CED
CAMBRIDGE ELECTRONIC DESIGN LIMITED

*e*NEWSLETTER

#15
October 2021

Contents

Welcome

Thank you for downloading our October newsletter. With Autumn officially here we wave goodbye to the sun in the UK for another 6 months. Our colleague Luke is also leaving us this month. Luke spent almost 3 years with us as a Regional Sales Manager and we wish him all the best with his future endeavours.

Version 10.12 of Spike2 is currently undergoing testing and should be released soon. Version 10.12 includes lots of new features like extra measures of clustering quality (Lratio and Isolation distance), and there are new options in the Measurements to XY views and to Data channels dialogs for cursor adjustments. Version 10.12 also includes several fixes, so make sure to update your copy of Spike2 on its release.

We are excited to be attending the Brain Stimulation conferences in person. These will be our first in person events since the beginning of the COVID-19 pandemic. Come see us at Brain Stimulation in Charleston, South Carolina, USA between December 6-9th.

Our software engineers have finished the development for the Talker for OT BioElettronica devices and is in the final stages of testing. The Talker currently supports the 64 channel EMG Sessantaquattro device. Our next effort will be to add support for the 400 channel EEG/EMG Quattrocento.

Training

Full recordings from this year’s earlier online training sessions may be viewed on our website. We have more sessions planned for later in the year, for which details will be shared nearer the time.

We also 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 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: 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 10.11a	10/2021	Version 7.06	04/2021
Version 9.14	06/2021	Version 6.06	04/2021
Version 8.22a	09/2021	Version 5.12a	02/2018
Demo	08/2021	Demo	04/2021

Back to contents

Here to help

We know access to laboratories has been erratic for the 18 months, but with the easing of lockdown measures we hope that conditions will continue to improve. We are now able to offer site visits when necessary, however we will continue to support remotely in the first instance. CED will also continue to do all in our power to support you for increased home working. Should you require any help or wish to discuss your system, email Marjorie@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. If you have a particular topic you think could benefit from a tutorial video, please let us know. All these videos are also available through our YouTube channels: Spike2 / Signal.

Try the CED Forums bulletin board for further software and hardware support. Ask your questions and receive valuable input from our super users.

Back to contents

Script Spotlight – App()

You may have often seen the line `View(App(3)).WindowVisible(0);` at the start of scripts downloaded from our website and wondered what this is used for. The script command `App({type%})` returns view handles for system specific windows (positive type% integers), with App(3) specifically returning the view handle for the current running script. The `View(vh%).x()` variant of the View() command saves the current view, makes the view with the view handle vh% the current view, runs the x() function and restores the original current view.

Armed with this knowledge, you can see that `View(App(3)).WindowVisible(0);` makes the running script the current view, hides it and restores the original current view. Placing this at the start of a script makes sense as most users of a script do not want to see it; they just want it to do its job and keep out of the way.

The App() command also affords you the ability to show/hide the various Spike2 toolbars, such as the Script bar, Sample bar, and Sample Status bar. If your script requires the user to interact with the Sequence control panel for example, you could use `View(App(9)).WindowVisible(1);` to ensure it is shown. Conversely, you may want the user to only use the controls created in your scripted toolbar, in which case you would hide the Sequence control panel.

Quite a common requirement at the start of a script is to hide all the screen furniture that is not relevant to the script and restore them all at the end. The online Help for the App() command includes a very useful Example that you can cut and paste into the end of your script. Then add the script line `HideAll();` At the start of your script and the line `RestoreAll();` just before the end. This will hide all the various toolbars to maximize your screen area for the script, then restore them all at the end.

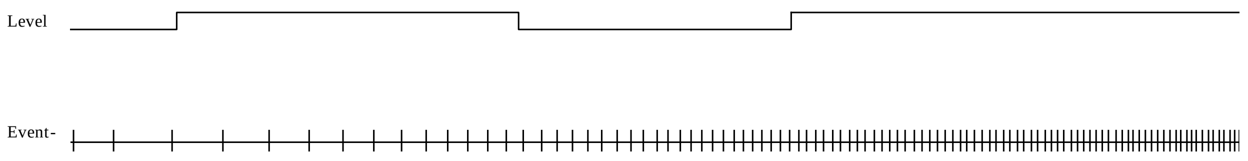
When used with negative type% values, App(type%) returns application specific information. Check the in-software App() help topic for the full list and details.

Back to contents

How do I sample event data?

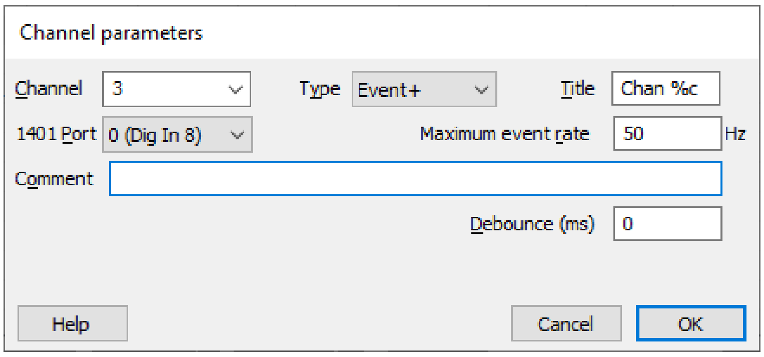
Event channels store the times of changes to TTL compatible signals connected to the 1401 digital input bits 15-8. These are typically used to record stimulus times or responses from an event detector. Event channels record time stamps based on the rising edge of a digital pulse (Event+), the falling edge of a digital pulse (Event-), or both (Level).

Event+ and Event- channels store only the time of the event. Level channels in a modern .smrx file, store the time and the level (high as code 1, low as code 0). In an old .smr file, Level channels stored the time only with the assumption that the level alternated between high and low. Level channels are used to indicate the on or off state of a connected device such as a heat lamp or light source where the duration of the event matters. Event+/- channels use fewer resources (time and storage space) than Level channels and are preferred where there is a choice.



Examples of an Event- and Level channel 1

You create Event and Level channels by clicking *New* in the *Sampling configuration* and selecting the kind to record in the *Type* drop-down list. Event and Level channels use 1401 digital input bits 8-15, as set by the *1401 Port* drop-down. Bits 8 and 9 are available on the 1401 front panel Event input 0 and 1 BNC connectors unless you use the *Edit* menu > *Edit Preferences* dialog > *Sampling* tab to route them to the rear panel. The rear panel D-type Digital Input connector has bits 10-15 (and 8-9, when enabled). The Spike2 on-line Help has the pin connections and electrical details.


Channel parameters

Channel: 3
Type: Event+
Title: Chan %c
1401 Port: 0 (Dig In 8)
Maximum event rate: 50 Hz
Comment:
Debounce (ms): 0

The *Maximum event rate* field should be set to your estimate of the maximum mean event rate that will be sustained over a few seconds. Spike2 uses this value to optimise buffer space allocation; a reasonable guess is all that is required. This field does not set the maximum instantaneous rate, which may be much higher.

The *Debounce* field sets the minimum acceptable interval between consecutive events in milliseconds. Events that fall closer than this value to the previous event are not saved to disk. Typically, you will use this if logging events from a mechanical switch. These types of switches commonly have 'bouncy' contacts so that, when the switch is activated, you get multiple switch actions in a very short period, all logged as events. Using the *Debounce* setting will log the first event and ignore all those that fall within the debounce time range.

The Marker channel in Spike2 is also used to log 'events' from an external source. Markers differ from Events in that they contain both a time stamp and four 8-bit marker codes. The first marker code is set by reading the values of digital input bits 0-7 from the rear panel Digital Input connector. Markers are often used to signify certain experimental conditions. For example, codes sent to the 1401 from an experiment design program such as E-prime or Presentation to signify different types of stimulus. Channel 32 is always reserved for digital marker data input and is set-up in the sampling configuration by selecting *Marker* from the *Type* drop-down list in the channel parameters dialog

Back to contents

Scripts: Spike2

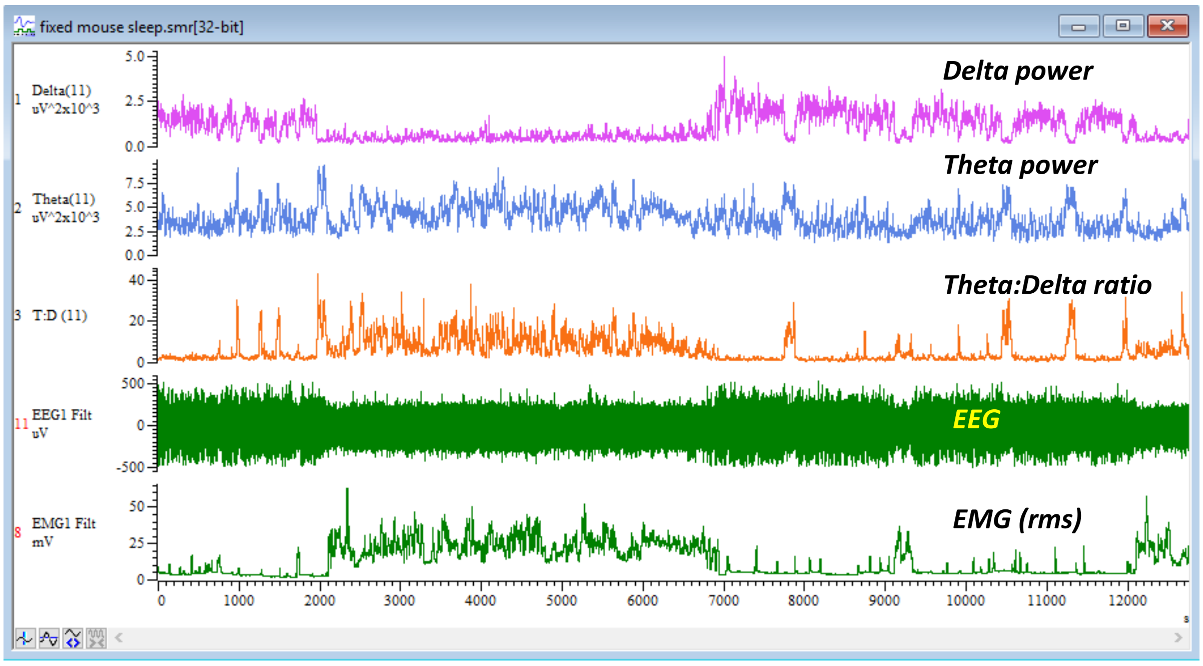
Our script writers have recently improved the On/off-line sleep-stage detector script available on our website. This script is intended for on- and off-line detection of sleep stages: WAKE, REM and NREM (Non-REM) in chicks, mice, or rats. The sleep staging is based on recordings of one EMG and one EEG channel per animal, recorded via chronically implanted electrodes. The script is also suitable for simpler applications such as off-line sleep staging or plotting EEG spectral power in up to 4 frequency bands off-line.

Up to 4 animals can be recorded and staged simultaneously. Recording more than 2 animals simultaneously requires a breakout box (CED 2805 DIO-8) or a home-made custom cable to connect to digital inputs and outputs on the rear panel of a 1401 interface. Optionally, the script can generate trains of digital output pulses when a particular sleep state is detected. These pulses are suitable for controlling a laser in optogenetic studies.

Example configuration

Full details and requirements of running experiments with the script can be found in the help file included with the script download. The example configuration OSD. s2cx and sequencer file GH_OSD3.pls are also included. The example configuration is set up for two animals. You can expand it to 3 or 4 animals by duplicating channels, editing channel titles etc. in the Channels tab. The script prompts you to browse to a suitable sample configuration before recording or to use the same configuration as the on the previous occasion that the script was run. So, you can create several configurations e. g. for different numbers of animals and choose the appropriate one when the time comes.

Operating Principles



The raw data consists of one EEG and one neck muscle EMG recording per animal. In the initial stage of analysis, the EMG is plotted as smoothed root mean square (rms). The script plots EEG spectral power in up to 4 discrete frequency bands, delta and theta plus two optional user-defined bands. Theta:delta ratio (T:D) is also displayed.

Sleep stages are assigned based on EMG level and the power in the different EEG spectral bands. The criteria are not fixed; you must set your own criteria for identifying sleep states in a dialog during setup. For each sleep state you can set a threshold level of EMG activity and a threshold level in one EEG spectral band or ratio of bands.

An example set of criteria might be:

- Assign to REM if smoothed rms EMG <= thr1 AND T:D ratio >= thr 2
- Assign to NREM if smoothed rms EMG <= thr3 AND Delta power >= thr 4
- Assign to WAKE if smoothed rms EMG >= thr5 AND T:D ratio >= thr 6

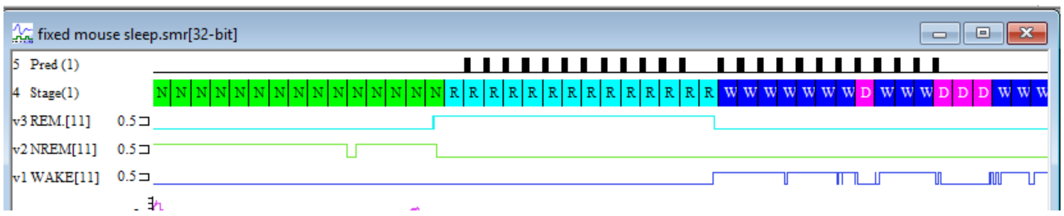
Time ranges where the criteria are fulfilled (or not) are displayed as separate level channels for WAKE, NREM and REM (i.e., a level of 1 where the criteria are met otherwise zero). These 3 level channels are then combined to generate a sleep state channel. Here epochs are assigned to a colour-coded sleep state based on a majority verdict (of time).

The duration of sleep stage epochs is user-defined (see *Preferences* in the script toolbar). The default epoch duration is 10s. When part of an epoch does not meet the criteria for any of the defined sleep states it can be marked as Doubtful. Similarly, if parts of an epoch are marked as belonging to more than one sleep state, they can be marked as Ambiguous. You can specify the precise criteria for epochs to be defined as Doubt (D) or Ambiguous (?) in the sleep stage definition dialogs.

Predicted pattern of output

Sleep state marker

REM, NREM and WAKE markers



The script can generate a train of digital pulses when a target sleep state is detected in any given animal. If 4 animals are recorded, then digital outputs 0 to 3 are used. These pulses can trigger external devices, e.g., to turn room lights on or off for example, or to trigger a laser for opto-genetic studies.

The system does not have 4 completely independent stimulators. Rather there is a single stimulator generating a single pulse pattern that plays out once for every epoch of sleep staging. The script determines whether to switch the output pulses on or off for a given animal and epoch based on its current sleep stage and various other criteria.

Trains of brief pulses are required for opto-genetic studies. The stimulus options provided should be sufficient for this application. The aspects of stimulation that are common to all digital outputs are set up via the *Preferences* button on the script toolbar. This, among other things, opens an *Outputs* dialog. Here, you can define a pulse train which must fit within a sleep staging epoch specifying:

- Duration of individual pulses (ms) and inter-pulse interval (ms)
- Number of pulses in a train.
- Number of consecutive epochs to repeat the stimulus (trains per sequence).
- An upper limit to the number of sequences of stimuli that are presented, i.e., the system will stop responding to the target sleep state.
- A refractory period/a minimum delay after presentation of a sequence of outputs before the system responds again to a target sleep state.

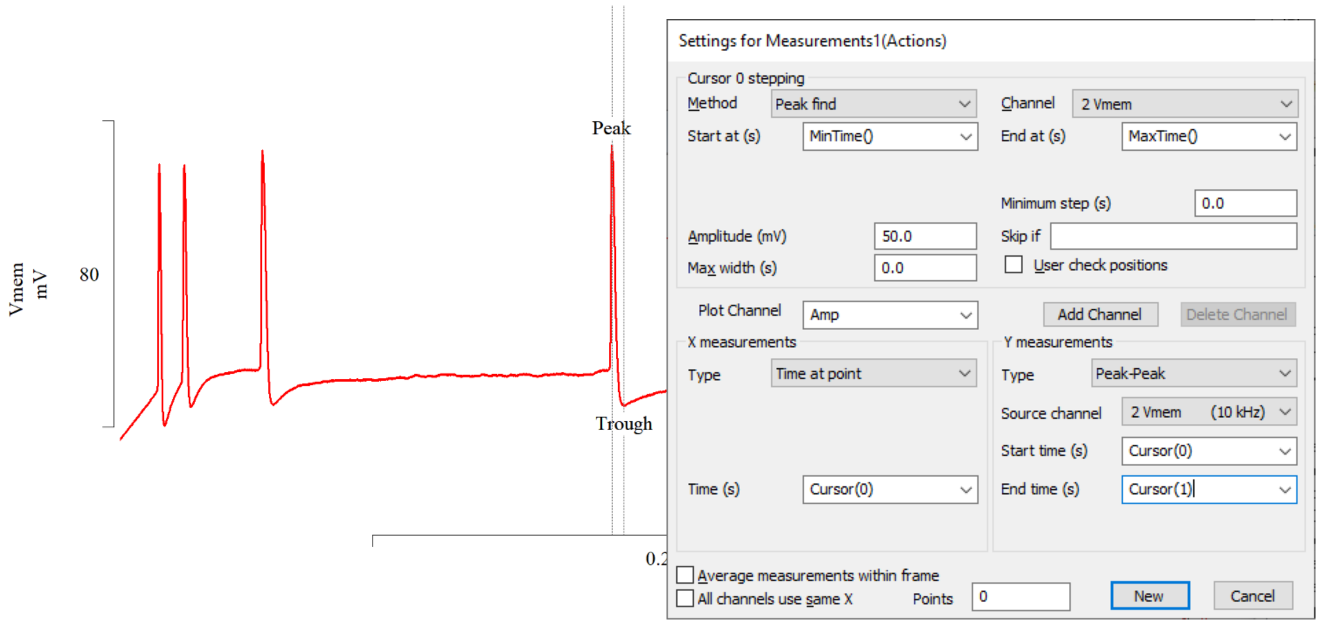
There is also a scheduling scheme that allows you to specify the time when recording will start, and stimulation will start and end. Such scheduling is set up separately at the start of each recording session. You can achieve an almost uninterrupted train of pulses extending over multiple epochs by careful choice of parameters. The refractory period is set in the *Stage Define* dialog. Thus, you can set different refractory periods for different animals if you wish.

Back to contents

Signal — *How do I use cursor measurements to generate an XY plot?*

Signal can generate automated measurements on or off-line based on cursors positioned by the results of feature detection (see our previous newsletter #12 for more information). An XY view can process multiple measurement channels simultaneously and the results can be easily exported to spreadsheet programs using the *Copy as Text* command from the *Edit* menu.

For our example below, we use Cursor 0 and Cursor 1 in *Active* mode. Cursor 0 is set to *Peak find*, and Cursor 1 set to *Trough find* (start search at Cursor(0), and end search at Cursor(0)+20ms). In this example, we set up a measurement of the peak-to-peak amplitude based on the cursor positions in our Actions.cfs example data, included with your copy of Signal (usually C:\Users\User\Documents\Signal7\data). After setting up your active cursors for feature detections open the *Analysis* menu > *Measurements* > *XY View* function. Measurements such as mean waveform value, slope, and time difference, and others can then be taken at cursor iteration and the results plotted to the XY view. The *Settings* dialog includes a section that defines the stepping mode for Cursor 0 (which uses the current *Active Mode* for the cursor if one is already set) as well as the type and time of each X and Y measurement value. The *Start at* and *End at* fields set the time range in the source data from which search for features and take measurements.



Settings for Measurements1(Actions)

Cursor 0 stepping

Method	Peak find	Channel	2 Vmem
Start at (s)	MinTime()	End at (s)	MaxTime()
		Minimum step (s)	0.0
Amplitude (mV)	50.0	Skip if	
Max width (s)	0.0	☐ User check positions	

Plot Channel: Amp — Add Channel — Delete Channel

X measurements

Type	Time at point
Time (s)	Cursor(0)

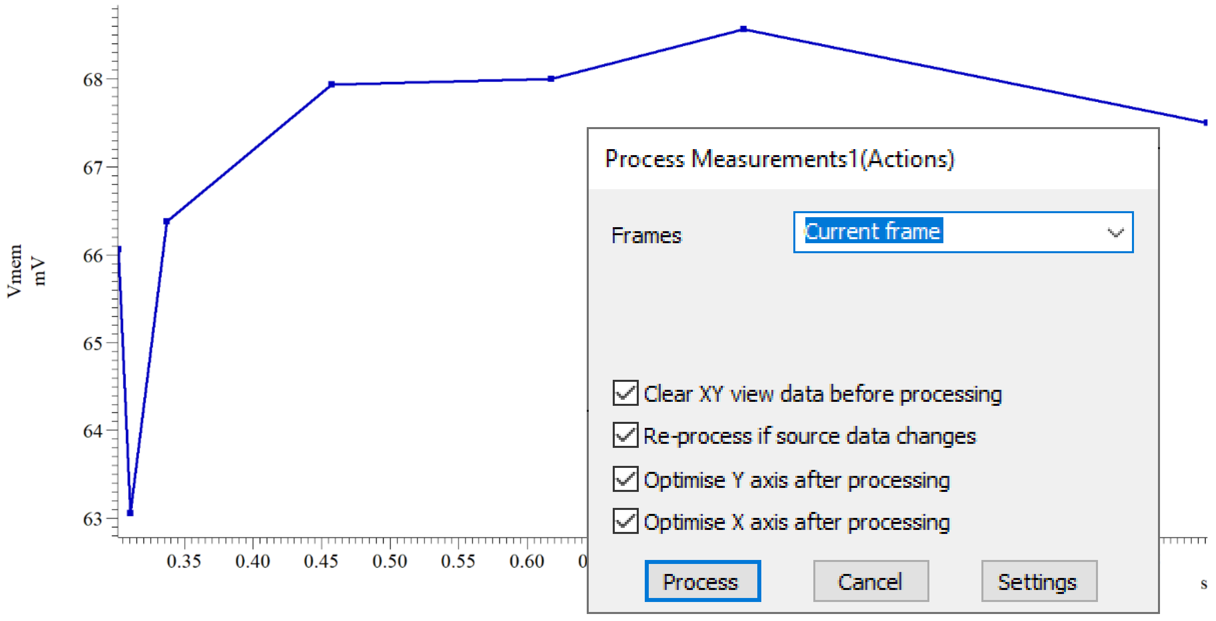
Y measurements

Type	Peak-Peak
Source channel	2 Vmem (10 kHz)
Start time (s)	Cursor(0)
End time (s)	Cursor(1)

☐ Average measurements within frame
☐ All channels use same X — Points: 0 — New — Cancel

Measurement settings

You can define the titles for the new *Plot Channel* as above by simply editing the text in the dialog field. Our *X Measurement* takes the *Time at point* of Cursor 0 for each step. Our Y measurement takes the *peak-to-peak* amplitude between Cursor 1 (trough) and Cursor 0 (peak) for every step through the data. Pressing *New* creates the XY view and opens a dialog that sets the frames in the data file from which to take measurements. Taking measurements from the current frame in our source file gives the plot shown here.



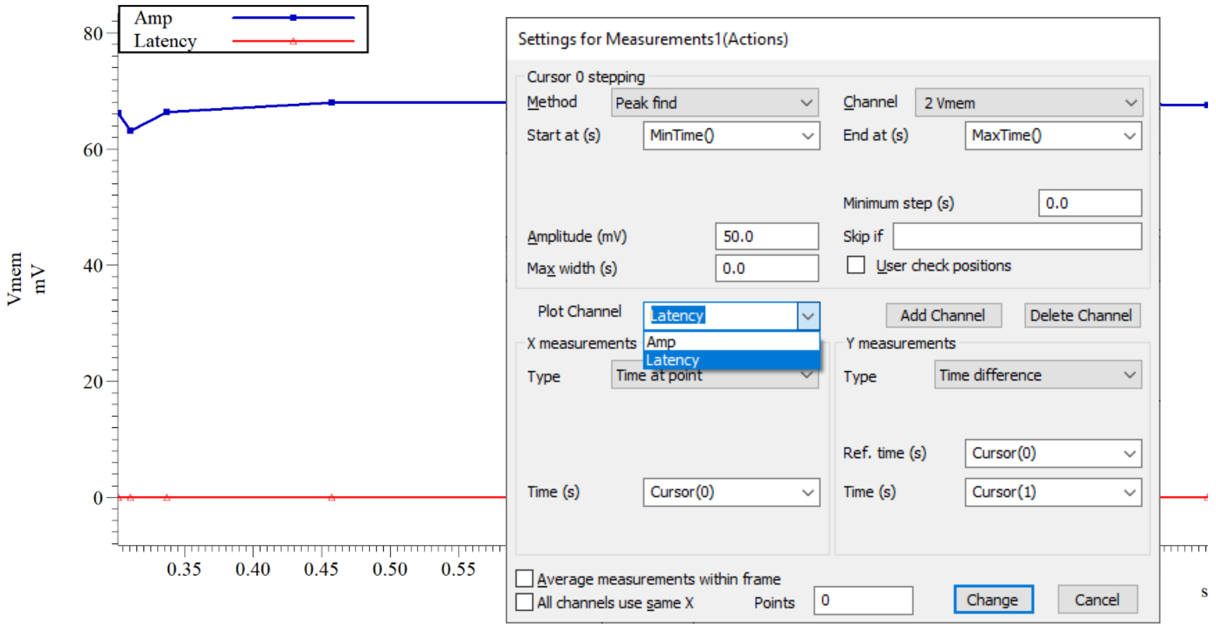
Process Measurements1(Actions)

Frames: Current frame

☑ Clear XY view data before processing
☑ Re-process if source data changes
☑ Optimise Y axis after processing
☑ Optimise X axis after processing

Measurement of peak-to-peak amplitude

By right-clicking on the XY view, we can make changes to the settings and add a second *Plot Channel* that will measure the time difference between Cursor 0 (peak) and Cursor 1 (trough) for every step, for example



Settings for Measurements1(Actions)

Cursor 0 stepping

Method	Peak find	Channel	2 Vmem
Start at (s)	MinTime()	End at (s)	MaxTime()
		Minimum step (s)	0.0
Amplitude (mV)	50.0	Skip if	
Max width (s)	0.0	☐ User check positions	

Plot Channel: Latency (Amp, Latency)

X measurements

Type	Time at point
Time (s)	Cursor(0)

Y measurements

Type	Time difference
Ref. time (s)	Cursor(0)
Time (s)	Cursor(1)

☐ Average measurements within frame
☐ All channels use same X   Points 0

XY view containing the results of 2 separate measurements.

The *Measurements* system can generate 32 Plot Channels holding the results of different measurements. The *Average measurements within frame* check box at the bottom of the dialog applies to all measurement channels. If set then only a single XY data point is generated per frame, this being the average of the data values measured. So, for example, you could use this to produce a plot of mean response amplitude against frame number.

The *All channels use same X* check box below disables the X measurement for all XY view channels apart from the first one and uses this X measurement for all XY channels. If appropriate this makes the setup quicker and easier and when the XY data is converted to text only a single column of X values is created which makes it easier to handle in spreadsheets.

For more information on using the measurement functions in Signal, see the manual or on-line help.

Back to contents

Scripts: Signal

We recently had a request via our forum for help with delaying sampling by 5-10 seconds. The laboratory used Signal to trigger a BiStim Transcranial Magnetic Stimulation (TMS) device. The dilemma the group were encountering was whenever a researcher was performing tests by themselves, they would miss the initial frames due to the time it takes to move pressing start on the laboratory computer to then position the TMS coil over the participant's head.

There are several physical solutions to such a problem like repositioning equipment or foot triggers, however such solutions are not always feasible. A scripting solution to perform a countdown and begin sampling automatically will also work for this case. The script in the barest sense is just a few lines of code:

```
Const Time:=10; 'Time in seconds to wait
Seconds(0);
While Seconds() < Time do yield() wend
SampleStart();
```

All the user would need to do is set up the experiment as normal, and instead of pressing *Start* on the sampling controls they would run the script. The yield() script command pauses script operation until the Seconds() is more than the Time constant. The script command SampleStart() then starts sampling. If the user needed to increase/decrease the delay, they would only need to change the Time constant at the beginning.

The script can further be expanded on, creating additional options like the triggered start from a foot switch, and even opening an info window to display the countdown in a large visible font. Scripts such as these are always useful to laboratory groups and do not take much time to write. If you have a dilemma would like help, please get in touch with us at Marjorie@ced.co.uk.

Back to contents

Scripters Corner – Arithmetic

The script language allows the use of basic maths functions (plus +, minus -, multiply *, divide /) and provides built-in functions to perform calculations on channel data, use the frame buffer in Signal, or manipulate arrays. You can derive most other reasonably common functions from the built-in set. However, if you find yourself unable to achieve something then let us know as it is relatively easy to add new ones.

There are five variations on assigning the result of an expression to a variable:

- variable := expression; Set the variable to the value of the expression
- variable += expression; Add the expression value to the variable
- variable -= expression; Subtract the expression value from the variable
- variable *= expression; Multiply the variable by the expression value
- variable /= expression; Divide the variable by the expression value

For example:

```
var result%:=12, intA%:=36, intB%:=4, intC%:=2;
result% += (intA% + intB%) / intC%; ' evaluates as 12 + ((36+4)/2) to give 32
```

A pedantic note: if you wrote this as:

```
var result%:=12;
const intA%:=36, intB%:=4, intC%:=2;
result% += (intA% + intB%) / intC%; ' evaluates as 12 + 20 to give 32
```

The script compiler is smart enough to know that it can evaluate `(intA% + intB%) / intC%` itself because they are `const`, so cannot change. This generates less code and runs faster.

The += assignment statement is often used in loops to increase a count or measure of time. For example, this simple loop performs a measurement of the standard deviation of channel 1 for every full minute of data in the current Spike2 Time view:

```
var sTime:= 0, value;
const chan% := 1, type% := 12; 'channel and measurement(12 = standard deviation)
while sTime + 60 <= maxtime() do
    value := ChanMeasure(chan%, type%, sTime, sTime + 60); 'measure for 1 minute
    PrintLog("From %d to %d = %g\n", sTime, sTime + 60, value);
    sTime += 60; 'increase time variable by 60
wend
```

The `chan%` and `type%` arguments should be replaced with your channel number and the type of measurement you wish to perform (see the `ChanMeasure()` help topic for the full list). `ChanMeasure()` is roughly the equivalent script command for the Cursor Regions dialog. Some of the measurements can be obtained using other commands, for example `Count()` or `MinMax()`. However, `ChanMeasure()` is usually the first place to look for common measurements (mean, maximum, minimum, standard deviation, areas...)

For point measurements, the `ChanValue()` script command generates the same values as you get from the Cursor Values dialog; it gives you the value of a channel at a particular time. This could be used to grab a single value at a time for a calculation. However, if you need direct access to the raw data, for example when you want all the waveform values or event times in a time range, `ChanValue()` is a slow way to get the values and you would instead want to use the `ChanData()` command to fill an array with the data.

In addition to using maths functions on a single value to return a single result, you can also apply them to an array. It is usually much faster to use the array method than using a program loop. In this example we fill a 10000 point array with the result of a simple calculation. The first time by writing it out as a loop, the second time using array arithmetic :

```
var data[10000];

seconds(0, 1);                'zero the timer in hi-res mode
var i%;
for i%:=0 to 9999 do
    data[i%] := 5.0 * cos(i% * 0.001);
next;
var t1 := Seconds();           'see how long it took

Seconds(0, 1);                 'zero the timer
data[0] := 0;                  'first point is 0
ArrConst(data[1:], 0.001);     'set the same except first
ArrIntgl(data[]);              'form a ramp
Cos(data[]);                   'take cosine
ArrMul(data[], 5.0);           'form product
var t2 := Seconds();

Message("By hand %g ms\nArrays %g ms", t1 * 1000, t2 * 1000);
```

In this example we use three array arithmetic commands:

- `ArrConst()` – Sets an array to a constant value, or copies one array to another
- `ArrIntgl()` – Replaces each point by the sum of the points from the start of the array
- `ArrMul()` – Used to form the product of a pair of arrays, or to scale an array by a constant.

On my desktop machine, the written out loop takes 5.07 milliseconds for 10000 points. The array arithmetic version takes 0.099 milliseconds, some 50 times faster. There are further commands available for array arithmetic, see the in-software help section *Script language > Script functions by topic > Array and matrix arithmetic* for a full list.

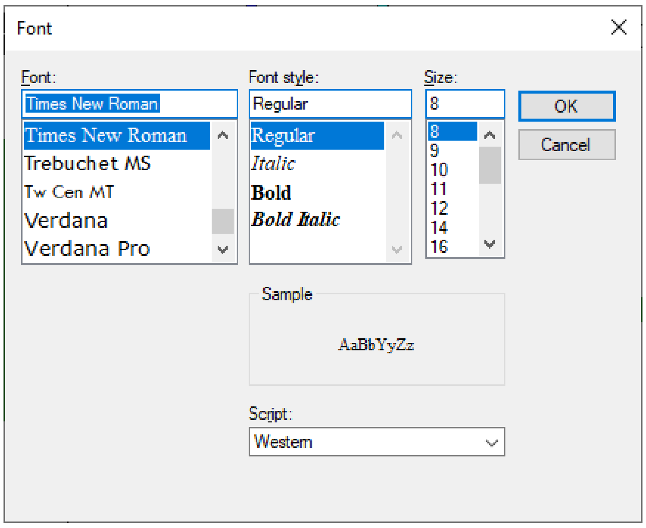
Back to contents

Recent Questions – *How to I change my font size?*

There are two view types as far as setting the Font is concerned: Text-based views and all the rest.

Non-text-based views (data views and grid views)

Click on the view you want to adjust and then use the *Set Font* button in your toolbar or the View menu Font… command to open the font settings dialog. The changes you make will apply to that view.



Font

Font: Times New Roman — Font style: Regular — Size: 8

Times New Roman, Trebuchet MS, Tw Cen MT, Verdana, Verdana Pro

Regular, *Italic*, **Bold**, ***Bold Italic***

8, 9, 10, 11, 12, 14, 16

Sample: AaBbYyZz

Script: Western

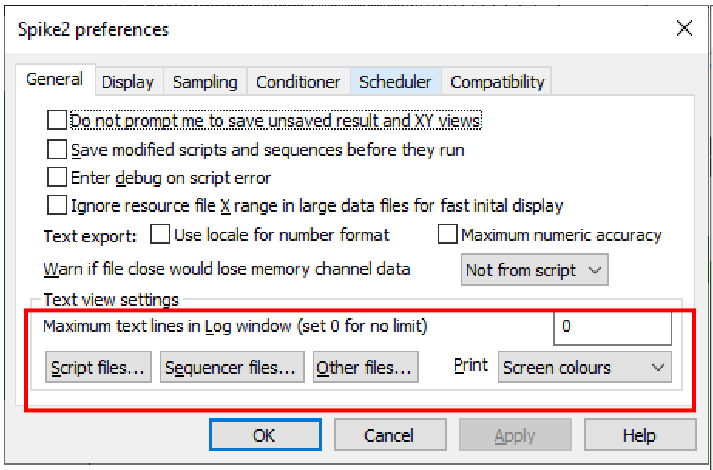
OK, Cancel

These settings are saved as part of the resource file associated with data files and in the grid view file. They will be restored each time you open the file (data files will only restore the font if the resource file is still present).

The default font for new views created without reference to any other view is currently Times New Roman 8 pt.

Text-based views (Script, output sequencer, Log and Text files)

From the *Edit Preferences* dialog, you can set the generic font settings for Script files, Output Sequencer files and Other text files (the Log view and Text files).



Spike2 preferences

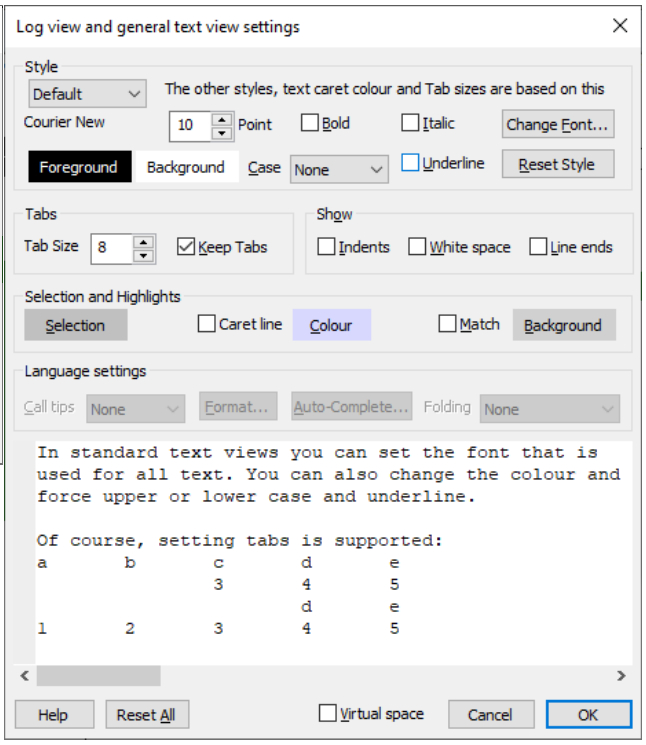
General | Display | Sampling | Conditioner | Scheduler | Compatibility

☐ Do not prompt me to save unsaved result and XY views
☐ Save modified scripts and sequences before they run
☐ Enter debug on script error
☐ Ignore resource file X range in large data files for fast inital display
Text export: ☐ Use locale for number format ☐ Maximum numeric accuracy
Warn if file close would lose memory channel data: Not from script

Text view settings
Maximum text lines in Log window (set 0 for no limit): 0
Script files… Sequencer files… Other files… Print: Screen colours

OK, Cancel, Apply, Help

Edit Preferences dialog



Log view and general text view settings

Style: Default — The other styles, text caret colour and Tab sizes are based on this
Courier New, 10 Point, ☐ Bold, ☐ Italic, Change Font…
Foreground, Background, Case: None, ☐ Underline, Reset Style

Tabs: Tab Size 8, ☑ Keep Tabs
Show: ☐ Indents, ☐ White space, ☐ Line ends

Selection and Highlights: Selection, ☐ Caret line, Colour, ☐ Match, Background

Language settings: Call tips None, Format…, Auto-Complete…, Folding None

```
In standard text views you can set the font that is
used for all text. You can also change the colour and
force upper or lower case and underline.

Of course, setting tabs is supported:
a       b       c       d       e
                3       4       5
                        d       e
1       2       3       4       5
```

Help, Reset All, ☐ Virtual space, Cancel, OK

Editor settings for all views except script and sequencer views

You may also access the individual text settings for Log, Text, Script and Sequencer views by selecting the view and using the View menu Font… command or the toolbar Font button. However, such changes last while the file is open unless you use the Apply to All button that appears when editing the view settings.

When working with Script and Sequencer views, you can set different fonts and styles for each category in the Style section. The Default style sets the basic font settings are the basis of the other styles.

Back to contents

Contact Us

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

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