CAMBRIDGE ELECTRONIC DESIGN LIMITED

eNEWSLETTER

#5
August 2020

Contents

Welcome

Thank you for downloading our August newsletter. We are now over halfway through summer and those sunny days keep on coming! For those of you working away in the lab, we have some more update news for you: the latest update to Spike2 is now available, download v10.06 from our website. With 20 new features and several fixes we hope you will continue to get the best out of Spike2.

With the new academic year approaching, we have also begun a new segment called Scripters Corner. With this segment we will walk you through the basics of script writing, to introduce scripts to beginners and help you on your way to writing your own. Feel free to pass this newsletter to students and other users of Spike2 or Signal, and we hope the segment proves useful for them. As always, feel free to email in comments, questions, and suggestions for topics to Marjorie@ced.co.uk.

Training Days

Due to the ongoing COVID-19 outbreaks, all current training events have been put on hold. We do however offer remote training sessions by video call either one-to-one or with groups.

Join us and learn how to make the best use of Spike2 and Signal to save hours of repetitive analysis. Our remote sessions are free to arrange and are suitable for both existing and prospective users of our data acquisition and analysis systems. If you would like to schedule a session, please get in touch.

If you are interested in hosting a training event in your local area once social distancing measures have been eased, please get in touch: Marjorie@ced.co.uk.

If you see these buttons in our newsletters, it means a file or script relating to the section is available to download:

Latest versions of Spike2 and Signal

Spike2	Released	Signal	Released
Version 10.06	07/2020	Version 7.05a	02/2020
Version 9.11	05/2020	Version 6.05b	10/2019
Version 8.19a	11/2019	Version 5.12a	02/2018
Demo	03/2020	Demo	02/2020

Back to contents

Future meetings and events

Neuroscience 2020
Washington, DC,
USA
October 24th – 28th 2020

Our meeting calendar is updated each time we receive news of cancellation or postponement due to the COVID-19 pandemic. The full calendar is located on our website.

Back to contents

Script Spotlight

With the release of Spike2 update 10.06, the script command `MMFrame()` has been extended to cover MP4 format files (.mp4). This command returns a list of real frame times in the current multimedia view if this is supported by the multimedia file format. Before v10.06 this was supported only for AVI format files (.avi). This command is particularly useful for returning the times of frames within a range, and from v10.06 it is possible to return only key frames or non-key frames for MP4 files.

This script command was previously used in the Activity from Video.s2s script mentioned in a recent issue of our newsletter. In that script, `MMFrame()` was used to calculate the frame rate across a time range for AVI files, however the same option was not available for MP4 files at that time so another scripting route was needed. Now with the extension to the command it is possible to prepare scripts using the same function for both AVI and MP4 files. Both the previous newsletter and the script are available on our website should they interest you.

We would strongly recommend using the MP4 file format over AVI if possible; as AVI is an older format, it lacks features found in modern formats. The encoders used with AVI do not offer as good compression, and it is also less-well supported by non-Windows video players.

Back to contents

I have seen virtual channels being used in scripts available on your website, what else could I use them for?

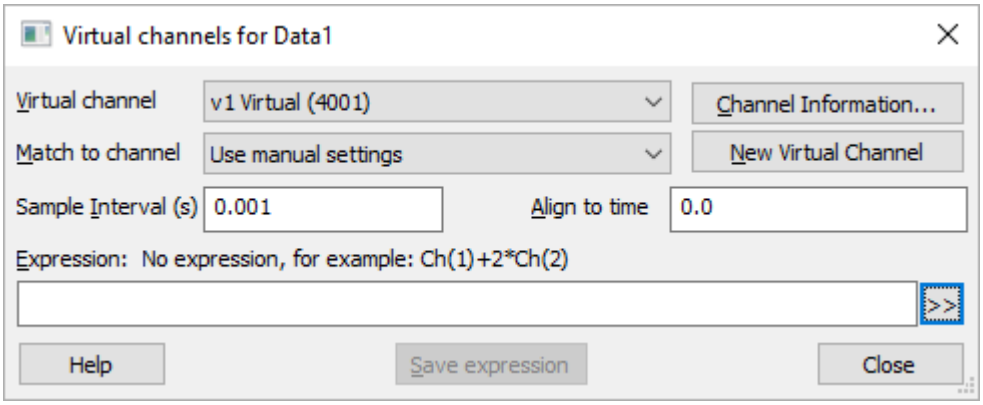
Virtual channels may be used for numerous things:

- Perform channel arithmetic (for example sums and differences of channels)
- Convert event channels into waveforms proportional to the event rate
- Linearise non-linear transducers
- Perform frequency analysis
- Generate waveforms
- Extract power in a frequency band
- Resample channels to a user-defined frequency or to match another channel

Virtual channels are therefore a powerful feature with a huge range of applications. They operate by holding RealWave data derived by a user-supplied expression from waveform, event and RealWave channels and built-in function generators. Channel sample intervals and alignments are matched by cubic splining the source waveforms, linear or cubic interpolation of RealMark data and by smoothing event rates. No data is stored; the channels are calculated each time you use them. Although the calculations are performed efficiently, you may want to save frequently used virtual channels as memory or disk-based permanent channels. Virtual channels are saved to disk with the *Analysis* menu *Save Channel* command or deleted with the *Delete Channel* command.

Virtual channels can also be created as part of a script. We have used of the script `VirtualChan()` function in our Spike2 script section, where we use virtual channels to generate waveforms to play out of the 1401 DACs.

To get started with virtual channels, create one yourself from the *Analysis* menu -> *Virtual channels* -> *Create new channel* command:



Virtual channel – Use this field to select a channel when you have more than one virtual channel.

Channel Information – This button is a short cut to the *View* menu *Channel Information* dialog; use that dialog to set the channel title and units.

New Virtual Channel – Click this button to create a new virtual channel.

Match to channel – Select an existing waveform-based channel (not another virtual channel) to copy the sample interval and data alignment settings from. Alternatively, select *Use manual settings* and type in the interval and alignment yourself.

Sample Interval – This field holds the sample interval between data points in the virtual channel in seconds and can be edited if you select *Use manual settings*. This field accepts expressions; for example, to set 27 Hz you can type 1/27.

Align to time – This field sets the time of a data point in the virtual channel, again it can be edited if you select *Use manual settings*. The time of any point and the sample interval completely defines all the sample times for the channel.

Expressions to generate the virtual channel result/data can be entered by hand or built by clicking the >> button (highlighted above) and selecting the following options from the context menu:

Waveform from channel – This option allows you to create data by copying from an existing waveform channel or generating a waveform based on instantaneous frequency, kernels of events and RealMark data items.

Spectral Functions – These commands create waveforms based on the spectral content of a waveform. This is a larger topic which we will cover in a subsequent newsletter.

Generate waveform – Create a waveform independently of any channel data using functions such as sinusoid, envelope, or triangle wave functions. Channels created using this method can then be used for waveform output (see our Spike2 Script section for a full example).

Rectify, Abs, Min and Max – This option inserts commands that can rectify, half-wave rectify and limit values.

Mathematical functions – Use this to insert mathematical functions including square root, sin, cos etc. A list of standard mathematical operators (+,-,x, / etc.) and comparison operators are also available from the context menu. This could be used to add two channels together, as seen in the example in the dialog.

Previously used expressions can be saved to the context menu using the *Save expression* button in the main virtual channel dialog. Full details of all the virtual channel commands can be found in the Spike2 online help, accessed by pressing F1.

Back to contents

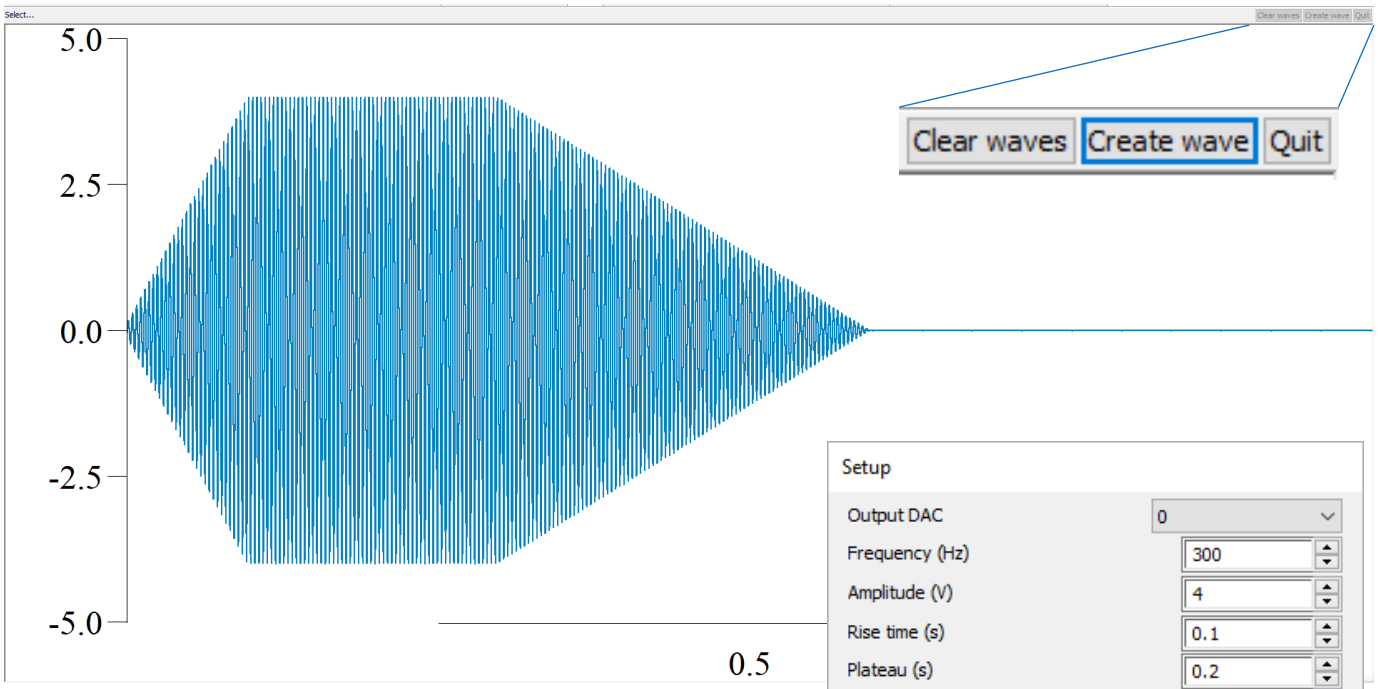
Scripts: Spike2

The `Virtualchan()` script command creates and controls virtual channels in the current time view. This command has the same functionality as the *Virtual Channel* dialog, described in the earlier Spike2 section. We can use this command to create or modify an existing channel, or to read back information about the virtual channel. This script command is the core feature of our attached script, Create waveform for output.s2s.

This script generates tones to play out of your 1401 DAC outputs. The script first attempts to open a 1401 and detects the number of available 1401 DAC outputs (8 are assumed if no 1401 is found). The available toolbar commands are:

- *Clear waves* – Remove all output waves from the sampling configuration
- *Create waves* – Open a new data file and dialog to create a new sinusoidal tone
- *Quit* – Halts the script and closes associated views

The script creates a new virtual channel with: `virt%:=VirtualChan(0, "", 0, 1.0/SampleRate, 0);` and generates a sinusoidal waveform within an envelope based on your chosen parameters, thus creating your tone. As well as generating tones, the script generates random noise. We achieve this by creating a memory channel in the background, filling an array with random numbers using the `Rand()` script command, and importing the random numbers into the memory channel with `MemSetItem()`. The virtual channel creates a copy of this memory channel and applies the envelope with your set parameters.



When creating a new waveform, enter the frequency (Hz), amplitude (V), rise time (s), plateau (s), fall time (s), and align time (s) of the tone. To generate random noise instead of a sinusoidal waveform, use the Noise tick box. With this ticked, you cannot alter the frequency (Hz), but the other parameters may still be altered. Once you are happy with the new tone, enter the *Key* to associate to playing the waveform online, and the label of the new tone. Then click *Add to online*. Keys currently in use are displayed under *Used Areas.* Choosing a Key that is already in use overwrites the old waveform, with a warning message displayed when the Key is entered.

Back to contents

Signal

How do I set up Signal for patch clamp recording?

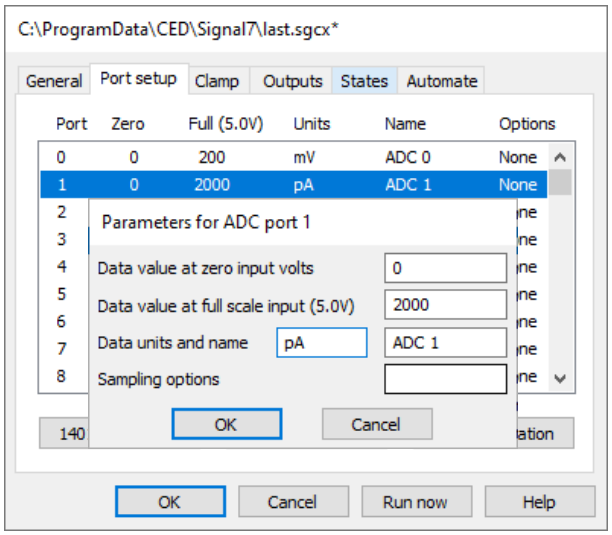
Setting up patch clamping is a large topic. Therefore, in this issue we discuss the relevant tabs of the sampling configuration and will delve deeper into the other aspects of patch clamping with Signal in subsequent newsletters.

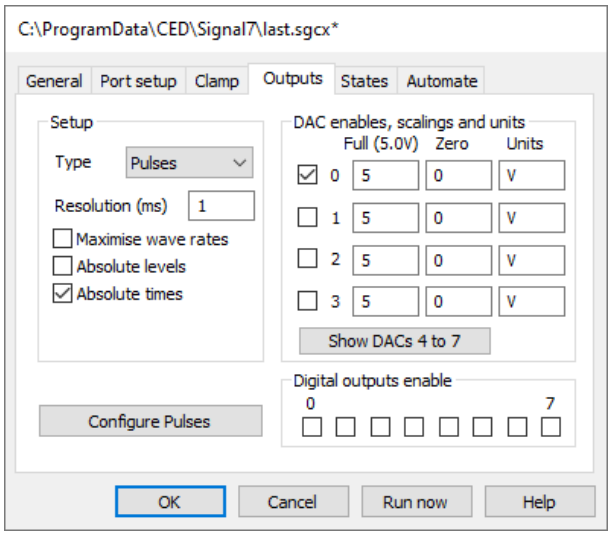
Signal incorporates several specialised features supporting whole-cell and single-channel clamping experiments - online holding potential control, resistance measurements and membrane analyses, dynamic clamping, leak subtraction and single-channel analysis. Signal automatically hides these features by default, they can be shown using the check box in the *Edit* menu > *Edit Preferences* dialog > *Clamp* tab.

DAC and ADC calibrations

We start defining a sampling configuration by determining what signals are going to be sampled and how the amplifier is controlled. For example, in voltage-clamp experiments the membrane potential imposed by the amplifier will be controlled by a signal generated by one of the 1401 DACs.

- Connect the outputs from the amplifier to 1401 ADC inputs and connect a DAC to the amplifier external control input.
- The *Ports* tab of the sampling configuration dialog is used to set the calibration and units for the ADCs connected to the amplifier outputs, double click a port to open the parameters dialog. The *Outputs* tab is used to set the calibration and units for the external control DAC.
- It is important these calibrations are correct otherwise your sampled data will not be correctly shown and the membrane analysis and holding potential controls will not operate correctly - see the user manual for your amplifier for details on this.



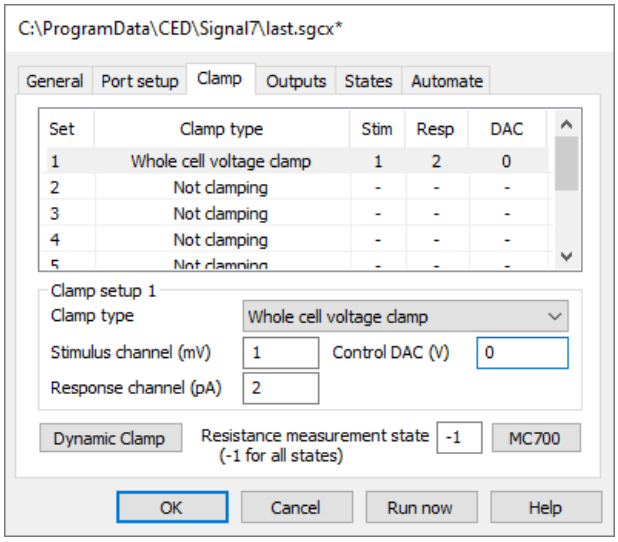


General setup

- Use the *General* tab to define the sampling rate to use, the ADC ports to sample-from and the duration of each frame of data.
- Use the *Outputs* tab to select pulse outputs. Text sequencer-controlled outputs may be used in clamping experiments for more detailed control if you are comfortable with these.
- If required, use the General tab to enable multiple frame states and set the multiple states mode to Dynamic outputs in the States tab.
    - Multiple frame states sampling in dynamic outputs mode is useful in clamping experiments as it allows you to have up to 256 separate sets of output stimuli in one sampling configuration and switch between them in various ways during the experiment. For a fuller explanation, see the help page *Sampling with extra states*, accessed by pressing F1.
- Generate sets of suitable stimuli using the DACs connected to the amplifier external control inputs. You do this using the *pulses configuration* dialog available by clicking *Configure Pulses* from the *Outputs* page of the sampling configuration. Configuring pulses has been covered in previous issues, but the *Pulses dialog* topic of the software help (F1) provides full detail on these.

Clamping sets

Once you have calibrated, you now must define your clamping sets in the Clamp tab of the sampling configuration. A clamping set is a pair of interlinked data file channels that hold the current and voltage data across a cell membrane, plus a control DAC that is used to control the membrane potential (for voltage clamp experiments) or current (for current clamp). It is necessary for the clamping sets to be defined in the sampling configuration so that the stimulus and response channels are known and the membrane analysis can find the correct data; this also allows Signal to check the units for these channels so that current and voltage values are scaled correctly to amps and volts. Defining the DAC used to control the membrane current or voltage allows Signal to manipulate the holding potential and stimulus pulses and again to check that the DAC units are correct for the clamping mode.



If you are using a supported auxiliary telegraph device, such as the MultiClamp 700, AxoClamp 900A, or EPC 800, Signal can automatically set up the clamping sets using information read back from the amplifiers. This ensures that all the clamping information plus the calibrations of relevant ports are always correct. See the specific help page (F1) under *Amplifier telegraphs* for more information.

And with all that you are ready to begin running a clamping experiment! As mentioned above we will be delving deeper into clamping with Signal in subsequent issues, but in the meantime further reading can be found in the *Sampling with clamp support* section of the software help, accessed by pressing F1.

Back to contents

Scripts: Signal

We have developed a script which automates the intra spike analysis for recordings of intracellular nerve impulses, and outputs the following statistics as a text file: spike amplitude (mV), threshold (mV), rise time (µs), half-width (mV), afterhyperpolarisation (AHP) (ms) and peak-AHP (ms). The script, IntraSpikeAnalysis.sgs is available from our website. The outputted text file can be saved to disk or its contents copied to a spreadsheet for further analysis.

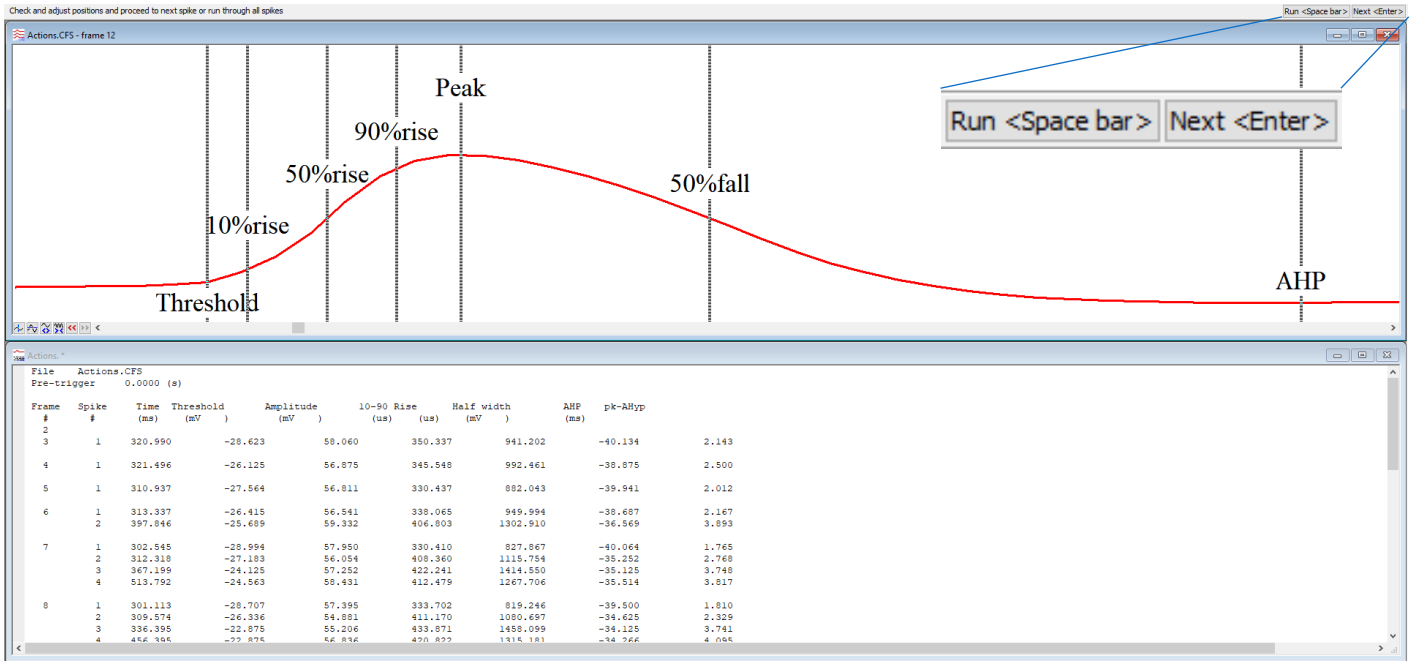
The script makes use of cursors marking features of your spikes, specifically using the `ChanSearch()`, `Cursor()` and `Hcursor()` functions. The script also creates a virtual channel copy of the data at a high sample rate and uses this for the analysis rather than the original data channel, thus avoiding any interpolation derived search errors. Virtual channels were added to Signal in version 4; consequently, you need at least version 4 to run this script. Simple arithmetic is performed with the points marked by cursors, which is then outputted to the text file.

To use the script, first load a data file and the script within Signal. An example data file, Actions.cfs is included with your user data folder (usually C:\Users\Username\Documents\Signal\Data) should you wish to experiment with the script first. If a suitable data file is not already open when running the script, a *Browse window* will open, prompting

you to open a data file to analyse. A prompt to navigate to a typical frame of data is then given. Drag the horizontal cursor to intersect the rise of all relevant spikes, click *OK* in the toolbar or press *Enter* when done.

A dialog is created for you to set the remaining parameters:

- The channel to analyse is shown but not selectable. It is the channel containing the horizontal cursor.
- The pre-trigger time is an offset relative to the start of the frame. Times to the peak of each nerve impulse are measured from this point. For example, this time could be the onset time of a stimulus that evokes a Spike train. Set the pre-trigger time to zero to record times relative to frame onset. The default value of this offset is remembered between runs.
- The threshold is the level change for detection of an action potential peak or after-hyperpolarisation trough. The default value when the dialog opens is 2% of the optimised Y-Range of the current frame. This value should work in most cases. However, if cursor searches fail consistently then you will need to perform the analysis with a different value.
- The range of frames to analyse may also be defined. The most recently used range becomes the default for the next run.
- The Check individual spikes checkbox is used to inspect each spike that was detected and check the cursor positions to ensure that the desired features have been detected correctly. Zoom in on the trace to see the spike in more detail. If you are not satisfied, you can drag cursors to their correct positions and press *Next* (hotkey *ENTER*) on the script toolbar to advance to the next spike. Alternatively, click on *Run* (hotkey *SPACE BAR*) to process all the remaining spikes without further interaction.



When your parameters are set, click *OK* to begin processing the data file. Results are displayed as a table in a text file as they are calculated. If the script encounters a search error, wherein the `ChanSearch()` command was unable to find a particular spike feature, the spike in question will be flagged up and displayed on screen with a new dialog. This allows you to reposition the cursors and either accept the new positions or reject the spike from the results entirely.

Back to contents

Scripters Corner – View Handles

With the new academic year approaching, we are starting a new segment aimed at first time scripters. In this segment we will break down the functions essential to script writing, for new scripters to gain a better understanding of how to write their own.

Each open window in Spike2 or Signal that can be manipulated by the script language is called a *View*. When a script performs a function like grabbing data or copying channels, it must know which view holds the data. We could require every script command that deals with a view to specify it, and some commands do this. However, we mostly deal with one view at a time, so we have the concept of the *Current View*. This is the view that is used by script commands that do not explicitly set one.

The current view is not the same as the view you may be currently using, these are independent. Only script actions can change the current view.

Views are identified by a *View Handle*. This is an integer number, greater than 0, that identifies a view. Script commands that create new views return the view handle of the view and make it the current view. View handles are assigned in ascending order. There is one exception; automatic sampling of a sequence of data files preserves the handle of the sampled file. There are never two views open with the same handle.

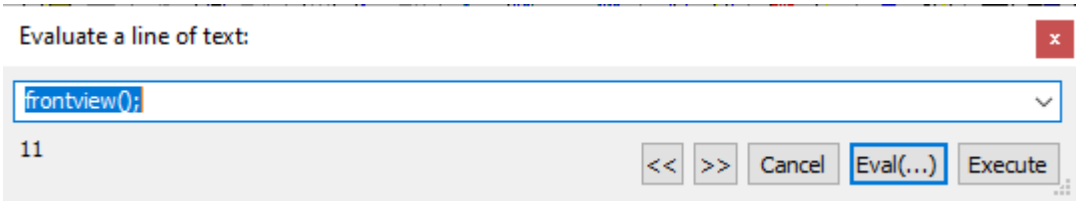
Apart from creating views, scripts set the current view with the `View(vh%)` command and the `FrontView(vh%)` command (which sets both the view with the input focus and the current view), where `vh%` is the view handle integer. It is worth understanding the difference between these two commands. When used without arguments `View()` returns the current view and `FrontView()` returns the view with the input focus (usually the view the user last clicked in). When used with a view handle as the argument, `View(vh%)` sets the current view and `FontView(vh%)` sets both the current view and the view with the input focus.

The `View()` command also has another mode of use:

```
View(vh%).ScriptCommand(…);
```

This saves the current view, swaps to a new current view, runs the ScriptCommand(), then swaps back to the saved current view.

To illustrate view handles we can use the *Evaluate* window; this is a useful tool for it lets you write a line of script and *execute* (run it) or *evaluate* it (run it and display the last result in the line). Open a data file, then navigate to the Evaluate window by pressing *ctrl+L*. Type `FrontView()` and click *Eval(…)*.



In the example image, my current view handle is 11. You can check this in the *Windows* menu *Windows…* command (which lists view handles). If you now use the `View()` command and click *Eval()* the result will also be 11. Now set the following text and click *Eval():*

```
View(LogHandle());View();
```

The `LogHandle()` script command returns the view handle of the *Log window* (which always exists). This command line leaves the data file as the front view, but makes the log view the current view and the final `View()` returns the current view (log), most likely with a view handle of 10.

Here is a little example of using a script to point to different views. It assumes that your data view is still open and has view handle 11, but you can replace this with your view handle. You can, of course, use variables to avoid dealing with view handles as fixed values (we will cover this in a future article).

```
View(11);                'Make data view current
Window(0, 0, 100, 100); 'Set to use full area
Message("View 11 uses full area");
View(LogHandle());       'Make Log view current
WindowVisible(0);        'Hide Log view
Message("Log view is invisible, current view is %d", View());
```

In this example, the view with handle 11 is left on screen and the log view is invisible. However, because `View(LogHandle())` was the last view command used, the log view is still the current view. If we were to change it to:

```
View(11);                'Make data view current
Window(0, 0, 100, 100); 'Set to use full area
Message("View 11 uses full area");
View(LogHandle()).WindowVisible(0); 'Swap, hide, swap back
Message("Log view is invisible, current view is %d", View());
```

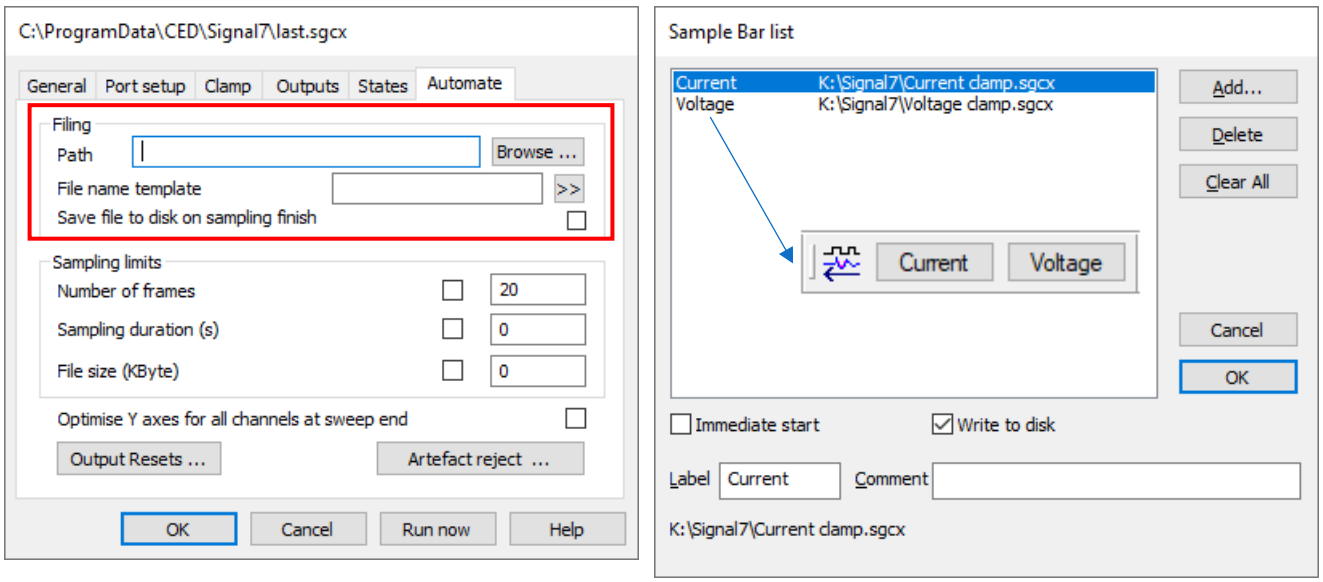
As we have used the swapping version of `View()`, view handle 11 is still the current view.

Back to contents

Recent Questions

I want to use voltage clamp and current clamp on the same cell, how would I set this up in the sampling configuration?

While Signal can easily handle this, it cannot perform the switch over within the same data file as the amplifier settings for the two modes are too different. To switch from one mode to another it is necessary to stop sampling and save the new data file, change the amplifier mode and sampling configuration, then start sampling again to a different data file. We recommend you make use of the *Automation* tab of the sampling configuration, which provides automatic file name generation and data file saving facilities that help to make this quicker:



The sampling configuration can be set up to switch and start at just the click of a button using the Sample Bar. Firstly, create your two sampling configurations as needed, and go to *Sample* > *Sample Bar list*. Here you can *Add* your sampling configurations and edit the labels. When added, your sampling configurations appear as buttons on the *Sample Bar*. If ticked, the immediate start option will automatically start your recording as soon as the sample bar

button is pressed. Similarly, the write to disk option sets the initial state of write to disk check box in the sampling control panel to on or off.

As an example, your steps to run a dual mode experiment may look like:

- Set the amplifier into voltage clamp mode
- Load in the voltage clamp sampling configuration file
- Run the experiment and collect data
- Stop sampling and save your data
- Set the amplifier into current clamp mode
- Load in the current clamp sampling configuration file
- Run the experiment and collect data

Remember that any changes you make to the amplifier gains or external control sensitivities must be accounted for by altering the relevant ADC port or DAC calibrations of your sampling configurations, in order for your signals and stimuli to be correctly calibrated. It is a good idea to set up the amplifier with suitable gains and sensitivities, get the calibrations correct for these, and refrain from changing the amplifier settings.

Back to contents

CED User forums

Try the CED Forums bulletin board for software and hardware support.

If you have any comments about the newsletter format and content, please get in touch: Marjorie@ced.co.uk.

To adjust your subscription preferences, please visit our website: www.ced.co.uk/upgrades/subscribeenews.

Back to contents

Contact us:

In the UK:
Technical Centre, 139 Cambridge Road,
Milton, Cambridge, CB24 6AZ, UK
Telephone: (01223) 420186
Fax: (01223) 420488

Email: info@ced.co.uk
International Tel: [44] 1223 420186
International Fax: [44] 1223 420488
USA and Canada Toll Free: 1 800 345 7794
Website: www.ced.co.uk