

Contents

Welcome

Training days

Latest software

Here to help

Script Spotlight

- #include



- Getting started with spike sorting
- Script – Blood pressure analysis



- Cursors
- Script – TMS Recruitment curve config setup

Scripters corner

- Dialogs

Recent questions

- Horizontal cursors as thresholds

Contact Us

Welcome

Thank you for downloading our June newsletter. After a very wet May in the UK, we have been lucky to finally see some sun now that June is here. Spike2 version 10.10 has recently been released; you can update your copy, [here](#).

We are increasing some prices of our products from the 1st July 2021. These will be available on our website in due course.

Our software engineers are working on a Talker for OT BioElettronica devices. Initially we aim to support the 64 channel EMG Sessantaquattro device, with plans to add support for the 400 channel EEG/EMG Quattrocento once development has progressed.

Spike2 10.10 includes an importer for XDF (eXtensible Data Format) files. This offers us a method of obtaining data from the LabStreamingLayer (LSL), a multi-stream protocol capable of collecting data online from many devices at once. This is potentially another target for a Talker, and we are interested to know if you or a colleague is either already using LSL, or perhaps using some of the [hardware that it supports](#). Please get in touch with us at Marjorie@ced.co.uk.

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 10.10	05/2021	Version 7.06	04/2021
Version 9.13	02/2021	Version 6.06	04/2021
Version 8.21	02/2021	Version 5.12a	02/2018
Demo	05/2021	Demo	04/2021

[Back to contents](#)

Here to help

We know access to laboratories has been erratic for the past year, but with the current lockdown measures easing for some 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 – #include

There are times you will want to re-use some of your user-defined procedures and functions in multiple script projects. This can be done by copying and pasting the code, but it can be much more convenient to use the #include command to save yourself extracting all relevant variables. This lets you access procedures and functions in other script files from within your current project using: #include "filename" where filename is either an absolute pathname "C:\\scripts\\MyFuncs.s2s", or a filename enclosed within angle brackets <MyFuncs.s2s>. These two examples use a Spike2 script file with the file extension .s2s. Signal script files use the .sgs file extension.

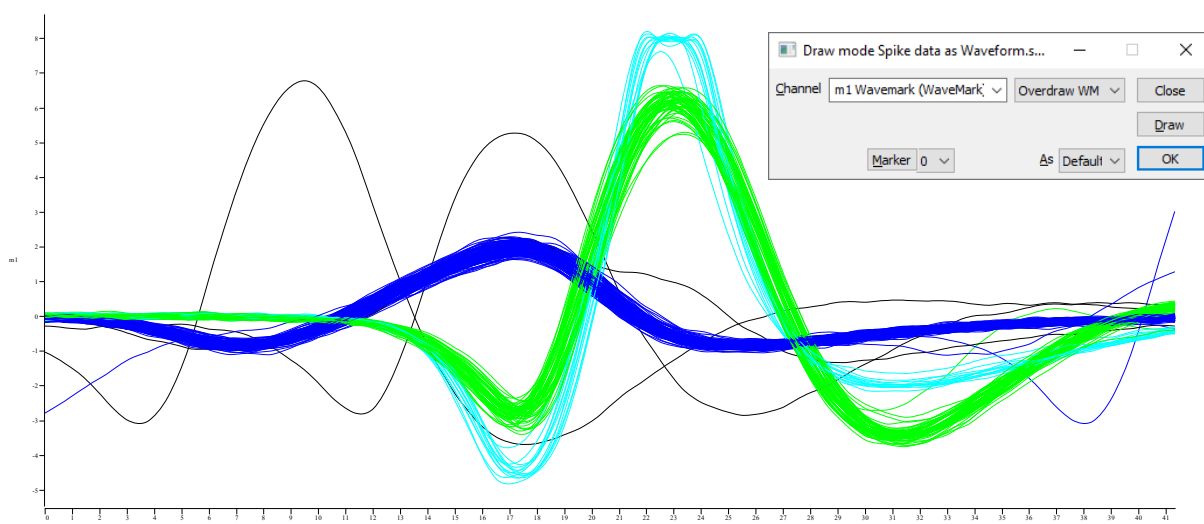
If only the file name is provided, the script expects the script file to reside either in the same folder as the open script, in the "current" folder, or in the include folder in your Spike2 or Signal installation folder (usually C:\\Users\\user\\Documents\\Spike10 or Signal7\\Include. The in-software help topic *Script language > Script language syntax > Include files* has the full details of all the places searched. It is usually a good idea to have all your #include commands at the start of your script so others who use the script understand that multiple script files are being used. Many of the scripts in the Spike10\\scripts and Signal7\\scripts folder make use of #include to read functions from a script file called GHutils. This file contains several useful functions for you to also make use of in your own scripts.

[Back to contents](#)

In this issue we introduce a useful method for discriminating spikes. We have discussed how to form templates and create WaveMarks offline in a previous issue; this method is useful for differentiating spikes from existing WaveMarks, outside of the template formation window.

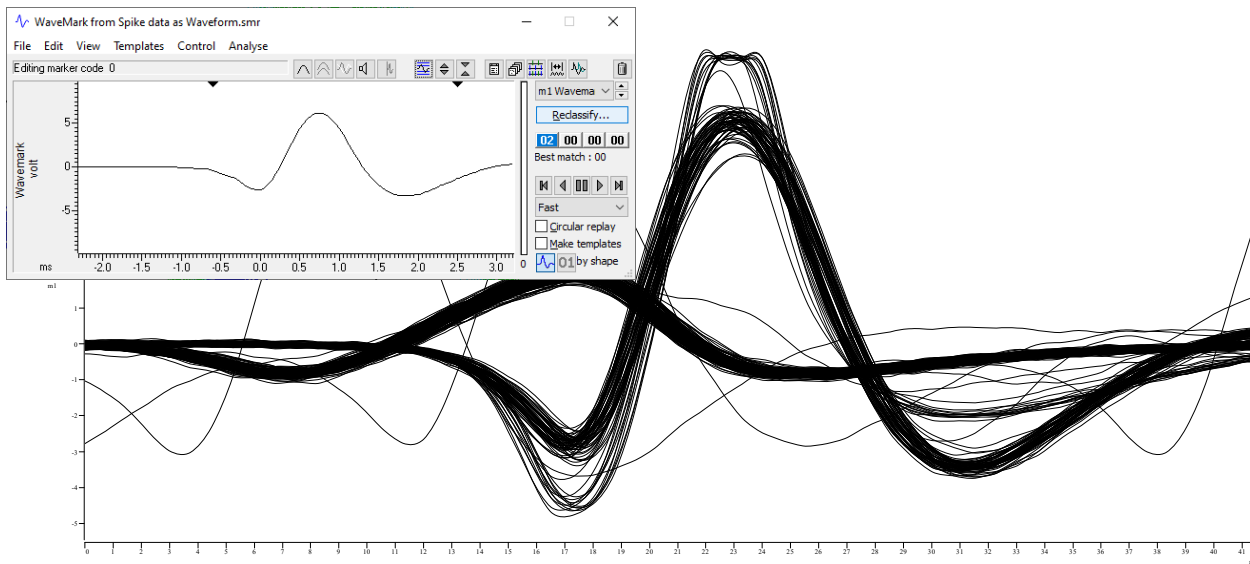
The channel draw mode Overdraw WaveMarks draws WaveMark data as superimposed waveforms. Channels drawn in this mode are moved to the top of the data window and separated from the x axis (which does not apply to them) by a hatched bar. When viewing data offline, the displayed x-axis range determines which spikes are overdrawn; if the x-axis display starts from 30 seconds and ends at 45 seconds, only data in that 15 second window is overdrawn. If this mode is used during data capture and the screen is scrolling to show the latest data, new WaveMark events are added, but old events are not erased (to stop the display flickering). In this case, click on the x axis scrollbar thumb to force an update, clearing old spikes.

While WaveMarks are overdrawn, it is possible to draw a line to intersect your spikes and assign a new marker code. Hold down **Ctrl+Alt** and *left click* to drag a line through the spikes you want to identify. Once you have started to drag you can release the **Ctrl+Alt** keys. If you continue to drag and hold down the **Shift** key, the line is constrained as horizontal. On releasing the mouse, a dialog opens in which you can set codes for the intersected events. If the channel has multiple traces the line must only extend over a single trace. This cutting method to change the marker codes is also available when the spikes are drawn in Waveform and WaveMark drawing modes.

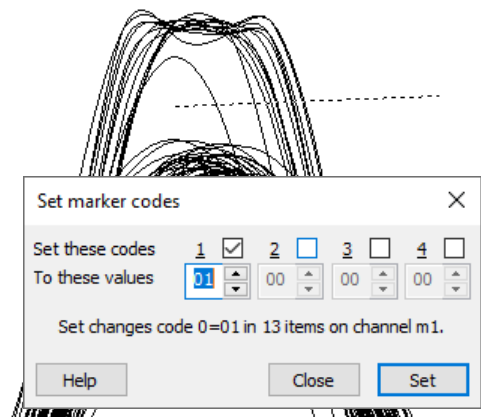


We have data available for you to practice this method yourself. Open the file Spike data as Waveform.smr from your Spike2 data folder installed with your copy of version of Spike2 (usually C:\Users\user\Documents\Spike10\Data).

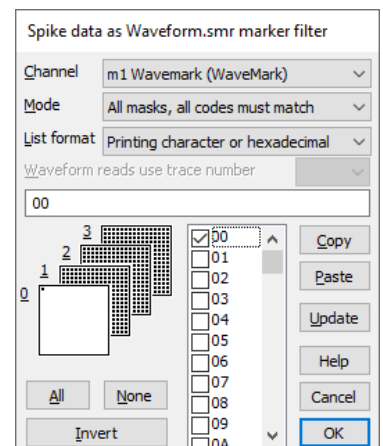
- Create a copy of the WaveMark channel 4 (right-click and *Copy as a memory channel*).
- Double click the new channel to zoom it and hide the others.
- Change the draw mode to Overdraw WM (right click the data and select *draw mode*).
- Show the full x-axis range (double click x-axis and select *show all*).
- Right click the data and select *Edit wavemark*.



- Untick *make templates* and select *Reclassify* to change all marker codes to 00. Your overdrawn WaveMark data should now all be shown in one colour (black is the default colour for code 00).
- Hold *Ctrl+Alt* and *left click* and drag a line to intersect the spikes you want to alter. Release the mouse to open the *Set marker code* dialog.
- Change the marker code in this dialog to another unique code.
- Your intersected spikes should now be showing as a new colour having been assigned the new code. We can now hide these using the *Marker filter*. Right-click the data and select the *Marker Filter*. Select *None* and then re-tick code 00, press *Update*.
- Each time you change a marker code from 00 for intersecting spikes, they will disappear allowing you to focus on the remaining spikes.
- Re-tick the codes in the marker filter at any time (and *Update*) to show those spikes again.



This method will not be useful 100% of the time; as we mentioned in our previous newsletter there is no one-fit method for spike sorting and you will need to tinker to find the method that works best with your data. For example, try opening the file *Extracellular Spikes .smr* from your *Spike2* data folder and attempt this same method. The larger quantity of spikes with only small amplitude changes makes this cutting method to separate them much more difficult.



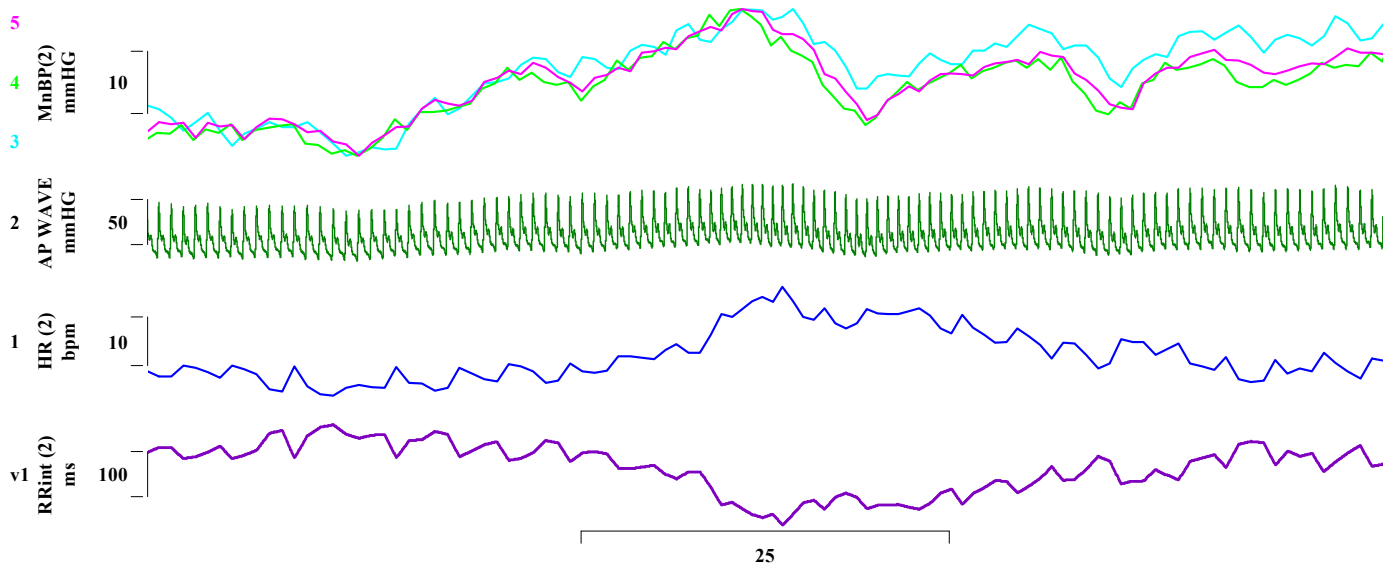
When using the Overdraw WM drawing mode to superimpose all the spike shapes in a time view, you can also find any individual spike by right clicking on it and selecting the *Find with Cursor 0* function in the pop-up menu. If you have followed the above steps, show all your channels again (double click the overdrawn channel or right click > *Show/Hide channels*) and the WaveMark in your original WaveMark channel will be highlighted by cursor 0.

[Back to contents](#)

Scripts: Spike2

Our script writers have recently updated the HR_BP.s2s script available on our [website](#). This script allows you to:

- Create additional channels in your data file derived from a blood pressure trace. The extra channels available are: Heart rate, RR interval, diastolic bp, systolic bp, pulse bp, true mean blood pressure and "conventional" mean BP, (i.e. diastolic + x% pulse where x is a value that you choose, typically between 30% and 40%).
- Process a single bp trace online or multiple bp traces offline.
- Simulate online recording by replaying an existing time view using the REPLAY button.
- Display a table of "current" blood pressure statistics that updates every second online or during REPLAY. (Please note, this is only available during *Replay* if the 1401 interface is connected and switched on.)
- Offline, you can generate a table of beat times and amplitudes for each channel created. This table can be saved or pasted into spreadsheet software for further analysis.



This script operates on the front view; ensure you minimise extra data files and bring the correct time view to the forefront (the title bar should be coloured) before running the script. The script has also been created so only channels calibrated in mm of Mercury and with a units label that includes the string "Hg" will be available for selection in the *HR_BP* dialog. If your BP channel does not have these units, double click the channel title in the Y axis to set the correct scaling and ensure that the label for the channel units includes "Hg". Run the script and you will be presented with a toolbar of 6 buttons:

HELP | HR_BP | REPLAY | Y-RANGE SET | PRINT | QUIT.

Full details of the controls and setup can be found by accessing the *Help* from within the script.

Default Y-axis Ranges			
	Low	High	
Heart rate	20	200	bpm
Systolic BP	100	200	mmHg
Diastolic BP	50	100	mmHg
Pulse BP	10	80	mmHg
Mean BP	70	120	mmHg
Dia. +33% pulse	70	120	mmHg
Raw BP	60	200	mmHg

Conventional Mean BP

Diastolic + 33 % pulse pressure

Cancel OK

Offline Analysis

To add blood-pressure related channels to your data, click the *HR_BP* button and select the following items in the dialog:

- Source channel – the blood pressure channel from which the data will be measured.
- Amplitude – the threshold for heartbeat detection. This is the fall in BP after a peak (systole) and the rise in BP after a trough (diastole) that must occur for that peak or trough to be registered. You need to set this lower than the minimum pulse pressure but greater than other oscillations in BP such as a dicrotic notch to avoid detection of multiple peaks per heartbeat.
- Min. heart rate – used to set the timing parameters for the active cursor search for bp peaks and trough. Set this value lower than the lowest frequency you expect during the study.
- Time range – the start and end time of interest. Alternatively, drag the cursors labelled *Start* and *End*.

Select channels to create/modify by checking the relevant checkboxes. For each channel that you select, check the *Optimise* checkbox to optimise the Y-Range of that channel, overriding the default Y-scale set via the *Y_Range Set* button.

There are several options for displaying heart rate: Instantaneous frequency (the reciprocal of the interval between systoles plotted at the time of a systole) or mean frequency (of systoles) over a time range specified in the dialog. Each channel can be displayed with its own Y-axis, with the BP channels overdrawn (diastolic, systolic and Mean BP) or all BP channels overdrawn on the raw BP trace (as in the example above).

Set up BP/Heart rate channels

Search parameters

Raw BP: 2 AP WAVE (Waveform)

Amplitude: 15

Minimum rate (bpm): 20

Time range

Start: Zero 0

End: MaxTime 119.999

Heart rate

Draw Mode: Inst frq (joined)

Channels

Channel	Optimise
<input checked="" type="checkbox"/> Heart rate / RRintvl	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> Systolic BP	<input type="checkbox"/>
<input checked="" type="checkbox"/> Diastolic BP	<input type="checkbox"/>
<input type="checkbox"/> Pulse pressure	<input type="checkbox"/>
<input checked="" type="checkbox"/> True mean BP	<input type="checkbox"/>
<input type="checkbox"/> Diastolic + 33% pulse	<input type="checkbox"/>
<input checked="" type="checkbox"/> Raw Blood pressure--	<input checked="" type="checkbox"/>

Display

Separate Y-axes: ☒

TIP: Only plausible BP channels, that is, with units of <mm.Hg> are included in the channel list.

Close Update

Online Analysis

You can also use this script to process a blood pressure trace online. Simply start sampling data or open a new data file ready for sampling before clicking on the *HR_BP* button in the script toolbar. The display section of the dialog has an additional checkbox enabled while sampling is in progress. Check it before clicking on *Update* to display a table of the most recent values of the channels generated by the script. These values are mean values of each channel and are updated approximately every second (indicated by flashing asterisk).

When you stop sampling a data file with channels added by this script, these channels will be incorporated into the sampling configuration. They will appear automatically when you sample using this configuration in future. Thus, you do not need to run this script to create the channels each time. However, it is useful to have the script running and the dialog open while recording so that you can adjust the threshold amplitude, y-axis ranges or display mode whenever you wish.

[Back to contents](#)

Normally, Signal cursors are static; they stay where they are put. Using the active cursor dialog, cursors in file and memory views can be made active; they will move to the position of a data feature, if it can be found. This search is repeated whenever the view data changes, cursor 0 is iterated, or the view switches to a different frame or the frame buffer. This repositioning is carried out in order of cursor number so cursor 2 can reliably make use of the current position of cursor 1 but not vice-versa. Active cursors can be used as a simple way of quickly finding features within your data; they are also very powerful tools for extending the capabilities of analyses that generate measurements from data files.

Signal can search for and detect features in waveform data and step to digital markers using the different search methods. These include, but are not limited to:

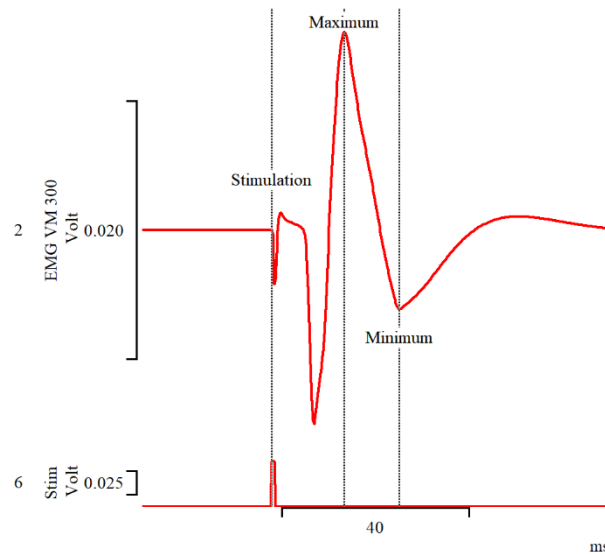
- Peaks and troughs
- Maxima and minima
- Threshold crossings, data within and data outside thresholds
- Slope measurements
- Percentage repolarisation
- Data points
- User expressions

Up to ten cursors (0 – 9) are available in a frame and individual cursors can search any channel for features of interest. Vertical cursor 0 is special. It always exists and cannot be deleted, but it can be hidden (and will normally be initially hidden by Signal). Unlike the other cursors, when cursor 0 is active it is designed to iterate through the data to repeatedly find features in the waveform. Whenever cursor 0 is moved, all the other active cursors will recalculate their positions in order of cursor number. To hide cursor 0, right-click on it and select Hide in the cursor 0 context menu. Script users can cause a nominated range of cursors to search with the `CursorSearch()` command.

As Cursor 0 acts as the iterator cursor, it is nearly always used to find a common reference, such as a stimulus or marker, around which other features can be found by additional active cursors. Cursors are created and Active modes set from within the Cursor menu.

Active cursor positions are either valid or invalid; invalid cursors have an exclamation mark at the end of the label. The cursor position is invalid if the active search fails and the, "Position if search fails" field is empty or does not contain a valid expression. Expressions that use invalid cursor positions are also invalid. The XY Trend plot and Measurements analyses reject points that come from invalid measurements. Cursor positions are made valid by any operation that moves them to a specific place such as dragging. If a search results in an invalid position, the cursor is not moved. Script users can test if a cursor has a valid position with the `CursorValid()` command.

Cursors may also be labelled to show the position (x-axis), cursor number, both position and number, or a user-defined label. This last option is useful to label features in your data. For example, this evoked response from a transcranial magnetic stimulation (TMS) device has the time of the stimulation labelled, as well as the maximum and minimum of the response:



In the above example, the stimulation pulse was recorded as a waveform. To find the features, we used cursor 0 to locate stimulations by stipulating a rising threshold. The Maximum and Minimum were found by stipulating search positions after the previous cursor:

Cursor active mode	Cursor active mode	Cursor active mode
Cursor: 0: 'Stimulation'	Cursor: 1: 'Maximum'	Cursor: 2: 'Minimum'
Active mode: Rising threshold	Active mode: Maximum	Active mode: Minimum
Search channel: 6 Stim (Waveform)	Search channel: 2 EMG VM 300 (Waveform)	Search channel: 2 EMG VM 300 (Waveform)
Start position for search (ms): MinTime()	Start position for search (ms): cursor(0)	Start position for search (ms): cursor(1)
End position for search (ms): MaxTime()	End position for search (ms): cursor(0)+30	End position for search (ms): cursor(1)+20
Threshold level (Volt): 0.045		
Noise rejection/hysteresis (Volt): 0		
Delay after crossing (ms): 0		
Minimum step (ms): 0		
Position if search fails:	Position if search fails:	Position if search fails:
Help OK Cancel	Help OK Cancel	Help OK Cancel

Horizontal cursors are also available in active mode. We recently received another question on this topic which we have answered in the Recent Questions section below. Active horizontal cursors act in much the same way as active vertical cursors, but search for different features in the data. They are usually treated as being subsidiary to vertical cursors and automatically repositioned after all vertical cursors upon which they depend have been repositioned. It may be useful to override this positioning, for example, so that all vertical cursors can make use of the horizontal cursor position. This is done with the *Position in sequence* field in the active horizontal cursor dialog (Cursor menu > Horizontal Active Modes).

[Back to contents](#)

Scripts: Signal



When performing transcranial magnetic stimulation (TMS) studies, you may need to create a recruitment curve by plotting the stimulation intensity against the amplitude of the response. In these experiments, single TMS pulses are usually given at a set range of stimulus intensities, with the order of stimulus intensities randomised across trials. In Signal, setting up these experiments requires multiple frame states, where each state in the experiment refers to a different stimulus intensity. The number of different stimulus intensity outputs (and therefore states) must be determined first. This can mean you will spend some time amending the settings for many individual states. We have created the example script [Recruitment curve config setup.sgs](#) to help quickly set up the stimulus intensities for each state of your TMS device, as well as create an outgoing pulse for each state in the sequencer. This script is written to work with Magstim 200 devices; however, the script commands can be adapted to suit other devices. Get in touch with us at Marjorie@ced.co.uk to tell us your requirements.

The stimulus intensity is usually expressed as either maximum stimulator output (MSO%) or as a percentage of a motor threshold (MT%) if it has been previously determined. For example, steps of 5-10% of MSO beginning from a pre-defined level, or 100-180% of RMT in 10% increments. This script is written for RMT% values where the RMT% to begin and end on will need to be entered (keeping in mind the safety and comfort of the subject). Before running the script, you will first need to setup your sampling configuration. The sample mode will automatically be set to Fixed Interval by the script, but you should setup your ports, change your output mode to pulses and enable the digital output port connected to your Magstim device. Lastly, add your Magstim device to the Sample configuration > States tab > auxiliary menu.

Upon running the script, you are presented with a short dialog of parameters:

Setup	
Motor threshold (MSO%)	30
Initial RMT percentage	95
Final RMT percentage	200
Percentage step	5
Number of repeats	5
Time between frames (s)	5.5
% vary	20
Time of pulse into frame (ms)	100
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

- Increments between stimulus intensity steps are kept constant, e.g. 5 or 10%, and are set in the *Percentage step* field.
- The *motor threshold* (as a percentage of MSO), *initial* and *final RMT percentage* values are defined in their respective fields.
- The *number of repeats* for each state may be changed. Enter 1 to have each state only play once.
- The delay between each frame of data may be changed (usually between 4-6s) as well as the % variance of the delay.
- The time of the outgoing pulse from the 1401 that causes the TMS device to fire may also be altered. Typically, this is done 100ms into a frame.

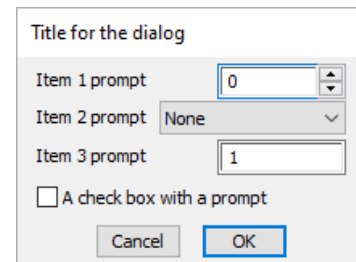
After running the script, you should open the sampling configuration and check all settings are as you need before starting an experiment.

[Back to contents](#)

Scripters Corner – Dialogs

Many scripts require some form of user interaction, like defining a parameter, selecting a channel for analysis, or a type of analysis to apply; the interaction is required to determine what happens next in the script. Dialogs are often best for multi-choice interactions and are used widely across both the Spike2 and Signal software packages, for example the sampling configuration dialog. We do not expect you will need to create something as complex as that, however the dialog family of functions allows you to build dialogs ranging from simple drop-down channel selection lists right through to multiple item dialogs for the control of external devices.

For beginners who have never approached scripting a dialog before, the number of commands and arguments for the various DlgXXX() commands can appear daunting. However, you can define dialogs in a simple way like the picture right, where each item of information has a prompt, and the dialog is laid out automatically in terms of items. Unless you specify positions, the dialog items are stacked vertically with buttons arranged at the bottom. The dialog has a title of your choosing at the top and an OK and Cancel (plus extra user-defined buttons) at the bottom. When the dialog is used, pressing the Enter key is equivalent to clicking on OK.



The above form of dialog is very easy to program; there is no need to specify any position or size information, the system works it all out. Some users require more complicated dialogs, with control over the field positions; this is also possible, but harder to program and is not covered here. There are three basic steps when building a dialog:

1. Use DlgCreate() to start the creating process, give the dialog a name, reset the allowed actions and optionally position and size the dialog. The full function is defined as:

```
Func DlgCreate(title$, x{, y{, wide{, high{, help{, scr%{, rel%}}}}}});
```

All arguments in {} are optional with the title\$ being the only required argument. Full details of the arguments can be found in the in-software help (F1), but if you were to just provide the title then the dialog will automatically be positioned in the centre of the screen and stretched to fit its contents.

2. Use DlgChan(), DlgCheck(), DlgInteger(), DlgLabel(), DlgList(), DlgReal(), DlgString() and DlgText() to define fields in your dialog. These commands (except for DlgText) begin as:

```
Func DlgXxx(item%, text$,...);
```

Each command has extra arguments (see the in-software help). The item% defines the variable position returned by DlgShow(), 1 being the first (item% must be unique for each field). DlgText() is unique in that it enters a non-editable text field into the dialog; you do not supply an item number, and it does not require a variable. For example, you would use this to label sections or groups of fields in your dialog.

3. Use DlgShow() to display the dialog you have built and return values from the fields identified by item numbers. It is defined as:

```
Func DlgShow(&item1|item1[], &item2|item2[], &item3|item3[] ...);
```

Each field created in step 2 must have a variable of the correct type included to collect the entered values; DlgInteger needs an integer, DlgReal needs a real, and so on. The results obtained from DlgList() fields are the indices into the list for the chosen element, not the string value of the list element.

Whilst it is useful knowing all the above (especially if you plan to make more complicated dialogs in the future), most of the time you will want a short simple dialog. We have included the DlgMake script with your software installation (usually C:\Users\User\Documents\Spike10\Scripts) which will help you generate such dialogs. Much like the ToolMake script we discussed in our last newsletter, this script provides the functions in a toolbar that generate the fields discussed above. Copy the output into your own script and tweak it as necessary.

For example, the output generated by the DlgMake script for the picture above is:

```
'Declare variables:
var ok%,int0%,list1%,int2%,check3%;
'Set the variables above for initial values
DlgCreate("Title for the dialog"); 'Start new dialog
DlgInteger(1,"Item 1 prompt",0,100,0,0,1);
DlgList(2,"Item 2 prompt","None|Item 1|Item 2|Item 3");
DlgInteger(3,"Item 3 prompt",1,100);
DlgCheck(4,"A check box with a prompt");
DlgButton(0,"Cancel");
DlgButton(1,"OK");
ok% := DlgShow(int0%,list1%,int2%,check3%);      'ok% is 0 if user cancels,  variables updated if
not
Message("Dialog result was %d", ok%);
```

The variables in DlgShow() are updated with the results if the user clicks OK. To change the default values of the dialog fields, you could update the variables. For example, set `var int0%:=50;` to change the default value for item 1 to 50, and set `var list1%:=2;` to set the default list item to the third list element (remembering the first element is 0). The DlgMake script itself would be an example of more complicated dialog creation.

In addition to the dialog commands already discussed, one more important script function is DlgAllow(). Much like the allow% argument of the Toolbar() function, this sets the user actions that are permitted whilst the dialog is active. DlgAllow() is called before DlgShow(); without DlgAllow() the user cannot interact with Spike2 while the dialog is active. For example, `DlgAllow(1023);` enables the use of all the toolbar menus, switching applications, and changing/resizing windows. As with the toolbar functions discussed previously, it is important to be wary of allowing the use of more items than necessary, as it increases the odds of the user performing an action which causes the script to fail.

One final note on DlgAllow(), the full command is given as:

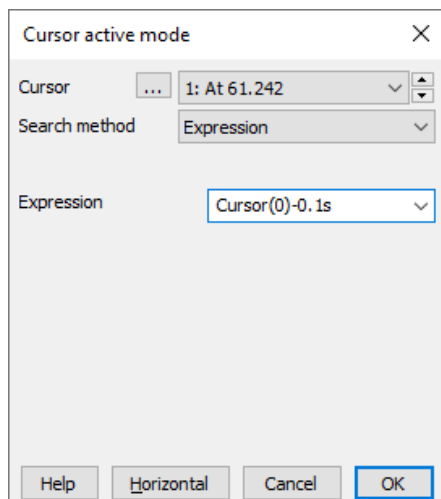
```
Proc DlgAllow(allow% {,func id%(){, func ch%(){}});
```

The extra id% and ch% arguments link idle and change functions to the dialog. These functions can modify the dialog in response to your actions or even to incoming data during sampling. If you are interested in learning more, go to the help topic *Script language > Alphabetical script function index > D > Dialogs*, which provides a full overview and a more detailed dialog example.

[Back to contents](#)

Recent Questions – *How do I define a variable threshold for my active cursor measurements?*

Many dialogs in Spike2 and Signal will accept expressions in place of numbers. These expressions can be either numeric (composed of numbers and arithmetic operators) or view based (allowing references to cursor positions and positions along the axes). The Cursor Mode dialog is used for setting active cursors to automatically search for data features. A horizontal cursor could be used in Active mode as your variable threshold, and entered as "HCursor(1)" (for horizontal cursor 1 – replace 1 with the actual number of your cursor) in the threshold field of the vertical active cursors.



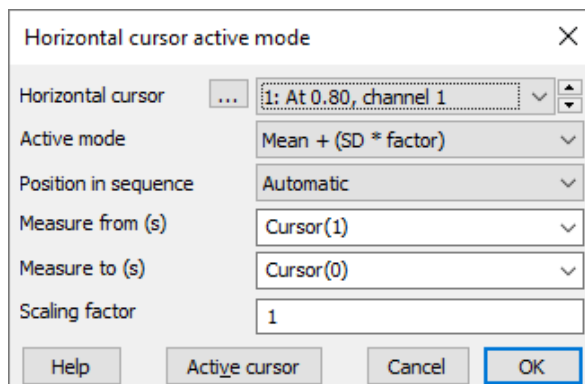
Cursor active mode

Cursor ... 1: At 61.242

Search method Expression

Expression Cursor(0)-0.1s

Help Horizontal Cancel OK



Horizontal cursor active mode

Horizontal cursor ... 1: At 0.80, channel 1

Active mode Mean + (SD * factor)

Position in sequence Automatic

Measure from (s) Cursor(1)

Measure to (s) Cursor(0)

Scaling factor 1

Help Active cursor Cancel OK

In the dialogs above, Cursor 1 is set to a specific time point behind Cursor 0. Horizontal cursor 1 is then set to 1 standard deviation above the mean of the waveform between Cursor 1 and Cursor 0. Each time Cursor 0's position is updated, both Cursor 1 and HCursor 1 will be updated, allowing you to use HCursor(1) as your expression for a variable threshold. Other horizontal active modes available are: Value at point, Mean level, Minimum value, Maximum value, Maximum excursion, and user Expression.

[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.