CED eNews
The eNewsletter from Cambridge Electronic Design
#4 ~ August 2003
Established 1970

Contents

News

Training Days in New Orleans
6th & 7th November 2003

The pre-Neuroscience training days will be held on Thursday 6th and Friday 7th of November at the Hotel Monteleone, New Orleans. Further details are available from our web site at www.ced.co.uk/nw9y.htm

Version **5.03** of Spike2 is now released, freely downloadable for registered v5 users.
A demonstration version of the software is also available from www.ced.co.uk/upu.shtml

Version **4.17** of Spike2 is freely downloadable for registered v4 users.
Version **2.14** of Signal is also freely downloadable for registered v2 users.

SPIKE2

Q. I'm modifying my data then closing the file using the channel process function. Each time I return to the file it has reverted back and has not saved my changes. Why?

A. Spike2 data files are saved with a resource file of the same name but with the extension .s2r. This file contains information on how the file was last viewed in terms of the channels displayed, their order and the axis ranges. It also contains process information such as whether the channels were rectified.

Many people back up their data to CD which also often contains the original resource file. When we open a data file from CD to further process it, then close the file, Spike2 is not able to write back the data changes. Therefore, when the data file is opened again it appears with the data ranges etc. held in the original resource file or possibly worse if no resources could be read, the channels will all appear over 0 to 1 second and the y axes will be at the maximum range.

To help with this we have implemented a function Apply Resource File now available from the File pull down. To use this you will need to have a data window open in Spike2. Choose Apply Resource File and this will give you the chance to select an alternative file on any storage media to apply to the open data window. The information available in this file will then be used to size the open data window and apply visible gains and processes. This function can also be used to apply resource settings to a whole list of data files.

This is a v5 function.

SIGNAL

Q How can I overdraw a particular frame against the incoming data on-line?

A Normally Signal has a frame list which is overdrawn. This could mean that there are hundreds of frames to draw. A number of Voltage Clampers as well as general evoked response users have requested the attached script. It allows the user to nominate a single frame which is superimposed on the incoming data for easy comparison. The nominated frame can be updated at any time.

Although we've given you a script in this section we are working on implementing this as a built-in function within Signal.

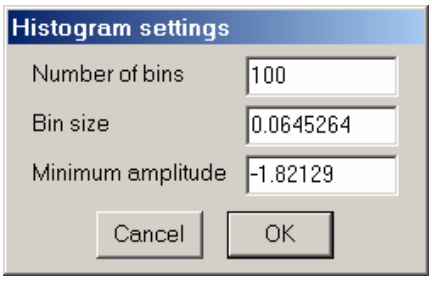
```
'*****************************
'$SetFrameOverdraw|Draw designated frame over incoming data
var dataView%;                'View handle of data file

dataView%:= FileNew(0,1);     'Open a new data file
Window(0,0,100,100);          'Display it full screen
SamplePause(0);               'No pause at sweep end
SampleWrite(1);               'Turn off writing to disk
DoToolbar();                  'Call script toolbar
Halt;

Func DoToolbar()
ToolbarClear();               'First make sure the toolbar is empty
ToolbarSet(1, "Quit");        'This quits the script
ToolbarSet(3, "Sample Start", Start%);     'Start sampling
ToolbarSet(4, "Sample Stop", Stop%);       'Stop sampling
ToolbarSet(6, "Set Frame", Set%);          'Set frame as reference
return Toolbar("Click 'Set Frame' for reference", 1023);
end;

Func Start%()
SampleStart();                'Sample first sweep
return 1;
end;

Func Stop%()
ToolbarClear(0);
SampleStop();                 'Stop sampling
return 1;
end;

Func Set%()
BuffCopy(0);                  'Copy frame data to buffer
SamplePause(1);
ToolbarSet(0,"Idle", Idle%);  'Start idle routine
return 1;
end;

Func Idle%()
View(dataView%);
if SampleStatus() > 2 then    'If paused at end of sweep
   OverDraw(0);               'Turn off overdraw
   BuffCopyTo(Frame());       'Copy buffer data to current frame
   OverDraw(1);               'Turn on overdraw
   SampleSweep();             'Sample next sweep
endif;
return 1;
end;
'*****************************
```

Scripts. Spike2

Q Is it possible to produce an amplitude histogram in Spike2?

A There are several types of analysis frequently requested for Spike2 which are not yet implemented as built-in functions. It is often possible to perform these analyses using the script language. The amplitude histogram is a good example of this.

The script 'NewsAmpHist.s2s' creates a basic amplitude histogram displaying the amount of time a waveform spends within specified amplitude bins. When the script is run you must first open a data file to analyse, then select a waveform channel. You then press the *Analyse* button and 2 cursors will appear which should be positioned around the area of interest. When this is done, press *OK* and a dialog will appear from which you specify the histogram settings.



Amplitude histogram settings dialog

The script automatically sets reasonable starting values. It calculates the maximum and minimum amplitudes in the data within the specified time range and divides this into 100 bins.

This is obviously a simple example script although it demonstrates some very useful features including array arithmetic and creation of dialogs, toolbars and user defined result views. The basic principles are quite straightforward and can be used as part of a more complex script for either off-line or on-line analysis. We have more examples of this and other types of custom analysis at CED so please contact us if you would like to discuss any particular requirements.

```
,*******************************
'$AmpHist.s2s|Script to produce an amplitude histogram from a waveform channel

var spchan%;                        'Channel to analyse
var numbin%:= 100;                  'Number of bins in histogram
var minamp;                         'Minimum amplitude in histogram
var dataView%;                      'Data file handle

ToolbarSet(1,"Quit");               'Set up toolbar
ToolbarSet(3,"Open file",Open%);
ToolbarSet(7,"Amplitude Histogram",AmpHist%);
Toolbar("Create Amplitude Histogram",1023);

Func Open%();                       'Open data file
if dataView% > 0 then               'Close any files opened previously by the script
       View(dataView%);
       FileClose(-1);
       dataView%:=0;
endif;
dataView%:=FileOpen("",0,0);
XRange(0,MaxTime());
WindowVisible(3);
SetChan%();
Return 1;
End;

Func SetChan%()                     'Select channel to analyse and hide all others
View(dataView%);
DlgCreate("Spike channel");
DlgChan(1,"Select channel",513);
DlgShow(spchan%);
```

```
ChanHide(-1);
ChanShow(spchan%);
Return 1;
End;

Func AmpHist%()                          'Main function
var hist%;                               'Histogram view handle
var bSize;                               'Bin size of histogram
var maxamp;                              'Maximum amplitude in histogram
var totAmp;                              'Total histogram amplitude
var bin%,i%;                             'Counters for loops
var tempVar;                             'Temporary variable for debugging
var data[1000000];                       'Array to contain data
var numPts%;                             'Number of data points in section
var data%[1000000];                      'Integer data array
var Out%;                                'Counter for number of points out of range
var Hist$;                               'Title for histogram
var ok%;

View(dataView%);
CursorSet(2);                            'Position cursors in data view
Interact("Set analysis time",1023);      'Allow user to set cursors
View(dataView%);
numPts%:=ChanData(spChan%,data[],cursor(1),cursor(2));      'Put data into array

'Section to estimate starting values
MinMax(spChan%,Cursor(1),Cursor(2),minAmp,maxAmp);    'Get range of data in 'section
totAmp := maxAmp - minAmp;           'Find total amplitude change
bSize:= totAmp / numBin%;            'Get bin size

DlgCreate("Histogram settings");     'Dialog to allow user to change parameters
DlgInteger(1,"Number of bins",1,1000);
DlgReal(2,"Bin size",0.0001,100);
DlgReal(3,"Minimum amplitude",-1000,1000);
ok% := DlgShow(numbin%,bSize,minamp);
if ok% then
     var binvals[numbin%];               'Working array
     maxamp:=minamp + (numbin% * bSize);       'Get new max amplitude based on
                                         'settings from dialog
     ArrConst(BinVals[],0);              'Set all array elements to zero
     ArrSub(data[],MinAmp);              'Subtract minimum histogram
                                         'amplitude from data
     ArrDiv(data[],bSize);               'Divide data by histogram bin size
                                         'to convert to bin number in histogram
     ArrConst(data%[],data[]);           'Truncate values by converting to integer
                                         'array
     Out%:=0;                            'Reset counter
     For i%:=0 to NumPts%-1 do           'Loop through each data point
          If (data%[i%]>=0) and (data%[i%]<NumBin%) then       'Check data is in
                                         'range of the histogram limits
               Bin%:=data%[i%];          'Get bin number of data point
               binvals[Bin%]+=1;         'Add 1 to count for appropriate bin
          else
               Out%+=1;
          endif;
     next;
     ArrMul(binVals[],View(dataView%).BinSize(spChan%));    'Convert count of data
                                                            'points to time
     Hist$:=Print$("Amplitude Histogram (Data out of range: %d)",Out%);
                                         'Set title of histogram window
     hist%:=SetResult(1,NumBin%,bSize,MinAmp,Hist$,ChanUnits$(spChan%),"Time(s)");
                                         'Create result view
     ArrConst([],binvals[]);             'Copy results to histogram
     View(dataView%).Window(0,0,100,50);
     View(hist%).Window(0,50,100,100);   'Set window size and display
     View(hist%).Optimise(0);
     View(hist%).WindowVisible(1);
endif;
```

```
Return 1;
End;
'******************************
```

Scripts. Signal

Q: In Signal, is it possible to establish a "sequential "configuration using a randomization of 5 states, chronologically followed by a separate 6th state, and finally a repetition of the randomized 5 states?

A: The basic idea behind multiple states in Signal is to identify data recorded under the different conditions for analysis. As well as marking the states Signal allows the user to control the sequencing of these states during a recording. Each state could, for example, contain a different set of stimulation pulses.

The built-in sequencing capabilities of Signal will cope with the vast majority of applications. The standard options are:

Numeric
Random
Protocol
Semi-random

Detailed explanations of these are given in the 'States sequencing' section of the Signal Help.

If more complex protocols are required, for example as specified in the question above then we can control the state sequencing by use of a script. The basic principle to achieve this is first to generate an array containing the order in which you wish the states to progress and then use an idle routine to set the state for the next sweep when the previous sweep has finished. The script should set up 'pause at sweep end' and the idle routine should check the current status of sampling whenever possible to detect when it is paused.

The artefact.sgs script supplied with Signal provides a good example of the method and can be used as a basic template as it sets up the idle routine leaving you to simply add in your state setting for the next sweep. The `Idle%()` function checks for the pause at the end of a sweep and then triggers the `FrameOK%()` function.

As an example, we can apply the following modifications to the artefact script to use 3 states and step through 10 frames in the order 1 1 2 1 3 3 1 2 1 1. First, copy the following variables to the beginning of the artefact.sgs script.

```
'******************************
var nextState% := 0;              'Index of next state to set
var states%[10];                  'Array containing required state sequence
states%[0] := 1;                  'Initialise array
states%[1] := 1;
states%[2] := 2;
states%[3] := 1;
states%[4] := 3;
states%[5] := 3;
states%[6] := 1;
states%[7] := 2;
states%[8] := 1;
states%[9] := 1;
'******************************
```

Next, replace the contents of the `Frameok%()` Function with:

```
'******************************
func FrameOK%()
if nextState% < 9 then    'If not more than item 9 in the array (10th item)
   nextState% += 1;                    'Increase state number by 1
   SampleState(states%[nextState%]);    'Set state as next value in array
endif;
```

```
return 1;
end;
'*****************************
```

The final step is to set the sequencing mode and initial state for the first frame sampled. To do this, add the lines below into the `Start%()` function before the `SampleStart()` command.

```
'*****************************
SampleStatesRun(0);                    'Set sequencing mode to manual
SampleState(states%[0]);               'Set initial state to first in array
'*****************************
```

Before running the script, ensure that at least 3 states are set up in the sampling configuration. When 'Start' is pressed it should then start sampling with the states changing according to the specified order.

From this simple example far more complex sequences can be set. The difficult part may then be to automatically generate the array containing the state orders. For tips on how to generate specific sequences please contact CED.

Did you know…?

In both Spike2 and Signal it is possible to use the `View().[]` notation to modify the data without needing to fill an array first.

For example, if you wish to suppress a DC range within a Power Spectrum we can use :

```
ArrConst(View(Frontview()).[:2],0);
```

If the current view is a result view then the above command will take the first 2 bins ([:2]) of the result array and make them zero. Equally, if you wished to find the mean….

```
Var Mean;
Var SD;

ArrSum(View(frontview()).[],Mean, SD);

Printlog("\nMean %f\tSum %f" ,mean,SD);
```

This produces a printout to the log view of the mean and standard deviation of the whole data range.

If you know the view handle (or the name) of the window you can replace `FrontView()` with that variable.

Recent questions

Q. When I delete a channel in Spike2 to save space, why is the file size not reduced?

A. The way that interleaved data or channels are stored in Spike2 means that the data although marked as 'deleted' is still there in the files data blocks. If you do want to reduce the size of the file use "Export As" from the file pull down. This has the benefit of rewriting the data file without the deleted data points but also allows you to cut sections from the data if you wish.

A benefit of the data not being fully deleted is that, if a channel is accidentally deleted, it may be recoverable using the UnDelCh script from the website.

User group

It will take a short time to set up an archive message board on our web site. As soon as this is done I will use this newsletter to announce that it is available. This seems to be the most sensible route to go rather than direct emailing both for security and for minimising spam.