

Contents

Welcome

Training days

Latest software

Future meetings

Script Spotlight

- TextMarks editor



- What is a Grid view?
- Script – Grid view functions



- How to display a timer
- Script – Measure from states

Did you know...?

- SonPy – Python interface

Recent questions

- Which stimulators can be controlled in Signal?

CED user forums

Welcome

Thank you for downloading our June newsletter, we have lots of exciting development news to share with you in this issue. We have recently released version 10.05 of Spike2 with over 30 improvements and additions, which can be downloaded via our [website](#). Furthermore, we are pleased to announce the release of our Python language interface, SonPy! This enables the conversion of Spike2 data to Python, see our [Did you know...?](#) section for more info. Check out our [Spike2](#) section as well for information on our updated Grid views or watch our new [tutorial video](#) to see them in action.

Whilst many countries are putting their plans in motion for easing lockdowns, many of you are still working from home. Whether you are in the lab or not, we are still here to help. Many users have already requested temporary copies of Spike2 or Signal to use at home, and we are continuing this practice for the time being. If you need a copy to continue your analysis, please get in touch:

Marjorie@ced.co.uk.

Training Days

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

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

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

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



Latest versions of Spike2 and Signal

Spike2	Released	Signal	Released
Version 10.05	05/2020	Version 7.05a	02/2020
Version 9.11	05/2020	Version 6.05b	10/2019
Version 8.19a	11/2019	Version 5.12a	02/2018
Demo	03/2020	Demo	02/2020

[Back to contents](#)

Future meetings and events

ISEK XXIII

Virtual Meeting,

July 12th – 14th 2020

Neuroscience 2020

Washington, DC,
USA

October 24th – 28th 2020

ISAC XXI 2020

Lisbon,
Portugal


November 3rd – 6th 2020

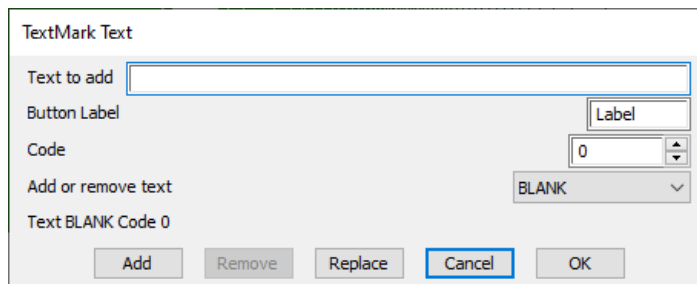
Our meeting calendar is updated each time we receive news of cancellation or postponement due to the COVID-19 pandemic. The full calendar is located on our [website](#).

[Back to contents](#)

Script Spotlight

TextMarks are useful event markers, holding a time stamp, a code, and a text string. With them you can annotate a recording using the same code for similar periods of activity, allowing subsequent automated analysis. If annotation is not possible during the recording you can add TextMarks afterward. We have developed this small script [TextMarks Editor.s2s](#) that lets you generate a range of pre-set TextMarks in a toolbar and place them with a button click.

To operate the script first open the data file you wish to annotate. Next, run the script which will search for your open data file and load the toolbar:  One pre-set TextMark is already available, labelled “Blank”, which may be overwritten. Use *Add/Remove* to open a dialog to create your own pre-sets:

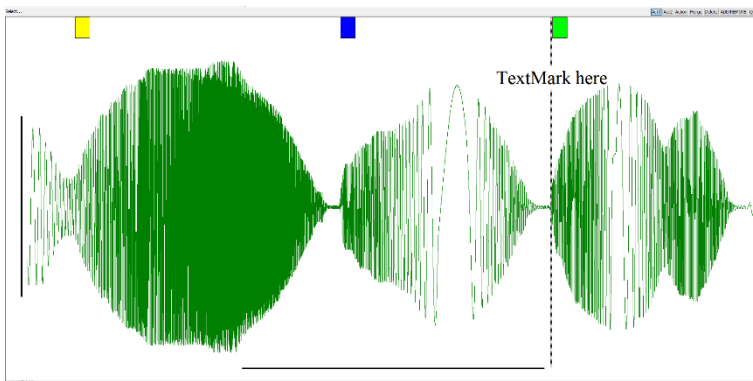


The dialog box titled "TextMark Text" contains the following fields and controls:

- Text to add:** A text input field.
- Button Label:** A text input field with the label "Label".
- Code:** A numeric input field with the value "0".
- Add or remove text:** A dropdown menu currently showing "BLANK".
- Text BLANK Code 0:** A label for the current pre-set.
- Buttons:** "Add", "Remove", "Replace", "Cancel", and "OK".

In this dialog you should add the text to store, create a label, assign the code, and click *Add* to create your new pre-set. Using *Replace* will overwrite the current pre-set in the drop-down box. To remove a pre-set, use the drop-down box to select a pre-set and click *Remove*. Once you have finished creating your pre-sets you are ready to begin annotating your data. Click *OK* to exit the dialog and update the toolbar.

A cursor is placed on your data file titled “TextMark here”, drag this to where you wish to place your first TextMark. Alternatively, right-click the cursor and follow *Cursor -> Set position* to enter an exact time. Now use one of your pre-set TextMark buttons to place it, this will create a new memory channel to hold all your annotations and place your first TextMark. Simply move the cursor and repeat to place where needed, using the *Delete* button to remove any mistakes.




It is also possible to copy existing TextMarks in your data file to your memory channel, should you have created others during recording. Click *Merge* and choose the channel you wish to copy, then use the cursors to select the range to copy across. When finished, *Quit* to exit the script. This will query if you wish to save the memory channel to the data file; if you do not save the channel it (and the TextMarks) will be lost upon closing the data file.

[Back to contents](#)



What is a Grid view for?

The grid view is not a replacement for a spreadsheet, though it has some of the characteristics of one. The main purpose is to provide an easy way to generate tabulated data. You can copy data interactively from other sources, for example the Cursor Values and Regions dialogs, and manipulate it from the script language. You can move data between the grid and other programs that support tabulated data using a Tab character to separate columns and end of line characters (\n) to separate lines.

To create a Grid view interactively, use the File menu New command or the  button in the toolbar and select Grid. You can either accept a default size or check the *Size grid* box to enter a size of your choice. Existing Grid views are resized by right clicking or using the View menu. There is a limit of 1,000 columns and 1,000,000 rows (version 8 had more restrictive limits), probably more than most people need. Grids are saved in .sgrx files using an XML format.

The Grid can be formatted with column headers, font, alignment, type, frame, and colouring:

Headers – The grid has one header row at the top and one header column on the left. These can be hidden or shown by navigating to View -> Show Column Header or Show Row Header. The column headers are changed by double clicking the header which opens a new dialog. The status bar at the bottom of the window (if enabled), displays the co-ordinates of the current grid selection as numbers.

Font – All cells in the body of the grid use the same font. The headers use a bold version of the same font. This is changed through the Font menu in your toolbar.

Row and column spacing – All rows are the same height, which is based on the font size. Columns can be sized individually by clicking and dragging in between cells of the column header. A standard size is set using the script command `GrdColWidth(col%, width%)`, or you can optimise the columns widths to match the data displayed in them through the View menu.

Alignment – You can apply Left, Centred or Right alignment on a single column basis by selecting the column and using the Align column function in the View menu. To apply to all columns, highlight the entire grid by clicking the top left most square where the headers meet and use the Align function.

Colouring – From version 10.05 the you can change the background and text colouring of the entire grid by opening the colours menu. You can further change individual cell colours using the script command `GrdColourSet()`, or by right clicking an individual cell and selecting Colour Selection.

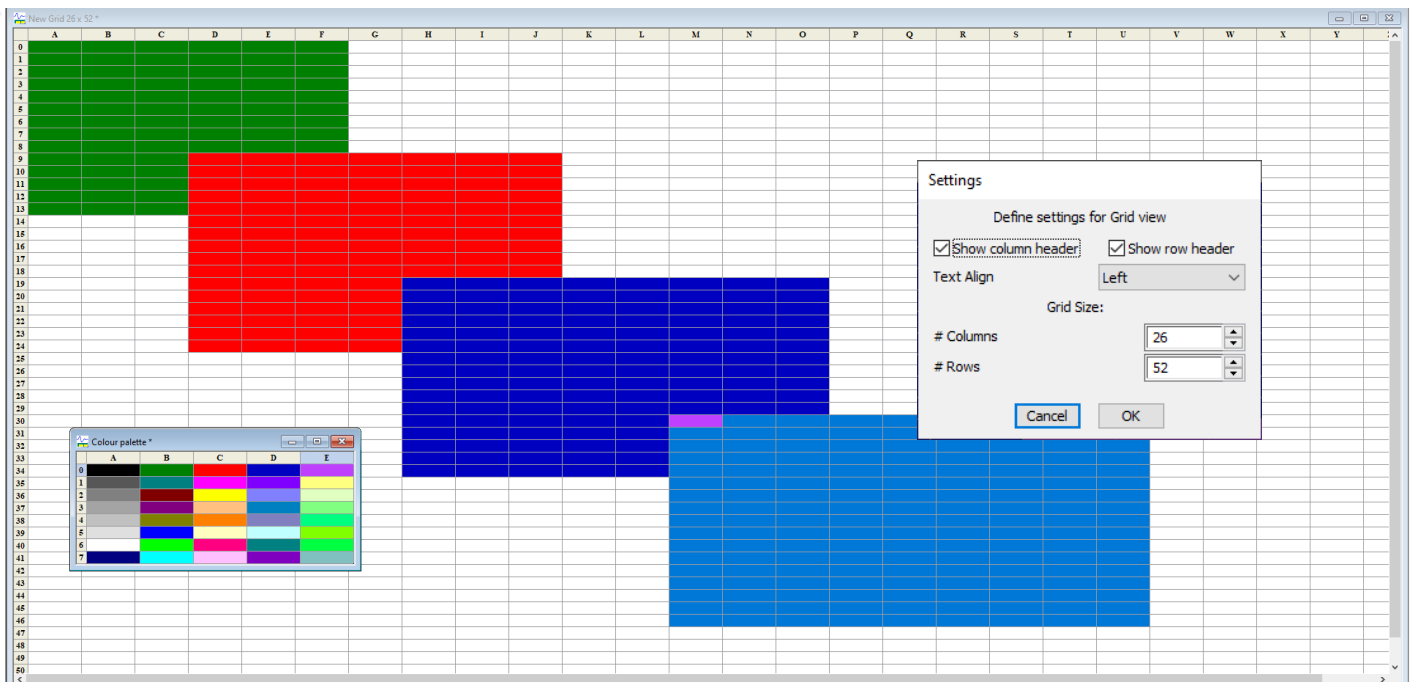
Improvements in version 10.05 – In addition to adding support for Grid colouring, we have added more script support for the Cursor dialogs and improved the recording of grid and cursor dialog actions to generate a script. For example you can now use the Script menu to Turn recording on, open a data file, open the cursor dialogs and a grid, select items in the cursor dialogs and paste them into the grid view, close down the data file and turn recording off. The resulting script will repeat your actions and serve as the basis for a data analysis procedure. See our Script Spotlight section for a script demonstrating these features and more.

Watch our helpful [tutorial video](#) on Grid views help you get started.

[Back to contents](#)

Scripts: Spike2

The example script [Grid View settings.s2s](#) illustrates the use of grids from a script. It creates a toolbar with buttons: Quit, Settings, Change colour, and Arithmetic. The linked functions demonstrate the new and updated Grid views commands. To get full details of the commands and their arguments, click on a command in the script and press F1 (or right click and use the context menu to open the online Help).



`GrdShow(cHead%, rHead%)` will show/hide both the column header and the row header. We have coupled this command to our `Settings%()` function, where check boxes will interactively turn the headers on or off.

`GrdAlign(align%, col%)` alters the text alignment by assigning `align%` with 0 (left), 1 (centred) or 2 (right). In our example script we alter the alignment of the entire grid by replacing `col%` with -2 in the `Settings%()` function; to only align selected columns use -1 instead.

`GrdSize(cols%, rows%)` simply resizes the Grid view to the defined columns and rows.

`GrdColourSet()` and `GrdColourGet()` set and return the colour values of a cell respectively. This can be done for both the text and background colour of cells. We have made use of `GrdColourSet()` in the `ColourSetup%()` function of our example script by creating a “palette” in a secondary Grid view, where the background of each cell uses a colour from the default palette. Using the *Change colour* function from the toolbar asks the user via `Interact()` to select a colour from this “palette”. `GrdColourGet()` obtains the red (r) blue (b) and green (g) colour values and applies them to your selected cells in the blank grid using `GrdColourSet()`.

`GrdSet()` and `GrdGet()` are used to enter text into a cell and read text from a cell respectively. Within the `Arithmetic%()` function of the script, `GrdGet()` first grabs the numbers from selected cells in a column and enters them into an array. The script then performs a user selected calculation via the `ArrSum()` command, and `GrdSet()`

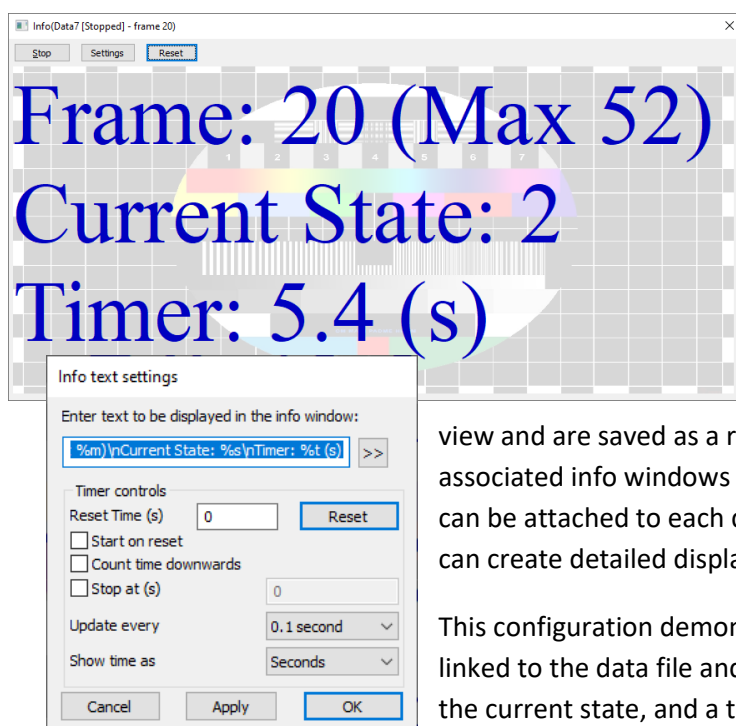
then overwrites the column header with the answer. There is an important caveat when dealing with numerical data in a Grid view; all data within the Grid is stored as a string of text. Therefore, any numbers entered into the grid must be converted back to real or integer data after using `GrdGet()`; much of the `Arithmetic%()` function deals with this, as well as converting the calculated output back to a string before inputting with `GrdSet()`.

`MoveTo()` is used to select cells in a Grid view, and `MoveBy()` will select cells in the grid relative to the last position. The `Selection()` command is used to obtain information on selected cells, i.e. the number of selections or the positions of a selection.

[Back to contents](#)

Signal *Is it possible to display a timer on screen?*

Information windows are ideal for displaying a timer. Info windows are either created with the View menu, Info windows -> Create New Window command or from a script with the `InfoNew()` function. This is only available when a data or XY view is the current view. Info windows are mostly used to display user-defined information in a large font for visibility from across the lab, with options for a timer to be configured. They could also be used to display your protocol steps at the same time. Simply typing out the protocol is an option, or info windows can also display an image file instead of the standard background; resizing the window will cause the contents to automatically scale to fill the available space. To further improve the amount of visible area, either the area containing the buttons or the window title area (or both) can be hidden if desired. These and other options are available by right-clicking on the info window.



To go into more detail, each info window contains a user-defined text string, the optional background image, and configurable colours. The text string uses special fields which are replaced by the timer value or other data extracted from the associated view and updated at user-defined intervals. Meaning for example, if you wished to display the most recent Y value of an XY plot in giant red letters which updated with every sweep of data, you could configure that. Info windows are attached to a file, memory or XY

view and are saved as a resource; when the associated view is re-opened any associated info windows will be re-created. Furthermore, up to 10 Info windows can be attached to each data view, and by using the available script functions one can create detailed displays from a script.

This configuration demonstrates two info windows in use. The first info window is linked to the data file and shows the current frame, maximum frames obtained, the current state, and a timer in seconds. The text string is written as *Frame: %f (Max %m)\nCurrent State: %s\nTimer: %t (s)*, where special fields are prefixed

with %, and \n denotes a new line. The second info window is linked to an XY trend plot of the current frame vs the maximum amplitude and shows the current frames maximum and the previous frames maximum. The text string is *Maximum: %ry1 (V)\nLast maximum: %py1 (V)*.

[Back to contents](#)

Scripts: Signal

Our [Measurefromstates.sgs](#) script will plot a selected measurement from an evoked response in an XY view. The script is specifically written to obtain measurements per state from a file containing multiple frame states. The measurements available are: curve area, mean, slope, area, sum, modulus, maximum, minimum, peak to peak, RMS amplitude, standard deviation, extreme, peaks, troughs, point count, SEM, and RMS error.

Essentially this script helpfully reduces the number of steps and button clicks needed to produce an XY trend plot of the simple measurements above. Being more visual than the XY trend plot dialog, and by automatically creating an XY channel per state, the script is far quicker than attempting to produce the plots manually. Upon running the script you are presented with a dialog that allows you to choose the channel, the measurement, and how many states to perform the measurement on. The script then uses the `Interact()` function to give you the chance to mark out the evoked response, before proceeding to the produce the XY trend plot.

The script is available to download from our website here. A suitable example data file, MEP example.cfs, is included in the .zip file for you to try out the scripts function.

[Back to contents](#)

Did you know...?

Alongside the release of Spike2 update 10.05 we have created SonPy, our Python language interface. With the interface, anyone with a knowledge of Python may access the data in Spike2 data files or create entirely new data files. It is possible to create and edit all Spike2 channels: waveform, events, and markers.

The interface is available to [download from our website](#), with a full help guide included with the installation. We have also included a few test scripts to demonstrate creating a new 32- or 64-bit file, reading an existing WaveForm or RealWave channel, and reading all data within a Spike2 data file.

We are interested to see our users run with this, so please let us know your comments and feedback. We hope the interface is of use to you!

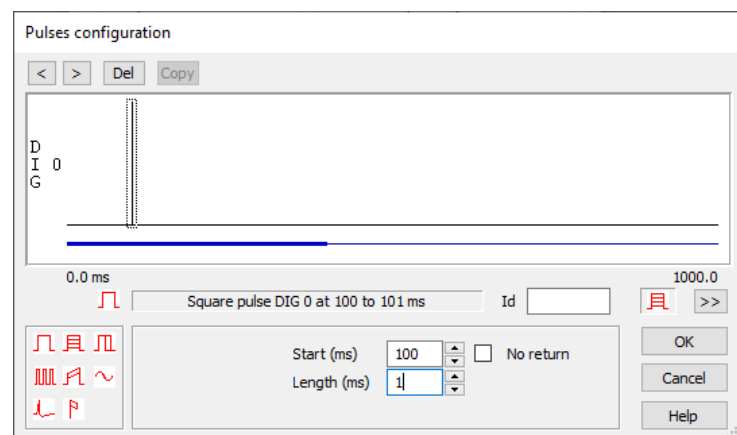
[Back to contents](#)

Recent Questions

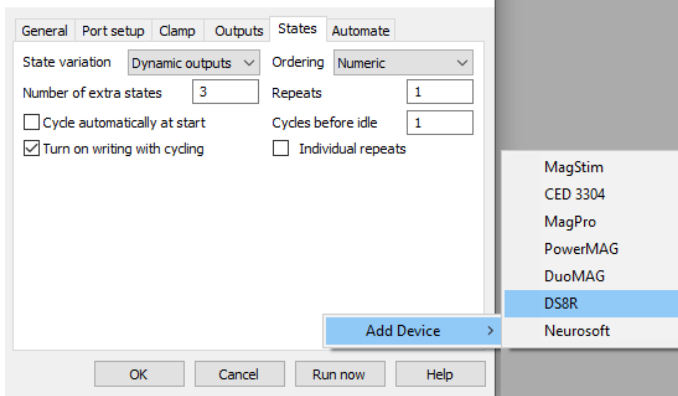
What peripheral stimulators can I trigger using Signal?

Any stimulator that has an input for digital triggering can be triggered by Signal. First configure a square TTL pulse in your pulse outputs of appropriate length (usually 1ms is sufficient but consult your stimulators handbook) and connect the correct Digital Output of your 1401 to the stimulator and you are set.

However, do note that in this type of setup full control of the stimulus intensity is given by the stimulator, it is only possible to control the timing of the stimulus in Signal.



In addition, the Digitimer DS5 can be controlled via the 1401s DAC output. Much like the digital triggered stimulators, this is set up in Signal via the pulse outputs, however instead a pulse being configure on the Digital Outputs, a waveform should be configured on a DAC output. The analogue voltage input on the DS5 is translated into an isolated constant current stimulus (up to $\pm 50\text{mA}$), replicating the shape of the input waveform.



The Digitimer DS8 on the other hand can be software controlled through Signal to set the stimulus parameters. Much like the Signal controlled Transcranial Magnetic Stimulation (TMS) devices, the DS8 is added through the Auxiliary devices menu when Multiple Frame States is enabled. There all the stimulus parameters may be configured, as well as the timing of the stimulus being set via the pulse outputs.

[Back to contents](#)

CED User forums

Try the [CED Forums](#) bulletin board for software and hardware support.

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

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

[Back to contents](#)

Contact us:

In the UK:

Technical Centre, 139 Cambridge Road,
Milton, Cambridge, CB24 6AZ, UK

Telephone: (01223) 420186

Fax: (01223) 420488

Email: info@ced.co.uk

International Tel: [44] 1223 420186

International Fax: [44] 1223 420488

USA and Canada Toll Free: 1 800 345 7794

Website: www.ced.co.uk

All Trademarks are acknowledged to be the Trademarks of the registered holders.
Copyright © 2020 Cambridge Electronic Design Ltd, All rights reserved.