CAMBRIDGE ELECTRONIC DESIGN LIMITED

eNEWSLETTER

#7
October 2020

Contents

Welcome

Training days

Latest software

Here to help

Script Spotlight

- Email notification

Spike2

- Channel draw modes available per channel type
- Script – Simple Auditory Brainstem Responses

Signal

- Dynamic clamping with Signal
- Script – Export frames as pictures

Scripters corner

- Arrays

Recent questions

- Overlaying traces from stimulations in Spike2

CED user forums

Welcome

Thank you for downloading our October newsletter. The clocks have turned back in the UK and our days have noticeably less daylight, however our software team has still been hard at work to bring you updates to Spike2 and Signal. Version updates 10.07, 9.12 and 8.20 of Spike2 were all released at the end of September, so make sure to update your copy.

In 10.07 we're pleased to bring you 21 new features including improvements to video recording, adding channels without a y-axis to channel grouping and the new script command ArrRev(). The full release notes are found on our website, or in the Technical help > Revision history section of the in-software help, accessed by pressing F1. Signal version 7.06 is under test and will be released soon. Changes include the script command Binzero() can now be used to set the start time of a frame in a file view, plus 27 other features and fixes. Major updates are also coming to Signal, watch this space!

Our script writers have been working to deliver custom scripts for your projects; check out the collection on our website or email us if there is something you wish to achieve with our help. Furthermore, we are interested in hearing about your suggestions for features and improvements to both Signal and Spike2. As always, should you have any suggestions, questions or queries for the newsletter please get in touch Marjorie@ced.co.uk.

Training Days

Due to the ongoing COVID-19 pandemic, all current training events have been put on hold. We do however offer remote training sessions by Skype/Zoom/Teams 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 10.07	09/2020	Version 7.05a	02/2020
Version 9.12	09/2020	Version 6.05b	10/2019
Version 8.20	09/2020	Version 5.12a	02/2018
Demo	09/2020	Demo	02/2020

Bug report: Spike2 versions 7-10 PlayWaveStatus$() position bug

A bug has been discovered that affects Spike2 users who sample data with a Power3, Power3A, Micro3 or Micro4 and use the PlayWaveStatus$() script command to read back the position that waveform output has reached. This is most unlikely to affect you unless you upgrade from an old version of Spike2. If you might be affected see this bulletin board post for details of the problem and a fix.

http://www.ced.co.uk/phpBB3/viewtopic.php?f=5&p=10868#p10868

We will add the fix to future releases of Spike2, so if you are unaffected there is nothing for you to do. If you have a script that depends on the wrong behaviour and are not able to modify your sampling script to cope with the change, please contact us for help.

Back to contents

Here to help

We know access to your labs has been erratic for the past 6 months, and with the new local lockdown measures that is unlikely to change over winter. However, CED will still do all in our power to support you for increased home working. Should you require any help or wish to discuss your system, email us at info@ced.co.uk and we can arrange for a video call via Skype/Zoom/Teams.

We also have tutorial videos for both Spike2 and Signal available on our website for you to peruse at your leisure. There are new videos and updates in the pipeline, however if you have a particular topic you think could benefit from a tutorial video please let us know.

Back to contents

Script Spotlight – EmailNotification.s2s 

Have you ever found yourself in the situation where you cannot afford the time to wait for an experiment to finish. Or perhaps you need to leave an experiment running overnight and are concerned about something interrupting your recordings. We have developed a small Spike2 script that will send you an email notification when your recording stops. This script will notify you of the date and time when recording finishes and include a small screenshot of the data.

We have successfully tested this script with both Gmail and Outlook addresses, however there are some limitations to the script. Firstly, both your email address and password must be added to the script text. We therefore recommend that email address receiving personal/secure information are not used, and you only use an email address you are happy to have the access information stored in the script. Alternatively, you could set yourself up a new address with a free email client specifically for this purpose. This script will not function correctly if some form of two factor authorisation is enabled.

To setup the script you must first edit the following fields:

```
Var ToAddr$:="recipient@gmail.com";     'Send to address
Const User$:="example@gmail.com";        'User credential
Const FromAddr$:="example@gmail.com";    'From address
Const Password$:="Password";          'User email password
Const Server$:="smtp.gmail.com";              'Users SMTP server
```

ToAddr$ is the recipient email address, which could be your work email or an email you have access to on your phone. The User$ field is the user login to the email account for sending (typically the same as the email address), and Password$ is the password (ensure case sensitive characters are entered exactly). The SMTP server must also be entered for the Server$ variable; for example, the Gmail server is smtp.gmail.com, and the outlook server is smtp.live.com.

With these variables updated, the script can be run in the background of Spike2 whilst you setup your recording and begin sampling. Each time a recording is started, the script updates to monitor the view that is sampling. Should the sampling stop, the script will attempt to send the email notifying you. This will not cover every scenario, for example if there is a power cut or you lose network connection, however for most cases it remains useful.

This script also has the potential to be altered for other uses, for example a notification could be sent if a particular threshold is passed. It is also possible to send text notifications with the use of third-party software, however we opted for the simpler email notification. If you have any ideas for other helpful uses of this script for yourself, please get in touch with us at marjorie@ced.co.uk.

Back to contents

Why can't I see the same draw modes available for different channels?

The Channel Draw mode option in the *View* menu gives access to a variety of data display modes. However, the available modes depend on the source channel data type. The following draw modes are available for *Event* and *Marker* type data:

Dots and Lines – Displays the events as simple time stamps using dots or vertical lines. Marker channels displayed as dots also show the Marker code. When displayed as lines, a central horizontal line to link the events is enabled with the Centre line check box.

Mean Frequency – Calculates and displays the mean frequency at each event by counting the number of events in the previous period as set by the time width field. Mean Frequency may be calculated in Hz or events per minute (BPM) using the Per minute option.

Interval – Draws the time interval between an event and the previous event on the same channel, measured in seconds. Displayed as either Dots, Lines (linking dots), or Skyline (horizontal lines between dots).

Instantaneous Frequency – This is the inverse of the time interval between an event and the previous event on the same channel. Using the option Per minute changes to rate per minute instead of per second. The result can be displayed as Dots, Lines (linking dots) or a Skyline (horizontal lines between dots).

Rate histograms – Displays a count of events in each time period (set by the Time Width field) as a Histogram. This form of display is especially useful for comparing the event rates before and after an operation.

Raster display – Shows event positions relative to trigger times on a selected channel and each trigger event is used to set time 0 on the Y-axis. When this draw mode is applied to a marker channel the events are drawn in the colours set for the individual Marker codes.

State – Displays a coloured state 'bar' for each marker code that spans the time range from the start of the marker up to the next marker in the channel.

The following draw modes are available to *Waveform* and *RealWave* data:

Waveform and Cubic Spline – In Waveform mode the spike shape is shown with the data points joined with straight lines. Cubic Spline mode joins the data points with smooth curves.

Dots and Skyline – As with event data, Dots draws each data point as a dot, and Skyline draws horizontal lines between data points.

Sonogram – This mode shows how the frequency content of a waveform channel changes with time. The y axis is displayed in Hz for the frequency range 0 to one half of the channel sampling rate. By default, intensity of the frequency content is indicated by a grey scale, the darker the image, the more intense the signal. However, you can choose a colour scale, or create your own scale in the *Edit* menu > *Edit Preferences* > *Display* tab.

Finally, the following draw modes apply to *WaveMark* data only:

WaveMark – Draws the spike shape as a Waveform and displays the associated marker code.

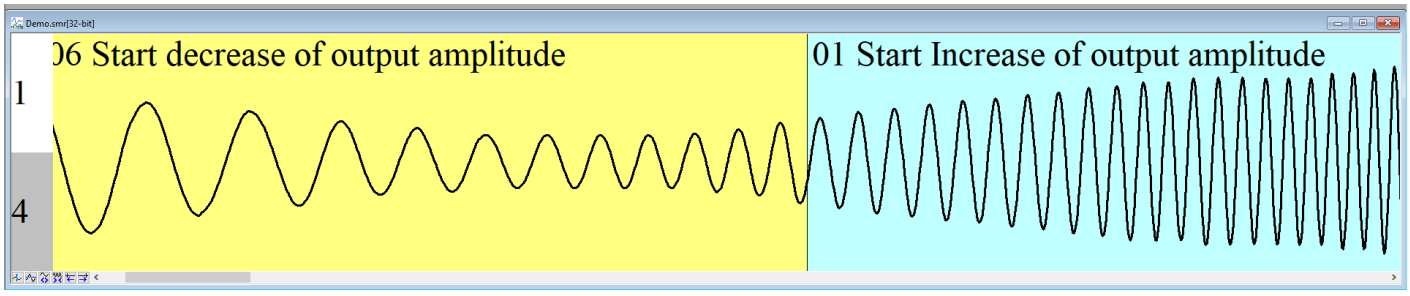
Overdraw WM - The overdraw mode draws all spike data in the time range as superimposed waveforms in a channel area at the top of the data window. This draw mode is particularly useful for comparing, contrasting, and identifying spike shapes.

Drawing mode tips:

Duplication of channels – You can display a channel in multiple display modes at the same time by duplicating the channel and setting different draw modes to each duplicate

Draw modes with the Marker Filter – The Analysis menu Marker Filter function is used to show or hide data associated with a marker code. Any channel draw mode or analysis performed on a channel with a marker filter applied is then only applied to the visible data.

Channel grouping – With Spike2 update 10.07 it is possible to group channels without a y axis, prior to 10.07 only channels with a y axis could be grouped. Therefore, you can now group a channel in State mode with others to colour the background of regions of a waveform:



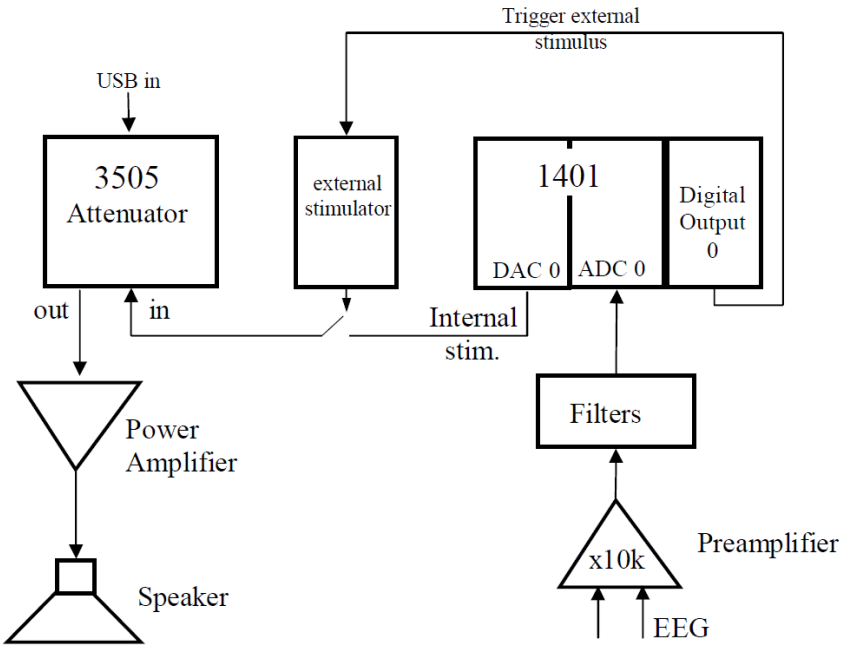
In this example, channel 4 (TextMark) is drawn in State mode and grouped with channel 1 (Waveform).

Back to contents

Scripts: Spike2

We have recently improved our Simple Auditory Brainstem Responses script package, available on our website. This Spike2 script package is suitable for recording auditory brainstem responses (ABR) to tone pip stimuli. It was developed for research on responses of fish to underwater sounds but may be equally useful for investigating the responses of other vertebrates to airborne sound. The ABR software consists of 3 files, the script: Fishabrn.s2s; the sampling configuration: fishabrn.s2c(x); and the sequencer file: fishabr v1.15a.pls. Before running this package for the first time you must calibrate the auditory stimuli using the included SoundCalSUxx script.

The script generates symmetrical tone pips with a linear rise and fall phase and filled with carrier frequencies ranging from 5Hz to 10kHz via DAC0. The user can set up the shape of the tone pip, select the polarity (or choose alternating polarity), stimulus repetition rate and number of sweeps to include in the average. The stimulus intensity is set in steps of 5dB over a range of 75dB using the CED 3505 program-controlled attenuator. A calibration curve to correct for variations in intensity with stimulus frequency is applied automatically.



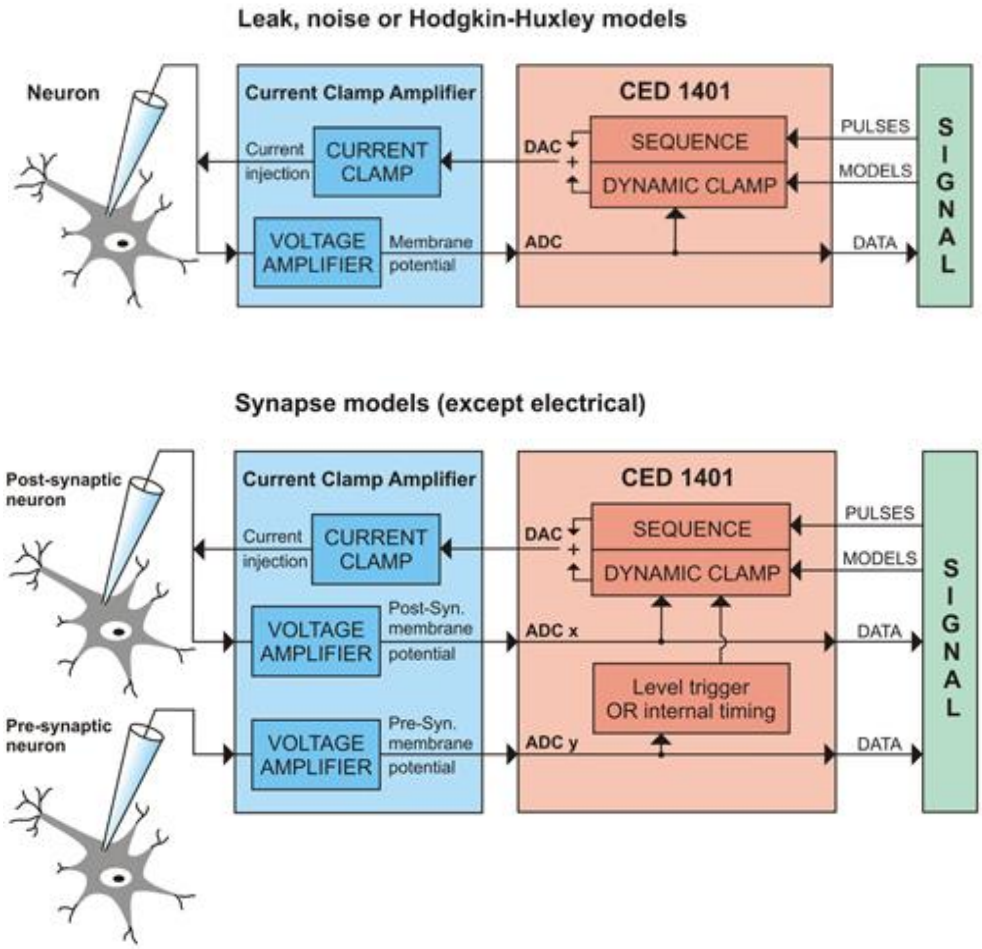
The script:

- Processes the pre-amplified neural responses recorded via ADC0 by signal averaging
- Displays the averaged responses at each stimulus frequency in descending order of stimulus intensity for easy estimation of thresholds
- Builds up a threshold-response curve as the experiment proceeds

Back to contents

I would like to know more about Signal's capabilities for Dynamic Clamping

Dynamic clamp is a specialised technique in which current is delivered to a cell to represent the actions of virtual ion channels, allowing ion channels or synapses to be simulated or the actions of existing channels to be nullified; in effect adding or subtracting conductance to or from the cell membrane. These virtual ion channels or synapses are modelled using equations. A model can define something as simple as an ohmic leak conductance or as complex as a population of Hodgkin-Huxley ion channels, or a synapse. These models take sampled waveform channels as inputs and apply a set of equations that determine the current to be injected by the amplifier, which is controlled by a DAC output from the 1401.



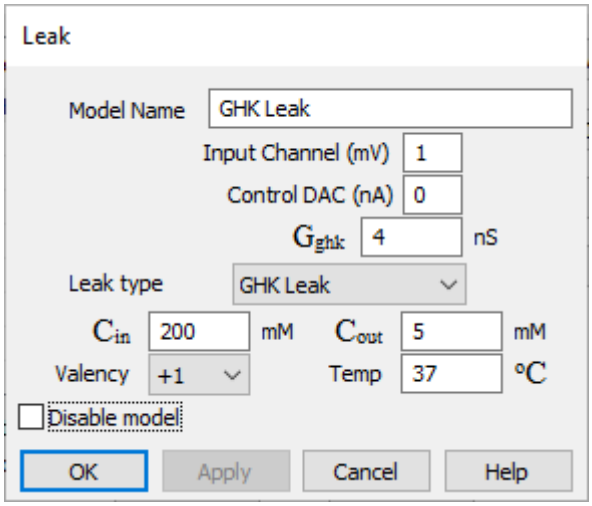
Signal control of dynamic clamping

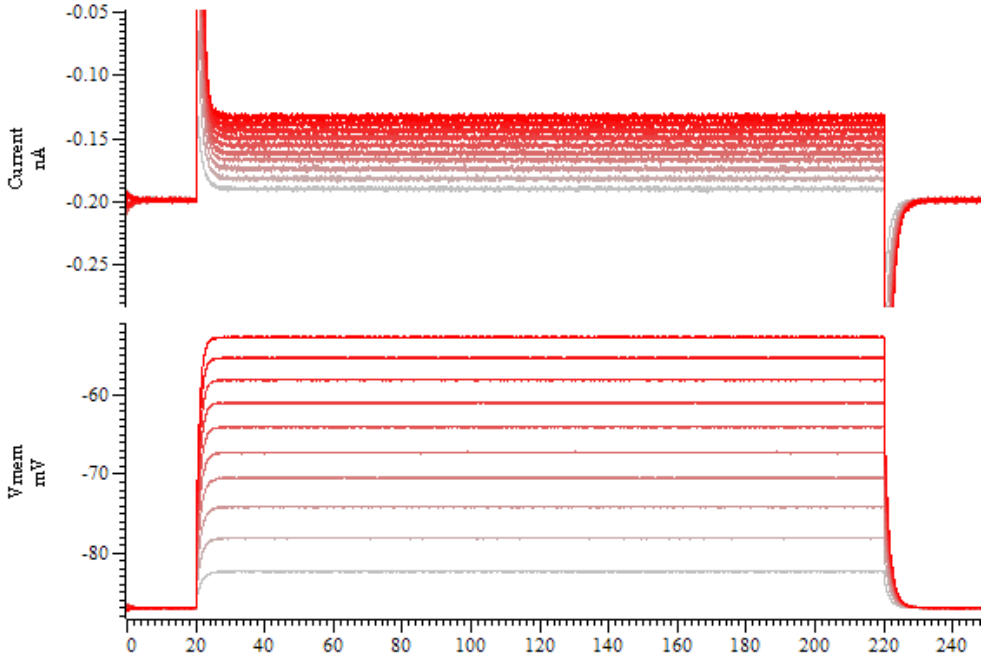
To be useful for research this process must be done quickly relative to the speed of the biological process being simulated; the mechanisms used to evaluate the equations need to be fast. Signal version 7 includes a high performance, easily configurable, dynamic clamping system which is fully integrated into the sampling configuration. It makes use of the Power1401-3A and Micro1401-4 design to evaluate the model equations at the same rate as the ADC inputs are sampled, ensuring that the timing is fast and unaffected by the non real-time aspects of the Windows operating system.

A simple GHK Leak example

A Goldman-Hodgkin-Katz leak is a model which generates a simulation of spontaneously opening ion channels where ionic movement through the channels is driven by a difference in the ionic concentration across the membrane.

The input channel is used to record the membrane potential, connect this to the scaled output port of your current clamp amplifier. Connect the Control DAC to both the external command input of the amplifier and an ADC channel of your choice to visualise the current being injected into the cell. Many amplifiers have a second output for connecting to an ADC channel without the need to connect the DAC directly. In this case the DAC only needs to be connected to the command input of the amplifier. The input channel should be calibrated in mV and the Control DAC and current channel calibrated in nA or pA. To show the leak in action we also need a varying current input, which is achieved by adding a varying current stimulation pulse (200 ms, amplitude 0.1 nA increasing to 1nA in steps of 0.1 nA) to the Control DAC output in the pulse configuration dialog.



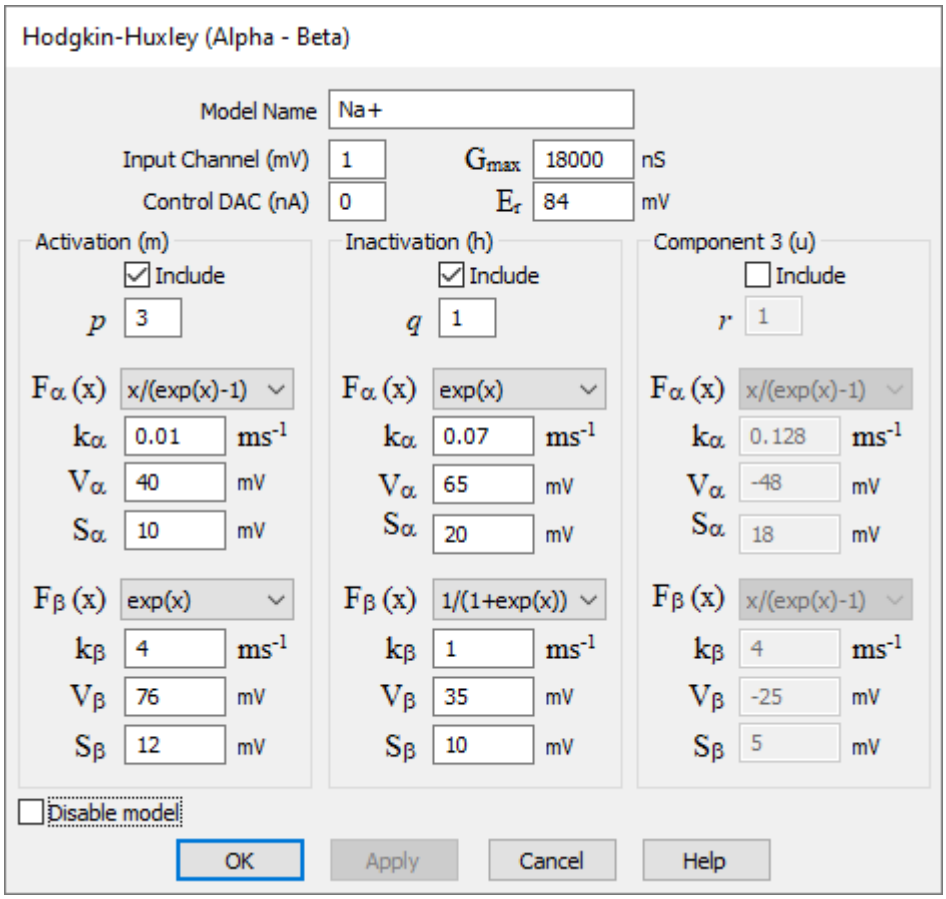


GHK leak example

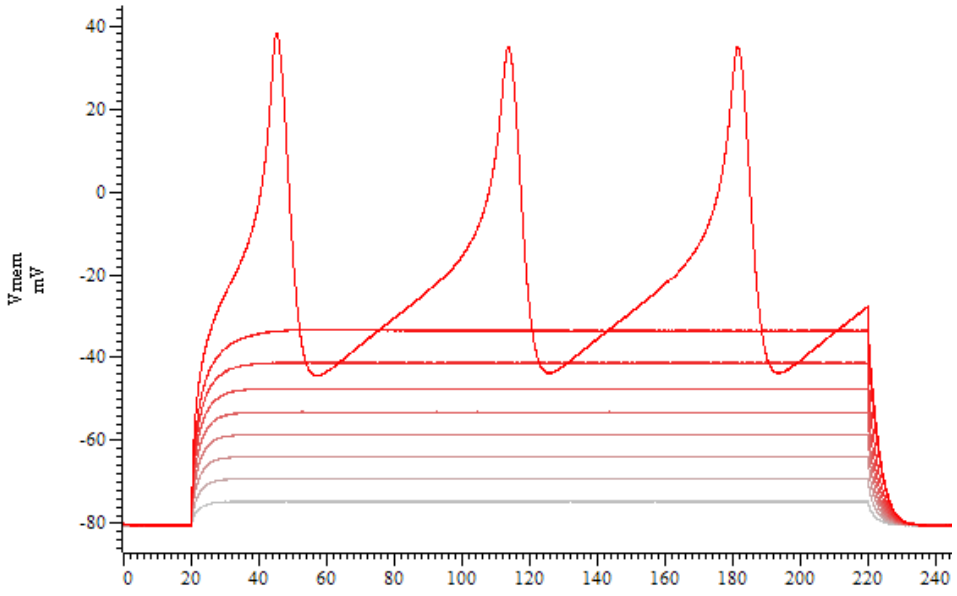
The simulated ion diffusion generated by this model has the effect of driving down the baseline membrane potential towards a resting potential generated by to the ionic concentration difference, partly cancelling out the effect of the input pulse

A more complex Hodgkin-Huxley example

A Hodgkin-Huxley model simulates ion channels whose behaviour is voltage dependent with complex dynamics. This example aims to simulate action potentials by starting with the example GHK model used above and adding suitable voltage-dependent sodium channels.



When this sampling configuration is run with both models enabled it generates simulated action potentials in response to the increasing current injection, as displayed below.



Simulated action potentials generated using dynamic clamping

The attached sampling configuration, DynClampTest.sgcx, contains both of the models described above and should be ready to run with Signal version 7, a Power1401-3A or Micro1401-4. You also require a suitable patch clamp amplifier in current clamp mode that has an external command sensitivity of 2 nA/V and output gain settings which give a scaled membrane potential sensitivity of 10 mV/mV. This should be connected to a model cell that has an internal 500 MOhm internal resistance for these testing purposes.

Back to contents

Scripts: Signal

We have developed the small script ExportFramesPictures.sgs to export a range of selected frames from a data file as individual images. The script will operate on an open data view or ask you to select one with the *File Open* dialog if a view is not found. Once the data view is opened, the script pauses to allow you alter the appearance of the data as required. When you have completed your adjustments, click *OK* from the top right toolbar. A new dialog opens, allowing you to select *All frames*, *tagged frames*, or *un-tagged frames* in the defined range. The available formats are Windows metafile (.wmf), enhanced metafile (.emf), bitmap (.bmp), Joint Photographic Expert Group (.jpeg), Portable Network Graphics (.png), and Tagged Image File Format (.tif).

Back to contents

Scripters Corner – Arrays

In a previous article we discussed simple variables that hold single values. However, it is often the case that we need to deal with multiple items of the same type, such as a waveform of sample points or a list of channel or file names to process.

To declare such data types we use the 'array' construct. An array is declared using the `var` keyword and, like other variables, can be a list of integers, real numbers, or strings (in version 10 you can also have arrays of user-defined Objects, but we leave that for a future article). The number of elements in the array is included in square brackets after the variable name in the `var` statement:

```
var MyInt%[10];    'an integer array of 10 elements
var MyReal[4];     'a real array of 4 elements
var MyStr$[7];     'a string array of 7 elements
```

As declared above, the array elements are initialized to default values; numeric elements are zero and strings are empty. An individual element from the list is denoted by the array name followed by square brackets enclosing its position in the list:

```
var data%[4];
data%[0] := 10;
data%[1] := 20;
data%[2] := 30;
data%[3] := 40;
```

It is also possible (since Spike2 version 8.03) to *initialize* an array to values that are *constant expressions* at the point of declaration. By this we mean the data must be known at the time of compilation. To initialise an array, enclose the list of data separated by commas in `{ }` brackets:

```
var data%[4]:={10, 20, 30, 40};      'initialised array
```

The size of the array in square brackets is optional when initialising arrays; were you to omit it, the number of elements in your `{ }` enclosed list determines the size of your array. However, it is an error to include a [size] smaller than the number of {elements} in the list. For example:

```
var bad%[4]:={1, 2, 3, 4, 5}; 'Error: dimension size exceeds declared size at '5'
```

Should you need to change the size of an array, we can use the resize statement:

```
var data%[100];  'array of 100 elements
resize data%[5]; 'now only 5 elements
```

One could use this to reduce a list with empty values at the end. For example I declare an array with plenty of room to store data as a list, then once I have grabbed all the data needed I use resize to reduce the list to just the elements holding data.

```
'Example: array
var data%[4], i%, total;
data%[0] := 1; data%[1] := 3; data%[2] := 8; data%[3] := 9;
for i% := 0 to 3 do
    total := total + data%[i%];
next;
PrintLog("Mean is %f\n", total / 4.0);
```

The `for ... next;` loop groups the statements between for and next and runs them for each value of the variable `i%` (in this case it has the values 0, 1, 2 and 3). Each iteration of the loop accumulates the values in the vector in the total variable. We will discuss flow of control statements (like `for ... next`) in detail in another article.

In this example, the array `data%[]` has one square bracket (an index). This is known as an array with one dimension, or a *vector*. If we declared an array of reals as `fred[8][6]` this creates an array with 2 dimensions and 48 elements addressed with two indices. We call an array with two dimensions a *matrix*. From Spike2 version 6, arrays with up to 5 dimensions are allowed; we do not have special names for 3 to 5 dimensions.

Note that Spike2 has built-in functions to perform many standard operations. The above calculation of the mean of an array of integer data could have been written as:

```
var data%[] := {1, 3, 8, 9}, mean;
ArrSum(data%, mean);    'symbol data% or data%[] means the whole array
PrintLog("Mean is %f\n", mean);
```

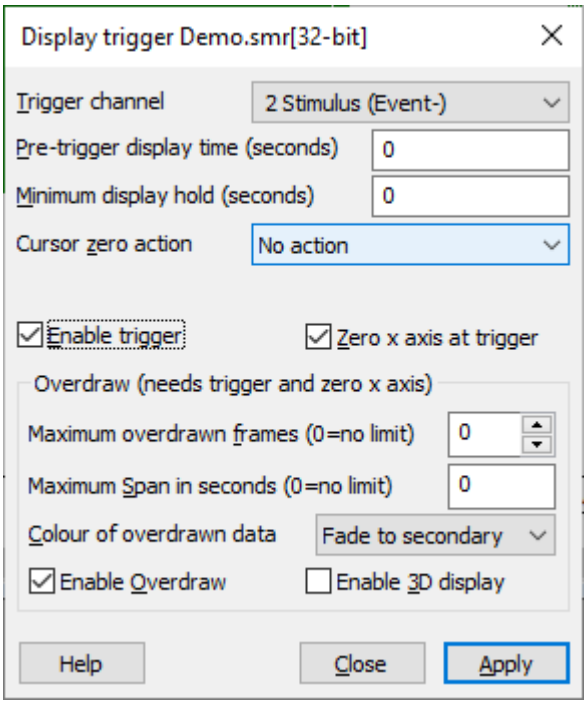
A common error when working with arrays is forgetting that the first element is [0], with the last element one number less than the total elements. A vector array with 36 elements would be written as `data%[36]`, with the list starting at element `[0]` up to element `[35]`.

You can find much more information about arrays in the Spike2 online Help. See: Script language > Script language syntax > Arrays of data.

Back to contents

Recent Questions

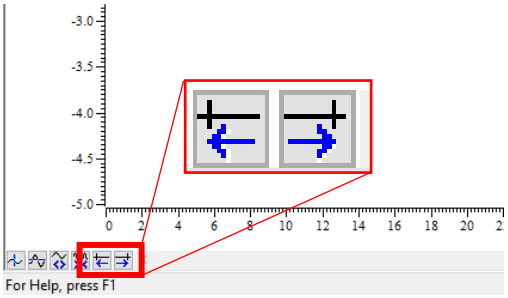
Is it possible to overlay traces from a stimulation point within Spike2?



To overlay traces we would use the *Trigger/Overdraw* feature within Spike2. In overdraw mode, data sections identified by triggers are overdrawn in time order, oldest first up to the current time. The displayed time range is altered through the *X axis range* dialog, but trigger times are ignored if time before 0 or time after the end of the file would be displayed. Your stimulation triggers would already be marked either as an event or a marker. To get started, access the *View menu > Trigger/Overdraw > Display Trigger* dialog, or right click your data and select *Trigger/Overdraw*.

This dialog controls the overdraw feature. First select your *Trigger channel*, and then select *Enable trigger* and *Zero x axis at trigger* to enable the overdraw feature. Click *Enable Overdraw* to begin. There are optional functions available, such as pre-trigger display times, limits for the maximum number of frames to overdraw and the maximum span in seconds, etc., more details on these are found by accessing the *Help* from the bottom left corner or pressing F1. Click *Apply* when finished.

From your data, the displayed time range is altered by double clicking the x axis to open the *X axis range* dialog. Sections of data identified by triggers, or frames, are added or removed by using the trigger buttons added to the bottom left toolbar when overdraw is enabled (right).



Alternatively, the *Overdraw list* dialog may be used to process all triggers within a time range. Use the *View menu > Trigger/Overdraw > Overdraw list* or select it from menu by right clicking your data. Choose the *event channel* and time range to process, and either *Replace* the current overdrawn data or *Add* to what is already there.

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