the filtered component (in %). The %-setting acts like a vernier to τ-*fast*. A useful way of adjusting the values manually would be to roughly adjust the value of *C-fast* at a setting of 50% contribution and then adjust τ-*fast*. Thereafter, one can fine-tune the %-setting to fully compensate capacitive transients. A much easier way of adjusting *C-fast* and τ-*fast* is provided by the automatic compensation feature (simply select the *Auto* button; see below).

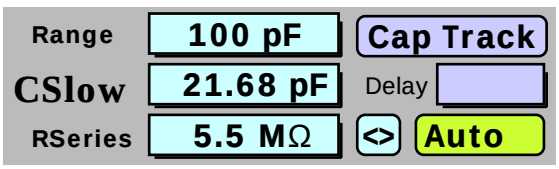
τ-fast: This determines the time constant of *C-fast* (range: 0.5-8 µs in steps of 0.5 µs). The degree to which the filtered component contributes to the compensation is specified by the %-setting in *C-fast*. The value of τ-*fast* may be adjusted by dragging the mouse, or typing, or automatically by selecting the *Auto* function.

<>: This will select both amplitude and time constant of *C-fast*. Values for each may be changed manually depending on whether the mouse is moved horizontally or vertically.

Auto CFast: Selection of this button performs an automatic compensation of *C-fast* and τ-*fast*. The procedure uses a routine that applies a number of small pulses (5 mV), averages the resulting currents and fits an exponential to deduce the capacitance compensation values required to cancel the current. During this procedure some parameters are changed temporarily: R_S-*comp* is turned off, the stimulation is interrupted and the *Gain* is internally set to an appropriate value (0.2, 2 or 50 mV/pA, depending on the effective gain range). The values are updated in the respective displays. If the automatic compensation routine fails to converge to ideal adjustment, the small box in the **Auto** button (**E**) will be darkened to indicate the error. Auto compensation works best with Sylgard™-coated pipettes in the cell-attached configuration with *C-slow* turned off.

The software allows for two ways of making the compensation. The fastest is to use a look-up table for setting *C-fast*. In order to do this you have run the program *E9Screen* and select the option **Make C-fast** (see also *Chapter Setting up the EPC9*). Otherwise, when starting *PULSE*, a message will appear that the file could not be found and that the (slower) default method will be used (iterative compensation).

Slow Capacitance Compensation



CSlow: This is used to cancel slow capacitive currents that charge the cell membrane in the whole-cell configuration. The 30, 100 and 1000 pF ranges actually allow capacitance values to be compensated in the ranges of 0.12-30 pF, 0.4-100 pF and 4-1000 pF, respectively. The adjustment range is also limited by the program in order to make the time constant *R-series* * *C-slow* greater than 5 µs to prevent oscillations. The *C-slow* value may be adjusted manually by dragging the mouse, or typing, or automatically by selecting the **Auto** function. Auto-compensation will adjust *C-slow* as well as *R-series*.

Note: If you are using the ***Cap. Track*** feature to measure small capacitance changes (see below), it is advantageous to use a higher C-slow range when possible. Counter-intuitive as it may seem, the digital control of the C-slow circuitry is more precise in higher ranges, resulting in higher resolution, consistent with the gain range and capacitance.

CSlow Range: Selects the range for slow capacitance compensation:

- **Off**: Turns cancellation off.

- **30 pF**: Small cells.
- **100 pF**: Small and medium-sized cells.
- **1000 pF**: Large cells (only in the low and intermediate *Gain Range*).

RSeries: Adjusts the resistance in series with the slow capacitance (range: 0.1 MΩ - 10 GΩ) to determine the time constant of the *C-slow* transient and also for R_S-compensation. Adjustment is limited by the capacitance values and the range as described above. The value can be changed manually by dragging the mouse, or typing, or automatically by clicking on **Auto**.

<>: This will select both *C-slow* and *R-series*. Values for each are changed depending on whether the mouse is moved horizontally or vertically.

Auto CSlow: Selecting this function performs an automatic compensation of *C-slow* and *R-series*. These settings are used by the Rs-compensation circuitry as the measure of series resistance. The procedure uses a routine that applies short trains of square-wave pulses (number and amplitude of these pulses are specified by *C-slow Num. Cycles* and *C-slow Peak Amplitude*), averages the resulting currents and fits an exponential to deduce the compensation values required to cancel the current (see *EPC9* Manual *Chapter Compensation Procedures*). During this procedure some parameters are changed temporarily: *Rs-comp* is turned off, the stimulation is interrupted and the gain is internally set to an appropriate value (0.2, 2 or 50 mV / pA, depending on the effective gain range). If the *C-slow Range* is set to *Off*, it will automatically be set to the most accurate *C-slow Range*. The most accurate *C-slow Range* is the highest possible, i.e., the 1000 pF range in the low and medium gain ranges, and the 100 pF range in the high gain range.

Auto-compensation works best when *C-fast* is canceled beforehand in the cell- attached configuration. The compensation may be improved by alternating cycles of *Auto C-fast* and *Auto C-slow*. If the compensation fails, the small box in the **Auto** button (**E**) will be darkened to indicate the error. Continuous Auto-compensation can be performed by selecting **Cap. Track** option, see below.

Note: It is necessary that reasonable estimates for C-slow and R-series are supplied before the Auto ***C-slow*** is started, otherwise the auto-compensation may fail. The auto-compensation routine will be faster and more reliable when the starting estimates are larger than the final values. If the estimates are too small, the routine will often fail.

Cap. Track: Selecting this button will perform repetitive *Auto C-slow* compensation. Clicking on *Cap. Track* again will turn it off. The frequency of *Cap. Track* is determined by the *Delay* setting. It is also possible to continue to perform *Cap. Track* even when switching to the *Oscilloscope* window. Since no test pulse is running when the *Oscilloscope* window is selected, the maximal frequency of capacitance measurements

is about twice the frequency one can achieve in the *EPC9* window. The time displayed when performing *Cap. Track* is the time of the *Timer* (the one listed in the *Display* menu). This allows to define the starting time, or to mark an event by resetting the timer.

Note: The ***Output Cap. Track*** option in the EPC9 menu determines whether the capacitance and G-series values are output as voltages on the DA channels 0 and 1. If selected, the C-slow value is available at DAC 0, and the G-series value (1/R-series) is available at DAC 1. The scaling for G-series is 100 nS/V; the scaling of C-slow is 0.5, 5 or 50 pF/V for the 30, 100 and 1000 pF range, respectively. The DAC values will be reset to zero when activating ***Reset***. Therefore, make sure that the trigger DAC (to be set in the Configuration window) is neither DAC 0 nor DAC 1 when you want to activate ***Output Cap. Track***.

Delay: This determines how many seconds to wait before the next auto-compensation is started. With short delays (e.g., 1 ms), membrane capacitance will be tracked at rates of >10 Hz (depending on the speed of the computer). If **Output Cap. Track** is selected in the **EPC9** menu, the *C-slow* and *G-series* values are output on DA channels 0 and 1.

Series-Resistance and Leak Compensation

R_S-comp: The series resistance compensation corrects for membrane voltage errors under conditions of high access resistance between pipette and cell interior (see *EPC9* Manual *Chapter Compensation Procedures*). The amount of compensation can be changed by dragging the mouse or typing (range 0-95%). The compensation is based on the value of R-series and will be effective only when *R_S-comp* is not *Off*, i.e., set to a speed value. The following settings determine the speed of feedback compensation:

- **Off**: Turns compensation off.
- **100 µs**: Slow compensation.
- **10 µs**: Fast compensation.
- **2 µs**: Very fast compensation.

The choice of speed depends on the recording time constant and the degree of compensation desired, as described in the *EPC9* Manual *Chapter 6*. Fast R_S-compensation requires more critical adjustment of the controls but provides the maximum voltage-clamp speed. In fast modes, *Filter* 1 will be automatically switched to its HQ-30 kHz setting. In *Current Clamp* mode, *R_S-comp* will act as a bridge compensation.

Note: When switching between Voltage Clamp and Current Clamp modes, R_S-comp will be turned off, so you have to re-enable it manually. However, PULSE will remember the %-settings of the compensation when switching modes. In order to avoid possible overcompensation you should decrease the %-setting before activating R_S-comp again.

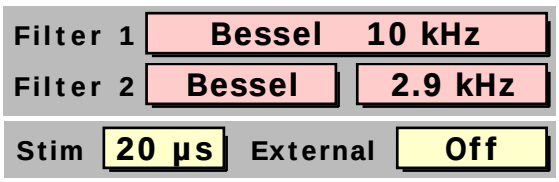
Leak Comp: This controls a hardware leak compensation which uses the inverted and scaled stimulus signal and adds it to the current monitor outputs. *Leak* can be adjusted manually by dragging the mouse, or typing, or automatically by clicking on *Auto*. The maximal values attainable depend on the *Gain Range*. They are 2 nS, 200 nS and 20 µS for high, medium, and low ranges, respectively.

Note: The data acquisition in PULSE does not consider a possible hardware "Leak" correction of the EPC9. The "Leak" value is stored and displayed together with the series parameters, if "Leak" was activated (i.e., Leak >= 0.1 pS). Be aware of the consequences using ***Leak Comp***, i.e., of the fact that the current measurements are not absolute values anymore. Because of this and because capacitive artifacts cannot be subtracted by this type of leak subtraction, it is advised to use the P/n leak subtraction instead whenever possible.

Auto: Automatically determines and corrects the leak conductance.

Track: Continuously updates the leak compensation.

Filter



Filter 1: Controls an analog Bessel filter (3-pole) in the current monitor pathway. The menu provides for the following settings:

- **Bessel 100 kHz**
- **Bessel 30 kHz**
- **Bessel 10 kHz**
- **HQ 30 kHz**

Under most conditions a 10 kHz bandwidth is more than ample, and the filtering reduces the high-frequency noise substantially. The *HQ 30 kHz* setting is selected automatically when the fast R_S-compensation is in use; it is of little use otherwise.

Filter 2: Controls an analog 4-pole filter for the current monitor 2 (*I-mon 2*). *Filter 2* is in series with *Filter 1*. Dragging the mouse or typing allows fine adjustment from 0.1-16 kHz in 0.1 kHz steps (guaranteed accuracy 0.5-15 kHz). Two filter characteristics can be selected from the pop-up menu:

- **Bessel**
- **Butterworth**

The -3 dB point is given in both cases. *Bessel* is the best characteristic for general use; the *Butterworth* response rolls off more rapidly with frequency and is mainly useful for power spectral analysis. The setting of the **Filter 2** control is the actual effective bandwidth of the series combination of *Filter 1* and *Filter 2*, i.e., the bandwidth of the signal actually recorded at *I-mon 2*.

If you want to use an external filter rather than *Filter 2*, you can proceed as follows: Connect current monitor 1 (*I-mon 1*) to the external filter input, feed the filter output back into one of the AD channels (AD 0...AD 5), and select this channel as current input channel in the *Configuration* window.

Note: Filter 2 is disabled when I-mon1 is selected. This is to prevent the user to mistake the Filter 2 bandwidth with the active filter bandwidth used for the sampled data, i.e., Filter 1 bandwidth. To change Filter 2 in this case, select I-mon2 first.

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

- **2 µs**
- **20 µs**

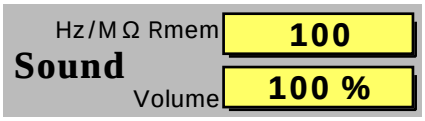
Usually a setting of 20 µs is sufficient, unless very fast currents such as Na^+ currents are studied.

External: The external *Stim. Input* is scaled by an editable factor (range: ±0-10x), to allow for different external stimulators. It is strongly recommended to set *External* to *Off* (i.e., equal to zero), if no external stimulator is connected to the *Stim. Input*. This will prevent pick-up of external noise. Please note that the internal *V-membrane* is not affected by changing the external scale factor. If the user sets the holding potential externally (e.g., with a stimulator or another computer), then the scaling will affect the holding potential.

Various Controls

Zap: A high voltage pulse is applied to the pipette in order to rupture the patch membrane. The parameters of the **Zap** pulse (duration and amplitude) can be specified in the *Configuration* window. There it can also be specified whether *Zap* is always enabled or whether it is restricted to the *On Cell* recording mode (see **Zap On Cell** only).

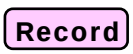
Note: If the user needs a more complex Zap pulse, e.g., a train of pulses, he can define and use a stimulus sequence. The stimulus can be executed from the EPC9 window by pressing the keys 1 to 9.



Sound: If this control is *On*, a sound is played proportional to *R-membrane*. The tone (Hz/MΩ) and the volume can be specified. The parameter settings for the sound are normally hidden, but may be accessed by expanding the *EPC9* window. The sensitivity (Hz/MΩ) and volume (in %) of the sound encoding of *R-membrane* can be specified there.

Reset: Selecting this button will reset the *EPC9* to its initial default configuration. It will reset the DA channels to zero, which is useful e.g. to cancel the output of *C-slow* and *R-series* set by *Cap. Track*. Also, **Reset** is very useful to define the initial state of the *EPC9*, when recording a macro.

Macros



Record: To start macro recording, click on the **Record** button. Then, perform all desired actions (the *Notebook* window will print a protocol of the macro actions). You may record actions in the *EPC9*, *Oscilloscope*, *Online Analysis*, and *Parameters* windows. While recording a macro you have the option of actually executing all actions as they are entered or disable execution and only log the actions to the macro (see **EPC9** menu **Macro** options). To specify a parameter value, enter it as usual by dragging or typing. When clicking on a macro-button you will see a dialog with the following options:

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

Do not forget to save the macros in a file on disk (you can only save all macros at once). Otherwise they will only be remembered until you leave the program. The default macro file is named **Default.Epc9_Macros** (MacOS) or **DefaultEPC9.mac** (Windows) and it will be automatically loaded next time the program is started (for more details about macros see *Chapter Macros*).

Note: There is a limit of 40 actions per macro. To abort recording of a macro, click again on "Record" and the just recorded sequence will be lost.

Hidden Controls

Some less often used controls are hidden to the right of the *EPC9* window. They can be accessed by clicking on the zoom box of the window:

Empty8 ... Empty20: These buttons call the macros *#8* to *#20*. As long as these macros are **empty** (i.e. undefined) they are drawn in white letters.

Relative Value: This button is useful when recording macros. A change of any value will be recorded as a relative change, e.g. if you hyperpolarize the holding potential from -60 to -70 mV while recording a macro, a step of *-10 mV* will be recorded (see *Chapter Macros*).

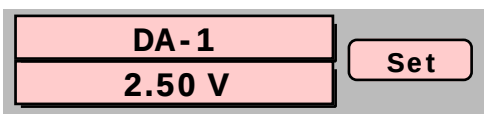
Ask to Continue: A macro function that prompts the user in a control to continue. Clicking the mouse in the window or pressing any key will continue the program.

Bell: Plays the system sound. This is useful to supply feedback when executing a macro.

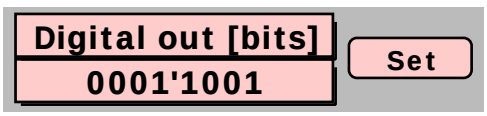
DA-Output and Digital Trigger Lines

DA-Channels / Digital Triggers: This section allows you to set the output of the DA-channels and the digital trigger lines of the *EPC9*. The digital trigger lines (*Dig 0, ... Dig 14*) are directly available at the DIGITAL-TRIGGER-OUT or DIGITAL-I/O interface on the backside of the amplifier if you are using an *EPC9 Double* or *Triple*. If you have an *EPC9,* you will need the *Digital Trigger Converter TIB14* in addition (please contact HEKA for more information). The following options are available:

- **DA-0 / DA-1 / DA-2 / DA-3**: Sets the voltage for the corresponding DA-channel.



- **Digital out (word)** : Sets the digital triggers. The output is defined as digital word. E.g. the value *27* would set *Dig0, Dig1, Dig3* and *Dig4* (27 = 1 + 2 + 8 + 16) to their *"high"* state, while the rest of the triggers would be set to *"low"*.
- **Digital out (bits)** : Sets the digital triggers. The output is defined as the bit representation using zeros and ones, low bit to the right (the single quote <'> can be used as a separator). E.g., *01'1001* would set: *Dig4, Dig3,* and *Dig0* high, all other low.



Set: This will actually output the voltage at the specified DA-channel or set the digital trigger lines.

The number of supported digital triggers depends on the selected amplifier and used AD/DA-board. The following table gives an overview of the possible combinations:

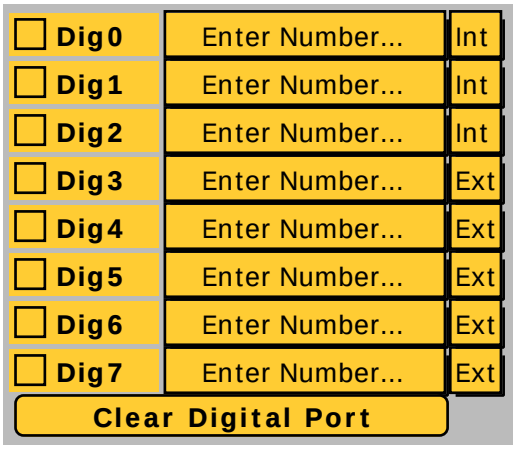
Amplifier / Digitizer	Available Triggers
EPC9 with TIB-14 ("trigger-box")	Dig0 to Dig13 = 14 lines The EPC9 should not be used without the trigger-box, see **note 1** below
EPC9 Double and Triple	Dig0 to Dig7 = 8 lines
EPC8, Remote mode Connected to ITC-16 or ITC-18	Dig2 to Dig4 = 3 lines Dig0 and Dig1 should not be used, see **note 2** below
EPC8, Local mode Connected to ITC-16 or ITC-18	Dig2 to Dig11 = 10 lines Dig0 and Dig1 should not be used, see **note 2** below
Other amplifiers Connected to ITC-16 or ITC-18	Dig0 to Dig13 = 14 lines
Other amplifiers Connected to Digidata 1200	Dig0 to Dig3 = 4 lines

Note 1: The EPC9 should not be used without the trigger-box, because the digital output lines (bit0…bit15) will get active while PULSE sends commands to the EPC9.

Note 2: One should not use bit 0 and bit 1, when an EPC8 is connected by the standard digital cable to an EPC9, ITC-16, or ITC-18 (neither in "local" nor "remote" mode, nor as "Aux"-amplifier). These digital lines control the "dithering" relays of the EPC8.

Digital Output Lines and the Solution Base Interface

These controls allow to control external solution changers which can be controlled by single digital TTL lines. The number of visible buttons depends on the number of supported digital lines (called “triggers” in the preceding table). Each checkbox sets and clears one digital line. In addition, when the line is set high the solution selected in the pop-list to the right is set as the internal or external solution as specified in the second pop-up list.



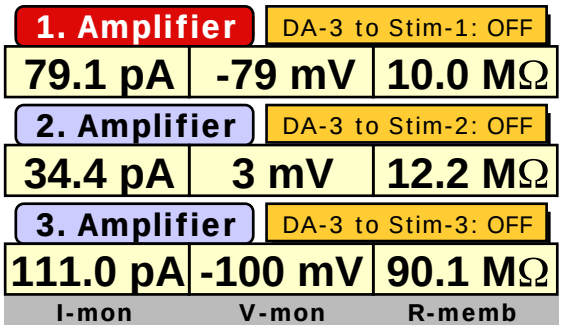
The “Clear Digital Port” button clears (sets to low) all digital output lines.

Pipette Pressure Control

For users equipped with a pipette pressure controller the following keys allow program-controlled pressure application. There are no special buttons or controls for this in the window. The corresponding voltage is output via the DA channel specified in the *Configuration* window and may be changed whenever the *EPC9* or *Oscilloscope* windows are active.

KEY	Function
O	Apply zero pressure
P	Apply positive pressure
SHIFT + P	Set positive pressure
S	Apply negative pressure
SHIFT + S	Set negative pressure
+	Increase negative pressure
-	Decrease negative pressure
R	Reset to default values (p = 3 cm H_2O, s = 10 cm H_2O)

Special Features of the EPC9 Double and Triple



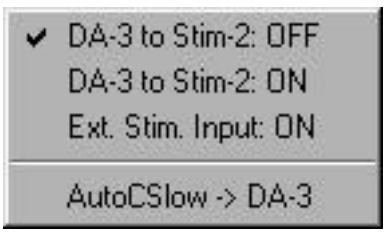
The *EPC9 Amplifier* window will include some additional fields, if an *EPC9 Double* or *Triple* is used instead of an *EPC9* amplifier. The controls for the third amplifier will be added only, when an *EPC9 Triple* is connected.

To make a specific *EPC9* amplifier the "active" one, you must click the corresponding *"Amplifier"* field. *PULSE* will immediately update all settings to display the state of the selected amplifier. Thus, amplifier settings can only be changed on the selected *"active"* amplifier. Meanwhile, the other amplifiers will keep their settings.

The **I-mon** and **V-mon** fields of any amplifier will be continuously measured and updated, as soon as the respective amplifier was selected at least once. For example, this allows to keep the third amplifier quiet during experiments in which it is not used. If one wants to activate all amplifier from the start on, one can record each amplifier selection as an operation in the default **SET-UP** macro.

The **R-memb** field is continuously measured and updated as well. Thus, only the membrane resistance of the *"active"* amplifier can be computed using the first, better estimation (see *p. 88*), and only when the test pulse is applied!

The *EPC9 Double* and *Triple* have three ways to stimulate an amplifier. These options can be selected in the pop-up menus **DA-3 to Stim-X** individually for each separate amplifier:



1. One can send the stimulus directly to the internal stimulus DA channel of the respective amplifier, e.g., to DA-0 for the first amplifier. This is the default setting and corresponds to the stimulation of the standard *EPC9*.
2. One can feed the stimulus directly to the internally connected DA-3 and connect the stimulus input of the respective amplifier to DA-3 as well. This allows to simultaneously stimulate multiple amplifiers. Separate scaling factors are available for each individual amplifier.
3. One can use an external stimulator and connect it to the EXTERNAL STIMULUS INPUT. Scaling factors are available for those inputs as well.

Individual scaling factors for each amplifier and each external stimulation mode can be

entered in the **External** field (near the bottom of the *EPC9* window), when either **DA-3 to Stim-X: ON** or **Ext. Stim Input: ON** is activated.

This option can be set even when the respective amplifier is not the *"active"* amplifier. There are two reasons for this exception: first, multiple amplifiers may be simultaneously affected. And second, it has to be possible to set that option by a macro which is executed at the beginning of a particular *Pulse Generator* stimulus. In that case one does not want to interfere with the selection of the *"active"* amplifier. The goal is only to connect the required amplifier inputs for that particular experiment without knowing or changing the current selection of the *"active"* amplifier.

One has to be aware that selecting the second or third option results in the stimulus **to be summed with the holding potential** which is already applied internally. Thus, one has to use the *Absolute Voltage* stimulation type in the *Pulse Generator*, scaling it with the inverse of **Stimulus Scale** (see *Configuration* window), i.e. with a factor of 10. The signal inversion according to the selected **Mode** (*OnCell, WholeCell,* etc.) is compensated in hardware and must not be considered for the stimulus.

One should be aware of the following limitations:

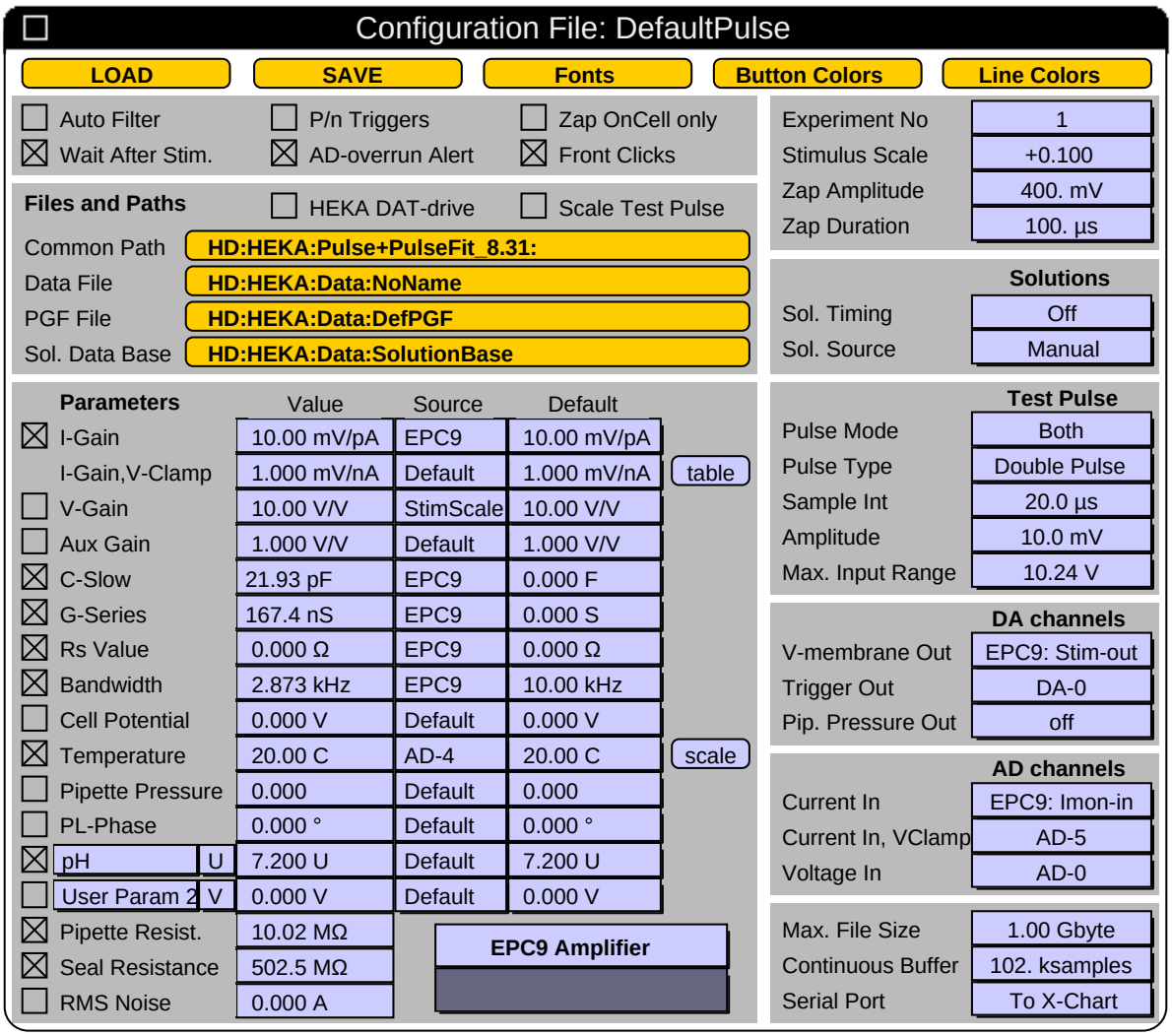
1. The test pulse is sent to all amplifiers with the selection **DA-3 to Stim-X: ON**, when the *"active"* amplifier uses that option!

2. The option **DA-3 to Stim-X: ON** uses DA-channel 3. Therefore, DA-3 will no longer be available for other applications, as soon as one amplifier is set to that option.

The option **AutoCSlow -> DA-3** can be used when compensating *C-Slow* capacitance in cells which are electrically coupled, e.g., when patching cells connected by gap-junctions. Normally, the stimulus used for the *Auto C-Slow* compensation is sent to the *Stim-DA* of the active amplifier. When the option **AutoCSlow -> DA-3** of the active amplifier is selected, the stimulus is send to all amplifiers which have the option **DA-3 to Stim-X: ON** selected. The purpose is to eliminate the current through, e.g., the gap-junctions, which otherwise would prevent a good *Auto C-Slow* compensation.

Configuration

Settings like sources for external parameters, default values, display settings, colors, fonts, default files, etc. can be edited in the *Configuration* window. Most important: this is the place where you define the connections to the hardware! To access the *Configuration* window type 'F11' (MacOS) or 'F8' (Windows) or select the drop-down menu **Pulse → Configuration**. All settings from herein can be stored as a specified file with the name extension ***.set** (*PULSE* asks you by default at the end of the program session if you wish to store the settings). This allows that every user can define her / his individual program layout to meet the specific requirements.



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

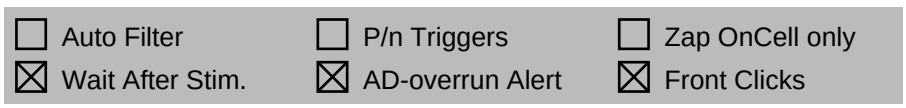
Fonts / Button Colors / Line Colors: Colors and text fonts for the program layout can be selected here. These are global settings for all window dialogs and they are installed upon restart of *PULSE*. The colors and fonts are stored in the configuration file (***.set**), i.e., they are independent of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

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

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

Under Windows on can modify the windows and desktop elements with the option “Start -> Settings -> Control Panel -> Appearance”.

General Settings



Auto Filter: The *Pulse Generator* (see *Chapter Pulse Generator*) provides the option to automatically set a hardware filter according to the chosen sample interval. *Auto Filter* enables and disables this automatic filter setting, which is only supported for the EPC9 with its built-in hardware filters. In the test pulse mode the EPC9 filter is set according to the sample interval if *Auto Filter* is on using a *Filter Factor*.

Note: Short test pulses will require a high sampling rate, i.e., a high filter bandwidth. For bandwidths greater than approximately 9 kHz, PULSE automatically selects I-mon 1 as input, if an EPC9 amplifier is used; otherwise PULSE selects I-mon 2. Therefore, it can happen that the channel selection automatically toggles between I-mon 1 and I-mon 2 if test pulses are followed by a sequence from the Pulse Generator or vice versa.

P/n Triggers: Output trigger signals for each P/n pulse in addition to the triggers output during the main pulse.

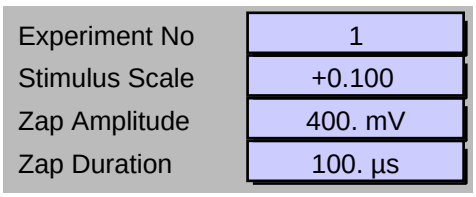
Zap OnCell only: If this checkbox is selected, the *Zap* feature is restricted to *On Cell* mode. This ensures that one does not accidentally zap a patch or a whole cell.

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

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

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

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

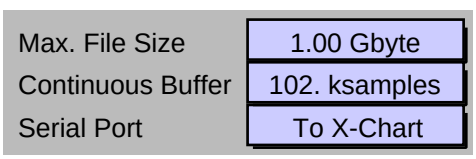


Experiment No: This number can be used to identify experiments. Whenever **Pulse → New Experiment** is executed, this number is incremented. It is stored at the group level, i.e., only one experiment is contained within one group, but one can have multiple groups with the same experiment number.

Stimulus Scale: Scaling factor of the stimulus signal in *Voltage Clamp* mode. The entry *CC-Stim. Scale* defines the stimulus signal in *Current Clamp* mode. E.g., the *EPC9* has a *CC-Stimulus Scale* of 1 pA / mV, while the *Axon 200* has a scale of 2 pA / mV.

Note: The EPC9 has a internal fixed Stimulus Scale factor of 0.1.

Zap Amplitude / Zap Duration: Amplitude and duration of a so-called *Zap* pulse can be edited here. *Zap* is used for rupturing a membrane patch electrically with a short pulse of high amplitude in order to achieve the whole-cell recording configuration. Zap pulses can be applied by 'CTRL' + 'Z' or the corresponding control in the *Amplifier* window.



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

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

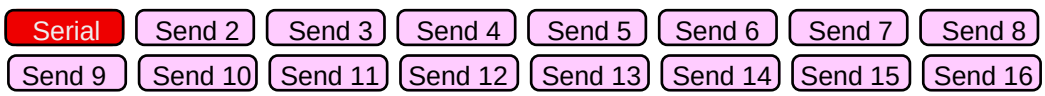
Note: This option requires a buffer size large enough to accommodate the data in memory. A comprehensive description is given in chapter PULSE Advanced -> Continuous Data Acquisition

Serial Communication

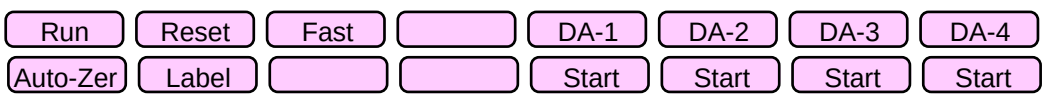
Serial Port: This control is used to open and set up the serial port communication mode. If a so called *Serial Communication* has been established between the setup computer and another device, *PULSE* can send strings over a serial port, but it will not receive instructions. After activating the protocol,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

The options for the serial communication are:

- **None**: No connected device.
- **User Defined**: Any device that can receive *Strings* through the serial port. Up to 16 strings can be defined and triggered by buttons in the *Oscilloscope*. These buttons are hidden at the bottom of the window and can be seen after expanding the *Oscilloscope* window (e.g. by clicking on the zoom box). The first button (**Serial**) is used to open the serial port (as well as sending an initialization string) or to edit the rest of the "*Send*" strings (**Send 2 ... Send 16**). The **Send** buttons can be custom-labeled by editing the button (see *Chapter User Interface*).

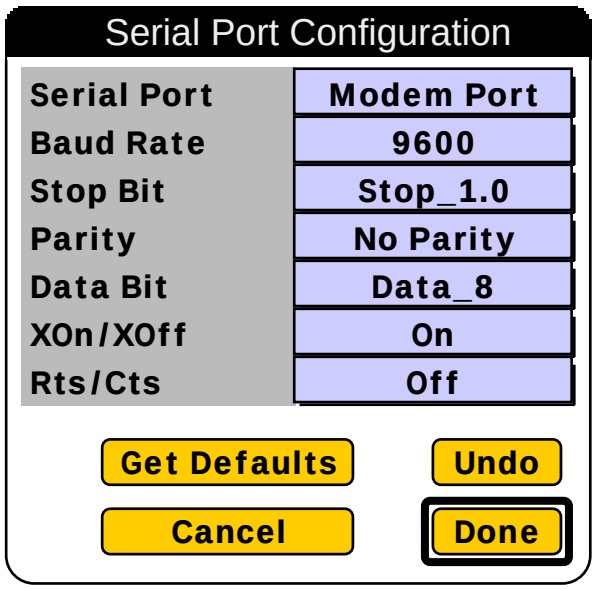


- **To X-Chart**: Special protocol to control the standalone version of *X-CHART* running on another computer. The new buttons correspond to the buttons from *X-CHART's Control* window and allow you to "remote control" the program running on a second machine from *PULSE* (see the *X-Chart Manual* for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



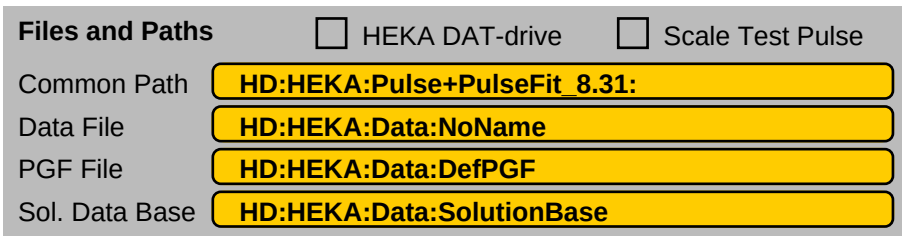
When opening a serial communication, *PULSE* will allow you to configure the serial device. Make sure, that the settings match on both communicating machines.

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



Files and Paths

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



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

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

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

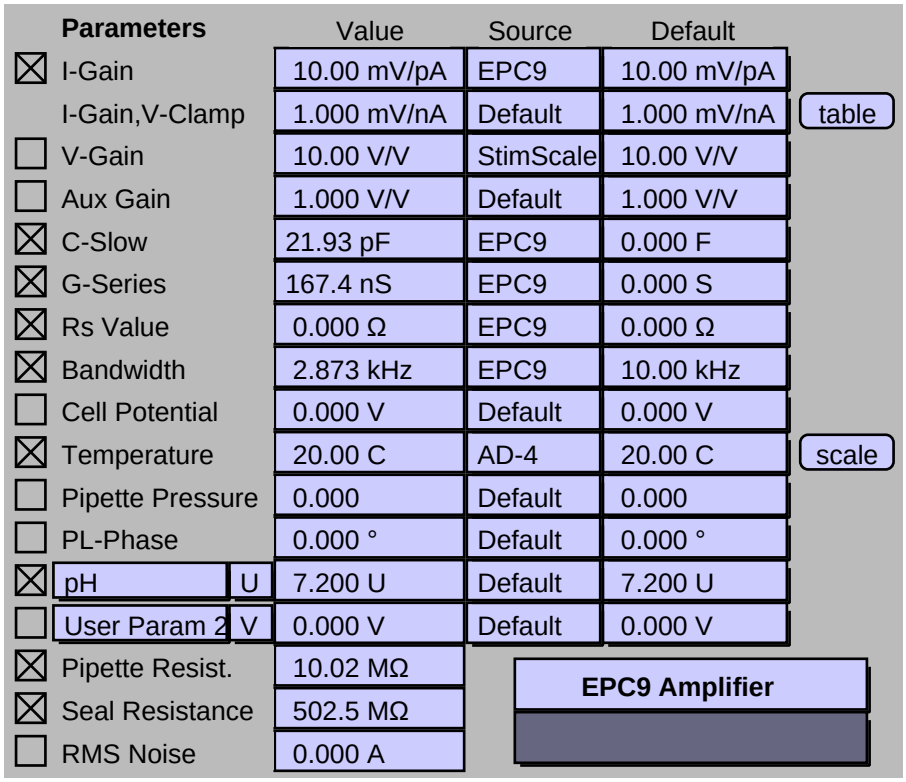
The title of the *Configuration* window contains the name of the file that holds the information shown in the *Configuration*, *Oscilloscope*, *Amplifier*, and *Online Analysis* windows. This file is read from or written to disk relative to the specified **Common Path**. The data file and the PGF File are generally written relative to the specified path. These paths and file names can be selected here and are used upon program restart as default values. **Sol. Data Base** provides the name of the common solution data base (see *Chapter Solutions*).

HEKA DAT-Drive: A special file handling is supported, when the digital tape drive available from HEKA is used as storage medium (this product has been discontinued). Such a medium is quite useful for very long files, such as continuous single-channel recordings. One can achieve sustained, continuous recordings up to 80 kHz and 2 Gigabytes of samples. Please note, the production and support of the *HEKA DAT-drive* has been discontinued!

Scale Test Pulse: Enabling this option will allow you to scale the oscilloscope when running the test pulse.

Parameters

Parameters such as temperature, gain, holding potential, etc. are entered here. These are used to customize the recording environment of a given experimental setup. The settings are automatically preset when using an *EPC9* amplifier. Other amplifiers will require specific settings in order for *PULSE* to know how to scale an input or output voltage. The checkbox on the left determines whether the parameter is displayed (and updated) in the *Parameters* window.



- **I-Gain**: Gain of the current monitor input.
- **I-Gain, V-clamp**: Gain of a voltage-clamp amplifier input.
- **V-Gain**: Gain of the stimulus voltage input.
- **Aux Gain**: Gain of an auxiliary voltage input.
- **C-slow**: Membrane capacitance.
- **G-series**: Series conductance.
- **Rs Value**: Series resistance compensation.
- **Bandwidth**: Recording bandwidth.
- **Cell Potential**: Cell potential.
- **Temperature**: Temperature (from a recording device).
- **Pipette Pressure**: Pressure (from an application device).
- **PL Phase**: Phase angle of a lock-in amplifier.
- **User Parameter 1**: Parameter from a custom device (e.g., pH).
- **User Parameter 2**: Parameter from a custom device.
- **Pipette Resistance**: Value of *R-membrane* after pressing 'W' (this will write the value of *R-membrane* into *Pipette Resistance*).

- **Seal Resistance**: Value of *R-membrane*.
- **RMS Noise**: Value of the RMS noise.

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

Value: Shows the actual value of the corresponding parameter.

Source: Determines how the value is obtained. For most of the parameters there are three alternatives (*Default*, *EPC9*, *AD Channels*):

- **Default**: The specified default value is taken, i.e. you can edit this value during the experiment directly in the *Parameters* window (see *Chapter Parameters*).
- **EPC9**: Parameters are obtained directly from the *EPC9*.
- **AD-Channels**: Parameters are sampled via a specified AD-channel.

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

Scale: If a parameter is read via an AD channel, a scaling feature allows to convert the acquired voltage into the desired variable. If gain settings are not read directly from the *EPC9*, analog input can be decoded via ASCII lookup tables; these tables can be easily modified by most text editors.

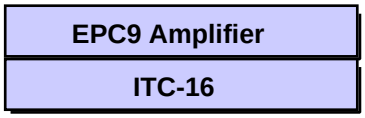
Note: It takes CPU time to read AD channels and to convert the data to the desired values. In order to minimize handling overhead only activate those parameter fields that are actually used.

Monitor Gain: Since there is no trivial equation that relates DC voltage to the gain, gain lookup tables in ASCII format can be provided by the user (see below: *Amplifier Selection*).

The actual gain that will be used (*I-Gain*, *I-Gain for Two-Electrode Voltage Clamp*, or *V-Gain*) is automatically selected according to the recording mode. *I-Gain* is used for the "Current In" channel, when a patch-clamp mode is selected. "*I-Gain*, V-Clamp" is used for *V-Clamp* mode, when a two-electrode clamp is used. *V-Gain* is used when reading the *Voltage In* input channel, and *Aux-Gain* is applied otherwise.

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

Amplifier: From the list at the bottom of the dialog one can select the patch-clamp amplifier used.



PULSE supports the following amplifiers:

- **EPC9/n Amplifier**: An *EPC9 Single, Double, or Triple* amplifier.
- **EPC8 Remote**: An *EPC8* amplifier, controlled by the software.
- **EPC8 Local**: An *EPC8* amplifier, controlled by the knobs on the front panel of the amplifier. *PULSE* monitors gain, amplifier modes, and filter settings.
- **EPC7 Amplifier :** any amplifier with no telegraphing options.
- **Axon 200 Amplifier**
- **Telegraphing Amplifier**

Digitizer Selection: Below the amplifier list one can select the AD/DA-converter used, when the amplifier is not an EPC9.

PULSE supports the following AD/DA-converters:

- **ITC-16.**:
- **ITC-18.**
- **Digidate 1200** (Windows 95/98 only).

If a *Telegraphing Amplifier* is used, the gain and/or bandwidth settings for the amplifier is read from a lookup table. Upon selection of this amplifier type, the user is asked to load first a gain table and then a bandwidth table. Some lookup tables for common patch-clamp amplifiers are already supplied (see folder **Pulse -> Lookup Tables** or *Appendix Lookup Tables*). If your amplifier is not in the list, select the *EPC7* table or create your own table. Lookup tables can be created easily because they are ASCII files with a simple structure with a series of text lines each containing:

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

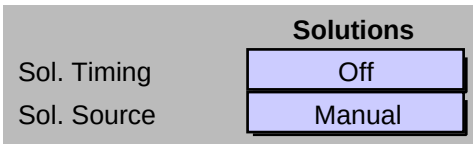
The voltage is the threshold above which the following gain or bandwidth setting applies. The voltage values are scanned beginning with the first value. Thus, voltages must be in descending order! The voltage thresholds for discrimination between adjacent settings should be halfway between the nominal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

If *PULSE* is only used for data review or demo purposes, the AD/DA hardware is not necessarily required. In this case *PULSE* will come with the message "AD/DA initialization failed: check power!" or "EPC9 initialization failed! Please, check power and connections". One can now enter *Continue* and select one of the demo modes (e.g., *EPC9 Demo* or *EPC7 Demo* mode). This ensures that the program does not try to access the hardware. Any output that is created is taken as input; i.e., if a *Stim. Scaling* of 0.1 is selected, the system now behaves as if an amplifier is connected

with a pipette having a resistance of 10 MΩ. The *Demo Mode* can be used to inspect and analyze data on an extra computer that is not connected to the setup and to evaluate the software package. In this mode no data can be stored to disk.

Solutions

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



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

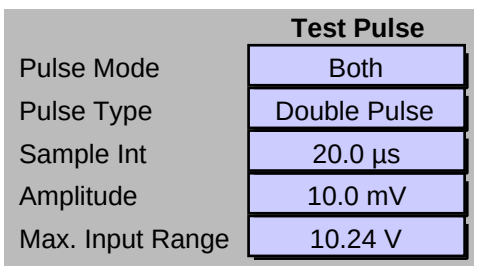
- **Off**: No solution file is created.
- **Before Series**: Inquire solution information before series acquisition.
- **After Series**: Inquire solution information after series acquisition.
- **After Group/Exp**: Inquire solution information after group acquisition, i.e., whenever a new group is created.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

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

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

Test Pulse

The following parameters determine the layout of the fast built-in test pulses. Note that these values are stored in the configuration file; they can, however, be overwritten by execution of macros in the *Amplifier* window.



Pulse Mode: Selects the measurement and display of the test pulses as defined in the DA channel selection.

- **Current**: Records and displays *I-mon*.
- **Voltage**: Records and displays *V-mon*.
- **Both**: Records and displays *I-mon* and *V-mon*.

Pulse Type: Selects the following options (these are defaults which can be overridden by the settings in the *Amplifier* window):

- **No Pulse**: Disables test pulses.
- **Single Pulse**: Stimulates with a single rectangular pulse.
- **Double Pulse**: Stimulates with a bipolar double pulse.
- **Test Series**: Stimulates with a sequence from the *Pulse Generator* file. The text of this menu item is the name of the selected Series.
- **Enter Name**: Allows to specify the name of the Series to be used as test pulse..

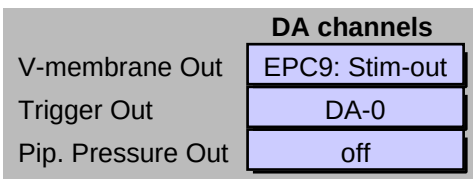
Sample Interval: Determines pulse sample rate (default: 20 µs = 50 kHz).

Amplitude: Pulse amplitude in millivolts.

Max. Input Range: This defines the maximal AD input range. This is useful, e.g., if the amplifier has a limited range. This variable defines the maximal input range displayed in the test pulse window (in *'V'*). The pulse generator will check any voltage against that voltage range. The scaling of the test pulse uses that voltage range as well.

DA-Channels

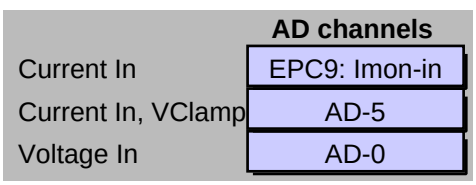
These items define the default output channel assignments used for stimulation, triggers, and pressure devices (see also *Chapter Pulse Generator*).



- **V-membrane Out**: Assigns the output DA channel of the stimulus.
- **Test Trigger Out**: Assigns the output DA channel of the trigger pulses.
- **Pipette Pressure Out**: Assigns the output DA-channel for a pressure device.

AD-Channels

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



- **Current In**: Assigns the input AD channel for the current trace
- **Current In, V-clamp**: Assigns the input AD for a voltage-clamp amplifier.

- **Voltage In**: Assigns the input AD for the voltage trace.

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

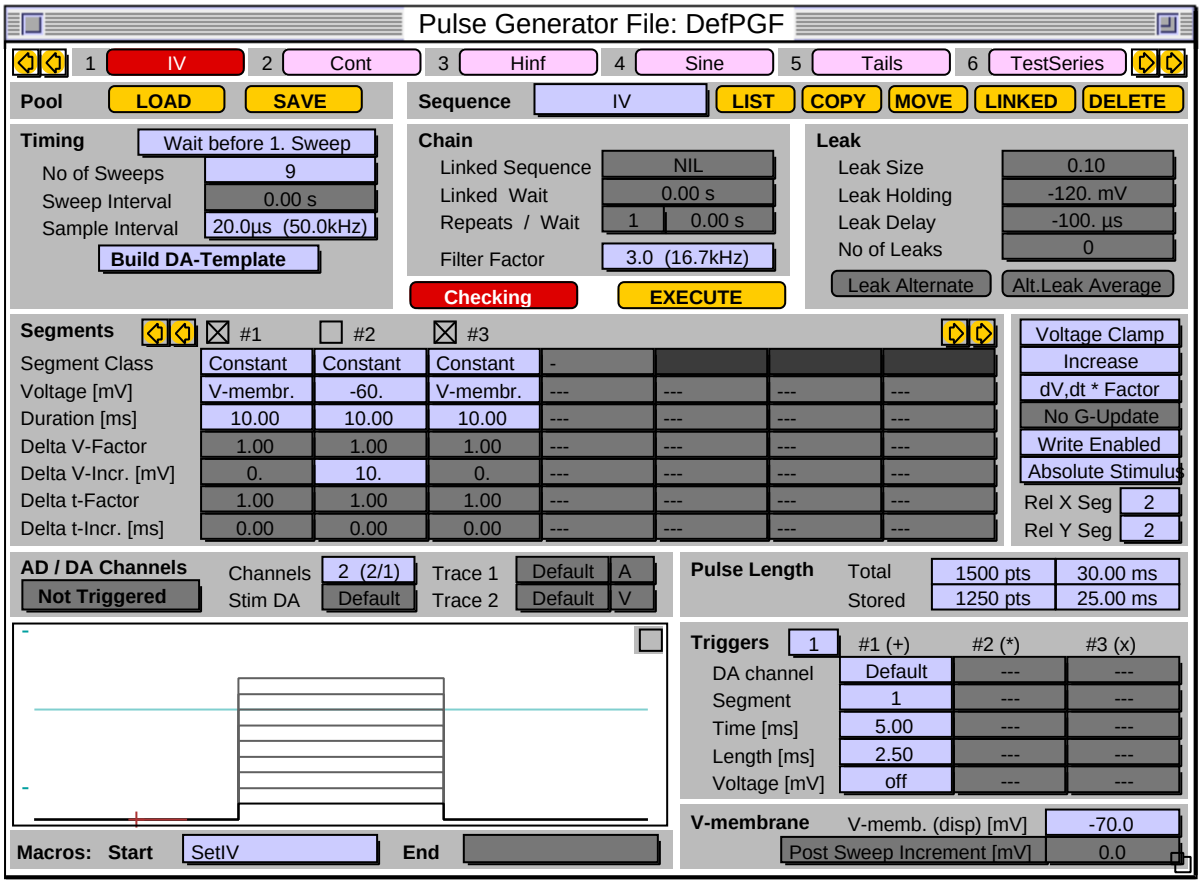
1. If EPC9 channels are used, their data scaling and logical Y-units are applied.
2. If the AD-channel is the *Current In* channel, then *I-Gain* and the Y-unit "A" are used.
3. If the AD-channel is the *Voltage In* channel, then *V-Gain* and the Y-unit "V" are used.
4. Otherwise, *Aux-Gain* and *Stimulus.Y-unit2* are used.
5. In *V-Clamp* mode *I-Gain* (*V-Clamp*) replaces *I-Gain*.

Special Features of the EPC9 Double and Triple

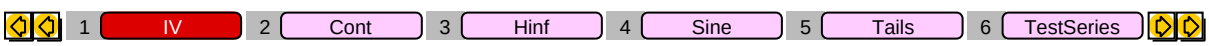
PULSE will automatically detect a connected *EPC9 Double* or *Triple*, and will pre-configure most settings. The fields **V-membrane Out**, **Current In**, and **Voltage In** cannot be specified. These AD- and DA-channels are internally connected and therefore cannot be changed.

Pulse Generator

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



Sequence Pool



Sequence Pool: The first row displays a section of the pool of available sequences. It is a paging bar in units of six sequences. Two arrows at each side allow to scroll through the available pulse protocols (the innermost arrows move in increments of

one page, i.e., six sequences; the outermost arrows move to the start/end of the sequence list). A sequence is selected by clicking on it.

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

If no *Pulse Generator File* is available, *PULSE* creates a default file (**DefPGF.pgf**). This file only contains one stimulation sequence, called *"Test"*. This sequence can be edited. More sequences are added to the pool by copying an already existing sequence (**Copy** button) or by clicking a free position in the pool. After modification of the existing pool of sequences, the entire *Pulse Generator File* (*PGF*) should be saved to disk (**Save** button). It can be saved under any name. *PULSE* automatically appends the extension **.pgf** to the file name. If this new *PGF* file should be loaded into the *Pulse Generator* as a default, the new name of the *PGF* file has to be specified in the *Configuration* window and the configuration file has to be saved.

Pool

LOAD / SAVE: Loads or saves the pool of available stimulation sequences (**.pgf** file).

The present file name is indicated in the title bar of the *Pulse Generator* window, e.g., *"Pulse Generator File: DefPGF"*.

Sequence

Sequence: This is the name of the present sequence which can be edited here.

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

```
PGF-File:,  Sequence 1: IV
EXTERNAL TRIGGER MODE: TrigNone
TIMING:
NumberSweeps: 9, SweepInterval: 0.000 s, SampleInterval: 20.00µs
Wait before 1. sweep:TRUE, FilterFactor: 3.0
CHAIN:
LinkedSequence: NIL, LinkedWait: 0.0 s, Repeats: 1, RepeatWait: 0.0 s
TRIGGERS:
1: Segment: 1, Channel: default, Time: 5.000ms, Length: 2.500ms, Am-
plitude: 5.000 V
RELEVANT SEGMENTS : X = 2, Y = 2, MaxSweepLength: 1250
Write Mode: Enabled, Absolute Stimulus, Post Sweep Inc.: 0.000 V
G and C Update: No Update
```

```
INCREMENT:   Mode : Increase,  LogIncrement: on
SEGMENTS:      Volt       Dur       VFact   VIncr      TFact   TIncr
1: Const.,    HOLD,      10.00ms,  1.00,   0.000 V,   1.00,   0.000 s
2: Const.,   -60.00mV,   10.00ms,  1.00,   10.00mV,   1.00,   0.000 s
3: Const.,    HOLD,      10.00ms,  1.00,   0.000 V,   1.00,   0.000 s
CHANNELS:
StimDA: default, Channel_1: default, Channel_2: default
Amplifier mode: VoltageClamp only
Macros: SetIV, none
```

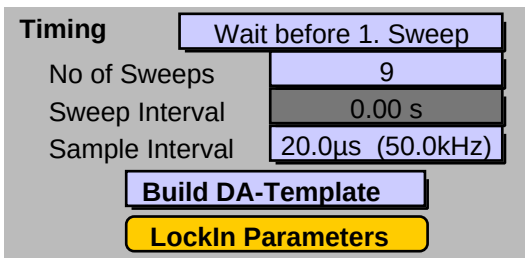
COPY: Duplicates the actual sequence into the first free position. A new name has to be entered.

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

LINKED: Switches to the linked sequence, if there is another sequence linked to the current one (see below, how to link sequences).

DELETE: Removes the present sequence from the pool.

Timing



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

No. of Sweeps: Determines the number of sweeps within one sequence.

Sweep Interval: The first sweep of a sequence has a waiting period during which the DAC template is computed and loaded. Then the first sweep is acquired. The next sweep will start **Sweep Interval** milliseconds after the beginning of the first sweep. *PULSE* will try to do the timing as exactly as possible, however there's minimum waiting period, the so called *"Sweep Gap"*, that depends on the speed of your computer system and the complexity and duration of the stimulus to be calculated and output.

Note: There is only an effective waiting time between pulses, if "Sweep Interval" is greater than "NumberLeak + 1" times the duration of a pulse plus "Leak Delay". This waiting time is measured in units of 1 ms.

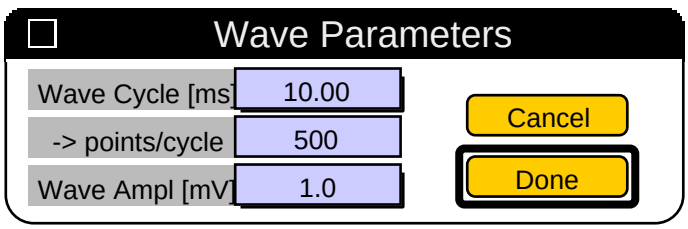
Sample Interval: The timing of data acquisition is given as *Sampling Interval* (in seconds) and as *Sampling Frequency* (in Hz). The minimum sample interval (in case no triggers are used) is 5 µs (200 kHz); the maximum interval is 60 s (0.017 Hz).

Note: The slow acquisition mode of PULSE (i.e. a sample interval slower than 65535 µs) is accomplished by continuously acquiring at a rate of 50 ms per point, averaging up to 8 samples every required time interval and ignoring the other samples.

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

Get File Template: Selecting *Get File Template* causes the DAC-stimulus template to be loaded from a file instead of being computed. The name of the template file must be: "PGF-path + StimulusEntryName + SweepIndex" (e.g., "HD:HEKA:Data:IV_1"). The data in the template file must be binary IEEE-short real (4 bytes) number values of the output-voltage (in volts), one value per output point. The template files may be placed in a folder named "StimulusEntryName " inside the "PGF-path" folder

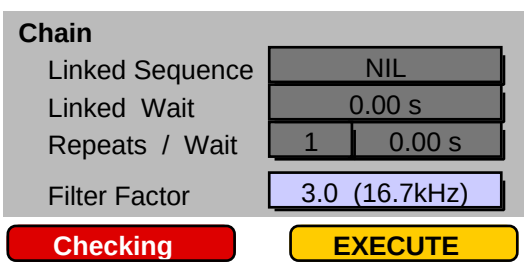
LockIn Parameters: This option allows to specify the sine characteristics for segments of type *SineWave* and *SquareWave*. It is only visible if the *LockIn* extension is active and the pulse protocol contains at least one sine- or squarewave segment (for further details please refer to the *LockIn Manual*). If the *LockIn* extension is inactive, the title of the button changes to **Wave Parameters**. This calls a dialog that allows you to specify the duration (**Wave Cycle [ms]**) and amplitude (**Wave Ampl [mV]**) of a single sinewave for segments of type *SineWave* and *SquareWave*. Note: the *Amplitude* is the half of the peak-to-peak amplitude. The field **points/cycle** is a display value only. It gives the number of points per sinus which is the cycle length divided by the sampling interval.



Chain

Linked Sequence: Stimulation sequences can be linked to other sequences within the pool or to themselves (**Repeats**).

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



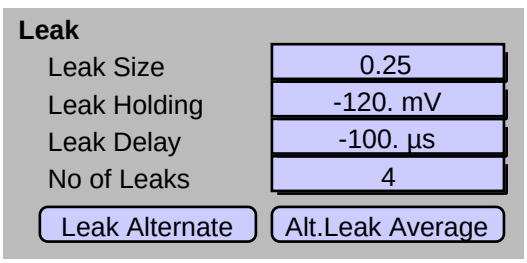
Repeats / Wait: If a sequence is linked to itself, the number of repeats as well as their interval has to be specified. The user is notified when a timing overrun occurs. To prevent overruns, the **Repeat Wait** and **Linked Wait** values are checked for consistency with the rest of the template when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

Filter Factor: The **Filter Factor** is implemented for the *EPC9*. It is used to define the automatic filter setting relative to the sample rate (activated by **Auto Filter** in the *Configuration* window). For the above example of **Sample Interval** = *250 µs* (4 kHz) and **Filter Factor** = *3*, a filter cutoff frequency closest to *1.33 kHz* (= 4 kHz / 3) will be selected. The suggested filter frequency is shown in parentheses.

Checking: If active, this option will check for any inconsistencies that might occur when entering values. When disabled, the validity checking of the sequence editing is suspended. This is convenient when one wants to perform multiple changes, especially when some intermediate steps would result in a (temporarily) faulty sequence. While checking is disabled, the **NOT Checking** control will flash. Upon re-enabling the checking, and when storing, linking, switching, or leaving the PGF-editor, the active sequence is checked and, if faulty, the user is notified and the mentioned operation is canceled, until the sequence is valid.

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

Leak Subtraction



Leak Size: Determines the amplitude of the leak pulses relative to the main pulse.

Leak Holding: Specifies the potential from which leak pulses are generated.

Leak Delay: This is a waiting time between the leak pulses and the main pulse. In between leak pulses no extra waiting time is supported. P/n pulses precede the test pulse if a negative **Leak Delay** is supplied or they follow it if **Leak Delay** ≥ *0*.

No. of Leaks: Arbitrary numbers of so-called P/n pulses are supported.

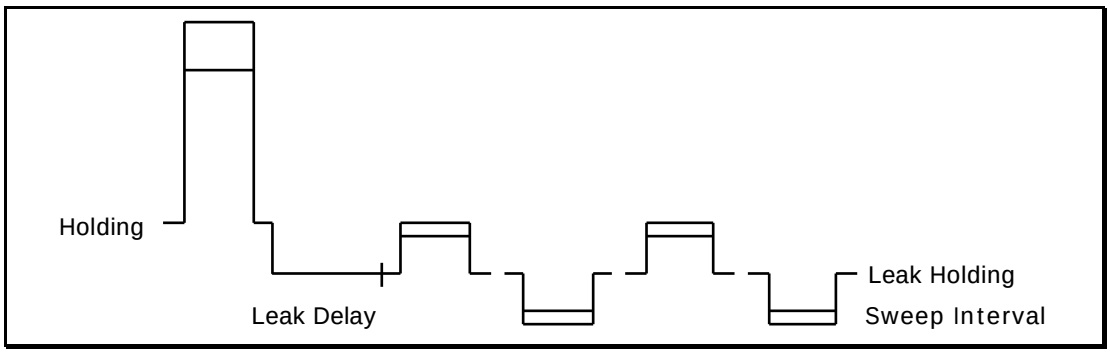
In this example **Leak Size** is *0.25* and **No. of Leaks** is *4*. This will generate a classical P/4 protocol with 4 leak pulses scaled down to a fourth of the original pulse amplitude. In the classical P/n protocol **Leak Size** is always equal to 1 divided by the **No.**

of Leaks. This is not necessary with the *PULSE* software: the leak pulses are summed and scaled with a factor of 1/(**No. of Leaks** * **Leak Size**).

Note: You can improve the signal-to-noise ratio if you increase ***Leak Size*** and ***No. of Leaks***, but be sure that the leak pulses do not reach the activation level of the channels you want to study.

Leak Alternate: If **Leak Alternate** is *On*, leak pulses of alternating polarity are generated.

Alt. Leak Average: This switch provides the option to generate leak pulses of alternating polarity while averaging to eliminate slow capacitive currents arising from the jump from *Holding* to *Leak Holding*. This procedure only gives ideal results if *# Averages* (in the *Oscilloscope* window) is an even number. The entire leak template of every other sweep is then inverted. The following figure illustrates a pulse template.



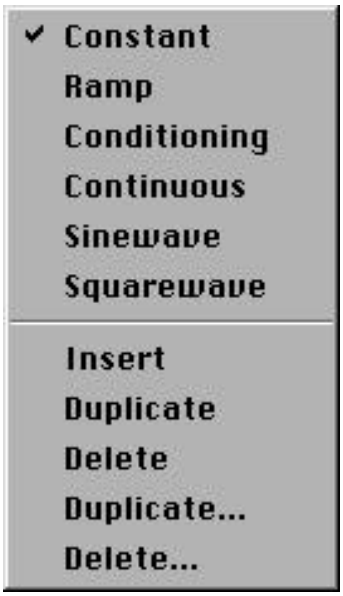
Example of stimulus template: No. of Leaks = 4, Leak Size = 0.25, Leak Delay = + 2, Leak Alternate = On, Sweep Count = 2. The gaps between the individual P/n pulses are determined by CPU time used for data processing.

Segments

A pulse pattern consists of an arbitrary number of segments (32000 maximum). Segments are shown as a horizontally scrolling matrix; scrolling is done by clicking on the arrows.

Segment Class: Segments can be the following:

Segments	#1	#2	#3
Segment Class	Constant	Constant	Constant
Voltage [mV]	V-membr.	-60.	V-membr.
Duration [ms]	10.00	10.00	10.00
Delta V-Factor	1.00	1.00	1.00
Delta V-Incr. [mV]	0.	10.	0.
Delta t-Factor	1.00	1.00	1.00
Delta t-Incr. [ms]	0.00	0.00	0.00



- **Constant**: Segment of constant voltage.
- **Ramp**: Segment with ramp from the voltage of the previous segment to the voltage of the current segment.
- **Conditioning**: Conditioning segment of arbitrary length. Triggers, continuous data acquisition, and P/n pulses are not supported for such a segment. The timing accuracy of conditioning segments is limited to the clock ticks of the computer clock (i.e., 1 ms).
- **Continuous**: Identifier for continuous data acquisition. Only the last segment can be of that class.
- **Sinewave**: Segment with sine characteristics. Amplitude and cycle length are defined in *LockIn Parameters* (see above).
- **Squarewave**: Segment with squarewave characteristics. Amplitude and cycle length are defined in *LockIn Parameters* (see above).

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

- **Insert**: Inserts a constant segment (default duration = 0).
- **Duplicate**: Creates a copy of the selected segment and inserts it at this location.
- **Delete**: Removes the selected segment.
- **Duplicate…**: Creates multiple copies of a number of segments. This feature is quite useful if one wants to create a tetanus-like pulse pattern. One only has to create two segments (pulse and inter-pulse segment); these are then duplicated together.
- **Delete…:** Deletes a specified number of segments at once.

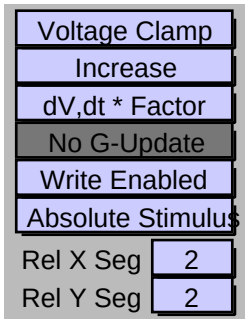
Voltage: The voltage of a segment is either a numeric value (in mV) or the membrane potential (*V-membrane*) at the time of sequence execution. *V-membrane* is activated by the checkbox in the upper row.

Duration: Duration of a segment in ms. Make sure that the durations of segments are even multiples of the sample interval. A warning is given if they are not.

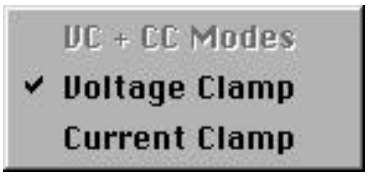
Delta V-Factor / V-Incr. / t-Factor / t-Incr. : Voltage and duration of a segment can be incremented between successive sweeps. There are two increment modes (see *Increment Mode* below). The magnitude of the increment is entered here.

Miscellaneous

Recording Mode: Stimulation sequences can be restricted to voltage-clamp or current-clamp modes. The following options are available:



- **VC + CC Modes**: Allows the stimulus to be executed in both modes. This option is not available any more, since a given pulse protocol only makes sense in *Current Clamp* or *Voltage Clamp* conditions, but never in both recording configurations. The switch is there for compatibility with *PGF files* created by older *PULSE* versions.
- **Voltage Clamp**: Restricts the protocol to *Voltage Clamp* modes. Stimulus amplitudes are entered in [mV], *Holding* corresponds to *V-membr*.
- **Current Clamp**: Restricts the protocol to *Current Clamp* mode. Stimulus amplitudes are entered in [pA], *Holding* corresponds to *I-membr*.



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

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

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

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

 switch (incrementMode) {
   case increase:   // first sweep comes first
     return (sweepCount);
   case decrease:   // last sweep comes first
     return (totalSweeps - sweepCount + 1);
   case interleaveIncr:
   case interleaveDecr:
```

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

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

There are two increment modes:

- **V, t * Factor**
- **ΔV, Δt * Factor**

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

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

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

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

for *Vfactor*, *tFactor = 1*:

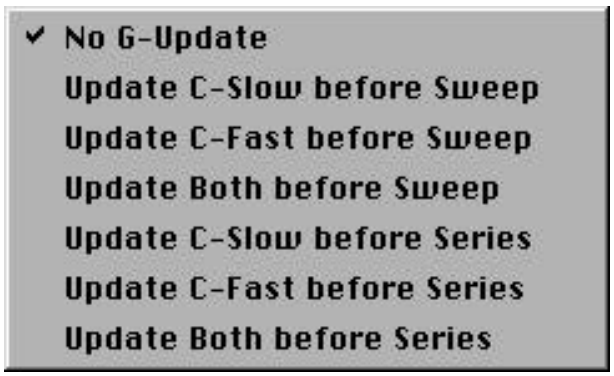
$$t_i = \text{Duration} + (i-1) * \Delta\text{t-incr}$$
$$V_i = \text{Voltage} + (i-1) * \Delta\text{V-incr}$$

for *Vfactor*, *tFactor ≠ 1*:

$$t_1 = \text{Duration};\ V_1 = \text{Voltage}$$
$$t_{i,i>1} = \text{Duration} + \Delta\text{t-incr} * \text{t-Factor}^{i-2}$$
$$V_{i,i>1} = \text{Voltage} + \Delta\text{V-incr} * \text{V-Factor}^{i-2}$$

In mode ΔV, Δt * *Factor* the first segment may be zero and is then logarithmically incremented. Let *Voltage = 0 mV*, Δ*VIncr = 1 mV*, and *VFactor = 2* then the series 0, 1, 2, 4 mV, … is obtained.

G- and C-Update: This list specifies when and how amplifier compensation is adjusted during data acquisition (for *EPC9* only). The fast (*C-fast*) and slow capacitance (*C-slow* and *G-series*) can be updated before each sweep or series. The options are:



- **No G-Update**
- **Update C-slow before every sweep**
- **Update C-fast before every sweep**
- **C-fast and C-slow cancellation before every sweep**
- **Update C-slow once before the series**
- **Update C-fast once before the Series**
- **C-fast and C-slow cancellation once before the series**

Since the values are stored in the *Pulsed Tree*, one can use this feature as slow capacitance tracking during pulsed data acquisition.

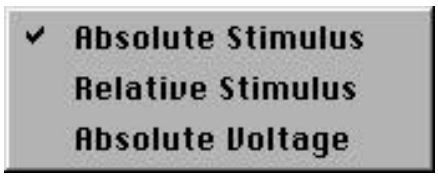
Note: G- and C-Update is performed only for the first sweep of an average. Off-line data averaging or off-line leak correction may become invalid if the amplifier settings are changed in between acquisition of sweeps of the same kind.

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

- **Write Enabled**: Data are saved to file.
- **Write Disabled**: If this flag is set, no data are written to disk even if the **Store** control in the *Oscilloscope* window is activated. This feature allows to execute test sequences or conditioning series that should not be stored (without turning **Store** off).
- **No Write no Show**: If this flag is set, no data are written to disk even if the **Store** control in the *Oscilloscope* window is activated and the data are not displayed on the oscilloscope. This option might be useful, e.g., to apply high-frequency conditioning pulses followed by a test pulse.

- **Write no Show**: If this flag is set, data are written to disk but the data are not displayed on the oscilloscope. This option will increase the maximal repetition rate of acquisition by not displaying nor analyzing a sweep during the acquisition.

Stimulus: This defines how the stimulus is applied:



- **Absolute Stimulus**: This is the default setting. A stimulus amplitude is always interpreted absolute. If a segment is set to *+10 mV*, +10 mV will applied. At the VOLTAGE-MONITOR output the stimulus will be scaled by the scaling factor from the *Configuration* window (e.g. *+0.1* with the *EPC9*):

$$V_{out} = \frac{V_{segment}}{StimulusScale}$$

- **Relative Stimulus**: If *Relative Stimulus* is activated, the segment voltages are interpreted relative to the holding potential. Assuming a holding potential of *-80 mV* and a segment voltage of *+100 mV*, an output yielding *+20 mV* would be created. The stimulus will be scaled by the scaling factor from the *Configuration* window:

$$V_{out} = \frac{V_{segment} + V_{hold}}{StimulusScale}$$

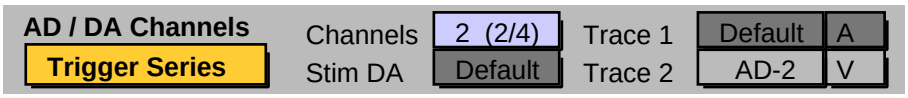
- **Absolute Voltage**: This outputs the voltage without any scaling:

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

Note: In the Current-Clamp mode, the command current will be output as a command voltage that is scaled by the factor ***CC-Stimulus Scale*** from the Configuration window (e.g. 1 pA / mV with the EPC9 or 2 pA / mV with the Axon 200). This parameter is available from the ***Stimulus Scale*** popup menu.

Relevant Segments: The **Rel X Seg** *(Relevant X-Segment)* specifies a segment of interest, which is mainly used as X-axis reference for later analysis. The **Rel Y Seg** *(Relevant Y-Segment)* specifies the segment where the analysis is performed (e.g., determination of peak current). For the measurement of an h∞ curve, for example, the *Relevant X-Segment* would be the conditioning segment of variable voltage, while the *Relevant Y-Segment* would be the test segment, where the peak current is determined.

AD/DA Channels



Trigger Mode: Determines if and how data acquisition is triggered by an external TTL pulse. The trigger input is located at the rear of the *EPC9* amplifier. The *EPC9* needs the **falling** edge of a trigger from 5 to 0 V! The default is *Not Triggered*, which means that stimulation is immediately elicited by the user within *PULSE*. Otherwise you have to activate the sequence and then one or more external triggers have to be applied.

- **Not Triggered**: No external triggering.
- **Trigger Series**: One trigger at start of series.
- **Trigger Sweeps**: A trigger is required for the main pulse and for each single leak pulse.
- **Trigger Sweep, first only**: One trigger is required to start the complete sweep, main as well as leak pulses.

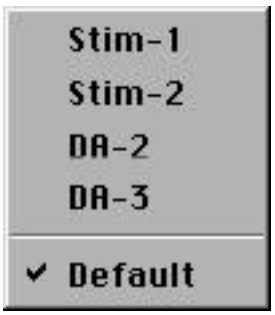
Channels: Number of channels acquired. The default is *1*, the second channel (**Channels** = *2*) may be used to simultaneously record the potential or an amperometric signal, for example. Some restrictions apply to the second channel:

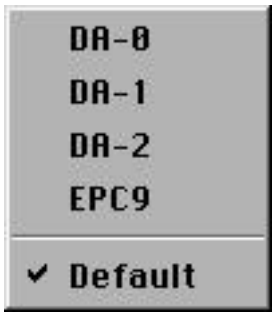
- The sampling interval must be larger than *5 µs*.
- No leak subtraction or zero correction is performed for it.
- It is acquired during the main pulse only; it is inactive during leak pulses.

The second trace can be shown on the screen by enabling the menu option **Display → Background Trace → 2nd Trace**. The number in brackets next to number of input channels gives the total number of *"effective"* input and output channels, i.e. physical hardware channels used.

Note: If you are using the FURA extension to PULSE, one to two additional AD-channels will be acquired (the signal of the photomultiplier, and ev. A sincronization trace) and possibly one additional DA-channel may be output (the stimulus for the monochromator). The number in brackets shows the "effective" number of channels. Thus, if your photomultiplier gets sampled via AD-2 and ***Trace 2*** has been defined to the same AD-channel, only one "effective" channel is acquired.

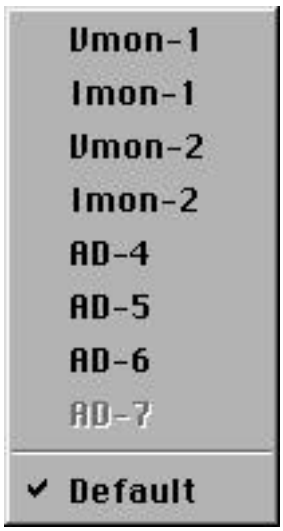
Stim DA: The DA channel for stimulation has to be specified. If *Default* is selected, the channel specified in the *Configuration* window is chosen. The figures show the options for the *EPC9* and the *EPC9/2*. In the later case (and with the *EPC9/3*) the pop-up list for DA-selection contains the function names: *Stim 1* is the explicit stimulus output of the first amplifier, and *Stim 2* is the output of the second one, while *Default* is the output of the *active* amplifier.





Trace 1 / Trace 2: Assignment and units for input channels have to be provided. Defaults are *"A"* for current and *"V"* for voltage. In the shown example, stimulation is output via the **Default** *EPC9 Stim. Out* channel (DA 3). Trace 1 is recorded from the **Default** *EPC9 I-mon* channel. The unit *"A"* is dimmed, because it depends on the recording mode and will be set automatically (e.g., *"V"* if *Current Clamp* is selected as recording mode). The second trace is to be read via **AD-2** and has the unit *"V"*.

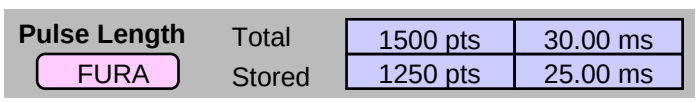
The following applies for the *EPC9 Double* and *EPC9 Triple* only: The pop-up lists for DA- and AD-selection contain the function names (**Vmon X**, **Imon X**). *Stim-DA*, *V-Monitor-AD* and *I-Monitor-AD* of the *"active"* amplifier are used for stimulation and acquisition, when one selects the **Default** entry. Selecting a specific Trace-AD (e.g. *Vmon 1* or *Imon 2*) will cause the stimulation and acquisition using those fixed channels, irrespective of which amplifier is the *"active"* one. *PULSE* will use the known current- and voltage-gains as well as the modes (*On-Cell*, *Whole-Cell*, etc.) of the addressed amplifier. This allows e.g. to keep the first amplifier in cell-attached mode with a high current gain, while reading at the same time from the second amplifier in the whole-cell mode with a medium current gain setting.



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

Pulse Length

Total: The number of total points and the total time of a sequence is



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

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

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

FURA: This will activate stimulation of a monochromator and acquisition of a photomultiplier, if the *FURA* extension to *PULSE* has been purchased and activated. Please refer to the *FURA* manual or contact HEKA if you need more information about the *FURA* extension.

Video: This will activate stimulation of imaging system, if the *Video* extension to *PULSE* has been purchased and activated. Please refer to the *Video* manual or contact HEKA if you need more information about the *Video* extension.

Triggers

The *Triggers* can be used to control external devices such as oscilloscopes, valves, flash lights, etc., and, because they can be output via analog channels with specified amplitude and duration, they are also useful as additional stimulation output (e.g., one cell could be stimulated via a field electrode with a trigger pulse, while a signal is recorded from another cell with the main patch clamp amplifier using the main pulse protocol).

Triggers	3	#1 (+)	#2 (*)	#3 (x)
DA channel		DA-0	dig-0	DA-2
Segment		1	2	3
Time [ms]		5.00	2.00	0.00
Length [ms]		2.50	2.50	10.00
Voltage [mV]		5000.	TTL-high	2000.

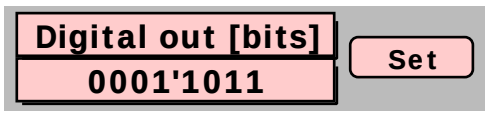
There are up to three triggers supported. The number of used triggers is defined by the entry **Triggers**. A trigger can be output via an DA channel (*DA-1*, ... *DA-2*) or via one of the 8 digital trigger lines (*Dig-0*, ... *Dig-7*) available at the DIGITAL-TRIGGER-OUT or DIGITAL-I/O interface on the backside of the *EPC9* or via the TTL OUTPUTs on the front side of the ITC-16 and ITC-18. The trigger DA channel, time, duration, and amplitude are to be specified within one segment of the pulse; it can extend up to the end of the pulse template. **If the trigger actually exceeds the pulse template, i.e. it is so long that it would end after the sweep, it is not reset after the pulse**. This feature can be used to e.g. turn valves on or off.

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

The symbols of the triggers as used in the sequence cartoon (see below) are given in parentheses.

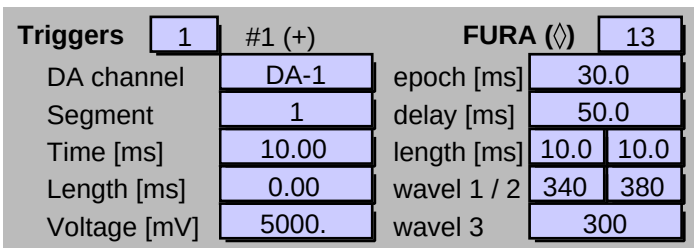
The digital trigger lines (*Dig 0*, ... *Dig 7*) are directly available if you are using an *ITC-16, ITC-18, EPC9/2* or *EPC9/3*. If you have an *EPC9* you will need the *Digital Trigger Box TIB14* in addition (please contact HEKA for more information). The following rules apply to the digital triggers:

- The **Digital out (bits)** control in the *Amplifier* window defines the basic port settings, see chapter "EPC9 Amplifier" above (Default: all trigger lines cleared or "*low*").



- Digital triggers are forced to be "*set*" during the **Length**, if its amplitude is greater than 0V and displayed as *TTL-high*.
- Digital triggers are forced to be "*cleared*" during the **Length**, if its amplitude is zero or negative and displayed as *TTL-low*.
- Triggers are unaffected outside the trigger **Length**.
- The digital trigger template is build segment after segment, and within a segment according to the trigger index.

If a *FURA* protocol is running, the third trigger is not freely available any more, since that DA-output is used to stimulate the monochromator (see the manual of the *FURA* extension for more details). The second channel is still available but it is hidden below the *FURA* settings. To change the parameters of the second trigger temporarily disable the **FURA** button in the **Pulse Length** section and reactivate it again afterwards.

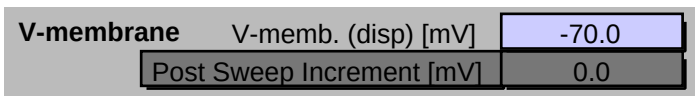


How to set the first trigger correctly

The first of the three triggers is a special one: it is used to determine the start of data display and storage; data before the first trigger are not stored! For this purpose the DA channel for a trigger can be set to *Off*; in this case no signal is output but the location of the trigger (if it is the first one) is used to set the start time for data storage. If no trigger has been defined, *PULSE* will store the whole sweep.

Since the first trigger specifies from where data are actually displayed and stored to disk, its position has implications to the recorded data. This is of particular importance when the voltage of the first segment is not equal to the holding potential. In this case the step from the holding potential to the potential of the first segment will cause a transient capacitive current. The same would be true if *Leak Holding* is not equal to the holding potential. In both cases there cannot be a perfect digital capacitance cancellation. Therefore, one can create a first segment with a duration such that this initial capacitive transient is over before the actual test pulse starts. The first trigger is then used to define this time, i.e., the time when the baseline is stable enough to be used as first segment (note that the zero current is calculated from the segment after the first trigger). Typical values for the decay of the transients while recording from small cells or patches are a few ms. If a baseline of 5 ms is desired, a first segment of duration 15 ms should be created and the time for the first trigger should be set to 10 ms.

V-membrane



V-memb: This control displays the presently selected membrane potential. It is **only** used for the sequence cartoon as reference and can be changed without actually affecting the membrane potential in the *Oscilloscope* window (*V-memb)* or *Amplifier* window (*V-membrane*).

Post Increment: This is used to change the holding potential after sequence execution to a new value (e.g., for the acquisition of steady-state inactivation curves). The new holding potential is indicated by a small bar in the sequence cartoon at the end of the stimulation. The options are:

- **Post Sweep Increment [mV]** : Increments *V-membrane* after each sweep.
- **V-membrane [mV]**: Forces *V-membrane* to the specified value after the series.
- **Post Series Increment [mV]**: Incre-

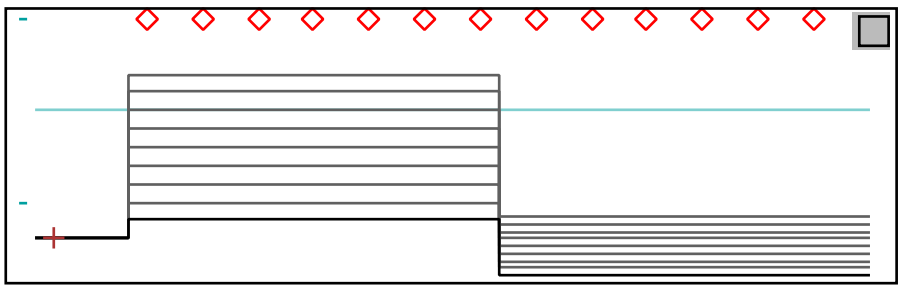
ments *V-membrane* after a series.

- **Last Segment Amplitude:** At the end of the sweep,*V-membrane* to is set to the amplitude of the last segment.

Macros

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

Stimulus Template



After each editing operation, the stimulus template is refreshed to reflect the changes made. The small horizontal bars on the left side indicate the ±50 mV level. Triggers are indicated by markers with a horizontal line indicating the length of the trigger pulse. Sweeps other than the first one are shown as dashed lines. If part of the pulse pattern exceeds the DA limits, the forbidden voltage region is indicated in the picture by shading. The corresponding warning box *"Segment with too large voltage encountered"* only appears once. The next holding potential after application of *Post Increment* is indicated by a bar a the end of the stimulation. Segments drawn in red color refer to the *relevant* segment. If a *FURA* protocol is running, each sample of the dual-wavelength excitation is shown as a marker in the top.

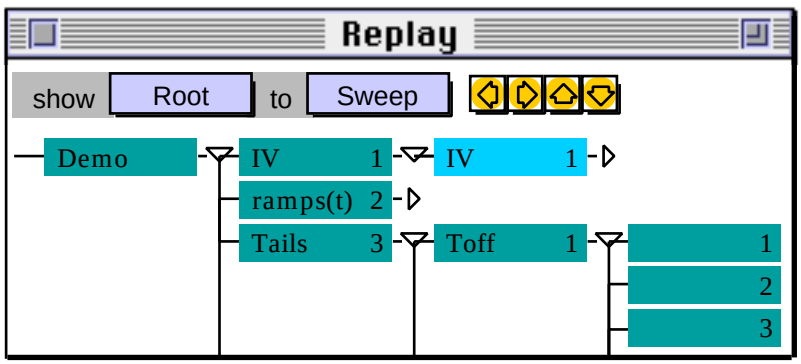
Note: If you want to disable the display of the stimulus template because your computer is too slow and the pulse protocol is too complex make it "invisible": click on the small box in the upper, left corner of the picture. The cartoon will be replaced by a checkbox which allows you to bring back the drawing of the stimulus template.

Error Handling

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

Values in the *Pulse Generator* controls are rounded to the exact values as displayed. This prevents unexpected results caused by rounding problems, such as the *Sample Interval* being an odd number. The drawback is that it can thereby happen that a segment duration cannot be both, a multiple of the *Sample Interval* and the exact value displayed. This, however, can only occur, if the *Sample Interval* is not a multiple of 10 µs.

Replay



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

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

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

The highlighted part of the data tree is referred to as *"Target"* throughout this manual (*Target Group, Target Series* or *Target Sweep*). While the *Replay* window is in front, you can activate several functions on the *Target* using the keyboard. Some of these functions only apply if the data file was opened with write permission (**File → Open Modify...** or **File → New...**) while others are restricted to certain kinds of targets. These functions are also accessible via the drop-down menu *Tree* that becomes active when the *Replay* window is selected (see *Chapter Menus*). The keyboard equivalents are shown below:

KEY	Function	Applicable to
A	Average	Group, Series
C	Compress by factor of two	Root, Group, Series
D	Delete	Group, Series, Sweep
E	Edit entry	Root, Group, Series, Sweep
P	Show PGF template	Group, Series, Sweep
R	Select as reference	Series, Sweep
S	Edit solution	Series
T	Enter text	Root, Group, Series, Sweep
Y	Show EPC9 state in *Notebook*	Series, Sweep
X	Export according to *Export Mode*	Root, Group, Series, Sweep
Z	Recalculate *Zero Current* value	Root, Group, Series, Sweep
RETURN	Display and analyze	Root, Group, Series, Sweep
CURSOR	Move in data tree	Root, Group, Series, Sweep

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

Besides a running index, text is displayed in the icons of the tree entries. Usually the *Root* icon contains the file name, the *Group* icon contains the experiment number, e.g., *E-12* and the *Series* icon shows the name of the *Stimulation Sequence*, e.g., *"IV"*. During data acquisition, the *Sweep* icon is empty. It holds the *Sweep Label*, which can be entered with the *Text* option. If, for example, a sweep with a channel opening should be identified among other sweeps with blanks, the sweep with the channel is

to be selected, then press 'T' for text and enter a sweep label, e.g., *"Channel"*. This way it is easy to find certain sweeps for analysis or printout.

Note: The text editing functions of the Operating System, like 'CMD' / 'CTRL' + 'C' for copy or 'CMD' / 'CTRL' + 'V' for paste can be used.

The raw current traces as well as the individual entries of the *Pulsed Tree* data structure can be edited. In addition, the commands of the **Buffer** and **Mark** menu allow to perform quick data analysis without the need of further analysis programs. All editing operations are invoked inside the active *Replay* window by keys or by the entries of the drop-down menu **Tree**. This section describes how to edit stored data and how to extract and manipulate information from a data file using the sweep buffer commands. If the active data file was opened using **New...** or **Open Modify...** in the **File** menu, stored data can be modified by several commands.

Editing Raw Data

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

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

Editing the Pulsed Tree

Even if one is careful, it will happen that one forgets to enter some parameters during an experiment. Therefore, each entry in the *Series Record* can be edited at any instant. Select a target in the tree and type 'E' or select **Edit** from the drop-down menu **Tree**. A small dialog will appear that holds a list of parameters to be edited, and the controls **Modify** and **Done**. If the file was opened as read only, then one can only **Display** the entries. Among the parameters are entries that point to higher branches in the tree, e.g., *All Sweeps*. By using such an entry, many parameters of the same kind can be edited at once. The entries that can be displayed or modified for each tree element are listed below:

Edit Root:

- All Sweeps
- All Series
- All Groups

Edit Group:

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

Edit Series:

- All Sweeps
- Comment
- Time
- Timer
- Bandwidth
- Background Noise
- Y-unit 1
- Y-unit 2
- UserParam 1: Name
- UserParam 1: Value
- UserParam 1: Unit
- UserParam 2: Name
- UserParam 2: Value
- UserParam 2: Unit
- External Phase
- Temperature
- Cell Potential
- Pipette Pressure
- External Solution
- Internal Solution
- Pipette Potential
- Pipette Resistance
- Seal Resistance
- Recording Mode

Edit Sweep:

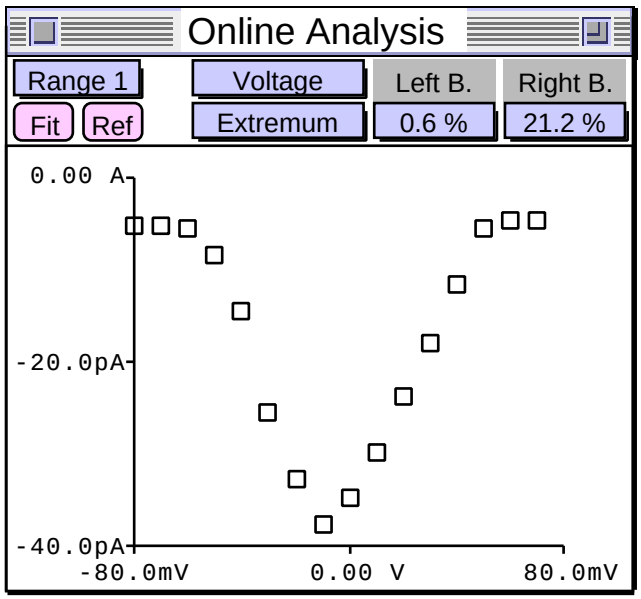
- Label
- Time
- Timer
- Gain 1
- Gain 2
- Zero Current
- C Slow
- G Series
- Rs-Value
- M-Conductance

Editing the Stim Tree

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

Online Analysis

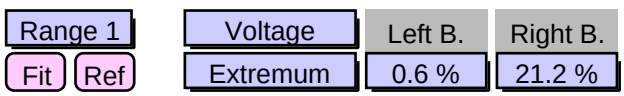
The online analysis is thought to give a quick overview of some characteristics of the just acquired or replayed data on the level of a series. For the determination of values for the abscissa and the ordinate, the *Relevant X-Segment* and the *Relevant Y-Segment* as specified in the *Pulse Generator* plus the respective relative segment offsets are used. *Pulse* will automatically plot the analysis to this window after or during execution of a series (based on the settings made in the various controls inside this window). To access additional plot options, click into the graph plot area. The current plot will be erased and the plot options will be shown.



Type of Online Analysis

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

- **Range 1**
- **Range 2**

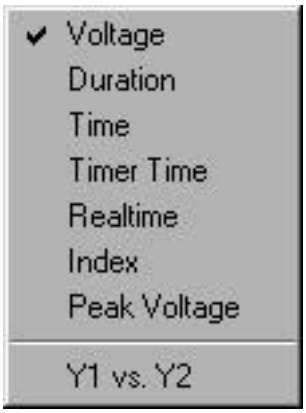


The default range is determined by the relevant segment as specified in the *Pulse Generator* plus the respective relative segment offsets. The analysis of the selected range will be shown in the graph.

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

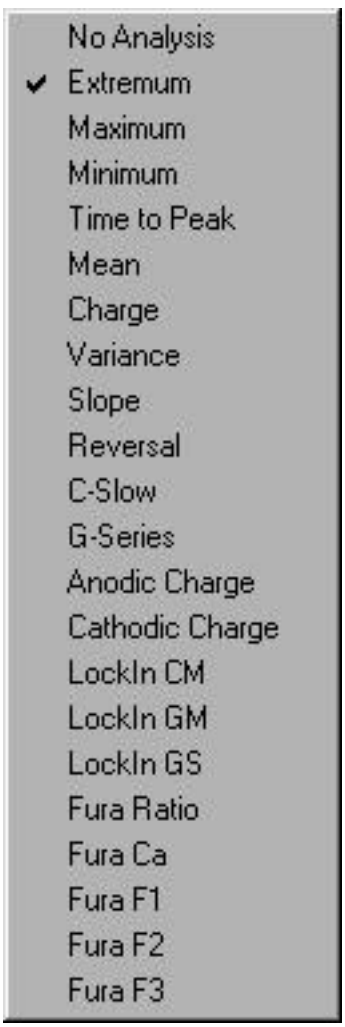
Ref: This control is highlighted when a reference analysis was selected (For selecting a series as reference see *Chapter Menus…Tree Menu* and *Chapter Replay*). Click on this control results in deactivation of the reference analysis.

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



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

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



No Analysis: Skips data analysis.

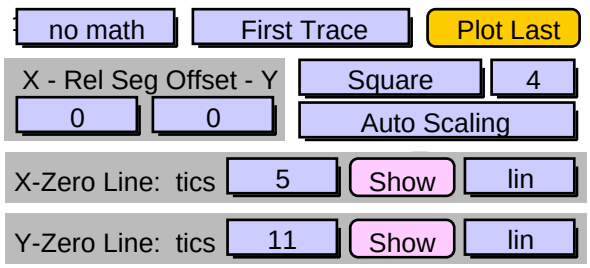
- **Extremum**: Extremum of current: the larger of the maximum or minimum.
- **Maximum**: Maximum of current.
- **Minimum**: Minimum of current.
- **Time to Peak**: Time to peak of current within specified segment.
- **Mean**: Mean of current.
- **Charge**: Integral of current.
- **Variance**: Variance of current.
- **Slope**: Slope of current (by linear regression).
- **Reversal (theo)**: Finds next zero crossing, starting at the left bound (see below) and searching to the right, using an average of 3 data points. Vref is then computed from the stimulus template.
- **Reversal (2.trace)**: Equivalent to “Reversal (theo)” above, but gets Vref from the voltage at the corresponding position in the 2. Trace (which is usually the voltage trace).

- **C-Slow**: Slow capacitance taken from each sweep (no analysis required).
- **G-Series**: Series conductance taken from each sweep (no analysis required).
- **Anodic Charge**: Integral of positive current.
- **Cathodic Charge**: Integral of negative current.
- **LockIn CM**: Capacitance taken from the *Lock-In* extension.
- **LockIn GM**: Membrane conductance taken from the *Lock-In* extension.
- **LockIn GS**: Series conductance taken from the *Lock-In* extension.
- **Fura Ratio**: Ratio taken from the *FURA* extension.
- **Fura Ca**: Free calcium concentration taken from the *FURA* extension.
- **Fura F1 / F2 / F3**: Fluorescence at excitation wavelength 1, 2 or 3 taken from the *FURA* extension.
- **Video R1 … R8**: Values of the regions of interest taken from the *Video* extension.

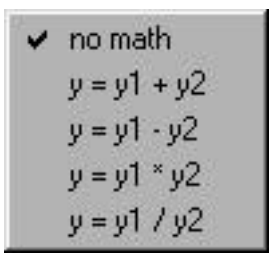
Left / Right Bound: Within a segment, *Left Bound* and *Right Bound* (in %) determine the actual time limits. These parameters can also be set using the cursors of the *Oscilloscope* window. They can be outside the limits of the selected segment, i.e., they can be smaller than 0% and greater than 100%.

Display Options

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



Math: Supplies different calculations on the two results. If a further calculation is required (i.e. the **Math Type** is set to anything else than *no math*), the *Online Analysis* window will show the calculation instead of the two online results.



- **no math**: No further calculations.
- **y = y1 + y2**: Calculates and shows the sum of the two *Online Analysis* results.
- **y = y1 - y2**: Calculates and shows the difference *Range 1* minus *Range 2*.
- **y = y1 * y2**: Calculates and shows the product.
- **y = y1 / y2**: Calculates and shows the quotient *Range 1* divided by *Range 2*.

Trace: The analysis is applied to the selected trace:

- **First Trace**
- **Second Trace**

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

Relevant Segment Offset: The analyzed segment always defaults to the relevant segment specified in the PGF file. It may be changed here by setting an x- and y-offset relative to the relevant segment.

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

- Point Symbol
- Plus Symbol
- Star Symbol
- Diamond Symbol
  - Cross Symbol
  - Square Symbol

Zero Line - X-Y Tics: Sets the number of tics plotted on the axis in case *Show* is activated. A value of zero suppresses the display of axis tics.

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

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

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

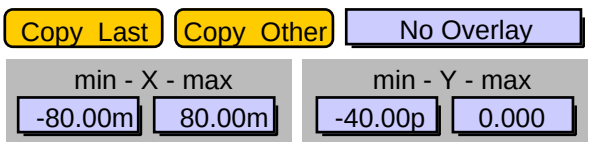
- Fixed Scaling: Online analysis results can be shown in the graph immediately after acquisition or replay of a sweep, if the scaling is known from the beginning. Therefore, a *Fixed Scaling* option is provided and the X-Y-scaling of the analysis graph can be specified. Now, the program knows the scaling before completion of a series, i.e., the online analysis results can be displayed in the graph after each sweep, which makes it a true online analysis.
- Auto Scaling: After all sweeps of a series have been acquired or replayed, the maximal and minimal values of the currently selected abscissa and ordinate values are determined and used to scale the graph.

- Cont. Auto Scale: This selection allows to display the Online results with auto scaling immediately after every acquired Sweep (beginning with the second Sweep).

Note: If a timer function ("Time", "Timer", or "Real Time") is selected as abscissa, data points are plotted in the given bounds until the time exceeds the high end of the abscissa scale: this causes a wrap-around of the axis scaling. Thus, the graph is cleared and redrawn from the left end (unless "Overlay" is selected, see below).

Scaling

If **Fixed Scaling** is selected, a further set of graphing options will appear:



Copy Last: This will copy the scaling of the last plot range into the fixed min / max controls. The scaling of *Range 1* will be used, if two ranges are available.

Copy Other: Copies the "other" scaling, i.e., if *Range 1* is currently active, the settings of *Range 2* will be copied and vice versa.

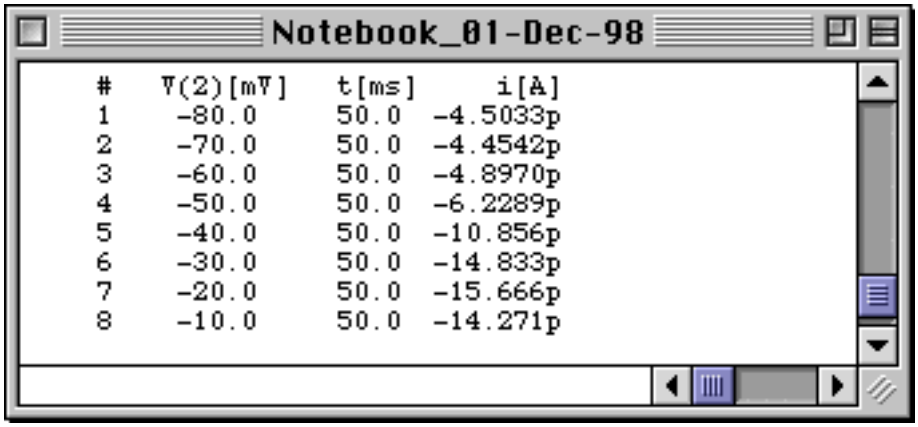
Overlay Mode: This determines how often the graph window is wiped out:

- **No Overlay**: This is the default mode, whenever performing a new analysis, the graph window will be cleared.
- **"Time" Wrap**: This suppresses the clear function before wrap-around.
- **Overlay + "T"-Wrap**: This suppresses to wipe the graph before a new series, i.e., it can be used to overlay the analysis of several series. The graph can be wiped by just clicking into it.

Axis Scaling: The axis ranges for the fixed scaling can be entered as min / max values.

Exporting Data

With the online analysis features classical analyses like current-voltage relationships, h_∞ plots, or τ_h plots can be performed easily during data replay and acquisition. The result of the *Online Analysis* is also displayed in the *Notebook*. From the *Notebook* these data columns can be written to disk or copied to the clipboard, but only when the option *Buffered Output* is selected in the *Notebook* menu, see chapter *Notebook*.



Alternatively, the information written to the *Online Analysis* window can be output by using the **Tree → Export** function if the **Export Mode** includes **Online Analysis**.

Changing the Size of the Analysis Window

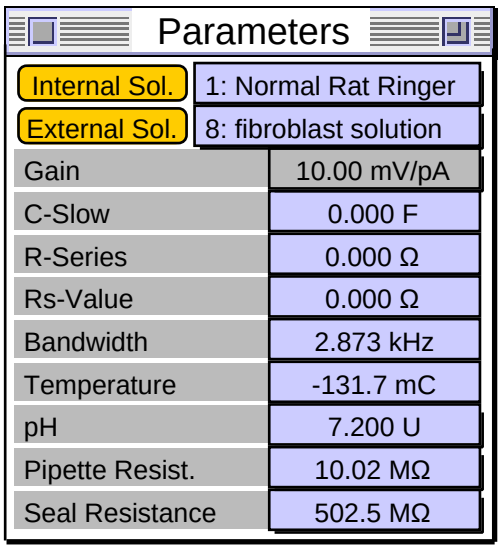
There are two ways to change the size of the *Online-Analysis* window:

1. The shortcut: Hold down the 'SHIFT' key while you resize the window.
2. The procedure described in the chapter *User Interface, subchapter 'Tutorial: Changing the Size of the Oscilloscope Window'*. That feature is not as easy, but it allows to define the position and graphic appearance of each single control in the window.

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

Parameters

The parameters as selected in the *Configuration* window (see *Chapter Configuration*) are shown in a separate window. They are determined before acquisition of each series or each sweep (depending on the parameter). In the test pulse mode the parameters are determined before each pulse, otherwise they are updated continuously, i.e., the repetition interval is just limited by the CPU time used.



Solution Numbers:

The top row displays the currently used identifiers for internal and external solutions. The solutions are given as integer numbers (0 to 2'147'483'648), when the "solution source" is set to "manual" in the *Configuration* window. The solution names are listed in a pop-up list, when the "solution source" is set to "DataBase" in the *Configuration* window.

Internal Sol. / External Sol. : These buttons are used to access the solution menu. In this menu new solutions can be specified or can be selected from the common solution data base. For more information concerning the solution files see *Chapter Solutions*.

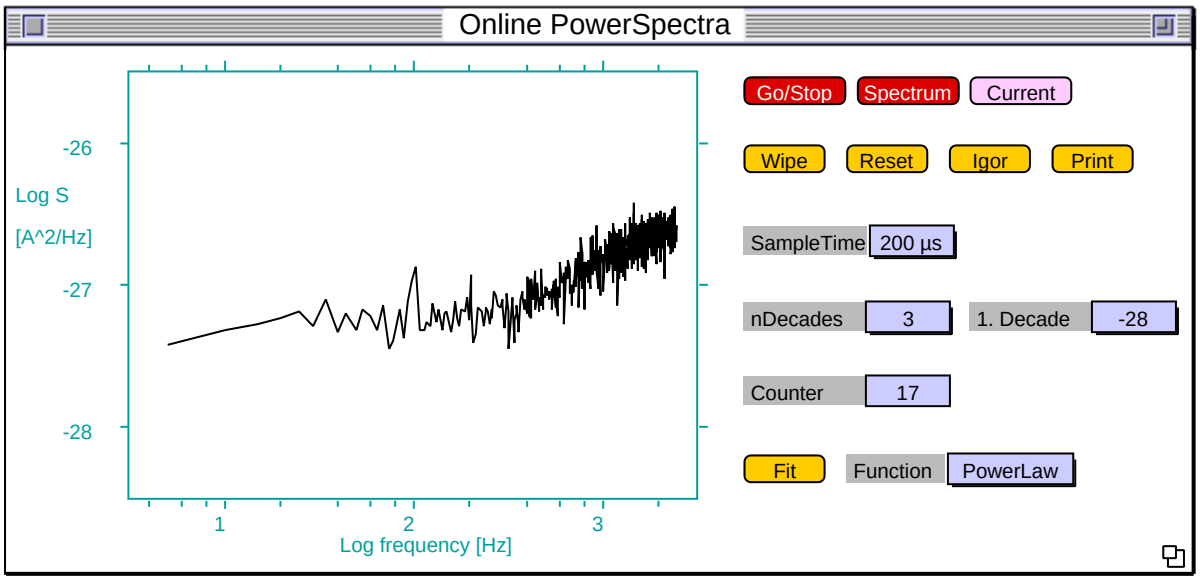
Parameter Values:

Apart from *RMS Noise*, *Pipette Resistance*, and *Seal Resistance*, parameters whose source was specified to be *Default* can be edited in this window (in the case shown above: pH (pH is a user parameter)); i.e., one does not need to edit the corresponding entries in the *Configuration* window. Parameters that are read from other sources cannot be edited.

After a replay action the parameter list is updated according to the replayed data. Mouse click in a window other than the *Replay* window restores the old actual parameter settings.

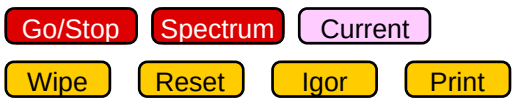
Power Spectra

Power Spectra can be measured, displayed, and averaged online. This mode is accessed via the drop-down menu **PULSE** → **Spectra** or by pressing 'F10' (MacOS). It is helpful for troubleshooting and for testing the frequency response of the recording system. The power spectrum is repetitively acquired, averaged and displayed.



Go / Stop: Starts or stops acquisition and accumulation of power spectra.

Spectrum: If enabled, the display of the spectrum will be updated after every FFT cycle.



Current: Displays the current trace in red color in addition to the power spectrum.

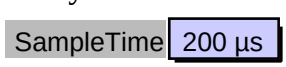
Wipe: Clears the display.

Reset: Restarts accumulation of power spectra.

Igor: Outputs the spectrum in "IGOR Text" format.

Print: Prints the spectrum.

Sample Time: Sets the sample interval. This determines the frequency band because a fixed number of 1,024 data points is always used for FFT calculation. Available sample intervals are:

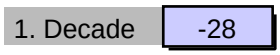


- **2 ms**
- **200 µs**
- **20 µs**
- **5 µs**

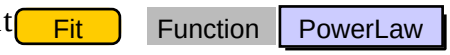
Counter: This is the number of spectra that were accumulated and averaged.

n Decades: Number of spectral density decades to be displayed.

1. Decade: First displayed decade.



Fit: Draws the fit function. Clicking on it allows to ent parameters for the fit function.

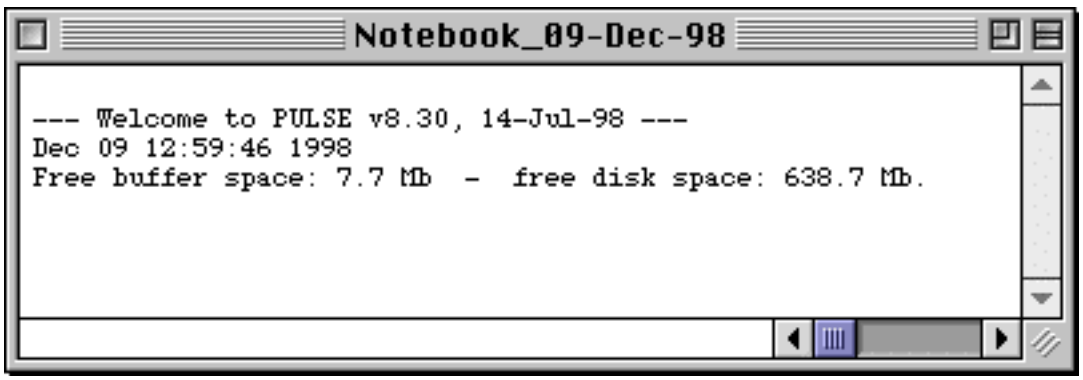


Function: Simple functions can be drawn; no automatic fit is performed. In the functions to follow f denotes the frequency, and S the spectral density.

- **Power Law** $S(f) = S_0 + \text{slope} \quad f^{\text{exponent}}$
- **One over** f $S(f) = S_0 + \sum_i \frac{a_i}{1 + f/fc_i}$
- **Lorentzian** $S(f) = S_0 + \sum_i \frac{a_i}{1 + (f/fc_i)^2}$

Notebook

The *Notebook* window is used to display messages and warnings of the program, such as error messages, analysis results, and information about displayed data. The content of the *Notebook* can be stored on disk; its maximal size has to be specified in the drop-down menu **Notebook → Set Length**. To keep a better log of an experiment, the names of opened files and of executed series are written to the *Notebook*.



When the window is activated (by clicking with the mouse pointer into it), the text editing functions are activated and applicable in the *Notebook*. Thus, the *Notebook* is basically an editor window of the memory-resident text file *Notebook*. Therefore, one can modify text in the *Notebook* just as in any other text file. This option can be used to add further information to the text file, or to get rid of messages that should not be stored to the disk file.

The applicable menu commands are described in the **Edit** and **Notebook** menus (see *Chapter Menus*).

Note: Text written by PULSE to the Notebook is not stored, when the option "Buffered Output" is not selected. The ***Cut***, ***Copy***, and ***Paste*** commands copy to and from the clipboard.

Solutions

In most electrophysiological experiments bathing solutions are changed during the experiment. Thus, it is of great importance to keep track, which solutions were on both sides of the membrane.

Besides using the text lines on all *Tree* levels (*Root*, *Group*, *Series*, *Sweep*) for the identification of the solutions, *PULSE* provides two additional ways to do so. A simple one is to use the entries **Internal Solution** and **External Solution** in the *Parameters* window. They are numbers that identify a certain solution. These numbers can be the entry of an external list of solutions, or the index of a specific solution in a *PULSE Solution Data Base* (see below).

For each data file a solution data base (***.sol**) can be created. Such a data base is a *Tree* of solution entries ordered by the given identifying index. How is such a solution file created? At first the *Solution Timing* in the *Configuration* window has to be set to something other than *Off*. This means that for each new index (either *Internal* or *External Solution*) an entry in the solution data base is created. Then, after acquisition of a series, *PULSE* checks the active solution file for the existence of both indices. If not present, a new entry has to be created. Several possibilities are provided to do this:

- The first is that *PULSE* searches the common solution data base for this entry. The name of that common data base is defined in the *Configuration* window. This search is done when the option for *Solution Source* is set to *Data Base*. If the desired solution is not found in the data base, one can enter the new solution manually. This means that after a series has been acquired with an unidentified solution index, the solution dialog will come up and will request entry of the specifications of this new solution. During a time-critical experiment in which delays due to entering new solutions are not desired, another option may be used (see below).
- The options in *Solution Timing* (*After Series*, *After Group*, and *End of File*) are used to delay the input to the end of a series, to the closure of the active group, or the active file, respectively. Thus, only when a new group or a new file is created, one is requested to enter the missing solutions.

Using Index

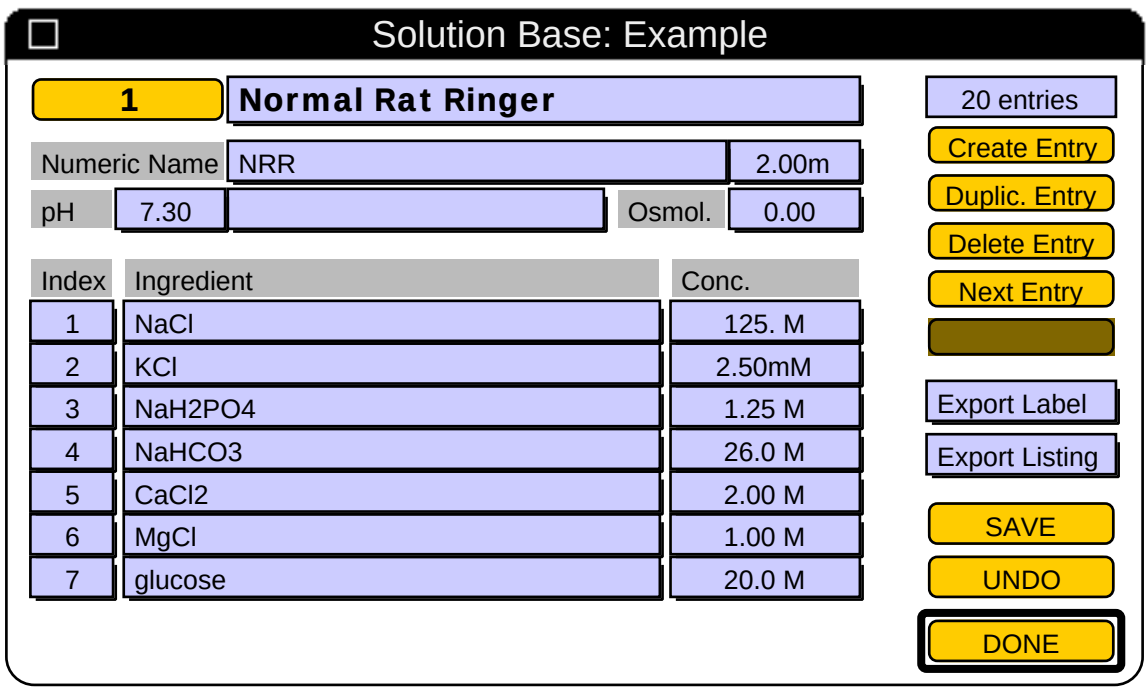
The indices can be given to solutions in any arbitrary way but it certainly is of advantage if one sticks to some consistent concept in order to be able to identify solutions by their index easily. Since the solution indices are numbers between 0 and

2’147’483’648 (many more than you will find bottles in the laboratory), there is plenty of freedom to organize them. Here is an example:

Usually one has several standard solutions, which are frequently used and modified slightly for various experiments. One could assign numbers divisible by *1000* or *10000* to them. Then one has *999* or *9999* possibilities for modifications of this solution, respectively. An example would be that certain concentrations of toxin are added to the standard solution *1000* yielding numbers *1001* through *1099*. Another toxin could occupy the numbers *1100* through *1199*, etc.

Note: The indices given for internal and external solutions are pointing into the same Solution Data Base. It is therefore a good idea to index internal and external solutions such that they are consistent and easily identifiable. For example, you could use odd thousands (or ten thousands) for external solutions and even thousands (or ten thousands) for internal solutions.

Solution Data Base



The solution data base can be edited using the Pulse drop-down menu entry Solution Base. A subset of this dialog appears if one is requested to enter a solution into the Pulse data structure.

Solution Index: Index number for the solution.

Name: Name of the solution.

Numeric Name: An editable field that may hold a feature of the solution that is not easily determined from its ingredients (e.g., free calcium concentration, etc.).

pH: Holds the value of the pH of the solution and the substance used to adjust it.

Osmol. : Holds the value of osmolarity of the solution.

Index / Ingredient / Conc.: Each ingredient is defined by its index number and its concentration in the solution.

Entries: Number of entries in the solution data base.

Create / Duplicate / Delete Entry: Used to generate, copy, and remove solutions in the data base.

Next / Last Entry: This option moves through the data base by selecting the next or last available solution.

Export Label: This is used to output labels of the shown solution:

- **ASCII File**: Exports the solution information as ASCII text.
- **Igor Text**: Exports the solution information as IGOR text.
- **Printer**: Prints the solution information. Two labels are created, a bigger one which can be used to label a bottle, and a smaller one which fits on a syringe.

Export Listing: This is used to output a list of the entire solution file:

- **ASCII File**: Exports the solution file as ASCII text.
- **Printer**: Prints the solution file.

Save: Saves the file to disk.

Undo: Reverts the edited solution to its original form.

Done: Exits the dialog.

Note: A user can generate the complete missing solution tree as follows. In the Configuration window he sets the "Sol. Source" to "Data Base" and "Sol. Timing" to anything but "Off". Then he has to open the data file in "modify" mode. Finally, he can clicks on the root in the Replay window, and select the option "Tree -> Solution".

Capacitance Measurements

Preparations

1. Move the file DefaultPulse.set out of the PULSE folder. This will make sure, that the starting conditions are always reproducible! You will need PULSE 8.11 or higher.

2. Attach the model circuit to the headstage (10 MΩ position), turn the EPC9 on and start PULSE+PULSEFIT.

3. Click the SET-UP button of the EPC9 window (for a Voltage Offset correction), switch to the medium position, click ON-CELL (for a C-Fast compensation), switch to the 0.5 GΩ position.

Using the automatic C-Slow Compensation

Automatic C-Slow Compensation

C-Slow and *G-Series* can be quickly estimated by the built in automatic compensation procedure.

Note: The C-Slow compensation works by repetitively applying small rectangular pulse and fitting an exponential to the current response. The settings for this routine are located in the EPC9 menu (amplitude: ***CSlow Peak Ampl.***, # of pulses per estimation: ***CSlow Cycles***, break after ***CSlow Timeout*** seconds, faster algorithm: ***Quick CSlow***).

To perform an automatic compensation, click the **Auto CSlow** button.

CapTrack Mode

CapTrack mode: runs *Auto C-Slow* repetitively.

1. Click the **Delay** button in the *EPC9* window. Type “0.001”, this will run the *Cap-Track* as fast as possible. Note, you must type a value > 0!

2. Activate **EPC9 → Log Tracking**. This is necessary to view the *CapTrack* results in the *Notebook* window. Activate **Notebook → Buffered Output**, necessary to keep the results in the *Notebook*.

3. Click the **CapTrack** button in the *EPC9* window. To stop tracking click it again (Tip: if you have a slow computer better 'SHIFT' + Click!). You should see something like the following in the *Notebook* window (*Time, C-Slow, G-Series*):

```
00:08:41.803    21.672pF   163.9nS
00:08:41.913    21.706pF   163.7nS
00:08:42.009    21.706pF   164.4nS
00:08:42.105    21.706pF   164.4nS
```

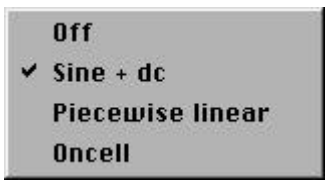
Using the LockIn Extension

Activate the LockIn Extension

1. To activate the LockIn extension select Pulse → Activate LockIn. PULSE will tell you to save the settings and quit (Save and quit). Do so!

2. Restart PULSE. The menu item Pulse → Activate LockIn now changed into LockIn Configuration.

3. LockIn is not activated yet. To do this, select Pulse → LockIn Configuration and set a valid LockIn Mode. With the EPC9 set Sine+DC.



Note: "Piecewise linear" is an older, simplified method. It only calculates **relative** changes of Cm and Gs or Gm. "Sinewave + DC" calculates **absolute** values of **all three** LockIn parameters, it is therefore the preferred method! Disadvantage: the software has to know all hardware settings which is the case with PULSE and the EPC9. See the separate LockIn manual for a comprehensive description of the LockIn modes.

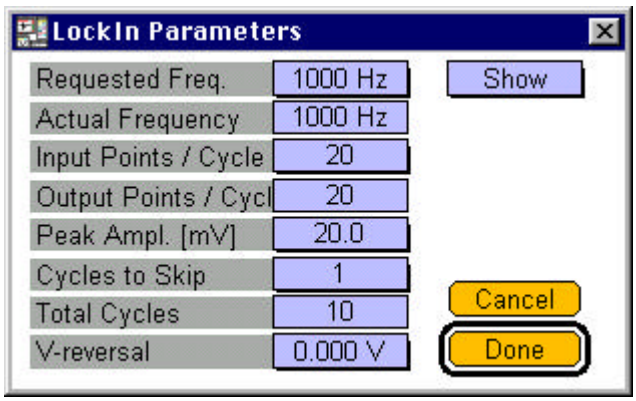
Run the LockIn Extension: Output to the Notebook

1. Activate Write to Notebook. This will print out the LockIn results.

2. Leave Phase Shift at 0.0° and Calibration Mode at Calculated. Close the dialog.

Note: Sine+DC method calculates 4 values: real part, imaginary part, DC value and phase shift. It needs two calibration parameters: phase shift and attenuation of the amplifier. "Hardware" calibration: more accurate, however has to be repeated every time if any setting (filter, gain, frequency, ...) changes. "Calculated" calibration: easy to handle, accurate enough → preferred!

3. Open the *Pulse Generator* and create a new sequence *"SineDemo"* as follows: one segment, **Segment Class** = *Sinewave*, **Voltage** = *-40 mV*, **Duration** = *10 ms*, **No of sweeps** = *5*, **Macros: Start** = *SetLockIn*. Note: you must set a holding potential ≠ 0 mV to calculate *G-Membrane*.



4. Click the button **LockIn Parameters** to set up the *LockIn* parameters and select these settings: **Req. Frequency** = *1000 Hz*, **Input Points / Cycle** = *20*, **Peak Ampl.** = *20 mV*, **Cycles to Skip** = *1*.

Notes - Important: Filter 2 should be at least twice as high as the LockIn frequency, the number of points should be at least 8. The onset of the sinewave produces an artifact, therefore the first cycles should be skipped.

5. Switch the *Amplifier Gain* to 2 mV / pA, set *Filter 1* to 10 and *Filter 2* to 3 kHz, *Stim. Filter* = 20 µs. Run the new sequence from the *Oscilloscope*. You should see the following output in the *Notebook*:

```
Execute: SineDemo

        Calculated LockIn Calibration:
 Est. Int. Phase: 291.9°   Est. Att.: 0.958

 A:   63.23nS   B:   83.34nS   b:   1.581nS   Phase:  52.8°
 RS:  5.685MOhm   RM:  626.8MOhm   CM: 20.896pF
   #   V(1)[mV]   t[ms]  Cap.[F]        time[s]    15:59:57.101
   1    -40.0     10.0   20.896p      0.000
```

6. *PULSE* calculated a calibration before running the sequence. The next lines are the results of the *LockIn* calculation.

Note: **One mean value** is calculated over the whole sweep (i.e. from 10 - 1 estimations, one estimation per cycle). The next lines are from the *Online Analysis* which has been set up to display **LockIn CM** vs. **Time** by the macro *SetLockIn*.

7. Make a *C-Slow* compensation and run the same sequence again:

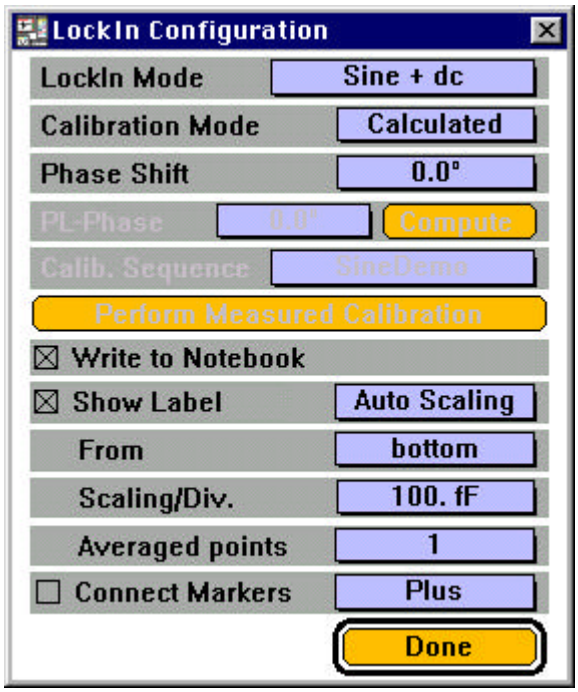
```
Execute: SineDemo
 A:   64.95nS   B:   84.53nS   b:   1.552nS   Phase:  52.5°
 RS:  5.629MOhm   RM:  638.5MOhm   CM: 21.392pF
   #   V(1)[mV]   t[ms]  Cap.[F]        time[s]    16:03:19.312
   1    -40.0     10.0   21.392p      0.000
```

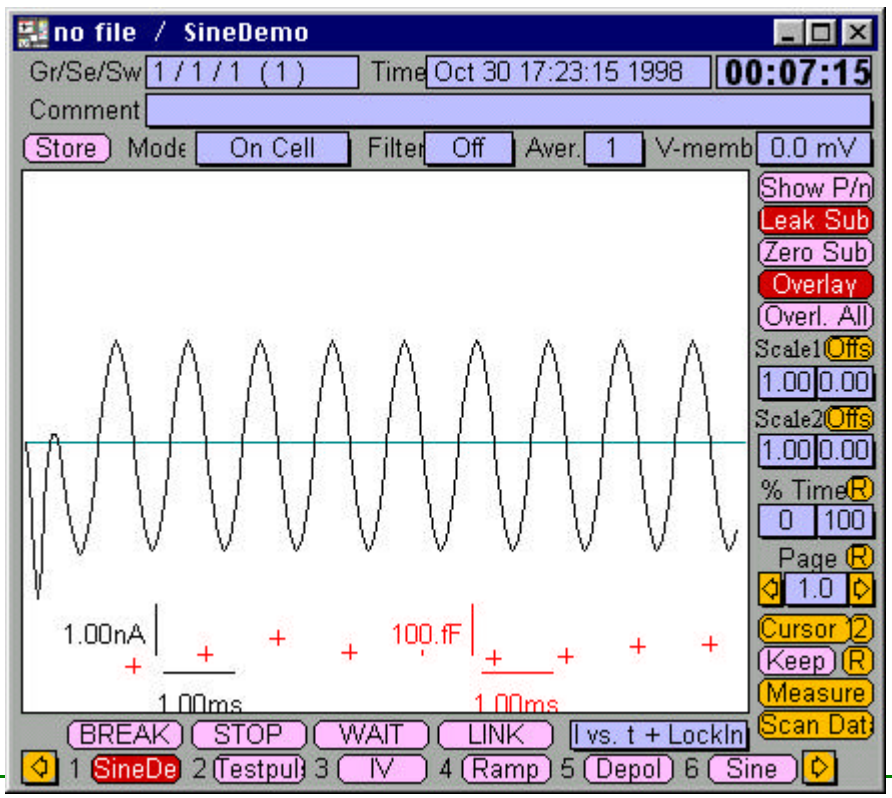
Note: The LockIn extension needs C-Slow uncompensated to work. However, with the EPC9, PULSE can calculate back raw currents from the compensated ones. The LockIn result will be almost identical in both cases!

Output to the Oscilloscope

To view the results of the *LockIn* extension in the *Oscilloscope* window in high time resolution proceed as follows:

1. Open the *LockIn Configuration* again (**Pulse → LockIn Configuration**).
2. Set the following display parameters: **Show Label** = *Auto Scaling*, **From** = *bottom*, **Scaling/Div.** = *100f*
3. Activate the menu option **Display → Display Mode → I vs t + LockIn**. This option is also available in the *Oscilloscope* window, alternatively. Run *"SineDemo"*, you should see the following:

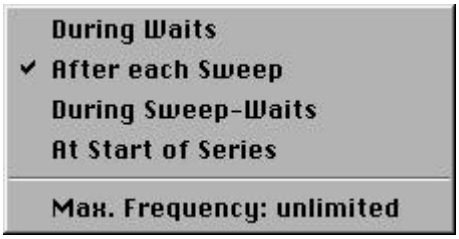




Sending LockIn Results to X-CHART

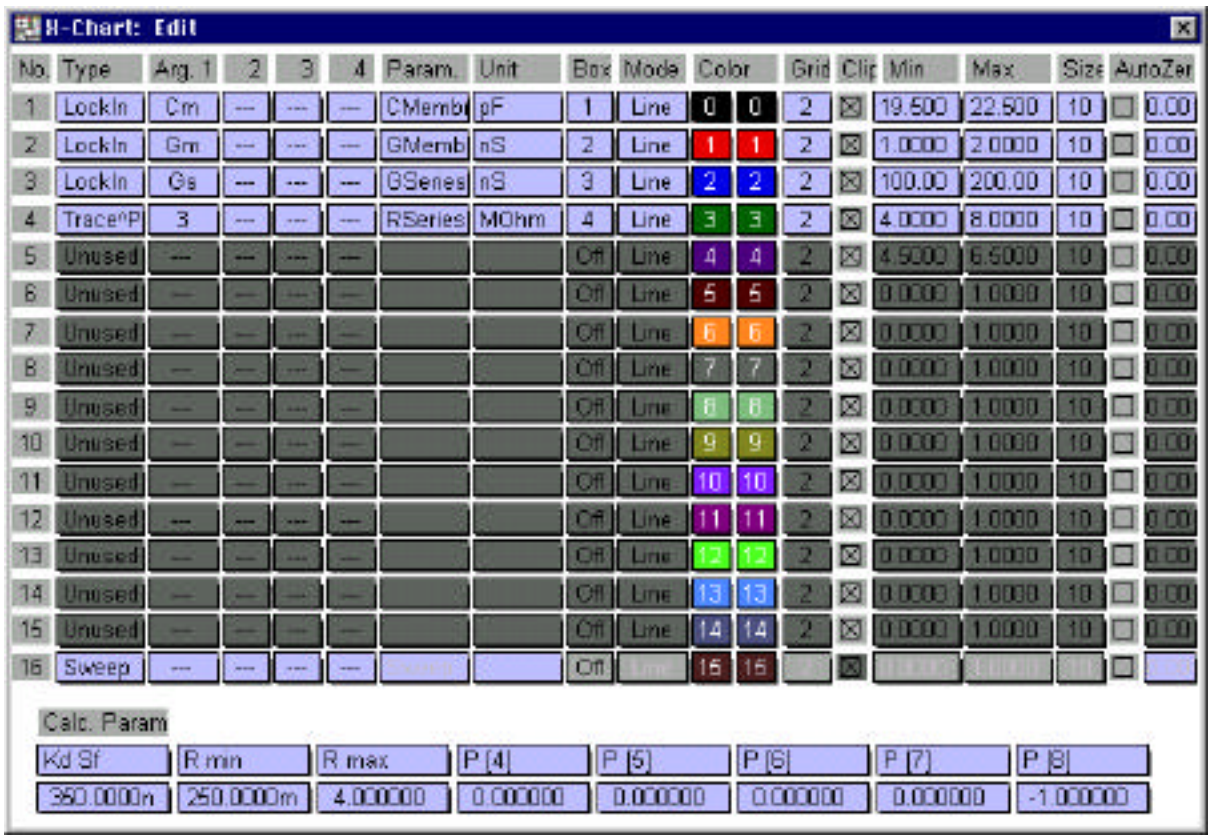
Setting up the X-CHART Extension

1. To activate *X-CHART* select **Pulse → Turn X-Chart On**. A new menu **X-Chart** appears to the right. To acquire *LockIn* results enable the option **X-Chart → Active → After each Sweep**. The other **Active** settings should be off.



2. Open the *X-CHART Control* window by selecting **X-Chart → Control**.

3. Open the *Edit* window by selecting **X-Chart → Edit Traces and Param.**
4. Set the options according to the following figure then close the window:

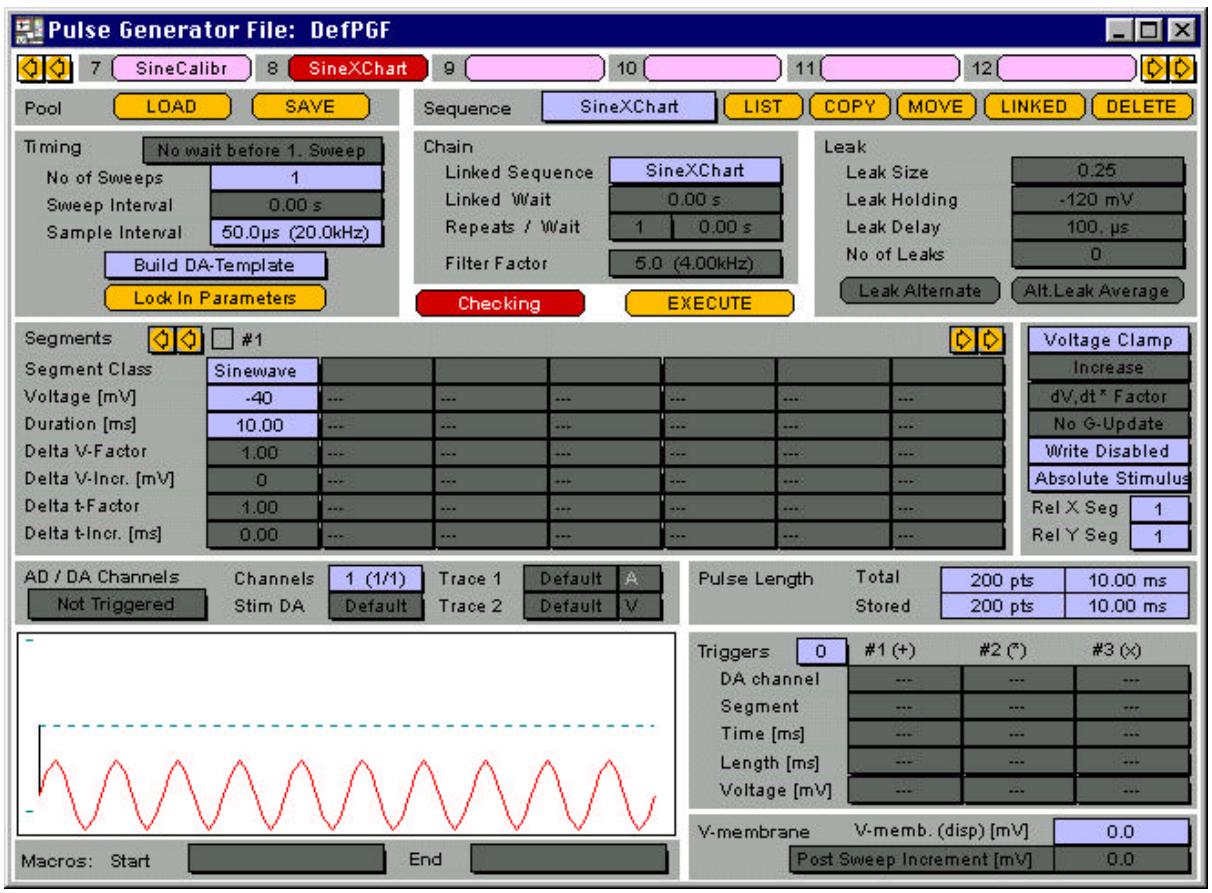


Note: If ***Type*** = LockIn, then ***Arg 1*** can be Real, Imag., DC-Value, Cm, Gm or Gs. The type ***Trace^P[8]*** allows to calculate the reciprocal of any trace (specified by ***Arg.1***), if ***P[8]*** = -1. We use this to calculate the Series Resistance. The ***Unit*** can be preceded by a scientific prefix (f, p, n, u, m, k, M, G). The other options we set in this dialog serve for display purposes only.

Creating a LockIn Pulse Generator Sequence

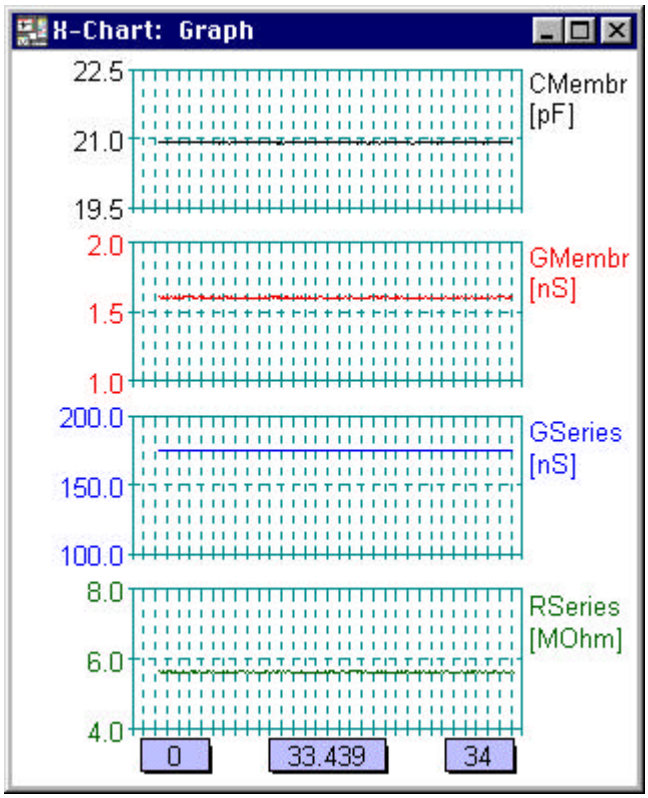
Before we can run *X-CHART* we will have to modify the pulse sequence.

1. Open the *Pulse Generator* again. Select the sequence *SineDemo*, click **COPY**, label the new entry *SineXChart*. This will duplicate the entry for modification.
2. Make the new sequence **Write Disabled** and link it to itself (**Linked Sequence** = *SineXChart*). This will later only store the *LockIn Results* and not the raw data, which saves space.



Sampling the LockIn Data

1. Open the *X-CHART Graph* windows (**X-Chart → Graph**).
2. Start acquisition in *X-CHART* by clicking the **Run** button in the *Control* window. The button will change its text to **Stop**. Note: you may reset the *X-CHART* data first by clicking the **Reset** button.
3. Execute the *LockIn* sequence *SineXChart*. This will run the sequence infinitively. To break it type 'CTRL' + 'B'. Watch the *X-CHART Graph* window!
4. After stopping the sequence, click also **Stop** in the *Control* window.



Keys

Most controls within windows can be accessed from the keyboard. Menu functions are called by a combination of 'CMD' (MacOS) or 'ALT' (Windows) plus a further key (e.g., 'CMD' / 'ALT' + 'Q' for **File → Quit**); they are always accessible. Some global functions are called by a combination of 'CTRL' and a key (e.g., 'CTRL' + 'B' for **Break**). Some single keys are also implemented in *PULSE* (e.g., 'E' for **Execute**). In addition to these global keys, each individual window can have further keys assigned to each control:

KEY	Function
SPACE	Toggle between *Amplifier* and *Oscilloscope* window
1, 2, 3, ... 9	Execute sequence with this index number
W	Write *Pipette Resistance* (Amplifier window)
BACKSPACE	Wipe screen
ESC	Switch to *Oscilloscope*, closing the front window
CURSOR + LEFT / RIGHT	Change *V-membrane* by 10 mV
SHIFT + CURSOR LEFT / RIGHT	Change *V-membrane* by 1 mV
T	Reset the timer clock
CTRL + Z	Apply *Zap* pulse in Amplifier window only)
CURSOR UP / DOWN	Change EPC9 amplifier gain
HELP	Show *Help* window
OPT + HELP	Show *Key* assignment
CTRL + CURSOR	Move in *Tree* (from any active window)
CTRL + RETURN	Display *Tree* target (from any active window)
CTRL + B	Interrupt sequence execution, i.e. "Break"
CTRL + E	New experiment number
CTRL + F	Show "Files" dialog
CTRL + I	Pause sequence execution, i.e. "Wait"

CTRL + L	Stop and continue with linked sequence
CTRL + N	Create new group
CTRL + Q	Resume paused sequence execution
CTRL + R	Show remaining buffer space
CTRL + S	Stop sequence or replay execution
CTRL + T	Edit Root text
CTRL + W	Toggle *Store* ("Write") control

The option **Show Keys** in the **Pulse** menu displays the key assignments of the various windows. Many key commands are active in all windows (e.g., the key assigned to the **Overlay** option will toggle **Overlay** from all windows). During data acquisition, *PULSE* will not accept mouse clicks, except for the **BREAK, STOP, LINK, and WAIT** button. However, *PULSE* will accept the following key strokes during data acquisition:

KEY	Function
'L'	Toggle *Leak* subtraction
'Z'	Toggle *Zero* subtraction
'P'	Show *P/n* trace
'.'	Overlay all
'T'	Reset *Timer*
'BACKSPACE'	Wipe screen
'CTRL' + 'W'	Toggle *Store* before next series
'CTRL' + 'B'	Interrupt sequence execution (*Break*)
'CTRL' + 'I'	Pause sequence execution (*Wait*)
'CTRL' + 'L'	Stop and continue with linked sequence (*Link*)
'CTRL' + 'Q'	Resume paused sequence execution
'CTRL' + 'R'	Show remaining buffer space
'CTRL' + 'S'	Stop after sequence execution (*Stop*)
'CMD' / 'ALT' + 'I'	Show file status info

'OPT' / 'ALT' + '1', '2', '3', …	Mark a sweep in the tree with the corresponding number
'SHIFT' + ANY KEY	Executes the Amplifier control with the corresponding letter
NUM '+' OR '-'	Change display scaling
NUM '*'	Center sweep in display
CURSOR UP OR DOWN	Change *Gain* before next series
SHIFT + CURSOR UP OR DOWN	Change *Gain* before next sweep
CURSOR LEFT OR RIGHT	Change *V-membrane* before next series

One can send key strokes to the Amplifier window during the waiting periods between sweeps, if one holds down the SHIFT key. Yet, be aware that changing the amplifier state within the acquisition of a series may make the stored information inconsistent, e.g., when one changes the holding potential within one series. Even worse, one can call a macro which switches modes or gets into a recursive macro call. This may cause the program to crash!

Data Format

In this chapter we describe the general structure of the files generated by *PULSE*.

Data Files

PULSE generates up to 4 files when you create a data file:

1. The *Data* file (file extension ***.dat**).
2. The *Stimulus Templates* file (file extension ***.pgf**).
3. The *Acquisition Parameters* file (file extension ***.pul**).
4. The *Notebook* file (file extension ***.txt**).
5. The *Solution Data Base* file (file extension ***.sol**).

PULSEFIT creates an additional file:

6. The Analysis file (file extension ".ana").

Except for the raw data file and the Notebook file, all other files have a *Tree* structure. The entire trees are kept in memory, whereas the raw data traces are always loaded from disk, when needed.

Stimulation Template: Stim

Stores the stimulation protocol. The structure of the *Stimulation File* (extension "pgf") is defined by the *Definition Module* "Stim.de" (see *Appendix I*). The *Stimulation File* has a tree structure:

Record	Description
Root	Version number
Stimulation	Description of an ensemble of pulse patterns; e.g., I-V curve
Segment	Individual segment of a pulse pattern

Stimulation files can be loaded into the *Pulse Generator*. In fact, the *Pulse Generator* files for the stimulation protocols used during the experiments have the same data structure as the PGF-files, which belong to the recorded data. In this way it is possi-

ble to exactly repeat an experiment by using a copy of a PGF-file as *Pulse Generator* file.

Data Description: Pulsed

Stores parameters, such as gain, capacitance, etc. The pointer to the data stored in the raw data file is also contained in this file. The structure of the *Pulsed File* (extension "pul") is defined by the *Definition Module* "Pulsed.de" (see *Appendix I*). The *Pulsed File* has a tree structure:

Record	Description
Root	Version number, text, time, file name
Group	Larger section of an experiment; e.g., cell or patch
Series	Description of an ensemble of traces; e.g., I-V curve
Sweep	Description of an individual data trace

A graphical template of the *Pulsed File* (*Tree*) is shown in the *Replay* window. It contains information necessary to reconstruct the experimental conditions as the data were recorded.

Solution Data Base: Solution

The fourth file (optional) that *PULSE* writes contains a data base of the solutions used during the experiment. For each series, two solutions can be specified by values between 0 and 2'147'483'648 (see variable in *Series*: *InternalSolution* and *ExternalSolution*). A more detailed description of these solutions is stored in the *Solution File* (extension "sol"). This file has a tree structure as defined by the *Definition Module* "Solution.de" (see *Appendix I*):

Record	Description
Root	Version number
Sol	Description of a solution
Chemical	Description of each ingredient

One entry in the *Sol record* is *NumericalValue*. It specifies a parameter of the solution that is not easily determined from the ingredients, like the free calcium concentration of the solution, for example. It can be used later for analysis purposes, such as to plot current as function of the calcium concentration, for example. The assignment of numbers to solutions must be unambiguous within one data file. It is recommended,

however, that unambiguous assignments are used within one laboratory to make life easier. To support this, *PULSE* makes use of a default common solution data base which is called "SolutionBase.sol". This data base is a file of the same format as the other "*.sol" files. It is thought to be a generally accessible data base of all solutions used in the laboratory (or at least within one project). During the experiment the user can select solutions directly from this base to store them in the current data file; this saves a lot of typing and reduces the number of errors. More solution data bases can be used; the requested name can be specified and saved in the *Configuration* window (see below).

Analyzed Data: Analysis

Stores the results of data analysis. The structure of the *Analysis File* (extension "ana") is defined by the *Definition Module* "Analysis.de" (see *Appendix I*). The *Analysis File* has a tree structure:

Record	Description
Root	Version number, description of the tree size
Group	Larger section of an experiment; e.g., cell or patch
Series	Analysis of an ensemble of traces; e.g., I-V curve
Sweep	Parameter of an individual data trace; e.g., peak current

Raw Data

This raw data file is a continuous data stream. Each data point is a 16-bit signed integer (exceptions are explicitly mentioned below). When a sweep is stored, *PULSE* stores the various single traces (if available) as follows:

- leak-subtracted current trace
- averaged leak traces
- second trace
- "Fura" data points (stored as short reals)

Inside the program, the data are processed as *Real* numbers. This helps to avoid round-off errors during data averaging and P/n procedures, for example.

Notebook File

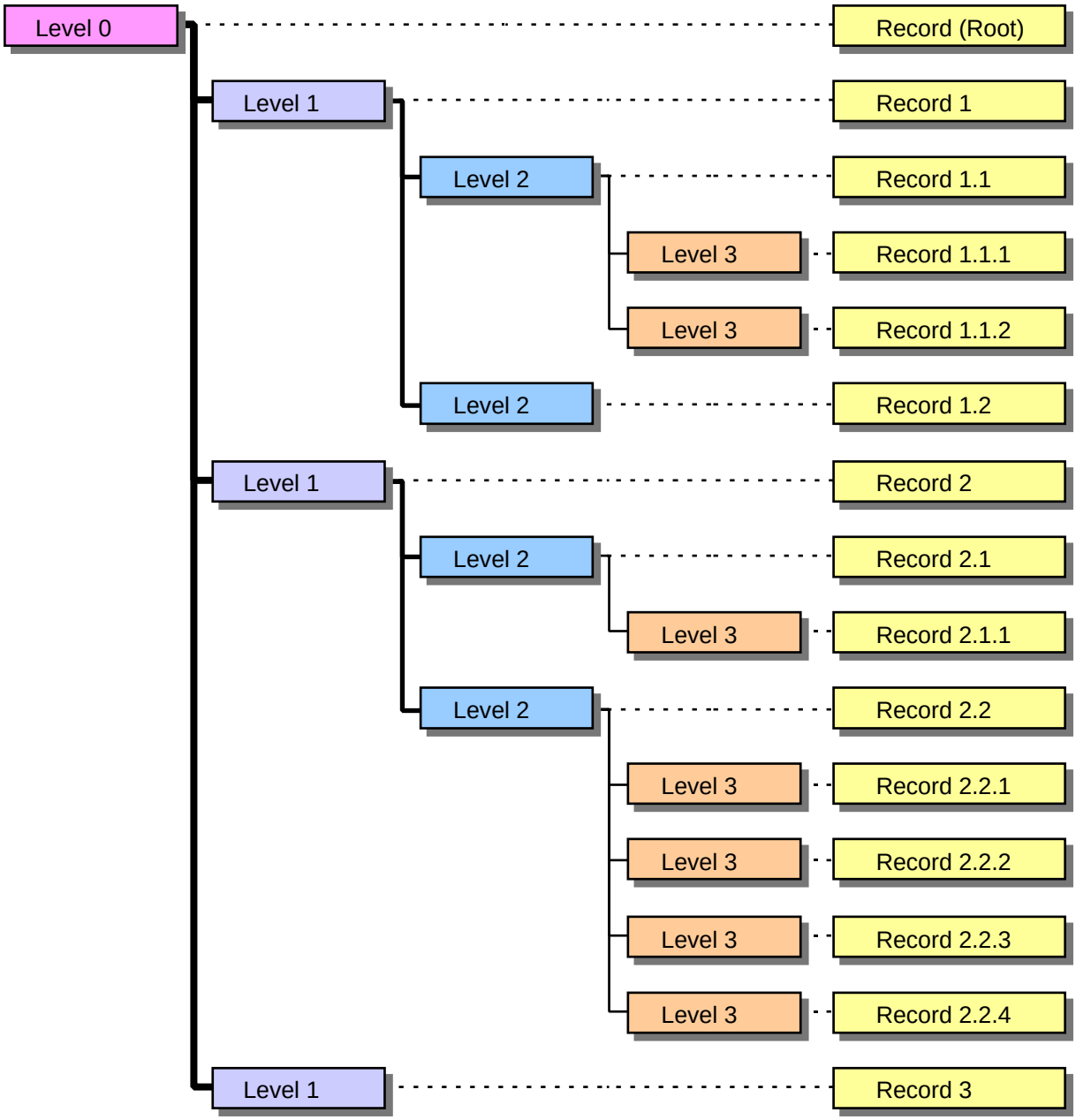
The notebook file is a standard ASCII text file with line breaks.

The Tree Format

The idea of *PULSE* is to order the data of an experiment in *Trees*. The trunk of the tree (*Root*) is the main descriptor of a data file (it could, for example, correspond to one cell or the entire experiments of one day). The next level is the *Group*. This level can be defined by the user to identify data that belong together. An example would be to open a new group for each patch. The group may contain several families of records. Such a family (e.g., records of a current–voltage relationship) is called *Series*. The individual records of a family are called *Sweeps*. Finally, each sweep may be composed of *Traces* (at the moment, there are two traces per sweep). The latter are not yet part of the data tree (it is planned to allow for more traces in the future). A copy of this data tree is accessible to the user throughout the experiment (so one has an overview of what was recorded, and one can immediately edit the entries (e.g., discard bad records).

The following is a description of the *Tree* format. *Appendix III* has a source code listing for a program that shows how to scan and load a *Tree* file. It can be compiled and executed by the *PowerMod* environment. The source code is commented and can easily be translated to other languages.

An example tree can be diagrammed as follows:



There is only one record of level 0, the root record. The above tree has four levels, the root (level 0), levels 1 and 2, and the leaf nodes (level 3).

The format of a tree stored to a file is as follows:

1. Magic number: 054726565H
2. Number of levels

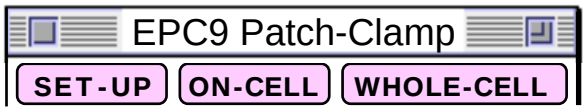
3. Level sizes, one per level
4. Tree records, top down, left-to-right. Each record has the format:
    - Record contents
    - Number of children

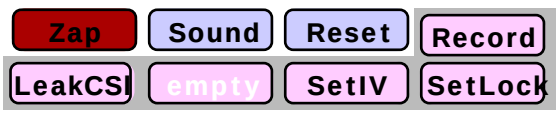
All of the values (except the record contents) are INT32 values, i.e., 32-bit (4 bytes) values.

Note: Check the record sizes in the file headers. The record sizes may differ from what you are expecting, e.g., because the file has been created by an older program version which used fewer fields than it is currently using, or a newer version with additional fields**. You must use the record sizes stored in the files themselves.**

Macros

One of the great advantages of the EPC9 is, that in combination with the *PULSE* software you are able to greatly automate your experiments. Using *Macros* instead of directly interacting with the program allows you to increase this automation to a higher level. *PULSE* can handle up to 20 macros available through the amplifier window and a separate menu. If you used the program you might already have noticed three built in macros: the three buttons **SET-UP**, **ON-CELL** and **WHOLE-CELL** represent three predefined macros that are distributed together with the software. They represent the macros *#1* to *#3* and are accessible through buttons in the top of the *Amplifier* window. Macros *#4* to *#7* are located in the bottom of the *EPC9* window (or the middle of the *Amplifier* window for the other amplifiers). The default installation for the EPC9 predefines positions *#6* and *#7* by **SetIV** and **SetLockIn** which set up the *Online Analysis* window for IV-curves and LockIn experiments. Macros *#8* to *#20* are hidden in the right part of the *Amplifier* window. To make them visible you will have to expand the amplifier window to the right.





Macros Menu

All macros are organized within the sub-menu **Macros** in the **Amplifier** menu (**EPC9**, **EPC8**, **EPC7**, **Axon-200.** or **Amplifier**, depending on your active amplifier).

Load... : Merges a macro file (extension ***.mac**) to the current macro pool. This will only redefine those macros that are available in the file. If a macro position already used is not defined in the macro file, it will not be cleared.

Save...: Writes the current macro pool to a file.

List: Lists the current macro pool to the *Notebook* window.

Start Recording: Starts (or stops) recording all key strokes and actions the user performs in the *Ampli-*

fier, *Oscilloscope*, *Online Analysis* or *Parameters* window. During recording a macro the name of the menu item changes into **Stop Recording**.

Stop Executing: If a macro is running, this will stop its execution.

Execute while recording macro: If this option is not enabled, the user commands will only be recorded but not executed while recording a macro. This is useful if you e.g. want to create a macro that changes amplifier settings but without affecting them while you are recording them.

Execute 1 ... 20: If the corresponding macro has been defined, the menu item will be enabled and entitled by the macro name (e.g. **Execute "SET-UP "**). Selecting the menu item will execute the associated macro. This is analogous to clicking the corresponding macro button in the *Amplifier* window.

Creating Macros

One can easily generate simple macros in *PULSE+PULSEFIT* through the button **Record** (EPC9) or **Macro Record** (other amplifiers) in the *Amplifier* window and via the *Amplifier* drop-down menu, **Macros → Start Recording**. If you start recording a macro, the **Record** button and the menu bar start blinking to notify you about this fact. If you want to finish macro recording click the **Record** button again to cancel the recording or click on any macro button to assign the recorded actions to that button. Alternatively you can select **Macros → Stop Recording** from the *Amplifier* menu. In this case you will be asked for an index (1 to 20) and an unambiguous name for the new macro.

While the above method works fine for simple macros it might be very inconvenient for more complex macros were you easily make a mistake during recording such as clicking the same button twice or in the wrong order or entering a wrong value. In this case you can store the macro file and modify it using any word processor which can generate plain ASCII text files (*Notepad, BBEdit, SimpleText*). E.g. you can use *Microsoft Word* and save the document as **Text Only with Line Breaks**, or you can use the editor of *PULSE+PULSEFIT* (see below).

The Editor of PULSE+PULSEFIT

One can use the built-in editor of *PULSE+PULSEFIT* to create or modify macro files. The following is a step-by-step description how to modify the first macro of the de-

fault macro file **DefaultEPC9.mac** as an example. This macro resets the amplifier, sets a reasonable test pulse, recording mode and amplifier gain and then initiates an automatic offset potential cancellation.

```
MACRO-FILE        820
  1 : SET-UP
  E  Reset
  E  Mode:             1: On Cell
  E  Gain:            10: 5.0 mV/pA
  E  PulseDur:                5.0ms
  E  PulseAmp:                5.0mV
  E  AutoZero
  E  Bell
```

We want to make sure that the test pulse will be activated in every case, even if it had been turned off. This can be done by adding a `PulseOn` command into the macro. In addition, we want to change the EPC9 gain from the default value of 5 mV/pA to 2 mV/pA. This is how the macro should look like (the changed parts are italic):

```
MACRO-FILE        820
  1 : SET-UP
  E  Reset
  E  Mode:             1: On Cell
  E  Gain:            10: 5.0 mV/pA
  E  PulseDur:                5.0ms
  E  PulseAmp:                2.0mV
  E  PulseOn:          TRUE
  E  AutoZero
  E  Bell
```

The first step is to load the macro text from the macro file and display it in the *Notebook* window. This allows us to use the editor of the *Notebook* to modify the macro instructions.

1. First we clear the Notebook by selecting **Clear** from the **Notebook** drop-down menu (MacOS CMD+B, Windows ALT+B). We also have to delete the header of the "blank" *Notebook*.

2. Second, we load the text of the macro file with the **Notebook → Merge...** command.

3. Next, we delete any text which does not belong to the macro text itself, such as the Notebook date, etc. For that, we select any text from the beginning of the *Notebook* down to the beginning of the line starting with `MACRO-FILE`. The text is then deleted by hitting the DELETE or BACKSPACE key.

4. We are now ready to insert the `PulseOn` command. It is best to use an existing text line as a template which can easily be modified. For our example, we duplicate the line containing the `"PulseAmp:"` string. To duplicate that text line, triple click on it. This will select the complete line. Now we perform a **Edit** → **Cut** operation (MacOS CMD+X, Windows ALT+X) followed by two **Edit** → **Paste** commands (MacOS CMD+V, Windows ALT+V).

5. We can now replace the `"PulseAmp"` string by the string `"PulseOn"`. For that we double click on the word `"PulseAmp"`. This will select the entire word. We now can type `"PulseOn"`. Make sure the colon does not get lost. Then we replace the `"5.0mV"` text with the `"TRUE"` string as shown in the listing above.

6. Now we can proceed to modify the default gain. Thus, we change the present gain index from `10` to `9`. Please note that menu indexes start to count at zero! The gain selection `"5mV/pA"` is the eleventh menu item. Therefore, the menu index is 10 (i.e., menu selection minus one). We can now modify also the text string `"5.0mV/pA"` to `"2.0mV/pA"`. This is optional, since any text after the second colon considered as a comment. Yet, it protocols the change we just made.

7. Finally, we save the modified macro file to disk with the **Save As...** command from the **Notebook** drop-down menu. Enter **DefaultEpc9.mac** as the file name, and make sure to have selected the correct directory. Then click on **Save** and acknowledge the **Replace existing...** alert. The new macro file will be saved and can subsequently be loaded into *PULSE+PULSEFIT* with the option **Macros** → **Load** of the **Amplifier** menu.

Parsing Rules for Macros Files

The following is a description of the rules which are used to interpret a macro file.

I. A macro file is a plain text file in ASCII. The first text line is the identifier `MACRO-FILE [version]`. The version number, if used, must start after position [10].

II. When a line has a `digit` in position [2], it contains the macro index plus the macro name. The macro index can start at position zero, but has to end at position [2]. The `macro name` is the text from position [6] to the end of the line.

III. A line is interpreted as a macro item, if it does not have a digit in position [2], and is longer than 6 letters.

   A. The letter at position [2] defines the owning window, e.g.:
   `"o"` and `"O"` → **O**scilloscope window

"e" and "E" → Amplifier window ("**E**PC9")
"a" and "A" → **A**nalysis window

B. Item values start after the first colon up to the second colon or the line end.

1. A Boolean value is considered TRUE, if the value string begins with "t" or "T", or is a digit ("1", "2", ... "9").
2. A list selection has to be the list index.
3. A floating point number can be in fixed, scientific, or engineering format (i.e., with unit such as "p" for pico). Values are scaled with the same factors as within the program.
4. Integer values should be within the expected range.
5. A string can be any text, but cannot contain a colon!

IV. Whatever comes after the second colon is ignored. Thus, comments can be added after the second colon.

Macros Reference

The following gives an overview of all macro commands used within *PULSE*. The description corresponds to the various windows one can control by macros. If you want to build up your own custom macros, you can simply copy the necessary lines into your macro file.

Note: A macro can invoke also other macros, and can call itself as the last command in the macro. A macro which calls itself, or calls another macro which finally calls the original macro again would run indefinitely. Thus, to interrupt such a macro execution, one can use the ***Break*** command ('CTRL' + 'B') to interrupt immediately, and the ***Stop*** command ('CTRL' + 'S') to stop execution at the end of the momentarily running macro (i.e., when the macro would re-start).

Amplifier Window (E)

Amplifier Gain: `Gain 0, 1, ... 19` sets the amplifier gain. The amplifier gain can be increased or decreased stepwise by simulating the up cursor (`CursorUp`) or down cursor (`CursorDown`) key.

```
E  Gain:            0; 0.005 mV/pA
E  Gain:            1; 0.010 mV/pA
E  Gain:            2; 0.020 mV/pA
```

```
E  Gain:             3; 0.050 mV/pA
E  Gain:             4; 0.1 mV/pA
E  Gain:             5; 0.2 mV/pA
E  Gain:             7; 0.5 mV/pA
E  Gain:             8; 1.0 mV/pA
E  Gain:             9; 2.0 mV/pA
E  Gain:            10; 5.0 mV/pA
E  Gain:            11; 10 mV/pA
E  Gain:            12; 20 mV/pA
E  Gain:            14; 50 mV/pA
E  Gain:            15; 100 mV/pA
E  Gain:            16; 200 mV/pA
E  Gain:            17; 500 mV/pA
E  Gain:            18; 1000 mV/pA
E  Gain:            19; 2000 mV/pA

e  CursorUp
e  CursorDown
```

Holding Potential: `VHold` sets the holding potential of the active amplifier. If the button **Relative Value** has been activated (`E Relative: TRUE`) the new holding potential will be changed relatively. The holding potential can be increased or decreased in steps of 10 mV by simulating the right cursor (`CursorRight`) or left cursor (`CursorLeft`) key

```
E  VHold:           10.0mV; sets holding potential to 10 mV
E  VHold:          -50.0mV

E  Relative:        TRUE; the following depolarizes VHold by 10 mV
E  VHold:           10.0mV
E  Relative:        FALSE

e  CursorRight
e  CursorLeft
```

AD-Input Channel: `TestAdc 0, 1, ... 12` sets the default input channel. Arguments `5` to `12` define the AD-channels 1 to 7.

```
E  TestAdc:         0; F2-Ext
E  TestAdc:         1; Vmon
E  TestAdc:         2; Imon1
E  TestAdc:         3; Imon2
E  TestAdc:         5; AD-0
E  TestAdc:         6; AD-1
E  TestAdc:         7; AD-2
E  TestAdc:         8; AD-3
E  TestAdc:         9; AD-4
E  TestAdc:        10; AD-5
E  TestAdc:        11; AD-6
E  TestAdc:        12; AD-7
```

Recording Mode: `Mode 0, 1, ... 5` sets the recording mode (*inside-out*, *on-cell*, ...).

```
E  Mode:            0; In Out
```

```
E  Mode:            0; In Out
E  Mode:            1; On Cell
E  Mode:            2; Outside Out
E  Mode:            3; Whole Cell
E  Mode:            4; C-Clamp
E  Mode:            5; V-Clamp
```

Test Pulse: `PulseDur` sets the duration and `PulseAmp` sets the amplitude of the test pulse. Both values can be changed relatively. `PulseOff, PulseOn` and `NoiseOn` turn the test pulse off/on or activate the noise mode. `PulseMode 0, 1, 2` defines the type of test pulse (single or double pulse or execution of the test series).

```
E  PulseDur:        5.0ms; test pulse 5 ms long
E  PulseAmp:        5.0mV; test pulse 5 mV in voltage clamp modes
E  PulseAmp:       10.0pA; test pulse 10 pA in current clamp mode
E  PulseOff:        TRUE
E  PulseOn:         TRUE
E  NoiseOn:         TRUE
E  PulseMode:       0; double
E  PulseMode:       1; single
E  PulseMode:       2; run the test series
```

Offsets: `Ljunc` sets the liquid junction potential and `Vzero` the pipette offset potential. Both values can be changed relatively. `AutoZero` automatically zeroes the pipette current, while `SearchMode` tracks the pipette offset by repetitively calling the `AutoZero` function.

```
E  Ljunc:           5.0mV;
E  Vzero:           5.0mV
E  AutoZero
E  SearchMode:      TRUE
E  SearchMode:      FALSE
```

C-Fast Compensation: `CFastTot` sets the value, `CFastTau` the time constant and `CFastPerc` the %-value of the fast capacitance compensation. `AutoCFast` performs an automatic C-Fast compensation. All values can be changed relatively.

```
E  CFastTot:        6.00pF
E  CFastTau:        0.5µs
E  CFastPerc:       80 %
E  AutoCFast
```

C-Slow Compensation: `CSlowRange` sets the range of the slow capacitance compensation circuit (off, 30, 100 or 1000 pF). `CSlow` sets the slow capacitance and `RSeries` the series resistance value. `AutoCSlow` performs a single and `CapTrack` a repetitive automatic C-Slow compensation. For the later you will have to set a `Delay`.

```
E  CSlowRange:      0; Off
E  CSlowRange:      1; 30 pF
E  CSlowRange:      2; 100 pF
E  CSlowRange:      3; 1000 pF
```

```
E  CSlow:          20.00pF
E  RSeries:         5.0MO

E  AutoCSlow
E  Delay:           0.10s; set frequency of CapTrack and TrackGLeak
E  CapTrack:        TRUE; turn CapTrack on
E  CapTrack:        FALSE; turn CapTrack off again
```

RS Compensation: `RsMode 0 ... 3` sets the range and `RsComp` sets the %-value of the RS compensation. The %-value can also be changed relatively.

```
E  RsMode:          0; Off
E  RsMode:          1; 100 µs
E  RsMode:          2; 10 µs
E  RsMode:          3; 2 µs
E  RsComp:         93%

E  Relative:        TRUE; the following decreases RsComp by 10%
E  RsComp:        -10%
E  Relative:        FALSE
```

Leak Compensation: `GLeak` sets the size of the hardware leak compensation. This value can also be changed relatively. `AutoGLeak` performs a single and `TrackGLeak` a repetitive automatic leak compensation. For the later you will have to set a `Delay`.

```
E  GLeak:           1.00nS

E  AutoGLeak
E  Delay:           0.10s
E  TrackGLeak:      TRUE
E  TrackGLeak:      FALSE
```

Filter: `Filter1 0 ... 3` sets the type of *Filter 1* (Bessel 100, 30 and 10 kHz, HQ 30 kHz). `Filter2Response` sets the response of *Filter 2* (Bessel or Butterworth). The bandwidth of Filter 2 can be set by `Filter2`.

```
E  Filter1:         0; Bessel 100 kHz
E  Filter1:         1; Bessel 30 kHz
E  Filter1:         2; Bessel 10 kHz
E  Filter1:         3; HQ 30 kHz
E  F2Response:      0; Bessel
E  F2Response:      1; Butterworth
E  Filter2:        10.0kHz
```

Stimulus: `StimFilter` sets the *Stimulus Filter* (2 or 20 µs). `ExtScale` sets the scaling of the external *Stimulus Input*. `TstDacToStim1/2/3` redirects the stimulus to the output of amplifier 1, 2 or 3 (EPC9/2 and EPC9/3 only).

```
E  StimFilter:      0; 2 µs
E  StimFilter:      1; 20 µs
E  ExtScale:        1.000x
E  TstDacToStim1:   0; DA-3 to Stim-1: OFF
E  TstDacToStim1:   1; DA-3 to Stim-1: ON
```

```
E  TstDacToStim1:   2; Ext. Stim. Input: ON
E  TstDacToStim1:   4(    1); AutoCSlow -> DA-3
E  TstDacToStim2:   0; DA-3 to Stim-2: OFF
E  TstDacToStim3:   0; DA-3 to Stim-3: OFF
```

Current-Clamp mode: `CCSpeed` toggles the *Fast Current Clamp* mode of the EPC9 board version 'C'. `CCRange` sets the scaling of the output in the *Current-Clamp* mode. This value is fix for every EPC9. `GentleCCSwitch` toggles the *Gentle CC-Switching*.

```
E  CCSpeed:          TRUE; activate fast current clamp mode
E  CCSpeed:          FALSE; deactivate fast current clamp mode
E  CCRange:          0; CC-Range: 1pA/mV
E  CCRange:          1; CC-Range: 10pA/mV
E  GentleCCSwitch:   0; Gentle CC-Switch: OFF
E  GentleCCSwitch:   1; Gentle CC-Switch: ON
```

DA-Channels: With these commands you can define and set the analog and digital outputs. `DAChannel` defines the output channel to be used (`0..3` = analog channels, `5` = digital output), `DAValue` defines the output value either as voltage (analog channels) or as unsigned byte (digital trigger lines). The digital lines are interpreted binary: 1 corresponds to the first trigger, 2 to the second, 4 to the third, 8 to the fourth, and so on. To set multiple triggers you have to add the binary values, e.g. 7 (= 1 + 2 + 4) activates the first, second and third trigger. To output the last value that was defined use `DASet`.

```
E  DAChannel:        0; DA-0
E  DAChannel:        1; DA-1
E  DAChannel:        2; DA-2
E  DAChannel:        3; DA-3
E  DAValue:          5.00V

E  DAChannel:        5; Digital out (word)
E  DAValue:          16; set trigger 5 high
E  DASet
E  DAValue:          27; set triggers 1, 2, 4 and 5 high
E  DASet
```

Calling a macro from a macro: `Macro1...Macro20` calls Macro #1 to #20 from a running macro. Note: there is no separator between `Macro` and `Index`. Take care to not create an endless macro by calling the same macro recursively!

```
E  Macro1
E  Macro19
```

Sound: The commands `SoundOn`, `SoundHz` and `SoundVol` activate the sound and define its frequency and volume.

```
E  SoundHz:        100
E  SoundVol:       100%
E  SoundOn:        FALSE
```

Various Commands: `LastVHold` restores the last holding potential before switching into *Current-Clamp* mode. `Relative` defines the next value to be a *relative* change instead of an *absolute* setting. This *relative* input feature is automatically deactivated after one value is entered. `Wait` interrupts the current macro execution and asks the user to continue. `Bell` gives an acoustic signal, e.g. after a macro has been executed. `Zap` applies a *Zap* pulse. `Reset` resets the active amplifier. `E9Board1/2/3` sets the active amplifier.

```
E  LastVHold
E  Relative:       TRUE
E  Wait
E  Bell
E  Zap
E  Reset
E  E9Board1:       TRUE
E  E9Board2:       TRUE
E  E9Board3:       TRUE
```

Oscilloscope Window (O)

Sweep Settings: `WriteData` toggles the **Store** button. `Mode` defines the *Recording* mode. `Filter` sets the *Oscilloscope* filter, `MemPot` sets the *Oscilloscope* holding potential. `Averages` defines the number of averages acquired for one sweep.

```
O  Comment:        This is a Comment
O  WriteData:      TRUE
O  Mode:           0; In Out
O  Mode:           1; On Cell
O  Mode:           2; Outside Out
O  Mode:           3; Whole Cell
O  Mode:           4; C-Clamp
O  Mode:           5; V-Clamp
O  Filter:         500. Hz
O  Filter:         2.00kHz
O  Averages:       4
O  MemPot:         -50.0mV
```

Display Flags: `ShowPn` turns the display of P/n data on or off. `LeakSubtract` turns the display of leak subtracted data on or off. `ZeroSubtract` turns the display of zero subtracted data on or off. `Superimpose` turns the overlay mode on or off. `StoreAll` in-/activates the **Overl. All** button.

```
O  ShowPn:         TRUE
O  LeakSubtract:   TRUE
O  ZeroSubtract:   TRUE
O  Superimpose:    TRUE
O  StoreAll:       TRUE
```

Display Scaling: `DisplayGain1/2` defines the scaling and `DisplayOffset1/2` the offset of display 1 or 2. To reset the display (gain = 1, offset = 0) use `DisplayZero1/2`. The scaling of the oscilloscope can be fixed to an absolute range using `FixedScaling`. The ranges are set with `XRange`, `Y1Range` and `Y2Range`.

```
O  DisplayGain1:   4.00
O  DisplayOffset1: 200.m
O  DisplayZero1
O  DisplayGain2:   1.00
O  DisplayOffset2: -500.m
O  DisplayZero2

O  FixedScaling:   TRUE
O  XRange:         1.00 s
O  Y1Range:        500.p
O  Y2Range:        1.00n
```

Display Mode: `Xmode` sets the x-axis of the *Oscilloscope* (t, V, V-ramp, ...). `InvertIV` is used to plot *V* versus *I*. With `ConnectSweeps` different sweeps in a series are connected.

```
O  Xmode:          0; I vs. t
O  Xmode:          2; I vs. V
O  Xmode:          3; I vs. V-ramp
O  Xmode:          4; I vs. V-ramp,theo
O  Xmode:          6; I vs. t + LockIn
O  InvertIV:       TRUE
O  ConnectSweeps:  TRUE
```

Display Timing: `StartTime` and `EndTime` define the left and right margin of the display (in % of the full scale). To reset the timing (start = 0%, end = 100%) use `TimeReset`. To reset the whole *Oscilloscope* window (scaling and timing) use `DisplayReset`. `PageLeft` and `PageRight` scroll one page to the left or right. To scroll to any page use `Page`.

```
O  StartTime:      10
O  EndTime:        90
O  TimeReset
O  DisplayReset

O  PageLeft
O  PageRight
O  Page:           5.0
```

Cursors: `ShowCursor` toggles display of the cursors, `ResetCursor` resets the cursors of the actual analysis range (i.e. sets them to 0 and 100%).

```
O  ShowCursor:     TRUE
O  ResetCursor
```

Controlling the Experiment: `Pool1...Pool6` starts the pulse sequence 1 ... 6 from the actual PGF pool. To change the pool (i.e. shift it by 6 to the left or right) use `LeftSeq` or `RightSeq`. `Break`, `Stop`, `Wait` and `Link` activate the corresponding button. `Timer` resets the *PULSE* timer. `User1...User16` sends corresponding the user defined serial string to the serial port.

```
O  Pool1:          TRUE
O  Pool6:          TRUE
O  LeftSeq
O  RightSeq
O  Break:          FALSE
O  Stop:           FALSE
O  Wait:           FALSE
O  Link:           TRUE
O  Timer
O  NewExperiment
O  NewGroup
O  User5:          TRUE
O  User12:         TRUE
```

Online Analysis Window (A)

Analysis Type: `Range` sets the actual analysis (1 or 2). `Abscissa` (0...8) sets the x- and `Mode` (0...20) sets the y-value.

```
A  Range:           0; Range 1
A  Range:           1; Range 2
A  Abscissa:        0; Voltage
A  Abscissa:        1; Duration
A  Abscissa:        2; Time
A  Abscissa:        3; Timer Time
A  Abscissa:        4; Realtime
A  Abscissa:        5; Index
A  Abscissa:        6; Peak Voltage
A  Abscissa:        8; Y1 vs. Y2
A  Mode:            0; No Analysis
A  Mode:            1; Extremum
A  Mode:            2; Maximum
A  Mode:            3; Minimum
A  Mode:            4; Time to Peak
A  Mode:            5; Mean
A  Mode:            6; Charge
A  Mode:            7; Variance
A  Mode:            8; Slope
A  Mode:            9; Reversal
A  Mode:           10; C-Slow
A  Mode:           11; G-Series
A  Mode:           12; Anodic Charge
A  Mode:           13; Cathodic Charge
A  Mode:           14; LockIn CM
A  Mode:           15; LockIn GM
A  Mode:           16; LockIn GS
A  Mode:           17; Fura Ratio
```

```
A  Mode:            18; Fura Ca
A  Mode:            19; Fura F1
A  Mode:            20; Fura F2

A  Fit:             TRUE
A  Reference:       FALSE
```

Analysis Settings: The online analysis will be calculated over the range defined by the two cursors (`LeftB`, `RightB`) for the segment defined as the *relevant* one in the *Pulse Generator*. Change this segment by adding an offset (`RelXSeg`, `RelYSeg`). `Trace` defines which trace is used for the calculation (first or second trace), `Math` enables you to do further calculations with the values obtained by the two ranges.

```
A  LeftB:           10.0%
A  RightB:          90.0%

A  RelXSeg:          1
A  RelYSeg:          1

A  Trace:            0; First Trace
A  Trace:            1; Second Trace

A  Math:             0; no math
A  Math:             1; y = y1 + y2
A  Math:             2; y = y1 - y2
A  Math:             3; y = y1 * y2
A  Math:             4; y = y1 / y2
```

Display Settings: `MarkerKind` sets the symbol type (0...5) and `MarkerSize` (1...12) sets the size of the makers in the *Online Analysis* plot. `Scale` sets the scaling to *automatic* or *fixed*. In the later case you can define the range of the graph with `Xmin`, `Xmax`, `Ymin` and `Ymax`. `X/YZeroLine` defines whether to display the x-/y-axis and `X/YZeroTics` the number of ticks. Each axis can be linear, logarithmic or exponential (`X/YTransform`). Use `CopyLast` to copy the last used settings or `CopyOther` to copy the range settings of the other analysis into the actual ones. `PlotLast` plots the last *Online Analysis*. `Overlay` sets the *Overlay* mode.

```
A  MarkerKind:       1; Plus
A  MarkerKind:       2; Star
A  MarkerKind:       3; Diamond
A  MarkerKind:       4; Cross
A  MarkerKind:       5; Square
A  MarkerSize:       2

A  Scale:            0; Fixed Scaling
A  Scale:            1; Auto Scaling

A  Xmin:            -50.000m
A  Xmax:             70.000m
A  Ymin:             0.000
A  Ymax:             80.000p
```

```
A  XZeroLine:       TRUE
A  XZeroTics:       5
A  YZeroLine:       TRUE
A  YZeroTics:       11

A  XTransform:      0; lin
A  XTransform:      1; log
A  XTransform:      2; exp
A  YTransform:      0; lin
A  YTransform:      1; log
A  YTransform:      2; exp

A  CopyLast
A  CopyOther
A  PlotLast

A  Overlay:         0; No Overlay
A  Overlay:         1; "Time" Wrap
A  Overlay:         2; Overlay + "T"-Wrap
```

Parameters Window (P)

Parameters: The commands `Gain`, `VGain`, `AuxGain`, `CSlow`, `GSeries`, `RsValue`, `Bandwidth`, `CellPotential`, `PipPressure`, `Temperature`, `LockinExtPhase`, `UserParam1`, `UserParam2`, `PipResistance`, `SealResistance`, and `RMSNoise` allow you to set any of the parameters stored together with the sweep.

```
P  Gain:            2.000mV/pA
P  VGain:           10.00 V/V
P  AuxGain:         1.000 V/V
P  CSlow:           22.00pF
P  GSeries:         5.000MOhm
P  RsValue:         5.000MOhm
P  Bandwidth:       3.000kHz
P  CellPotential:  -50.00mV
P  PipPressure:     5.000 cm
P  Temperature:     20.00 C
P  LockinExtPhase:  0.000 °
P  UserParam1:      2.500 V
P  UserParam2:     -1.300 V
P  PipResistance:   5.000MOhm
P  SealResistance:  5.000GOhm
P  RMSNoise:        110.0fA
```

Solution Timing: `IntSol` sets the internal and `ExtSol` sets the external solution used.

```
P  IntSol:          1
P  ExtSol:          3
P  IntSolEdit
P  ExtSolEdit
```

Key Translations

The following macros simulate key strokes during the execution of a macro:

```
Char: [character]  separated by space
DeleteLeft, DeleteRight
END, Enter
F1, F2, ... F15, Help, HOME
Numeric *, Numeric +, Numeric -, Numeric ., Numeric /, Numeric =, Nu-
meric Clear
Numeric 0, Numeric 1, ... Numeric 9
PageDown, PageUp
```

Input of a Relative Increment

It is possible to change values by a given increment instead of having to specify the absolute value. Thus, one would have to click first on the button **Relative Value** in the *Amplifier* window and then enter the new value in the target control. The macro recorder will calculate the difference between the new and the old setting and will store only this increment. E.g. if you change the holding potential from -60 to -70 mV while recording a macro, a step of *-10 mV* will be recorded. The *"relative"* input feature is automatically deactivated after one value is entered.

PULSE+PULSEFIT Advanced

Continuous Recording

PULSE treats continuous recording just as pulsed recording with a pulse template having a very long segment. These segments are called *Continuous* segments in the *Pulse Generator* allowing the user to execute pulsed recording followed by continuous recording in the same run (this is often also referred to as open end stimulation). Most of the functions for purely pulsed data are available. In other words, a *Continuous* segment can be incremented in duration and voltage like the other segments, repeats and links of sequences are allowed.

The restrictions are that the continuous segment has to be the last segment of a template and that no P/n pulses are supported for it or the rest of the template. Also, no averages are supported for continuous segments.

The timing is done like for the other segments in ms. If a quasi infinite recording is desired (note that a hard disk is filled up very quickly), this duration is to be set to an accordingly large value. The user can interrupt a continuous recording at any time during execution with 'CTRL' + 'B' or the *Break* control. In this case the data acquired up to this point are saved, if at least "one page" of data were acquired. The *Stop* key ('CTRL' + 'S') is used to stop the acquisition at the end of the running sweep. Other keyboard functions that are active during continuous data acquisition are (remember: holding the Option key pressed will apply the functions to the second trace!):

Key	Function
Numeric '+'	Increase display gain for trace 1
Numeric '-'	Decrease display gain for trace 1
Shift + Numeric '+'	Shift the trace up for trace 1
Shift + Numeric '-'	Shift the trace down for trace 1
Numeric '*'	Center the 1. trace in the display
'O'	Toggle *Overlay* mode
'.'	Toggle *Overlay All* mode

‘BACKSPACE	Clear display
‘T’	Reset the Timer clock

Some key strokes will be buffer until the end of the ongoing acquisition of a series. The corresponding functions will be applied before the next series is started. There keys are:

Key	Function
Cursor Right	Increases Holding by 10mV or 10pA
Cursor Left	Decrease Holding by 10mV or 10pA
Cursor Up	Increase the amplifier gain by one step
Cursor Down	Decrease the amplifier gain by one step
CTRL + ‘W’	Toggle writing next series

These functions allow to easily switch to a new continuous record, when e.g. the series with the continuous acquisition is linked to itself. Thus, one can start acquiring till one decides to change one parameters such as storing, the amplifier gain, or the holding potential. Then one presses the appropriate key, and clicks on the “LINK’ button (or hits Ctrl+ ‘L’).

For replay the initial part of the template (without conditioning and continuous segments) is shown as one display or page (0-100%) by default. Thus, by creating an initial constant segment of a certain length one can specify the default display time resolution. However, the minimum time basis per page is 200 ms. The rest of the trace can be reviewed by paging through the data.

The output functions take either the section of the data as shown on the screen or the entire sweep.

Continuous (“gap-free”) Data Acquisition

PULSE can acquire very long sweeps. There are two completely different approaches. Before describing them there is a one advice to be recalled: in most situations where a repetitive stimulus is applied - be it a voltage or a chemical stimulus - it is very important to know how the data are to be analyzed. E.g. an current-voltage experiment could be acquired by applying all voltage steps within one sweep. Yet, it will be very difficult to get the online analysis to correctly and **easily** analyze that

train of pulses. In that example it is clear that it would be much preferable to acquire one voltage pulse per sweep. Then the online analysis gets obvious while it would be very tedious to analyze the data, when all voltage pulses were within one sweep. Please, keep that in mind when designing your sweep structure.

Using the "Continuous" Segment Type

The segment type **Continuous** signalizes to *PULSE* that acquisition of long data stretches are asked for. The maximal segment duration of such a "continuous" segment can be as high as 10^7 samples, which is more than 2.5 hours at 100 kHz sampling rate. Please keep in mind, that only the **last segment** can be "continuous" and that there has to be at least one "non-continuous" segment before that, since the length of the sweep in the oscilloscope window is determined by the length of all segments without the last one.

Segments	#1	#2	#3	#4	
Segment Class	Constant	Constant	Constant	Continuous	-
Voltage [mV]	V-membr.	-50.	V-membr.	V-membr.	---
Duration [ms]	250.00	500.00	250.00	19000.00	---
Delta V-Factor	1.00	1.00	1.00	1.00	---
Delta V-Incr. [mV]	0.	20.	0.	0.	---
Delta t-Factor	1.00	1.00	1.00	1.00	---
Delta t-Incr. [ms]	0.00	0.00	0.00	0.00	---

Note: The complete sweep before the "continuous" segment has to fit into the allocated stimulus buffer, which is 16384 points (= 16 kSamples) by default (see below, how to surpass this limitation). The size of all "non-continuous" segments is the ***Total Pulse Length*** in the Pulse Generator window. The ***Stored Pulse Length*** is the physical size of the whole sweep as written to disk, i.e. the size off all segments including the last "continuos" one.

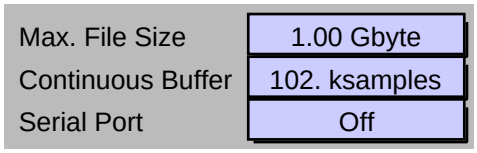
Pulse Length			
	Total	4000 pts	1.00 ms
	Stored	80000 pts	20.00 ms

The "continuous" segment can either be stored directly to disk (*"hard disk recording"*), or the data can temporarily be stored in memory in the allocated *"Continuous Buffer"*:

- Storing directly to disk is possible when one acquires from one AD-channel only. To store directly to disk, enter zero for the size of the **Continuous Buffer** in the *Configuration* window. Keep in mind that storing to disk requires your computer system and hard disk to be fast enough. A contemporary Pentium II system with a SCSI hard disk can record data directly to disk at a rate of up to 100 kHz. If you notice repetitive messages about *"AD-*

overrun" you should decrease the sampling rate or you will have to store the data in memory (see below).

- To store "continuous" data in memory, one has to reserve an appropriate amount of memory for the **Continuous Buffer** in the *Configuration* window.



The total required space of the "continuous" data (on disk and in RAM) can easily be computed with the following formula:

$$Bytes_{total} = \frac{2\ Bytes}{Sample} \cdot Input\ Channels \cdot Frequency \cdot Duration$$

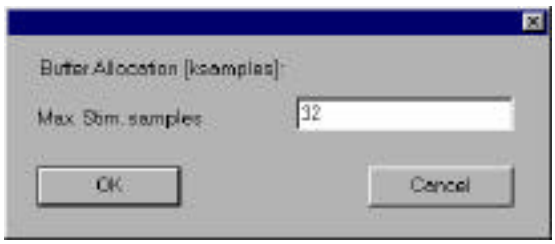
For example, sampling 2 input channels at 10 kHz will produce 2.3 Mbytes of data every minute or 137 Mbytes every hour, respectively (1Mbyte = 1024^2 bytes).

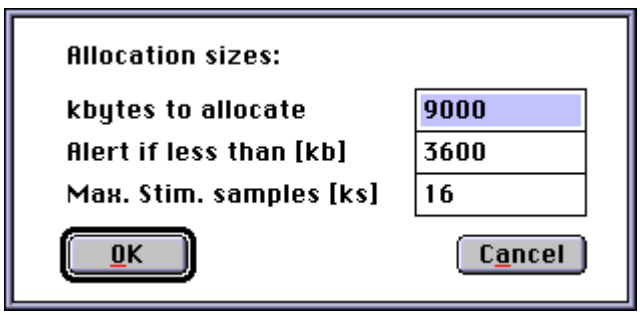
Under MacOS, you can enable **Virtual Memory** in the **Memory** control panel, when the physical memory of your computer system is too limited. However, the required acquisition rate should not be too fast and the hard disk speed should not be too slow. Only practical tests can determine, whether your system will satisfactorily work with *"virtual"* memory enabled.

Under Windows, virtual memory is always active and cannot be turned off.

Stimulating with more than 16 kSamples in one Sweep

While the "continuous" segment type allows you to acquire sweeps longer than the default stimulus length, it does not allow you to stimulate with complex voltage patterns during the "continuous" part of the stimulus. If you are limited by the default maximal sweep length of 16 kSamples and you need to acquire longer sweeps, you can increase the size of the stimulus buffer in the **Buffer Allocation...** dialog. You can call this dialog from the **Pulse** drop-down menu and enter the new value into the field **Max. Stim. samples [ks]**. Please be warned that increasing the stimulus buffer size will consume plenty of computer memory, approximately 48 bytes per additional stimulus sample.

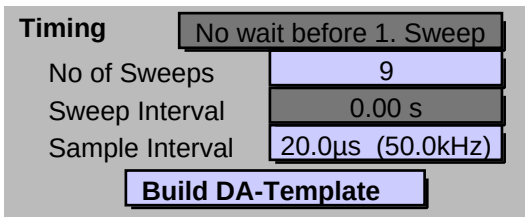




Under MacOS (see picture) you will also have to increase the values given in the **kbytes to allocate** and **Alert if less than [kb]** fields accordingly. Those changes are drastically. If it should happen that there is not enough memory left, start *PULSE+PULSEFIT* again from the Finder with the selected memory allocation. Then, perform the following:

1. Remove the file **DefaultPulse.set** from the **Pulse** folder.
2. Remove the files **Pulse.Settings** and **PulseFit.Settings** from the **Preferences** folder inside the **System Folder**.
3. Increase and / or enable **Virtual Memory** in the **Memory** control panel.
4. Restart *PULSE+PULSEFIT*.
5. Enter an new allocation size which is appropriate to you computer configuration.

Minimizing the Sweep Interval



To make the gap between two sweeps as short as possible set the **Sweep Interval** of the stimulus template to zero - this tells *PULSE+PULSEFIT* never to wait before starting the next Sweep - and turn off as many options as the experiment allows. If possible, acquire only one input channel. Deactivate any background programs, such as networking, screen savers, virus scanner and so on. Close all windows that are not essential, such as the *Notebook*, the *Pulse Generator*, the *Amplifier* window, the *Parameter* window, the *Replay* window and the *Online Analysis* window. Check for the following options:

- Both online analysis ranges should be set to **No Analysis**.
- Deactivate **Notebook → Buffered Output**.
- MacOS computers: reduce the number of colors to **256**. On some Power Macs the lowest setting is **Thousands of Colors**; use that setting.
- Keep the *Oscilloscope* window small.
- Turn **Display → Dimmed Overlay** off . Dimming is performed by redrawing with gray.
- Turn the **Overlay** mode in the *Oscilloscope* window off, since clearing the screen in-between sweeps uses time!

- MacOS: Set **File → Disk Write Options** to **Write after Series**.
- Windows: Turn **File → Disable Data File Caching** off. Note: usually Windows NT is faster than Windows 95 which is faster than Windows 3.1.
- If possible, use a large fast SCSI hard disk to store data. If you are using Windows NT consider using the NTSF format which is faster than the FAT system.
- Use the smallest reasonable screen resolution. The program needs more time to draw on a higher resolution screen for the same physical size of the oscilloscope.

Here are the results of comparing the achieved minimal sweep intervals of three typical computers: The PPC8500 and the Pentium used the PCI-bus cards, while the Quadra used the NUBUS card. The MacOS computers had a SCSI hard drive, while the Pentium computer had an EIDE hard drive.

The "default" test uses the default *PULSE+PULSEFIT* configuration as set by the installation procedure. The "minimal" test uses the above described optimizations and the "Write NoShow" test shows the times when data is not stored using the option **Write NoShow** in the *Pulse Generator* window. We measured the time used to acquire the series **IV** from the **DefPGF.pgf** pool, subtracted the sweep time and divided that value by the number of gaps in the series. The following table shows the minimum sweep gap for these three tests:

Computer	default	minimal	WriteNoShow
Mac Quadra 650 33 MHz	201	173	84
Mac PPC 8500 120 MHz	130	93	34
Pentium (NT) 90 MHz	191	111	70

These minimal processing times depend of course on the speed of the complete computer system, such as the CPU, the hard disk, the graphics card and the operating system.

The Sweep Gap on a Pentium II Computers

Using a Pentium II computer 300 MHz (Intel 82443LX/EX, 64 MB, 60 ns DIMM RAM, 4 GB UW-SCSI hard drive, Winner 1000/T2D S3 Trio64V2/DX 2 Mb) running at a resolution of 1024x768 high color under Windows 95B, the minimum duration

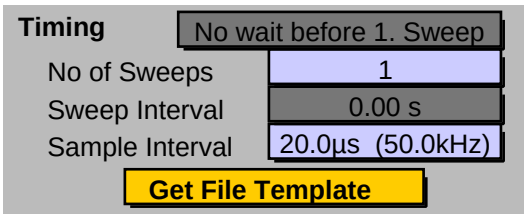
between two sweeps could be as low as 30 ms. With all extensions (X-CHART and LockIn) and the *Online Analysis* enabled the mean sweep gap was about 90 ms.

The following is a benchmark that gives you a better feeling what effects the sweep gap. For every test we ran the series **IV curve** from the **DefPGF.pgf** pool 5 times. The sequence was slightly modified in a way that the first trigger was set to 0 ms and 13 instead of 9 sweeps were used. This protocol produces a total of 60 (= 12 x 5) gaps per test. The complete results are as follows:

OS	configuration	Gap [ms]
Windows 95	no show, caching	29 ± 13
	+ show	28 ± 15
	+ no caching	31 ± 21
	+ show everything, 2x analysis	47 ± 21
	+ LockIn and X-Chart extension	91 ± 15
Windows NT 4	no show, caching	33 ± 9
	show everything, no caching, analysis	44 ± 10
PULSE 8.11, Win 95	no show, caching	295 ± 42

The last line is the same test as the first one, however, *PULSE* v8.11 instead of v8.31 was used. The later version has been “hand optimized” with respect to the Intel code, making the gap 270 ms (or ten times) shorter. As can be seen from the table, the only thing that really affects the minimum duration are the *Online Analysis* and the extensions. Therefore, if you need maximal speed, you should turn off the *Online Analysis* first.

The "Get File Template" Feature



The DAC-stimulus template of *PULSE* can be either computed by the program or loaded from a file instead, when you activate **Get File Template** in the **Timing** section of the *Pulse Generator* window. This way you can stimulate any complex pulse pattern that *PULSE* otherwise can not offer. You can even stimulate a prerecorded voltage trace such as an action potential. There are the following things to consider, when using the *File Template* feature:

1. The template file must be in the folder where the “pgf”-files are. Alternatively, you can put the files into a sub-folder inside the folder where the “pgf”-files are. In this case, the folder name must be the name of the stimulus.
2. The name of the template file must be “[stimulus name]_[sweep number]”. E.g., if the stimulus name is “IV”, then *PULSE* looks for the template file “IV_1” to be the first sweep, “IV_2” for the second sweep, etc.
3. The file must contain one voltage value per stimulus point. The voltage value must be a “short” (4 byte), binary IEEE-floating point format number. All values must be in volt, i.e., if a voltage of -80 mV has to be output, then the required value is -0.080.

The total number of output samples must be equal to the total number of input samples. Thus the stimulus length is defined by:

$$DA-Length = AD-Length \times \frac{AD-Channels}{DA-Channels}$$

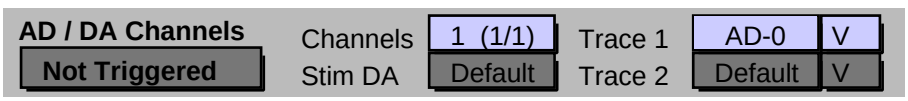
where:

DA-Length	total number of samples per output channel
AD-Length	total number of samples per input channel
DA-Channels	number of physical DA-channels used
AD-Channels	number of physical AD-channels used

The number of physical output channels depends on the used acquisition option, and is computed as follows:

$$Stimulus + Number\ of\ physical\ Triggers + FURA-Channels$$

You can direct more than one *logical* trigger to the same DA-channel. In this case the triggers are output to one *physical* DA-channel. If a trigger is set to **off**, then it does not use a physical DA-channel at all. To help comprehend the acquisition conditions of *PULSE*, the number of physical AD- and DA-channels are indicated in the *Pulse Generator* window, field **AD / DA-Channels**, option **Channels** between brackets.



The first value is the number of physical *input* channels, the second is the number of physical *output* channel. The total number of samples per input channel is displayed in the fields **Pulse Length**, option **Total**.

It is advisable to begin using but one output DA-channel and only one input AD-channel, and not to use any triggers until one is familiar with this option. The following describes the conditions to be considered when building a file template using multiple input and output channels, and how to implement the triggers.

Here is an example that demonstrate how the *File Template* works to stimulate a pre-recorded pulse pattern. You can easily test this using the model circuit:

1. Generate a simple stimulus named "Get" with one sweep per series (**No of Sweeps** = 1) and three **Constant** segments (**Duration**: 20, 10, and 50 ms, **Voltage**: -80, 0 and -80 mV). There should be no trigger (**Triggers** = 0), the **Sample Interval** should be 0.1 ms and one input channel has to be acquired (**Channels** = 1). The sequence should not be linked (**Linked Sequence** = NIL) or use leak pulses (**No of Leaks** = 0). **Stim DA** and input **Channels** are set to **Default**.
2. In the oscilloscope window, execute the "Get" stimulus. You should see the current response with its corresponding "pulse" shape. Let's assume that the response is 50 pA in amplitude. Note: the **Store** button must be on in the oscilloscope, otherwise the sweep would not be stored.
3. To generate a file template, clear the "sweep buffer" first (**Buffer → Clear**) and then add the acquired sweep into it (**Buffer → Add**).
4. The sweep data has been acquired pA and now has to be scaled to mV. This is done by calling **Buffer → Scale** and entering "1e9" as the **Scale factor**.
5. Store the sweep buffer to disk as "Get_1" (**Buffer → Save as binary File...**).
6. Now, go back to the *Pulse Generator* window, select the "Get" stimulus and activate **Get File Template**.
7. Finally, execute the "Get" stimulus again in the *Oscilloscope* window. The file template is read and used as the template, and you should see the corresponding current response.

Alternatively, you can use third party programs such as *IGOR Pro* (see next section p. 189, *Creating a Stimulus Template with IGOR Pro*) to generate the stimulus template file.

Using IGOR Pro together with PULSE

PULSE tightly cooperates with *IGOR Pro* from *WaveMetrics*. You can export sweeps and results from the *Online Analysis* to *IGOR Pro* for further analysis and you can import *IGOR Binary Files* into the sweep buffer.

Creating a Stimulus Template with IGOR Pro

One can import a template generated or modified by *IGOR Pro*. Similarly to the procedure described in the preceding paragraph, one uses the **Buffer → Add Igor Binary** option to import an **IGOR binary wave** into the "sweep buffer". One creates an appropriate "IGOR binary wave" with the IGOR command **Data → Make Waves**. In the *Make Wave* dialog one should not select the **Double Precision** check-box. Double precision generates 8 byte long variables which correspond to the 'C'-type *"double"*, while single precision generates 4 byte long variables which correspond to the 'C'-type *"float"*. The later format is the one used by *PULSE+PULSEFIT*. One can export the template generated in the example of the preceding paragraph with the command **Buffer → Save as ASCII** and import the template in *IGOR Pro* using the command **File → Open File** as a text wave.

The total number of output samples is computed by the formula given in the preceding section "The Get File Template Feature".

The following is an *IGOR* macro that allows you to save any wave within *IGOR Pro* as a file suitable to be loaded by the *Pulse Generator* as a stimulus template:

```
// ************************************************
// Save an IGOR wave as PULSE stimulus template
// ************************************************

Macro SaveStimulusTemplate (waveName, platform)
  String waveName
  Variable platform
  Prompt waveName, "Select Wave to save", popup, WaveList ("*", ";", "")
  Prompt platform, "Select Target", popup, "Native;MacOS;Windows"

Silent 1

  if (! waveExists ($waveName))
    Abort "Please, select an existing wave!"
  endif

  DoSaveStimulusTemplate ($waveName, platform)
End Macro

// Coded as function to improve speed
```

```
Function DoSaveStimulusTemplate (waveName, platform)
  Wave waveName
  Variable platform
  Variable fRefNum, V_Flag, V_filePos, V_logEOF
  String S_fileName, S_path, S_info
  Variable byteOrder, num, inx, value

  if (platform == 1)
    byteOrder = 0           // native format
  else if (platform == 2)
    byteOrder = 2           // BigEndian MacOS
  else
    byteOrder = 3           // LittleEndian Windows
  endif

  inx = 0
  num = numPnts (waveName)
  Open fRefNum as NameOfWave (waveName)
  FStatus fRefNum           // check if file has been opened
  if (V_Flag)
    do
      value = waveName [inx]
      FBinWrite /B=(byteOrder)/F=4 fRefNum, value
      inx += 1
    while (inx < num)
    Close fRefNum
  endif
End Function
```

Using Stimulus File Templates with the LockIn Extension to PULSE

You can even use the *Get File Template* feature together with the *LockIn* extension to *PULSE*. For example, you can create a stimulus that contains prerecorded data such as an action potential plus sinewave segments to calculate the membrane capacitance immediately before and after the action potential. *PULSE* will be able to calculate the LockIn results from these sweeps, if the sinewave segments match exactly their definition in the corresponding pulse protocol. You can easily create such a file template by using *PULSE* together with *IGOR Pro* and the *DataAccess* or *PPT* extension to *IGOR Pro* the following way:

1. In *PULSE* generate a pulse protocol with one segment long enough for the action potential and containing the necessary sinewave segments.
2. Run this protocol once in the *Oscilloscope* window to generate one experiment. It is sufficient to store only one sweep.
3. In *IGOR Pro* load the *PULSE* data file using *DataAccess* or the *PPT* XOP extension **LoadPulse** with the option **Calculate Stimulus** activated (**/I**). This will generate a wave that contains the stimulus template.

4. Now you can insert the action potential into that stimulus wave overwriting the corresponding points from the original wave.
5. Use the macro shown above (or **Macros → Pulse Utilities → Save Wave as Stimulus Template...**, when you have the *PPT* installed) to export the template into a file.
6. In *PULSE* you can now use this modified template instead of a calculated one (**Get File Template** instead of **Build DA-Template**). *PULSE* will calculate the LockIn results correctly, when you run this sequence. However, you should not modify this pulse protocol in the *Pulse Generator* anymore, otherwise the settings won't match the file.

Please take care to use the same number of input and output channels, otherwise this trick won't work.

Loading an IGOR Macro File generated by PULSE

When you export data from *PULSE* to *IGOR* Pro, a so called "recreation macro" is created. The data itself may be stored into a sub-directory or may be loaded directly from the data file, while the macro in the main directory has the instructions how to import these data and display them in *IGOR Pro*o. It is easiest to load such a macro file by simply opening it from the MacOS Finder or the Windows Explorer by double-clicking on it. *IGOR Pro* will start-up and automatically load and execute the macro. On a Windows computer the macro file is recognized by its file name extension ***.itx**.

To load the "recreation macro" within *IGOR Pro* itself, select the menu option **Data → Load Wave → Load Igor Text**. This will bring-up a file selector, allowing you to select the desired macro file to be loaded and thereby executed. Sometimes it can happen that the folder with the data files is no longer accessible, or its name has changed. This can happen, e.g., when one accesses the macro file via a network or when the data are transferred by a floppy disk. When the macro file is executed in such a case, *IGOR Pro* will put an alert on the screen telling you that the volume or folder has not been found and asking you, whether you want to help. In this case *IGOR Pro* will bring up a file selector allowing you to select the target folder in which the data files are located. Once the correct folder has been selected, *IGOR Pro* will proceed executing the macro file.

Note: All macro files generated by PULSE are text files and must be loaded by ***Load Igor Text***, not ***Load Igor Binary***, even when the export format in PULSE was set to ***IGOR Binary***. The macro file does never contain the binary data. If required, the binary data are separately stored in IGOR Pro wave

files, while the macro file contains the instructions how to load and scale the binary data.

Subtracting a linear Leak using IGOR Pro

The following example will show you how to subtract a linear leak from a ramp segment, however, using this scheme you can also subtract anything else. The basic principle is that you will have to create a leak sweep and put it into the buffer. Then you can subtract this buffer by using the option: **Subtract: Buffer** from the **Tree** menu. The best way to create this leak sweep is by using *IGOR Pro*.

- Export your reference sweep as an IGOR wave. Select the options **Tree** → **Export: Igor Text** and **Tree** → **Export Mode: Sweep**. Then activate the reference sweep in the *Replay* window and select **Export** from the **Tree** menu.
- Import the sweep into IGOR. You may do this by double-clicking the exported file (extension ***.go**) from the Finder / Explorer.
- Now you have to create the leak wave in IGOR. Assuming that the imported wave is called *"wave0"*, replace this name in the following *IGOR Pro* commands by the real name of your wave. If you want to directly adopt the following commands, just rename your wave into *"wave0"* before continuing:

```
•Display wave0
•ShowInfo
```

- This created a graph with your sweep and activated the **Info** line at the bottom of the window. Drag cursor A (the circle) to the beginning of the linear section of the sweep and cursor B (the square) to the end of this section, then perform a linear regression over that range:

```
•CurveFit line wave0(xcsr(A),xcsr(B)) /D
```

- Create an output wave by duplicating the input wave and append it to the graph. By duplicating the wave, all parameters of the leak wave will be the same as for the original input wave:

```
•Append wave0_leak
```

- Calculate the leak wave from the fit parameters. This will generate a linear wave over the whole sweep:

```
•startY = fit_wave0[hcsr(A)]-(hcsr(A)-
          leftx(wave0_leak))*W_coef[1]
•wave0_leak = startY + W_coef[1] * x
```

- In order to restrict the linear leak to the ramp, drag cursor A to the start and cursor B to the end of the ramp segment and set everything outside to zero:

```
•wave0_leak[0,pcsr(A)]=0
```

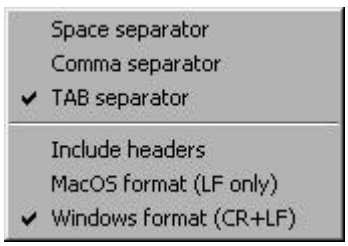
```
•wave0_leak[pcsr(B),]=0
```

- Export the leak wave by using **Save Igor Binary...** from the **Save Waves** command in *IGOR Pro's* **Data** menu.
- In *PULSE+PULSEFIT* load the *IGOR Pro* wave into the buffer by using the **Add IGOR Binary...** command from the **Buffer** menu. If something was in the buffer before, delete it by using **Clear** from the **Buffer** menu.
- In the **Tree** menu select **Subtract: Buffer** in order to subtract the linear leak from the sweeps.

Copying a Table from the Notebook into a Table in IGOR Pro

First, you have configure *PULSE+PULSEFIT* for the special requirements for table data in *IGOR Pro*, as for any program which can import tables. *IGOR Pro*, for example, requires that the values in a table are separated by tabs - other programs may request commas or blank spaces. Also, most programs are not aware of the "engineering" representation of data like "2.345µ". Thus, you must define some settings before copying the notebook and pasting it into *IGOR Pro*:

1. Select the option **Scientific Notation** in the **Notebook** drop-down menu.
2. Select the option **TAB Separator** in the menu **Tree → ASCII-text Format**.



1. Select the appropriate **MacOS Format (LF only)** or **Windows Format (CR+LF)** option in the menu **Tree → ASCII-text Format**.
2. Generate the table in the *Notebook* the usual way in *PULSE+PULSEFIT* by replaying the data and performing the desired analysis.
3. Switch to the *Notebook* window and select the table by dragging the text selection with the mouse.
4. Perform a **Copy** command from the **Edit** drop-down menu.
5. Switch now to *IGOR Pro* and activate the Table window into which you want to paste the data.
6. Finally, issue a **Paste** command from the **Edit** menu of *IGOR Pro*.

Timing Schedule of the Digitizer Board ITC-16

Although modern digitizers such as the ITC-16 which is built into the EPC9 seem to stimulate up to 4 and acquire up to 8 channels simultaneously, in reality this hap-

pens sequentially. In principle, the digitizer reads one AD- and writes one DA-channel at a time. This happens at a minimum of every 5 µs or a multiple of that. First, the AD-channel will be read, then the DA-channel is output with a small delay (1 µs) so that it cannot affect the reading. If several channels have to be read and written, this occurs one after the other. Please note, that the output of a stimulus is always coupled to the acquisition of an input, the ITC-16 can not stimulate a DA-channel without reading an AD-channel.

One example shall clarify the timing scheduling: Let's assume we are sampling 2 channels and stimulating 2 channels. The sampling interval per input channel as defined within *PULSE* is 200 µs (5 kHz). Thus, the digitizer will acquire and stimulate one channel every 100 µs (= sampling interval per input channel divided by the number of input channels). Please note, that this value can not be smaller than 5 µs. In the example the second DA should be set “high” during the time of the first sample (see column “Trigger”).

Time [µs]	Input Channel	Input Sample	Output Channel	Output Sample	Trigger
0	AD-1	1			
1			DA-1	1	
100	AD-2	1			
101			DA-2	1	high
200	AD-1	2			high
201			DA-1	2	high
300	AD-2	2			high
301			DA-2	2	
400	AD-1	3			
401			DA-1	3	
500	AD-2	3			
501			DA-2	3	

As long as the number of input and output channels is the same, things are pretty easy, the minimum duration of a trigger can be as long as the sampling interval, which is 200 µs in the example above: the channel is “high” from 101 to 301 µs. However, this changes, when a the number of read and write differ. The same example as above with 3 instead of 2 output channels gives a different scheme:

Time [µs]	Input Channel	Sample	Output Channel	Sample	Trigger
0	AD-1	1			
1			DA-1	1	
100	AD-2	1			
101			DA-2	1	high
200	AD-1	2			high
201			DA-3	1	high
300	AD-2	2			high
301			DA-1	2	high
400	AD-1	3			high
401			DA-2	2	
500	AD-2	3			
501			DA-3	2	
600	AD-1	4			
601			DA-1	3	

Thus, the minimum trigger length increases from 200 to 300 µs: the second DA-channel is “high” from 101 to 401 µs. One can easily calculate the minimum duration of a trigger by the following formula:

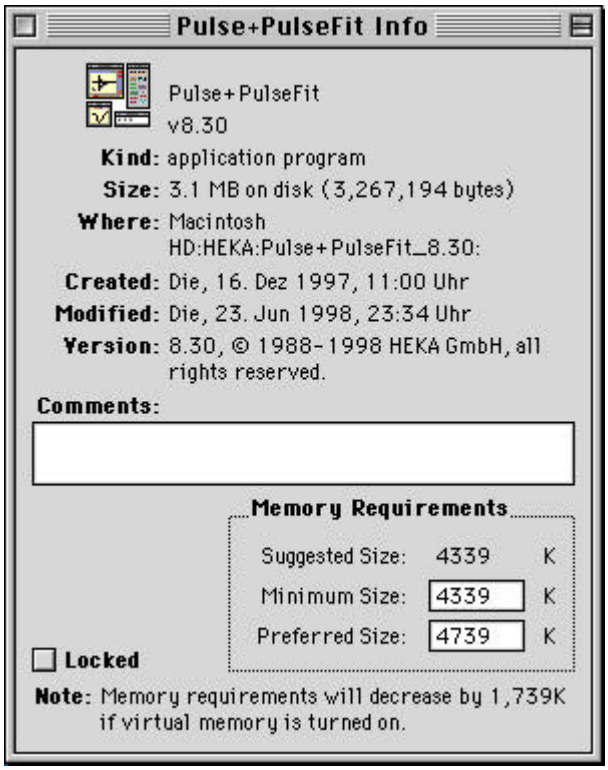
$$MinimalOutputTime = \frac{OutputChannels}{InputChannels} \times SamplingTimeperInputChannel$$

One can get a misleading trigger length, when one uses the AD-channels to measure the duration during which a trigger is high. Although the trigger time is different in the two examples above (200 versus 300 µs) AD-channel 2 reports 200 µs in both cases (low at 100 and 500 µs, high at 300 µs → 500 - 300 = 200 µs). In addition, AD-channel 1 reports 200 (= 400 - 200) µs in the first and 400 (= 600 - 200) µs in the second example (low at 0 and 600, high at 200 and 400 µs).

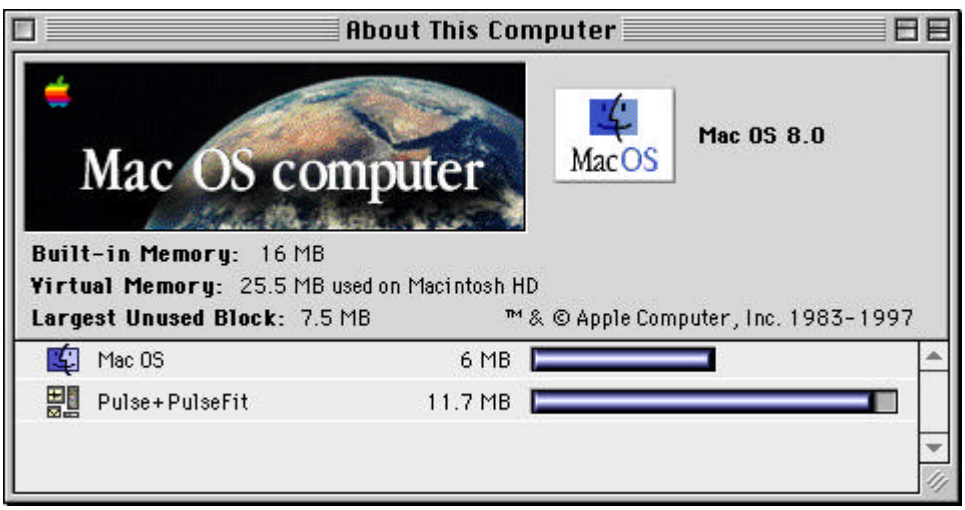
Memory Requirements of the MacOS Version

On a MacOS, *PULSE+PULSEFIT* - like every other application - needs memory for two major tasks: one is the program itself, e.g. the code to be executed, the windows to be drawn, global variables to store the text, position, size and color of every single button, and so on.

The first amount of memory - the so called *"application memory"* - is specified within the finder by the two settings **Minimum Size** and **Preferred Size** in the **Get Info ...** dialog. The preset application memory is usually sufficient for most configurations and does not need to be changed by the user. Only when using a very large screen, more than 256 colors, or when opening many windows simultaneously, there may be a need for *increasing* the application memory. Do not try to *decrease* the **Minimum Size** below the value given as **Suggested Size**.



To inspect whether or not *PULSE+PULSEFIT* needs more memory, simply run the application under the conditions you want and check the memory settings in the Finder by selecting **About this Computer...** in the **Apple** menu: The thermometer bar will indicate the amount of the allocated memory used by *PULSE+PULSEFIT* and you need to increase the application memory only when you are about to exceed that memory.



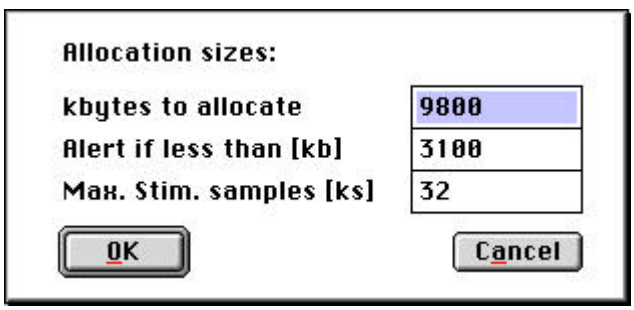
The second task memory is needed for is the data itself, either produced or acquired by the user. This *"data memory"* will be allocated from the system in *addition* to the *application memory* and is specified in the **Pulse → Buffer Allocation** dialog of *PULSE*.

The entry **Kbytes to allocate** in that dialog is analogous to **Preferred size**, i.e. *PULSE* will try to get this amount of memory from the system. The entry **Alert if less than [kb]** is analogous to **Minimum size** in the Finder, however with one difference, the Finder won't run a program, if less than the minimum RAM is available, *PULSE* will only give an alert.

The third entry in the **Buffer Allocation** dialog **Max. Stim. Samples [ks]** specifies, how much of the data memory will be reserved for one single sweep, the so called *"sweep buffer"*. This has to be specified as the number of data points one wishes to acquire maximally per sweep. We have chosen the unit *"kSample"* (and not *"Kbytes"*) because this is what the user is normally interested in, e.g. you want to be able to collect 50,000 data points. Otherwise one would have to calculate the real RAM requirement necessary for the sweep buffer, which is not trivial. Each data point will need approximately 48 bytes of RAM. This sounds huge, given the fact that one floating point value only consumes 4 bytes. However, you have to keep in mind that *PULSE+PULSEFIT* needs also memory for the leak data, the reference sweep, the leak subtracted reference sweep, the LockIn data, the Fura data, the stimulus (plus the triggers) and a scratch buffer for acquisition, filtering, reference subtraction and various other sweep operations.

One example will show you how to set up the amount of data memory: let's assume a typical experiment with 1 Mbyte of raw data (***.dat**), 2 kbytes for the pulse protocols (***.pgf**) and 64 Kbytes for the acquisition parameters (***.pul**). Beside of this, the maximum length of a single sweep should be increased to 32 kSamples. Also, *PULSE+PULSEFIT* needs about 3 Mbyte as a minimum requirement. This makes a total memory requirement of about 1 Mbyte + 2 kbytes + 64 kbytes + 48 x 32 kbytes + 3 Mbytes = 5.5 Mbytes. A good compromise would be the following:

- Buffer Space to allocate = 5.5 MB
- Alert if less than = 3 MB
- Sweep Buffer = 32 kSamples.



These settings make sure, that at least one typical experiment fits into RAM, however *PULSE+PULSEFIT* will try to get three times as much of memory which should be enough for even the longest possible experiment.

Note: Based on the sweep buffer size PULSE+PULSEFIT makes estimations about the minimum and optimum memory requirements. The program does not allow you to enter a lower size into ***Kbytes to allocate*** than it estimates, which is usually a very generous value (9800 kB in the above example).

Comments to the Data Format of *PULSE+PULSEFIT*

The data format is extensively described in chapter *"Data Format"* and the appendices I, II, and III. *Appendix III* contains a simple program depicting how to correctly load an experiment file. Here some additional tips for **programmers**:

Never assume you know how large a record is in a file. You **must** always use the sizes as stored in the file header itself. Otherwise the files may not be readable. E.g., newer files may add fields to a record. This is the way *PULSE+PULSEFIT* had never required a file conversion program, because it could add newly required fields to the end of the respective records.

Remember that many compilers handle **variable alignments** in memory, which differ from the one our compilers use. On a 680x0 MacOS computer, memory alignment is typically on even addresses for fields of 2 or more bytes. PowerPCs run better on 4 byte alignments, and programs for Pentium processors use 4 or 8 byte alignments. Thus, do not use the given record structures, but load the structures in your program byte-wise!

The byte offsets of the record fields are identical for both platforms, although the alignment in memory differs. Thus, one has to de-compress the Tree files upon loading them. The byte offsets and field descriptors are listed in the *PULSE+PULSEFIT* manual, appendix 1, Data Structure. The record fields define all bytes positions by using filler bytes, where necessary.

The **size of variables** and other common definitions are listed, e.g., in Appendix II. Remember: one BYTE is 1 byte; in 'C' one *"char"* can be 1 byte (the old default) or 2 bytes when using Unicode support!

Platform specific issues: there are two byte orders of storing variables in RAM: **big- and little-endians**. "Intel" processors use the "little-endian" format while "Motorola" processors are "big-endians". "Little-endian" means that the last byte (the one with the highest address in RAM) contains the "little" part of a variable. Please, refer to Appendix IV. The following explains how to correctly interpret the *LittleEndianBit* of the sweep structure:

- All sweeps written under MacOS have the *LittleEndianBit* **cleared**. When the MacOS version of *PULSE* reads a sweep with the LittleEndianBit **set**, it swaps the bytes.
- All sweeps written under Windows have the *LittleEndianBit* set. When the Windows version of *PULSE* reads a sweep with the *LittleEndianBit* **cleared**, it swaps the bytes, otherwise no swapping takes place.

Be prepared to byte-swap the **Tree** file as well. The *"Tree"* identifier in the first 4 bytes of a "Tree" file is written as INT32. Thus, when a program reads in the first 4 bytes, and that number is equal to "Tree" (hex 054726565H) no swapping is required. If that "magic number" however is "eerT" (hex 065657254H), then byte-swapping is required. Please note, that the swapping requirement for a **Tree** file (***.tree**) always applies to the complete file, while the raw data file (***.dat**) must be swapped on a per-sweep basis. The reason is that **Tree** files are always written as one "native" file, while the raw data can originate from two different platforms, e.g., acquired on a Windows machine, and modified or analyzed under MacOS.

Note: For C/C++ programmers we offer a package called "DataAccess" which includes libraries that can be linked to your projects to read PULSE files under MacOS as well as Windows. If you need further information, please contact HEKA or your local distributor.

Using the Digidata 1200 Board

The following gives a description how to setup the Digidata 1200 digitizer board to use it with Windows 95.

Limitation of the Digidata 1200 board

1. A *Digidata 1200*, version *A*, *B*, or *C* is supported. The *Digidata 1200* board itself must be factory upgraded before it will run under Windows 95.
2. A minimum of 32 MB physical memory (RAM) is required.
3. Only the first 4 digital I/O lines can be used as synchronous signals, i.e., during the acquisition of a sweep.
4. The second DA-channel (DA-1) can only be used in conjunction with the first DA-channel (DA-0). Thus, one can stimulate on DA-0 alone or on DA-0 plus DA-1.

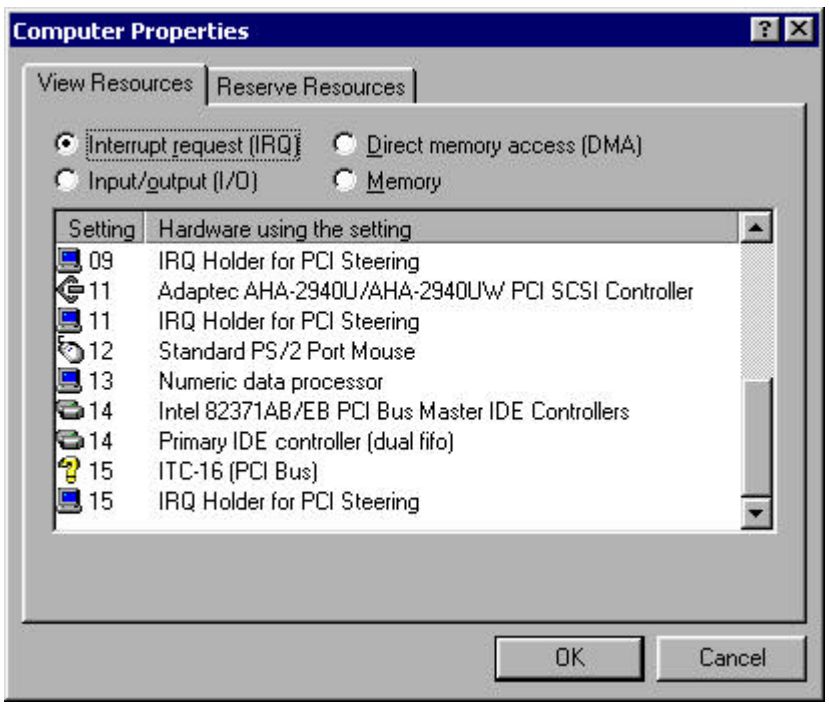
Installing the Digidata 1200 drivers

The *Digidata 1200* board needs drivers to be installed on your machine. This can be accomplished by the *PULSE* setup program. In addition, since the *Digidata* board is an ISA-slot board that does not support plug-and-play, you have to manually configure the computer and the installed *Digidata 1200* driver.

The *Digidata 1200* comes with some pre-set settings which in most cases are not compatible with the default settings of Windows 95. The default "base address" of

the *Digidata 1200* board and the IRQ settings as well usually conflicts with Windows 95 configuration for network cards. It is not advisable to change the Windows 95 default settings. This can lead to surprisingly nasty complications. Thus, it is less time consuming and more predictable to change the settings of the *Digidata 1200* board, instead of changing the settings of the network card or any other conflicting configuration.

Before you plug the *Digidata 1200* board into the computer, you must check whether you have enough free ("non conflicting") resources on your computer. You will need one high IRQ (10, 11, 12 or 15), two high DMA channels (5, 6 or 7) and a free I/O base address (the default is 0x320 to 0x33f). To check the resources of the computer go to **Control Panel → System → Device Manager**, select **Computer** from the list of shown devices and click **Properties**. This will give you a list of all used IRQ's, I/O addresses and DMA channels.



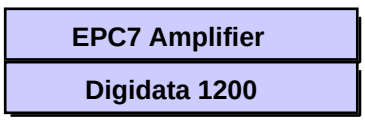
Note: Axon's manual has a chapter about the base address of the Digidata 1200 board. The addresses are given in hexadecimal notation. You have to pay attention that one usually has to enter the base address in decimal notation, not hexadecimal! Be aware of that possible source of troubles.

After you found the necessary free resources write down these values and set the selected I/O base address of the *Digidata 1200* board using the DIP-switches as described in the *Digidata 1200* manual. Then plug the *Digidata 1200* board in a free, long AT-bus slot.

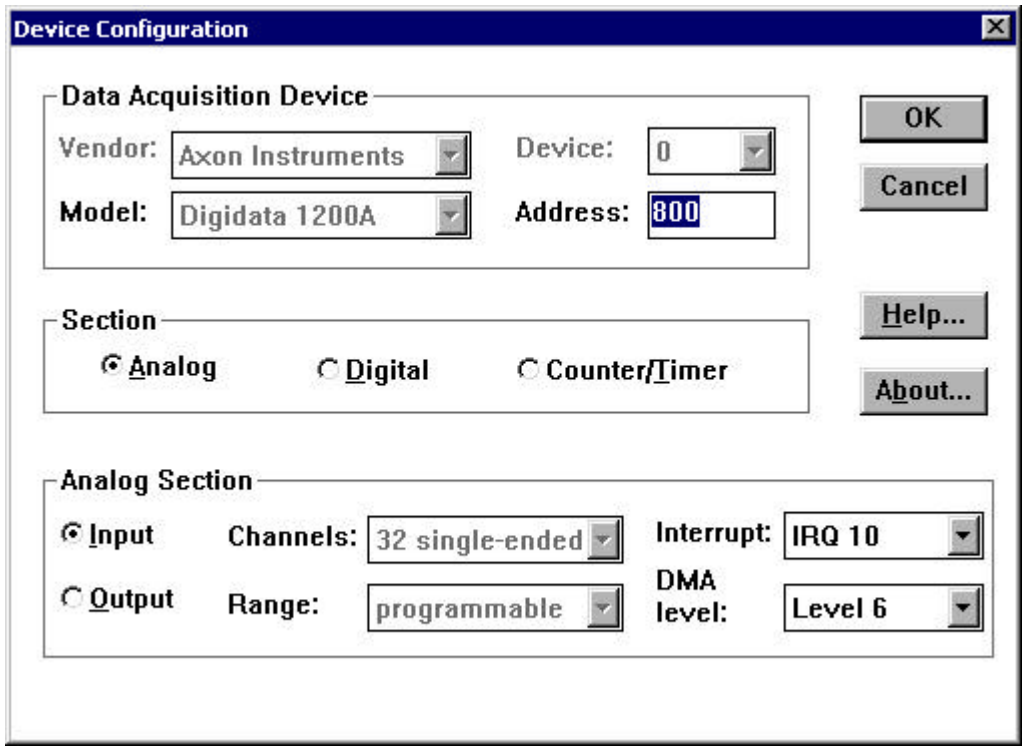
Install the *Digidata 1200* driver from the *PULSE* installation CD: This is done either by installing *PULSE* from the HEKA CD with the appropriate option enabled or by running Axon's installer **Setup.exe** which is located in the folder **Digidata\Drvl95** (for Windows 95) or **Digidata\Drvlnt** (for Windows NT). At the end of the driver installation you will have to reboot the computer.

After installing the driver you have to set the DMA-memory to be reserved for the *digidata 1200* board. Go to **Control Panel** → **DMA Memory** and check the following settings: the **Size** should be 64 KB and the number of **Buffers** should be 2 for *PULSE*. If this is not the case click the **Change**>> button an set the adequate amount. If you change one of the parameters you will have to reboot the computer again before the new settings get active.

Now, you can start up *PULSE* and switch to the *Configuration* window. You will have to configure *PULSE* to use the *digidata 1200* board as AD/DA-board. If the present amplifier is **EPC9** or **EPC8**, you cannot select the *digidata 1200* board. Thus, select any other amplifier, e.g., **EPC7** or **Telegraphing Amplifier** (*Note*:: when you choose **Telegraphing Amplifier**, *PULSE* will ask you immediately for the *I-Gain* ***.iga** and *Bandwidth* ***.bwt** lookup tables), then select the **Digidata 1200** as the AD/DA-board. *PULSE* will now try to initialize the *Digidata 1200* board for acquisition. For that, it will bring up the *Digidata 1200 Configuration* dialog. In this dialog you will have to enter the hardware settings to be used for the *Digidata 1200* board. You will have to supply the following information:



- The *base address* to be used: Please remember to supply the address that corresponds to the settings of the DIP switch on the board, either in decimal notation (e.g. 768) or in hexadecimal notation by preceding the value with "0x" (e.g. 0x300). Again, remember that Axon's manual gives the base address in hexadecimal notation without the "0x" preceding it! Some possible settings are 0x300 (decimal: 768), 0x320 (800) or 0x340 (832).
- The *IRQ* value: You will have to find a free IRQ value as shown above (10, 11, 12 or 15). Please, check the notes you made, when you searched for free resources.
- The DMA-channels: You have to search for two free DMA-channels as described above (5 and 6, 6 and 7 or 5 and 7). Please note, that you have to assign the higher DMA channel to the output DA-channels, and the lower to the input DA-channel.


Device Configuration
Data Acquisition Device
Vendor:
Axon Instruments
Device:
0
Model:
Digidata 1200A
Address:
800
OK
Cancel
Help...
About...
Section
Analog
Digital
Counter/Timer
Analog Section
Input
Output
Channels:
32 single-ended
Range:
programmable
Interrupt:
IRQ 10
DMA level:
Level 6

Questions & Answers

Q: Why is the trigger inverted? I checked it by reading it back and displaying it on the oscilloscope.

A: The trigger is not inverted. You are just using the wrong AD-channel to read it in. Read the trigger on any AD-channel which is neither the *"Current iI"* nor the *"Voltage In"* channel. These two AD-channels invert the signal depending on the selected acquisition modes (*On-Cell, Whole-Cell*, etc.).

Q: Is there a limit on the number of data points that can be displayed on the Oscilloscope window?

A: The maximal number of data which *PULSE+PULSEFIT* can display is the maximal number of samples in the stimulus buffer. So, if you increase that value, you increase also the other limit.

Q: Why does PULSE+PULSEFIT sometimes not display the complete sweep?

A: This is true, when the sweep is longer than the presently allocated stimulus buffer. This can happen e.g. if you acquired the data on a machine were you increased the stimulus buffer size and now try to analyze the data on another machine that has the default size. In this case sweeps are only displayed up to the current buffer limit. Increase the buffer size appropriately.

Q: Why does PULSE always switch to **Imon1** *channel in the Amplifier window, even when I select* **Imon2***?*

A: *PULSE+PULSEFIT* always will switch to the **Imon1** channel in the *Amplifier* window, when one selects **Auto Filter** in the *Configuration* window and the selected bandwidth is above 15kHz. The bandwidth of **Imon2** is limited by the highest filter 2 setting of about 15kHz, while **Imon1** has a much higher maximal bandwidth.

Q: Why is the option **VC and CC** *in the Pulse Generator no longer available?*

A: That option is there for compatibility reasons with earlier versions only. A stimulus is reasonable either for the *Voltage-Clamp* (amplitudes in [mV]) or the *Current-Clamp* mode (amplitudes in [pA]), but never both! Thus, the user has to select the appropriate mode before the experiment.

Q: There is always a red, "shadow" trace in the Oscilloscope. How can I get rid of it?

A: The red trace is the second acquired AD-channel. Its input, i.e. the BNC-input, is apparently not connected to the amplifier, and one records the capacitive

coupled image of trace one. If you do not require to acquire two traces, change the number of input channels in the *Pulse Generator* window from 2 to 1, and the "red" trace will disappear. You can delete all already acquired unwanted second traces with the **Tree → Delete 2nd Trace** option.

Q: In PULSEFIT one can edit the displayed membrane potential in the Oscilloscope window. Isn't it wrong since this value is a measured one and should not be allowed to be modified? Also, sometimes the displayed membrane potential seems to be wrong.

A: The behavior of *PULSEFIT* does not differ from the one of *PULSE* in that respect. Thus, the "membrane potential" field displays the membrane potential of the selected target in the *Replay* window **as long as the *Replay* window is the active one**! Also, one does not modify the stored value for the membrane potential in the *Oscilloscope* window (as soon as one switches to the *Oscilloscope* window the *Replay* window is no longer the front window!). The stored parameters within the sweeps can only be modified via the **Edit** option in the **Tree**-menu, provided that the data file is opened with the "modify" attribute.

Q: When I acquire continuous data from 2 AD-channels at 25 kHz I get huge data files larger than 1MB.What is going wrong?

A: The acquisition of one sweep during 1 minute (= 60,000 ms) at 40 µs (= 0.04 ms) per point (25 kHz) gives about 1,500,000 samples per sweep per acquired channel. With 2 acquired channels this results in 2.9 MB for every single sweep!

Q: I urgently need a button which will reduce the RS-setting by 10% in one "click". This is required to quickly reduce the RS-settings ,when the amplifier starts to oscillate.

A: Such an additional function can easily be added by recording a macro which reduces the RS-setting by a relative step of 10%. Then assign this new macro to a convenient macro button. You can even assign a command key to that new button such reducing the present RS-value by 10% is as quick as hitting one convenient key.

```
5 : RSDecrease
E  Relative:          TRUE
E  RsComp:          -10%
E  Relative:          FALSE
```

Another approach to quickly reduce the RS-settings of an oscillating amplifier is to quickly drag the mouse downwards (which will turn the compensation off) or use the standard feature to enter parameters by directly entering the numerical value. In the above example you can proceed as follows: When you slowly increment the RS-value and the amplifier starts to oscillate so badly hit the <ENTER> key of the numerical keypad, type a zero and press <ENTER>

again (to recall: hitting the <ENTER> key will allow you to type in the numeric value in the last edit field one was "dragging" the mouse).

An additional note to the above example: When the EPC9 or EPC8 gets into oscillations, one often has to reduce the RS-value much beyond 10% to bring back the amplifier to a stable state. This depends from many factors, such as gain and RS-speed settings.

Q: How can I store a modified trace?

A: You can use the **Replace Target Trace** in the **Buffer** menu to replace a trace with whatever is in the sweep buffer. The trace to be replaced (i.e., first, second, or leak trace) is set with the **Buffer → Use: Trace** option. Of course, the data file must have been opened with write permission (**File → Open Modify...** or **File → New...**).

Q: How can I improve the Auto-CFast and -CSlow compensation routines,, when there is a considerable leak?

A: You could use the automatic leak compensation routine to compensate the leak current when executing an automatic CSlow compensation. Thus, generate a macro that calls the Auto-GLeak just before calling Auto-CSlow, and then disables GLeak immediately thereafter. Here the listing of the proposed macro named "LeakCSlow":

```
4 : LeakCSlow
E  Mode:              3: Whole Cell
E  Gain:             11: 10 mV/pA
E  AutoGLeak
E  CSlow:            30.00pF
E  RSeries:          10.0MOhm
E  AutoCSlow
E  AutoCSlow
E  GLeak:            0.00 S
E  Bell
```

Q: How can I improve the Auto-CSlow compensation routine, when the two patched cells are connected by gap-junctions with a considerable conductance?

A: *PULSE+PULSEFIT* has an option to simultaneously send the stimulus used to estimate the capacitance to both cells. This will eliminate the leak current through the gap-junction, and, thus, allows the Auto-CSlow to succeed.

Q: The value for R_{memb} is correct, when a test pulse is running. Why then does the R_{memb} value appear to be meaningless, when I turn the test pulse off or switch to the oscilloscope window?

A: The R_{memb} value is computed in two ways:

When a test pulse is running, R_{memb} is computed as:

$$R_{memb} = \frac{\Delta U}{\Delta I} = \frac{Test\ Pulse\ Amplitude}{Current_{TestPulseSegment\ 1} - Current_{TestPulseSegment\ 2}}$$

When no test pulse is running, R_{memb} is computed as:

$$R_{memb} = \frac{V_{mon}}{I_{pip}} = \frac{Measured\ Steady\ StateVoltage}{Measured\ Steady\ State\ Pipette\ Current}$$

The first formula estimates R_{memb} by the relative current change evoked by the voltage jump in the test pulse. This kind of R_{memb} estimation is **insensitive** against leak currents, a possible reversal potential, and uncompensated amplifier offsets. Also, the applied test pulse amplitude as well as the evoked current can be measured with good precision.

The second formula has some major short-comings. It is **sensitive** to leak currents, a possible reversal potential, and uncompensated amplifier offsets. Also, precision artifacts can occur, especially when holding the cell at zero holding potential or when the measured current is very small. In most situations it is sufficient to apply a holding potential other than zero to get R_{memb} showing reasonable estimates again.

R_{memb} is displayed as "----", when a zero or negative resistance value is computed, e.g, when the holding voltage is between the reversal voltage and zero!

Q: How can I check, if the EPC9 produces the correct command potential?

A: You can easily check if the EPC9 produces the correct command potential by measuring the voltage monitor output, which is the command that will also be output at the probe. To do this you can connect the VOLTAGE MONITOR output of the EPC9 to an external oscilloscope or a potentiometer, generate a command potential and read back the value. Alternatively you can read back this value by using the EPC9 itself, just connect the VOLTAGE MONITOR with AD0 which is the default voltage input (this can be changed in the *Configuration* dialog of *PULSE*). In the EPC9 panel you should now read the potential in the **V-mon** field. You can change the precision of that display by clicking on it with CAPSLOCK on and CTRL+ALT (MacOS: CTRL+COMMAND) pressed. Set the type of the field to **engineering**, text to **V** and precision to **7** digits. The difference between the command and the output you read back should be less than 1 mV.

Q: I recorded currents using a wrong gain setting and now I want to correct for that. However, when I scale the sweeps by copying them into the buffer and using the **Scale...** *command from the* **Buffer** *menu I get artifacts at the very beginning of the segments as soon as I replace the target sweep by the buffer. Did I miss the point?*

A: The problem you describe results from the fact, that the buffer is represented internally by floating point values whereas the sweep data are 16 bit integers. A 16-bit signed integer can be any value between -2^{15} to $2^{15}-1$ (-32768 to 32767). When you scale a sweep (e.g. by multiplying it by a factor of 5) the integer value can easily overflow ("clip"). As long as you look at the buffer you don't see this overflow, since floating points have a much bigger range - however, the overflow will become eminent as soon as you copy the buffer back into the sweep again. An integer overflow will usually show up as a sign inversion. You can easily check this overflow by scaling the sweeps of group 3, series 1 of the experiment file **DEMO.DAT** from the HEKA CD: multiplying the first sweeps by 5 will overflow the integer range during the peak inward current while this does not occur with the last sweeps.

If you want to correct for a wrong amplifier gain during acquisition of the data you can do this much easier: in the *Replay* window select the target sweep and then type "E" (or select **Tree** → **Edit**). In the upcoming dialog select **Gain 1** (or **Gain 2**, if you recorded the second channel) from the popup and then click the **Modify** button. Now you can enter the correct amplifier gain!

Q: I have made a perfusion system that can be controlled from the EPC9 and I would like to make an online concentration-response curve and fit it to obtain EC_{50} or IC_{50}. This ***should*** *be possible - or not?*

A: *PULSEFIT* is able to do Dose-Response fits based on the solutions used during the experiment! Since *PULSE+PULSEFIT* stores the solution data together with the series, the dose-response fit is available within the "Group Fit" window were you can select the concentration of any ingredient of your solution as abscissa.

To sketch a possible bottom-up scenario let's assume you want to determine the effect of drugs on a voltage dependent channel:

1. In a typical experiment you apply repetitive sweeps and determine one parameter - such as the peak current at different potentials - at the **Sweep Fit** level. This will give you **one set of x- and y-values for every sweep**. The x-values were acquired with the data (e.g. voltage of the relevant segment, time, GSeries, CSlow...) the y-values are obtained from the sweep fit (e.g. peak or mean current, time constant of in-/activation, ...).
2. On the next level, the so called **Series Fit** you analyze the sweep fit parameters to obtain e.g. a current-voltage relationship by looking at the peak current versus the holding potential of the relevant segment of each sweep.

This will give you the next set of x- and y-data, **one set per series**. The x-values were acquired with the data (e.g. temperature, pipette pressure, solutions...) the y-values are obtained from the sweep fit itself (e.g. reversal potential, V_{half}, ...).

3. Each series of *PULSE* data was obtained using different internal or external solutions. On the top level - the **Group Fit** - you can analyze the data sets obtained from different series representing one pharmacological experiment (e.g. one substance). The **Group Fit** allows you to do a Dose-Response fit of any y-value obtained in the series fit versus the concentration of any internal or external substance or the pH / osmolarity of the solution.

Q: How can I include command characters in a serial command string?

A: To include command characters, i.e., and "non-ASCII" character, in a serial command string one has the following options:

A '\' character followed by a number is interpreted as its character code in octal representation. E.g., the string '\015' corresponds to the <RETURN> character. Alternatively the '\" character can be followed by the following **lower case** alphanumeric letters:

'\r' <RETURN>

'\l' <LINEFEED>

'\\'the backslash character itself

The above described feature is used in many environments, such as, e.g., PostScript and 'C'.

Troubleshooting

Restoring PULSE Data after a Computer Crash

Pulse stores every experiment into three files:

- The file with the extension ***.pgf** contains the stimulus templates used in a given experiment. The templates will be copied from your PGF pool into the running experiment.
- The ***.pul** file has the complete data tree as visualized in the **Replay** window. It includes e.g. all the amplifier settings such as filters or gain and the timing of the data.
- The ***.dat** file has only the actual raw data without any timing or scaling information.

With the option **Auto File Update** enabled *PULSE* writes the raw data after every sweep. The other two files are written only when you close or update the experiment by selecting **Update File** or **Close** from the **Pulse** menu. If the computer crashes during a running experiment, you will lose the ***.pgf** and the ***.pul** file. *PULSE* will not be able to reopen this experiment, although the ***.dat** file contains valid data with a high probability. However, there's a good chance to restore the two corrupted files in order to be able to access the experiment:

Backup the ***.dat** file to a safe place.

In *PULSE* create a new experiment. Call it whatever you want, e.g. something like **Test.dat**. With the new experiment opened, simulate your lost experiment by calling all the pgf protocols in the same order as you did during the experiment. You can use the model circuit or you may shield the probe.

Now you can throw away the new **Test.dat** file to replace it by your old experiment. Rename your raw data file into **Test.dat** or the two files **Test.pul** and **Test.pgf** using the name of your experiment. It is important, that all three files have the same name and only differ in their file name extension. This is the only criterion *PULSE* checks for files that belong to one experiment.

Now you can reopen the data file!

Unfortunately, the time information and all the CSlow, RSeries ... values are completely useless, however, the most important thing - your data - should still be there.

Lost Seals after going Whole-cell

PULSE tries to do as many corrections as possible - such as voltage conventions, leak correction, capacitance cancellation, liquid junction correction - *online* during the ongoing experiment so you can fully concentrate on the experiment instead of operating the amplifier. Also, you will have less work with the later *off-line* analysis. However, this sometimes might lead to situations, were one can easily misunderstand, what is going on.

Polarity of Command Potentials Applied by PULSE

PULSE tries to use the physiological convention whenever stimulating or acquiring data: therefore a **downward current** always reflects a **cationic influx into** the cell. However, for this to work properly you have to always instruct the program about the actual recording configuration. The way of doing this is the **Mode** popup: **On Cell**, **In Out**, **Whole Cell** and **Out Out**. In the two modes *On-Cell* and *Inside-Out* the stimulus and the acquired data are automatically inverted by the program since inward currents into the cell are **outward** currents from the pipette. With the other two modes (*Outside-Out* and *Whole-Cell*), no inversion occurs, since inward currents into the cell are also inward currents into the pipette.

Setting up the Right Recording Mode

If you need hyperpolarization to stabilize the seal formation, select the mode **Whole Cell** before trying to establish the gigaohm seal. This will force negative voltages to become negative compared to the **pipette** not compared to the **cell membrane**. However, the predefined **SET-UP** macro sets the recording mode to **On Cell**. Therefore you should modify the file **DefaultEpc9.mac** (or **DefaultEpc7.mac, if** you are using an EPC7/8) which contains the default macros used by *PULSE*. This file is an ASCII text so you can use any text editor such as **Simple Text** under MacOS or the **Notepad** of Windows 95/NT. In the first section (**SET-UP** macro) replace the line

```
E  Mode:  1; On Cell
```

by

```
E  Mode:  3; Whole Cell
```

This will activate the *Whole-Cell* recording configuration whenever you execute the **SET-UP** macro. This macro can be executed by pressing the first button on the top of the amplifier panel or typing “1” on the numerical keypad.

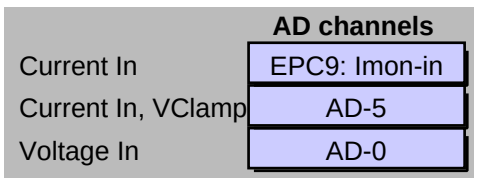
If you use the Windows 95 Notepad, go into the explorer after saving the file and delete the extension "**.txt**" from the filename. The MS Notepad has a known “bug”, it ignores file extensions and instead always appends "**.txt**" to every

file it saves. To see the complete file name including the extensions, disable the option **View → Options… → View → Hide MS-DOS file extensions for file types that are registered** in the Windows Explorer.

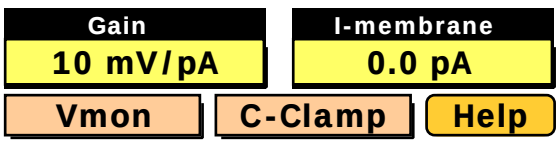
After rupturing the patch you should execute the **WHOLE-CELL** macro (the third button on the top of the panel). This will activate the *Whole Cell* recording mode and execute an automatic C-Slow compensation.

Current-Clamp Recordings with PULSE and the EPC9

The VOLTAGE MONITOR output of the EPC9 must be connected to a free AD channel with a BNC cable. The predefined channel is **AD-0**, but you can use any available channel, if you change the "Voltage In" assignment in the Configuration Window.



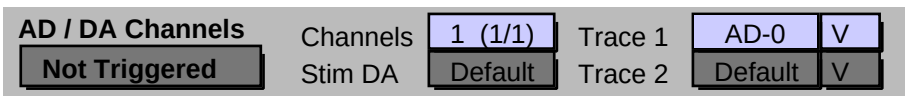
Whenever you switch into the *Current-Clamp* mode in the **Amplifier** window, *PULSE* changes the active channel to **Vmon**. This is necessary to observe the voltage changes due to the injected current.



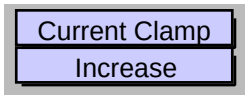
Alternatively you can enable the **Pulse Mode: Both** option in the **Test Pulse** section of the **Configuration** window. This will instruct *PULSE* to always show the current plus the voltage when it is running the test pulse.

Note: The CC- and the VC mode have different settings for the test pulse, therefore PULSE sets its amplitude to 0 pA the first time you switch into CC-mode. In order to not accidentally damage the cell, the test pulse is turned off every time you switch from CC- into VC and vice versa. This way you can safely change its amplitude and duration and then turn it on again manually (or make a macro which switches to CC and reactivates the test pulse)!

In the **AD / DA Channels** section of your pulse protocol (in the PGF editor window) set the channel to be acquired to the ADC you connected with the VOLTAGE MONITOR e.g. **Trace 1 = AD-0**, when you are using the default connection.



In the PGF editor set the mode of the pulse sequence to be **Current Clamp** using the popup in the middle right part of the window.



If you want to run the protocol directly from the *Whole Cell* mode, you should create a macro that switches into the *Current Clamp* mode and assign this macro to your PGF sequence (**Macros: Start**).

There are a few more things to keep in mind when using the *Current-Clamp* mode:

- If *RS-Compensation* has been activated in the *Voltage-Clamp* mode this becomes a **bridge compensation** in the *Current-Clamp* mode. *PULSE* will remember the %-settings of the compensation for both modes separately. However, you have to turn the compensation on every time, because it will be turned off automatically in order to avoid possible overcompensation. If you wish the *RS-Compensation* to be on after mode switches automatically, you can easily create a macro that sets the right mode (*Whole-Cell* or *Current Clamp*) and then re-activates the *RS-Compensation* again.
- The injected current is normally scaled at a gain of 1 pA/mV. Thus the maximum current you can inject into the circuitry is ±1 nA: 1 pA/mV * 10 V (range of the amplifier) * 0.1 (stimulus scaling of the EPC9). If a higher maximal current is required, once can increase the CC stimulus range (for EPC9 version "B" and "D" or later) to 10 pA/mV, which results in a maximal current of ±10 nA.
- The *Fast Current-Clamp* mode is very sensitive against misadjustment of the *C-Fast* setting. Especially overcompensation causes the signal to oscillate. Therefore *PULSE* automatically subtracts a small amount of fast capacitance (0.5 pF) whenever you switch into the *Current-Clamp* mode. (the previous C-Fast value remains desplayed).
- *PULSE* uses different settings for the test pulse in the voltage and current clamp mode. If you switch into current clamp the first time, the amplitude of test pulse will be 0 pA. If you then turn the test pulse on and set its duration and amplitude to any desired value you can easily adjust the **Rs %**-*setting* of the bridge compensation. Overcompensation will show up immediately if your test pulse is not too long.
- Maybe the *Current-Clamp* test pulse is to small or short to show up in the oscilloscope. When you want to increase the **display** of the test pulse in the oscilloscope (just like in *E9Screen*) turn the setting **Scale Test Pulse** on. You can find it in the **Files and Paths** section of the **Configuration** dialog.

Printing Problems

Printing Problems under MacOS

Most printing problems on MacOS computers are caused by three types of problems:

You changed the printer in the **Chooser** while running *PULSE+PULSEFIT*, version 8.09 and earlier. Thus, if a printing problem occurs, check whether it does occur when the printer selection is made before you start *PULSE+PULSEFIT*.

Another source of problems may be the printer driver itself. Some of Apple's printer drivers show peculiar incompatibilities. All our programs do print with the **LaserWriter** printer driver 8.51. Printer driver 7.0 and 8.1 were the cause of some incompatibilities with *PULSE+PULSEFIT*.

The most frequent printing problem type is due to *out-of-memory* situations of the printer drivers. The default memory settings are usually too tight especially when you print to a laser printer. This produces misleading error messages, such as *"printing 312 jobs"* and the like. The printer driver is located in the **Extensions** folder within the **System Folder** and called **PrintMonitor**, in case of a HP-printer its name is **HP PrintMonitor**. If you have the *Desktop Printing Extension* installed that comes with System 7.5 and higher, there is a second printer driver called **Desktop PrintMonitor** in the same place. The default memory assignments is 160 kB for the **PrintMonitor** and 117 kB for the **Desktop PrintMonitor**. To relieve out-of-memory situations you should increase the memory allocation of **both** drivers by about 400 kB using the Finder's **File → Get Info** menu.

Here some further steps you can try to relieve persisting printing problems:

- Turn **Virtual Memory** on in the **Memory** Control Panel and make sure that the **Modern memory manager** is selected and 32-bit memory is enabled (if your computer allows to turn it off). This may cure the general lack of memory.
- Assign 500 kB more memory to *PULSE+PULSEFIT* itself in the **Get Info** window. This may cure those drivers which grab memory directly from the application instead of the system heap like the HP-drivers!
- Install more physical RAM. It may not only cure the printing problems, but also speed up the performance of your computer. 16 MB of physical RAM is the minimum to run *PULSE+PULSEFIT* with System 7.5, 24 MB is at least required with System 8.x.
- Print to a PostScript file and send this file to the printer after quitting *PULSE+PULSEFIT*.

Finally, does your printer have enough RAM to print? The typical RAM-requirement is at least 2 MB for a 300x300 dpi printer, 4 MB for 600x300, and 8 MB for 600x600 dpi, plus about 1 MB.

Printing Problems under Windows 3.1 and 95

It can happen that *PULSE+PULSEFIT* begins to crash after one printed something. This can be caused by a printer driver which enables math-exceptions for its own use. By that, *PULSE+PULSEFIT* will not be able to handle correctly math-exceptions, and may crash, e.g., when performing power functions during fitting routines. There is no solution to that problem, besides finding another printer driver which does not show that problem. You can switch to Windows NT where this problem does not occur.

Printing Troubleshooter


Try the following to identify what the source of the problems may be:

Q: Can you print the Notebook?

A: This is a small and fast job. It tests the printer connection, initialization and page ejection. If that test is working it indicates that the *PULSE+PULSEFIT* printer routines are working properly.

Q: Can you print one Series on one page? Are you sure you waited long enough?

A: We measured the time it took to print the "Tail → IV 2" series of the **Demo.dat** experiment. On an otherwise speedy PowerPC 8500 printing to a Personal Laser Writer NT took **five** minutes without and **three** minutes with **Tree → Export → Print Compressed Vectors** selected. The same using a Pentium II (300 MHz) on a HP LaserJet 6P took about 15 seconds in both cases.

Q: Does it make a difference, if you print with or without the option **Tree → Export → Print Compressed Vectors** *selected?*

A: If these tests perform correctly, then most probably the problems are located outside *PULSE+PULSEFIT*!

MacOS Specific Problems

“

Offscreen” Error Message after starting PULSE

The so called “offscreen” is a feature to accelerate the redrawing of windows. *PULSE* reserves some memory to redraw the window in the background and updates it after the redraw has finished. The amount of necessary memory depends on the size of the window. Sometimes however, *PULSE* or one of its extensions stores a wrong (negative) size of windows. When the program restarts again it tries to allocate the buffer for that window according to the dimensions, which become tremendously huge because the negative values are converted automatically into huge positive ones. E.g. -1 x -1 becomes 32767 x 32767 which would result in a memory requirement of 1 GB at a resolution of 256 colors. Of course, you don't have that amount of memory, so you get this error message. You should get rid of the problem by deleting all *PULSE* preferences: **DefaultPulse.set** in the **PULSE** folder and the files **Pulse.Settings** and **PulseFit.Settings** in the **Preferences** folder inside the **System Folder**. Take care, you will have to reconfigure *PULSE* again after throwing away its preferences!

Wrong Paths when storing Files

The MacOS gives you three possibilities, which path will be set by default when calling the *Open* or *Save* dialog. This setting is available through the **General Controls** control panel. Usually, you should keep the **Folder that is set by the application** setting. This will allow *PULSE+PULSEFIT* to preset the paths you specified within the *Configuration* dialog, i.e., if you wish to save or open a PGF file, *PULSE* will take you to your PGF-folder by default. If you, however, enable the option **Last folder used in the application**, *PULSE* will always take you to the last folder used. E.g., if you save an experiment into your data folder and then wish to open a PGF file, *PULSE* will take you to the last data folder instead of the PGF folder from the *Configuration*.

The Default Configuration File on a PowerPC

One can starts *PULSE+PULSEFIT* by double clicking on any ***.set** file or on an alias pointing to the ***.set** file. On a Quadra computer this works as expected, and *PULSE+PULSEFIT* will boot up and load the configuration settings from that selected file. Yet, on a PowerPC computer, *PULSE* will always load the configuration file named **DefaultPulse.set** and *PULSEFIT* will load the configuration file **DefaultFit.set**.

Thus, on a PowerPC one must proceed differently to be able to use multiple configurations. The easiest way is to store the *Configuration* file with the default file name in the **Common Path** folder. Then, you can move that file to any other folder you want! You should create a special folder for each individual *Configuration* file. Starting *PULSE+PULSEFIT* by double clicking on this Configuration file, even when it is not in the **Common Path**, will load the settings from that file.

Tip: If you remove the files ***DefaultPulse.set*** and ***DefaultFit.set*** from the ***Pulse*** folder, PULSE+PULSEFIT will ask you during startup whether to use the default configuration or to look for a ****.set*** file. In the later case you can run the program with any configuration thus allowing you to quickly change experimental conditions.

Resolving Minor Problems

Problem: PULSE+PULSEFIT cannot read the protection key (dongle):

- Install the latest version of **EvE Init**. The latest version is 1.9 and is supplied on the HEKA CD.

Problem: Computer is slow and "Time Overrun" messages appear:

- The first reason may be "memory starvation". If you have only 8 Mbyte of RAM and you installed memory-hungry system extensions, there may not be enough memory for the system itself to run. The operating system needs additional memory, e.g., for every new font and font size used, for every new created file, etc. Thus the system will perform a "memory compaction" every time the used font size changes. If your "System Software" memory allocation is 2.5 Mbytes, there will be no memory left for the memory requirement of the operating system when *PULSE+PULSEFIT* is running. To solve this problem, remove unnecessary system extensions, control panels, INITs, and the like, until the **About this Macintosh** dialog reports some free memory when *PULSE+PULSEFIT* is running.
- A second source are other applications running in the background, such as networking. You should turn them off during any important experiment.
- Another possibility is that you may have activated one or both of the options **Calculate folder sizes** and **Show disk info in header** in the **Views** control panel. This will continuously compute the exact folder sizes and waste a lot of computing power, especially when you are writing to disk. Thus, always deselect these options.

Resolving Major Problems

If you encounter more severe problems when trying to start or run *PULSE+PULSEFIT* there may be a problem with any of the following:

- *PULSE+PULSEFIT* itself (i.e., software bugs or incompatibilities)
- The computer hardware used (RAM-chips, SCSI-connections, etc.)
- The EPC9 hardware, including the Mac-23 board
- The settings of the computer used
- System extensions, utilities, and INITs installed on the user computer
- Viruses

First off, you should check the following:

- **Connections**: Check, if the Mac-23 board is properly installed and the connecting cable is plugged in. Check, if the EPC9 is powered up.
- **Control Panel settings**. These are the suggested defaults of the Control Panels:
  - **Memory:** File Cache Size 128 kbytes, Virtual memory OFF; 32-bit Addressing; RAM-disk OFF.
  - **Cache Control**: CPU cache ON
  - **Monitors**: 256 colors, more colors will slow down the graphics
  - **Views**: Calculate folder sizes OFF, Show disk info in header OFF
  - **Sharing Setup**: File Sharing OFF, Program Sharing OFF
  - **General Controls**: Documents…Folder which contains the Application.

If the problem persists, you should perform the following:

- **Rebuild the desktop file**. Rebuilding the desktop is often necessary when documents fail to launch after double-clicking, or when custom icons are replaced with generic document or application icons. Traditionally, pressing 'OPT' + 'CMD' before the Finder loads will force the invisible desktop file to "rebuild" itself.

Note: Quite often, this is not completely effective because the original desktop file was corrupted, so rebuilding it only yields an updated desktop file which is still damaged. "TechTool" (a utility program which can be found on Info-Mac or any of its numerous mirror sites) provides a better solution to rebuilding the desktop. It actually deletes the original desktop. The next time the Finder loads, it will create a brand new desktop file.

- **System Extension, Control Panel, and INIT conflicts**. To test this, restart your computer while keeping 'SHIFT' + 'SPACE' pressed. This will disable any extension from loading.
- **Network driver or printer driver conflicts**. Inactivate networks and printers in the "Chooser", then restart your computer.

And finally:

- **Zap the PRAM**. The parameter RAM (PRAM) contains user-definable settings that must be retained after the computer has been deactivated. Settings such as time of day, mouse scaling, keyboard repeat rate, and startup drive preferences are all stored in the upper 64 bytes of PRAM. Traditionally, one could clear or "zap" these upper 64 bytes of PRAM by holding a special key combination ('OPT' + 'CMD' + 'P' + 'R') at startup. This often cures behavioral anomalies which cannot be remedied with software replacement.

Note: Below the standard 64 bytes of PRAM lies another 192 bytes of memory which are, for the most part, publicly undocumented. These are secret storage areas that Apple uses for such things as Manufacture Date and Factory Service settings. When these portions of the PRAM become corrupted with invalid data, odd problems can occur and sometimes the machine will fail to work at all. Traditional PRAM zapping does not clear the lower 192 bytes of PRAM. The only alternative is to remove the PRAM battery, which is often soldered to the logic board. TechTool clears all 256 bytes of PRAM memory without the need to remove the battery. Once the system is rebooted, the MacOS ROMs will replace the PRAM contents with its default or factory settings. Some settings will revert to their factory defaults (color-capable Macs will revert to 1-bit B&W, the printer port will revert to AppleTalk Active, mouse speed will revert to super-slow). Adjusting Control Panel and Chooser settings will correct the above problems.

- **Virus disinfection**. Scan all your system and disks with a virus-checker (e.g., "Disinfectant"; which can be found on Info-Mac or any of its numerous mirror sites).

If these procedures do not cure the problem, there might be hardware incompatibilities:

- **Hardware conflicts**. Remove all installed boards, including the Mac23 board. Does *PULSE+PULSEFIT* now start? Re-install the boards one after the other, testing the performance of *PULSE+PULSEFIT* each time.
- **Bad RAM chips**. RAM chips are never checked during bootup. It can occur that a bad RAM chip is failing only when it is hot, or only sporadically. Thus, it can happen that, e.g., *E9Screen* is running (it uses only 2 Mbyte of space) but *PULSE+PULSEFIT* does not (it needs at least 5.5 Mbyte).

If none of this helps, call our customer hotline.

Windows Specific Problems

Crash with "Page Fault" Error under Windows 95

This error message is issued by the memory manager of Windows 95. The error **page fault** means that it run out off virtual memory. Probably the swap file on disk is too small, or too many programs are simultaneously executed. If the second condition does not hold, check the size of the Windows swap-file, and increase it, if required. More physical RAM may also be very beneficial, 16 MB is too limited for Windows 95 anyway. 32 MB are more advisable. But RAM itself cannot be the problem: *PULSE+PULSEFIT* did run without problems under Windows 3.1 on a 33 MHz 86486, with as little as 4 MB (four!) of RAM. It was slow, though, acquiring a sweep once in three seconds.

Appendix I: Data Structure

These are the definitions of the formats of the various data files created by *PULSE*:

Stim.de

```
DEFINITION MODULE Stim;

(*
 * Stim data file format
 *
 * This module defines the data types to be used for the stimulation in
 * experiments involving pulsed data; it uses the tree structures as
 * defined in module Tree and provides types for a 'StimTree'.
 *)

FROM SYSTEMp1 IMPORT INT16, INT32, SET16;

CONST
   VersionNumber  = 7;

(* Structure of the trees *)

CONST
   TreeLevels     = 3;   (* 3 Levels: SHeader, Stimulation, StimSegment *)

VAR
   RamSizes       : ARRAY [0..TreeLevels-1] OF INT32;

(*
 * RecordTypes for the StimTree:  Root, Stimulation, Segment
 *
 * Stimulation
 * StimulationRecord
 *
 * A StimulationRecord describes a stimulation pulse made up of one or
 * more segments. Each segment has a voltage and a duration. The segment
 * can be a constant output voltage (SegmentConstant) or a ramp from the
 * previous segment voltage to the present segment voltage (SegmentRamp).
 * For conditioning pulses a segment "SegmentConditioning" is provided.
 * No P/n is performed for such segments. For long conditioning pulses
 * a possibly less accurate timing is applied.
 * Incrementing voltages or times in a ramped record is allowed, but be
 * sure to think about what will happen; incrementing is done before the
 * ramp parameters are evaluated.
 *
 * When a series of pulses are output, the Delta entries in each
 * StimulationSegment describe the parameter change to perform for each
 * pulse in the series. We have:
 *
```

```
 *       Voltage(n)  := Voltage(0) * DeltaVFactor^n + DeltaVIncrement * n;
 *       Duration(n) := Duration(0) * DeltaTFactor^n + DeltaTIncrement * n;
 *
 * The Leak sweeps usually follow the test sweeps with a delay LeakDelay.
 * If LeakDelay is negative then leak sweeps precede test sweeps and Leak
 * Delay is the time between the end of the last leak sweep and the
 * beginning of the test sweep
 *)

TYPE SegmentClass = ( SegmentConstant,
                      SegmentRamp,
                      SegmentConditioning,
                      SegmentContinuous,
                      SegmentConstSine,
                      SegmentRampSine );

TYPE StimSegmentRecord = RECORD
   Class                : SegmentClass; (* constant, ramp, or condit.   *)
   IsHolding            : BOOLEAN;      (* TRUE if Voltage is to be set *)
   Voltage              : LONGREAL;     (* to holding                   *)
   Duration             : LONGREAL;
   DeltaVFactor         : LONGREAL;
   DeltaVIncrement      : LONGREAL;
   DeltaTFactor         : LONGREAL;
   DeltaTIncrement      : LONGREAL;
END (* RECORD *);

TYPE StimSegment = POINTER TO StimSegmentRecord;

TYPE Trigger = RECORD
   TriggerSegment       : INT16;
   TriggerTime          : LONGREAL;   (* Seconds, within trig.Segment *)
   TriggerLength        : LONGREAL;   (* Length of Trig.Pulse, sec    *)
   TriggerAmplitude     : LONGREAL;   (* Amplitude of Trig.Pulse      *)
   TriggerDac           : INT16;
END; (* RECORD *)

TYPE IncrementModeType  = ( ModeInc,
                            ModeDec,
                            ModeIncInterleaved,
                            ModeDecInterleaved,
                            ModeAlternate );

TYPE GUpdateType        = ( NoGUpdate,
                            SwGSlow,
                            SwGFast,
                            SwGBoth,
                            SeGSlow,
                            SeGFast,
                            SeGBoth );

TYPE WriteModeType      = ( WriteEnabled,
                            WriteDisabled,
                            NoWriteNoShow,
                            WriteButNoShow );

TYPE ExtTriggerType     = ( TrigNone, TrigSeries, TrigSweep );
```

```
TYPE PostVmembType      = ( VmembSweepIncr,
                            VmembValue,
                            VmembSeriesIncr );
TYPE LockInType         = ( Loff, Lnormal, Lpwlinear );

TYPE AmplModeType       = ( AllAmplModes, VCAmplMode, CCAmplMode );

TYPE StimulationRecord  = RECORD
   FileName          : ARRAY [0..13] OF CHAR;  (* Source File            *)
   EntryName         : ARRAY [0..13] OF CHAR;  (* Identifier             *)
   SampleInterval    : LONGREAL;       (* Seconds                       *)
   FilterFactor      : LONGREAL;       (* oversampling factor           *)
   SweepInterval     : LONGREAL;       (* Repetition-Interval           *)
   NumberSweeps      : INT32;          (* Number of sweeps in Series    *)
   NumberRepeats     : INT32;          (* Number of Seq. repeats        *)
   RepeatWait        : LONGREAL;       (* Wait between repeats          *)
   LinkedSequence    : ARRAY [0..13] OF CHAR;
   LinkedWait        : LONGREAL;       (* Wait between linked sequences *)

   LeakCount         : INT32;          (* number of leak sweeps         *)
   LeakSize          : LONGREAL;       (* rel. amplitude of leak pulse  *)
                                       (* e.g. 0.25 for P/4 protocol    *)
   LeakHolding       : LONGREAL;
   LeakAlternate     : BOOLEAN;        (* norm. or alt. leak protoc.    *)
   AltLeakAveraging  : BOOLEAN;        (* norm. or alt. leak while aver.*)
   LeakDelay         : LONGREAL;       (* Seconds                       *)
   Trig              : ARRAY [0..2] OF Trigger;
   NumberOfTriggers  : INT16;          (* Number actually used; 0<x<4   *)
   RelevantXSegment  : INT16;          (* usually that which changes    *)
   RelevantYSegment  : INT16;          (* usually that which changes    *)

   WriteMode         : WriteModeType;
   IncrementMode     : IncrementModeType;

   TotalSweepLength  : INT32;        (* Total sweeplength including a   *)
                                     (* possible continuous segment,    *)
                                     (* in units of sample intervals.   *)

   MaxSweepLength    : INT32;        (* max length of sweep to be shown.*)
                                     (* For a pulse without a continuous*)
                                     (* segment it is identical to the  *)
                                     (* max sweeplength from trig #1,   *)
                                     (* in units of sample intervals    *)

   InputChannels     : INT16;        (* # input channels. Default=1     *)
   GUpdate           : GUpdateType;  (* make C-slow/fast bef. each pulse*)
                                     (* only for EPC9                   *)
   RelAbsPot         : BOOLEAN;      (* absolute or relativ potentials  *)

   HasContinuous     : BOOLEAN;
   LogIncrement      : BOOLEAN;

   StimDac           : INT16;
   Adc1              : INT16;
   Adc2              : INT16;
   YUnit1            : ARRAY[0..1] OF CHAR;
```

```
   YUnit2            : ARRAY[0..1] OF CHAR;

   VmembIncrement    : REAL;

   ExtTrigger        : ExtTriggerType;
   FileTemplate      : BOOLEAN;

   StimKind          : SET16;    (* meaning of bits:
                                      0 => extended fields defined
                                      1 => LockIn.Active
                                      2 => Fura.Active
                                     15 => set: use scan-rate!
                                  *)

   LockInCycle          : LONGREAL;
   LockInAmplitude      : LONGREAL;

   FuraOn               : BOOLEAN;
   VmembMode            : PostVmembType;
   FuraTotLength        : LONGREAL;
   FuraDelay            : LONGREAL;
   FuraLength1          : LONGREAL;
   FuraLength2          : LONGREAL;
   FuraWaveLength0      : LONGREAL;
   FuraWaveLength1      : LONGREAL;
   FuraWaveLength2      : LONGREAL;
   FuraRepeats          : INT16;

   LockInSkip           : INT32;
   LockInVReversal      : LONGREAL;
   LockInMode           : LockInType;
   LockInShow           : BOOLEAN;

   ConfigMacro          : ARRAY[0..15] OF CHAR;
   EndMacro             : ARRAY[0..15] OF CHAR;

   AmplModeKind         : AmplModeType;
   NoStartWait          : BOOLEAN;

   ActualInChannels     : INT16;
   ActualOutChannels    : INT16;
   ActualAdc1           : INT16;
   ActualAdc2           : INT16;
 END (* RECORD *);

TYPE Stimulation        = POINTER TO StimulationRecord;


TYPE RootRecord         = RECORD
   Version              : INT16;
END (* RECORD *);

TYPE Root               = POINTER TO RootRecord;

END Stim.
```

Pulsed.de

```
DEFINITION MODULE Pulsed;

(*
 * Pulsed data file format
 *
 * This module defines the data types to be used for control information on
 * experiments involving pulsed data; it uses the tree structures as
 * defined in module Tree and provides types for a 'PulsedTree'.
 *)

FROM SYSTEM IMPORT ADDRESS, BYTE; FROM SYSTEMp1 IMPORT INT16, INT32, SET16;

CONST
   VersionNumber     = 7;

(* Structure of the trees
 *)

CONST
   TreeLevels        = 4;  (* 4 Levels: PHeader, Group, Series, Sweep *)

VAR
   RamSizes          : ARRAY [0..TreeLevels-1] OF INT32;

(*    RecordTypes    : Root, Group, Series, Sweep
 *
 * Sweep
 * SweepRecord
 *
 * A SweepRecord describes the sampled data from a single stimulation/
 * acquisition.
 *)

CONST
   ExtendedBit       = 0;
   FuraBit           = 1;
   LockInBit         = 2;
   DataFormatBit     = 3;
   LittleEndianBit   = 4;
   ScanRateBit       = 15;

TYPE
   StringType        = ARRAY [0..13] OF CHAR;
   CommentType       = ARRAY [0..79] OF CHAR;
   RootTextType      = ARRAY [0..4],[0..79] OF CHAR;

   DataAbscissaType = ( Time,
                        Amplitude, LnAmplitude, LogAmplitude,
                        Frequency, LnFrequency, LogFrequency );

   DataFormatType    = ( int16, int32, real32, real64 );

TYPE SweepRecord     = RECORD
   Time              : LONGREAL;       (* UNIX (seconds)+ Rest            *)
```

```
    StimCount        : INT32;          (* relevant entry in StimTree      *)
    SweepCount       : INT32;          (* .. at time of acquisition       *)
    AverageCount     : INT32;          (* number of on-line averages      *)
    Leak             : BOOLEAN;        (* TRUE, if Record has a leaksweep *)
    SecondTrace      : BOOLEAN;        (* TRUE, if Record has a 2. trace  *)
    Label            : StringType;
    DataPoints       : INT32;          (* Number of data points in RAM    *)
    Data             : INT32;          (* Offset in raw data file         *)
    DataPointer      : ADDRESS;        (* Raw data ADDRESS in memory      *)
        (* EPC 7/9 settings *)
    DataFactor1      : LONGREAL;       (* Amperes/ADC-unit                *)
    DataFactor2      : LONGREAL;       (* Volts/ADC-unit                  *)
    CSlow            : LONGREAL;       (* Capacitance, Farad              *)
    GSeries          : LONGREAL;       (* Series conductance, Siemens     *)
    RsValue          : LONGREAL;       (* Series resistance setting, Ohms *)
    Mconductance     : LONGREAL;
        (* Results of pre-analysis  *)
    ZeroCurrent      : LONGREAL;       (* Amperes                         *)
    OnlineYResult    : LONGREAL;       (* Param. determined by last anal. *)
    OnlineXResult    : LONGREAL;       (* Param. determined by last anal. *)

    TotalPoints      : INT32;          (* Total data points in file       *)
    Offset           : INT32;          (* Offset of loaded data           *)

    SweepKind        : SET16;   (* meaning of bits:
                                   ExtendedBit   => extended fields defined
                                   FuraBit       => Fura.Active
                                   LockInBit     => LockIn.Active
                                   DataFormatBit => "DataFormat" field valid
                                   LittleEndianBit => byte sequence
                                                      "Mac" = cleared
                                                      "DOS" = set
                                   ScanRateBit   => set: use scan-rate!
                                *)

    FuraPoints       : INT32;          (* Number of data points in RAM    *)
    FuraData         : INT32;          (* Offset in raw data file         *)
    FuraPointer      : ADDRESS;        (* Raw data ADDRESS in memory      *)
    OnlineYResult2   : LONGREAL;       (* Param. determined by last anal. *)
    OnlineXResult2   : LONGREAL;       (* Param. determined by last anal. *)

    DispFactor1      : LONGREAL;       (* reserved by PULSE               *)
    DispFactor2      : LONGREAL;       (* reserved by PULSE               *)

      (*
        Since we introduced the new "DataFormat" field in a field
        which before was without importance, we are interpreting it
        ONLY if "SweepKind" has ExtendedBit and DataFormatBit set!
        Thus, the procedure "StimIO.InitializeSweep" scans
        the "*.pul" tree upon loading and enforces that check.
       *)
    DataFormat       : DataFormatType;   (* Kind of abscissa, INT or REAL *)
    DataAbscissa     : DataAbscissaType;
    Timer            : LONGREAL;

    FuraStart        : INT32;
    VideoTime        : LONGREAL;
```

```
   Spares           : ARRAY[0..3] OF CHAR;
END (* RECORD *);

TYPE Sweep = POINTER TO SweepRecord;

(*
 * Series
 * SeriesRecord
 *
 * A SeriesRecord describes a Series of Sweeps; this is characterized
 * by the StimulationRecord, that generates it; also the record in-
 * cludes a number of environmental parameters that are obtained
 * before the series is started.
 *)

TYPE
  RecordingModeType     = ( InOut,
                            OnCell,
                            OutOut,
                            WholeCell,
                            PCClamp,
                            VClamp,
                            NoRec,
                            TestInt,
                            TestExt );

  UserParamType         = RECORD
    Value               : LONGREAL;
    Name                : StringType;
    Unit                : ARRAY [0..1] OF CHAR;
  END;

  EPC9StateType         = ARRAY[0..103] OF BYTE;   (* E9Panel.StateType *)

  SeriesRecord          = RECORD
    Time                : LONGREAL;       (* seconds                    *)
        (* Patch parameters, usually obtained before series *)
    Bandwidth           : LONGREAL;       (* Hertz                      *)
    PipettePotential    : LONGREAL;       (* Volts                      *)
    CellPotential       : LONGREAL;       (* Volts                      *)
    PipetteResistance   : LONGREAL;       (* Ohms                       *)
    SealResistance      : LONGREAL;       (* Ohms                       *)
    BackgroundNoise     : LONGREAL;       (* Amperes, RMS               *)
    Temperature         : LONGREAL;       (* Degrees C                  *)
    PipettePressure     : LONGREAL;       (* in cm H20                  *)
    UserParam1          : UserParamType;
    UserParam2          : UserParamType;
    RecordingMode       : RecordingModeType;
    VideoMode           : CHAR;
    Comment             : CommentType;
      (* EPC9 state record *)
    EPC9State           : EPC9StateType;
    InternalSolution,
    ExternalSolution    : INT32;        (* solutions according to solution
                                           code as stored in '*.sol'  *)
    ExtraYUnit1         : ARRAY [0..1] OF CHAR;
    ExtraYUnit2         : ARRAY [0..1] OF CHAR;
```

```
    DispYUnit1          : ARRAY [0..3] OF CHAR; (* reserved by PULSE    *)
    DispYUnit2          : ARRAY [0..3] OF CHAR; (* reserved by PULSE    *)

    FuraK               : LONGREAL;
    FuraMin             : LONGREAL;
    FuraMax             : LONGREAL;

    LockInExtPhase      : LONGREAL;

    Timer               : LONGREAL;

      (* Reserved for extensions *)

    ExtCalPhase         : LONGREAL;
    ExtCalAttenuation   : LONGREAL;
    PLPhase             : LONGREAL;
    PLPhaseY1           : LONGREAL;
    PLPhaseY2           : LONGREAL;
    ExtCalValid         : BOOLEAN;
    PLPhaseValid        : BOOLEAN;

      (* EPC9 state record *)
    EPC9State           : EPC9StateType;

  END (* RECORD *);

TYPE Series = POINTER TO SeriesRecord;

(*
 * Group
 * GroupRecord
 *
 * A GroupRecord describes a group of series, such as patch and
 * whole cell currents obtained simultanuously, or groups of series
 * obtained in sequence under different sets of conditions
 *)

TYPE GroupRecord        = RECORD
   Label                : StringType;
   Text                 : CommentType;
   ExperimentNumber     : INT32;
   ExtraLongReal        : LONGREAL;
END (* RECORD *);

TYPE Group = POINTER TO GroupRecord;

(*
 * Root
 * RootRecord
 *
 * A RootRecord describes a complete experiment.
 *)

TYPE RootRecord         = RECORD
   Version              : INT16;
   VersionName          : StringType;
```

```
   FileName             : StringType; (* the part common to all *)
   Comments             : RootTextType;
   StartTime            : LONGREAL;        (* UNIX Time  +Rest     *)
END (* RECORD *);

TYPE Root = POINTER TO RootRecord;

END Pulsed.
```

Solution.de

```
DEFINITION MODULE Solution;

(*
 * Purpose: Handling of Pulsed solution files and database.
 *)

FROM SYSTEM IMPORT ADDRESS; FROM SYSTEMp1 IMPORT INT16, INT32;

CONST
   VersionNumber  = 1;

   TreeLevels     = 3; (* Root, Sol, Chemical *)

VAR
   RamSizes       : ARRAY [0..TreeLevels-1] OF INT32;

(*
   RecordTypes: Root, Sol, Chemical

   Chemical - Description of one component of the solution
*)

CONST
   MaxChemicalNameLength  = 29;

TYPE
   ChemicalNameType = ARRAY [0..MaxChemicalNameLength] OF CHAR;

TYPE ChemicalRecord = RECORD
   Concentration : REAL;              (* molar concentration *)
   Name          : ChemicalNameType;  (* ASCII identifier, e.g. "NaCl" *)
END (* RECORD *);

TYPE Chemical = POINTER TO ChemicalRecord;

(*
   Sol - Description of a solution
*)

CONST
   MaxSolutionNameLength  = 79;

TYPE
```

```
   SolutionNameType = ARRAY [0..MaxSolutionNameLength] OF CHAR;

TYPE SolutionRecord = RECORD
    Number      : INT32;              (* number in common data base      *)
    Name        : SolutionNameType;   (* ASCII identifier                *)
    Numeric     : REAL;               (* numerical value, like Ca Conc   *)
    NumericName : ChemicalNameType;   (* description of numeric,         *)
                                      (* e.g. "Free Ca"                  *)
    pH          : REAL;               (* optimum pH                      *)
    pHCompound  : ChemicalNameType;   (* adjusted the pH with            *)
    Osmol       : REAL;               (* measured osmolarity, mosm       *)
END (* RECORD *);

TYPE Sol = POINTER TO SolutionRecord;

(*
   Root - File description
*)

TYPE RootRecord = RECORD
   Version       : INT16;
   DataBaseName  : ARRAY [0..79] OF CHAR; (* name of Solution Data Base
that is
                                             used to define Sol.Number *)
END (* RECORD *);

TYPE Root = POINTER TO RootRecord;

END Solution.
```

Analysis.de

```
DEFINITION MODULE Analysis;

(*
   Purpose:    Structure of analyzed data.

   Pulse Analysis Format:

   --------      ---------     ---------     -----------
   |Header| -->  |Group  | --> |Series | --> | Sweep 1 |
   |      | .    |       | .   |       | .   |         |
   -------- .    |Group- | .   |Series-| .   | SweepRec|
            .    | Descr | .   | Descr | .   -----------
                 |       | .   |       | .
                 |Group- | .   |Series-| --> -----------
                 | Record| .   | Record| .   | Sweep 2 |
                 --------- .   |       | .   |         |
                           .   |Sweep- | .   | SweepRec|
                           .   | Descr | .   -----------
                           .   --------- .      ...
                           .             --> -----------
                           .                 | Sweep n |
                           .                 |         |
                           .                 | SweepRec|
                           .                 -----------
                           .
                           .   ---------     -----------
                           --> |Series | --> | Sweep 1 |
                           .   |       | .   |         |
                           .   |Series-| .   | SweepRec|
                           .   | Descr | .   -----------
                               |       | .
                               |Series-| --> -----------
                               | Record| .   | Sweep 2 |
                               |       | .   |         |
                               |Sweep- | .   | SweepRec|
                               | Descr | .   -----------
                               --------- .      ...
                                         --> -----------
                                             | Sweep m |
                                             |         |
                                             | SweepRec|
                                             -----------
*)

FROM SYSTEMp1 IMPORT INT16, INT32;

CONST
   VersionNumber        = 5;

   TreeLevels           = 4;  (* Root , Group , Series , Sweep *)
```

```
VAR
   RamSizes             : ARRAY [0..TreeLevels-1] OF INT32;

 CONST
   LABELLENGTH          = 12;
   MAXENTRIES           = 15;

   SWEEPLABELLENGTH     = 12;
   MAXSWEEPENTRIES      = 15;

TYPE
   GeneralInfo = RECORD
      Time              : LONGREAL;     (* time of analysis *)
      AnalysisType      : INT16 ;       (* type of analysis *)
   END;

   TYPE
      StatusType = ( StatusFitted, StatusFixed, StatusSkipped );

   EntryRecord          = RECORD
      Value             : LONGREAL;
      Status            : StatusType;
      Filler            : BOOLEAN;
   END;

   DescriptionType = ARRAY [0..MAXENTRIES-1],
                           [0..LABELLENGTH-1] OF CHAR;
   DescrPointer    = POINTER TO DescriptionType;

   AnalysisType    = ARRAY [0..MAXENTRIES-1] OF EntryRecord;
   AnalysisPointer = POINTER TO AnalysisType ;

(*
   Root
*)

TYPE

   RootRecord           = RECORD
      Version           : INT16;

      SeriesLabelLength : INT16;
      MaxSeriesEntries  : INT16;

      SweepLabelLength  : INT16;
      MaxSweepEntries   : INT16;

      GroupLabelLength  : INT16;
      MaxGroupEntries   : INT16;
   END;

   Root                 = POINTER TO RootRecord;

(*
  Group
*)
```

```
   GroupRecord          = RECORD
      General           : GeneralInfo ;
      GroupDescription  : DescriptionType ;
      GroupAnalysis     : AnalysisType;
   END;

   Group                = POINTER TO GroupRecord;

(*
   Series
*)

   SweepDescriptionType  = ARRAY [0..MAXSWEEPENTRIES-1],
                           [0..SWEEPLABELLENGTH-1] OF CHAR;
   SweepDescrPointer     = POINTER TO SweepDescriptionType;

   SeriesRecord         = RECORD
      General           : GeneralInfo;
      SeriesDescription : DescriptionType;
      SeriesAnalysis    : AnalysisType;
      SweepDescription  : SweepDescriptionType;
   END;

   Series               = POINTER TO SeriesRecord;

(*
   Sweep
*)

TYPE
   AnalysisRangeType = RECORD
     YSegmentOffset : INT16 ;
     LeftCursor     : LONGREAL ;
     RightCursor    : LONGREAL ;
   END ;

TYPE

   GeneralSweepInfo     = RECORD
      SweepCount        : INT32;       (* Number in parent Pulsed Series *)
      AnalysisType      : INT16 ;
      AnalysisRange1    : AnalysisRangeType ;
      AnalysisRange2    : AnalysisRangeType ;
   END;

   SweepAnalysisType        = ARRAY [0..MAXSWEEPENTRIES-1] OF EntryRecord;
   SweepAnalysisTypePointer = POINTER TO SweepAnalysisType;

   SweepRecord          = RECORD
      General           : GeneralSweepInfo;
      SweepAnalysis     : SweepAnalysisType;
   END;

   Sweep                = POINTER TO SweepRecord;

END Analysis.
```

Appendix II: Record Offset Bytes

These are the record offset bytes of the various data files created by *PULSE*:

StimDef.de

```
DEFINITION MODULE StimDef;

(*
 * Stim data file format, see module Stim.de in Appendix I.
 *
 * This module defines the data types to be used for the stimulation
 * in experiments involving pulsed data; it uses the tree structures
 * as defined in module Tree and provides types for a 'StimTree'.
 *
 * The sizes in bytes of the used variables are:
 *    BYTE           1
 *    CHAR           1
 *    enumeration    1
 *    SET16          2
 *    INT16          2
 *    INT32          4
 *    ADDRESS        4
 *    REAL           4
 *    LONGREAL       8
 *
 * Here, we define the record fields by their offsets from the
 * beginning of their respective record blocks:
 *)

CONST
   Unit2Size            =  2;
   StringSize           = 14;
   MacroNameSize        = 16;

   (* StimSegmentRecord = RECORD *)
   SeClass              =   0;
   SeIsHolding          =   1;
   SeVoltage            =   2;
   SeDuration           =  10;
   SeDeltaVFactor       =  18;
   SeDeltaVIncrement    =  26;
   SeDeltaTFactor       =  34;
   SeDeltaTIncrement    =  42;
   StimSegmentSize      =  50;
   (* END StimSegmentRecord *)

   (* StimulationRecord = RECORD *)
   StFileName           =   0;
   StEntryName          =  14;
```

```
StSampleInterval     =  28;
StFilterFactor       =  36;
StSweepInterval      =  44;
StNumberSweeps       =  52;
StNumberRepeats      =  56;
StRepeatWait         =  60;
StLinkedSequence     =  68;
StLinkedWait         =  82;
StLeakCount          =  90;
StLeakSize           =  94;
StLeakHolding        = 102;
StLeakAlternate      = 110;
StAltLeakAveraging   = 111;
StLeakDelay          = 112;
StTriggerSegment1    = 120;
StTriggerTime1       = 122;
StTriggerLength1     = 130;
StTriggerAmplitude1  = 138;
StTriggerDac1        = 146;
StTriggerSegment2    = 148;
StTriggerTime2       = 150;
StTriggerLength2     = 158;
StTriggerAmplitude2  = 166;
StTriggerDac2        = 174;
StTriggerSegment3    = 176;
StTriggerTime3       = 178;
StTriggerLength3     = 186;
StTriggerAmplitude3  = 194;
StTriggerDac3        = 202;
StNumberOfTriggers   = 204;
StRelevantXSegment   = 206;
StRelevantYSegment   = 208;
StWriteMode          = 210;
StIncrementMode      = 211;
StTotalSweepLength   = 212;
StMaxSweepLength     = 216;
StInputChannels      = 220;
StGUpdate            = 222;
StRelAbsPot          = 223;
StHasContinuous      = 224;
StLogIncrement       = 225;
StStimDac            = 226;
StAdc1               = 228;
StAdc2               = 230;
StYUnit1             = 232;
StYUnit2             = 234;
StVmembIncrement     = 236;
StExtTrigger         = 240;
StFileTemplate       = 241;
StStimKind           = 242;
StLockInCycle        = 244;
StLockInAmplitude    = 252;
StFuraOn             = 260;
StVmembMode          = 261;
StFuraTotLength      = 262;
StFuraDelay          = 270;
StFuraLength1        = 278;
```

```
   StFuraLength2        = 286;
   StFuraWaveLength0    = 294;
   StFuraWaveLength1    = 302;
   StFuraWaveLength2    = 310;
   StFuraRepeats        = 318;
   StLockInSkip         = 320;
   StLockInVReversal    = 324;
   StLockInMode         = 332;
   StLockInShow         = 333;
   StConfigMacro        = 334;
   StEndMacro           = 350;
   StAmplModeKind       = 366;
   StNoStartWait        = 367;
   StActualInChannels   = 368;
   StActualOutChannels  = 370;
   StActualAdc1         = 372;
   StActualAdc2         = 374;
   StimulationSize      = 376;
   (* END StimulationRecord *)

   (* RootRecord        = RECORD *)
   RoVersion            =   0;
   RootSize             =   2;
   (* END RootRecord *)

END StimDef.
```

PulsedDef.de

```
DEFINITION MODULE PulsedDef;

(*
 * Pulsed data file format, see module Pulsed.de in Appendix I.
 *
 * This module defines the data types to be used for control information on
 * experiments involving pulsed data; it uses the tree structures as
 * defined in module Tree and provides types for a 'PulsedTree'.
 *
 * The sizes in bytes of the used variables are:
 *    BYTE           1
 *    CHAR           1
 *    enumeration    1
 *    SET16          2
 *    INT16          2
 *    INT32          4
 *    ADDRESS        4
 *    REAL           4
 *    LONGREAL       8
 *
 * Here, we define the record fields by their offsets from the
 * beginning of their respective record blocks:
 *)

CONST
```

```
Unit2Size            =   2;
Unit4Size            =   4;
SwSpareSize          =  10;
StringSize           =  14;
CommentSize          =  80;
EPC9StateSize        = 104;
RootTextSize         = 400;
SeExtraLongReals     =   4;

(* SweepRecord       = RECORD *)
SwTime               =   0;
SwStimCount          =   8;
SwSweepCount         =  12;
SwAverageCount       =  16;
SwLeak               =  20;
SwSecondTrace        =  21;
SwLabel              =  22;
SwDataPoints         =  36;
SwData               =  40;
SwDataPointer        =  44;
SwDataFactor1        =  48;
SwDataFactor2        =  56;
SwCSlow              =  64;
SwGSeries            =  72;
SwRsValue            =  80;
SwMConductance       =  88;
SwZeroCurrent        =  96;
SwOnlineYResult      = 104;
SwOnlineXResult      = 112;
SwTotalPoints        = 120;
SwOffset             = 124;
SwSweepKind          = 128;
SwFuraPoints         = 130;
SwFuraData           = 134;
SwFuraPointer        = 138;
SwOnlineYResult2     = 142;
SwOnlineXResult2     = 150;
SwDispFactor1        = 158;
SwDispFactor2        = 166;
SwDataFormat         = 174;
SwDataAbscissa       = 175;
SwTimer              = 176;
SwFuraStart          = 184;
SwVideoTime          = 188;
SwSpares             = 196;
SweepSize            = 200;
(* END SweepRecord *)

(* SeriesRecord      = RECORD *)
SeTime               =   0;
SeBandwidth          =   8;
SePipettePotential   =  16;
SeCellPotential      =  24;
SePipetteResistance  =  32;
SeSealResistance     =  40;
SeBackgroundNoise    =  48;
```

```
   SeTemperature        =  56;
   SePipettePressure    =  64;
   SeUserParam1Value    =  72;
   SeUserParam1Name     =  80;
   SeUserParam1Unit     =  94;
   SeUserParam2Value    =  96;
   SeUserParam2Name     = 104;
   SeUserParam2Unit     = 118;
   SeRecordingMode      = 120;
   SeVideoMode          = 121;
   SeComment            = 122;
   SeEPC9State          = 202;
   SeInternalSolution   = 306;
   SeExternalSolution   = 310;
   SeExtraYUnit1        = 314;
   SeExtraYUnit2        = 316;
   SeDispYUnit1         = 318;
   SeDispYUnit2         = 322;
   SeFuraK              = 326;
   SeFuraMin            = 334;
   SeFuraMax            = 342;
   SeLockInExtPhase     = 350;
   SeTimer              = 358;
   SeExtraLongReal      = 366;
   SeExtCalPhase        = 398;
   SeExtCalAttenuation  = 406;
   SePLPhase            = 414;
   SePLPhaseY1          = 422;
   SePLPhaseY2          = 430;
   SeExtCalValid        = 438;
   SePLPhaseValid       = 439;
   SeriesSize           = 440;
(* END SeriesRecord *)

   (* GroupRecord       = RECORD *)
   GrLabel              =   0;
   GrText               =  14;
   GrExperimentNumber   =  94;
   GrExtraLongReal      =  98;
   GroupSize            = 106;
   (* END GroupRecord *)

   (* RootRecord        = RECORD *)
   RoVersion            =   0;
   RoVersionName        =   2;
   RoFileName           =  16;
   RoComments           =  30;
   RoStartTime          = 430;
   RootSize             = 438;
   (* END RootRecord *)

END PulsedDef.
```

SolutionDef.de

```
DEFINITION MODULE SolutionDef;

(*
 * Handling of Pulsed solution files and database, see module Solution.de
 * in Appendix I.
 *
 * The sizes in bytes of the used variables are:
 *    BYTE           1
 *    CHAR           1
 *    enumeration    1
 *    SET16          2
 *    INT16          2
 *    INT32          4
 *    ADDRESS        4
 *    REAL           4
 *    LONGREAL       8
 *
 * Here, we define the record fields by their offsets from the
 * beginning of their respective record blocks:
 *)

CONST
   ChemicalNameLength   =  30;
   SolutionNameLength   =  80;

   (* ChemicalRecord    = RECORD *)
   ChConcentration      =   0;
   ChName               =   4;
   ChemicalSize         =  34;
   (* END ChemicalRecord *)

   (* SolutionRecord    = RECORD *)
   SoNumber             =   0;
   SoName               =   4;
   SoNumeric            =  84;
   SoNumericName        =  88;
   SopH                 = 118;
   SopHCompound         = 122;
   SoOsmol              = 152;
   SolutionSize         = 156;
   (* END SolutionRecord *)

   (* RootRecord        = RECORD *)
   RoVersion            =   0;
   SoDataBaseName       =   2;
   RootSize             =  82;
   (* END RootRecord *)

END SolutionDef.
```

AnalysisDef.de

```
DEFINITION MODULE Analysis;

(*
 * Structure of analyzed data file, see module Analysis.de in Appendix I.
 *
 * The sizes in bytes of the used variables are:
 *    BYTE           1
 *    CHAR           1
 *    enumeration    1
 *    SET16          2
 *    INT16          2
 *    INT32          4
 *    ADDRESS        4
 *    REAL           4
 *    LONGREAL       8
 *
 * Here, we define the record fields by their offsets from the
 * beginning of their respective record blocks:
 *)

CONST

   MAXENTRIES         = 15;
   AnalysisSize       = MAXENTRIES * EntrySize;
   DescriptionSize    = 180;

   (* EntryRecord     = RECORD *)
   Value              =   0;
   Status             =   8;
   Filler             =   9;
   EntrySize          =  10;
   (* END EntryRecord *)

   (* RootRecord      = RECORD *)
   Version            =   0;
   SeriesLabelLength  =   2;
   MaxSeriesEntries   =   4;
   SweepLabelLength   =   6;
   MaxSweepEntries    =   8;
   GroupLabelLength   =  10;
   MaxGroupEntries    =  12;
   RootSize           =  14;
   (* END RootRecord *)

   (* GroupRecord     = RECORD *)
   GrGenTime          =   0;
   GrGenAnalysisType  =   8;
   GroupDescription   =  10;
   GroupAnalysis      = 190;
   GroupSize          = 340;
   (* END GroupRecord *)

   (* SeriesRecord    = RECORD *)
   SeGenTime          =   0;
```

```
   SeGenAnalysisType  =   8;
   SeriesDescription  =  10;
   SeriesAnalysis     = 190;
   SweepDescription   = 340;
   SeriesSize         = 520;
   (* END SeriesRecord *)

   (* SweepRecord     = RECORD *)
   SweepCount         =   0;
   SwAnalysisType     =   4;
   YSegmentOffset1    =   6;
   LeftCursor1        =   8;
   RightCursor1       =  16;
   YSegmentOffset2    =  24;
   LeftCursor2        =  26;
   RightCursor2       =  34;
   SweepAnalysis      =  42;
   SweepSize          = 192;
   (* END SweepRecord *)

END AnalysisDef.
```

Appendix III: 'C'-Headers

These are header files for the 'C' programming language.

Warning: The variables on disk are **packed**, i.e., there are no additional padding bytes anywhere. You must keep in mind that most compilers will align the variables in the structures differently!

Stim.h

```
// Stim.h

typedef unsigned char BOOLEAN;
typedef unsigned char CHAR;
typedef float REAL;
typedef double LONGREAL;
typedef void *ADDRESS;
typedef signed char INT8;
typedef signed short INT16;
typedef signed long INT32;
typedef unsigned char SET8;
typedef unsigned short SET16;
typedef unsigned char ENUM_8;

typedef struct String2Type_Struct {CHAR a [2];} String2Type;
typedef struct String14Type_Struct {CHAR a [14];} String14Type;
typedef struct String16Type_Struct {CHAR a [16];} String16Type;

typedef ENUM_8 SegmentClass;

typedef struct StimSegment_Struct
{
    SegmentClass Class;
    BOOLEAN IsHolding;
    LONGREAL Voltage;
    LONGREAL Duration;
    LONGREAL DeltaVFactor;
    LONGREAL DeltaVIncrement;
    LONGREAL DeltaTFactor;
    LONGREAL DeltaTIncrement;
} StimSegment;

typedef ENUM_8 IncrementModeType;
typedef ENUM_8 GUpdateType;
typedef ENUM_8 WriteModeType;
typedef ENUM_8 ExtTriggerType;
typedef ENUM_8 PostVmembType;
typedef ENUM_8 LockInShowType;
typedef ENUM_8 AutoRangingType;
```

```
typedef ENUM_8 AmplModeType;
typedef ENUM_8 RelAbsPotType;

typedef struct Stimulation_Struct
{
    String14Type FileName;
    String14Type EntryName;
    LONGREAL SampleInterval;
    LONGREAL FilterFactor;
    LONGREAL SweepInterval;
    INT32 NumberSweeps;
    INT32 NumberRepeats;
    LONGREAL RepeatWait;
    String14Type LinkedSequence;
    LONGREAL LinkedWait;
    INT32 LeakCount;
    LONGREAL LeakSize;
    LONGREAL LeakHolding;
    BOOLEAN LeakAlternate;
    BOOLEAN AltLeakAveraging;
    LONGREAL LeakDelay;
    INT16 Trigger1Segment;
    LONGREAL Trigger1Time;
    LONGREAL Trigger1Length;
    LONGREAL Trigger1Amplitude;
    INT8 Trigger1Dac;
    SET8 Trigger1DigValue;
    INT16 Trigger2Segment;
    LONGREAL Trigger2Time;
    LONGREAL Trigger2Length;
    LONGREAL Trigger2Amplitude;
    INT8 Trigger2Dac;
    SET8 Trigger2DigValue;
    INT16 Trigger3Segment;
    LONGREAL Trigger3Time;
    LONGREAL Trigger3Length;
    LONGREAL Trigger3Amplitude;
    INT8 Trigger3Dac;
    SET8 Trigger3DigValue;
    INT16 NumberOfTriggers;
    INT16 RelevantXSegment;
    INT16 RelevantYSegment;
    WriteModeType WriteMode;
    IncrementModeType IncrementMode;
    INT32 TotalSweepLength;
    INT32 MaxSweepLength;
    INT16 InputChannels;
    GUpdateType GUpdate;
    RelAbsPotType RelAbsPot;
    BOOLEAN HasContinuous;
    BOOLEAN LogIncrement;
    INT16 StimDac;
    INT16 Adc1;
    INT16 Adc2;
    String2Type YUnit1;
    String2Type YUnit2;
    REAL VmembIncrement;         // warning: this is only a REAL!
```

```
    ExtTriggerType ExtTrigger;
    BOOLEAN FileTemplate;
    SET16 StimKind;
    LONGREAL LockInCycle;
    LONGREAL LockInAmplitude;
    BOOLEAN FuraOn;
    PostVmembType VmembMode;
    LONGREAL FuraTotLength;
    LONGREAL FuraDelay;
    LONGREAL FuraLength1;
    LONGREAL FuraLength2;
    LONGREAL FuraWaveLength0;
    LONGREAL FuraWaveLength1;
    LONGREAL FuraWaveLength2;
    INT16 FuraRepeats;
    INT32 LockInSkip;
    LONGREAL LockInVReversal;
    AutoRangingType AutoRanging;
    LockInShowType LockInShow;
    String16Type ConfigMacro;
    String16Type EndMacro;
    AmplModeType AmplModeKind;
    BOOLEAN NoStartWait;
    INT16 ActualInChannels;
    INT16 ActualOutChannels;
    INT16 ActualAdc1;
    INT16 ActualAdc2;
} Stimulation;

typedef struct Root_Struct
{
    INT16 Version;
} Root;
```

Pulsed.h

```
// Pulsed.h

typedef unsigned char BOOLEAN;
typedef unsigned char CHAR;
typedef double LONGREAL;
typedef void *ADDRESS;
typedef signed short INT16;
typedef signed long INT32;
typedef unsigned short SET16;
typedef unsigned char ENUM_8;

typedef struct String2Type_Struct {CHAR a [2];} String2Type;
typedef struct String4Type_Struct {CHAR a [4];} String4Type;
typedef struct String10Type_Struct {CHAR a [10];} String10Type;
typedef struct String14Type_Struct {CHAR a [14];} String14Type;
typedef struct CommentType_Struct {CHAR a [80];} CommentType;
typedef struct RootTextType_Struct {CHAR a [400];} RootTextType;
typedef ENUM_8 DataAbscissaType;
```

```
typedef ENUM_8 DataFormatType;

typedef struct Sweep_Struct
{
    LONGREAL Time;
    INT32 StimCount;
    INT32 SweepCount;
    INT32 AverageCount;
    BOOLEAN Leak;
    BOOLEAN SecondTrace;
    String14Type Label;
    INT32 DataPoints;
    INT32 Data;
    ADDRESS DataPointer;
    LONGREAL DataFactor1;
    LONGREAL DataFactor2;
    LONGREAL CSlow;
    LONGREAL GSeries;
    LONGREAL RsValue;
    LONGREAL MConductance;
    LONGREAL ZeroCurrent;
    LONGREAL OnlineYResult;
    LONGREAL OnlineXResult;
    INT32 TotalPoints;
    INT32 Offset;
    SET16 SweepKind;
    INT32 FuraPoints;
    INT32 FuraData;
    ADDRESS FuraPointer;
    LONGREAL OnlineYResult2;
    LONGREAL OnlineXResult2;
    LONGREAL DispFactor1;
    LONGREAL DispFactor2;
    DataFormatType DataFormat;
    DataAbscissaType DataAbscissa;
    LONGREAL Timer;
    INT32 FuraStart;
    LONGREAL VideoTime;
    String4Type Spares;
} Sweep;

typedef ENUM_8 RecordingModeType;
typedef struct EPC9Type_Struct {unsigned char a [104];} EPC9Type;

typedef struct Series_Struct
{
    LONGREAL Time;
    LONGREAL Bandwidth;
    LONGREAL PipettePotential;
    LONGREAL CellPotential;
    LONGREAL PipetteResistance;
    LONGREAL SealResistance;
    LONGREAL BackgroundNoise;
    LONGREAL Temperature;
    LONGREAL PipettePressure;
    LONGREAL UserParam1Value;
    String14Type UserParam1Name;
```

```
    String2Type UserParam1Unit;
    LONGREAL UserParam2Value;
    String14Type UserParam2Name;
    String2Type UserParam2Unit;
    RecordingModeType RecordingMode;
    BYTE SeVideoMode;
    CommentType Comment;
    EPC9Type EPC9State;
    INT32 InternalSolution;
    INT32 ExternalSolution;
    String2Type ExtraYUnit1;
    String2Type ExtraYUnit2;
    String4Type DispYUnit1;
    String4Type DispYUnit2;
    LONGREAL FuraK;
    LONGREAL FuraMin;
    LONGREAL FuraMax;
    LONGREAL LockInExtPhase;
    LONGREAL Timer;
    LONGREAL ExtraLongReal1;
    LONGREAL ExtraLongReal2;
    LONGREAL ExtraLongReal3;
    LONGREAL ExtraLongReal4;
    LONGREAL ExtCalPhase;
    LONGREAL ExtCalAttenuation;
    LONGREAL PLPhase;
    LONGREAL PLPhaseY1;
    LONGREAL PLPhaseY2;
    BOOLEAN ExtCalValid;
    BOOLEAN PLPhaseValid;
} Series;

typedef struct Group_Struct
{
    String14Type Label;
    CommentType Text;
    INT32 ExperimentNumber;
    LONGREAL ExtraLongReal;
} Group;

typedef struct Root_Struct
{
    INT16 Version;
    String14Type VersionName;
    String14Type FileName;
    RootTextType Comments;
    LONGREAL StartTime;
} Root;
```

Solution.h

```
// Solution.h

typedef unsigned char BOOLEAN;
```

```
typedef unsigned char CHAR;
typedef float REAL;
typedef double LONGREAL;
typedef signed short INT16;
typedef signed long INT32;
typedef unsigned char ENUM_8;

typedef struct String30Type_Struct {CHAR a [30];} String30Type;
typedef struct String80Type_Struct {CHAR a [80];} String80Type;

typedef struct Chemical_Struct
{
    REAL Concentration;
    String30Type Name;
} Chemical;

typedef struct Solution_Struct
{
    INT32 Number;
    String80Type Name;
    REAL Numeric;
    String30Type NumericName;
    REAL pH;
    String30Type pHCompound;
    REAL Osmol;
} Solution;

typedef struct Root_Struct
{
    INT16 Version;
    String30Type DataBaseName;
} Root;
```

Appendix IV: Loading Files

Here comes the actual source of a module that could be used to read a tree:

Sample Program

```
MODULE FileFormat;

FROM SYSTEM IMPORT ADR, ADDRESS, BYTE, LONG, SHORT;
FROM SYSTEMp1 IMPORT INT32, ADDADR;

IMPORT Alert, FileSelect, IOBytes, IOFiles, Strings, TermIO, Buffer;

(*
 * MagicNumber - This is a special value used as a prefix for a tree
 * stored to a file.  The value contains the four byte values of the
 * characters 'Tree' in order.
 *
 * SwappedMagicNumber - This is the MagicNumber written by a CPU
 * which used the opposite byte ordering.
 *)

CONST
   MagicNumber        = 054726565H; (* i.e. "Tree" in ASCII *)
   SwappedMagicNumber = 065657254H; (* i.e. "eerT" in ASCII *)

(*
 * SwappedInt32 - Swaps the byte of a 32 bit long variable.
 *)

PROCEDURE SwappedInt32( Value : INT32 ): INT32;
VAR
   Source,
   Target      : POINTER TO ARRAY[0..3] OF BYTE;
   Result      : INT32;
BEGIN
   Source      := ADR( Value );
   Target      := ADR( Result );
   Target^[0]  := Source^[3];
   Target^[1]  := Source^[2];
   Target^[2]  := Source^[1];
   Target^[3]  := Source^[0];
   RETURN Result;
END SwappedInt32;

(*
 * LoadOneRecord
 *
 *    Loads one data block.
 *    All "TermIO" statements are for demonstration purpose only.
 *
 *    The variables are
```

```
 *           Stream        : the file handle to the open file
 *           FileSize      : the number of bytes the data block has in
 *                           the file.
 *           MemorySize    : the byte length of the memory block where
 *                           the data is going to be stored.
 *           WhereToStore  : the address of where the data block is
 *                           going to be stored.
 *
 *    The procedure returns TRUE, if it encountered no errors.
 *)

PROCEDURE LoadOneRecord(
            Stream       : IOFiles.FileHandleType;
            FileSize     : LONGINT;
            MemorySize   : LONGINT;
            WhereToStore : ADDRESS )
            : BOOLEAN;

VAR
   Excess    : LONGINT;
   FileBytes : LONGINT;

BEGIN

   (* Here we load the next block of data into memory.
    *
    * First, we have to compare the number of bytes we can load from
    * the file with the bytes of allocated memory for that record.
    *
    * There are 3 possibilities:
    *    1. The size of the allocated memory ("MemorySize") equals the
    *       number of bytes of the data block in the file ("FileSize").
    *       Thus, we can load the complete block.
    *    2. There are fewer bytes in the file than we expect. This can
    *       occur, e.g., when the file has been written by an earlier
    *       version which used fewer parameters than the present one.
    *       In this case, we would have to zero out those fields which
    *       are not filled with data from the file.
    *    3. There are more bytes in the file than we expect. This can
    *       happen, when the program which created that tree file was
    *       using more parameters than we presently know of. In that
    *       case, we would load only as much byte as we had reserved
    *       RAM for.
    *)

   Excess    := MemorySize - FileSize;

   IF Excess = 0D THEN
      (* The file record has as many bytes as there is space in RAM *)

      FileBytes := MemorySize;

   ELSIF Excess < 0D THEN
      (* The file record has more many bytes than there is space in RAM.
       * Load only as many bytes as there is space in RAM.
       *)

      FileBytes := MemorySize;
```

```
   ELSE (* i.e., Excess > 0D *)
      (* The file record has fewer bytes than there is space in RAM.
       * Load only as many bytes as there are in the file.
       *)

      FileBytes := FileSize;

      (* Do not forget to clear the remaining fields which are not going
       * to be filled from the file.
       *)

      Buffer.Set( ADDADR( WhereToStore, FileSize ), Excess, 0 );

   END (* IF *);

   RETURN IOBytes.Read( Stream, FileBytes, WhereToStore );

END LoadOneRecord;

(*
 * LoadOneLevel
 *
 *    Processes the loading of one data block from a "Tree", and all
 *    its "children".
 *    All "TermIO" statements are for demonstration purpose only.
 *
 *    The variables are
 *           Stream        : the file handle to the open file
 *           Sizes         : the array containing the level sizes
 *           Levels        : the number of levels in the tree
 *           NeedsByteSwap : the flag telling, whether byte-swapping is
 *                           needed
 *           Level         : the actual tree level to load
 *           IsFirst       : a flag telling, whether it is the first child
 *                           loaded. This is only required for text output!
 *           Position      : the variable containing the position in the
 *                           file.
 *
 *    The procedure returns TRUE, if it encountered no errors.
 *)

PROCEDURE LoadOneLevel(
             VAR Stream    : IOFiles.FileHandleType;
             VAR Sizes     : ARRAY OF INT32;
             VAR Levels    : LONGINT;
             NeedsByteSwap : BOOLEAN;
             Level         : LONGINT;
             IsFirst       : BOOLEAN;
             VAR Position  : LONGINT )
             : BOOLEAN;

VAR
   Count     : INT32;
   Size      : LONGINT;
   Children  : LONGINT;
   i         : INTEGER;
   WriteInfo : BOOLEAN;

BEGIN
```

```
   WriteInfo := IsFirst OR ( Level < Levels - 1D );

   IF WriteInfo THEN
      FOR i := 1 TO SHORT( Level ) DO
         TermIO.WriteString( '  ' );
      END; (* FOR *)
      TermIO.WriteString( 'level: ' );
      TermIO.WriteLongInt( Level, 0 );
   END; (* IF *)


   (* Here would normally be the code which loads the next block of
    * data somewhere into memory. In the present example, we just skip
    * the bytes containing these data.
    *
    * In case we would load the data block from the file, we would call
    * the following procedure:

   IF NOT
      LoadOneRecord
         ( Stream, Sizes[SHORT(Level)], MemorySize, WhereToStore )
   THEN
      Alert.IOError( '7-Error' );
      RETURN FALSE;
   END;

   (* If byte-swapping is required, we would now have to swap the bytes
      of all fields in the loaded record!
    *)

   IF NeedsByteSwap THEN ( go and swap the record fields ... ) END;

    * End of code we would call.
    *)


   (* Increase the file pointer by "Sizes[Level]" bytes and set the
    * file position just beyond the next data block:
    *)

   INC( Position, Sizes[SHORT(Level)] );

   IF NOT
      IOBytes.SetPosition( Stream, IOFiles.FromStart, Position )
   THEN
      Alert.IOError( '8-Error' );
      RETURN FALSE;
   END;


   (* The next 4 bytes contain the number of children of the present
    * level.
    *)

   Size := SIZE( INT32 );

IF NOT IOBytes.Read( Stream, Size, ADR(Count) ) THEN
      Alert.IOError( '9-Error' );
```

```
      RETURN FALSE;
   END;

   (* The file pointer increased by 4 bytes: *)
   INC( Position, 4 );

   (* And we swap the bytes, if needed: *)

   IF NeedsByteSwap THEN Count := SwappedInt32( Count ); END;

   IF WriteInfo THEN
      TermIO.WriteString( ';  children: ' );
      TermIO.WriteLongInt( Count, 0 );
      TermIO.WriteString( ';  file offset: ' );
      TermIO.WriteLongInt( Position, 0 );
      TermIO.WriteLn;
   END; (* IF *)

   (* Now, we can proceed to load all the children of the present
    * level, if there are any:
    *)

   INC( Level );

   Children := 0D;

   IF Level < Levels THEN

      WHILE Children < Count DO

         IF NOT
            LoadOneLevel(
               Stream,
               Sizes,
               Levels,
               NeedsByteSwap,
               Level,
               Children = 0D,
               Position )
         THEN
            RETURN FALSE;
         END; (* IF *)

         INC( Children );

      END (* WHILE *);

   END (* IF *);

   RETURN TRUE;

END LoadOneLevel;

(*
 * LoadTree
 *
 *    Scans a complete Tree.
 *    All "TermIO" statements are for demonstration purpose only.
 *    The variables are
```

```
 *           Stream        : the file handle to the open file
 *           Sizes         : the array returns the level sizes in
 *                           the Tree on disk.
 *           Levels        : the number of levels in the tree
 *           NeedsByteSwap : the flag telling, whether byte-swapping
 *                           is needed
 *
 *    The procedure returns TRUE, if it encountered no errors.
 *)

PROCEDURE LoadTree(
            VAR Stream         : IOFiles.FileHandleType;
            VAR Sizes          : ARRAY OF INT32;
            VAR Levels         : LONGINT;
            VAR NeedsByteSwap  : BOOLEAN )
            : BOOLEAN;
VAR
   Value     : INT32;
   Position  : LONGINT;
   Size      : LONGINT;
   i         : INTEGER;
   Success   : BOOLEAN;

BEGIN

   (* We start at the beginning of the file. We keep the variable
    * "Position" containing the actual position in the file.
    *)

   Position := 0D;

   (* The first 4 bytes should contain the "MagicNumber", see above.
    * a variable of type INT32 is a 32-bit long, signed word.
    *)

   Size := SIZE( INT32 );

   IF NOT IOBytes.Read( Stream, Size, ADR(Value) ) THEN
      Alert.IOError( '2-Error' );
      RETURN FALSE;
   END;

   IF Value = MagicNumber THEN
      NeedsByteSwap := FALSE;
   ELSIF Value = SwappedMagicNumber THEN
      NeedsByteSwap := TRUE;
   ELSE
      Alert.OK( '3-Error: File does not start with "Tree" !' );
      RETURN FALSE;
   END; (* IF *)

   (* The file pointer increased by 4 bytes: *)
   INC( Position, 4 );

   (* Next we load the number of levels in the Tree, which is stored
    * in the next 4 bytes (at offset 4):
    *)

   Size := SIZE( INT32 );
```

```
IF NOT IOBytes.Read( Stream, Size, ADR(Levels) ) THEN
   Alert.IOError( '4-Error' );
   RETURN FALSE;
END;

(* The file pointer increased by 4 bytes: *)
INC( Position, 4 );

(* If the file originates from a platform with opposite byte ordering,
 * then we have to swap the bytes:
 *)

IF NeedsByteSwap THEN Levels := SwappedInt32( Levels ); END;

TermIO.WriteString( '  -> levels: ' );
TermIO.WriteLongInt( Levels, 0 );

(* The next bytes contain the sizes of all levels. Thus, there is
 * one 4-byte variable for each level, totaling in "Levels" times
 * 4 bytes.
 *
 * First, we check, if the array "Sizes" passed to this procedure
 * is large enough to contain all level sizes:
 *)

IF ( Levels <= 0D ) OR ( Levels > LONG(HIGH(Sizes)+1) ) THEN
   Alert.OK( '5-Error: number of level either <= 0 or too large!' );
   RETURN FALSE;
END (* IF *);

(* Next, we load the "Level Size": *)

Size := Levels * LONG( SIZE( INT32 ) );

IF NOT IOBytes.Read( Stream, Size, ADR(Sizes) ) THEN
   Alert.IOError( '6-Error' );
   RETURN FALSE;
END;

(* The file pointer increased by "Size" bytes: *)
INC( Position, Size );

(* And we swap the bytes, if needed: *)

IF NeedsByteSwap THEN
   FOR i := 0 TO SHORT( Levels - 1D ) DO
      Sizes[i] := SwappedInt32( Sizes[i] );
   END; (* FOR *)
END; (* IF *)

TermIO.WriteString( ';  sizes: ' );
FOR i := 0 TO SHORT( Levels - 1D ) DO
   TermIO.WriteLongInt( Sizes[i], 0 );
   IF i < SHORT( Levels - 1D ) THEN
      TermIO.WriteString( ', ' );
   END; (* IF *)
END; (* FOR *)
```

```
   TermIO.WriteString( ';  swap: ' );
   TermIO.WriteBoolean( NeedsByteSwap );
   TermIO.WriteLn;

   (* Now, the tree data follow.
    * We can load them by a recursive procedure:
    *)

   Success :=
      LoadOneLevel(
         Stream,
         Sizes,
         Levels,
         NeedsByteSwap,
         0D,
         TRUE,
         Position );

   IF Success THEN
      TermIO.WriteString( 'total file length: ' );
      TermIO.WriteLongInt( Position, 0 );
   END; (* IF *)

   TermIO.WriteLn;
   TermIO.WriteLn;

   RETURN Success;

END LoadTree;

VAR
   FileName       : IOFiles.FileNameType;
   Path           : IOFiles.FileNameType;
   Stream         : IOFiles.FileHandleType;
   Sizes          : ARRAY[0..9] OF INT32;
   Levels         : LONGINT;
   NeedsByteSwap  : BOOLEAN;
   Success        : BOOLEAN;
   Dummy          : BOOLEAN;

BEGIN

   (* Get a filename of a tree file to load: *)

   FileName[0]    := 0C;
   Path           := 0C;

   IF NOT
      FileSelect.Select(
         FileName,
         Path,
         '*.*',
         FileSelect.ExistingFile,
         'Select the TREE file to scan:' )
   THEN
      RETURN;
   END; (* IF *)

   Strings.Insert( FileName, Path, 0 );
```

```
   TermIO.DoBuffer := TRUE;
   TermIO.WriteLn;
   TermIO.WriteLine( FileName );

   (* Open the file : *)

   IF NOT IOBytes.Open( FileName, IOFiles.Read, Stream ) THEN
      Alert.IOError( '1-Error' );
      RETURN;
   END;

   (* Now, load the "Tree" : *)

   Success := LoadTree( Stream, Sizes, Levels, NeedsByteSwap );

   (* And, finally, we are done and can close the file. *)

   Dummy   := IOBytes.Close( Stream );

END FileFormat.
```

Appendix V: Lookup Tables

If you are using a *"Telegraphing"* amplifier like the *Axoclamp 2A*, *PULSE* reads all possible gain settings from the so called *"I-Gain"* table and all filter settings from the *"Filter Bandwidth"* table. These tables are ASCII text files located inside the folder **Pulse\LookupTables** by default, with each line representing a pair of *minimal voltage* the amplifier sends to the program plus the *actual value* of the corresponding amplifier gain or filter setting. For example the following two lines in an *I-Gain* lookup table

```
6.3 500.0
5.8 200.0
```

will be interpreted the following way: a voltage **higher** than *6.3 V* corresponds to the setting *500 mV/pA*, a value **higher** than *5.8 V* (and lower 6.3 V) is *200 mV/pA*. For more details please refer to *Chapter Configuration*.

I-Gain Lookup Tables

EPC7 Amplifier (default table):

File IGainTable_EPC7

```
 9.5      0.5
 7.0      1.0   ; i.e. 1 mV/pA (milli-volts per pico-amperes)
 5.0      2.0
 3.0      5.0
 1.0     10.0
-1.0     20.0
-3.0     50.0
-5.1    100.0
-7.0    200.0
-9.5    500.0
-12.0  1000.0
```

Axon-200 Amplifier:

File IGainTable_Axon 200

```
6.3   500.0   ; i.e. 500 mV/pA (milli-volts per pico-amperes)
5.8   200.0
5.3   100.0
4.8    50.0
4.3    20.0
3.8    10.0
```

```
  3.3     5.0
  2.8     2.0
  2.3     1.0
  1.8     0.5
  1.3     0.2
  0.8     0.1
-10.0     0.05
```

Dagan 3900A Amplifier:

File IGainTable_Degan 3900A

```
3.0   500   ; i.e. 500 mV/pA (milli-volts per pico-amperes)
2.6   100
2.2   50
1.8   20
1.4   10
1.0   5
0.6   2
0.2   1
```

Warner PC-501A Amplifier:

File IGainTable_Warner PC-501A

```
4.1   1000     ; switch position 100, 10 GΩ, in mV/pA
3.9   500      ; switch position  50, 10 GΩ
3.7   200      ; switch position  20, 10 GΩ
3.5   100      ; switch position  10, 10 GΩ
3.3   50       ; switch position   5, 10 GΩ
3.1   20       ; switch position   2, 10 GΩ
2.9   10       ; switch position   1, 10 GΩ
2.7   100      ; switch position 100, 1 GΩ
2.5   50       ; switch position  50, 1 GΩ
2.3   20       ; switch position  20, 1 GΩ
2.1   10       ; switch position  10, 1 GΩ
1.9   5        ; switch position   5, 1 GΩ
1.7   2        ; switch position   2, 1 GΩ
1.5   1        ; switch position   1, 1 GΩ
1.3   10       ; switch position 100, 100 MΩ
1.1   5        ; switch position  50, 100 MΩ
0.9   2        ; switch position  20, 100 MΩ
0.7   1        ; switch position  10, 100 MΩ
0.5   0.5      ; switch position   5, 100 MΩ
0.3   0.2      ; switch position,  2, 100 MΩ
0.1   0.1      ; switch position   1, 100 MΩ
```

I-Gain (V-Clamp) Lookup Tables

Default Table:

```
  7.3    500.0   ; i.e. 500 mV/nA
  6.3     50.0
  5.4      5.0
  4.5    200.0
  3.6     20.0
  2.7      2.0
  1.9    100.0
  0.9     10.0
  0.0      1.0
 -0.9     50.0
 -1.9      5.0
 -2.7      0.5
 -3.5     20.0
 -4.5      2.0
 -5.4      0.2
 -6.2     10.0
 -7.2      1.0
-10.3      0.1
```

TEC-5 Amplifier:

File IGainVClampTable_TEC-05

```
  6.5    10.0   ; i.e. 10 mV/nA (milli-volts per nano-amperes)
  5.5     5.0
  4.5     2.0
  3.5     1.0
  2.5     0.5
  1.5     0.2
  0.5     0.125
 -1.0     0.1
```

TEC-10 Amplifier:

File IGainVClampTable_TEC-10

```
  4.5    10.0  ; i.e. 10 mV/nA (milli-volts per nano-amperes)
  3.5     5.0  ; this lookup table is for the "I-Gain, V-Clamp"
                 (not "I-Gain")
  2.5     1.0
  1.5     0.5
  0.5     0.2
-10.0     0.1  ; this is the "else" case
```

Filter Bandwidth Lookup Tables

Axon-200 Amplifier:

File BandwidthTable_Axon 200

```
  9.0   50000.0   ; in Hz => 50 kHz
```

```
  7.0   10000.0
  5.0    5000.0
  3.0    2000.0
-10.0    1000.0
```

Axon-1D Amplifier:

File BandwidthTable_Axon1D

```
  4.6  100000.0   ; in Hz
  4.2   50000.0
  3.8   20000.0
  3.4   10000.0
  3.0    5000.0
  2.6    2000.0
  2.2    1000.0
  1.8     500.0
  1.4     200.0
  1.0     100.0
  0.6      50.0
-10.0      20.0
```

Dagan 3900A Amplifier:

File BandwidthTable_Dagan 3900A

```
  4.6   100000   ; in Hz, Dagan Integrating Patch Clamp 3900A
  4.2    50000
  3.8    20000
  3.4    10000
  3.0     5000
  2.6     2000
  2.2     1000
  1.8      500
  1.4      200
  1.0      100
  0.6       50
  0.2       20
```

Aux-Gain Lookup Tables

The following *Aux-Gain* lookup table would allow to read from two amplifiers at the same time, e.g. to measure trans-synaptic currents. The current output of the first amplifier would be read through the normal *Current In* AD-channel as defined in the *Configuration* window. The current output of the second channel would be read through an AD-channel which is neither the *Current In* nor the *Voltage In* AD-channel. Please note, that the values here are the absolute conversion factors, i.e., volts per amperes! The following example lookup table (filename **AuxGainTable**) assumes an *Axon-200* amplifier:

```
   6.3   500.0E9 ; i.e. 500 V/V => mV/pA (milli-volts per pA)
   5.8   200.0E9
   5.3   100.0E9
   4.8    50.0E9
   4.3    20.0E9
   3.8    10.0E9
   3.3     5.0E9
   2.8     2.0E9
   2.3     1.0E9
   1.8     0.5E9
   1.3     0.2E9
   0.8     0.1E9
 -10.0     0.05E
```

Appendix VI: Controlling *PULSE*

Controlling *PULSE* from another Program

PULSE can be controlled from another program by a simple "batch file control" protocol. This "batch file control" protocol is simple, fast, and platform independent. The new control protocol allows to control the EPC9 over a network, even one with different platforms, such as Windows, OS/2, MacOS, or workstations. Thus, it is now possible to control the EPC9 from computers running a multitasking operating system which can create and read shared files (e.g., Windows 95, Windows NT, MacOS, etc.).

Controlling *PULSE* from another program is possible by communicating via two ASCII-files. The user writes the commands to one file (the "command" file) and *PULSE* 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. *PULSE* will access that file with read and shared permission only. The reverse is used on the second file, the "response" file: *PULSE* will have write and sharing permission, and the user program read permission only (plus sharing permission, of course).

The first line in the "command" file must contain one positive number (as ASCII, e.g., "+1234"). This "command index" is interpreted by *PULSE* as follows:

- If this number is zero or negative, *PULSE* does not execute the commands in the "command" file.
- If the number is larger than zero, *PULSE* will execute the instructions immediately. *PULSE* will write that number to the "response" file to flag execution once all commands have been executed.
- To prevent *PULSE* to execute the instructions more than once, *PULSE* 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.
- *PULSE* writes the responses to the "response" file. In the first line of that file, *PULSE* 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**”, the one of the “response” file “**E9Batch.Out**”. Both files will be inside the default folder of *PULSE*, usually the folder “*PULSE*”.
- A text string must be set within double quotes, when it contains non-alphanumeric characters (e.g. commas, colons, blank spaces, etc.). A filename with path should always be within double quotes!

Thus, communication would proceed as follows:

1. The user program is started first: It has to create a file in the “*PULSE*” folder named “E9Batch.In”. It has to keep this file open with “write” and “shared” access permission.

2. Then *PULSE* is started: Enable the “Enable Batch Control” option. *PULSE* will open the file “E9Batch.In” with “read” and “shared” access permission. Also, *PULSE* will now create the "E9Batch.Out" file with “write” and “shared” access permission.

3. Now, *PULSE* will immediately execute the commands in the command file, provided that the “command index” is larger than zero. *PULSE* writes the “command index” and eventually any error and requested answer to the “response” file.

4. Next, the user switches back to the user program.

5. Any time the user program writes to the “command” file, *PULSE* will scan the command file and execute the commands, if the “command index” changed.
   Windows 95 and Windows NT: On computers running Windows 95 or Windows NT, *PULSE* will immediately execute the commands.
   MacOS: On computers running MacOS, *PULSE* will immediately execute the commands, if it is the front application. If *PULSE* is not the front application, then *PULSE* will get active when the front application calls the “Toolbox” routine “GetNextEvent” or “WaitNextEvent”.

6. The user program can now read the “response” file. The first line should mirror the “command index”. Subsequent text lines may contain responses and error messages (see list below). There are the following possibilities:
   - There is no further line in the response file. This means, that no error occurred during execution and that no response was requested.
   - There is additional text in the file. The user can easily recognize error messages, because the first character in an error message is a lower case letter. All other responses start with an upper case letter.

7. When the user program wants to issue new commands, it writes a new "command index" and the new commands to the "command" file, then continues with step 5.

Here is an example of such a command file:

```
1. line: "1234"
2. line: "SetVHold -0.080"
3. line: "GetCurrentRange"
4. line: "MakeAnError"
```

And this would be the content of the "response" file:

```
1. line: "1234"
2. line: "CurrentGain 1.000E+09"
3. line: "error_not_found"
```

Error Messages

Errors begin with lower case letters:

```
'error_syntax':         parameter missing or misspelled
'error_range':          parameter is out of allowed range
'error_not_found':      command is unknown
'error_ioerror':        error during an I/O-operation
'error_unknown':        an unidentified error occurred
```

Implemented Commands and Messages

Messages send by *PULSE*. They do not contain a response string.

```
Started                 PULSE started communication link
Terminated              PULSE terminated communication
SeriesStart [gr se]     PULSE started acquiring a Series
SeriesEnd [gr se]       PULSE finished acquiring a Series
                              [gr se] is the group and series
                              indexes of the series. The 2
                              indexes are replaced by the string
                              "NIL", if the acquisition was aborted.
SweepStart [gr se sw]   PULSE started acquiring a Sweep
SweepEnd [gr se sw]     PULSE finished acquiring a Sweep
                              [gr se sw] is the group, series, and
                              sweep indexes of the sweep. The 3
                              indexes are replaced by the string
                              "NIL", if the acquisition was aborted.
```

Operations without parameters. They do not return a response.

```
AutoCFast
AutoCSlow
AutoGLeak
AutoVpOffset
```

"Get" operations: They have no parameters. The response always repeats the command string without the "Get" sub-string.

```
GetCCFastSpeed      boolean: 0 or 1
GetCCIHold          real:    current in amperes
GetCCRange          boolean: 0 or 1
GetCCStimScale      real
GetCFast1           real:    capacitance in farad
GetCFast2           real:    capacitance in farad
GetCFastError       boolean: 0 or 1
GetCFastTau         real:    tau in seconds
GetClipping         boolean: 0 or 1
GetComment          quoted string
GetCSlow            real:    capacitance in farad
GetCSlowError       boolean: 0 or 1
GetCSlowRange       integer: 0,1,2,3 -> off,30,100,1000pF range
GetCurrentGain      get gain in Ohms (i.e., V per A)
GetE9Board          integer: 1,2,3
GetE9SerialNo       string
GetF2Butterworth    boolean: 0 or 1
GetF2Frequency      real:    bandwidth in hertz
GetFilter1          integer: 0,1,2,3 -> 100,30,10kHz,HQ30kHz
GetGentleCCSwitch   boolean: 0 or 1
GetGLeak            real:    in siemens (i.e., 1/ohms)
GetGSeries          real:    in siemens (i.e., 1/ohms)
GetIpipette         real:    current in amperes
GetLastVHold        real:    potential in volts
GetMode             0 to 5,  corresponding to InOut, OnCell, OutOut,
                             WholeCell, PCClamp, VClamp
GetOnline1          2 reals: X- and Y-value
GetOnline2          2 reals: X- and Y-value
GetRmembrane        real:    resistance in Ohm
GetRsFraction       real:    0.0 to 0.95
GetRsMode           integer: 0,1,2,3 -> off,100,10,2µs
GetRsValue          real:    in ohms
GetStimFilterOn     boolean: 0 or 1
GetSweepName        3 integers: group series sweep
GetSweepStart       real:    seconds since midnight
GetTime             real:    seconds since midnight
GetVCStimScale      real
GetVersion          string
GetVHold            real:    -1.0 to +1.0 volt
GetVLiquidJunction  real:    -1.0 to +1.0 volt minus VHold
GetVmonitor         real:    in volts
GetVpOffset         real:    about +/- 200mV
```

"Set" operations: They have no or one parameter and do not return a response.

```
SetAdcChannel       integer: 0 to 12 -> FExt, Vmon, Imon1, Imon2,
                                        none, ADC0 .. ADC6
SetCCFastSpeed      boolean: 0 or 1
SetCCIHold          real:    current in amperes
SetCCRange          boolean: 0 or 1
SetCFast1           real:    capacitance in farad
SetCFast2           real:    capacitance in farad
SetCFastTau         real:    tau in seconds
SetComment          quoted string (comment in last acquired Series)
SetCSlow            real:    capacitance in farad
```

```
SetCSlowRange        integer: 0,1,2,3 -> off,30,100,1000pF range
SetCurrentGain       set gain in Ohms (i.e., V per A)
SetE9Board           integer: 1,2,3
SetExportMode        integer: 0,1,2 -> sweep, online, both
SetExportType        integer: 0 to 8 -> printer (vectors),
                                        printer (screen dump),
                                        LogBook, ASCII,
                                        WMF/PICT, IgorText,
                                        IgorLayout, IgorInfo,
                                        IgorBinary
SetF2Butterworth     boolean: 0 or 1
SetF2Frequency       real:    bandwidth in hertz
SetFilter1           integer: 0,1,2,3 -> 100,30,10kHz,HQ30kHz
SetGentleCCSwitch    boolean: 0 or 1
SetGLeak             real:    in siemens (i.e., 1/ohms)
SetGroupText         quoted string (comment in last acquired Group)
SetGSeries           real:    in siemens (i.e., 1/ohms)
SetLastVHold         no parameters
SetMode              0 to 5,  corresponding to InOut, OnCell, OutOut,
                              WholeCell, PCClamp, Vclamp
SetReferenceTarget   no parameter
         if PULSE is still acquiring, the command is ignored
         and the string "error_acquiring" is returned.
SetRestrictWindows   boolean: 0 or 1
                              if set will cause PULSE to restrict
                              selecting only the following windows:
                              amplifier, oscilloscope, parameters,
                              and online analysis.
SetRootText          quoted string
SetRsFraction        real:    0.0 to 0.95
SetRsMode            integer: 0,1,2,3 -> off,100,10,2µs
SetRsValue           real:    in ohms
SetSleep             real:    in seconds
SetStimFilterOn      boolean: 0 or 1
SetStimScale         real:    -10.0 to +10.0 (0.0 means off)
SetSubtract          integer: 0 to 3 -> none, buffer, ref.sweep,
                                        ref series
SetTestPulse         boolean: 0 or 1
SetVHold             real:    -1.0 to +1.0 volt
SetVLiquidJunction   real:    -1.0 to +1.0 volt minus VHold
SetVpOffset          real:    about +/- 200mV
SetWaitResume        integer: 0, 1 or 2
         If set to 1 (Sweep wait) or 2 (Series wait) causes PULSE to
         wait before and after acquiring a Sweep (1) or Series (2)
         until a "Resume" command is received.
         PULSE waits after having send the message
         "SweepStart [NIL | group series sweep]" or
         "SeriesStart [NIL | group series sweep]" and
         "SweepEnd [NIL | group series sweep]" or
         "SeriesEnd [NIL | group series sweep]".
         NIL is sent when the acquisition was aborted.
```

Miscellaneous Operations

```
acknowledged (re-synchronize command index)
   syntax: "acknowledged"
      no parameters
      no response

ADRead
   syntax: "ADRead AD-channel"
      AD-channel: integer 0 to 7
      returns a real (read AD-value in volt)

ADRelease
   syntax: "ADRelease"
       no parameters
       returns "ADReleased" if board could be released

ADReserve
   syntax: "ADReserve"
       no parameters
       returns "ADReserved" if board could be reserved
       returns "ADReleased" otherwise

Break
   syntax: "Break"
      no parameters
      no response

ClearClipping
   syntax: " ClearClipping "
      no parameters
      no response

CloseFile
   syntax: "CloseFile"
      no parameters
      response:
         if PULSE is still acquiring, the command is ignored
         and the string "error_acquiring" is returned.
         Otherwise, the data file is closed, and all files
         are flushed to disk.

DAWrite
   syntax: "DAWrite  AD-channel  Volts"
      DA-channel: integer, 0 to 3
      Volts:      real,    +/-10.24 volt
      no response

DigGet
   syntax: "DigGet"
      no parameters
      returns an integer: 0..65535
```

```
DigSet
   syntax: "DigSet  Mask  Word"
      Mask:       integer, 0 to 16383
      Word:       integer, 0 to 16383
      no response

DoExport
   syntax: "DoExportFull  Overwrite  Filename"
      Overwrite:  integer, 0 or 1
      Filename:   string, may include a path
      no response
         if PULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.

DoExportFull
   syntax: "DoExportFull  Overwrite  Filename"
      Overwrite:  integer, 0 or 1
      Filename:   string, may include a path
      no response
         if PULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.

DoZap
   syntax: "DoZap  Duration  Voltage"
      Duration:   real, in seconds
      Voltage:    real, in volts
      no response
         if PULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.

ExecMacro
   syntax: "ExecMacro  MacroIndex"
      MacroIndex: integer, 0 to 19
      no response

GetSeriesOnline
   syntax: "GetSeriesOnline"
      no parameters
      returns as many text lines as there are sweeps in the target
         series. Each line has 4 reals: X1, Y1, X2, Y2

LoadMacros
   syntax: "LoadMacros “FileName”"
      parameter: file name (within double quotes)
      response:
         It returns the error "error_ioerror", if the file could
         not be loaded.

MeasureNoise
   syntax: "MeasureNoise"
      no parameters
      returns one real

MeasureNoise2
   syntax: "MeasureNoise2"
      no parameters
      returns two reals: high-pass noise and low-pass noise
```

```
MoveTarget
   syntax: "MoveTarget  Direction  Moves"
      Direction:  integer, 0 to 5 -> left, right, up, down, HOME, END
      Moves:      integer, 1 to 5, optional, default = 1
      no response
         if PULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.

NewGroup
   syntax: "NewGroup"
      no parameters
      response:
         if PULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.
         Otherwise, a new Group is created in the Tree (if the
         last group is not empty).

NotebookClear
   syntax: "NotebookClear"
      no parameters
      no response

NotebookWrite
   syntax: "NotebookWrite “Text”"
      Text: the string (within double quotes) to write to the
         Notebook
      no response

OpenFile
   syntax: "OpenFile “FileName”"
      parameter: file name (within double quotes),
         with or without a path.
      response:
         if PULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.
         Otherwise, "CloseFile" is called to close all files, and
         the file with the given name is opened. The file is
         opened in "modify" mode, if it exists. Otherwise a new
         file is created.
         It returns the error "error_ioerror", if the file could
         not be opened.

Query
   syntax: "Query"
      no parameters
      returns:
         "Acquiring" -> when still acquiring sweeps
         "Disabled"  -> when the amplifier window is active, but
                        PULSE is not the foremost application and
                        "Enable Background" is disabled.
         "Waiting"   -> when "RestrictWindows" is set and the active
                        window of PULSE is NOT one of the fol-lowing:
                        - the amplifier window
                        - the oscilloscope window
                        - the online window
                        - the parameters window.
         "Ready"     -> otherwise.
```

```
Reset
   syntax: "Reset"
      no parameters
      no response

Resume
   syntax: "Resume"
      no parameters
      no response

SetOnline1           2 required and 1 optional integer
   syntax: "SetOnline1  mode  abscissa  math"
      mode:       integer (required): 0..24
      abscissa:   integer (required): 0..9
      math:       integer (optional)" 0..4
   no response
         if PULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.

SetOnline2           2 required and 1 optional integer
   syntax: "SetOnline2  mode  abscissa  math"
      mode:       integer (required): 0..24
      abscissa:   integer (required): 0..9
      math:       integer (optional)" 0..4
   no response
         if PULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.

SetTarget            3 optional integer
   syntax: "SetTarget  group-no  series-np  sweep-no"
      group-no:   integer (optional), 0 means “previous selection”
      series-no:  integer (optional), 0 means “previous selection”
      sweep-no:   integer (optional), 0 means “previous selection”
   no response
         if PULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.
   Examples:
      "SetTarget"          -> select the root
      "SetTarget 3"        -> select 3. group
      "SetTarget 3, 2"     -> select 2. series of 3. group
      "SetTarget 3, 2, 1"  -> select 1. sweep of 2. series of
                              3. group

ShowTarget
   syntax: "ShowTarget "
      no parameters
      no response
         if PULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.

Shutdown
   syntax: "Shutdown"
      no parameters
      response:
         if PULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.
         Otherwise, the string "Shutdown" is returned, and then
         PULSE shuts down.
```

```
SwitchToEpc9
SwitchToOnline
SwitchToOsci
SwitchToParams
   syntax: "SwitchToXXXX"
      no parameters
      no response

TargetInfo
   syntax: "TargetInfo "
      no parameters
      response:
         Returns 7 parameters:
            TotalPoints   : Total Points
            SampleInterval: Sample Interval
            DataFactor1   : Amperes per ADC-unit
            DataFactor2   : Volts   per ADC-unit
            Trace1        : Bytes from beginning of file
            LeakTrace     : Bytes from beginning of file or NIL
            Trace2        : Bytes from beginning of file or NIL

Terminate
   syntax: "Terminate"
      no parameters
      returns the string "Terminated"

UpdateFile
   syntax: "UpdateFile"
      no parameters
      response:
         if PULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.
         Otherwise, the all files are flushed to disk.

Key
   (only "dialog" keys, not menu keys)
   syntax: "Key  ASCII-code  Key-Code  Modifiers"

   Key-Codes:
      2 -> Normal character input

      cursor keys:
      3 -> KeyCursorUp
      4 -> KeyCursorDown
      5 -> KeyCursorLeft
      6 -> KeyCursorRight

      numeric keypad:
      7 -> 0 (zero) on numeric keypad
      8 -> 1 on numeric keypad
      9 -> 2 on numeric keypad
     10 -> 3 on numeric keypad
     11 -> 4 on numeric keypad
     12 -> 5 on numeric keypad
     13 -> 6 on numeric keypad
     14 -> 7 on numeric keypad
     15 -> 8 on numeric keypad
     16 -> 9 on numeric keypad
     17 -> Enter
```

```
  18 -> minus  on numeric keypad
  19 -> plus   on numeric keypad
  20 -> period on numeric keypad
  21 -> clear  on numeric keypad (above '7')
  22 -> '='    on numeric keypad (above '8')
  23 -> '/'    on numeric keypad (above '9')
  24 -> '*'    on numeric keypad

   function keys (F1 to F15):
  24 + function key number, e.g., 25 for F1

   key block above cursor keys:
  40 -> HOME
  41 -> END
  42 -> PageUp
  43 -> PageDown
  44 -> HELP
  45 -> Delete Left
  46 -> Delete Right

Modifiers:
  CommandKey -> 1
  ShiftKey   -> 2
  CapsKey    -> 4
  OptionKey  -> 8
  CtrKey     -> 16
```

Notes for Programmers

One has to diligently select when to open the message file. That file is created by *PULSE*. Therefore it cannot and should not be opened before *PULSE* 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 *PULSE* is located and to wait for the message file to appear in the *PULSE* directory. At that moment one can start *PULSE* and enable the option "Enable Batch Control" in the EPC9 drop-down menu. The message file will then be created and the user program can now open it for reading and sharing.

The first response *PULSE* writes to the message file is "started". Thereafter the user can proceed to send commands to *PULSE* 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 *PULSE* 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, *PULSE* as well as the user program, run concurrently. Thus, *PULSE* may attempt to read from the message file while the user program is writing to it!

2. Write the remaining of the first text line. The first text line must be the signature number. The sample code below writes the negative value of the signature to achieve:
   - a negative sign is placed in the first byte of the file;
   - the signature value is written; and
   - the final, positive signature can be obtained by replacing the negative sign with a plus ("+").
3. Write to following text lines the required instructions, one instruction per text line.
4. Finally, replace the negative sign in the first byte of the file with a plus ("+"). This will signal to *PULSE* 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. *PULSE* 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.

Controlling *PULSE* from E9Screen

PULSE can be controlled from *E9Screen* by the communication protocol described here. Thus, *E9Screen* can be used to understand, how the protocol functions, since all communication commands are listed in the Notebooks of both programs. To start *E9Screen* in the mode required for that mode, proceed as follows:

- Place a file named “E9Batch.On” in the directory where *E9Screen* resides.
- Launch *E9Screen*.
- *E9Screen* will display the file selector at some point during the start-up phase asking to select the folder in which *Pulse+PulseFit* resides.
- When *E9Screen* finished booting, launch *Pulse+PulseFit*.
- Activate the options "Enable Batch Control" and "Enable Background" in the "EPC9" drop-down menu of *Pulse+PulseFit*.

E9Screen will no longer perform any acquisition and will have the oscilloscope display closed. When an EPC9 setting is changed in *E9Screen* (e.g. changing the EPC9

gain), it will be send to *PULSE*. Thus, *PULSE* will "mirror" the EPC9 settings of *E9Screen*..

E9Screen will display an additional item in the middle of where the oscilloscope usually resides. This item is available only in this special mode. It allows to enter an arbitrary text string which upon pressing the ENTER or RETURN key is then send to *PULSE*.

Sample Programs

The listing below is an excerpt from the *E9Screen* code implementing the communication protocol used to control *PULSE* as described above. It demonstrates the functions needed to write a program to control the EPC9 through the *PULSE* program. The code is written in Modula-2. The code can be easily 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    = 'EPC9In.EPC';
   MessageName    = 'EPC9Out.EPC';

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 PULSE 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 terminationg 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 explana-
      tion.
 *)

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 PULSE 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;
VAR
   Temp  : IOFiles.FileNameType;
   Path  : IOFiles.FileNameType;
BEGIN

   IF CommandFile.IsOpen THEN
      Alert.String( 'Command file already open! ' );
      RETURN FALSE;
   END; (* IF *)

   Strings.Assign( Temp, 'PULSE.exe' );
   Strings.Clear( Path );

   IF NOT
      FileSelect.Select(
         Temp,
         Path,
         '*.*',
         FileSelect.ExistingFile + FileSelect.ChangeDirectory,
         'Select any file in target PULSE folder:' )
   THEN
      RETURN FALSE;
   END; (* IF *)

   IF NOT
      IOText.Open(
         CommandName,
         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.
```

Appendix VII: Telltale Files

Telltale files are another way to customize *PULSE+PULSEFIT*. Usually *Telltale* files are small files that only contain one line of text or nothing at all (i.e. they work as a switch). To work, they have to reside inside the same folder as *PULSE+PULSFIT*..

Telltale File	Description
AqAvMax.On	Overrules the default number of samples averaged when compressing the data during slow acquisition, i.e., acquisition slower than 65 ms per sample.
AqLimit.On	Overrules the default time limit from which on a sweep is acquired while continuously displaying the acquired part of it (if conditions allow that mode). The ASCII-number in that file must be in seconds, e.g. “0.5”
DemoProg.On	Suppresses the messages about "protection key not found" and "non existent file paths".
E9Batch.In	Input file used by E9Screen for "batch"-file communication. This file must exist in the E9Screen folder before E9Screen starts batch-file communication.
E9Batch.On	Active in E9Screen when “EPC9-Batch Mode” is active. In that case, E9Screen will send the selected commands to another E9Screen program in order to control it.
E9Batch.Out	Output file created by E9Screen for "batch"-file communication. This file is deleted when E9Screen terminates.
EPC9out.EPC	File to which E9Screen writes the actual mode and Vhold, when "Background Enable" is selected. This may be used, e.g., by X-Chart when it runs as a second application to read the EPC9 status (see EPC9 manual).
Pulse.On	Automatically start PULSE without asking whether to run PULSE or PULSEFIT.

PulseFit.On	Automatically start PULSEFIT without asking whether to run PULSE or PULSEFIT.
WinTiny.On	Causes most windows to be only 2/3 of its normal size. If it contains a scaling factor (as ASCII-text!), the scaling factor is used instead of the default value of 0.667. “0.9” is a good value for most monitors, it will make all dialog windows 10% smaller. That way one could have PULSE running on screens smaller than 15".

Index

D

E

N

O

P

R

S

T

U

V

W