CAMBRIDGE ELECTRONIC DESIGN LIMITED

eNEWSLETTER

#10
March 2021

Contents

Welcome

Thank you for downloading our March newsletter. The days are gradually getting longer in the UK with Spring seemingly on the horizon. In this newsletter we answer more user questions, continue getting started with spike sorting in Spike2, discuss leak subtraction analysis in Signal, and continue our Scripters corner for beginner script writers.

Updates 10.09a, 9.13 and 8.21 for Spike2 were all recently released. Update 7.06 is finishing testing and should be released within the next week. Keep a look out for our update notification or sign up via our website if you have not already.

Our first online training event of 2021 is now over halfway through with only a few more sessions to go. We have received some excellent responses and encouragement and hope you have found the sessions helpful so far. If you have not been able to catch them all, the sessions have been recorded with the videos made available on our website.

Training

Our free 2021 online training sessions began on the 2nd February for Spike2 and 4th February for Signal. We planned 8 hourly sessions for each software package spread out over 8 weeks. The aim of these sessions is to share top tips and tricks on varying subjects related to Spike2 and Signal acquisition and analysis. Visit our website for full details and recordings of previous sessions; email us at Marjorie@ced.co.uk to book your place.

Due to the ongoing COVID-19 pandemic, all in-person training events have been put on hold. We do however offer remote training sessions by Skype/Zoom/Teams either one-to-one or with groups.

Join us and learn how to make the best use of Spike2 and Signal to save hours of repetitive analysis. Our remote sessions are free to arrange and are suitable for both existing and prospective users of our data acquisition and analysis systems. If you would like to schedule a session, please get in touch: 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 10.09a	02/2021	Version 7.05a	02/2020
Version 9.13	02/2021	Version 6.05b	10/2019
Version 8.21	02/2021	Version 5.12a	02/2018
Demo	02/2021	Demo	02/2020

Back to contents

Here to help

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

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

Many experiments consist of several stages; for example, you may record baseline activity for a period before providing a stimulus and then record the response over time. It is simple to build timers into your scripts to keep yourself informed.

The Seconds() script command can be used for time measurements, as well as setting a timer relative to the system clock. The command with no arguments returns the current time in seconds (the time is zero when Spike2 starts sampling). With one argument, `Seconds(set)` will set the timer to the defined time. Most script-built timers will use an idle function to keep track of the current time with Seconds() along with flow of control statements (`if...endif`, `while...wend`, etc.) to direct the script based on the current time.

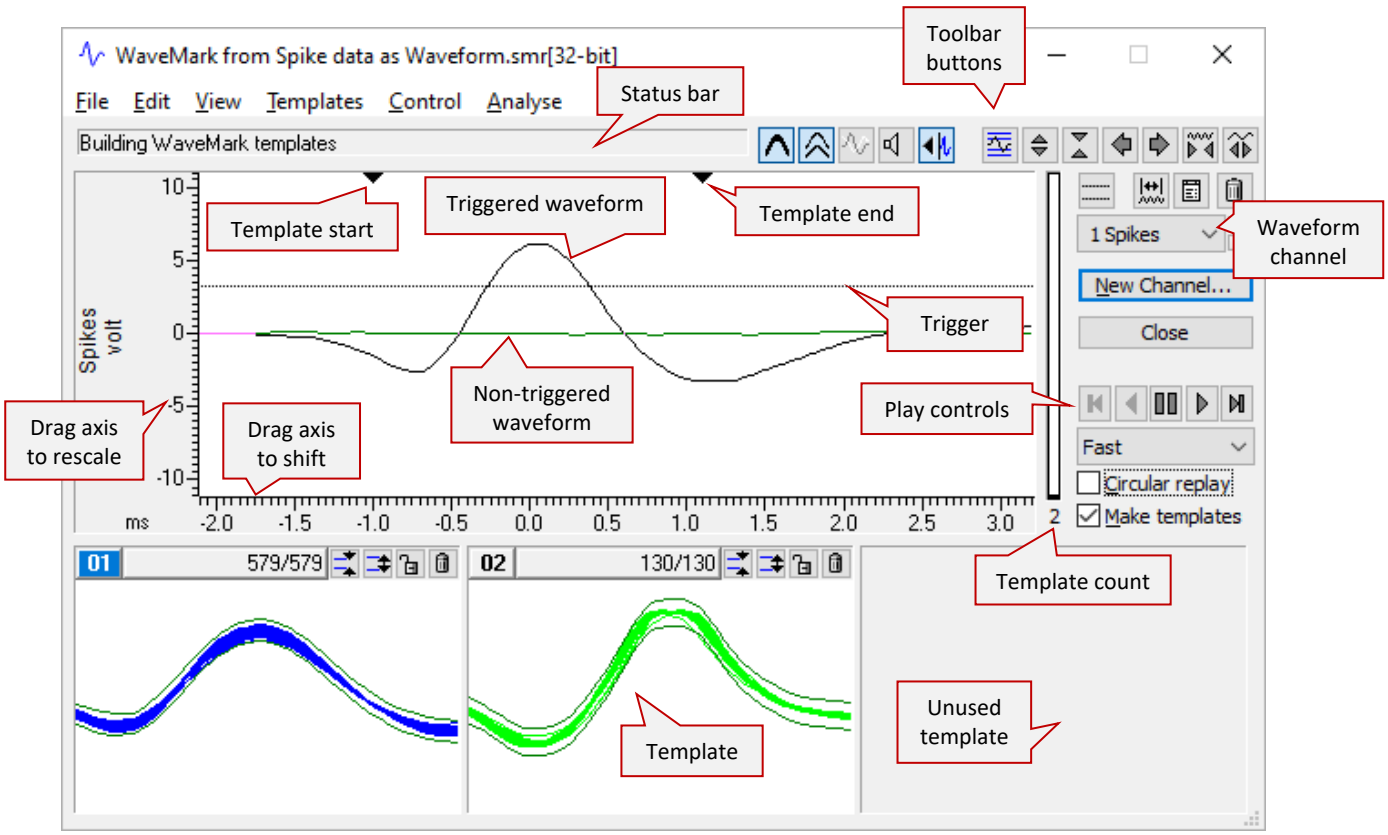
The attached script Example dialog timer self-destruct.s2s creates a dialog showing a 5 second countdown timer which will automatically close once the timer reaches 0. This is more intricate than usual scripts but demonstrates the versatility of the script language and may prove useful for your own scripts.

Back to contents

How do I get started with Spike sorting?

This is our second feature covering spike sorting. In our last issue we introduced the topic, discussed selecting the template area, and covered setting the trigger levels. This time we will focus on off-line template formation and its dialog.

To begin extracting WaveMark data from waveform channels, right-click your waveform channel and select *New WaveMark* or navigate to the *Analysis* menu > *New WaveMark*. The file Spike data as Waveform.smr is included with your Spike2 installation (usually located in C:\Users\User\Documents\Spike10\Data) and can be used to experiment with.



Before forming templates, ensure you are happy with your template area and trigger levels. The program scans the data in the channel set by the channel drop-down menu and displays triggered and non-triggered events, as if the data were being sampled. Click the play button to start replay, then click the estimate trigger button (maybe more than once) to get triggering started. Adjust the upper trigger level until you see both small and larger spikes triggering.

With the off-line template formation window open, cursor 0 is shown in the time view. This cursor marks the position of the currently displayed spike. When template matching is paused the cursor may be dragged to a new position to trigger a search for a trigger condition. Ctrl+J moves Cursor 0 to the start of the time range and starts a search for a trigger condition. If you drag cursor 0 in the associated time view, this triggers a spike search from the trigger position to display a new spike in the template formation window. You can use this feature to quickly search through your data, picking out spikes of interest and ensuring your triggering parameters are correctly finding your spikes.

Hints and tips

- Unticking *Make templates* allows you to cycle through triggered spikes without templates being made.
- Using the  button to move forward spikes in turn, you can check your current parameters are working.
- Right click the trigger cursors to *Link time view cursors*, allowing you to view the triggers as horizontal cursors on your data. These can then be dragged and set in the time view. The right-click menu also lets you fetch triggers or move them away.
- Use the  button or press the Home key to guess suitable trigger levels.
- The  button (Ctrl+K) will track the current spike in the template matching window on your time view, keeping cursor 0 centred.
- If needed, use the  time range button to set a specific time range to process. This is automatically set to the full time range when the window is opened.
- Full details of the dialog's controls are found in the software on-line help (F1 – *Spike Sorting* > *Spike shape sorting dialogs* > *Menus in spike shape sorting dialogs*).

Template matching

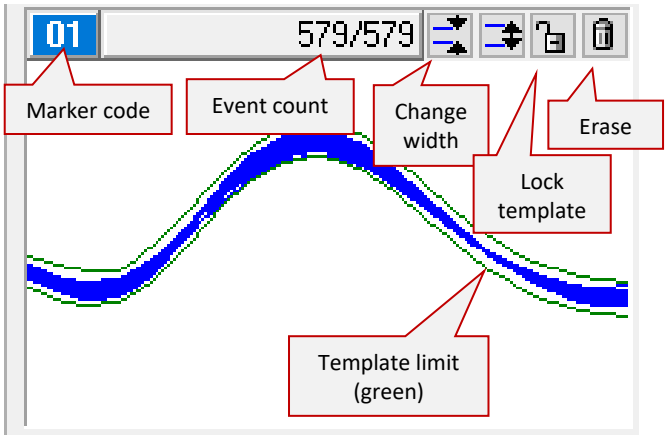
When satisfied with your triggering parameters, you can begin template matching. Make sure that *Make templates* and *Circular replay* are checked and click the  clear all templates button for a clean start. Hit the  play button to begin. Template searching and matching will begin from the current position of cursor 0 to maximum time. With *Circular replay* ticked (Ctrl+Q), template searching is performed on a continuous loop from start time to end time, wrapping back around. To control the speed of template matching, use the drop-down menu underneath the play controls (note that 'Fast' will process templates as fast as possible).

Spike2 creates templates automatically by searching for triggered spikes and following the rules set in the template parameters dialog. We will cover this dialog in detail in a further issue, but for now know that it is accessed through the  button, Ctrl+Enter, or the *Templates* menu > *Parameters*. Triggered spikes are compared first with confirmed templates. If none match, they are compared with the provisional templates. If they match nothing, they form a new provisional template.

Provisional templates are triggered spike shapes with the potential of becoming a confirmed template should enough similar triggered spikes be found. They are not shown in the template area of the dialog. When a provisional template matches a new spike and this causes the template spike count to reach the *Number of similar spikes for a new template* set in the template parameters, it becomes a confirmed template. If this matches a current confirmed template it is merged with that confirmed template. Otherwise, a new confirmed template is created and displayed in the template area.

When a spike matches any template, it modifies the mean template shape and width (unless the template is locked). If the spike matches more than one template it is added to the one with the smallest error between the spike and the mean template shape.

As spikes are processed, the vertical bar (template count) on the right of the data display area indicates the number of provisional templates that are being considered in grey and the number of confirmed templates in black. If your parameters are causing many provisional templates to be generated, the bar increases whilst changing to red, indicating a problem with your parameters. The first 20 confirmed templates appear in the template area below the data display.



Unused template spaces appear as grey. If more templates are generated than currently on display, drag the bottom or bottom right corner of the dialog to resize. Click the marker code to edit.

Once you are satisfied with your templates, click the *New Channel*... button to open a dialog in which you can select the channel to write to and the new data type.

Back to contents

Scripts: Spike2

Sometimes we are unable to fully shield our recordings from electrical interference. A typical source of interference is mains power, and ideally this interference should be removed at the source by identifying and eliminating earth loops, using Faraday cages, and improving shielding. However, in many cases there remains a signal that is linked to the mains frequency (50 or 60 Hz, depending on where you are in the world). It is possible to use digital filtering techniques to eliminate the 50 or 60 Hz bands. However, mains-related interference is never pure 50/60 Hz cycles; there are variable components at higher harmonics of the mains frequency. A notch filter inevitably distorts the signal and does nothing to remove the higher harmonics in the data.

Our script writers have created two scripts to help: HumRemove.s2s and HumRemExpress.s2s. These scripts are designed to improve the signal-to-noise ratio of electrophysiological recordings by removing the mains-related interference. This is done not by filtering, but by subtracting an estimate of the mains interference from the raw data and storing the results in a new channel.

The advantage of this method is that by using markers at the onset of each hum cycle, we generate a triggered average signal. Averaging over many hum cycles results in an average waveform that is dominated by the signal component, correlated with the mains frequency (i.e. the unwanted interference including any harmonics). Signals that are not correlated to the mains cycle are "averaged out". The original signal is then cleaned up by subtracting this “average interference" waveform from each individual mains cycle.

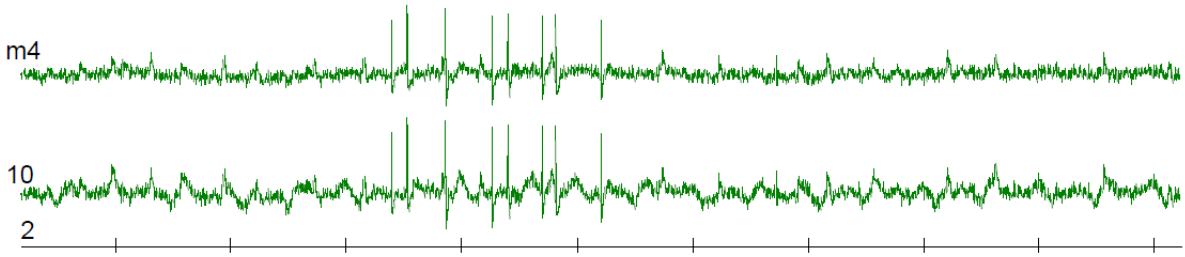
Both scripts can be downloaded from our website with full details of their workings, including instructions on their use. The HumRemExpress.s2s script is a streamlined hum removal script for use when you have recorded a reliable hum marker event channel (see requirements). If you need to derive a hum marker from a waveform channel that contains hum then you should use the original HumRemove.s2s script.

Requirements for running:

- Both scripts run on Spike2 version 7.20 or later.
- They will process one or more waveform or RealWave channels to remove mains-frequency related interference. They can also remove mains-like pick-up of frequencies other than 50/60Hz, for example, from a motor.
- Channels should be sampled at least 10 times faster (500/600Hz) than the hum frequency; typically, they will be sampled more than 100 times faster (5/6kHz) than the hum frequency.
- Titles of channels to be cleaned must be unique and, in the case of 32- bit .smr files, must be no more than 6 characters long.

- The scripts require a channel of events that mark the hum cycles, or a waveform channel from which we can deduce where the mains cycles occur (HumRemove.s2s only). This waveform could be a recording from a piece of unshielded wire or one of your data channels that contains a significant hum-related component.

The best solution is to use a circuit to convert the mains to TTL pulses, one per cycle. This is much more economical in terms of data storage than saving an additional waveform channel. The CED 4001-16 Mains Pulser box will do this for you – see our website for details. If you use a CED mains pulser box or similar device to create the hum cycle marker then you can process your data more conveniently using the HumRemoveExpress.s2s script.



In the above example, the event channel (2) at the bottom holds mains cycle markers, channel 10 has the raw data and channel m4 contains data with the mains related interference reduced by subtracting an average of the original waveform with respect to the cycle markers.

Back to contents

How does the leak subtraction work in Signal and how do I access it?

Leak subtraction is a particular analysis used by voltage clamp researchers; it will not be available unless clamping support has been enabled in the Clamp tab of the *Edit* menu > *Edit preferences* dialog. Within Signal, this analysis creates a multi-frame memory view by carrying out a leak subtraction analysis on the source data.

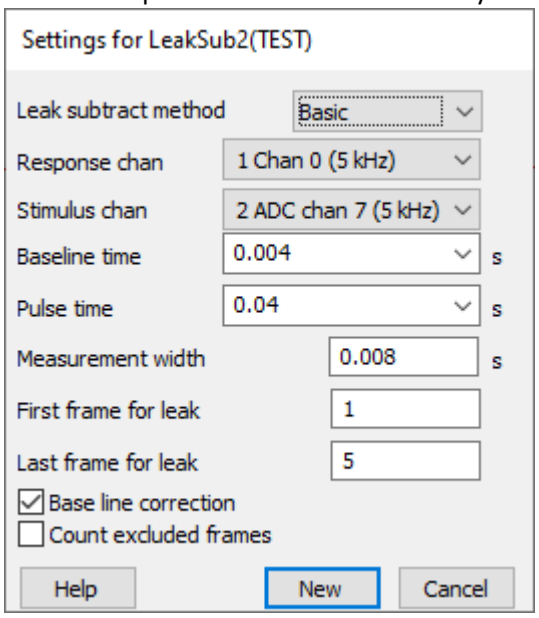
The leak subtraction method we use is based on employing a small stimulus that does not cause voltage activated ion channels in the cell membrane to open. The resulting current flow through the membrane, made up from resistive and capacitive components, is then measured. This gives us a current waveform representing the resistive and capacitive response of the cell to the low amplitude stimulus. This response is assumed to be proportional to the size of the stimulus pulse. The leak measurement is then scaled by the stimulus amplitude used in following frames to generate the leakage currents that will be generated by these stimuli. This scaled leak current is then subtracted from the recorded traces to leave only the voltage activated ion-channel effects.

Leak subtraction uses two channels of your data; the stimulus channel which is used to measure the amplitude of the stimulus pulse so that the leak can be scaled (but is not modified by the analysis) and the response channel which is the only channel modified by the leak subtraction process. All other channels are ignored and copied into the memory view unchanged.

To access the leak subtraction, navigate to the Analysis menu > New memory view > Leak subtraction. The example data file TEST.cfs included with your Signal installation (usually located in C:\Users\User\Documents\Signal7\data) may be used to try the method out.

Fields of the settings dialog will alter depending on the leak subtract method chooses. Common fields are:

- Response channel – channel of your recorded response signal.



• Stimulus channel – channel of your recorded stimulus signal

• Baseline time – time point during baseline, pre-stimulus.

• Pulse time – time point during pulse, usually mid-pulse.

• Measurement width – a time span to form averages of the baseline and pulse to even out noise. Select a width that spans only the baseline/pulse from your chosen point in the above fields (i.e., the width from your baseline time point does not over-shoot into your pulse and vice versa).

The baseline and pulse times are used to measure the stimulus amplitude from the relevant channel; the measurements are averaged over the *Measurement width* (using the time range from time-width/2 to time+width/2).

- Baseline correction – removes the DC offset from the corrected data so that at the baseline time of the response level will be unchanged.

Leak subtract methods:

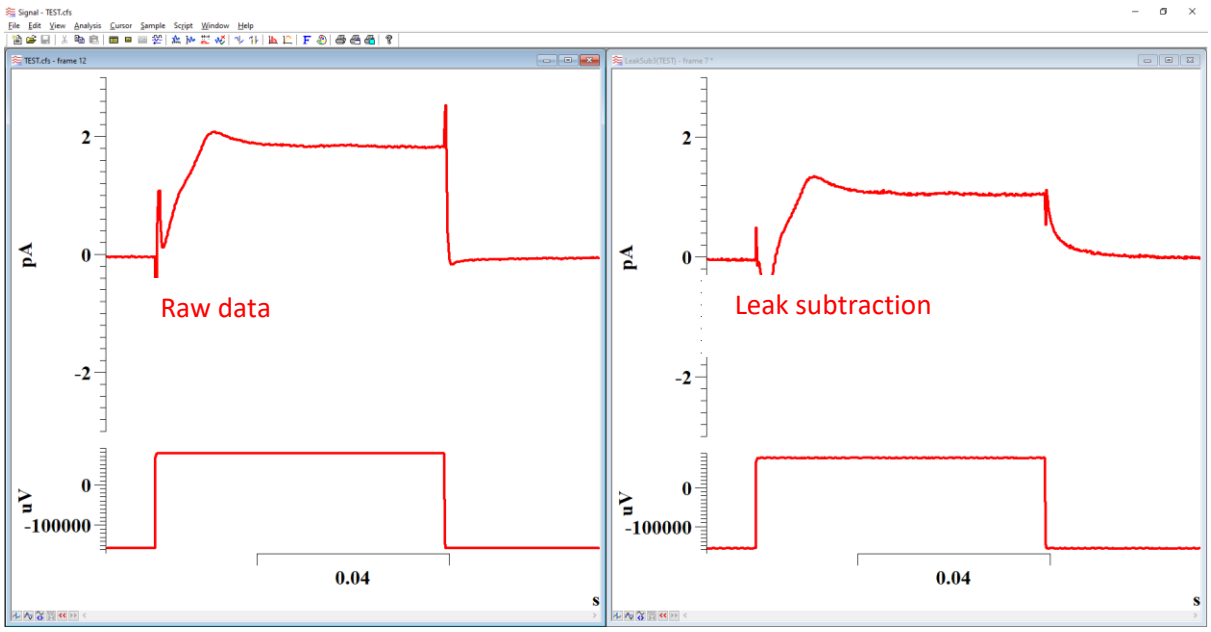
Basic – measurement is generated from a fixed range of frames. Frames are set with the *First frame for leak* and *Last frame for leak* items. All frames processed use this single leak measurement. Frames contributing to the leak data are automatically skipped during processing.

P/N – measurements are extracted only from frames being processed. The first *n* frames processed make the leak data, the next *m* frames are processed using this leak data. The next *n* frames make a new leak data set and the following *m* frames are processed using the new leak data. This cycle continues throughout the frames being processed. The values for *n* and *m* are entered in the settings dialog using the *Form from first* and *Subtract from next* fields respectively.

States – leak data is assembled from all frames within the file with a given state, with the state number entered in the settings dialog. All the frames processed use this set of leak data, frames contributing to the leak data are automatically skipped.

These three modes are very similar, the only real difference being how frames used to generate the leak trace are selected. The *Count excluded frames* field is most relevant to the P/N method. With this enabled, frames which are excluded through tagging are still counted as part of the frames to use. Otherwise, the program will continue to search beyond your excluded frames to use the number of frames specified in either the *Form from first* or the *Subtract from next* fields. If the data was acquired using a fixed number of frames for forming the leak this option should be disabled.

Click *New* to create a new memory view and open the Process dialog. Select your frames to process and click *Process* when ready.



Hints and tips

- If possible, record several low amplitude stimulus frames to be averaged together to minimise the effect of noise.

- The times and duration of the stimulus pulse(s) should be the same in all frames. If the pulse times in the averaged low-stimulus data differ from the data to be modified, then the subtraction of the scaled low amplitude response will not work correctly. This leak subtraction method is not suitable for protocols using stimuli that start at varying times or varying duration stimuli.

- The stimuli should be simple square pulses. The stimulus amplitude is measured by averaging over a specified time range, which will not work correctly if the stimulus is not held at this level. This method is not suitable for protocols using arbitrary waveform stimuli. If using ramps and sine waves you would have to make sure that the measured amplitudes gave an accurate estimation of the relative scale of the stimuli.

- There should only be two stimulus levels per frame: the baseline/holding potential and the stimulus potential. Measuring the stimulus amplitude and scaling the low amplitude response will fail if scaled by two different values at once. This restriction presents the most difficulty as it makes it problematic to use leak subtraction with paired pulse protocols. To use this method with multiple pulses of different amplitudes you must ensure the amplitude of the pulse measured is the one giving the response you are interested in. You should also consider what part of your data trace you feel gives the most accurate representation of the baseline.

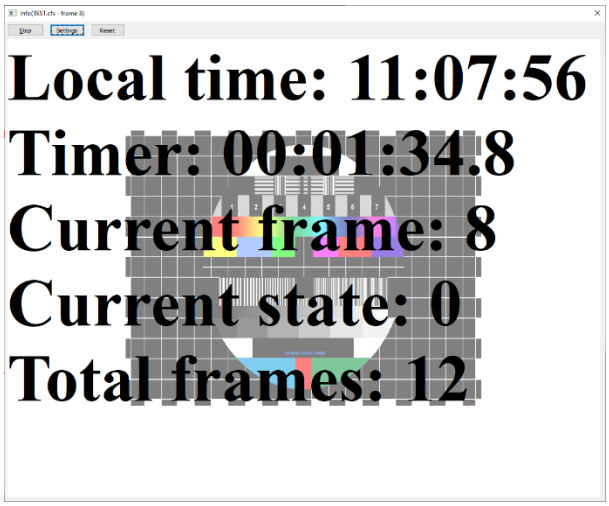
Back to contents

Scripts: Signal  

The information window is a specialised type of view used to display times and special information using a large font for extra visibility. Information windows are attached to a time, memory or XY view. When saved to disk these views will also save the information needed to re-create any associated information windows when the file is re-opened.

You can create an information window interactively with the *View* menu > *New Info window* command or from a script with the `InfoNew(mode%)` function, with other script functions available to manipulate information windows. The display content is set by a text string which includes special fields replaced by other information (for example, the current frame number).

Information windows may also display an image file instead of the standard background and either the area containing the buttons, or the window title area (or both) can be hidden if desired. These and other options are available via a menu generated when you right-click on the information window.



We have produced the example script InfoWindowCreate.sgs to quickly create an info window from an open view. We hope these functions will prove useful to adapt and include with your own scripts.

Upon launching the script, you will be asked to select the source (if more than one is open) from which an info window is created displaying all the available options for that source type. The script makes assumption of channel numbers and points; the string used to create the content is accessed by navigating to Settings at the top of the window. The script text includes explanations for special fields that can be changed.

The image Test Card.jpg (above with the script) is displayed in the background with transparency applied. This image can easily be replaced with an image of your choosing; save the image of your choice to the same folder location as the script file, then replace the filename within the script text where prompted.

The script language used should be self-explanatory, however details of any commands you are unsure of can be looked up in the software help by clicking on the script command and pressing F1.

Back to contents

Scripters Corner – Debugging

In our last newsletter we discussed user-defined Functions and Procedures. In this issue we will begin to delve further into the built-in script functions, and cover debugging your own scripts.

Script editor controls

This drop down allows you to quickly jump to one of the Func or Procs in your script. 

Script compiler – checks the syntax of the script and, if no errors are found, it creates the compiled version ready to run. Only the first error found is highlighted, and you will need to check again once you have corrected your syntax. 

Run script – if the script has not been compiled it is compiled first. If no errors are found, Spike2 runs the compiled version, starting from the beginning. Remember that user-defined functions will only be run at the point they are 'called' in the script. 

Set/clear break point – will set a break point on the line containing the text caret or clears a break if one is already set. A break point pauses a running script when it reaches the line containing the break point. They are also set and cleared by clicking the mouse pointer in the left margin of your script view, or by right clicking the line and using Toggle break. Break points appear as a red marker in the gutter (margin to the left of the script). They are integral to debugging and are discussed in more detail later.

Clear all break points – remove all break points from the script. Break points can be set and cleared at any time, even before your script has been compiled.

Full details of these controls and more are found in the in-software help by pressing F1.

Built-in functions

An example of a typical function is: `FileOpen(name$, type% {,mode% {,text$}});`

The items within the brackets are the function arguments. In this case, `name$` sets the name of the file, `type%` determines which type of file it should be (for example a data file or a text file) and `mode%` determines the state of the window when opened (for example visible or maximised). Details of required arguments for any built-in function are in the online help. When editing a script or using the Evaluate window (Ctrl+L), place the text cursor in the function name and press F1 to display the relevant help page. You can also get a pop-up call tip to appear if you hover the mouse pointer over a user defined or built-in Proc or Func name.

All built-in script commands return a value. The returned value is used to confirm the action of the function. For example, FileOpen() will return the view handle of the file or a negative error code if unsuccessful. Hit F1 to open the in-software help and *Search* for your script command. Underneath the arguments you will see the *Returns* line.

Debugging

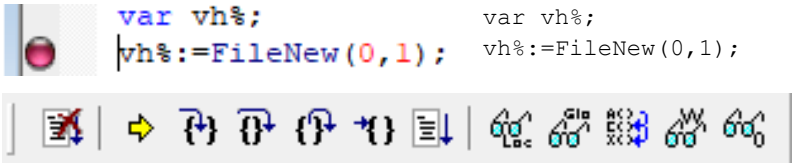
```
FileNew(0, 1);
```

The above syntax is correct and will compile, creating a new data view based on the current sampling configuration and making it visible. If the above script command succeeds and creates a new data file, the function returns the view handle of the new data file. However, if no 1401 is connected to the PC then the command will fail and return a negative error code. Pretending we do not know the cause and we do not see our new data view being created; how do we work out this command is failing? There are several methods available to us.

We could copy the command into Evaluate window (Ctrl+L) and use Eval... to check what value is returned. However, that is less useful if we have used variables in place of the arguments as the evaluate window operates outside of the script, so the variables need to be replaced with actual values. The debugging toolbar provides a better method:

```
var vh%;
vh% := FileNew(0, 1);
if vh% < 0 then halt endif;
```

In this example, adding a break point to the line to debug and running the script will pause the script at the break and open the debugging toolbar:



The debugging toolbar will also open if an error occurs whilst running the script (and the option is ticked in your Edit menu > Edit preferences dialog > General tab). Full details of these controls and more are found in the in-software help by pressing F1. However, a quick look at the ones most used:

Stop – Stop running the script (Ctrl+Alt+Shift+F5).

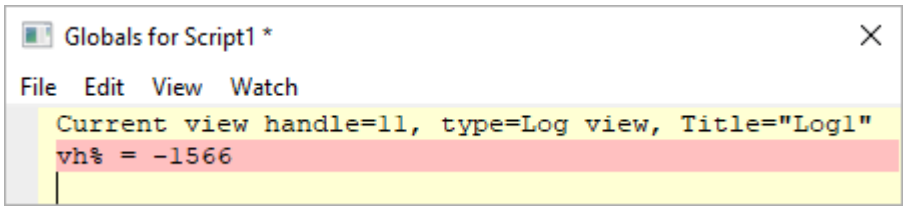
Step-in – This steps to the next statement or in the case of a user-defined Proc or a Func, steps into the function (Ctrl+F5).

Run – Runs the script (Ctrl+Shift+F5) from the current break point to the end of the script or the next break point if present.

Local variables – Shows local variables for the current user-defined func or proc (F7). If there is no current routine, the window is empty.

Global variables – Shows global variable values (Ctrl+F7). The first entry in this window lists the script current view by handle, type, and window name.

By opening the global variables and stepping once from our break point, the variable vh% of our example is updated and we see a negative error code returned:



Now we know there is a failure in our script we can build in more checks:

```
var vh%;
vh%:=FileNew(0,1);
if vh% < 0 then
    Message("Open failed, code %d: %s", vh%, Error$(vh%));  'Alternative 1
    PrintLog("Open failed, code %d: %s", vh%, Error$(vh%)); 'Alternative 2
halt;
endif;
```

The Message() and PrintLog() commands may be used to report back the negative error code; special character %d is replaced with the integer from variable vh%. The Error$() script command will print information on a negative error code; special character %s is replaced with a string, coupled in the example above. The error code -1566 tells us that the 1401 device driver could not be found, which in other words means the software is not connected to any hardware (in this case the 1401 was simply not switched on).

Finding the best method to debug your own scripts will come with experience, but most often adding break points to statements that failed and stepping through to view how your variables are updated is generally the most useful. You can even add variables of interest to a watch list  by right clicking a variable in the local or global variables window and selecting *Add to watch window*.

Back to contents

Recent Questions – *How many licences do I need?*

Licences for our software are supplied as a master licence, the first licence purchased by an individual/lab/group, and additional licences supplied at roughly 1/3rd the price. Master licences and additional licences will share the same serial number. We do not limit the number of PCs a particular licence can be installed on, providing the machines are owned/associated with the lab and/or group that purchased the licence. We do however limit the number of concurrent users of that licence, in that the number of users cannot exceed the number of licences owned. Therefore, you will need the same number of licences as your concurrent users.

If a 1401 is sampling with the software, that is counted as a licence in use. For example, if a lab owns one licence and a lab member is sampling data with the 1401, that is the licence currently in use and should not be used elsewhere. However, the first lab member could sample data in the morning and, once sampling has finished, a different lab member then analyse the data on a different PC later in the day. This still falls within the terms of our licencing agreement.

If a lab needed to sample data with the 1401 and be analysing data simultaneously on a different PC, that is counted as two users and the lab would therefore need both a master licence and additional licence. All additional licensed copies are expected to be used on the same site and in the same building/laboratory and by people working in the same group.

Full details and restrictions of additional licences can be found in the software help (F1) *Spike2 version # > CED software licences* for Spike2, and *Technical support > Licence information* for Signal.

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
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