

Contents

Welcome

Training days

Latest software

Here to help

Script Spotlight

- Sample Scripts

Spike2

- Spike sorting – Collision analysis
- Script – Burst analysis

Signal

- Cursor regions
- Script – Split frames

Scripters corner

- Idle functions

Recent questions

- How do I change the hardware voltage range?

Contact Us

Welcome

Thank you for downloading our July newsletter. UK restrictions have eased this month, but it will not greatly affect us at CED. We have remained open throughout and will continue to support your research wherever you are in the world. Moreover, we are excited to be at the Brain stimulation and SfN meetings later this year should things continue to improve, where we will be showing our latest products.

Spike2 version 10.11, 9.14 and 8.22 have all recently been released; you can update your copy, [here](#). Version 10.11 includes useful new script like PolyEval() to evaluate polynomials and PolyRoot() to find the roots of polynomials, as well as EditImageLoad() which loads an image from a file and saves it to the clipboard. All new updates also include several fixes to existing features to improve your experience.

This year's annual Neural Systems & Behaviour course at the Marine Biology Laboratory of The University of Chicago took place last month. Students were treated to an eight-week laboratory and lecture course focusing on the study of neural mechanisms underlying behaviour, perception, and cognition. CED were again happy to loan multiple 1401 systems and Spike2 software for the student's practical work. Even with a small fly outbreak, the students still thoroughly enjoyed performing experiments with the equipment and we are happy the course was once again a huge success.

We increased some prices of our products on the 1st July 2021. These are now available on our [website](#).

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 this button in our newsletters, it means a file or script relating to the section is available to download:



Latest versions of Spike2 and Signal

<u>Spike2</u>	<u>Released</u>	<u>Signal</u>	<u>Released</u>
Version 10.11	07/2021	Version 7.06	04/2021
Version 9.14	06/2021	Version 6.06	04/2021
Version 8.22	06/2021	Version 5.12a	02/2018
Demo	07/2021	Demo	04/2021

[Back to contents](#)

Here to help

We know access to laboratories has been erratic for the past year, but with the lockdown measures easing for some we hope that conditions will continue to improve. We are now able to offer site visits in the UK 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 – Sample Scripts

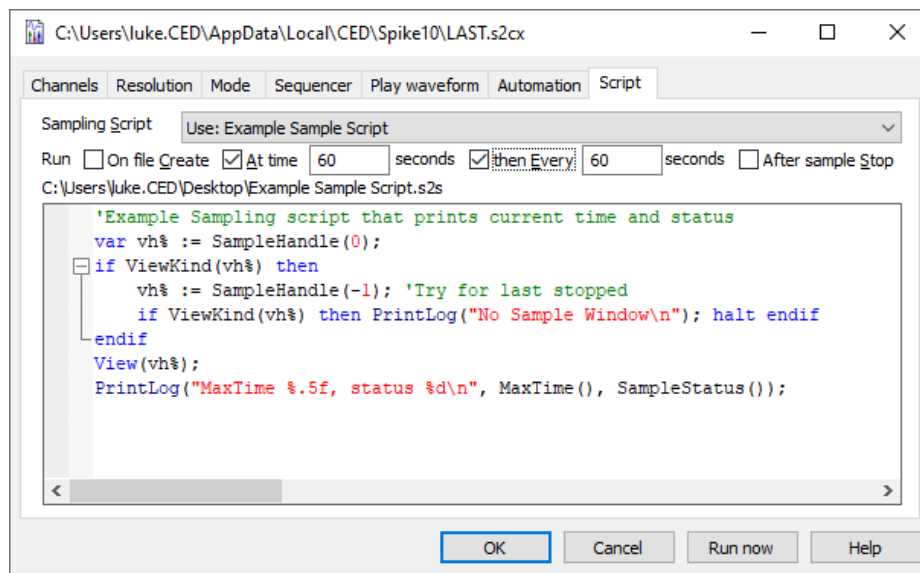
Sampling Scripts were new in Spike2 version 10.09. It allows you to associate a script file with the sampling configuration, as opposed to running independently from your data file. There are three ways to associate a script with your sampling configuration:

Mode tab Timed sampling – In timed sampling mode you can nominate a script that is run at the start of each timed data capture, or at the end of each timed data capture.

Mode tab Triggered sampling – In triggered sampling mode, each trigger can have an associated script that runs at the trigger time, or at the end of the associated data capture.

Script tab – The Script tab can set a script that runs at user-selected times: each time a data file opens, when sampling stops, at a particular time during sampling and at given intervals after that time. This script can run in addition to the timed or triggered scripts.

Typically, your script will run for a short time and end. Only one script can run at a time, so if another script is running, the script will not run. One use of this feature would be to save sections of a large data file periodically to allow for analysis elsewhere on a network. Another would be to run a particular stimulation or analysis protocol during a long sampling session.



An advantage of running a script periodically, rather than a script that runs with only scripted idle time for the user to interact with the data, is that between script runs there is no limitation on user actions. The user can even run other scripts providing they do not prevent the Sample Script from running. A disadvantage is that each time a sampling script runs, it starts with a clean slate of script variables. Therefore, any information that must be carried between script runs must be saved separately, usually using the Profile() script command or in a file.

Full details of Sample Scripts can be found in the software Help, along with a useful example script: Sampling data > Sampling configuration > Script > Sampling scripts.

[Back to contents](#)



Getting started with Spike Sorting – Collision analysis


If a channel contains more than one class of spikes, and the spikes are independent, it is inevitable that there will be collisions. Collision Analysis Mode allows you to match collided spikes to your existing templates.

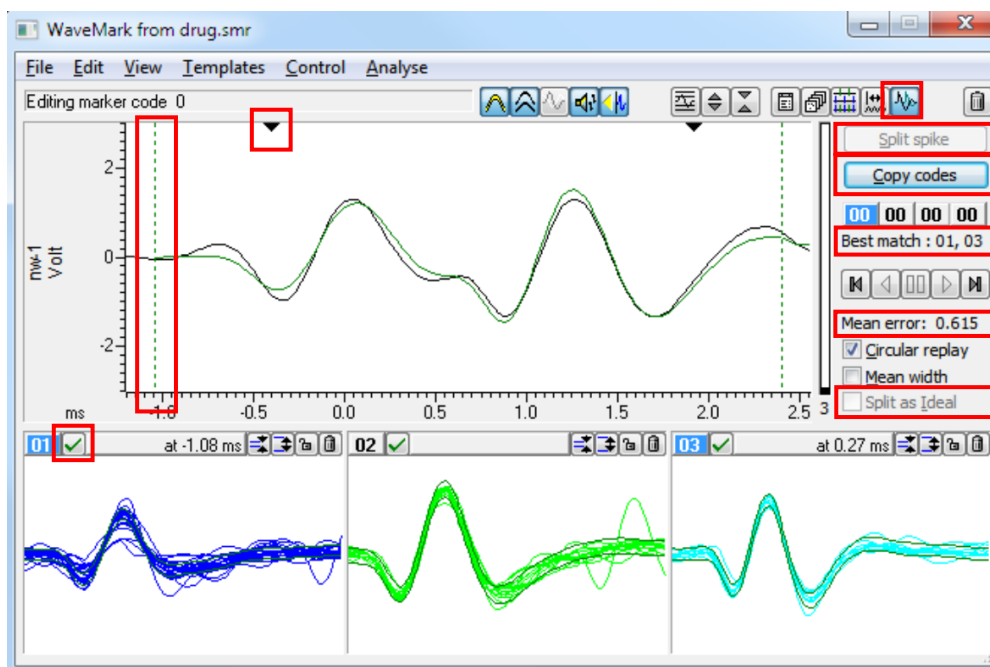
For the collision analysis to be useful, you should have followed the general guidance when creating your templates. It is important your templates are centred vertically on zero. If not, adding templates will create offsets and the results will not be useful. Each template should start and end around zero. Otherwise, you have likely not selected enough data when forming the templates. If templates start and/or end with a substantial non-zero value, the non-zero ends cause discontinuities in the best match waveform. You must also have sampled fast enough that the change between two consecutive samples is not so large that the matching algorithm fails.

The collision analysis method will work best in cases where you have a small number of well-defined templates. If you have many indistinct templates, you will be able to match just about any shape, but the results will be meaningless. We suggest that you use this method where there is an obvious collision and that you treat the results with great caution. Look at the firing patterns of the spike classes and check that the resolved collisions fit the patterns. It can be revealing to exclude a template from a collision to see how close the analysis is performed with another pair of spikes.

If the source channel is not a memory channel, you can modify the marker codes of spikes and mark collisions with multiple codes to indicate the spike templates that contributed towards each event. If it is a memory channel, you can replace a collision with an estimation of the spike shapes that generated the collision. You can use the *Analysis* menu > *Memory Buffer* > *Create Channel Copy* command to generate a memory channel from a WaveMark channel.



If you like the result of the collision analysis you can use the *Analysis* menu > *Save Channel* command to save the memory channel as a permanent channel.

To enter or leave Collision Analysis Mode, use *Analysis* menu > *Collision Analysis Mode* or click on the  button at the right-hand end of the toolbar. You will notice that some of the controls on the right of the window change, as does the main data display:



Main display – This shows the current spike overlaid with the best match combination of templates. The dashed vertical lines show the start and end of the template-forming region. The two black triangles mark the start and end of the *Important area*.

Important area – Change the important area by clicking and dragging the black triangles. If the area is smaller than the templates (the usual case), it is included in the best match. If it is more than twice the size of the templates, the best match area will lie within it. In an intermediate case, it marks the area we would like to match and some or all the *best matches* will lie in the region.


Template display area   – The button to the right of the template code excludes the template from the matching process. Use this if a collision is too close to another spike of the same class. You can exclude all but 1 template. Templates used to make a best match have the template code highlighted and list the offset in milliseconds into the main display at which the template starts.

Best match – This lists the template code or codes that were used to make the best match trace in the data display. The code for the earlier template is listed first. Templates are allowed to overlap the start and end of the displayed spike when making a match, but at least half the template points must be used in the match.

Copy codes – Use this button to copy the *best match* codes to the current spike. The first code is applied to the currently selected marker code. If there are two codes, the second is applied to the next marker code to the right (wrapping around to the first if necessary).

Mean error – This field displays the mean square error per point. The error is scaled by either the template width at each point, or by the mean template width over all the templates (see below).

Mean width – This option scales the errors by the mean width of all the templates (including excluded templates). This gives all points the same importance when calculating the best template combination. Think of this as "least squares" fitting. When this option is disabled, the template width is used point by point to scale the error. This gives



points with a small template width more importance than points with a larger template width. Think of this as "Chi-squared" fitting. If your templates have a wide variety of widths, you may get better results with this option enabled.

Split spike – This button is enabled if the spike source is a memory channel, and the current spike is worth separating (i.e., the spike matches two templates, or it matches a single template that does not overlap the x axis zero, laying entirely before or after).

This replaces the current spike with spikes aligned to the best match templates. How the replacements are calculated depends on the *Split as Ideal* check box. Each matching template generates one output spike. If the best match doesn't include the x axis zero, the original spike (less the best match waveform) is also preserved and *Split as Ideal* is assumed to be checked.

Split as Ideal – With this is checked (or assumed checked as above), the replacement waveforms are the template shapes. If this is not checked, the replacements share half the difference between the original data and the best match so that the sum of the two spikes recreates the original where the two created spikes overlap. In both cases, any data that is not available from the original data is set to zero.

[Back to contents](#)

Scripts: Spike2

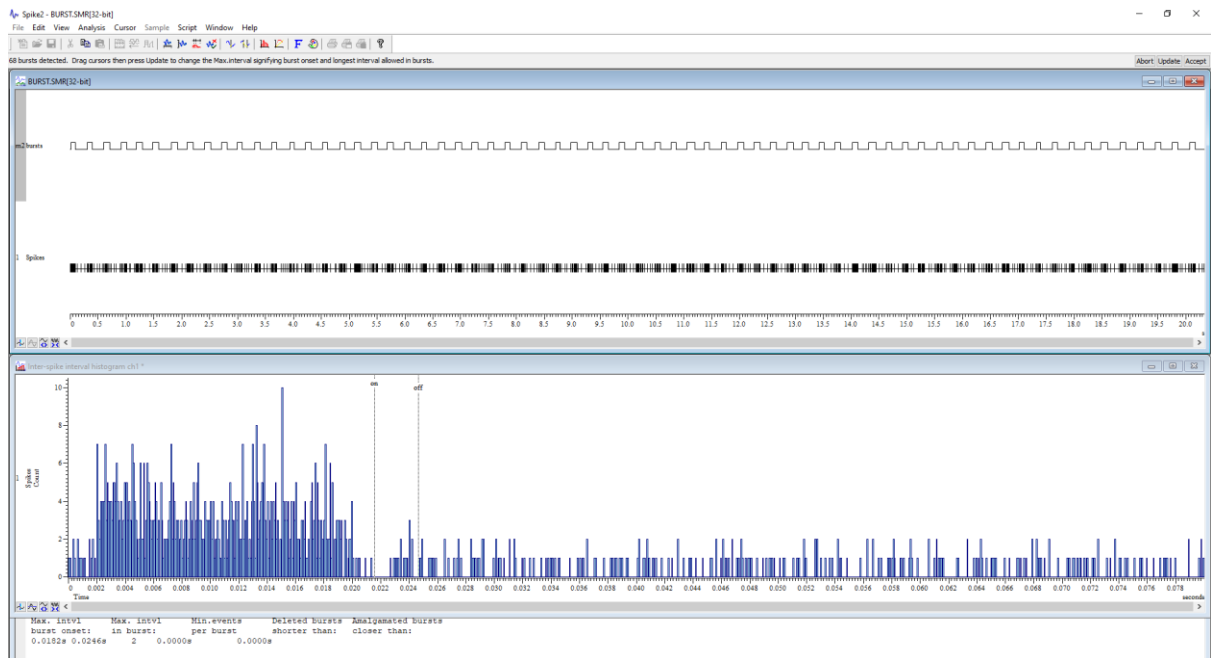
Marking spikes as events in Spike2 provides you further tools for analysing your data. Bursts of spikes may be further visualised and analysed in level channels. The bursts.s2s script included with your Spike copy (usually C:\Users\Username\Documents\Spike10\Scripts, or available from our [website](#)) allows you to group events into bursts and provides extra tools for quick analysis. The events to be analysed must be in an Event, Marker or Wavemark channel. If your data is recorded as a waveform, first use the ProcessEvents script (also in your scripts folder) or the peak detection facilities of Spike2 to create an event channel of spike times before running this script. When analysing Marker or WaveMark data, you can use the marker filter to display the unit or units of interest. The bursts.s2s script generates and analyses bursts in a single session or allows you to analyse a pre-existing burst channel created by the script or other methods.

Upon running the script, you are prompted to enter your criteria for what constitutes a burst. The minimum requirements for defining bursts are:

- The maximum interval between two events that signifies the start of a burst.
- The longest interval between two events within a burst (i.e. intervals longer than this terminate a burst).
- Minimum number of events in a burst.

You select values for these three parameters interactively, either using dialogs or by dragging cursors. The resulting bursts are shown in the data file as a level channel. You can accept the burst data or go back, adjust the parameters, and retry until you are satisfied with the results. A *Revise bursts* option allows you to amalgamate bursts that are very close together or delete very short bursts.

After the burst channel has been saved, you can generate a table of statistics for each burst and/or summary statistics relating to all the bursts in the chosen time range. The burst statistics can also be shown graphically. Plots include: distribution of burst durations, inter-burst intervals and number of events/burst as well as burst duration, inter-burst interval, period, and events/burst plotted as a function of time. These files can be printed or saved to disk as required. The data files BURST.smr and BURTS.smr included with your Spike2 copy (usually C:\Users\Username\Documents\Spike10\Data) are available to practice with the script.



The analysis functions available from the script are:

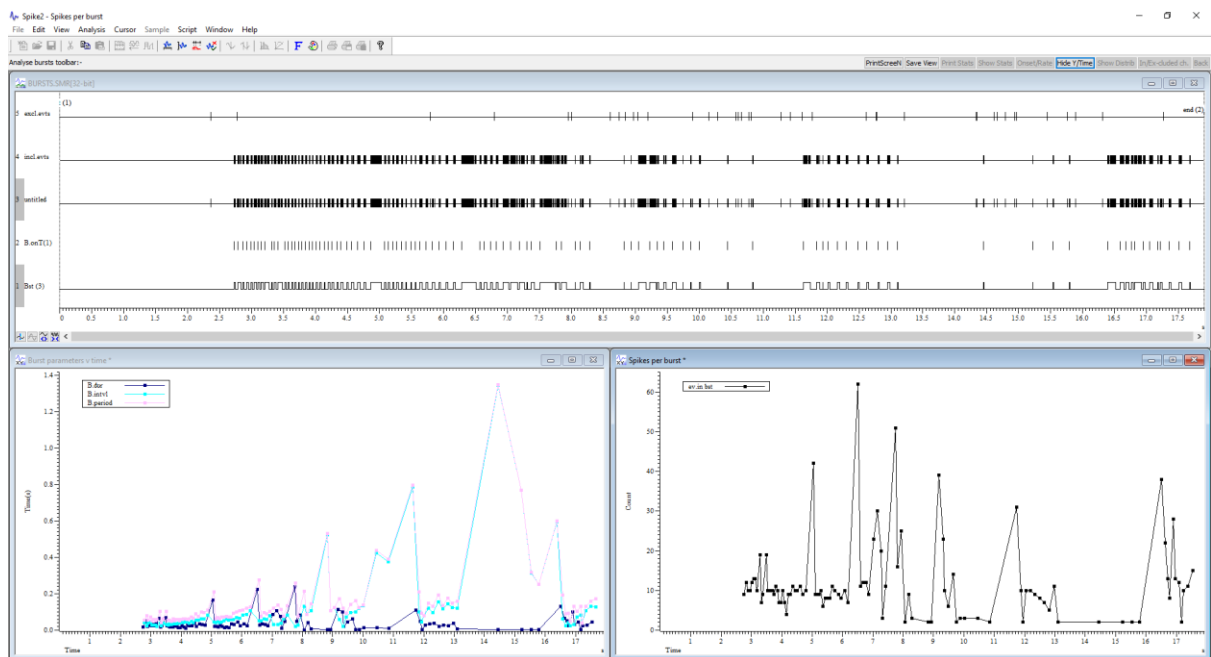
Show Stats – Select which statistics to display and generate a table of statistics. Note that there are no valid inter-burst time and period for the last burst in the time range because the analysis always terminates with the last burst.

Onset/Rate – Add a channel to the data file containing the onset times of each burst in the time-range displayed as lines, or alternatively as a burst rate histogram.

Y/Time – Displays two XY plots in the lower half of the screen. The left-hand frame shows Burst duration, inter-burst interval, and burst period plotted on a time axis. The right-hand frame shows events per burst over time.

In/Ex-cluded ch – Creates event channels in the data file that show only the events that occurred in bursts or only the events that were not in bursts, or both.

Show Distrib – Hides the time view and displays histograms of the distribution events/burst, burst duration and inter-burst intervals.



A full pdf help guide is included alongside the script for your reference.

Signal *What do the cursor regions show and how do I use them?*

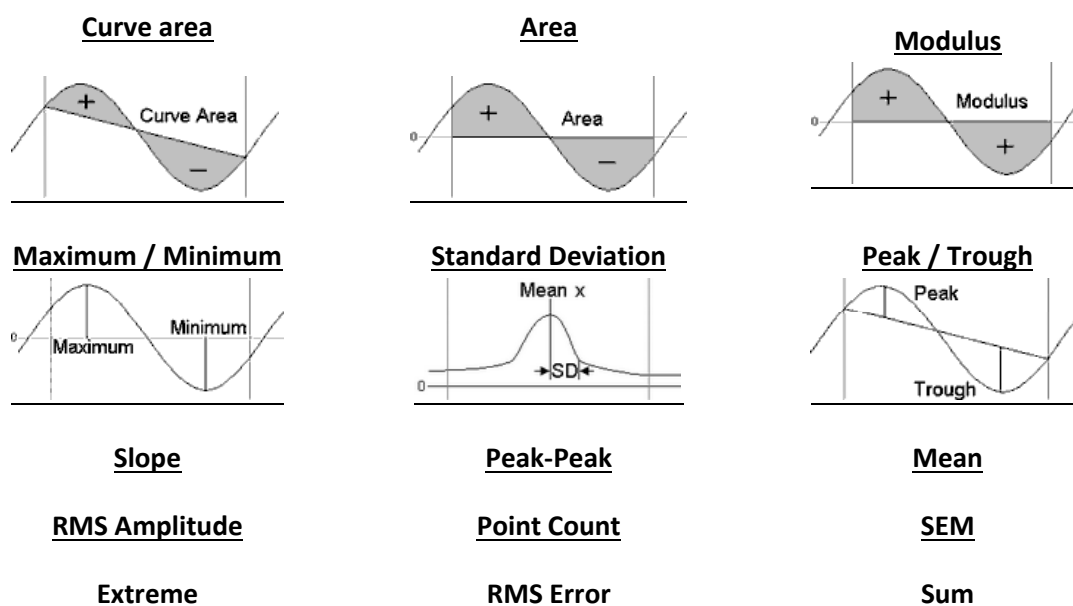
The cursor region window calculates values for data regions between cursors. It is a handy source for common measurements of your data. Access the window through the *Cursor* menu > *Cursor Regions*.

Cursors	0 - 1	1 - 2	2 - 3	3 - 4
Time (ms)	6.4268585	7.4340528	3.0215827	4.2685851
1/Time (Hz)	155.59701	134.51613	330.95238	234.26966
1 ADC 0	0.023574829	0.13414885	0.31103516	0.024879092
2 ADC 1	0.068740845	0.083457545	0.51204427	0.063941592
3 ADC 2	0.048217773	0.10632967	0.22900391	0.023716518
4 ADC 3	0.043334961	0.051847759	0.11263021	0.07405599
<input type="checkbox"/> Zero region	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Mean	< [Slider] >			

One column can be designated the *Zero region* by checking the box and selecting the column with a radio button. The value in this column is then subtracted from the values in the other columns. The field at the bottom left sets the measurement method; click it for a full list. The effect of the selected measurement type depends on the channel type (waveform or marker) regardless of the channel display mode. Cells with no cursors or data are left blank.

The region set by a pair of cursors is the data starting at the first cursor up to, but not including the data at the second cursor. The dialog fields are updated when cursors move or channel data changes. Using the *Customise display* dialog (accessed by right clicking the data), you can hide/show channels from your data and subsequently the cursor region window. If a channel is not displayed in the cursor regions window, ensure it is enabled in the customise display dialog and expand the cursor regions window by clicking bottom right corner and dragging to show more channels.

The available measurements for waveform channels are:



For marker channels, only Mean, Sum, Maximum, Minimum, Peak-Peak and Extreme are applicable. Similarly, only Curve Area, Mean, Area, Sum, Modulus, Maximum, Minimum, Peak-peak, Extreme, and Point Count are applicable for Real Marker channels. Only Point Count is applicable for Idealised Trace data. If you select other measurements

the result is a blank field. Full details of the calculated measurements can be found in the software help: Cursor menu > Cursor Regions > Cursor region measurements.

In addition to calculated measurements, it is also possible to display the Y values at the position of any cursors in the current data view. Open the *Cursor* menu > *Display Y values* window to view them. Like the cursor regions dialog, columns for cursors that are absent or for which there is no data are blank.

Cursors	Cursor 0	Cursor 1	Cursor 2	Cursor 3	Cursor 4
Time (ms)	-5.6354914	0.79136713	8.2254199	11.247003	15.515588
1 ADC 0	0.02788004	0.084922297	0.68110551	0.071105225	0.015028992
2 ADC 1	0.0073242188	0.15214001	0.95876424	0.20150091	0.021211546
3 ADC 2	0.0040163182	0.090332031	0.67124035	0.14420105	0.0086239602
4 ADC 3	0.047095136	0.020268941	0.023213851	0.15194095	0.0048828125
<input type="checkbox"/> X Zero	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="checkbox"/> Y Zero	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

The values displayed depend upon the channel type and display mode. There is an entry in the table showing the time for each cursor, plus entries for each channel displayed. The displayed values are:

Waveform – The y axis value of the nearest data point that is within one sample period of the cursor, or nothing if there is no data point close enough. Waveform measurements are not affected by the drawing mode.

Marker as Rate – The height of the rate bin that the cursor crosses. If the cursor lies on a bin boundary, the cursor is considered to lie in the bin to the right.

Marker – The time of the next marker at or to the right of the cursor.

It is possible to choose both an X zero and Y Zero column in this window. If checked, the cursor marked with the radio button is taken as the reference, and the remaining cursor times (for X) or Y values are given relative to it. The values for the reference cursor are not changed.

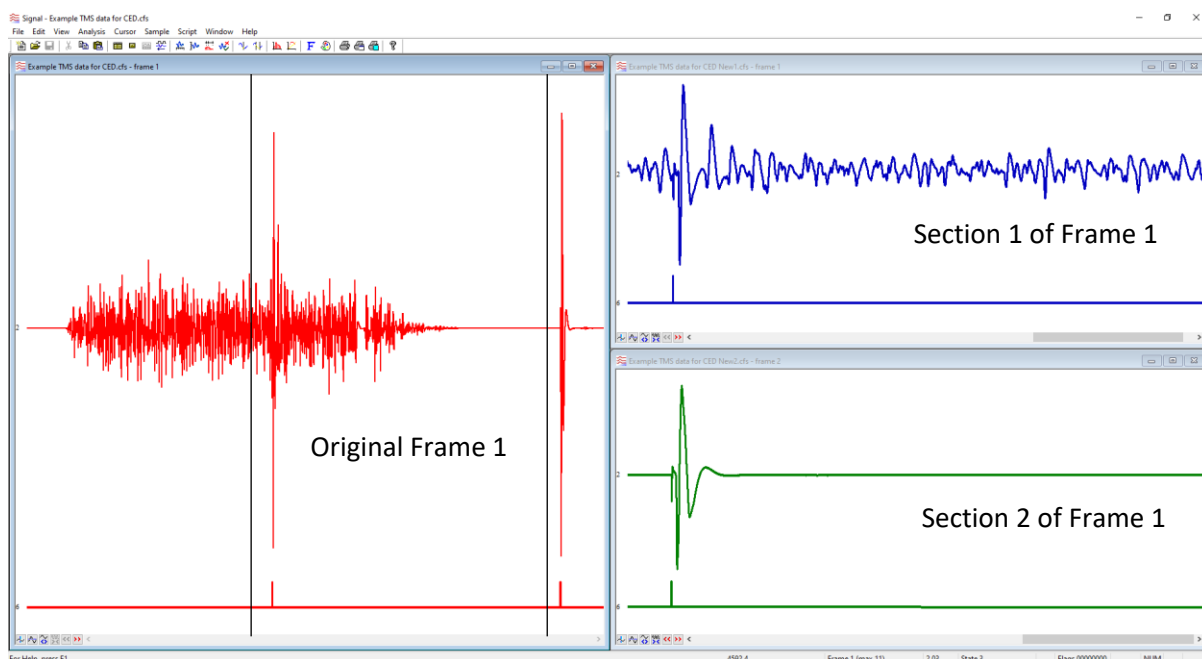
You can select areas of either window by clicking them. Hold down the *Shift* key for extended selections. You can select entire rows and columns by clicking in the cursor and channel title fields. Use the *Ctrl* key to select non-contiguous rows and columns.

To copy selected rows and columns to the clipboard, use *Ctrl+C* or right-click in the values window and use the *Copy* command in the popup menu. If you use the *Log* command (*Ctrl+L*) the selected text is copied and pasted directly into the log window in one operation. You can also print the selected portions of the window by with *Ctrl+P* or right-clicking and using the *Print* command in the popup menu. The *Font* command (*Ctrl+F*) changes the window font.

[Back to contents](#)

Scripts: Signal

Previously a user with a paired pulse recording asked us for a way to split their frames in two up so that each trigger and response is stored in a separate frame. The attached script [SplitFrame.sgs](#) allows the user to select the start times of the two sections and optionally offset the start of new frames to time zero.



Setup

Area 1 start (s)

Area 2 Start (s)

Offset to zero

List ▼

Frame state

Frames

Upon running the script, you will be asked to select your desired file. A new dialog will appear for you to select the start times of the two sections. Alternatively use the new cursors on your data to select the times. These start times will apply across all frames, so ensure you scan through your frames to ensure you are picking applicable times.

Tick *Offset to zero* for the start times of the new data to begin at 0s, otherwise the frames will begin at the selected start times for the relevant section. You may choose to split *All Frames*, the *Current Frame*, all *Tagged Frames*, all *Untagged Frames*, *Frames of State XXX*, or a *Frame List*. For the last two options, extra dialog boxes are enabled for you to denote the state or list of frames separated by a comma (e.g. "1,2,3,4,5"). Click OK when you are ready, and the

script will generate a new data file for you and display it alongside the original data. The new data frames have an added comment (right click > *File information* > *Frame information* tab) that states which section of which original frame the data relates to.

This script is not strictly for separating data based on trigger times, such as times of a paired pulse in the example above, it may separate data around any time range and could be adapted to automatically split frames into sections with more than two triggers. This script is an example of scripting a solution for a user, which may have a broader use for others. If you have an idea of a script or have a particular task you cannot achieve with the existing tools, get in touch with us at Marjorie@ced.co.uk and we can produce a script for you to achieve your needs.

[Back to contents](#)

Scripters Corner – Idle functions

When writing scripts it is often the case that you require a function to continually operate in the background, whether to monitor changes or conditions on-line, or for off-line analysis to check when cursor positions are updated and new values need to be called. Idle routines are part of the Toolbar and Dialog script functions in Spike2 and Signal.

Toolbar Idle

Specifically for the Toolbar, you may consider the idle routine as an invisible button; when a Toolbar containing the idle routine is active in a script, the idle function is repeatedly called whilst the script is waiting (idle), possibly checks if a given situation such as a new event has occurred, and if so branches to another function.

To enable the idle routine in a toolbar, we use the `ToolbarSet()` function when creating the toolbar. With the `item%` argument set to 0, we define the idle function `ToolbarSet(0, "", idle%)`. The `Idle%` function is then defined elsewhere in the script:

```
Func idle%;
`Your test and branching code goes here
return 1;
end;
```

You will notice we return a positive integer for the idle function. If it returns 0 or a negative integer the `Toolbar()` function returns to the caller, passing back the button number (`item% 0` for the idle function), closing the toolbar. If it returns a number greater than 0, the `Toolbar()` function does not return, but waits for the next button. Most idle functions return 1, though they may test a condition and return 0 to terminate:

```
Func idle%;
...
return AreWeDone() ? 0 : 1;
end;
```

Here we have used the ternary operator (`?`), which is defined as: `expr1 ? expr2 : expr3`. The result is `expr2` (0) if `expr1` (`AreWeDone()`) evaluates to a non-zero value, thus terminating the toolbar, and `expr3` (1) if `expr1` evaluates to zero.

There are a couple of scripts readily available to help you get started with creating a toolbar with an idle function. The script `onskel.s2s` gives you a template to begin making your own on-line script, whereas the `ToolMake.s2s` Spike2 script and `ToolMake.sgs` Signal script discussed in a previous issue are included with your user data folder helps you build your own toolbar from scratch: usually `C:\Users\Username\Documents\Spike2\Scripts` or `Signal\Scripts`.

The example script, [FindIntersect.s2s](#), sets up a vertical cursor on channel 1 and then automatically places a horizontal cursor at the first intersect point where the data crosses the vertical cursor in the current time range. If the cursor is re-positioned, the idle routine will update the horizontal position in response. You can replace the example code in this Idle routine with your own analysis functions.

Dialog Idle

The idle routine in a dialog is different in that the system will only be in idle whilst the dialog is active. To add the idle routine to a Dialog, we use the `DlgAllow()` function. This is called after `DlgCreate()` and before `DlgShow()` in order to enable dialog idle time processing. However, it is often used in conjunction with a Change routine. The full script function is:

```
Proc DlgAllow(allow% {,func id%(){, func ch%()});
```

Where `id%`() and `ch%`() link to the respective idle and change functions. The idle function is called repeatedly in system idle time unless the changed function is called. The `id%`() and `ch%`() functions are defined elsewhere in the script, for example:

```
Func dlgidle%();  
'Your code here  
return 1;  
end;  
  
Func changed%(item%);  
'Your code here  
return 1;  
end;
```

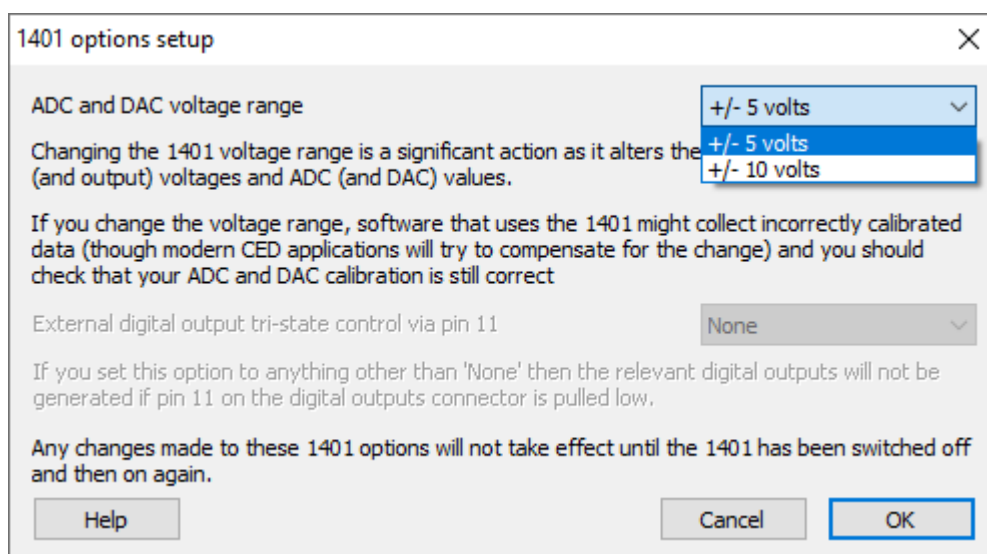
With the change routine enabled, the function is called each time the user changes a dialog item with the `item%` argument set to the changed item number. There is an initial call with the argument set to 0 when the dialog is about to be displayed.

The example script, [Example dialog timer self-destruct.s2s](#), makes use of both a idle and change function in order to display a dialog that automatically exits after a set amount of time.

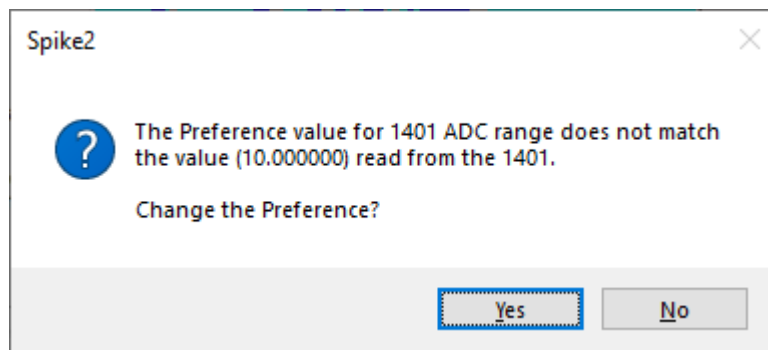
[Back to contents](#)

Recent Questions – *How do I change the voltage range of my 1401?*

Modern 1401 interfaces (Micro1401-4, Micro1401-3, Power1401-3 and Power1401 mkII) can be set to use +/-5V or +/-10V inputs and outputs via the *Try 1401* diagnostic software, installed alongside Signal and Spike2. Access the software in the Spike2/Signal folder through your windows *Start* menu. With a 1401 connected and switched on, run the *Try 1401* application and select *1401 options* from the *File* dropdown menu. This opens a dialog where you can select the input/output range to set for the 1401.



Change the voltage range and click *OK*, then switch your 1401 off and on again for the changes to take effect. Spike2 and Signal detect the range of connected Power1401s and Micro1401s automatically. You will be warned if the software detects a conflict between the user settings and connected hardware; it will prompt you to update the user settings.



The voltage range affects scaling in the sampling configuration and DAC output values in the output sequencer. It has no effect on scale values in previously sampled data files.

[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

All Trademarks are acknowledged to be the Trademarks of the registered holders.
Copyright © 2021 Cambridge Electronic Design Ltd, All rights reserved.