CAMBRIDGE ELECTRONIC DESIGN LIMITED

eNEWSLETTER

#2
March 2020

Contents

Welcome

Spring is finally here; the tree blossoms are starting to appear outside our windows and warmer temperatures are on the horizon. Let's all hope for some good weather!

We'd like to thank all those who attended our recent training days in Paris and London, we hope you found them useful and would welcome any feedback to make our next training days even more successful.

Unfortunately, due to the ongoing COVID-19 outbreaks, the Experimental Biology 2020 meeting in San Diego has been cancelled and we are therefore no longer attending. The decision to cancel EB was not taken lightly by the society, with the health and safety of their members, exhibitors, attendees, staff, partners and communities being their top priority. We encourage everyone to follow their national health advice on travel and cleanliness and hope you all stay safe and healthy. We will however be attending Experimental Biology 2021 in Indianapolis and hope to see you there!

On the same note, Cambridge Neuroscience 2020, PDN 2020 and Canadian Neuroscience 2020 have also been postponed until further notice, and we fully expect other events to follow. Our website will be updated regularly with any changes to both our exhibiting and training schedules as the situation changes. If you have any questions, please don't hesitate to get in touch.

Training Days

Due to the ongoing COVID-19 outbreaks, all current training events will be scheduled on a case-by-case basis. We also offer remote training sessions via Skype either one-to-one or with small groups.

Join us and learn how to make the best use of Spike2 and Signal to save hours of repetitive analysis. These sessions are free to attend and are suitable for both existing and prospective users of our data acquisition and analysis systems. If you'd like to register your place, please get in touch.

If you are interested in hosting a training event in your local area, 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. Right-click, and select Save:



Latest versions of Spike2 and Signal

Spike2	Released	Signal	Released
Version 10.04	03/2020	Version 7.05a	02/2020
Version 9.10	02/2020	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

Bristol Brain Research 2020
Bristol,
UK
June 23rd 2020

UK Sensorimotor Conference
Birmingham,
UK
June 24th – 26th 2020

ISAC XXI 2020
Lisbon,
Portugal
June 29th – July 2nd 2020

12th FENS Forum of Neuroscience
Glasgow,
UK
July 11th – 15th 2020

ISEK XXIII 2020
Nagoya,
Japan
July 11th – 14th 2020

Europhysiology 2020
Berlin,
Germany
September 11th – 13th 2020

Neuroscience 2020
Washington, DC,
USA
October 24th – 28th 2020

PDN 2020
Cambridge,
UK
POSTPONED

Canadian Neuroscience 2020
Montreal,
Canada
POSTPONED

Back to contents

Script Spotlight 

In our last newsletter we introduced idle functions and their usefulness in monitoring changes or conditions in a script. We also mentioned the `ToolMake.s2s` script for Spike2 and `ToolMake.sgs` script for Signal, which we will now cover in more detail. These scripts create custom toolbars for use in your own scripts. These scripts are in your user data folder, for example: C:\Users\Username\Documents\Spike10\Scripts. They let you build toolbars by adding buttons, giving them a label, and then linking them to functions. Once you've added the required buttons, the *Write Code* button in the toolbar generates a skeleton script that you can add to your own code.

Toolbars allow the user to access multiple functions in a script. For example, a button for starting/stopping recording, buttons for different types of analysis and a *Quit* button to end the script. Toolbar buttons can have pop-up tips explaining the function and can be enabled/disabled to guide the user through a sequence of task.

The script function `ToolbarSet(item%, label$ {,func ff%()});` generates toolbar buttons. You link toolbar buttons to a user-defined function with the third argument. For example `ToolbarSet(1,"&Go",LetsGo%);` creates a button 1 linked to a function named `LetsGo%`. Using `&Go` in the label also links the button to the Alt+G key combination. Button 0 is special as the linked function is called as an Idle function (see our last newsletter).

The Toolbar() script command displays the toolbar and waits for user button clicks or linked key presses. A click on a button with no linked function closes the toolbar and the Toolbar() function returns the button number. However, if you link a function, the function is called and the function return code determines what happens:

```
Func LetsGo%()
…script actions…
return <code>;  'greater than 0 to keep running, else end Toolbar()
end;
```

In the example script, a toolbar has been created using ToolMake.s2s which creates a new data file, allows the user to perform two types analysis that continually update as new data is recorded, demonstrates the changing of a button's label as well as enabling/disabling other buttons.

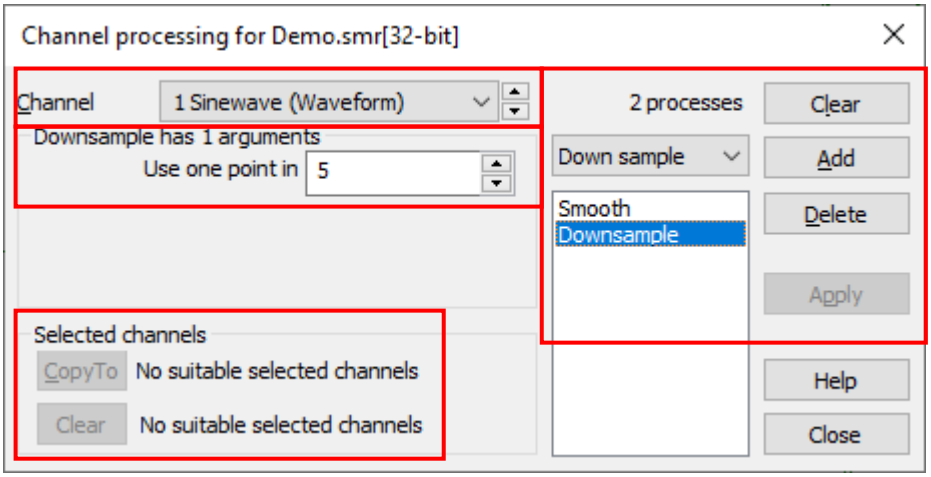
Back to contents

How do channel processes differ from sample processes?

Channel processes are operations which can be added to data channels both on and off-line. These processes do not replace the source data, instead you can think of each one as a set of calculations that are applied dynamically to update the view and are just as easily removed. This is different to the *Derived Channels* and *Real time sample processes* discussed in the last newsletter which change the source data; these types cannot be removed once data is sampled.

You can apply multiple *Channel processes* sequentially. For example: DC Remove, Rectify, Smooth to generate the envelope of an EEG signal. Spike2 applies *Channel processes* dynamically; each data fetch starts with the original data and applies the processes, in order. Most processes take very little time, so this extra work is not usually noticeable.

To open the *Channel processing* dialog, right-click the data view and select *Channel process…* or alternatively go to the *Analysis* menu in the toolbar to select it. Within the dialog, select the channel using the *Channel* drop-down list in the top left, and the process to apply from the drop-down list on the right. The *Add* button appends the process to the list. The *Delete* button removes the selected process from the list. The *Clear* button removes all processes. If you select a group of channels by clicking on the channel numbers, you can apply the current process settings to all or *Clear* all processes with the buttons in the bottom left corner of the dialog. Spike2 applies any changes you make immediately so you can see the result.



Some processes have *arguments* that modify the process behaviour. When required, these appear below the selected channel. The types of processes available depend on the channel type: waveform-based or event-based. Several processes use a user-defined time range defined by the argument *p* (short for *pre-* and *post-time*). The following processes are currently available in Spike2:

Rectify (waveform) – Replaces negative data values with equivalent positive values and leaves positive values unchanged

Smooth (waveform) – Low pass filter. The output at time *t* is the average value of the input from *t-p* to *t+p* seconds

DC Remove (waveform) – This is a high pass filter equivalent to the source data minus the result of the *Smooth* process described above; the channel offset becomes zero

Slope (waveform) – The slope of a waveform at time *t* obtained by calculating the mean of the points ahead (*t+p*) and behind (*t-p*) each data point. The slope is from the line through the centre of the points behind to those ahead

Time shift (any channel) – Shifts all data by the user specified time, in seconds. A positive number shifts the data forwards in time; a negative number shifts it backwards

Down sample (waveform) – Applying this process changes the sample rate of the waveform by taking one point in every *n*, with the user supplying *n*. This process would generally be used after filtering or smoothing a waveform and is faster than using interpolate

Interpolate (waveform) – This changes the channel sample rate and alignment. Interpolation is by cubic spline of the original data; no data is generated outside the time range of the original points. *Interpolate* has two arguments: *sample interval (s)* is the time between output samples, a*lign to (s)* aligns the output data to a time

Match channel (waveform) – This is the same as the *Interpolate*, except sample interval and alignment are copied from a nominated channel

RMS amplitude (waveform) – The RMS (root mean squared) value for each point is calculated at time *t* using data points from time *t-p* to *t+p*

Median filter (waveform) –The output at time *t* is the median value of the input data points from *t-p to t+p.* This is used to remove short artefacts from data

Fill gaps (waveform) – Waveform and RealWave channels can have gaps, this process fills gaps greater than a specified time with a fixed level and by linearly interpolating across smaller gaps

Skip NaN (waveform) – A NaN (Not a Number) is a floating-point value that is either undefined or infinite. These can occur when importing RealWave data into Spike2 or in a virtual channel if you divide by zero. This process removes these numbers creating gaps in the data (follow *Skip NaN* with *Fill gaps* to eliminate these)

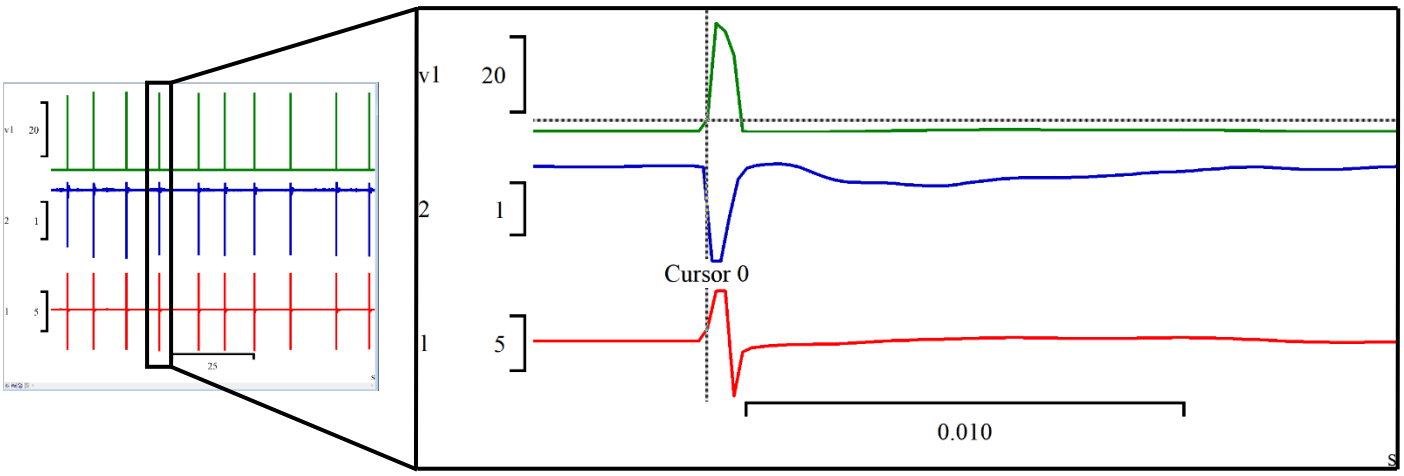
Debounce (event) – This process removes events that are too close to the previous event for all event-based channels except Level events. The *minimal interval* argument specifies how close events can get before removal

Back to contents

Scripts: Spike2

Recently, a Spike2 user came to us with an interesting problem. The user had multiple data files of evoked response data which needed processing into a single stimulus response curve. Each file records multiple stimulations of one intensity and their response on two channels of EMG data from surface electrodes. At first it seems like a relatively simple project, however, there were several hurdles to overcome.

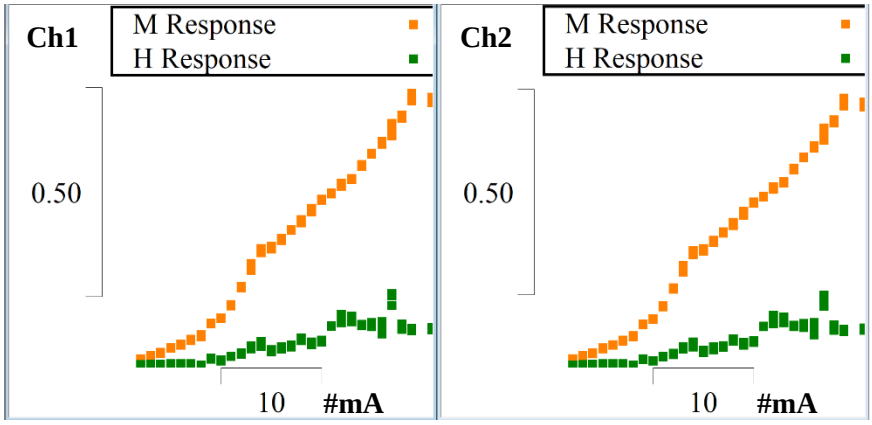
With the data in separate files, we needed a way to identify the stimulus intensity applied in each file in order to begin. The files were given a name that included the stimulus intensity in the same format: #mA<more text> where # stands for intensity. The script loads all the file names into an array, then scans the name of each file to extract the intensity.



Within the data there were no event markers identifying the time of stimulus; we had only the stimulation artefact to mark the response. However, the stimulus in both channels were of different polarity and magnitude, so to get around this the script squares the data in both channels and adds them together in a new virtual channel. Cursor 0 is used in active mode to find the rising threshold of the stimulus artefact above a set value on the virtual channel, and from this point the P and H wave measurements are taken from source channels 1 and 2. This is repeated for each data file, adding the stimulus level and response measurements to a created XY view as they are grabbed.

This type of script is a useful example of batch processing, where common elements in each data file may be used to create automated analysis instead of the user needing to search for the data and plot by hand.

Download the script and example library here.

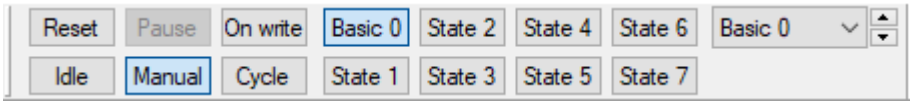


Back to contents

Signal

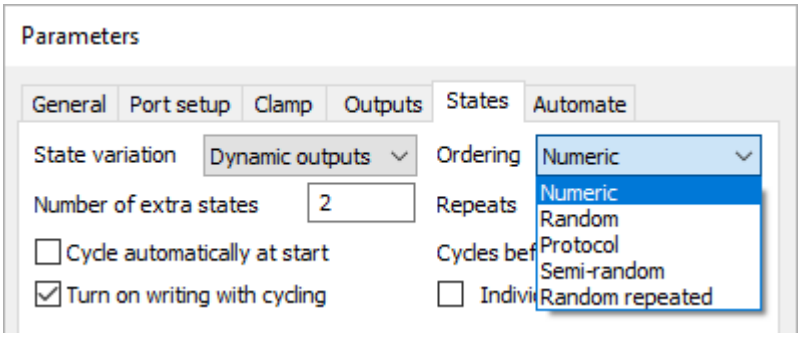
How can I choose which state to play during sampling?

In our last issue we introduced multiple frame states and discussed adding stimulation devices to Signal for external control. Manual control of which state to play is achieved by clicking the labelled buttons on the *States bar* during sampling. If this bar does not appear when creating a new sampling file, check you have multiple frame states enabled, then right click the toolbar and select the *States bar* from the list.



Alternatively, the states are cycled through automatically by selecting an order of states. This ordering, also referred to as sequencing mode, allows the user to specify the order in which the states are cycled. There are five different modes to choose from: *Numeric*, *Random*, *Semi-Random*, *Random repeated*, and *Protocol*. These modes are used in conjunction with the *Repeats* and *Cycles* before idle fields.

The *Repeats* field is used to specify how many times each state is repeated, with the option to have certain states repeat more/less by ticking the *Individual repeats* field and choosing the number of repeats for each state. The *Cycles before idle* field specifies how many full cycles (including repeats) of all states will happen. For example, entering 1 cycles through all states once and then switches to State 0. Entering 0 makes the states cycle indefinitely until sampling is manually stopped. One cycle of sequencing consists of #states x #repeats = #frames.



Numeric – This mode runs through each state in ascending order for the specified number of repeats, for the specified number of cycles (e.g. 1 1 1 | 2 2 2 | 3 3 3).

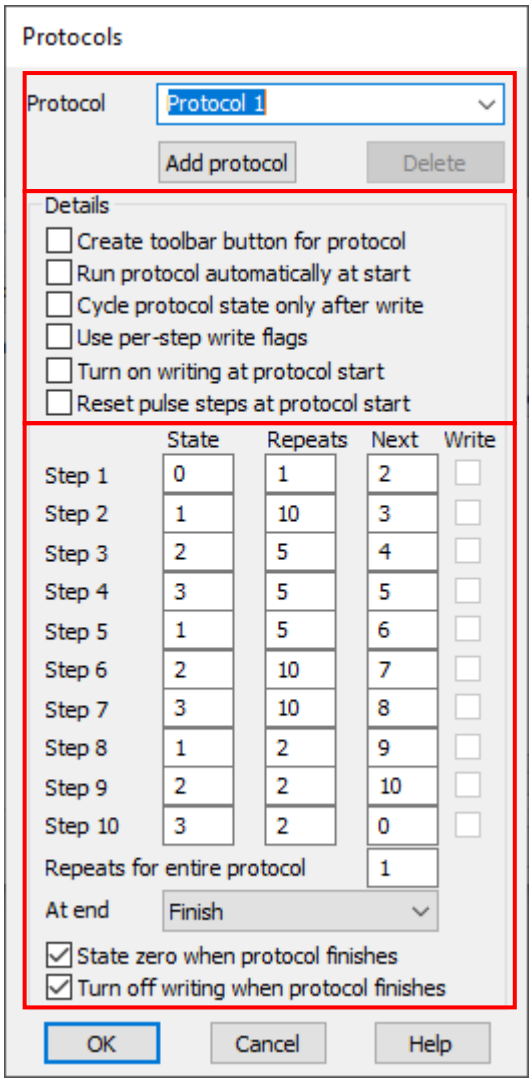
Random – One cycle of this mode loads all states for their specified number of repeats and fully randomises the order. Therefore, it's possible to get states repeated (e.g. 3 1 2 2 2 1 3 1 etc.). When a new cycle begins the order is completely randomised again.

Semi-random – This is a different method of randomisation where states are only randomised within one repetition. Therefore, if you have multiple repeats of states it's impossible to have more than one consecutive repeat of a state (e.g. 3 1 2 | 2 3 1 | 3 2 1 etc.).

Random repeated – This is another method of randomisation where the order of states is randomised, but each state repeats the number of times set by Repeats before moving onto the next state (e.g. 2 2 2 | 3 3 3 | 1 1 1).

Protocol – This mode is extremely useful and underutilised; *Protocol* sequencing gives the user full control of the experiment. Selecting *Protocol* from the *Ordering* drop down list provides you another button labelled *Protocols*. Clicking this opens the *Protocols* dialog where you can begin designing your experiment. Here you're able to name your protocol by overwriting the title in the top box. It's possible to create more than one protocol, just click *Add protocol* to create another and select which protocol to edit using the drop-down list above and click *Delete* to remove a protocol entirely.

Each protocol provides you with ten steps you're able to edit: altering which state will run, the number of repeats, and which step the protocol will then go to next. There's an option to enter the number of repeats for the entire protocol underneath the step settings, and a drop-down list to choose to either finish once the protocol is complete or move to another protocol.



In the top half of the dialog there are several more options available to you, those that require more explanation are described here:

- Ticking *Create toolbar button for protocol* when using more than one protocol allows you swap between protocols during sampling using the created toolbar.
- You are likely to use *Cycle protocol state only after write* if you are rejecting sweeps of data. Essentially if the data is not written to disk, then the protocol repeats the current state until subsequent sweeps are written to disk.
- *Use per-step write flags* is useful if you do not wish to write data to disk for a particular step of your protocol, for example if you were providing multiple stimulations in one step before moving to another step to record the response. Ticking the associated box enables the *Write* tick boxes to the right of the numbered steps, allowing you to select which steps are written to disk.
- *Reset pulse steps at protocol start* reverts varying pulses added to *Pulse Outputs* to their initial level. For example, a square pulse varying by 1V between 1V to 5V reverts to 1V each time the protocol is run with this option enabled, regardless of the level it was previously at.

Back to contents

Scripts: Signal

Occasionally we finish recording a data file only to find we have not set up the recording correctly, however we can often use the scripting language for changes to be applied retroactively instead starting again. A common problem Signal users face is the positioning of zero on the x-axis, in that an offset was not applied before recording data. In order to set the x axis offset, one should initially set the value in the *General* tab of the *Sampling Configuration*. If you are using Peri-trigger mode however, the time before trigger is defined in the *Peri* tab. It is possible to mistakenly use both the offset in the *General* tab and the offset in the *Peri* tab, which adds cumulatively.

Thankfully the script function `BinZero()` is available in version 7, which returns the position of the first bin in a time view but more importantly allows you to set this position in a memory view. This script makes use of the `BinZero()` function by copying all frames of a file to a memory view created through the Auto Average analysis function. Auto Average would normally average several frames with a specified overlap for all frames, however here we have used it to produce copies of waveform data in all frames. We can then use `BinZero()` to set a new x-axis offset for time zero, leaving the source data untouched.

Upon launching the script, you are be presented with a dialog asking you to specify the x-axis offset to apply. Each time you enter a value or use the increment slider the memory view updates the x axis. When you are happy with the changes, click OK. Whilst this only currently works with memory views, the `BinZero()` function will soon work with data files.

Back to contents

Did you know…?

If like us you spend a large amount of time staring at computer screens, sometimes a glaring white background can be tiring on the eyes. Thankfully, the colours of Spike2 and Signal can be altered to alleviate this strain. Spike2 even has a dark mode that inverts all the colours for you. Open the *Set Colour* palette on the Spike2 toolbar, and whilst holding *ctrl*, click the *Reset All* button. Ta dah, your Spike2 is now set to dark mode. To revert the colours back click the *Reset All* button without holding *ctrl*. The same function is not yet available for Signal; however, this little script will change all items set white to black, and vice versa:

```
var i%, r, g, b;
for i%:= 0 to 30 do
    ColourGet(0, i%, r, g, b);
    docase
    case r=1.0 and g=1.0 and b=1.0 then
        ColourSet(0, i%, 0.0, 0.0, 0.0);
    case r=0.0 and g=0.0 and b=0.0 then
        ColourSet(0, i%, 1.0, 1.0, 1.0);
    endcase;
next;
```

Back to contents

Recent Questions

How can I record the timing of my stimulus output in Signal?

In terms of evoked response, knowing the exact timing of your stimulus not only lets you accurately measure response times, it also provides valuable insight for your data analysis.

There are two ways to achieve this marking in your data. The first is simply a case of using the trigger output of your stimulation device and connecting it to the CED 1401. Connecting to the *Trigger* input enables you to set a configuration wherein the stimulus triggers a sweep of data, therefore the stimulus itself is at time point 0 in your fame of data. To set this, tick the box labelled *Sweep trigger* under *Options* in the *General* tab of the sampling configuration.

Alternatively, one may use the 1401s *Digital Inputs* which affords the use of digital markers to mark the stimulus in your data. Cursor 0 is then used to find data points and relative times of responses. These markers are both a time stamp and a code. Due to this, it's possible to record two separate triggers and mark them with different codes. It is important to note that digital markers cannot be recorded in fast sweep modes. Digital marker data is also only sampled when a low going TTL compatible pulse is detected on pin 23 of the 25-way D-type connector on the rear of the 1401. To set this up, ensure *Digital Markers* are enabled in the *General* tab of the *Sampling configuration*, and connect your stimulators *trigger output* to pin 23 of the digital inputs of the 1401.

Digital markers are also generated by the *Pulse Outputs* and *Text Sequencer* systems, for instance to show the stimulator was triggered by the 1401. In these cases, the marker data may either be read from the 1401 digital inputs or set directly when setting up your trigger pulses using the *Pulses configuration* in the *Output* tab of the sampling configuration. However, this only informs the user that a pulse was sent to the stimulator, not that the stimulator fired. Ideally one should be using digital markers created by a pulse delivered from the stimulator to the 1401.

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