Spike2 for Windows

Version 4

Version 4.00	January 2001
Version 4.03	September 2001
Version 4.04	January 2002
Version 4.05	February 2002
Version 4.06	March 2002
Version 4.07	April 2002
Version 4.08	June 2002
Version 4.09	July 2002
Version 4.10	August 2002
Version 4.11	September 2002
Version 4.14	February 2003
Version 4.16	May 2003
Version 4.17	June 2003
Version 4.18	October 2003
Version 4.19	January 2004

Published by:

Cambridge Electronic Design Limited
Science Park
Milton Road
Cambridge
CB4 0FE
UK

Telephone:	Cambridge (01223) 420186
	International +44 1223 420186
Fax:	Cambridge (01223) 420488
	International +44 1223 420488
Email:	info@ced.co.uk
Home page:	www.ced.co.uk

Table of Contents

Spike2 for Windows version 4

Introduction

With Spike2 version 4 and a CED 1401 family interface (not standard 1401) you can capture and analyse waveform, event and marker data and output precisely timed pulses and voltages using the familiar and easy to use Windows environments. This program will not run in Windows 3.x, however Spike2 for Windows version 2 does support that environment. If you have a standard 1401 you can sample data with Spike2 version 3.

You can arrange the windows to display the data within them to best advantage and cut and paste the results to other applications. Alternatively, you can obtain printer hard copy directly from the application. When you close a data file, Spike2 saves the screen format and channel display settings. When you open a file, Spike2 restores the configuration, so it is easy to resume work where you stopped in a previous session.

You can analyse sections of data by reading off values at and between cursors, or by applying the built-in functions, for example waveform averaging, digital filtering, spike detection, histogram formation and power spectra. More ambitious users can automate both data capture and analysis with scripts and the output sequencer. The script language is described in *The Spike2 script language* manual.

This manual describes the general interactive use of Spike2 version 4. You will find installation information on page 4.

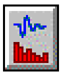
New features in version 4

We have tried very hard to keep version 4 of Spike2 compatible with version 3. It reads data files from all previous versions. Resource files are mostly compatible; some resource formats have changed to support new features. Scripts that ran with 3.18 should work unchanged with version 4 unless you used rasters in result views. New features include:

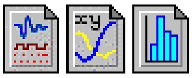
- Samples up to 100 channels of data, version 3 sampled up to 18 channels.
- RealWave data type stores waveforms as 32-bit real data.
- Faster sampling rates allowed with the Power1401 and Micro1401 mk II.
- There are now 10 vertical cursors and a button to add horizontal cursors.
- Active cursors automatically seek data features when cursor 0 moves.
- Automatic extraction of data to XY trend graphs.
- Multi-channel result views with multi-channel processing.
- Standard deviation and SEM in result views for waveform averages.
- File Import of wide range of data file types into Spike2 format.
- More control over the channel displays including channel order, size and colour.
- More control over x and y axes, can set tick spacing, draw as scale bar.
- Channels can be drawn on top of each other to simplify channel comparisons.
- Dynamic channel processing: rectify, smooth, AC couple, down-sample, slope.
- Both x and y axes can be used to scale and scroll the data. Just click and drag.
- New waveform calibration with a wide variety of calibration options from simple two point calibrations to areas under curves.
- Spike sorting trigger levels simplified, added limit levels, can save templates.
- New dialogs for easy edit of channel title, units and comment.
- Many dialogs now allow expressions where previously you could only type numbers.
- The Undo system knows how to undo much more, including channel order and size.
- Many dialogs have been changed to make them easier to use.
- Output sequencer rewritten with new instructions, graphical editor and many restrictions removed (but will still run sequences from previous versions).

There are many other improvements and more are planned. You can find a full list of new features, bug fixes and changes in the Revision History in the on-line Help. Licensed users of version 4 can download updated versions of Spike2 version 4 from our Web site `www.ced.co.uk` as they become available.

File icons

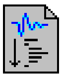


The various file types in Spike2 have icons so that you can easily recognise them in directory listings. The icon to the left is the Spike2 application icon that you double-click to launch Spike2.



The icon on the left is for Spike2 data files. The icon on the right is for saved result views. The central icon is for an XY file. If you double-click one of these it will launch the Spike2 application (if it is not already running) and open the file.

Spike2 can output sequences of pulses, sine waves and voltage levels as it samples data. Output sequence files have this icon.



This icon is for Spike2 script files. A Spike2 script can automate data capture and analysis operations and extend the capabilities of the Spike2 program.

Licence information

CED software is protected by both United Kingdom Copyright Law and International Treaty provisions. Unless you have purchased multiple licences, you are licensed to run one copy of the software. Each copy of the software is identified by a serial number (located in the Help menu About Spike2… dialog box). You may make archival copies of the software for the sole purpose of back up in case of damage to the original. You may install the software on more than one computer as long as there is **No Possibility** of it being used at one location while it is being used at another. If multiple simultaneous use is possible, you must purchase additional software licences.

Additional software licences

The original licensee of a CED software product can purchase additional licences to run multiple copies of the same software. CED supplies an additional manual set with each licence. CED does not supply additional software media. As these additional licences are at a substantially reduced price, there are limitations on their use:

1. The additional licences cannot be separated from the original software and are recorded at CED in the name of the original licensee.
2. All support for the software is expected to be through one nominated person, usually the original licensee.
3. The additional licensed copies are expected to be used on the same site and in the same building/laboratory and by people working within the same group.
4. When upgrades to the software become available that require payment, both the original licence and the additional licences must be upgraded together. If the upgrade price is date dependent, the date used is the date of purchase of the original licence. If some or all of the additional licences are no longer required, you can cancel the unwanted additional licences before the upgrade.
5. If you are the user of an additional licence and circumstances change such that you no longer meet the conditions for use of an additional licence, you may no longer use the software. In this case, with the agreement of the original licensee, it may be possible for you to purchase a full licence at a price that takes into account any monies paid for the additional licence. Contact CED to discuss your circumstances.
6. If you hold the original licence and you move, all licences are presumed to move with you unless you notify us that the software should be registered to someone else.

Direct access to the raw data

Some users may wish to write their own applications that manipulate the Spike2 data files directly. A C library *Spike2: Son data storage library* is available from CED together with documentation sufficient for an experienced C programmer to use it. The library documentation is also available as a `pdf` file on the Spike2 distribution CD. To install it, select Custom Install and check the Additional documentation box.

Versions of Spike2

This manual describes Spike2 for Windows version 4. The following versions of “Spike2 for …” exist, listed in more or less chronological order of release:

DOS	This is the original version of Spike2. Data and output sequencer files from this version are compatible with all other versions of Spike2. The script language is very different. The other versions include utilities to convert scripts (as far as is possible).
Macintosh 68k	This is the 68000 Macintosh version of Spike2. Data files, scripts and output sequences written for this version are compatible with the PowerPC and Windows versions. Scripts written for the PowerPC and Windows versions may not run on this version if they use new or extended features.
Windows version 2	This version will operate with Windows 3.1 and 3.11.
Windows version 3	The previous Windows version of the product.
PowerPC Macintosh	This is the Macintosh version of the product equivalent to the Windows version 2. It is compatible with the 68000 Macintosh version in data file, script and output sequencer format.
Windows version 4	The version described in this manual.

Using this manual

The first section of this manual introduces you to Spike2 by suggesting a few tasks you might undertake to get started with the system. We have supplied example data files for you to experiment with; there is no need to have your own data available at this time.

The second section explains the use of the menu commands. We suggest that you work through as much (or as little) of the familiarisation as you feel you need, then dip into the menu section as required for more detailed information.

We do not explain standard procedures, for example clicking and dragging, using menu short-cut keys or using a file open dialog; we expect that you are already familiar with them. We use standard Windows idioms wherever possible so that you feel at home and have a consistent interface to work with.

Once you are familiar with the program, you may wish to investigate the script language so you can automate your data capture and analysis. This is described in the separate reference manual *The Spike2 script language*. The on-line Help system duplicates the information in the manuals and is often the fastest way to look-up a topic. It is usually more up-to-date as the on-line Help is revised with each program revision.

The *Spike2 Training Course Manual* covers selected topics in more detail. It is particularly useful for script writers as the approach is much more descriptive than *The Spike2 script language* reference manual.

Hardware required

The absolute minimum system to run Spike2 is a 486 PC with 32 MB of memory with Windows 95. As I write this, in late 2000, almost any new PC is at least a Pentium 500 with 64 MB of memory and a 10 GB disk. This type of system runs Spike2 well. Like any Windows program, the more powerful the processor and the more memory your system has, the better Spike2 runs.

To sample data, you need a CED Micro1401, 1401*plus* or Power1401. See the *Owners Handbook* that came with your 1401 for hardware installation instructions. Spike2 comes with all required 1401 drivers and will install the Try1401 test program to verify correct 1401 operation.

Installation

Your installation disk(s) are serialised to personalise them to you. Please do not allow others to install unlicensed copies of Spike2.

From CD-ROM

Just put the CD-ROM in the drive and it will start the installation. You can also run the installation manually by opening the folder `Spike4xx` on the CD-ROM (the xx is the revision number), then open the `disk1` folder and run `setup.exe`.

From floppy disk

If you are installing from floppy disk you may wish to make backup copies of the installation disks before you begin and keep them in a safe place. We supply the disks write-protected to avoid accidental over-writing, but they can be physically damaged.

If we shipped you a CD-ROM and you must install from floppy disk you can create the disks by copying the contents of the folders `disk1`, `disk2` etc. in the `Spike4xx` folder to separate floppy disks. The resulting disks should hold only the files, NOT the folders!

Place the Spike2 for Windows installation disk 1 in a suitable drive (assumed to be a: below). Click Start, then Run, then type:

```
a:setup
```

During installation (CD-ROM and floppy disk)

You must select a suitable drive and folder for Spike2 and personalise your copy with your name and organisation. You can run versions 2, 3 and 4 on the same system (but not simultaneously) as long as they are in different folders. If you have a previous version on your system, install version 4 to a different folder. The installation program copies the Spike2 program plus help, demonstration, example and tutorial files. It can also install 1401 support in Windows NT 4. For other versions of Windows there are instructions for loading the 1401 drivers using the Windows Device Manager.

To copy additional technical documentation, which includes the SON filing system documentation and some technical notes on the Fast Fourier Transform, choose Custom installation.

After Installation

If you are new to Spike2, please work through the *Getting Started* tutorial in the *Spike2 for Windows* manual. Where you go next depends on your requirements. The *Spike2 Training Course Manual* is more descriptive than the other manuals, which are organised as reference material. However, it covers all versions of Spike2 and you will occasionally need to refer to the other manuals for version 4 specific details. The on-line Help in Spike2 has a lot of information; if in doubt use the `F1` key for Help.

Updating Spike2

You can update your copy of Spike2 to the latest version 4 release free of charge from our Web site: `www.ced.co.uk`. You can only update a correctly installed copy of Spike2 version 4. There are full instructions for downloading the update on the Web site.

Once you have downloaded the Spike2 update, you will find that the update program is very similar to the original installation, except that you must already have a properly installed copy of Spike2 for Windows version 4 on your computer.

Updates will include both bug fixes and new features. We will notify you by email (if we know your email address) of new releases. You can also register for this service on our web site. To stop emails, reply to them and ask to be removed from the list.

Removing Spike2

To remove Spike2: open the system Control Panel, select Add/Remove Programs, select CED Spike2 for Windows version 4 and click Remove. This removes files installed with Spike2; you will not lose files you created.

Introduction

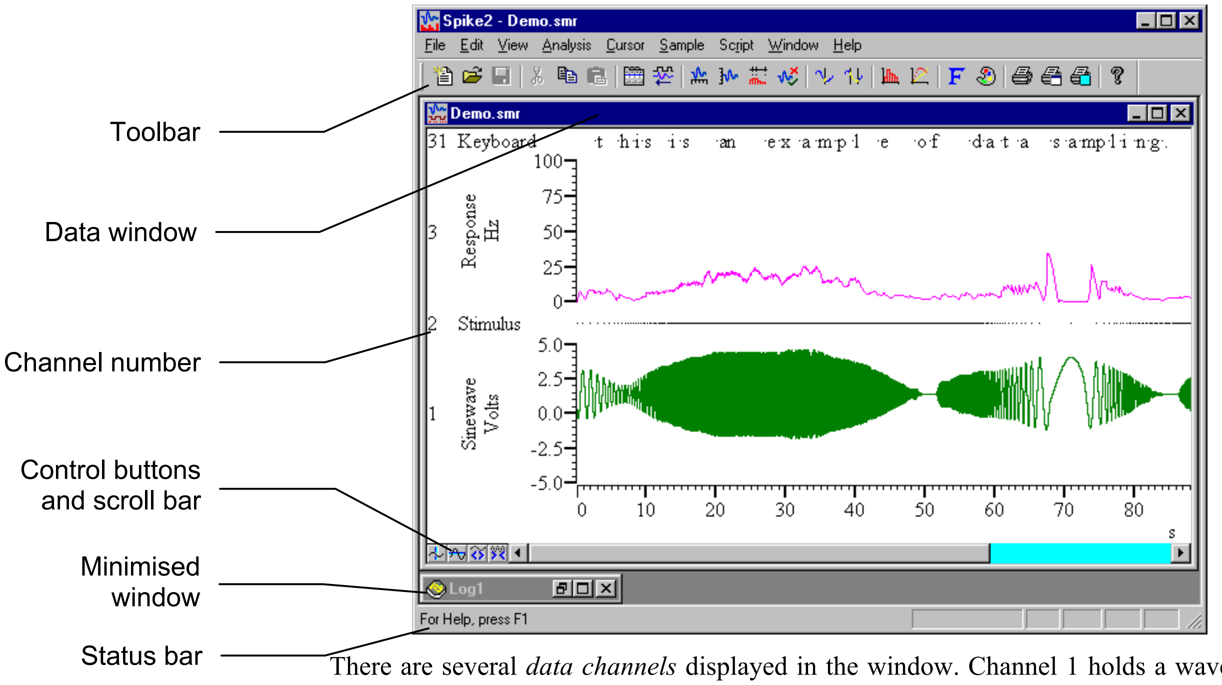
In this section you will open a Spike2 data file, manipulate the contents and familiarise yourself with the basic display controls. Instructions that you must follow to keep in step with the text are in **bold** type with a pointing finger. Explanations are in normal type.

Basic operations

The first task is to become familiar with the basic operations that are always needed to manipulate Spike2 files. We use a sample file called `demo.smr`. The Help menu Index option *Getting started* topic duplicates this chapter. Once you have started Spike2 you may prefer to follow the remainder of the chapter from the on-screen help.

From the Start button choose Programs, then Spike4, then demo.

Spike2 displays the file as it was last saved. The picture shows the state as shipped. You are looking at the raw data in the file. We call this a *time view*. It displays a time history of the data; the axis at the bottom is in seconds.



There are several *data channels* displayed in the window. Channel 1 holds a waveform, channel 2 holds events marked by dots, channel 3 holds events displayed as a mean frequency and channel 31 holds keyboard markers.

Selecting channels

Click on the channel number to select a channel. Spike2 highlights the channel number. Hold down the Shift key and click on a channel to select all channels between it and the last selection. Hold down the Control key to select discontinuous channels. Several commands work on a list of selected channels (for example y axis optimisation).

Toolbar and Status bar

The Toolbar is a shortcut to menu items. The Status bar provides information about your current activity. You can hide and show the bars with the View menu. You can also drag the Toolbar and "dock" it to any of the four sides. To find out what a Toolbar button does, leave the mouse pointer over the button for a few seconds.

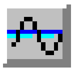
Spike2 has several other dockable toolbars. If you click the right-hand mouse button over an empty portion of the application window, or over a toolbar, a pop-up context menu appears from which you can show and hide any of the toolbars and the Status bar. Spike2 remembers the positions of toolbars each time you close the program and will restore them the next time you start. Each user who logs onto the system has a set of toolbar positions.

The bottom edge of the data window holds four buttons and a scroll bar. Try them.

The scroll bar controls movement through the file. You can also scroll the window one screen pixel with the `Left` and `Right` arrow keys. `Shift+Left` and `Shift+Right` move by several pixels and `Ctrl+Left` and `Ctrl+Right` move by half the screen width. The `Home` and `End` keys scroll to the start and end of the window.

Click this button or type `Ctrl+R` to halve the displayed time range (zoom in). The left hand edge of the display is fixed. `Ctrl+I` zooms in around the centre of the display.

Click this button or type `Ctrl+E` to double the displayed time range (zoom out). The left hand edge of the window is fixed unless the start plus the new width exceeds the file length, in which case the left edge moves back. If the new width would exceed the total length of the file, the entire file is displayed. `Ctrl+U` zooms around the centre.



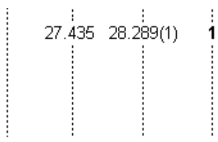
Click this button to add a horizontal cursor to the display. You can delete and manipulate these cursors through the Cursor menu. Up to 4 horizontal cursors are allowed. These cursors are mainly for the use of the script language. However, we expect to make more use of them from interactive dialogs in future releases of Spike2.

Click this button to add a vertical cursor to the display. Up to 10 vertical cursors can be present in a window. A cursor is a dashed line used to mark positions. You can remove cursors with the Cursor menu Delete command. You can add a cursor in these ways.

1. Click and release the button at the bottom left of the window (cursors 1 to 9).
2. Use the Cursor menu New cursor command (cursors 1 to 9).
3. Type `Ctrl+0` through `Ctrl+9` to show cursors 0 to 9 in the centre of the window.
4. Right click in the channel area and select New Cursor in the context menu.

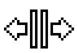
You can scroll the screen to centre it on a cursor with the `Ctrl+Shift+0` through `Ctrl+Shift+9` key combinations. If the cursor does not exist, this has no effect.

Click the cursor button so that a vertical cursor is visible. Drag the cursor and observe how the mouse pointer changes. Use the Cursor menu Label mode option.



There are four labelling styles for the cursor: no label, position, position and cursor number, and number alone. You can select the most appropriate for your application with the Cursor menu Label mode option. To avoid confusion between the cursor number and position, Spike2 draws the cursor number in **bold** type when it appears alone, and in brackets when it appears with the position.

The mouse pointer changes when it is over a cursor into one of two possible shapes to indicate the two actions you can take with a cursor:



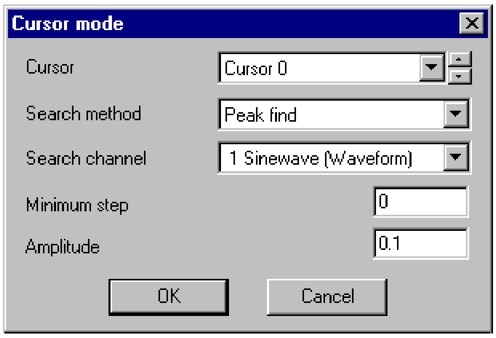
This shape indicates that you can drag the cursor from side to side. If you drag the cursor beyond the window edge, the window contents scroll to keep the cursor visible. The position vanishes when dragging unless the control key is down or you click on the label.

If you position the mouse pointer over the cursor label, the pointer changes to this shape to indicate that you can drag the label up and down the cursor. This is useful if you are preparing an image for publication and you need the cursor label to be clear of data.

If you click the right mouse button with the mouse pointer over a vertical cursor, the pop-up context menu includes commands to delete the cursor and also to set the cursor mode. Try deleting the cursors you have created. You can also delete one or all cursors from the Cursor menu.

Cursors can search for data features automatically. Use the Cursor menu Active modes command to open the Cursor mode dialog or right-click on a cursor and select Cursor mode from the context menu.

All vertical cursors can be Static or Active. A static cursor stays where you leave it. An active cursor can reposition itself by searching for user-defined data features. Cursor 0 is special; when active, you can command it to seek the next and previous data feature with keyboard commands. Cursors 1 to 9 are slaves of cursor 0; when active, they seek features in the order cursor 1 to cursor 9 when cursor 0 moves.



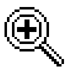
Set cursor 0 to Peak find on channel 1. Set Amplitude to 0.1, Minimum Step to 0 and click the OK button. Now try the `Ctrl+Shift+Right` key combination (`Right` is the right arrow key). Each time you press these keys, cursor 0 seeks the next peak on channel 1 that is at least 0.1 y axis units high. `Ctrl+Shift+Left` moves cursor 0 in the opposite direction.

Set cursor 0 search method to Static (no automatic search). Click on the data window then select one of the event channels (click the channel number) and zoom in so that only one or two events are visible on that channel. Type `Ctrl+Shift+Right` arrow.

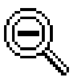
If you have an event channel marking regions of interest, you can jump to the next or previous event by selecting the event channel, then using the `Ctrl+Shift+Right` arrow or `Ctrl+Shift+Left` arrow key combinations. Spike2 searches for the nearest event to the centre of the screen in the specified direction. If more than one event channel is selected, all selected channels are scanned for the nearest event. This method does not work if cursor 0 is set to an active mode, so remember to set cursor 0 static first.

Move the mouse pointer to the waveform channel, clear of any cursor. Click and drag a rectangle round a waveform feature, then release the button.

This action zooms the display so that the area within the rectangle expands to fill the entire waveform channel region. If your rectangle covers more than one channel, only the time axis expands. If your rectangle fits in one channel and has zero width, the y (vertical) axis changes to display the selected range and the time axis remains unchanged.



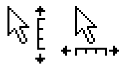
The mouse pointer changes to a magnifying glass when you hold the mouse button down in the data area to show that you are about to drag a rectangle or line to magnify the data.



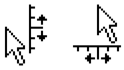
If you hold down the control key before you hold the mouse button down, the mouse pointer changes to the un-magnify symbol. If you drag a rectangle, the rectangle holding the channel area shrinks to the rectangle you have dragged.

Whichever method you use to scale the data, you can return to the previous display with Undo. If you decide not to expand the display after starting to drag, return the mouse pointer to the original click position (so the rectangle has zero width and height). The rectangle will vanish and you can release the button without changing the display.

Move the mouse pointer over the x and y axes and experiment with clicking and dragging the axes. Try it with the `Ctrl` key held down.



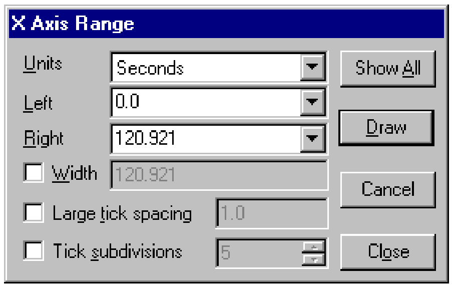
When the cursor is over the tick marks of an axis, you can drag the axis. This maintains the current axis scaling and the window moves to keep pace with the mouse pointer. You can do this with most x and y axes in Spike2. This is particularly useful for y axes as they do not have a vertical scroll bar. The window does not update until you release the mouse button. If you hold down the `Ctrl` key, the window will update continuously.



When the cursor is over the axis numbers, a click and drag changes the axis scaling. The effect depends on the position of zero on the axis. If the zero point is visible, the scaling is done around the zero point; the zero point is fixed and you drag the point you clicked towards or away from zero. If the zero point is not visible, the fixed point is the middle of the axis and you drag the point you clicked towards and away from the middle of the axis

In a time view, result view, or XY view, you can drag the y axis so as to invert the axis. You are not allowed to invert the x axis. You are not allowed to invert or scroll the y axis in the spike shape dialogs.

Now double-click on the time (x) axis of the display to bring up the X Axis Range dialog box. Experiment with the settings to vary the time axis.



Spike2 draws the x axis in seconds, hh:mm:ss (hours, minutes and seconds), or as time of day, as set by the Units field.

The Left and Right fields set the window start and end times. The Width field shows the window width. Set the left and right positions, or check the Width box and set the left position and the window width.

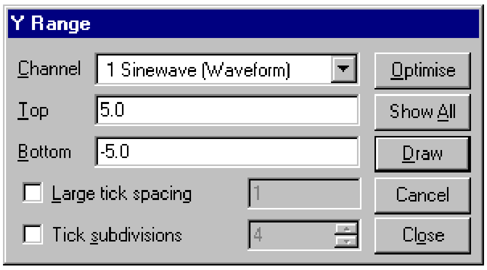
You can type new positions or use the drop down lists next to each field that give you access to cursor positions. The Show All button expands the time axis to display all the data. The Draw button updates the display to show the time range set by the Left, Right and Width fields.

The format for a time is `{{{days:}hours:}minutes:}seconds` where the seconds may include a decimal point and items enclosed in curly brackets are optional. If you have a file that spans multiple days, the first day (regardless of the day of the month) is day 0, the second is day 1 and so on. Each colon in the time promotes the number to the left of the colon from seconds to minutes to hours to days. Times may only contain numbers and colons. One decimal point is allowed at the end of the time to introduce fractional seconds.

In addition to typing times, or selecting a time from the drop-down list, you can type in expressions using the maths symbols `+` (add), `-` (subtract), `*` (multiply) and `/` (divide). You can also use round brackets. For example, to display from 1 second before cursor 1 to one second past cursor 1 set Left to `Cursor(1)-1` and Right to `Cursor(1)+1`. The Draw button is disabled if you type an invalid expression, or if the Right value is less than or equal to the Left value or if the new range is the same as the current range.

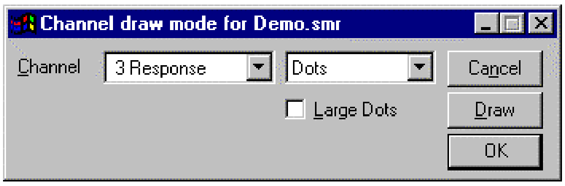
The Large tick spacing and Tick subdivisions fields let you customise the axis. Values that would produce an illegible axis are ignored. Changes to these fields cause the axis to change immediately; you do not need to click Draw.

☞ **Now double-click on the y axis of the waveform channel to open the Y Range dialog. Experiment with adjusting the y axis ranges on different channels.**

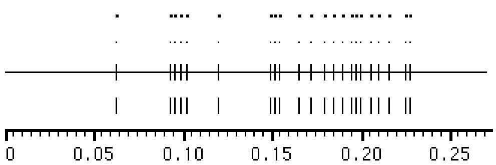


This dialog changes the y axis range of one or more channels. The Channel field is a drop down list from which you can select any channel with a y axis, or all channels with y axes, or all selected channels. You can either Optimise the display, which makes sure that all the data in the window on the selected channel(s) fits in the y axis range, or you can set the y axis limits as numbers, or you can use Show All to display the full range of the y axis. You can also control the axis tick spacing, as for the x axis.

☞ **Open the View menu Channel Draw Mode dialog. Experiment with different drawing modes for channel 3.**

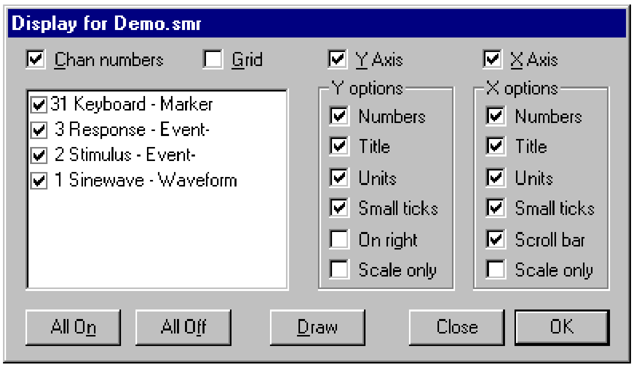


Spike2 data files hold two basic channel types: waveform and event. Waveform channels hold values representing the waveform amplitude at equal time intervals. Event channels hold the times at which something happened. There are four waveform drawing modes and many event drawing modes. For now, select channel 3 and try out the event modes.



The simplest method is to draw the event channel as dots. You can choose large or small dots (small dots can be difficult to see on some displays or when printed). You can also select Lines in place of Dots. The picture shows the result of both types of display. If you select lines for a marker channel (like the keyboard channel), the marker codes are not drawn. Click the Draw button to cause an update without closing the dialog. See the *View menu* chapter for a description of all drawing modes.

☞ **Open the View menu Show/Hide Channels dialog. Experiment with the channels, axes and grid.**

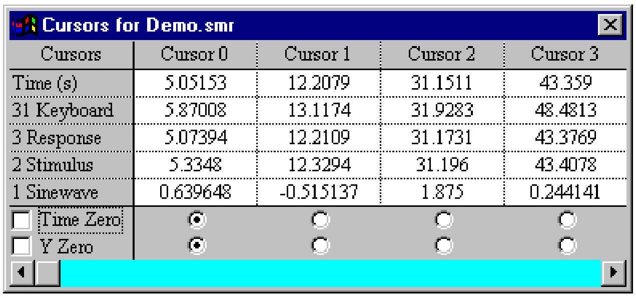


This dialog sets the channels to display in your window. Spike2 can handle up to 32 channels in a file, so the ability to display selected channels is quite important if you are to see any detail! The list on the left of the dialog holds all the channels that can be displayed.

You can show and hide the axes, grid and scroll bar in the window from this dialog and control the appearance of the x and y axes. Check the boxes next to the items for display and click the Draw button to see the result. The Scale only option draws axes as scale bars.

Make sure you have some cursors in the window. Use the Cursor menu Display Y Values command. Experiment with cursor positions and channel draw modes.

This is the Cursor Values dialog. The columns show the cursor times and data values for channels with a y axis. Channels with no axis display the time of the next event after the cursor. Check Time zero and Y zero for relative rather than absolute measurements.



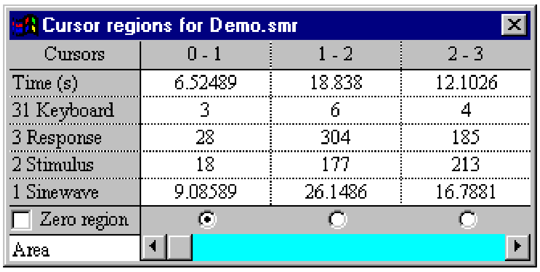
Cursors	Cursor 0	Cursor 1	Cursor 2	Cursor 3
Time (s)	5.05153	12.2079	31.1511	43.359
31 Keyboard	5.87008	13.1174	31.9283	48.4813
3 Response	5.07394	12.2109	31.1731	43.3769
2 Stimulus	5.3348	12.3294	31.196	43.4078
1 Sinewave	0.639648	-0.515137	1.875	0.244141
Time Zero				
Y Zero				

You can choose which cursor to make the reference. The values at the reference cursor are unchanged, but the values at the other cursors have the value at the reference cursor subtracted. You can use this feature to show how data values have changed from a reference value.

If you move the cursors or change the channel display mode, the values in this window update to reflect the change of position. Likewise, if you add or remove data channels or cursors from the display, the cursor dialog will change.

You can select fields in this dialog and copy them to the clipboard. Click on a field to select it or drag across the data area to make a rectangular selection of fields. Click at the top or left hand edge to select an entire column or row. Click in the top left hand box to select all the fields. Hold down the control key and click at the top or left hand edge for non-contiguous selection of rows or columns.

Now open the Cursor menu Cursor Regions dialog. Experiment with changing cursor positions and channel display types.



Cursors	0 - 1	1 - 2	2 - 3
Time (s)	6.52489	18.838	12.1026
31 Keyboard	3	6	4
3 Response	28	304	185
2 Stimulus	18	177	213
1 Sinewave	9.08589	26.1486	16.7881
Zero region			
Area			

The regions dialog looks at the data values between cursors. There are many modes of operation, for example Area, Mean and Slope. See the *Cursor menu* chapter for a full list. To change the mode, click in the box at the bottom left of the dialog (holding “Area” in the picture) and you will see a drop down list of all available modes. You can also make measurements relative to one of the regions by checking the Zero region box and choosing a reference region.

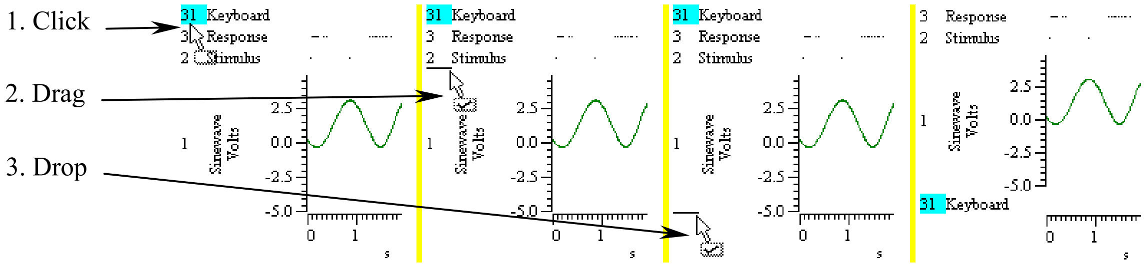
For a waveform channel, the Area is the area between the waveform and the y zero level, the Mean is the mean level of the signal. If you select Slope, Spike2 calculates and displays the gradient of the least-squares best-fit line to the data.

For an event channel, the Area is the number of events in the region. Mean is the count of events divided by the width of the region. Slope has no meaning for an event channel.

Long drawing and calculation times

Spike2 can display huge data files (sizes of hundreds of megabytes and more). However, it takes time to move such large quantities of data from disk into memory. If you accidentally try to draw a huge file, or use the Cursor regions dialog to calculate a value from a huge file, you may end up waiting for a long time. You can break out of such operations with the `Ctrl+Break` key combination.

Use the View menu Standard Display command. Click on the Keyboard channel number (31) and drag it down over the other channel numbers.

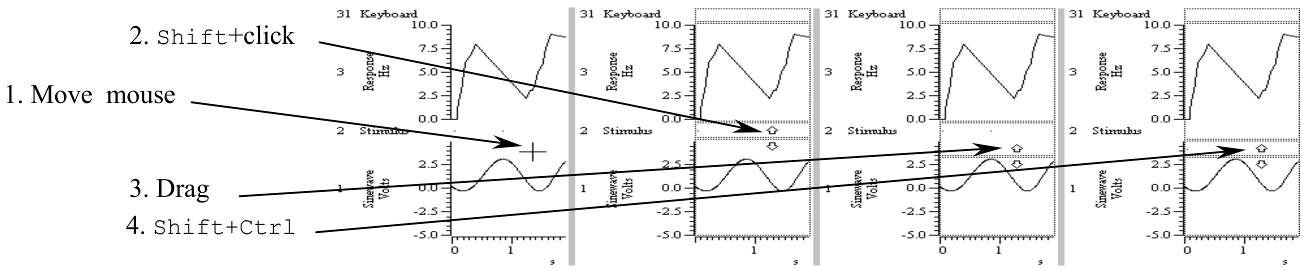


As the mouse pointer passes over each channel, a horizontal line appears above or below the channel. This horizontal line shows where the selected channel will be dropped. Drag until you have a horizontal line below channel 1 and release the mouse button. Channel 31 will now move to the bottom of the channel list. Type `Ctrl+Z` or use the Edit menu Undo to remove your change.

You can move more than one channel at a time. Spike2 moves all the channels that are selected when you start the drag operation. For example, hold down `Ctrl` and click on the channel 3 number. Keep `Ctrl` down and click and drag the channel 2 number. When you release, both channels will move. The mouse pointer shows a tick when you are in a position where dropping will work.

The usual Spike2 channel order is with low numbers at the bottom of the screen. If you prefer low numbers at the top of the screen, open the Edit menu Preferences and check Standard Display shows lowest numbered channel at the top, then use the View menu Standard Display command.

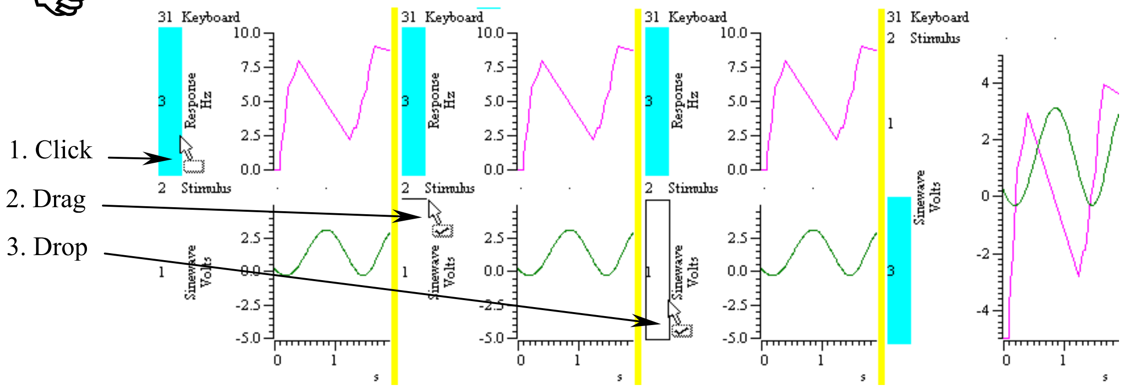
Change the channel 3 drawing mode to Mean frequency. Hold down the `Shift` key and move the mouse over the data area. Hold the `Shift` key down and click. Drag up and down and release the mouse.



When you click with `Shift` down, the mouse jumps to the nearest channel boundary and you can change the boundary position by dragging. With `Shift` down, you can move the edge up and down as far as the next channel edge. You can undo changes or use Standard Display to restore normal sizes.

If you add `Ctrl`, all channels with a Y axis are scaled. If there are no channels with a Y axis above or below the drag point, then all channels scale. You can force all channels to scale by lifting your finger off the `Shift` key (leaving `Ctrl` down) after you start to drag the boundary.

Click the "3" of channel 3 and drag it on top of Channel 1 and release.



The channels now share the same space with the channel numbers stacked up next to the y axis. The visible y axis is for the top channel number in the stack. To move a stacked channel to the top, double-click the channel number. Stacked channels keep their own y axes and scaling. To remove a channel, drag the channel number to a new position.

When you drag channels, and at least one of the selected channels has a Y axis, you can drop the channels with a Y axis on top of another channel with a Y axis. As you drag, a hollow rectangle appears around suitable dropping zones. You can also drop between channels when a horizontal line appears.

Merged channels are drawn such that the channel with the visible Y axis is drawn last. If you have a channel that fills in areas, such as a sonogram or an event channel drawn as rate mode, put it at the bottom of the stack, as it will mask channels below it in the stack.

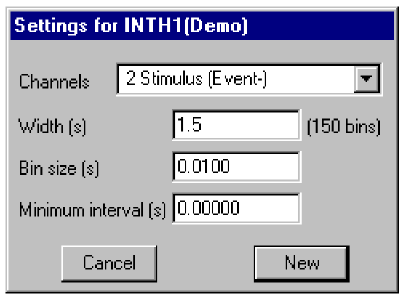
Result views

So far, you have been looking at windows holding raw, unprocessed data. We call these *Time views*. There is another type of data window, called a *Result view*, which holds the result of analysing time view data. There are two steps in the analysis:

1. Set the type of analysis, the channels to use, the width of the analysis or bins to generate and any other parameters. This creates a new, empty result window.
2. Set a time range in the time view. Spike2 processes the data and adds it to the result.

Repeat step two as often as is required to accumulate results from different sections of data. Most analyses generate windows that can be thought of as histograms or waveforms, that is a list (array) of data values equally spaced in the x direction.

Make the original time view of the data the current window by clicking on it. You may find it easier if you close all the other windows first. Use the Analysis menu New Result View command to select an Interval Histogram.



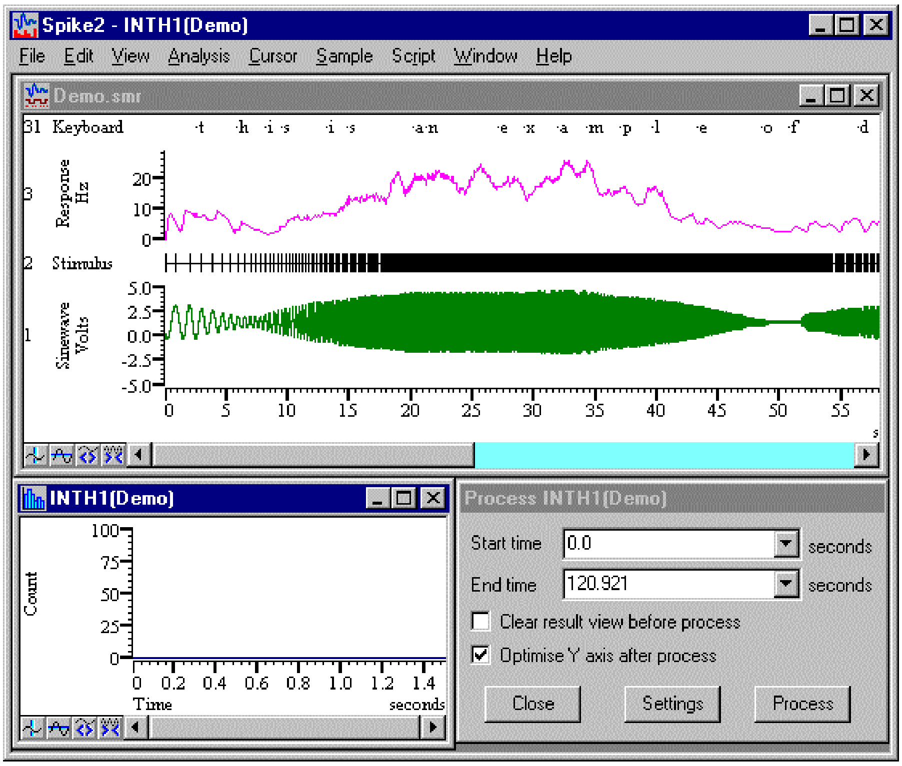
The new dialog is prompting you for information to define the new window. There are four fields to fill in that define the interval histogram. The first field selects the channel to analyse. You can select any channel that does not hold waveform data. The drop down channel list only includes suitable channels for analysis.

If you prefer, you can type in a channel list made up from channel numbers and channel ranges separated by commas. A channel range is a list of consecutive channels written as the first and last channel separated by two dots. For example, 1..3 means the same as 1,2,3 and 3..1 is the same as 3,2,1. You can combine the two, 1,3,4..8 is a valid channel list.

The next two fields set the histogram width and bin size in seconds. Spike2 calculates the number of bins and displays them to the right of Width. The width is limited only by the available memory to store the bins. The histogram bins must be at least one Spike2 clock tick wide (you set the clock tick size when configuring the data sampling and it is usually in the range 2 to 50 µs). The bin width is rounded to a multiple of this clock tick, so may not stay exactly as you set it. The last field sets the minimum interval that appears in the new window. In most cases you will leave this set to 0.0 seconds.

The first thing to do is to select a channel to analyse. A suitable one for our purposes is channel 2, so select this one now using the drop down list. Set the remaining fields as they are in the picture above; 1.5 seconds wide, 0.01 seconds bin width (10 milliseconds), and a minimum interval of 0.

Once you have set these values, click the New button to generate the new result window. Now set the region of data to analyse.

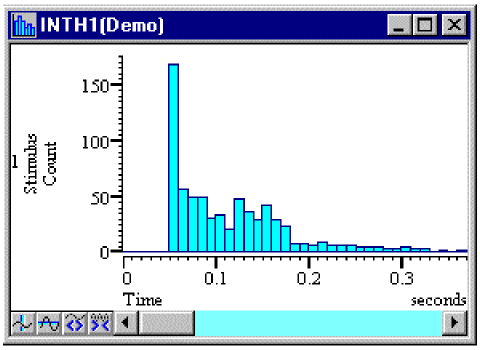


When you click the New button Spike2 creates a new window ready to display the result of the analysis, the setup dialog vanishes and the Process dialog appears. You must now set the region of the data document to analyse.

The two check boxes in the Process dialog determine how to treat the result of the analysis. You can choose to clear the result window contents before you analyse the data, otherwise each new result is added to the previous one. You can also choose to optimise the result view display after each analysis so that the full range of the data is visible.

If you decide that you have not set the original parameters correctly, you can click on the Settings button to go back to the previous step and correct the values.

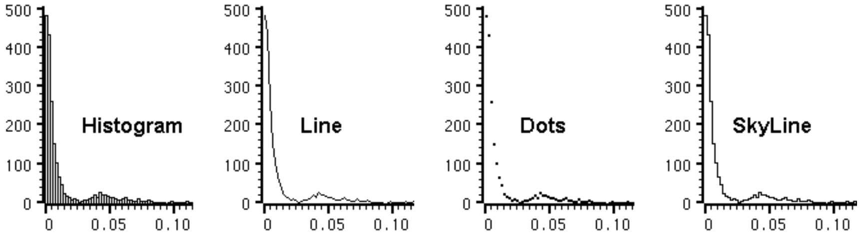
When you have set the time region, click the Process button.



The dialog vanishes and the result window shows the analysed data. The picture shows the data with the zoom-in button clicked twice.

You can recall the analysis dialog by selecting the Process command from the Analysis menu. Do this now and click the Process button again. The data in the result window will double in size (as long as you have not checked the Clear result view before process box).

Experiment with the Channel Draw Mode command in the View menu.



There are four different drawing styles available for result window histograms: Histogram, Line, Dots and SkyLine. These styles are self-explanatory. The various analysis routines that create result windows will select an appropriate style.

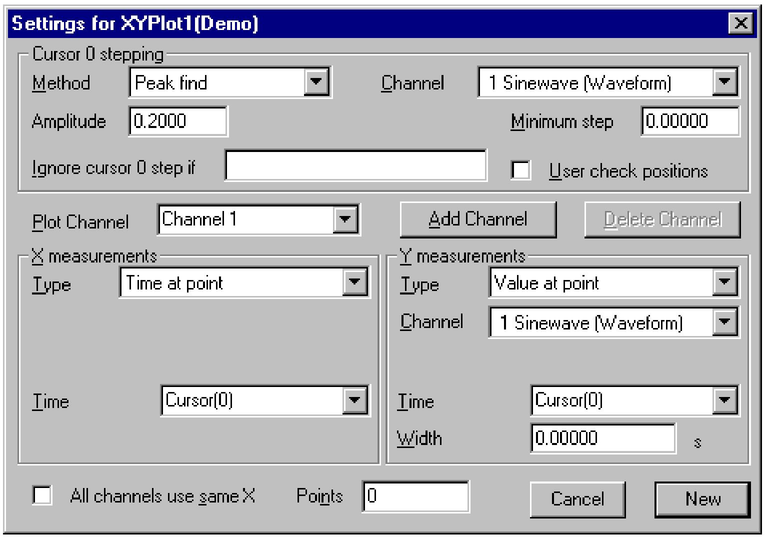
Experiment with cursors in this new window.

You will find that the cursors behave in a very similar manner to the original time window. You can create and position up to 9 vertical cursors and 4 horizontal cursors and display the Cursor regions and Cursor values dialogs from the Cursor menu. However, there are no active cursors in a result window and no vertical cursor 0.

XY views

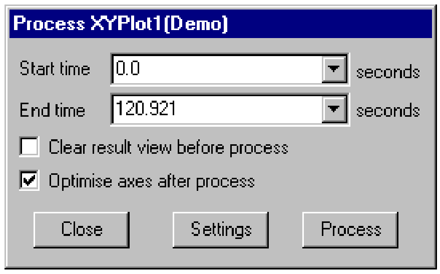
In addition to Time and result views, there are XY views. These hold up to 256 data channels that share the same x and y axes. Each channel is a list of (x,y) co-ordinates and has its own point marking style, line style and colour. XY views are commonly used to draw trend plots of data extracted from time views and for user-defined images and graphs generated by a script. We will start by generating a trend plot.

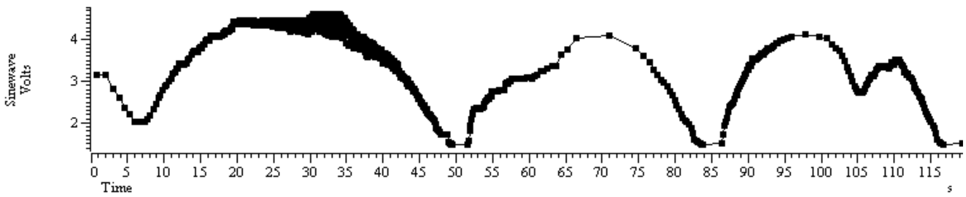
Close all windows except demo.smr. Set cursor 0 to Peak find mode as you did earlier, then use the Analysis menu Measurements->XY view command.



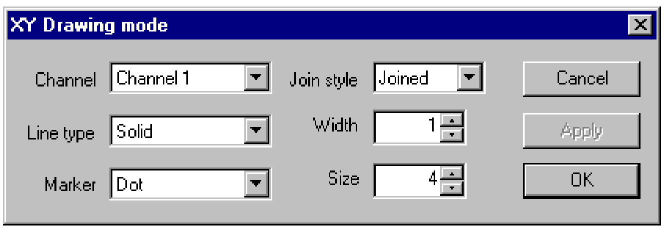
In a trend plot, cursor 0 steps through a data region. At each step, x and y values are derived from the data and added to an XY view. We will plot the peak heights of the waveform channel in the `demo.smr` file. If you set up cursor 0 correctly, the Cursor 0 stepping section of your dialog will match the picture; adjust it to match if it differs.

Set the X measurement area and the Y measurement area to match the picture. Set the Type fields first, then adjust the remaining fields and finally click the New button. A new XY window opens together with the Process dialog. The Start time and End time fields set the limits for cursor 0 stepping. In this case we want to step through the entire file. Now click the Process button.

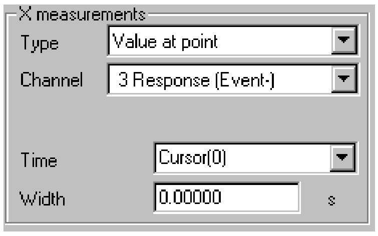




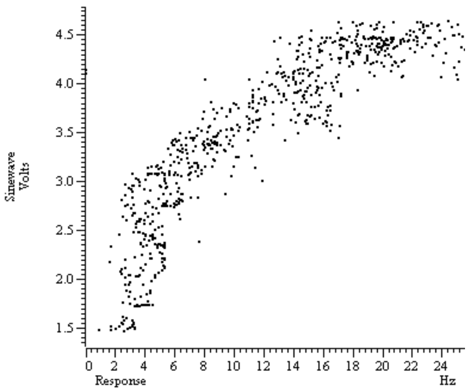
The result looks a little blotchy! Double click the data area of the XY window to open the XY Drawing mode dialog. Set the Size field to 0 and click Apply to hide the markers at each data point. You can use a variety of drawing styles. When you have finished, click OK.



So, what have we done? In each case our x measurement was set to a time, and that time was the cursor 0 position. The y measurement was the value of channel 1 at a time, again being the cursor 0 position. Cursor 0 was set to step through the data on channel 1 looking for peaks in the data. The result is a graph of peaks of each cycle of data.



Suppose we suspected that the event rate of channel 3 is related to the amplitude of the peaks. To test this we will change our plot. Click on the `demo.smr` window and change the drawing mode of channel 3 to Mean frequency over 1 second. Right click on the XY view and select Process settings. Change the Type to Value at point, select channel 3, then click the Change button. You will be warned that this will delete the previous result, so say this is OK.



The result will look a little messy with lines joining the points, so activate the XY Drawing mode dialog and set the Join style to Not joined and the marker size to 2. The result should look something like the picture.

You will find detailed information about all the different measurements you can make in a trend plot in the *Analysis menu* chapter. The *Cursor menu* chapter has details of the active cursors.

Use the Script menu Run Script command and select the Load and run… command. Locate the `Scripts` folder (in the folder where you installed Spike2), and open the file `clock.s2s`.

Spike2 will load and run this script. It generates an analogue clock in an XY view. You can move and resize the clock window. You can stop the script running (and regain control of Spike2) by clicking on the OK button at the upper right hand side of the Spike2 window. You can read more about XY views in the script language manual and in the *Spike2 Training Course* manual.

Summary

If you have followed this chapter, you will now be familiar with the basic actions required to use Spike2. The next chapter contains useful general information about Spike2 and the following one deals with the special actions required to configure the system for sampling your own data. The remainder of the manual covers the menu commands in the system, the memory buffer, the sequenced output system, cut and paste to other applications, printing, the spike shape capture and editing module, digital filtering and utility programs for fixing damaged files and testing your 1401.

If you have a Spike2 supported programmable signal conditioner, see the *Programmable Signal Conditioners* chapter. Standard 1401 or 1401*plus* users with the optional 1401-18 discriminator card should see the *1401-18 Programmable Discriminator* chapter.

The *Script menu* chapter describes the menu options that control the script system. The script language itself is not covered in this manual; see the companion text *The Spike2 script language* for a full description.

General information

This chapter gathers together information that would otherwise be scattered and repeated throughout the manual. Channel lists are used in many dialogs and also in the script language. Expressions can be used in many dialogs where x axis values are wanted and also more generally. There are many keyboard shortcuts in Spike2; they are gathered together here. The command line lets you control Spike2 from other programs; script users can even launch another copy of Spike2.

Channel lists

In many places where you are prompted for a channel in Spike2 version 4, you can select a channel from a drop down list, or you can type a channel list. A channel list is a list of channel numbers or channel ranges separated by commas. A channel range is two channel numbers separated by two periods, for example `4..7`, which is equivalent to channels 4, 5, 6 and 7. The channel range `7..4` is equivalent to channels 7, 6, 5 and 4. The following channel list:

```
1,3..5,7
```

means channels 1, 3, 4, 5 and 7. In most cases, Spike2 checks channel lists and removes channels that are not suitable for the operation. For example, if you open the `demo.smr` file supplied with Spike2, select an Interval histogram and type a channel list of 1..32 and then click on another field, Spike2 reformats the list to 2,3,31 as these are the only suitable channels.

It is not an error for a channel list to include unsuitable channels, however it is an error for a channel list to include no suitable channels.

Channel lists can also be used in script commands, for example:

```
ChanShow("1..4");
```

Script commands that will accept this format describe the argument as `cSpc`. You can find more information about channel specifications in the script language documentation.

Dialog expressions

There are many dialogs in Spike2 that will accept an expression in place of a number. These expressions can be divided into two types: numeric expressions and view-based expressions.

Numeric expressions

A numeric expression is composed of numbers, the arithmetic operators `+`, `-`, `*` and `/`, the logical operators <, <=, =, >=, <> and `?` and round brackets `(` and `)`. Most of these operators are familiar and do what you expect. The result of a logical comparison is 1 if the result is true and 0 if the result is false. You may not have come across `?` which is used as:

```
expr1 ? expr2 : expr3
```

The symbols `expr1`, `expr2` and `expr3` stand for numerical expressions. The result is `expr2` if `expr1` evaluates to a non-zero value and `expr3` if `expr1` evaluates to zero. This can be used in the Cursor mode dialog to give a cursor position if a search fails based on some other information, for example:

```
Cursor(2)>Cursor(1) ? Cursor(2) : Cursor(1)
```

This evaluates to the position of the rightmost of cursors 1 and 2.

If you write expressions involving more than one operator, for example 1+2*3 you need to know if this is evaluated as (1+2)*3 or as 1+(2*3). This is determined by the precedence level of the operators.

View-based expressions

A view-based expression follows the rules for numeric expressions and allows references to cursor positions and positions along the x axis. If a dialog field is documented as allowing expressions, and the field supplies an x axis position (for example a time), then you are allowed a view-based expression. The following are allowed:

`Cursor(n)`	Where n is 0 to 9 returns the position of the cursor. If the cursor does not exist or the position is invalid, the expression evaluation fails.
`C0` to `C9`	This is shorthand for `Cursor(0)` to `Cursor(9)`.
`XLow()`	The left hand end of the visible x axis in seconds for a time view, bins for a result view, and x axis units for a XY view.
`XHigh()`	The right hand end of the visible x axis.
`MaxTime()`	The right hand end of the time axis in seconds for a time view and bins for a result view. It is not valid in an XY view.

You can add `View(-1).` before these expressions to force the expression to be evaluated for the time view linked to the current view, for example `View(-1).Cursor(0)`. This is required when the current view is a result view and you wish to access timing information from the time view that the result view is based on.

Times as numbers

All times are in units of seconds. However, where a time is typed into a dialog you can use `{{{days:}hours:}minutes:}seconds` where the seconds may include a decimal point and items enclosed in curly brackets are optional. Each colon promotes the number to the left of the colon from seconds to minutes to hours to days. Times may only contain numbers and colons, white space is not allowed. One decimal point is allowed at the end of the time to introduce fractional seconds. We also allow a number with no colons to be followed by `ms` or `us` to interpret the time in milliseconds or microseconds. So the following are equivalent: `1.6`, `00:00:01.6`, `1600ms`, `1600000us`.

Operator precedence

In the table, `LHS` means the value of the expression to the left of the operator as far as the next operator of same or lower precedence, `RHS` means the value of the expression on the right up to the next operator of the same or lower precedence. Where operators have the same level, evaluation is from left to right. The order from high to low is:

Level		Name	Return value
5	`()`	Brackets	Everything inside a pair of brackets is evaluated before considering the effect of an adjacent operator.
4	`*`	Multiply	`LHS` multiplied by `RHS`
	`/`	Divide	`LHS` divided by `RHS`. It is an error for `RHS` to be zero
3	`+`	Add	`LHS` plus `RHS`
	`-`	Subtract	`LHS` minus `RHS`
2	`<`	Less than	If `LHS` less than `RHS` then 1 else 0
	`<=`	Less or equal	If `LHS` less than or equal to `RHS` then 1 else 0
	`=`	Equal	If `LHS` equal to `RHS` then 1 else 0
	`>=`	Greater or equal	If `LHS` greater than or equal to `RHS` then 1 else 0
	`>`	Greater than	If `LHS` greater than `RHS` then 1 else 0
1	`?`	Ternary operator	`LHS?A : B` has the value `A` if `LHS` is not 0, and `B` is it is 0. Put spaces around the colon to distinguish it from a time.

`1+2*3` has the value 7 because multiply has a higher precedence level than add.

Script language compatibility

The expressions are compatible with the script language except for the ternary operator, the use of `C0` to `C9` as shorthand for `Cursor(0)` to `Cursor(9)` and the use of colons, `ms` and `us` to denote times. If you attempt to use these in the script language you will get syntax errors. However, you can use these constructs in strings passed as expressions to `CursorActive()` or `MeasureChan()`.

Data view keyboard shortcuts

The following shortcut key combinations can be used in a time view, a result view or an XY view except for `Ctrl+Shift+Left` and `Ctrl+Shift+Right`, which are only available in a time view.

Key	Operation
`Left arrow`	Scroll 1 pixel left.
`Right arrow`	Scroll 1 pixel right.
`Shift+Left`	Scroll several pixels left.
`Shift+Right`	Scroll several pixels right.
`Ctrl+Left`	Scroll half a screen left.
`Ctrl+Right`	Scroll half a screen right.
`Ctrl+Shift+ Left/Right`	Time views only. If cursor 0 is active, search left/right. Otherwise, if there are selected event channels, search them for the previous/next event that is nearest to the screen centre and make it the screen centre or make a sound if there are no selected channels or no more events.
`Home/End`	Scroll to the start/end of the data.
`Ctrl+n`	Where `n` is 0 to 9. Fetch vertical cursor 0 to 9. If the cursor does not exist it is created. Cursor 0 exists only in time views.
`Ctrl+Shift+n`	Where `n` is 0 to 9. Centre the window on the cursor, if it exists.
`Ctrl+A`	Select all channels. If all channels are selected, unselect all channels.
`Ctrl+C`	Copy the image of the view to the clipboard.
`Ctrl+E`	Expand (zoom out) the data view around the left edge of the window.
`Ctrl+I`	Reduce (zoom In) the data view around the centre of the window.
`Ctrl+K`	Show the x axis dialog.
`Ctrl+N`	Open a new data file.
`Ctrl+O`	Open the file open dialog.
`Ctrl+P`	Print the current data file.
`Ctrl+Q`	Optimise selected channels. If none selected, optimise all channels.
`Ctrl+L`	Open the Evaluate window to run single line script commands.
`Ctrl+R`	Reduce (zoom in) the data view around the left edge of the window.
`Ctrl+T`	Create a TextMark during sampling (if the TextMark channel exists).
`Ctrl+U`	Expand (zoom out/Up) the data view around the centre of the window.
`Ctrl+Y`	Show the y axis dialog.
`Ctrl+Z`	Undo the last undoable operation.
`Ctrl+Break`	Break out of long drawing or calculation operations.

Text view keyboard shortcuts

Text views have more keyboard short cuts than any other area of Spike2. We have grouped them by function to make the huge list more digestible.

Text caret control

The text caret is the vertical flashing bar that indicates the current position in the text window. Do not confuse this with the I-beam mouse pointer which does not flash and which indicates the mouse position. When text is selected, the caret vanishes (except during drag and drop operations when it shows the drop site of the dragged text). The quickest way to move the text caret is with the mouse. Each time you click and release the left mouse button (we assume you haven't swapped the mouse buttons), the caret position changes to the nearest character position to the click point. To select text with the mouse, click at one end of the text you want to select and drag (move the mouse with the button held down) to the other end of the text. You can also use the keyboard to move the caret and select text:

Key	Operation (`+Shift` to extend a text selection)
`Left arrow`	Move the caret one character to the left. At the start of a line it wraps to the end of the previous line.
`Right arrow`	Move the text caret one character right. You can move it into uncharted territory beyond the end of the line. It does not wrap to the next line.
`Up arrow`	Move up one line.
`Down arrow`	Move down one line.
`Ctrl+Left` `Ctrl+Right`	Move one word to the left/right. Words are defined to be useful when operating on scripts. This will find words on the previous/next line once the current line is exhausted.
`End`	Move the caret to the right of the last character on the line.
`Home`	Move the text caret to the start of the current line.
`Ctrl+End`	Move the text caret to the right of the last character in the file.
`Ctrl+Home`	Move the text caret to the left of the first character in the file.
`F6`	Next pane (when you have split a script or output sequencer window).
`Shift+F6`	Previous pane in a split script or output sequencer window.

Cut, Copy and Paste

These operations are available from the Edit menu and also from the main toolbar. There are also keyboard short cuts:

Key	Operation
`Ctrl+A`	Select all the text in the document.
`Ctrl+C` `Ctrl+Insert`	Copy selected text to the clipboard. If no text is selected, nothing is copied. Some keyboards have `Ins` in place of `Insert`.
`Ctrl+V` `Shift+Insert`	Paste the contents of the clipboard into the text at the caret. If there is a selection, the selection is replaced.
`Ctrl+X`	Cut the selected text and copy it to the clipboard.
`Shift+Del`	Cut the selected text and copy it to the clipboard.

Delete, Undo, Redo

These operations are available in the Edit menu when a text window is active. You can also use the following keyboard short cuts:

Key	Operation
`BackSpace`	If there is a selection, delete it. If there is no selection, delete the character to the left of the text caret.
`Del`	If there is a selection, delete it. If there is no selection, delete the character to the right of the text caret.

`Ctrl+Z`
`Alt+`
`Backspace`
	Undo the last text operation. The editor supports more or less unlimited levels of Undo. You cannot undo changes made to a text window by a script.

`Shift+Ctrl+Z`	Redo the immediately previous Undo operation.

Find, Replace and Bookmarks

The Find and Replace commands can be accessed from the Edit menu, from the Edit Toolbar and by keyboard short cuts:

Key	Operation
`Ctrl+F`	Open the Edit menu Find dialog. In addition to searching for text you can also use this dialog to bookmark all matching text.
`Ctrl+H`	This shortcut key opens the Edit menu Replace dialog.
`F3`	Repeat the last find operation in the same direction. You can use the toolbar to search forwards or backwards.
`F2`	Move the text caret to the next bookmark. You can use the edit toolbar to move to the next or previous bookmark.
`Ctrl+F2`	Toggle bookmark on the current line. You can use the edit toolbar to set or clear a bookmark and to clear all bookmarks.

Bookmarks tag a line for future reference. They are displayed as a blue circle to the left of the text. Bookmarks are kept as long as the current file is open; they are lost when you close the file. The easiest way to use a bookmark is from the Edit Toolbar. You can show and hide this from the Edit menu (when a text-based window is active), or by clicking the right mouse button on any toolbar or on the Spike2 application title bar and using the pop-up context menu that appears.

Indent and Outdent

The structure of Spike2 scripts is often made clearer by indenting program structures. To make this easier, you can indent and outdent selected blocks of text to the next or previous tab stops. The tab size is set in the Edit menu Preferences option.

Key	Operation
`Tab`	If there is a selection, all lines included in the selection are indented so that the first non-white space character is at the next tab stop. If there is no selection, a tab character is inserted (or spaces to the next tab stop depending on the Edit menu Preferences settings).
`Shift+Tab`	If there is a selection, all selected lines are out-dented so that the first non-white space character on the line is at the previous tab stop. If there is no selection, the text caret moves to the previous tab stop unless it is already at one.

Drag and drop

The editor supports drag and drop of text both within Spike2 and between Spike2 and other applications that support it (for example the Spike2 Help system). Spike2 also supports drag and drop for rectangular text areas.

Operation	Method
Move block	Select the text to move. Move the mouse pointer over the selected text and hold down the left mouse button. The mouse pointer will change to indicate that you can now drag the text and the text caret will show the insertion point. Drag the text to the desired insertion point and release.
Copy block	Select the text to copy. Hold down the `Ctrl` key and move the mouse pointer over the selected text. The mouse pointer will change to show that you can drag the text. The small + symbol indicates the copy operation and the text caret will show the insertion point for the duplicate. Drag the text to the target position and release the mouse button to duplicate the text. The `Ctrl` key must be down when you release the mouse button or the operation will move the text.

Select rectangular text area

You can select, cut, paste and drag rectangular selections with Spike2. This feature is intended for use within Spike2. You can paste such text into other applications, but you will find that the first line pasted contains text that indicates the source of the rectangular selection.

To select a rectangular area hold down the `Alt` key then select text with the mouse. The point where you hold down the mouse button will be one corner of the selection, the point where you release the mouse will be the other corner.

You can use this feature to change the alignment of comments in a script, or to convert a single column of numbers into multiple columns.

The Spike2 command line

When Spike2 starts, it checks the command line for option switches and for files to load. If there is no command line, Spike2 looks in the folder that Spike2 ran from for a script called `startup.s2s` and runs it if it exists. If you run Spike2 with a non-blank command line, any `startup.s2s` script is ignored.

The command line holds options and file names. If a file name contains spaces, you must surround the file name with quotation marks. File names and options are separated by white space characters (space and tab). Options start with / or – followed by a character that identifies the option. There is currently only one option:

`/M` Allow multiple copies of Spike2. When Spike2 starts it searches for a running copy of Spike2. If it finds one and this option is not given, it activates the other copy and the command line is ignored. You must set this option if you use the script language `ProgRun()` command to launch another copy of Spike2.

The remaining items in the command line are assumed to be file names. Spike2 attempts to load the files in command line order (from left to right). The files must have extensions so that the file type is known. If a script file is included in the command line, Spike2 runs it before continuing with the remainder of the command line.

As an example, suppose we want to launch Spike2 so that it automatically opens a data file called `lots of data.smr` and runs `doit.s2s` to process it. Follow these steps:

1. Create a short cut to `sonview.exe` (this is the Spike2 program).
2. Right-click on the new short cut and select Properties and open the Shortcut tab.
3. Add `/M "lots of data.smr" doit.s2s` to the end of the Target field.
4. Set the Start in field to the folder that contains your files.
5. Click OK.

This example assumes that both files are in the same folder. You could also have included the full path to each file in the command line.

Shell extensions

The Spike2 installation adds shell extensions that display additional information in Windows Explorer when the mouse pointer hovers over Spike2 data and script files. In Windows NT2000 and XP you can also display file comments. To do this, set Details display mode, then right click in the column headers and check Comments.

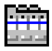
These shell extensions are automatically removed if you uninstall Spike2. To remove the shell extensions manually, open a command prompt window and type:

`C:\>`**`cd \Spike4`**	*Change to the Spike2 folder*
`C:\Spike4>`**`regsvr32 /u SonInfo`**	*For all Windows versions*
`C:\Spike4>`**`regsvr32 /u SonCols`**	*Only for NT2000 or XP*

Sampling introduction

If you worked through the *Getting started* section you already have most of the skills to sample a new data document. Sampling a new document is the same as working with an old document, except that the new document grows in length.

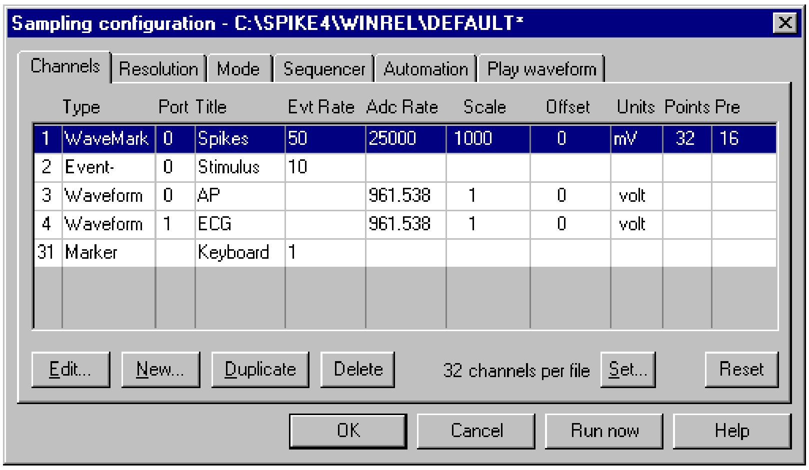
Sampling configuration



Before you start to sample data for real you must set the sampling configuration. This is done in the Sample menu Sampling Configuration dialog. This is a tabbed dialog, that is, it holds several pages activated by the tabs at the top, each page controlling a different aspect of data capture. The title bar holds the name of the configuration file from which the configuration was read. There is a * at the end of the name if the configuration has been changed. The OK button accepts the current state, Cancel rejects any changes you have made since you opened the dialog and the Run now button opens a new data document, ready to sample. You can also open this dialog from the toolbar.

Channels

The Channels tab lists the channels to sample. One channel is always selected. The left-hand column is the channel number; the remaining columns list the channel type, the physical port to sample from, the title, then data that varies with the channel type.



You edit the channel settings by double-clicking on a channel, or by selecting a channel with the mouse or keyboard and clicking the Edit button. The Reset button deletes all editable channels and sets a standard sampling state.

Number of channels and backwards compatibility

The Set… button opens a dialog in which you can set the maximum number of channels that can be stored in a data file created by the File menu New command. The standard maximum number of channels is 32, but you can create files with space for up to 100 channels. If you reduce the number of data channels and there were channels defined with numbers above the new limit, these higher numbered channels will be deleted.

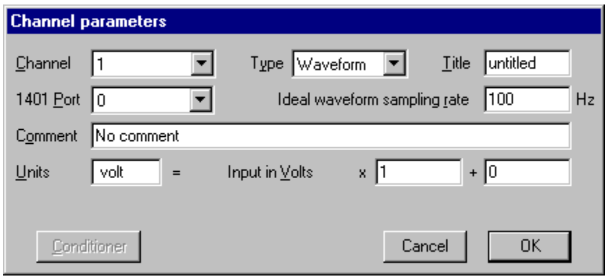
All versions of Spike2 up to 4.03 created and read Spike2 data files that could hold up to 32 data channels. From 4.03 onwards you can create and read files with up to 100 channels. However, if you create files with space for more than 32 channels you will not be able to read them with versions of Spike2 before 4.02. You can use the File menu Export command to write channels from a data file to a new file with 32 channels as the limit so that it can be read by old versions of Spike2.

You will not be allowed to sample if you request a physical port number that does not exist in your 1401.

Create a new channel

The New… button creates a new channel at the next lowest free channel number. You can then edit the new channel to the desired state. To make several similar channels, set up the first channel, then use the Duplicate button to insert a new channel at the next free channel number (and physical 1401 port) with the same settings. You remove unwanted channels with the Delete button.

When you click the Edit… button or double-click on a channel, a new dialog opens where you set the channel characteristics. The picture to the right shows the dialog for a waveform data channel; other data types have variants of this dialog. The first five fields apply to all data types.



Channel — The channel number identifies this channel in the document. Channels 30 to 32 are reserved for Text markers, Digital markers and keyboard markers. The remaining channels are for waveform, WaveMark and event data. You can change a channel number, but not to a channel that is already in use.

Type — This determines the type of the data to sample on the channel. You can set any of the following channel types:

Off	The channel is unused, equivalent to deleting the channel
Waveform	The channel holds waveform data
Event-	Event channel timed on the falling edge of the data
Event+	Event channel timed on the rising edge of the data
Level	Event channel with the times of both data edges saved
TextMark	The channel holds (short) text messages and their times.
Marker	The data is an event with either a keyboard character or a digital value attached
WaveMark	The data is a small waveform fragment, usually a spike shape

Title — Up to 8 characters to identify the channel.

1401 port — For a waveform or WaveMark channel, this is the 1401 ADC input number. For an event channel, this is the digital input port number (see page 4-4 for the connector pin numbers). Marker channels do not use this field.

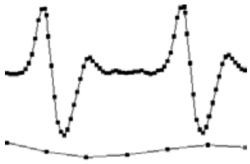
Comment — Some text to give more information about this channel.

The Conditioner… button is present for Waveform and WaveMark data types and opens the signal conditioner dialog if a conditioner exists for the channel (see the *Programmable signal conditioners* chapter for more information).

Maximum sampling rates

With a micro1401 or a 1401*plus*, the maximum continuous waveform and WaveMark throughput (sum of the rates of all channels) is 166 kHz. Power1401 and a Micro1401 mk II users can set any rate up to the maximum allowed by their 1401. If you use Event or marker channels or the Play waveform feature at high data rates or you may have to reduce the overall sampling rate if you experience data overrun errors. You may find that display updates become slower as the overall sampling rate increases. Of course, the rate you can achieve also depends on your computer and interface card. The ISA interface card is the slowest and will not achieve the maximum rates possible; upgrading to a PCI or USB interface will show significant throughput improvements.

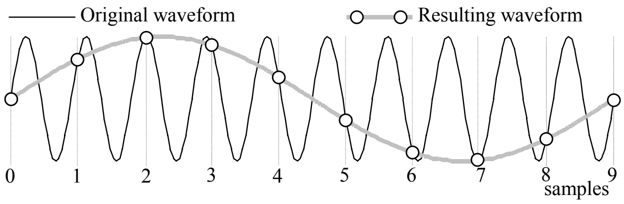
Waveform channels



The waveforms you record are continuously changing voltages. Spike2 stores waveforms as a list of numbers that represent the waveform amplitude at equally spaced time intervals. The process of converting a waveform into a number at a particular time is called *sampling*. The time between two samples is the *sample interval* and the reciprocal of this is the *sample rate*, which is the number of samples per second. The dots in the diagram to the left of this text represent samples; the lines show the original waveform.

Minimum sample rate

The waveform sample rate must be high enough to represent the data. The sample rate must be at least double the highest frequency contained in the data. If you do not sample fast enough, high frequency signals are aliased to lower frequencies, as illustrated above where the ten samples (0 to 9) of a high frequency signal have the appearance of a low frequency signal. On the other hand, you want to sample at the lowest frequency possible, otherwise your disk will soon fill. Unlike many data capture systems, Spike2 lets you capture different waveform channels at different rates to minimise data file sizes.



Use of filters

Many users pass waveform data through amplifiers or signal conditioners with filter options such as the CED 1902 to limit the frequency range. Some transducers have a limited frequency response and require no filtering.

Input connections

Channel	8	9	10	11	12	13	14	15	Gnd
1401*plus* pin	1	2	3	4	5	6	7	8	9-15
Power1401 pin	28	29	30	31	32	33	34	35	1-19

Connect your waveform channels to the 1401 ADC inputs. Channels 0-7 (0-3 for a micro1401) are the labelled BNC connectors. Channels 8-15 are on the 1401*plus* 15-way front panel Cannon connector and on the Power1401 rear panel 37-way Cannon connector. Pin numbers are given in the table. If you have an ADC expansion fitted you will have more channels; see the accompanying documentation for the connections. Power1401 top boxes can reassign the rear panel channels. The standard input Voltage range is ±5 Volts. If you have a ±10 Volt system check that the Voltage range is set correctly in the Edit menu Preferences.

Waveform dialog

The Waveform channel dialog has all the standard fields, plus:

Ideal rate Set the Ideal waveform sampling rate field to the desired sampling rate for this channel. The actual sample rate will be as close to this ideal rate as possible. You can see the actual rate in the Sampling Configuration dialog (in red if it is more than 20% different from the desired rate). If the rates differ too much, adjust the optimisation and clock settings in the Resolution tab.

Units This field holds the waveform units, up to 5 characters long. The following fields are the *scale* and *offset* to convert the input from Volts into user units.

input in user units = input in Volts * *scale* + *offset*

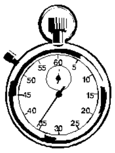
The *scale* is the number of units for every one Volt increase in input; *offset* is the value represented by 0 Volts at the 1401 input.

As an example, consider a situation where a waveform represents a position. 1 Volt is equivalent to 10 mm and 3 Volts is equivalent to 50 mm. In this case you would set:

scale = (50 - 10) / (3 - 1) = 20.0 mm V^{-1}
offset = 10 - (1 Volt) * *scale* = -10.0 mm
Units = `mm`

For the scaling to work as expected, the Edit menu Preferences option for Voltage range must be set correctly for your 1401. To display in metres in place of mm, set *scale* to `0.02`, *offset* to `-0.01` and *units* to `m`. The easiest way to set the calibration from known input data is to use the Analysis menu Calibrate command.

Event data



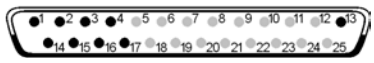
Spike2 stores time stamps very efficiently as integer multiples of the time resolution set in the Resolution tab of the sampling configuration. Simple time stamps with no other attached data are called *events*. Each event uses 4 bytes of storage. The 1401 recognises events as changes of state of TTL compatible signals connected to the 1401 digital input bits 15-8. We refer to these inputs as *event ports* 7-0. There are three types of event:

Event- Spike2 saves the time of the falling edge of the input signal. The minimum input pulse width is 1 µs; wider is better.

Event+ The same as Event-, but Spike2 saves the time of the rising edge.

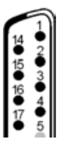
Level Spike2 saves the time of both edges. Pulses should be a minimum of 50 µs wide or the time resolution set for sampling, whichever is the larger. Do not use this type unless you need the times of both edges.

Micro1401 and Power1401 event port connections



Event ports 0 and 1 are BNC sockets on the front panel. If you have the optional Spike2 top box, event ports 7-2 are also on BNC sockets, otherwise you must use digital input bits 15-10. The digital input connector is on the rear panel. If you want to use the rear panel digital input connector for event ports 0 and 1 to be pin compatible with the 1401*plus* there is an option in the Edit menu Preferences… to use the rear panel connector for all events.

1401plus event port connections



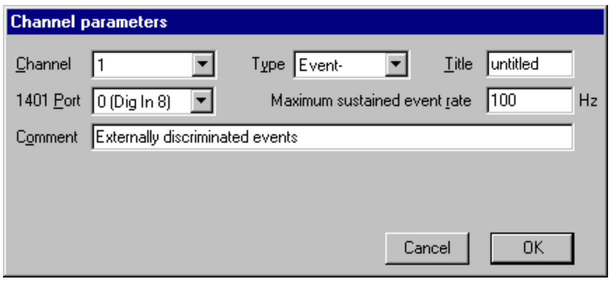
The event ports are bits 15-8 of the front panel digital input connector. Do not use the BNC sockets labelled Events 0 to 4; these have different functions: Event 1 is the digital marker input, 3 can be used to start sampling and also as a trigger for arbitrary waveforms. Events 0, 2 and 4 are used internally by Spike2 and have no external function.

Input connections (digital input)

The pin connections for the digital input connector are the same for all 1401s:

Digital input bit	15	14	13	12	11	10	9	8	Gnd
Digital input pin	1	14	2	15	3	16	4	17	13
Event port	7	6	5	4	3	2	1	0	

Event dialog



The event channel dialog is similar to the waveform dialog. There is no Units field and there is a new field for the Maximum sustained event rate. This is your estimate of the maximum mean event rate sustained over a few seconds. This is not the maximum instantaneous rate, which may be much higher. Spike2 uses this information to optimise buffer space allocation.

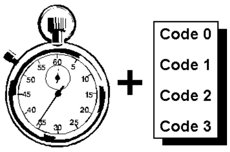
As an example, an event channel might have a mean rate of 30 events per second, but it could have an instantaneous maximum rate of 1 kHz if two events fell within 1 millisecond of each other. In this case, the rate you should enter is 30 Hz, not 1000 Hz.

TTL compatible signals

TTL stands for Transistor-Transistor Logic, a method for passing logical information between devices using voltage levels. Levels above 3.0 Volts are in the High state, levels below 0.8 Volts are in the Low state. Levels in between 0.8 and 3.0 Volts are undefined.

Do not subject 1401 TTL inputs to voltages above 5.0 Volts or less than 0.0 Volts. CED hardware has special circuits on TTL compatible inputs to provide some protection, however determined abuse will damage them. The 1401 TTL compatible inputs are pulled up by a resistor to 5 Volts. They require a current of no more than 0.8 mA to pull them into the Low TTL state. Alternatively, you can connect them to ground to pull them low (this can be useful for the Event 3 input). See the *Owners handbook* of your interface for full details of all input ports.

Marker data



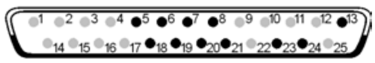
Spike2 samples keyboard markers on channel 31 and digital markers on channel 32. A Marker is a 32 bit time plus 4 bytes of marker information. The first of these 4 bytes is the ASCII code of the keyboard character pressed by the user (channel 31) or an 8 bit digital code read by the 1401 (channel 32). The remaining three bytes are set to zero.

Spike2 treats all marker types identically once the data has been captured; they differ only in their source. You can also treat data types that are derived from markers (such as WaveMark, TextMark and RealMark) as if they were markers.

Keyboard markers

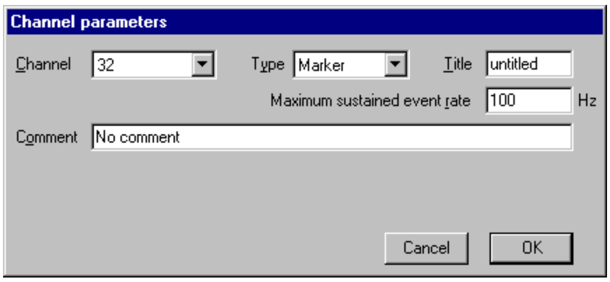
The timing of Keyboard markers is not precise; it depends on the load on the computer. Use the event inputs for exact timing. Any keyboard character that is not trapped for a special purpose (for example `Ctrl+L` opens the Evaluate windows) is recorded, but *only when the sampling document window is the current window*. In the special case where keyboard markers trigger data sampling (see page 4-12), the precise time of the trigger is stored. You can link keyboard markers to the output sequencer (see the *Data output during sampling* chapter). The keyboard marker channel is always enabled.

Digital markers



The digital markers are timed as accurately as the event data. They record 8 separate channels of on/off information, or one channel of 8 bit numbers, or any combination in between. Digital marker data is sampled when a low going TTL compatible pulse is detected as described below. The data is read from bits 7-0 of the 1401 digital input.

Digital marker dialog



The Maximum sustained event rate field is used to allocate the system resources for data capture on this channel. Set a reasonable estimate of the maximum sustained data rate over several seconds. Do not set the peak rate or you will waste resources.

Digital marker connections

Marker data bit	7	6	5	4	3	2	1	0	Gnd
Digital input pin	5	18	6	19	7	20	8	21	13

The digital marker is read from data bits 7-0 of the digital input connector. The 1401*plus* needs a pulse on the E1 front panel input to flag a digital marker. The micro1401 and Power1401 require a pulse on digital input pin 23. In addition to the data lines, there is an optional handshake (h/s) signal.

Signal	*plus*	micro & Power
Event flag	E1	pin 23
h/s	pin 23	pin 24
h/s active	TTL high	TTL low

To flag an event, apply a low going TTL pulse at least 1 µs wide to the Event flag input. When the 1401 detects a falling edge at the Event flag input, it sets the h/s line active (within a few microseconds). The falling edge of the Event flag input latches the input data in the micro1401 and the Power1401. The h/s returns to a non-active state after the 1401 reads the input. If you use a 1401*plus* you must keep the digital input data signals stable until the h/s line returns to the non-active state (this should never be more than 50 µs after the Event flag input goes low).

1401plus with the 1401-18 event discriminator

If your 1401*plus* has a 1401-18 event discriminator card (see the *1401-18 Programmable Discriminator* chapter), the 1401-18 uses all the pins on the digital input connector. The 1401*plus* duplicates the digital marker data bits on the same pin numbers on the digital output connector. However, pin 23 is not duplicated, so the h/s output for the digital marker is not available if you also have a discriminator card.

Output sequencer link

There are links between the digital marker channel and the output sequencer. If the `REPORT` instruction is used in an output sequence, this simulates a digital marker input pulse and causes the digital input to be read and the time to be recorded. As this is an internal activity, the handshaking described above is not available.

You can also use the `MARK` output sequence instruction to record a digital marker without reading the digital inputs (the instruction sets the 8-bit marker code). This instruction is often used to record output sequencer actions as part of the data file.

You can mix externally and internally generated digital markers, but this is not recommended unless care is taken to differentiate between the two sources of markers during analysis. This could be done by connecting the external marker handshake line to one of the digital marker data bits so that all external markers were flagged.

Warning: The `DIBEQ`, `DIBNE`, `DIGIN` and `WAIT` sequencer instructions use the same inputs as the digital marker and can cause digital marker events to be missed. Spike2 will warn you if this is an issue with your hardware.

Marker codes

When Spike2 displays markers, or data derived from markers, such as WaveMark or TextMark data, it shows the code of the first of the four markers. Marker codes occur in several other guises, for example to set trigger codes, as arbitrary waveform output codes and in the spike shape module.

Marker codes have values from 0 to 255. This is the same range of numbers that the 8-bit ASCII character set uses, and it is sometimes convenient to treat the codes as ASCII character codes (for instance when dealing with keyboard markers). At other times it is more convenient to deal with the codes as numbers.

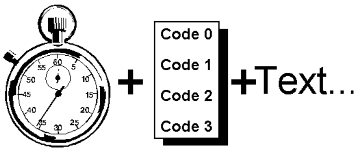
Whenever Spike2 displays a marker code that is the same as the ASCII code of a printing character, it shows the printing character, otherwise it displays the character as a two digit hexadecimal code. Hexadecimal (base 16) numbers use the standard digits 0 to 9, but also use a to f (for decimal 10 to 15). Thus 00 to 09 hexadecimal is equivalent to 0 to 9 decimal. 0a to 0f is equivalent to 10 to 15 decimal. 10 to 1f hexadecimal is 16 to 31 decimal, 20 to 2f is 32 to 47 decimal and so on.

+	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
20		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

The printing characters are 20 to 7e hexadecimal, 32 to 126 decimal, as in the table. To find the hexadecimal code of a printing character, add the number above it to the number to the left of it. For example, the code for A is 41. To convert a code to a character, look up the first digit in the left column and the second in the top row. For example, `3f` codes to `?`, the intersection of the row for `30` and the column for `f`.

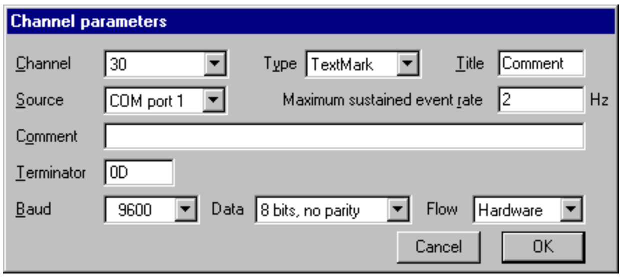
When typing marker codes (for example in the on-line Process dialog in Triggered mode, or when assigning codes in the spike shape module), type two hexadecimal digits for a code or type a single character to stand for itself.

TextMark data



This type is a combination of a marker and a text string. It is stored as a 32-bit time and 4 bytes of marker information followed by a text string that can be up to 80 characters long. This type allows you to insert timed comments into a data file. In the special case where the TextMark channel triggers data sampling, the precise trigger time is stored. From a script you can set longer or shorter strings with the `SampleTextMark()` command.

The channel number is forced to 30. If you select a channel number of 30, the data type is forced to TextMark. The Maximum sustained event rate is not used; please set it to a low number as it may be used in the future to optimise disk buffer sizes.



Serial line input

The Source field can be set to Manual or a choice of serial ports. If you select a serial port more fields can be set. The Terminator field sets the input character that marks the end of a text line. This field uses the same coding as for marker codes, so a single character stands for itself; two characters are interpreted as a hexadecimal code. Common codes are `0D` for carriage return (the default) and `0A` for line feed; `00` cannot be used. If no terminator is set, `0D` is used.

You can also set the standard serial line parameters for Baud rate, data bits and parity and handshaking. These must match the data source for reliable operation. See your computer hardware manual for pin connections and Baud rate limits.

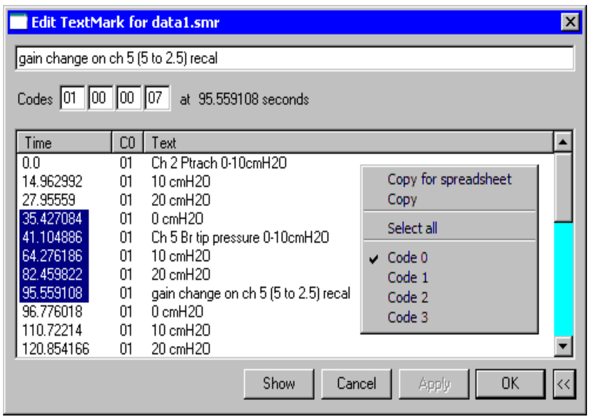
When reading the serial line, characters codes below 32 are ignored unless they match the terminator character. The marker time is set when the first character arrives except for the special case where the TextMark channel triggers data sampling, when the precise trigger time is stored. The serial data can set the first marker code by ending the text with a vertical bar followed by the marker code as a decimal or hexadecimal number. The vertical bar and the following text are excluded from the recorded data. If `<term>` stands for the terminating character, all the following are acceptable inputs:

```
Message from serial input code will be 00<term>
Message setting code 0e (decimal 14)|14<term>
Message setting code 14 as hexadecimal|0xe<term>
```

Manual input

TextMark data is added to your file during sampling from a serial line and with the Sample menu Create a TextMark... command. You can also activate the dialog with the `Ctrl+T` key combination as long as the sampling data file is the active window.

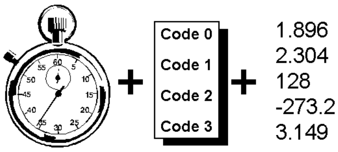
TextMark data is drawn as small rectangles. The rectangles are yellow unless the first marker code is non-zero, in which case the same colour coding as for WaveMark data is used.



Move the mouse pointer over a marker to see the attached text. Double click to view and edit the text and codes and display a list of the markers in the file.

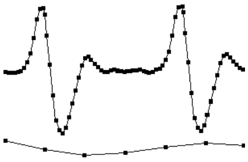
If the TextMark channel is used, programmable signal conditioner changes are saved automatically as TextMark items.

RealMark data



This is a data type that is supported by Spike2, but one that you cannot create interactively (except as a memory channel). It stores a 32 bit time, 4 bytes of marker information, then a user-defined number of single precision floating point numbers. You can create and manipulate this data type from the script language.

RealWave data

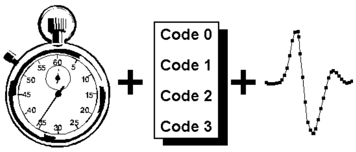


This data type was added at version 4.03. It is identical to waveform data except that the data is stored as 32-bit IEEE floating-point data, not as 16-bit integers. The channel has a scale and offset. These are used to convert between waveform data and RealWave data:

RealWave data = integer data * scale / 6553.6 + offset

You cannot sample RealWave data; you can create it as memory channels and manipulate it from the script language. We expect to exploit RealWave data in future Spike2 releases. Versions of Spike2 prior to 4.02 cannot open data files holding this data type.

WaveMark data

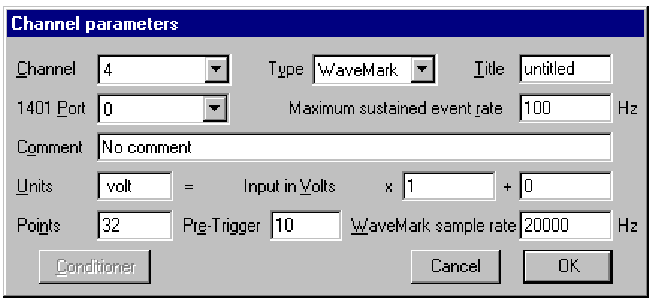


This type combines waveform and marker data. It is stored as a 32 bit time and 4 marker bytes, followed by up to 126 waveform points. The waveform holds a spike shape, and the first marker byte holds the classification code of the Spike, or 0 if it is unclassified. Script users can create WaveMark data for use off-line with up to 1000 data points and up to 4 interleaved traces. You cannot sample multiple traces in Spike2 version 4.

Use WaveMark data where a high waveform sampling rate is needed to characterise very short events, for example nerve spikes. The 1401 searches the incoming waveform for sections that might include a spike. When the waveform crosses a trigger level, the signal is tracked to the next peak (or trough), and a data around the peak is saved.

The maximum number of WaveMark channels you can sample depends on the 1401 type: 32 for Power1401, 16 for Micro1401 mk II, 8 for 1401*plus* and micro1401, 1 for 1401*plus* with the old non-silo ADC board.

WaveMark dialog



The start of the dialog is a combination of the event and waveform dialogs. The Maximum sustained event rate is not the waveform sampling rate; it is the expected mean rate of trigger events (these are usually spikes) per second.

The Units field is the same as for a waveform channel. You can use the Analysis menu Calibrate command to calibrate known values. There are new fields to set the number of data and pre-trigger points per event; these values can be adjusted dynamically during template formation. You can also set the desired waveform sample rate.

Points The number of waveform points to store for each WaveMark captured on this channel. You should set this to the smallest value you can (the larger the value, the more space is used on disk, and the slower it is to process).

Pre-trigger This is the number of data points to keep before the first peak or trough to exceed the trigger level for this channel.

sample rate This field sets the ideal sample rate for all WaveMark channels. Spike2 will adjust the sampling parameters to get as close to this rate as it can. See the description of the Resolution Tab for more information about rates.

The Conditioner button is enabled if there is a programmable signal conditioner present .

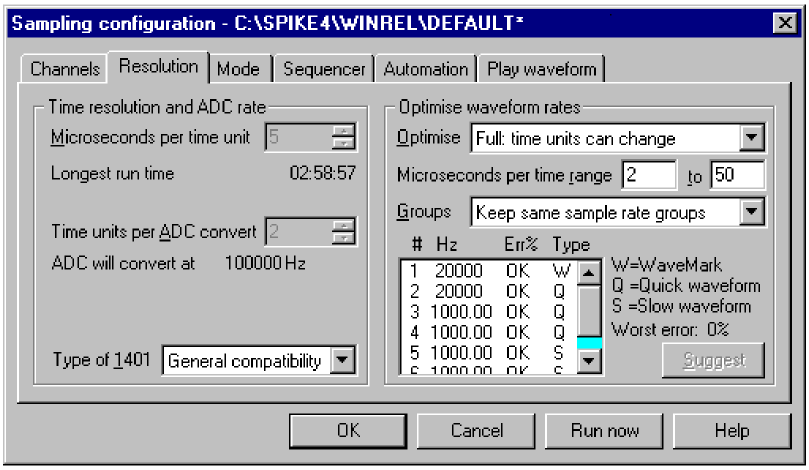
Spike sorting

Spike2 can match waveforms to a set of pre-determined shapes. This is normally used to extract single spike units from multi-unit recordings, but other uses are also possible (for example extracting R waves from ECG waveforms). If one of your channels is WaveMark data, Spike2 offers you a template setup window when you open a new file for sampling (see the *Spike shapes* chapter for a description of Template setup).

Resolution

The Resolution page of the sampling configuration dialog sets the time resolution of the document and the ADC sampling rate. To get the best possible results, you should read all this section. However, to get started quickly, follow these "cookbook" instructions:

1. Set the Type of 1401 field as appropriate (set General compatibility if unsure).
2. Set Optimise to Partial: fixed time units.
3. Set Groups to Keep same sample rate groups.
4. Set Microseconds per time unit so that the Longest run time field is at least as long as the time you want to sample to the data file.



This dialog controls how Spike2 optimises the sampling rates. Spike2 minimises the sum of the proportional errors between the desired and the actual rates for the waveform and WaveMark channels. By proportional we mean that an error of 200 Hz in a sampling rate of 10 kHz is the same as an error of 2 Hz in a rate of 100 Hz.

Microseconds per time unit

This field sets the time units for a new data file in the range 2 to 1000 microseconds. All data of any type stored in the file occurs at a multiple of this time unit. The maximum time stored in a file is 2,147,483,647 units. The Longest run time is the maximum file length in hours, minutes and seconds for the current time units. To edit this field, set Optimise to None or Partial. If you select Power1401 or Micro1401 mk II in the Type of 1401 field, you can set this field to a resolution of 0.1 instead of 1 microsecond. If you do this, the resulting file cannot be read by versions of Spike2 prior to 4.02.

Time units per ADC convert

This sets the number of time units set by Microseconds per time unit between each ADC (Analogue to Digital Converter) sample. To edit this field set Optimise to None. The ADC will convert at field is the equivalent rate in Hz. This is the maximum rate for a waveform stored in the data file, even for waveforms created off-line by analysing data.

Type of 1401

Members of the 1401 family have different capabilities. The choice you make here sets the absolute maximum settings for sampling. There is no guarantee that the maximum settings are achievable. The maximum rate depends on the entire sampling configuration, the input data load, the speed of the 1401, the speed of the interface connection (PCI, USB, ISA) and on the capabilities of the host computer. You can set:

General compatibility Any 1401 except a 1401*plus* with the old analogue card (an upgraded standard 1401). This is a *lowest common denominator* setting with a maximum waveform sampling rate of 166 kHz and does not assume that your monitor firmware is the most recent.

1401plus, old ADC If your 1401*plus* was upgraded from a standard 1401 without an upgrade of the analogue card you must select this option.

Power1401 Select this option to take advantage of the advanced capabilities of the Power1401. This allows Microseconds per time unit to be set in units of 0.1, Time units per ADC convert to be set to 1 and sets the maximum ADC convert rate to 400 kHz.

micro1401/1401plus This setting is for a micro1401 or a 1401*plus* with an up-to-date monitor. You can find if there is more recent firmware available in the Help menu About Spike2 command.

Micro1401 mk II Select this option to take advantage of all the capabilities of the Micro1401 mk II. This allows Microseconds per time unit to be set in units of 0.1, Time units per ADC convert to be set to 1 and sets the maximum ADC convert rate to 500 kHz.

Optimise The Optimise field sets the parameters Spike2 changes to minimise sample-rate error:

None: use manual settings You have control over the values in the Microseconds per time unit and Time units per ADC convert fields. In this mode click the Suggest button to change the fields to the values that minimise sample rate errors.

Partial: fixed time units You set the Microseconds per time unit field and Spike2 adjusts the Time units per ADC convert field to minimise the sampling rate errors. If there is more than one solution, Spike2 chooses the one with the slowest ADC convert rate.

Full: time units can change Spike2 sets the Microseconds per time unit and Time units per ADC convert fields. If all channels have a very slow rate, this can take a noticeable time. The Microseconds per time range fields set the acceptable range of time units. If there are multiple solutions, Spike2 chooses the one with the biggest Microseconds per time unit.

Groups This field controls how Spike2 divides the 1401 ADC conversions between WaveMark and waveform channels.

Version 3 compatible This gives the same waveform rates for a given Microseconds per time unit and Time units per ADC convert as in previous versions. Only use this if compatibility with version 3 is very important to you, for example if you change version halfway through a project and it is vital that sampling rates match old data files.

Keep same sample rate groups If you select this option, Spike2 will make sure that all waveforms channels with the same ideal sampling rate have the same actual rate. You will normally use this option.

Ignore same sample rate groups This option gives the smallest sampling rate errors. The price you pay is that channels with the same ideal sampling rates may get actual sampling rates that are different. Some data analyses, such as waveform correlations or multiple channel averages and power spectra demand that channels have identical sampling rates.

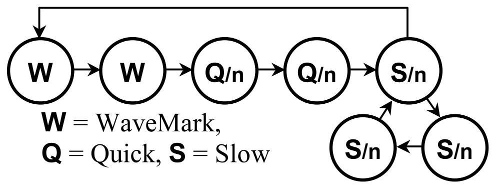
1 MHz, same sample rate groups This is the same as the *Keep same sample rate groups* option, but it also forces the Microseconds per time unit field to be an integral number of microseconds.

Technical details A master clock in the 1401 controls all sampling. The Microseconds per time unit field sets the tick period of this clock. The maximum sample time is 2,147,483,647 (or 2^{31}-1) clock ticks. At 2 µs per tick this is 71½ minutes, at 10 this is almost 6 hours, and at 1000 this is nearly 25 days. All times in a Spike2 data file are multiples of this time unit.

The ADC in the 1401 is shared between the waveform and WaveMark input channels. The Time units per ADC convert field determines how often the ADC samples. For example, with 10 µs per tick and the ADC converting every 2 units (once every 20 µs), this is a sample rate of 50 kHz. The maximum rate depends on the type of 1401.

To generate the sample rates, the ADC samples a cycle of channels. Some waveform channels are set as Quick and are sampled every time round the cycle. The other waveform channels are set as Slow, and these share one position in the cycle. Quick and Slow channels save every nth data point (n in the range 1 to 65535). The Version 3 compatibility setting in Groups sets all waveforms as Slow channels.

The diagram shows a possible cycle for two WaveMark and five waveform channels. Spike2 samples all WaveMark and Quick channels and one Slow channel each time around the main loop. When waveform and WaveMark channels are sampled together, the fastest waveform rate is the same as the WaveMark sample rate.

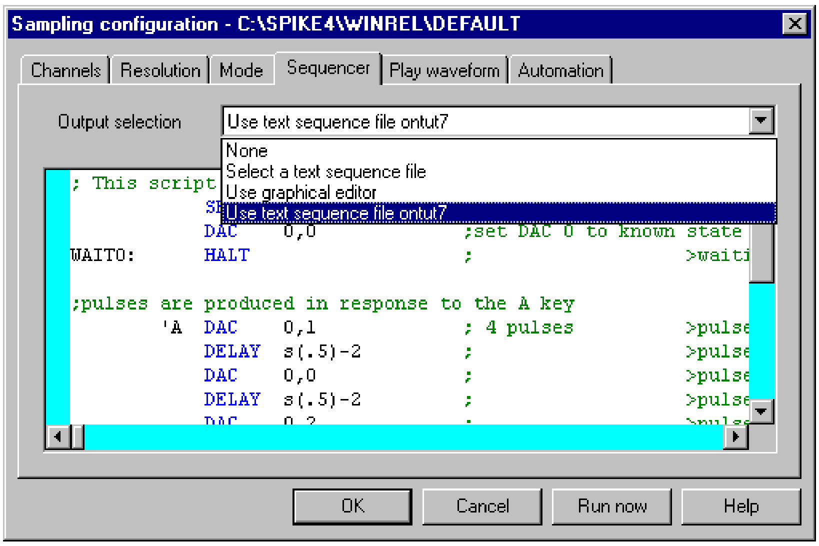


Spike2 searches all ADC rates allowed by the Optimise setting and the Type of 1401 field for the best combination of Quick and Slow and the best value of n for each channel to get as close as possible to the ideal sample rates. With slow waveform rates there can be millions of combinations to search. If the Channels Tab feels very sluggish, set Optimise to None, set the channels, then restore the Optimise value. Impossible combinations display a warning in the lower left corner of the dialog.

The table to the left of the Suggest button lists the channels in order of descending sample rate error, showing the actual sample rate, the error as a percentage of the desired rate or OK if there is no error, and the channel type (WaveMark, Quick or Slow).

Sequencer

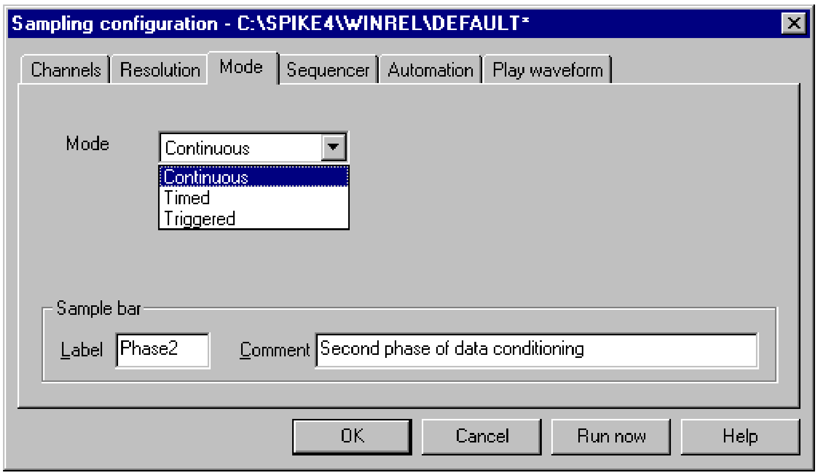
The Sequencer page of the Sampling configuration dialog sets the output sequence to use during sampling. The drop down list enables and disables sequence output and selects an output sequence file or sequences generated by the graphical sequence editor. You can select None, the current file name, Select new file or use the output sequence configuration dialog. If you select a new file, a file selection dialog opens and you can choose the output sequence file (`*.pls`) to attach to this sampling configuration. These files are created with the output sequence text editor or by exporting sequences generated by the graphical editor (see the *Data output during sampling* chapter).



If a file is selected it is displayed in the lower portion of the dialog. To modify an output sequence file you must open it from the File menu. If you select the graphical sequence editor, the lower portion of the dialog displays general sequence control settings.

Sampling mode

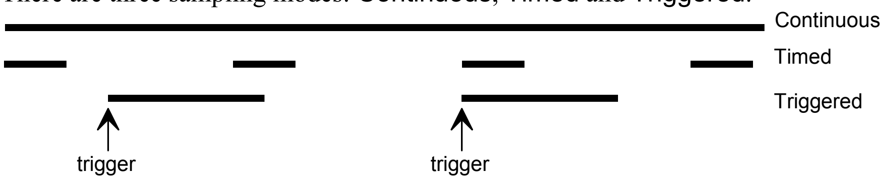
The Mode page of the sampling configuration dialog determines when data is captured by Spike2 and saved to the disk system. Whichever sampling mode you select, there is also a manual control during sampling that lets you disable data saving to the data document. Script users can enable and disable data saving channel by channel. In all modes, time passes at a constant rate, even when nothing is written to disk. When reviewed, areas of a file with no saved data are empty.



This page also holds a label of up to 8 characters and a comment of up to 80 characters for the Sample bar. When configurations have been saved to a `.s2c` file, they can be added to the Sample bar by the Sample menu Sample Bar List… command and any label or comment is used to provide information about the configuration.

Once a configuration is in the Sample bar, you can open a new data file ready to sample with a click of the mouse. See the *Sample menu* chapter for more information about the Sample bar.

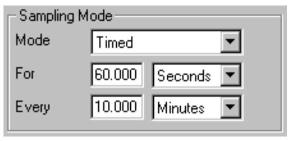
There are three sampling modes: Continuous, Timed and Triggered.



Continuous mode

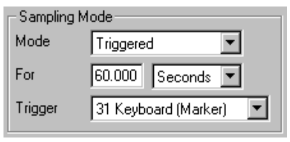
The simplest sampling mode is Continuous mode which records data continuously.

Timed mode



The second mode is Timed data capture. In this mode, data is saved to the document at intervals. You set the periods for which data is saved and for which data is not saved. Spike2 captures data on each channel in blocks. The start and end of a block does not, in general, fall at the same times as the start and end of a sampling period. Spike2 saves *at least* the data you request, however you may get data before and data after the requested period. The times at which the blocks were requested are saved in the keyboard marker channel. Marker code 00 is placed at the start of each timed block and marker code 01 is placed at the end of each block.

Triggered mode



The third mode is Triggered capture. Data is not captured (except on Level event channels), until a trigger is detected. Triggers can be from an event channel, a digital marker channel or the keyboard. WaveMark channels cannot act as a trigger. For keyboard triggers, the time saved is exactly the trigger time (normal keyboard markers do not have such precise timing). You set the duration of data capture after each trigger and the trigger channel. Sampling is always at least as long as you request.

Special keyboard trigger features

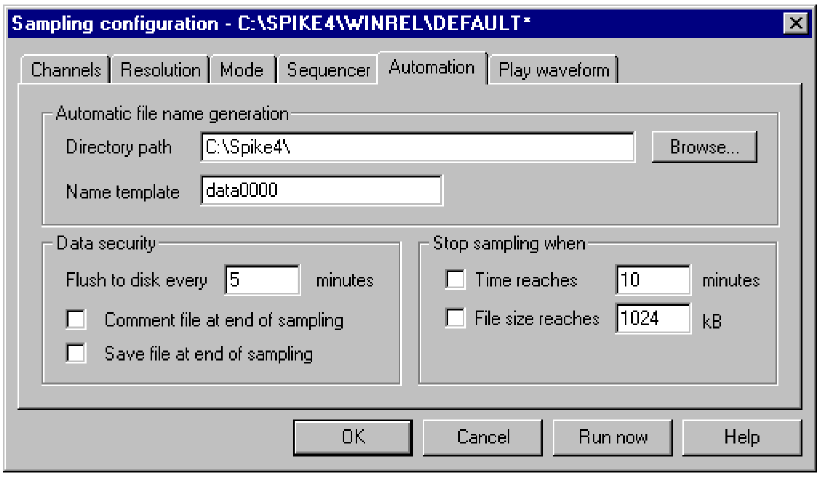
A keyboard trigger within a triggered sweep extends the sample period by the sweep time from the current time. For other trigger sources, extra triggers within a sweep are ignored. Keyboard channel marker codes 00 and 01 indicate that writing to disk is enabled or disabled. These two codes do not trigger sampling.

Display during data capture

In both Continuous and Timed capture, the on-line display shows newly sampled data, even when you are not saving to disk. Recent 1401 data is saved in a memory buffer in these modes so the current data is always available. However, if you scroll far enough back into an unsaved time region, the display may become blank. In Triggered mode, the 1401 does not capture data at all outside the triggered region (except for Level mode event data), so there is nothing to see outside the triggered regions.

Automation

The Automation page of the sampling configuration dialog sets the file path and name for automatic data filing, sets the security parameters for new data files and restricts the total sampling time and the size of the data file.



Automatic file name generation

Spike2 samples data to temporary files in the folder set by the Edit menu Preferences. These files have names like `Data1`, `Data2` with no `.smr` extension. When sampling ends, you save the file to a name of your choice. You can automate file naming by setting a Name template (up to 22 characters), and Spike2 will generate a sequence of file names from it. If the Name template is blank, automatic name generation is disabled.

If the template does not end in a number, `000` is added before using it. To make a file name, Spike2 increments the number until a name is formed that is not in use in the Directory path folder (or the current folder if Directory path is blank). A template of `test` generates `test000` to `test999`. A template of `test10` generates `test10` to `test99`. The Directory path must be less than 200 characters long.

With a file name template set, the generated name is used when the data file is saved. A different name can be specified using the File Save As… command. If there are no free names, or the path set for file saving does not exist, you are prompted for a file name.

Data security

For efficient sampling, Spike2 buffers several megabytes of the most recently sampled data in memory and writes data blocks to disk when the buffers are full. However if the power failed, this data would be lost. The Flush to disk every n minutes field sets how often the data buffered in memory is written to the physical disk to guarantee that your data is safe. There is a time penalty for doing this, so if you want the fastest possible sample rate you should turn this feature off (by setting a zero period). However, with it on, even if the computer power is lost or the computer crashes, your data should be safe up to at least the last flush time. You must run the `SonFix` program on such a data file to complete the data recovery and to tidy up data blocks written after the last flush.

If the Comment file at end of sampling box is checked, you will be prompted to provide a file comment when sampling finishes.

If the Save file at end of sampling box is checked, the new data file is saved to disk automatically when sampling finishes. If automatic filename generation is in use, the generated filename is used, otherwise the usual prompts for a file name are provided.

Stop sampling when

You can cause sampling to stop automatically at a set run time, or when the data file is a set size. If you do not check a box, the associated limit is not used. In general, you will get a little more data than implied by an active time or data limit as data buffered in the 1401 at the time the limit was reached will be added to the file.

Arbitrary waveform output

You can replay arbitrary waveforms during data capture. Up to ten different "Play wave" areas can be defined for output. Each area is identified by a key code, typically a printing character such as "A". No two areas may have the same code. You do not have to use a printing character, you can use a two digit hexadecimal code if you prefer, however code 00 is not allowed. The format of this code is the same as for marker codes (see page 4-6).

The waveforms are played from 1401 memory and are copied there just before sampling starts. This reduces the memory available for recording data. The maximum data you can store in the 1401 for replay is the free space in the 1401 less 256 kB which Spike2 reserves for recording. A micro1401 or a 1401*plus* with 1 MB of memory can store around 750 kB of waveform. A 1401*plus* with expanded memory can store nearly 16 MB, a fully expanded Power1401 can store almost 256 MB. Script users can update the memory dynamically during replay, so huge memories are not necessarily required!

1401	*plus*	micro	Power
kHz	62.5	100	250

The table shows the maximum rate we measured replaying two waveforms while sampling one. The input and output rates were the same. If you sample or replay more channels or use the output sequencer, the maximum rate will be lower. The rates were measured on a 450 MHz Pentium with a PCI interface card.

Play waveform toolbar

There is a dockable toolbar associated with the waveform output. This is enabled when you are sampling data with waveforms defined. The toolbar can be docked on any edge of the application and can be resized when it is floating. You can assign your own labels to the buttons, or you can let Spike2 generate labels itself of the form Wave 0, Wave 1 and so on. The first button in the toolbar is used to stop a currently playing waveform.

Channels and DACs

Each play wave area contains from 1 to 4 data channels. You can select the 1401 DAC (Digital to Analogue Converter) each channel plays through. With multiple channels, all channels play at the same rate and all DACs update together. You can play data from waveform and WaveMark channels in a Spike2 data file, or data generated by a script. When data comes from a Spike2 file, the channels need not all have the same sample rate; Spike2 takes the rate of the first channel and interpolates data from the subsequent channels to make the rates the same. Script users can play arbitrarily long data by updating the waveforms in the 1401 online with the `PlayWaveCopy()` command.

The rate at which a wave plays can be modified in the range 4 times slower to 4 times faster than normal (as long as your hardware can output fast enough). From a script, you can change the rate during sampling, even while the wave is playing.

Playing waves

There are several ways to initiate output of a wave during data sampling:

- Click the associated button in the Play waveform toolbar
- Record the associated key code in the Keyboard marker channel
- Use the script language `SampleKey()` command to record the key code
- Use the output sequencer `WAVEGO` command

Unless you use the output sequencer to start the output, the associated key is recorded in the Keyboard channel and marks the time at which output was requested.

When a wave starts to play, there is a time delay of one waveform output between the moment that output is requested and the first data point appearing. This is not usually important, but at a slow replay rate, a one sample delay could be significant. This delay can be useful if you are using the output sequencer as it gives the output sequencer the opportunity to know about a change in the DAC outputs *before* it happens.

Triggered output

Each wave you play can be marked as "Triggered". In triggered mode, when the waveform play is requested, output does not start immediately. Instead, the 1401 hardware waits for a trigger input (high to low edge) on the 1401*plus* E3 input or the Trigger input for the micro1401 and Power1401 unless this is routed to the rear panel Event connector pin 4 (Ground is pins 9-15) by the Edit menu Preferences. In triggered mode, the first data point is output at the time of the trigger.

If you use the output sequencer you choose between triggered and non-triggered in the `WAVEGO` instruction. You can also trigger the wave from the sequencer with the `WAVEST` instruction and detect if the wave has started to play with the `WAVEBR` instruction.

Repeated plays and links

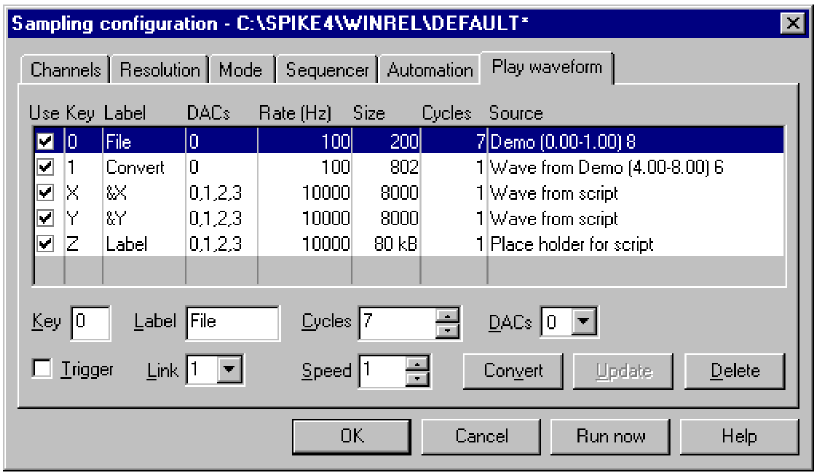
Each wave can be set to play cyclically for a set number of times. You can also link waves together if they have the same DAC output list. Linked waves play at the rate of the first wave, that is the sample rate of subsequent waves is ignored.

For example, you might have a sound output that needs to ramp up, stay at a constant level, then ramp down. This could be done with three waves. The first holds the ramp up waveform, the second which repeats cyclically many times would hold the constant sound, and the final part would ramp down.

Scripts and the output sequencer can command a wave with many cycles to finish the current cycle then continue to the next linked area. You could use this to simulate a blood pressure signal with an occasional errant beat by having two waves, one with a normal beat set to repeat many times and one with the errant beat set to play once. By linking the two areas to each other, you get one errant beat after a fixed number of normal ones. Further, by using the randomisation functions in the output sequencer you could produce errant beats randomly and mark the times at which they occurred.

Play waveform

The Sampling configuration dialog Play waveform page holds the list of waves for on-line output. You add waves with the Sample menu Offline waveform output dialog and by scripts. The dialog settings are used the next time you sample data.



Use When checked, 1401 space is reserved for the wave and it can be played.

Key This is a single character or two hexadecimal digit code in the same format as a marker code (see page 4-6) that identifies the wave; no other wave may have the same code nor may you use code 00. This code is added to the Keyboard marker channel when you click on a button in the Play waveform control bar.

Label A label of up to 7 characters that is used to label buttons in the Play waveform control bar. If you use `&`, the next letter is underlined on the button and you can use it as a short-cut to the key when the control bar has the input focus.

DACs This field lists the Digital to Analogue Converters to play your waveform out of. You can change the list with the DACs control below the list of waves.

Rate The number of samples to output per second per channel. This is set when the wave is added to the list. You can vary the replay rate with the Speed control in the range 0.25 to 4.00. If the speed control is set to any value other than 1.00 the Rate field shows the multiplying factor as well as the rate.

Size This is the number of bytes of 1401 memory that are needed to hold the wave.

Cycles The times to play the wave. Use 0 for a very large number (about 4 billion).

Source This field describes where the data for the wave is stored. In the following descriptions, Name is a data file name, sTime and eTime are the start and end times of the data and chans is the list of channels to read the wave from.

Name (sTime-eTime) chans: the wave is in the data file.

Wave from Name (sTime-eTime) chans: the wave is held in memory and saved in the sampling configuration and was read from the file.

Wave from script: the wave is held in memory and saved in the sampling configuration and was generated by a script.

Place holder for script: a script reserved space, but did not generate a wave.

Trigger Check this box for waveform output that is enabled by play requests and that starts on a 1401*plus* E3 trigger or the Trigger input for other 1401s. If this is not checked, a play request starts the waveform playing immediately.

Link You can link waves with identical DAC channel lists together. Linked areas play in order with no time gap between them at the rate set by the first wave played.

Convert This button changes data held in a file into data held in memory and vice-versa.

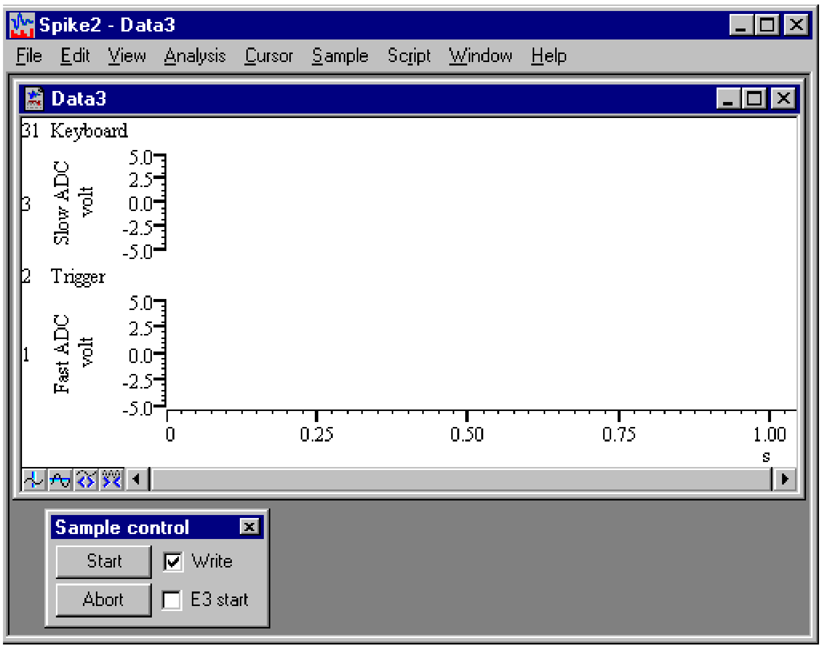
Update Applies any changes you have made to the current wave.

Delete Delete the current wave.

Opening a new document

Once you have set the sampling configuration you can open a new data document. Select New from the File menu, then Data Document for the file type. Data documents differ from all other Spike2 documents as they are always stored on disk. Other document types are kept in memory until you save them. We keep data documents on disk because they can be very large. When you save a new data document after sampling, Spike2 moves it to the disk volume and directory you specify. When you use the File menu New command, Spike2 creates a temporary file in the directory specified in the Edit menu Preferences. If you do not specify a directory in the preferences, the location of the temporary file is system dependent.

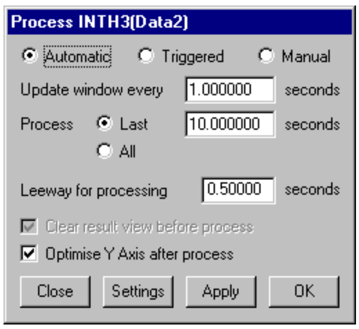
Typical display after opening a new file



The exact appearance varies, depending on the configuration. Sampling begins when you click Start in the Sample control toolbar (the Sample menu duplicates the controls in this window). If the E3 start box is checked, sampling waits for an external signal. You can set the display and analyses required before sampling. For example, to set an interval histogram you can select that analysis exactly as you did in the *Getting started* chapter. There is a difference, however. When you click on New, a new dialog appears.

Process dialog for a new file

You create result views with the Analysis menu New Result View command in the same way as when working off-line. However, the Process dialog, which controls when and how to update the result window, has additional options to work with a data file that grows in length. The radio buttons select Automatic, Triggered or Manual updates (see the *Analysis menu* for a full description).



The standard on-line mode is Automatic. This mode adds new data to the result at a user-defined interval. You can choose to accumulate a result for all the data, or produce a result for the last few seconds. The other two modes are for specialised uses and should not be used unless you are certain they are what you want.

This dialog disappears once you select either OK or Cancel, however you can recall it with the Process command in the Analysis menu.

Sample control toolbar

The Sample control toolbar holds several buttons and a check box and controls the data sampling process. You can dock this bar to any edge of the Spike2 application window or leave it floating. The toolbar becomes visible (if it was invisible) whenever sampling starts unless the start command comes from the script language. The position of the toolbar is saved in the sampling configuration. However, if the toolbar is visible and docked, we do not reposition it as we assume it is where the user wants it.

Before you click the Start button you can choose if you want to start on a low-going TTL compatible pulse on the 1401*plus* Event 3 front panel or on the Trigger input with the micro1401 and Power1401.

If you check the E3 start option and click Start, the toolbar contents change and the word Waiting flashes until a suitable signal is applied to the E3 or Trigger input. Use this method to synchronise the start of sampling with an external event. Sampling starts within 1 or 2 µs of the external signal.

You can also decide which portions of your data are to be saved on disk, and which portions are of only transitory interest. The controls are duplicated in the Sample menu; however the Sample control toolbar needs fewer mouse clicks. The buttons are:

Start — This is displayed before data capture starts. Click the button to start sampling. If the E3 start is checked, Spike2 waits for the trigger before continuing.

Stop — This is displayed while data is sampled. Click this button to stop sampling and keep the data. If no data was saved, the empty file is discarded.

Abort — This button is used to abandon sampling and discard the new file. You can use this button before sampling starts, or while sampling is in progress.

Reset — This button appears when you click Start. It stops sampling, discards any saved data, and waits for you to start sampling again with the same document.

The Write check-box is normally checked, to save data to the data document. If you clear this box, no data is saved until it is checked again. Spike2 keeps a certain amount of data in buffers in memory, so it can display recent data on screen even if you have decided not to save it to disk. Whenever you change the state of the check-box, a marker is added to the keyboard channel (code 00 when writing is enabled and code 01 when it is disabled). If you clear the box, the Write text moves from side to side to alert you and new data is drawn in the Not saving to disk colour.

High sampling rates

We do not recommend that you swap to other applications during sampling as Spike2 uses system idle time to save data. If another application takes over the machine for long periods, data buffer overflow can occur. Buffer overflow can also occur if the data rate is so high that the 1401 device driver cannot empty 1401 memory before it has become full. This should not occur with PCI and USB interfaces unless the overall data rate is very high. Spike2 detects buffer overflow and stops sampling if this happens.

If the 1401 detects that the host computer is slow writing to disk, it requests a “catch-up” mode where Spike2 abandons on-line display of new data (the display for new data is greyed out) and more time is given to writing to disk.

If the 1401 detects that the input event rate is too high for the 1401 to process, the special keyboard marker code `FF` is added to the keyboard channel. Sampling does not stop, as subsequent event times will be correct. If this happens, check that you have set realistic expected sample rates for the event channels.

Saving configurations

It would be very tedious if you had to setup the exact screen configuration you wanted each time you sampled data. To avoid this, you can save and load sampling configurations from the File menu. The saved sampling configuration includes:

The position and size of the application window
The list of channels set for sampling and their sampling parameters
The position of all windows associated with the new file
The displayed channels and event display modes of the channels in time windows
The name of any output sequence document to be used during sampling
The list of waves and any associated waveform data for on-line waveform output
The processing and update modes and positions of all result windows

The configuration does not include the contents of result windows. Whenever sampling finishes, the application saves the configuration as `last.s2c`. When you run Spike2, it searches for and loads the configuration file `default.s2c`. If this cannot be found, it uses `last.s2c`. These files are kept in the directory from which Spike2 was run. Remember that you can always recall the configuration that you used most recently, even if you forgot to save it.

If you have several configurations that you use very regularly, you can add them to the Sample bar with the Sample menu Sample Bar List… command. Once you have done this you will be able to start sampling with a saved configuration by clicking a button on the Sample bar (see the *Sample menu* chapter for a full description of the Sample bar). Alternatively, you could keep shortcuts to configuration files on your desktop and start Spike2 by double clicking them.

Warning We suggest that you do not rely on `last.s2c` for important sampling configurations as it is overwritten each time you sample. It is better to save your configuration to a named file and load it using the File menu or the Sample Bar or by double clicking the file.

Sequence of operations to set the configuration

This section describes a sequence of operations that you should follow to build a new sampling configuration from scratch. You will find that once you have built a few configurations, it is much simpler to load an existing configuration and change the sections that do not fit your requirements, rather than re-build entirely. The steps are:

1. Open the Sampling Configuration dialog from the Sample menu
2. In the Channels tab set the channels to be sampled and their sampling rates
3. Adjust the Resolution values to give the best fit of sampling rates for any waveform channels
4. Select the appropriate sampling Mode for your needs
5. In the Sequencer tab select any required output sequence file or select None
6. Set the Automation values as required by your application.
7. In the Play waveform tab select any waves needed for waveform output.
8. Click the Run now button.
9. Arrange the time view as you require and add any duplicate windows
10. Add any result windows, set their update mode and position on screen
11. Use the File menu Save Configuration command to save the configuration

Once you have saved a configuration, you can re-use it by loading it before you use the File menu New command to start a new data file. You can combine the loading and opening a new file by adding the saved configuration to the Sample bar with the Sample menu Sample Bar List… command.

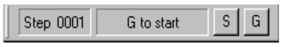
Overview

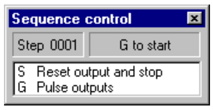
While sampling data you can generate precisely timed digital pulses and analogue voltages, monitor your experiment and respond to input data in real time with the Spike2 output sequencer. An output sequence is a list of up to 1023 instructions. The sequencer runs at a constant, user-defined rate of up to 100 instructions per millisecond with the Power1401, up to 50 per ms with the Micro1401 mk II and up to 20 per ms with the micro1401 and 1401*plus*. The sequencer has the following features:

- It controls digital output bits 15-8 to produce precisely timed digital pulse sequences. In the Power1401 and micro1401 it also controls digital output bits 7-0.
- It controls the 1401 DACs (Digital to Analogue Converters) to produce voltage pulses and ramps.
- It can play two cosine waves at variable speed and amplitude through the DACs.
- It can test digital input bits 7-0 and branch on the result.
- It can record the digital input state or an 8-bit code to the digital marker channel.
- It supports loops and branches.
- It can randomise delays and stimuli.
- It has 64 variables (`V1` to `V64`) that can be read and set by on-line scripts. The variables are used in instructions and for simple mathematical operations.
- It can read the latest waveform value from a sampled channel and the number of events captured from an event channel. Using this information, real-time (fractions of a millisecond) responses to input data changes are possible.
- It can control and monitor the arbitrary waveform output.

You write sequences with the text editor where each line of text generates one instruction or with the graphical editor where each graphical item generates one or more instructions.

Sequencer control panel





You can show and hide the control panel with the Sequencer Controls option in the Sample menu or by right clicking on any toolbar to activate the context menu. You can also dock it on any edge of the Spike2 application window. When undocked, the control panel displays any sequence entry points as the key that activates them and a descriptive comment. When docked, the keys are displayed as buttons and the comment is hidden to save space. Move the mouse pointer over a key to see the comment as pop-up text. Click the mouse on a key and the sequencer will jump to the instruction associated with the key. The key is also stored as a keyboard marker. This is equivalent to pressing the same key in the time window or using the script language `SampleKey()` routine.

The control panel displays the next step in the sequence and any display string associated with it. You can use display strings to prompt the user for an action or to tell the user what the sequence is doing. Display strings are set with the text editor. The sequencer always starts at the first instruction. You cannot re-route the sequencer until sampling has started.

The control panel is always displayed if you start sampling from a Spike2 menu command or from a dialog. If the sample command comes from the script language, the control panel visible state does not change. The control panel position is saved in the sampling configuration. However, the position is restored only if the control is currently invisible or floating, and the saved position was floating; if the control panel is docked, we assume it is positioned where you want it.

Creating sequences

There are two ways to create output sequences: as an output sequence text file with the `.PLS` extension or as part of the sampling configuration using the graphical sequence editor. The table summarises the main differences between them.

	Text sequence	**Graphical sequence**
Edited with	Built-in text editor	Built-in graphical editor
Visualise output	No	Yes
Stored as	Output sequence `.PLS` files	Part of the sampling configuration
Implemented by	Machine code like language	Drag and drop editing
Ease of use	Takes time to learn	Very easy to learn and use
Flexibility	All features available	Uses pre-set building blocks
Timing	One instruction per text line	Several instructions per item

If your requirements can be met by the graphical sequence editor, you will find it much easier to use than the text sequence editor. However, it has limitations and you can write more complex sequences using all sequencer features with the text editor.

Sequences produced by the graphical editor are converted internally to the `.PLS` file format before they are used. You can save a graphical sequence as a `.PLS` file; you cannot convert a `.PLS` file into a graphical sequence.

The rest of this chapter describes the graphical output sequence editor, then the text editor, and finally the low level instructions used by both. You do not need to read about the low level instructions to use the graphical editor. However, knowledge of how the sequencer works will give you a better understanding of its capabilities and limitations.

Sequencer speed

The output sequencer runs at a fixed rate set by a clock inside the 1401. You set the clock rate by setting the interval between clock ticks in milliseconds. This is done in the Sequencer Tab of the Sampling configuration dialog when using the graphical editor or using the `SET` directive in the text editor. You can set intervals from 0.05 milliseconds up to 3000 milliseconds on any 1401. The Power1401 and Micro1401 mk II allow intervals down to 0.01 milliseconds. If you intend to set intervals less than a millisecond or will use the sequencer text editor you should read the following technical information.

Sequencer technical information

The sequencer clock starts within a microsecond of recording time zero and is time locked to the 1401 event timing and waveform channel recording. Each clock tick books an interrupt to run the next sequencer instruction and updates digital output bits 15-8 if they were changed by the previous instruction.

An interrupt is a request to the 1401 processor to interrupt what it is doing at the earliest opportunity and do something else, then continue the original task. The time delay between the interrupt request and the instruction running depends on what the 1401 is doing when the clock ticks and the speed of the 1401. This delay is typically a few microseconds, so instructions do not occur precisely at the clock ticks but changes to digital output bits 15-8 do. Changes made by the sequencer to the 1401 DACs and digital output bits 7-0 occur a few microseconds after the clock tick.

The table shows the minimum clock intervals and the timing resolution, and the approximate time per step and the extra time used for cosine output for each 1401. Notice that the first two are in units of milliseconds and the second two are in microseconds.

	Power	Micro mk II	micro1401	1401*plus*
Minimum tick (ms)	.010	.010	.050	.050
Resolution (ms)	.001	.001	.004, .006, .010	.004, .006, .010
Time used per tick (μs)	<1	~1	<8	<10
Cosine penalty/tick (μs)	<1	~1	~4	~10

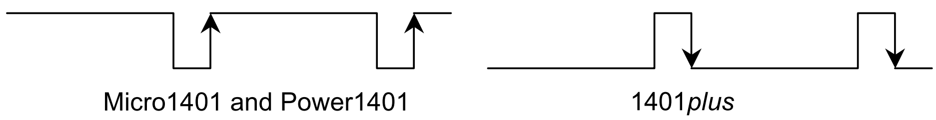
The Minimum tick is the shortest interval we allow you to set. The Time used per tick is how long it takes to process a typical instruction. The Cosine penalty/tick is the extra time taken per cosine output. Time used by the sequencer is time that is not available for sampling, spike sorting or arbitrary waveform output. To make best use of the capabilities of your 1401 you should set the slowest sequencer step rate that is fast enough for your purposes.

If you overload the system so much that the output sequencer cannot keep up, the result depends on the type of 1401. With the Micro1401 and Power1401, sampling stops with an explanatory message. The 1401*plus* does not detect this error and waits for the next clock tick to run the instruction.

The interval you set must be a multiple of the Resolution field for your 1401. This is not an issue for the Power1401 and Micro1401 mk II. There is a choice of values for the micro1401 and 1401*plus* due to their hardware implementation. The table shows three possible values; in fact any value given by `n * m / 1000` where `n` and `m` are integers greater than 2 is an acceptable resolution. Spike2 will not use a sequence if the interval is not an exact multiple of an achievable resolution for your 1401 as this would lead to inaccurate timing.

Sequencer clock output

The output of the clock that controls the sequencer is available on the 1401*plus* OUT front panel connector and the Power1401 and Micro1401 Clock connector. This TTL output signal starts high with the Micro1401 and Power1401 and low with the 1401*plus*. The sequencer steps are synchronised with the rising edges of this output for the Micro1401 and Power1401 and the falling edges with the 1401*plus*. The minimum pulse width is 1 microsecond.



This clock runs whenever you sample with an output sequence. It can be used to synchronise external equipment. For example, if you connect this output to a counter, and place the counter in the field of view of a video camera that is used to record some other aspect of your experiment, the number on the counter links visual data with an exact time in the Spike2 file. If your sequencer ran at 1 millisecond per step, the counter would display time in milliseconds.

Marker channel and digital input conflict

If you record a digital marker channel, this uses the same hardware in your 1401 as the sequencer instructions `DIBEQ`, `DIBNE`, `WAIT` and `DIGIN` that read the digital input bits. Reading digital input bits 7-0 clears the hardware flag that indicates that a marker event has occurred. These instructions can cause events to be missed on the marker channel if they read the digital input at exactly the same time as the marker event arrives. The graphical editor commands that wait for a digital input condition also have this problem.

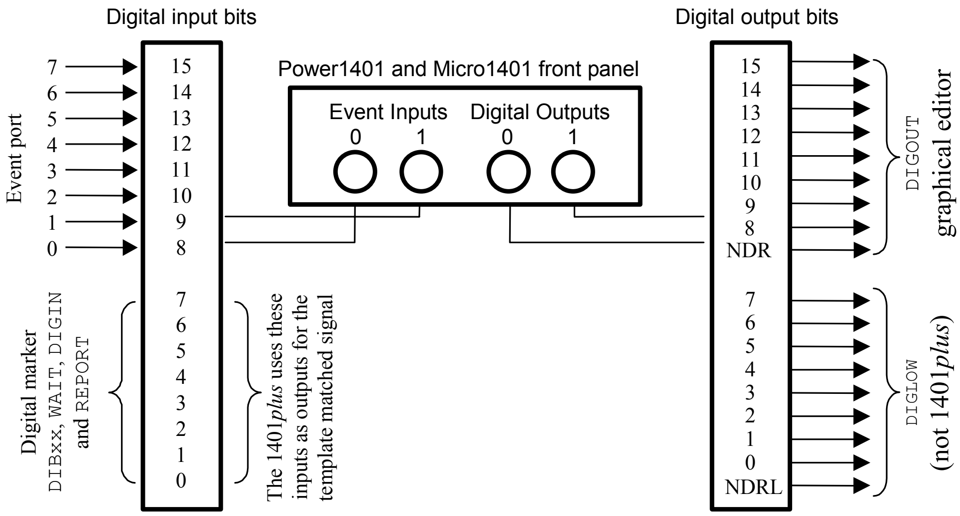
When you sample, Spike2 will warn you if there is a conflict in the use of these digital input bits. You can choose to hide the warning for the remainder of your Spike2 session. We will release updates to the Micro1401 mk II and Power1401 firmware to allow reads of the digital input that do not clear the hardware flag. The warning will be suppressed when the new firmware is installed in your 1401.

Digital input and output

The 1401 family has 16 digital inputs and 16 digital outputs. Digital input bits 15-8 are the event ports and are not used by the output sequencer. Digital input bits 7-0 are used for the Digital marker channel; you can also test these bits from the sequencer.

The sequencer controls digital output bits 15-8 individually to generate accurately timed pulses. Output bits 7-0 can be set with the `DIGLOW` instruction. The on-line template matching code can also use bits 7-0 of the output to signal spikes that match templates.

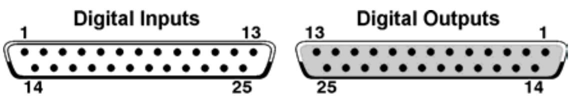
In a text-based sequence, the digital outputs are controlled by the `DIGOUT` and `DIGLOW` instructions and the digital inputs are tested with `DIBxx`, `WAIT`, `DIGIN` and `REPORT`. The graphical editor controls output bits 15-8 and tests input bits 7-0 with the delay and branching instructions. **Warning**: testing input bits 7-0 can conflict with recording events on the digital marker channel.



The digital input and output ports are both 25-way connectors. The pins for the data and ground are the same on both. See your *1401 Owners Manual* for full details of all pins.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	GND
Pin	1	14	2	15	3	16	4	17	5	18	6	19	7	20	8	21	13

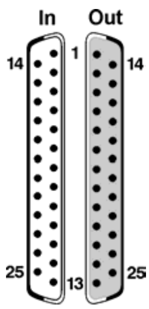
Power1401 and Micro1401



Digital output bits 8 and 9 are connected to the front panel as Digital Outputs 0 and 1 in addition to the rear panel. If you have the Spike2 top box, the remaining digital output bits 10-15 are also available on the front panel as Digital Outputs 2 to 7. The NDR output (New Data Ready, output connector pin 12) pulses low for 1 microsecond after a change to the digital output bits 8 to 15. NDRL (output connector pin 23) pulses low for 1 microsecond after a change to output bits 0 to 7.

1401plus

The 1401*plus* digital bits 7-0 are bi-directional (they can be set individually as inputs or outputs) and are connected to both the input and output connector. Spike2 normally uses them as inputs. The `DIGLOW` instruction has no effect on the 1401*plus*. There is no equivalent of the NDR signal available with the 1401*plus*.

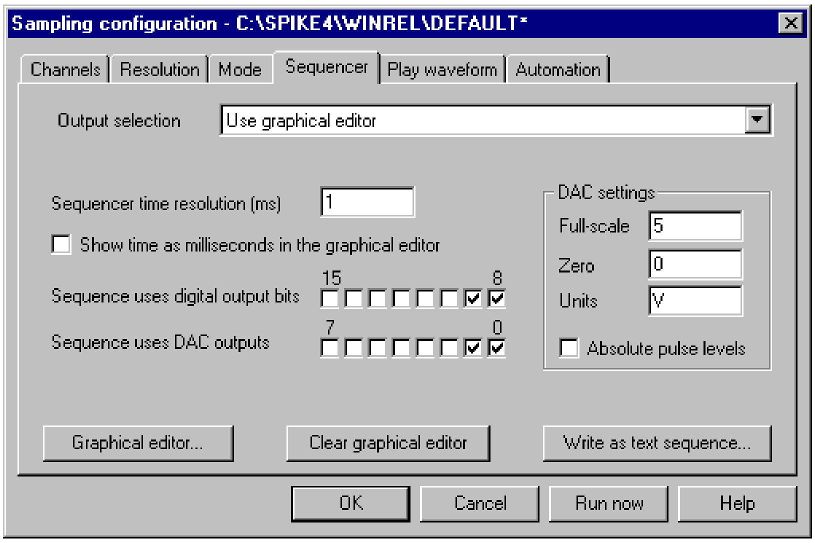


If you use the WaveMark *template matched* output signal, some or all of digital input bits 7-0 are set as outputs. The `DIBxx`, `DIGIN` and `WAIT` instructions read back the output state of the bits used for template matching. Do not drive these pins as inputs when they are used as outputs!

If the 1401*plus* has a 1401-18 event discriminator card, digital input bits 7-0 are the *event detected* TTL output connections; use the output connector bits 7-0 for the input signals.

The graphical editor

To view the graphical editor, open the Sampling configuration dialog and select the Sequencer tab. Then select Use graphical editor from the Output selection drop down list. The editable fields in this dialog set values that apply to the entire sequence:



Sequencer time resolution

This sets the time resolution of your sequence and the clock interval of the sequencer clock. This is also the minimum duration of any pulse. All actions in the sequence occur at integer multiples of the time you set here. You can set values in the range 0.01 to 3000 milliseconds. You need a Power1401 or Micro1401 mk II to set values less than 0.05 milliseconds (see page 5-2). Some actions take more than one clock interval.

Show time as milliseconds

Check this box to display and edit time in the graphical editor as milliseconds and not seconds. This is purely for your convenience; if your sequence sections are all less than a second you will probably find it more convenient to use milliseconds.

Sequence uses digital output bits

Check the boxes for the dedicated digital outputs that you will use. Only these outputs will appear in the editor, reducing visual clutter. If you do not require any digital outputs, clearing all the check boxes will save an instruction at the start of each sequencer section.

Sequence uses DAC outputs

Check the boxes for the Digital to Analogue Converters (voltage output devices) that you will use in your sequence. Unused DACs are not included in the graphical editor (to reduce visual clutter) and no sequence code is generated for them, which saves instructions at the start of each sequence section.

If you use one or more DACs for arbitrary waveform output only do not include them here unless you want to be certain that they have a defined value when you enter a sequence section. All 1401s have at least DACs 0 and 1; the 1401*plus* and the Power1401 have DACs 0 to 3. Top boxes allowing up to 8 DACs can be added to recent 1401s.

DAC full-scale, zero and units

You can define the DAC outputs in units of your choice. 1401 DACs normally have a range of ±5 Volts, but ±10 Volts systems exist. Set the full-scale value to the value in the units that you want to use that corresponds to the maximum output from the DAC. Set the zero value to the value in your units that corresponds to a DAC output of 0 Volts. Set the units to the units you want to use.

The usual settings for a ±5 Volt system is to set the DAC full scale to 5 and the DAC 0 value to 0 and units to `V`. If you want to set the output in millivolts, set the full scale to 5000, zero to 0 and units to `mV`.

Absolute pulse levels The DAC pulses take their starting level as the current DAC value at the pulse start time. The DAC then changes to another value, then back to the original level. Normally you define pulses in terms of the pulse amplitude relative to the starting level and all pulses add. If you check this box, then you set the absolute level for the DAC to change to.

Write as text sequence… You can use this button to output the graphical sequence as a text sequence. You might do this if the graphical interface almost does what you need and you need to hand-edit a few extra instructions, or if you need to save the sequence for documentation purposes. A file selector dialog opens for you to choose a name for the `.PLS` file.

Clear graphical editor… This wipes all graphical sequences. You are asked if you really want to do this; you cannot undo this action.

Graphical editor… This opens the editor so you can make changes to the sequence.

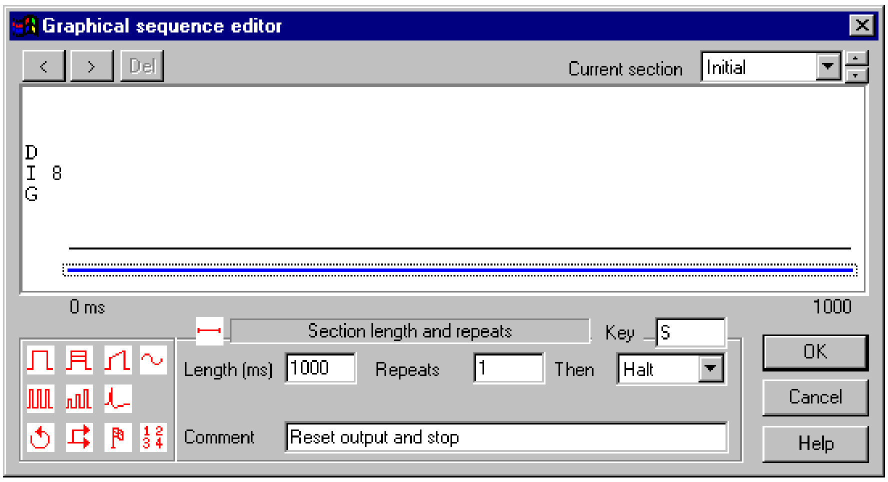
Getting started

To get accustomed to the graphical editor, we will produce 10 millisecond wide TTL pulses at 1 second intervals from digital output bit 8. We will associate the key `G` for Go with the pulses, and the key `S` for Stop will stop them.

Open the Sampling configuration dialog Sequencer tab and select Use graphical editor. Click the Clear graphical editor button to remove any previous sequence. Set the Sequencer time resolution to 10 milliseconds and check the Show time as milliseconds box. We do not need any DAC outputs, so clear all the DAC outputs check boxes. We only need one digital output, so check the digital output 8 box and clear all the others. Now click the Graphical editor button.

At the top right of the new dialog there is a drop down list to select the current sequence section. A section has a length and repeat count. When the repeats are done, the section either stops or jumps to another section. To start with, make sure Initial is selected as the Current section. The Initial section always runs first and it sets the initial conditions.

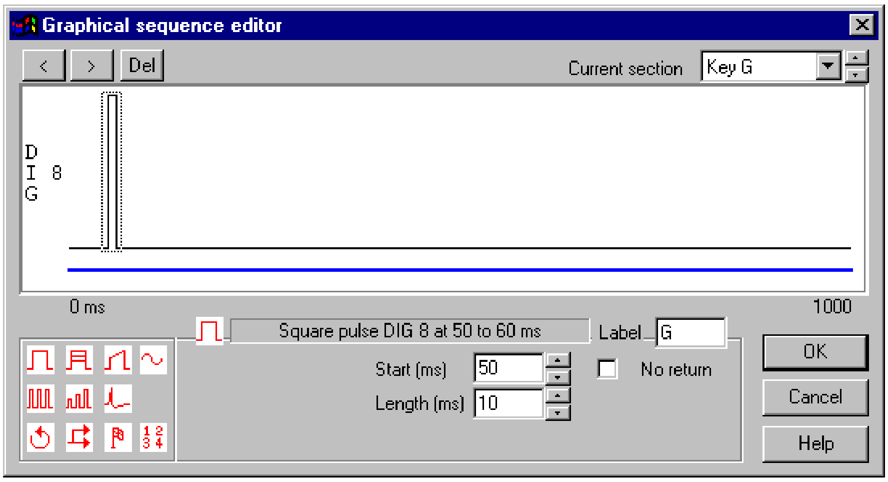
The graphical representation of a section always contains a *control track* drawn as a thick line at the bottom. We choose output on digital bit 8 only, so there is a single digital output in the remainder of the space. There is always one item selected in this area; the selected item has a grey rectangle around it. Click on the control track now.



In addition to setting the initial state, we will use this section to stop our pulse output, so set the Key field to `S`, and set the Comment field to `Reset output and stop`. This comment will appear in the sequencer control panel for the `S` key. You can leave the Length, Repeats and Then fields at their default values (`1000`, `1` and Halt).

The next task is to create the pulse train and associate it with the G key. Select Key G in the Current section field. The control track will be selected; click on it if it is not selected. Make sure that the Key field is set to G. Set the Repeats to 0 to mean repeat forever and the section length to 1000 milliseconds. The Then field will grey out as the repeats never end. Set the comment to `Pulse outputs`.

There is a palette at the bottom left of the dialog; you can drag icons from the palette and drop them on the DAC, digital output and control tracks. Click on the top left item in the palette and drag it to the digital output track and release it to create a pulse.



Edit the start time of the pulse to 50 and the length to 10 milliseconds. This completes the pulse setup. Click the OK button.

You will now be back in the Sampling configuration dialog. Click the Run now button to start sampling with the output sequence you have just created. The sequencer control panel will display two items that you can select: G Pulse outputs and S Reset output and stop.

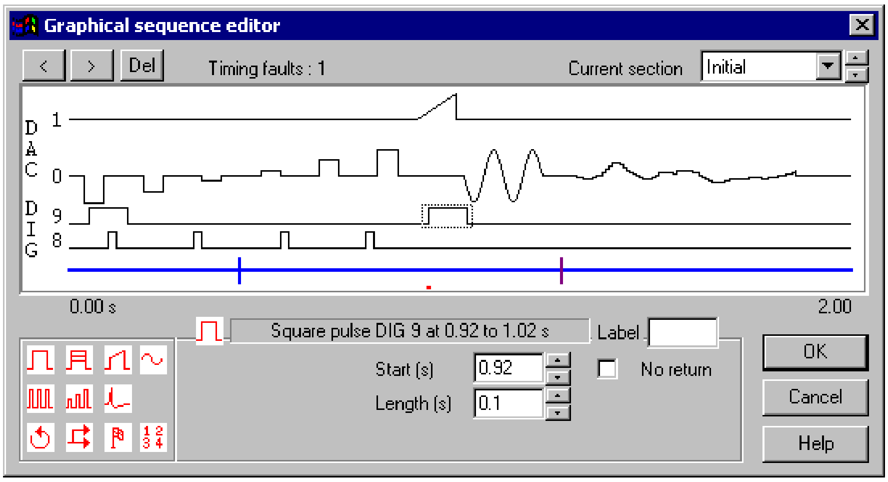
Start sampling – the sequencer will run the Initial section, which sets the starting values for all the digital and DAC outputs we have chosen to control. We did not request any other action in this section so nothing will happen until you use the G button in the sequencer control panel or select the sampling window and press the G key to start the output pulses from digital output 8. You can stop the pulses with the S button in the control panel or the S key on the keyboard.

Graphical editing

To open the graphical editor, select Use graphical editor and click the Graphical editor… button in the Sequencer tab of the Sampling configuration dialog. You can resize the dialog by clicking and dragging an edge. Double-click the title bar to maximise the dialog, double-click again to minimise it. The editor window has five areas:

1. a window with a graphical representation of the output sequence
2. above the graphical window are controls to iterate through and delete selected items, a message area and the Current section selector
3. the lower left-hand corner holds a palette of items to drag into the graphical window
4. the lower right hand corner has control buttons
5. the settings for a selected graphical item lie between the palette and the buttons

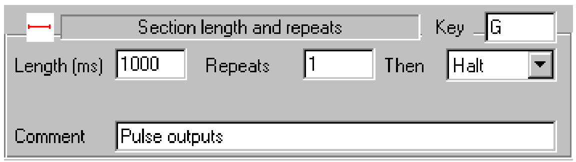
The OK and Cancel buttons both close the dialog. OK accepts all changes, Cancel rejects all changes. The Help button displays the information you are reading!



Sections There are 27 sections called Initial and Key A through Key Z. The Current section field sets the section to display and edit. The Initial section runs when the sequence starts; in many cases this may be the only section you need. The remaining sections are optional. An unedited section displays one horizontal line for each DAC and digital output that is set for use in the Sequencer tab of the sampling configuration, plus a thicker line at the bottom for the control track. The lines for the DAC and digital output show the state of these outputs, the control track holds all other sequencing actions.

Selecting items Click an item in the graphical view to select it. A grey rectangle marks the selected item and the item settings appear below the display. The < and > buttons at the top left select the previous and next item on the current track; they are very useful when items overlap.

Section settings Click on the control track clear of any dropped items. You can now set the associated key, the section length, number of repeats, action when the repeats are done, and a comment for the section that is displayed in the sequencer control panel to identify sections with keys. You can set a section length up to 10000 seconds and up to 1000000 repeats! If you want a section to repeat forever, set the repeat count to 0. The Then field determines what happens when the section ends. You can stop the sequencer with Halt, or select another section to run.



Key field Normally, the Key A to Key Z section are assigned `A` to `Z` and the Initial key is blank. However, you can set the key to any of `A-Z`, `a-z` or `0-9` (upper and lower case are different). If you delete the key, the section cannot be run from the control panel. Two sections that contain outputs cannot share the same key.

Adding and deleting items The graphical palette at the bottom left-hand corner of the dialog contains all the items that you can add to the display. Move the mouse over an item so see a short description. Click an item and drag it to a suitable track, then release to add it to the section. The Del button on the top line of the dialog removes the selected item. You cannot remove the control track or the lines that represent the initial state of the DACs and digital outputs.

Timing faults The sequencer will try to match the exact timing you request. However, if this cannot be done, timing conflicts are marked in red below the control track and the number of conflicts is given in the message area. You will usually get a conflict if you try to position any action at the start of a section. This is because the first instructions in a section set the initial digital output state and the state of each DAC.

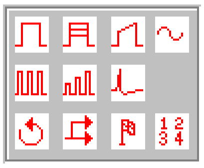
You can choose to ignore timing faults. If you do, the sequence will still run and the changes will occur as close to the requested time as they can.

Setting initial DAC and digital levels

To set the initial levels, click on the DAC and digital tracks clear of any added items. You can set the initial digital output level as 0 or 1 and the initial DAC level in the units set by the Sequencer tab of the Sampling configuration. The sequencer instructions that set these levels are the first in each section so any output change caused by them will be as close to time 0 in a section as possible. The fact that these levels may not be set at exactly time zero is not considered a timing fault.

Graphical palette

The palette contains 11 items you can drag and drop in the graphical window to generate a sequence. There are three dropping zones: a digital output, a DAC output and the control track. Items will only drop onto suitable targets. For example, you cannot drop a sinusoid on a digital track. There is no need to drop the items at exactly the right time point; you can edit the position afterwards.



Common editable fields

When you select an item in the graphical window you can edit the fields that relate to it. The following are described here to avoid repeating the descriptions.

Label

This field is normally left blank. You can use it to label the selected item so that you can branch to it. A label can be up to 6 alphanumeric (A-Z, 0-9) characters long and is case insensitive; `abc23` and `ABC23` are the same label. Labels must be unique in each section, but you can have the same label in a different section.

You cannot set a label for the initial levels of the digital and DAC tracks or for the control track as these items all start at time 0 and can be branched to be referring to the section name. You cannot set a label for arbitrary waveform output as this would prevent an important optimisation required when the sequence is generated.

Start time
At

All digital and DAC change items have a Start time field and all items on the control track have an At field. The time is in units of seconds or milliseconds, as set by the Graphical editor settings, and is relative to the start time of the sequence section. There is a spin control to nudge the time on or back by the sequencer time resolution.

If you add items to the control track that take an unknown time to complete, for example a random delay or a wait for an input signal to achieve a set value, the At time determines where the items are drawn in the graphical editor. In this case the sequencer will maintain the time intervals between items wherever possible.

For pulses, ramps, sinusoidal and arbitrary waveform output, the sequencer attempts to produce the first output change at exactly the time you specify. For other item types, the sequencer attempts to run the first instruction of the item at the specified time.

Length

Several items have a length in seconds or milliseconds. For all the pulse types, this is the period for which the pulse changes to the amplitude or level you set before it reverts to the original level. For a ramp and a sinusoid, this is the length of the output. For arbitrary waveform output, this is set to the length of the arbitrary wave you associate with the command. You can set it to be less than the length of the wave, in which case the output is stopped after the time you have set. There is a spin control to nudge the time on or back by the sequencer time resolution.

Gap

The two pulse train items set the length of each pulse and the gap between them in seconds or milliseconds.

Level (units)
Size (units)

These fields are used with DAC pulse outputs. The field you see depends on the state of the Absolute pulse levels checkbox in the Graphical editor settings. The Size field sets the amplitude of a pulse, the Level field sets the absolute level of a pulse. You set the value in the DAC units set in the Graphical editor settings.

No return

Normally the output from single pulses, ramps and sinusoidal output returns to the background level at the end of the item. If you check this box, the output change produced by this item is not removed at the end of the item. To indicate this, the grey rectangle surrounding the item extends to the end of the section. You could use this to ramp a DAC up to a value and leave it there.

Overlapping items

If digital pulses overlap, the result is the logical OR of the pulses.

If DAC items overlap on the same channel, the output depends on the state of the Absolute pulse levels checkbox in the Graphical editor settings. If this box is clear, the result is the sum of the outputs. If this box is checked, the output level is set by the last item in the overlap. There is an exception: arbitrary waveform output overrides all other items.

When adding single pulses and pulse trains where the result would exceed the range of the DAC, the output is limited to the DAC range. However, pulses with an amplitude change on repeats can exceed the DAC range and wrap around. A value that goes off the top of the range will reappear at the bottom; a value that goes off the bottom of the range will reappear at the top.

Single pulse

You can drag this item onto either the digital or the DAC outputs. For a digital output, this sets the output to be the inverse of the initial level set for this output in this section.

For a DAC output, you set the amplitude of the pulse with the Size field or the absolute level with the Level field.

Single pulse, amplitude change on repeat

You can drag this item to a DAC output. It is for use in a repeated section. The Size/Level field sets the initial amplitude or absolute level of the pulse the first time the section runs. The Change field sets the amplitude change to apply on each subsequent repeat. You set the number of repeats by selecting the control track and editing the Repeats field.

The changes are calculated in real time in the 1401. If the initial level plus the number of changes times the number of repeats exceeds the DAC range, the output value will wrap around.

Ramp

You can drag this item to a DAC output. The From and To fields set the initial and final amplitudes or levels of the ramp depending on the state of the Absolute pulse levels checkbox in the Graphical editor settings.

In the current implementation, no other activity is allowed while a ramp is generated except a sinusoid. You will find that any item that starts within the time range of a ramp will cause a timing error to be flagged and the item start will be delayed until after the ramp when you run the sequence. This may be fixed in a future release.

Sinusoid

You can request sinusoids on DACs 0 to 3. However, a 1401*plus* can only generate them on DACs 2 and 3, all other 1401s can only generate them on DACs 0 and 1. In most cases, you will be using only one type of 1401, so you should set the sinusoids on the correct pair of channels. If you request output on channel 0 and 1 and use a 1401*plus*, the output will appear on DACs 2 and 3. If you request output on DAC 2 and 3 and don't use a 1401*plus*, the output will appear on DACs 0 and 1. We may tidy this up in a future release.

The sinusoid amplitude is defined by the Size (units) field; this is not affected by the Absolute pulse levels checkbox. You can offset the sinusoid with the Centre (units) field. If the Absolute pulse levels checkbox is clear, the sinusoid and offset is added to the DAC value. If the checkbox is set, the DAC output is defined by the sinusoid and offset.

The Period field sets the time for one cycle of the sinusoid in seconds or in milliseconds. The Start phase field sets the initial phase in degrees. The output is a cosine, so a phase of 0 means start at maximum amplitude. A phase of –90 or 270 produces a sine output.

Pulse train

You can drag this to a DAC or digital output. It generates a train of pulses defined by the number of pulses (Pulses field), the length of each pulse (Length field) and the gap between each pulse and the next (Gap field). There is no gap after the final pulse in the train. For a DAC output you can also set the amplitude with the Size/Level field.

This method is inefficient if you need to generate a large number of pulses as each pulse takes several instructions. You should consider setting up a single pulse as a section and repeating the section to create the pulse train.

Pulse train with varying amplitude

You can drag this to a DAC output. It generates a train of pulses defined by the number of pulses (Pulses field), the length of each pulse (Length field) and the gap between each pulse and the next (Gap field). There is no gap after the final pulse in the train. You set the amplitude of the first pulse with the Size/Level field. The pulse size changes by the value in the Change field for each repeat.

This method is inefficient if you need to generate a large number of pulses as each pulse takes several instructions. You should consider setting up a single pulse with changing amplitude as a section and repeating the section to create the pulse train.

Arbitrary waveforms

You can drag this item to the control track to start playing a waveform through the 1401 DAC outputs. You can play arbitrary waveforms through DACs 0-3 (or any combination of these DACs). The DACs used are set when you create the waveform and can be edited in the Play Waveform tab of the Sampling configuration. Create the arbitrary outputs with the Sample menu Output Waveform command or from a script.

Each waveform is identified by a key code. This is either a single alphanumeric key code or two hexadecimal digits for a non-printable code. When you set this in the Key field, the Length field is filled in to the length of the entire wave (or to 0 if there is no matching wave). You can set the length shorter than the entire wave to truncate the output.

Arbitrary output takes appreciable time to set it up; times in excess of 10 milliseconds are possible. When generating the output, this setup is moved as far forward in the sequence section as possible so that the output starts at the exact time you set. Avoid placing a label on the arbitrary waveform item as this blocks this optimisation. If the preparation has not completed by the time you request output to start, the sequence will stall until it is ready.

When the arbitrary output ends, the DAC outputs are returned to the background level as soon as possible. If the arbitrary waveform has more than one channel, there will be one sequencer clock period between each DAC changing to the background level.

Wait for time, condition or variable You can drag this item to the control track only. This item pauses the sequence until a specified condition is met. Because the duration of this item may not be known, it is drawn as if it was of zero width. You can select the following delay types:

Fixed delay Set the time to wait in the Delay field. You will find this useful if you have short periods of sequencer activity separated by long delays.

Random delay This holds the sequence for a time between Min and Max seconds or milliseconds. All delays between the minimum and maximum have equal probability (within the capabilities of the sequencer). You will get the best result for delay ranges that are long compared to the time resolution.

Poisson delay This generates a delay with a Poisson statistic; the delay has the same probability of ending at any time during it. The Time const field sets the average delay length.

Digital input high/low You can wait for a nominated digital input bit in the range 0-7 to be high or low. These are not the same bits as used for the event inputs. If you want to synchronise to a bit changing you must wait for it to be in the opposite state to the one you want first. If you need the sequence to perform actions while you wait, use a branch item. You can wait for combinations of input bits with the text sequencer.

Channel above/below/outside/within You can wait for a nominated waveform channel sampled by the 1401 to be above or below a level set by the Threshold field, or to be outside or within a pair of levels set by the Lower and Upper levels fields. The levels are set in the channel units, as stored in the sampling configuration. If you subsequently change the channel type, the results will not be harmful, but the sequence will not operate as intended!

Next event You can wait for the number of events set by the Count field to occur on a nominated event, marker or WaveMark channel that is sampled by the 1401. When this delay item is reached, the sequence notes how many events have been sampled on the channel and waits until the number increases by the count.

Time reached This item waits until the sample time reaches the value you set. If the sample time has already passed this time, there is no wait.

Cosine phase 0 This item waits for the next time that cosine output on the nominated DAC channel passes through phase zero.

Variable comparisons You can wait for various conditions based on sequence variables. You can use variables 1 to 25 of the 64 sequencer variables. The remaining variables are used to implement the sequence sections. Variable values are 32-bit integers. You can manipulate the variables in the Variable arithmetic item. Variables are described in detail for the text sequencer.

Event burst This item monitors an event, marker or WaveMark channel sampled by the 1401 for a group of events with a user-defined maximum separation. The Intervals field sets the number of gaps to check and the Max time field sets the maximum acceptable interval. If any interval is greater than the maximum, the sequence starts the search again. For this to work well, the maximum interval must be significantly greater than the time resolution of the sequence.

Branch on condition, probability or variable With this item you can break the normal flow of the sequence and branch to a different section or to a label you have defined for an item in the current section. All branches have a Branch destination field in which you can select a section to branch to, or you can type the name of a label in the current section.

When you branch, the timing to the target may not be exactly what you expect. The sequence will take one or more steps to implement the branch and the target instruction may require preparatory steps. Such effects are small unless you use arbitrary waveform output where the preparatory steps can take several milliseconds. If you need the tightest possible control over branch timing you should consider using the text sequence editor. The branches you can set are:

Probability Percent sets the probability of the branch, 0% never branches, 100% always branches.

Digital input high/low The Bit field sets the digital input bit number in the range 0-7 to test.

Channel above/below/outside/within You can branch if a nominated waveform channel sampled by the 1401 is above or below a level set by the Threshold field, or is outside or within a pair of levels set by the Lower and Upper levels fields. The levels are set in the channel units, as stored in the sampling configuration. If you subsequently change the channel type, the result is not harmful, but the sequence will not operate as intended!

Variable comparisons You can branch on the result of comparing sequence variables with constant values and other variables. There are 64 variables, numbers 1 to 64 that can be used. Some variables have special uses. Variable values are 32-bit integers. You can manipulate the variables in the Variable arithmetic item. Variables are described in detail for the text sequencer.

Unconditional This always branches to the destination.

Time comparisons You can compare a variable plus a time offset with the current time and branch on the result. You can set the variable to the current time (plus a time offset) with the variable arithmetic *current time* instruction.

Response with timeout You can wait for a new data item in an event, marker or WaveMark channel sampled by the 1401. The branch is taken if a new item is detected within the timeout period.

Generate digital marker channel event This adds an event to the Marker channel (if it is enabled). You can set the marker code with the Code field or check the Record data box to record the state of digital input bits 0 to 7 (these are not the same bits used for event inputs). The Code field should be set to one character or to two hexadecimal digits.

Variable arithmetic Although the use of variables is more commonly done with the text editor, you can perform basic variable manipulation here. You can use variables 1 to 25 out of the 64 that exist in the sequencer. Variable values are 32-bit integers. In all cases, the variable that is changed is set by the Target var field. Operations are:

Set to value/variable Replace the target variable with the contents of the Value field or of a variable.

Add/subtract value/variable Adds or subtracts the contents of the variable or value.

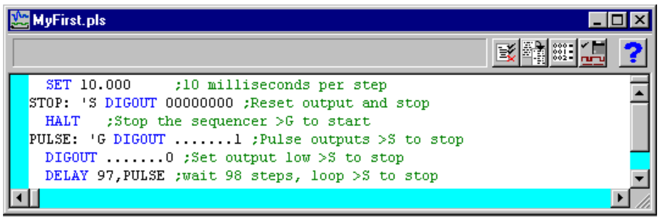
Multiply by value/variable Multiplies the target variable by the variable or value.

Random value This replaces the target variable by a random number that is from 1 to 30 bits long, set by the Bits field. The possible values for an n bit number are 0 up to 2^n minus 1. For example, if the Bits field is 4, the possible results are 0 to 15.

Current time The variable is set to the current sample time plus a time offset, in Spike2 clock ticks, as set in the Resolution tab of the Sampling configuration.

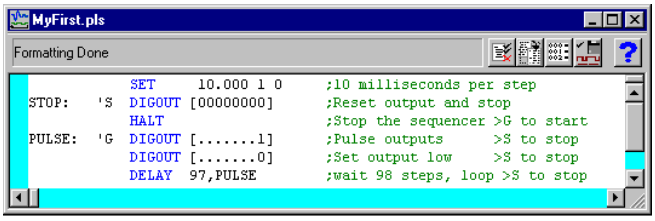
Fixed time The variable is set to a time, in Spike2 clock ticks, as set in the Resolution tab of the Sampling configuration.

The text editor



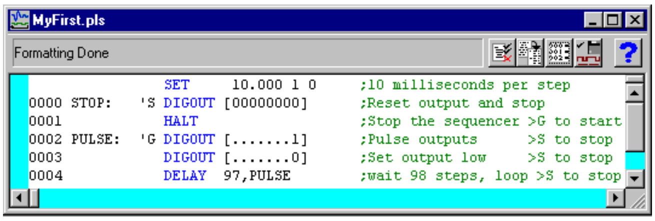
The text output sequence is stored as a text file with the extension `.PLS`. You can open existing Spike2 output sequence files or create new ones with File menu New… (see the *File menu* chapter for details of file types). This picture shows an output sequence that has been typed in without formatting. There are five buttons at the top of the window:

Format



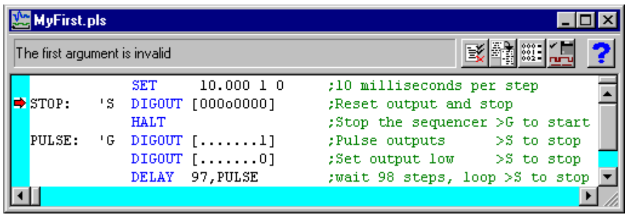
This aligns the labels, key codes, instructions and any arguments, output text and comments and removes step numbers. Undefined labels are not flagged but there must be no other errors. The picture shows the result.

Format with step numbers



This does the same job as the Format button, and also starts each line with the step number. Step numbers can be useful as they give an indication when you are running out of space and can pinpoint the line where your sequence is not behaving as you expect.

Compile



This checks the sequence to make sure that it is correct with no labels missing or duplicated and no duplicated key codes. The picture shows a sequence with a simple error (the o in the second line should be a zero). The line in error is marked and an explanatory message is shown at the top of the screen.

Current

This compiles the document, and then if the text is correct, saves it and makes this the current sequence for use during sampling. If you were to open a new data file and start sampling, this sequence would be used. You can also set the output sequence file from the Sampling Configuration dialog accessed from the Sample menu.

Help

This is the Help button. It opens a window holding a list of the sequencer topics for which help is available. You can copy and paste text from the help window into your sequence.

Loading sequencer files for sampling

The name of the output sequencer file to use during sampling is part of the sampling configuration. The file name, including the path to the folder containing it, must be less than 200 characters long. You set the file name either with the Current button, as described above, or in the Sampling Configuration dialog. When you start sampling, Spike2 searches for any output sequence file named in the sampling configuration.

Spike2 compiles output sequence files whenever you use them. If a file contains errors you are warned and the file is ignored.

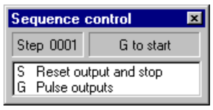
Getting started

The sequencer runs instructions in order unless told to branch. It can be re-routed during data acquisition by associating an instruction with a key on the keyboard. Each time the key is pressed or the associated sequencer control panel button is clicked or the script language `SampleKey()` command is used, the sequencer jumps to the associated instruction. Spike2 records keys pressed during sampling, so you have a record of where in the document you switched to a new portion of the sequence.

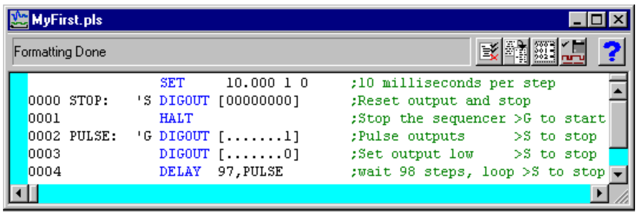
Here is a simple sequence that will pulse digital output bit 0 for 10 milliseconds once per second. You can start and stop the pulses with keys or by clicking buttons. You will find this in `Spike4\Sequence\MyFirst.pls` to save you typing it in.

```
            SET    10            ;10 milliseconds per step
         'S DIGOUT [00000000]    ;Reset output and Stop
            HALT                 ;Stop the sequencer    >G to start
PULSE:   'G DIGOUT [.......1]    ;Pulse outputs         >S to stop
            DIGOUT [.......0]    ;Set output low        >S to stop
            DELAY  97,PULSE      ;wait 98 steps, loop   >S to stop
```

Open `MyFirst.pls` and click the Check and make current sequence button at the upper right of the window (leave the mouse over each button for a second or so to see the descriptive text). Open the Sampling configuration dialog, and set a configuration with one event channel on port 0. To record the output pulses you must connect the digital output to the event input. This is easy with the micro1401 and Power1401 as you can connect Digital output 0 to Event input 0 on the front panel. For the 1401*plus* you need to connect the digital output pin 17 to the digital input pin 17. You do not need to make the connection to follow this description.



Click the Run now button in the sampling configuration, and then click the Start button. The sequencer control panel is now visible. It displays Step 0001 in the top left window and G to start to the right. If the control panel is docked there are two buttons labelled S and G, otherwise you can see the text S Reset output and Stop and G Pulse outputs. To start the output, click on the G or type `G` on the keyboard. While it runs, the display will change to Step 0004 (with occasional flickers) and S to stop. If you have connected the digital output to the input, you will see your pulses recorded, once per second. Click the S or type `S` on the keyboard to stop the output.



As this is our first sequence we will explain it in detail. Click the Format and add step numbers button at the top of the sequencer window. All the lines get a step number except the first. This is the step number displayed in the control panel. You can remove step numbers with the Format button. Lines that get a number are part of the sequence run by the 1401. The `SET` line tells the sequencer how fast to run, in this case 10 milliseconds per step, and is not part of the sequence run by the 1401. The `SET` directive is usually the first line in the sequence.

The next line (step 0000) is the first to run when you start sampling. The `'S` means *jump to this line every time the S key is pressed during sampling* (as long as the data window is the current window). To allow you a wide choice of keyboard jumps, lower case s and upper case S are treated as different keys.

The instruction on this line is `DIGOUT [00000000]`, which sets 8 digital outputs (bits 15 to 8 of the digital output) to the low state, nominally 0 Volts. The remainder of the line is a comment. Because there is a keyboard jump set for this line, the comment is also used as a label for the sequencer control panel.

The next line, (step 0001) holds the `HALT` instruction. This stops the sequencer and nothing will happen until the sequencer is told to jump to one of the steps labelled with a key code. The sequencer control panel displays the text to the right of the > for the current step. When the sequencer halts, it stays at step 0001, so you see the message G to start.

Steps 0002 and 0003 set bit 8 of the digital output high (nominally 5 Volts) and low again. Because the sequencer is running at 10 milliseconds per step, the pulse is 10 milliseconds wide. Step 0002 starts with `PULSE:` to indicate a label. This is so we can branch back here in the next step.

Step 0004 has the instruction `DELAY 97,PULSE` to make the sequence wait for 97 extra step times in addition to the step time that every instruction takes, and then branch to the instruction labelled `PULSE`. This instruction takes 980 milliseconds, and together with the 20 milliseconds taken by steps 0002 and 0003, this makes 1000 milliseconds for the loop. In place of 97 we could have written `s(1)-3`. The `s(1)` function returns the number of sequencer steps in 1 second.

The remainder of this chapter is organised as a reference manual for the sequencer instructions. You can find more information about the output sequencer in the *Spike2 Training Course* manual.

Instructions

These are the output sequencer instructions in Spike2 version 4. Instructions in brackets are obsolete and should be avoided in new sequences.

Digital (TTL compatible) input and output (see page 5-4)	DIGOUT	Write to digital output bits 15-8
	DIGLOW	Write to digital output bits 7-0 (does nothing in 1401*plus*)
	DIBNE,DIBEQ	Read digital input bits 7-0, copy to V56 test and branch
	DISBNE,DISBEQ	Test last read digital inputs (in V56) and branch
	WAIT	Wait for a pattern on the lower 8 digital inputs
	(DIGIN)	Read and test digital inputs
	(BZERO,BNZERO)	Branch as result of DIGIN test
DAC (waveform/voltage) outputs (see page 5-24)	DAC	Set DAC value (for DACs 0-7)
	(DACn)	Version 3 compatible, set DAC n (0-3) to a value
	ADDAC	Increment DAC by a value
	ADDACn	Version 3 compatible, increment DAC n (0-3) by a value
Sinusoidal waveform output (see page 5-25)	CSZ,DSZ	Set the first and second cosine output amplitude
	CSZINC,DSZINC	Increment/decrement the cosine output amplitude
	CRATE,DRATE	Set the cosine angular increment per step
	CRATEW,DRATEW	As C/DRATE, but waits for phase 0, clears new cycle flag
	CANGLE,DANGLE	Set the cosine angular position for the next step
	CWAIT,DWAIT	Branch until phase 0 reached in cosine output
	CRINC,DRINC	Increment/decrement the cosine increment per step
	CRINCW,DRINCW	As C/DRINC but waits for phase 0, clears new cycle flag
	CPHASE,DPHASE	Defines what phase 0 means (sine/cosine switch)
	COFF,DOFF	Offset for sinusoidal output
	CWCLR,DWCLR	Clear the new cycle flag
General control (see page 5-31)	DELAY	Do nothing for a set number of steps
	DBNZ	Decrement a variable and branch if not zero
	(LDCNTn,DBNZn)	Load counter 1 to 4 (V33-V36), decrement, branch if not zero
	Bxx	Compare variables and branch (xx = GT, GE, EQ, LE, LT, NE)
	CALL	Branch to a label, save return position
	CALLV,(CALLn)	Like CALL, but load a variable (counter 1-4) with a value
	RETURN	Branch to instruction after last CALL, CALLV or CALLn
	JUMP	Unconditional branch to a label
	HALT	Stops the sequencer and waits to be re-routed
	NOP	This does nothing for one step (NO OPeration)
Variable arithmetic (see page 5-33)	ADD	Add one variable to another
	ADDI,(ADDIL)	Add a constant value to a variable
	MOV	Copy one variable to another
	MOVI,(MOVIL)	Move a constant value into a variable
	MUL,MULI	Multiply two variables, multiply by a constant
	NEG	Move minus the value of a variable to another
	SUB	Subtract one variable from another
Access to data capture (see page 5-35)	REPORT,MARK	Simulate an external E1 pulse to record/set a digital marker
	CHAN	Get the latest waveform value or event count from a channel
	TICKS	Load a variable with the time in Spike2 time units
Randomisation (see page 5-37)	BRAND,(BRANDV)	Random branch with a probability
	MOVRND	Load a variable with a random number
	(LD1RAN)	Load counter 1 (V33) with a random number 1-256
Arbitrary waveform output (see page 5-39)	WAVEGO	Start or prepare arbitrary waveform output area
	WAVEBR	Test arbitrary waveform output and branch on the result
	WAVEST	Start or stop arbitrary waveform output

Instruction format

Blank text lines and lines with a semicolon as the first non-blank character, are ignored. Instructions are not case sensitive. Each instruction has the format:

```
num lab:      'key   code   arg1,arg2,…  ;Comment      >display
```

`num` An optional step number in the range 0 to 1022, for information only.

`lab:` An optional label, up to 8 characters long followed by a colon. The first character must be alphabetic (A-Z). Labels are not case sensitive. Labels may not be the same as instruction codes or variable names.

`'key` In this optional field, `key` is one alphanumeric (a-z, A-Z, 0-9) character. When this character is recorded as a keyboard marker during data capture, the sequencer jumps to this instruction. Each `key` can occur once. Upper and lower case are distinct. The `key` appears in the sequencer control panel.

`code` This field defines the instruction to be executed. It is not case sensitive.

`arg1,…` Instructions need up to 4 arguments and are separated by commas or spaces. These are described with the instructions.

`comment` The text after the semicolon is to remind you of the reason for the instruction. If a `key` is set, this comment also appears in the sequencer control panel.

`>display` When a sequence runs, text following a ">" in a comment is displayed in the sequencer control panel to indicate the current instruction.

Expressions

Many instructions allow the use of an expression in place of a constant value, indicated by `expr`. An expression is formed from numbers, round brackets `(` and `)`, the operators `+`, `-`, `*` and `/`, and sequencer expression functions.

The operators `*` and `/` (multiply and divide) have higher priority than `+` and `-` (add and subtract). This means that `1+2*3` is interpreted as `1+(2*3)` and not as `(1+2)*3`. Apart from this, evaluation is from left to right unless modified by brackets.

The output sequence compiler evaluates expressions using real numbers, so `3/2` has the value `1.5`. However, the result is often used as an integer, for example `DELAY expr` waits for an integral number of sequencer steps. From version 4.06 onwards, the result is rounded to an integer, so values in the range 3.5 to 4.49999… are treated as 4. Before version 4.06 the result was truncated, so 3.0 to 3.9999… was treated as 3.

Sequencer expression functions

These functions can be used as part of expressions to give you access to Spike2 clock and sequencer step timing and to convert between user units and DAC and ADC values.

`s(expr)`
`ms(expr)`
`us(expr)`
The number of sequencer steps in `expr` seconds, milliseconds and microseconds. For example, with a step size of 200 milliseconds, `s(1.1)` returns 5.5. This is often used with the `DELAY` instruction. Each instruction uses 1 step, so use `DELAY s(1)-1` for a delay of 1 second.

`sTick(expr)`
`msTick(expr)`
`usTick(expr)`
The number of Spike2 sample clock ticks in `expr` seconds, milliseconds and microseconds. The result is in the same units as returned by the `TICKS` instruction, so `TICKS V1,sTick(1)` sets `V1` to the value that `TICKS` will return 1 second later.

`Hz(expr)` The angle change in degrees per step for a cosine output of `expr` Hz. For a 2 Hz cosine, use `CRATE Hz(2)`. To slow the current rate down by 0.1 degrees per step use `CRINC Hz(-0.1)`. Use in `CRATE`, `DRATE`, `CRATEW`, `DRATEW`, `CRINC`, `DRINC`, `CRINCW` and `DRINCW` instructions.

`VHz(expr)` The same as `Hz()`, but the result is scaled into the 32-bit integer units used when a variable sets the rate. `MOVI V1,VHz(2)` followed by `CRATE V1` will set a 2 Hz rate.

`VAngle(expr)` Converts an angle in degrees into the internal angle format. The 32-bit integer range is 360 degrees. The result is `expr * 11930464.71`.

`VDAC16(expr)` Converts user DAC units so that the full DAC range spans the full range of a 16-bit integer. Use this with variables for `DACn` and `ADDACn`.

`VDAC32(expr)` Converts `expr` user DAC units into a 32-bit integer value such that the full DAC range spans the 32-bit integer value. Use this to load variables for use with the `DAC` and `ADDAC` instructions.

Used with care, the built-in functions allow you to write sequences that operate in the same way regardless of the sequencer step time or DAC scaling values.

Variables

You can use the 64 variables, `V1` to `V64`, in place of fixed values in many instructions. In the sequencer command descriptions, `Vn` indicates the use of a variable. Where a variable is an alternative to a fixed value expression we use `expr|Vn`. Variables hold 32-bit integer numbers that you can set and read with the `SampleSeqVar()` script command.

Some variables have specific uses: Variables `V57` through `V64` hold the last value written by the sequencer to DACs 0 through 7, `V56` holds the last bit pattern read from the digital inputs with the `DIBxx` or `DIGIN` instructions, `V33` through `V36` are used to emulate the counters for the `LDCNDn` and `DBNZn` instructions used in previous versions of Spike2.

VAR directive

You can assign each variable a name and an initial value with the `VAR` directive. Names must be assigned before they are used, usually at the start of the sequence. The syntax is:

```
VAR    Vn,name=expr          ;comment
```

`VAR` does not generate any instructions. It makes the symbol `name` equivalent to variable `Vn` and sets the initial value that is used when the sequence is loaded. Anywhere in the remainder of the sequence where `Vn` is acceptable, `name` can be used. `name` can be up to 8 characters, must start with an alphabetic character and can contain alphabetic characters and the digits 0 to 9. Variable names are not case sensitive. A variable name must not be the same as an instruction code or a label.

There is no need to specify a name or an initial value. If no initial value is set, a variable is initialised to 0 even if not included in a `VAR` statement. Spike2 automatically assigns `V56` the name `VDigIn` and variables `V57` through `V64` the names `VDAC0` through `VDAC7`. The following are all acceptable examples:

```
VAR    V1,Wait1=ms(100)     ;Set name and initial value
VAR    V2,UseMe             ;Set name only, so value is 0
VAR    V3=200               ;No name, initialise to a value
VAR    V4                   ;No name, initialised to 0
```

When a variable is used in place of a bit pattern in a digital input or output instruction, bits 15 to 8 and bits 7 to 0 have different uses. In the expressions that describe these operations we write `Vn(7-0)` and `Vn(15-8)` to describe which bits are used. `BAND` means bitwise binary `AND` (if both bits are 1, the output is 1, otherwise 0), `BXOR` means bitwise exclusive `OR` (if both bits are different the output is 1, otherwise 0).

When used in one of the cosine output angle instructions, the 32-bit variable range from -2147483648 to 2147483647 represents -180° up to +180°. The `VarValue` script in the `Scripts` folder calculates variable values for the digital and the cosine instructions.

Script access to variables

Scripts can set and read the variable values with the `SampleSeqVar()` script command. See *The Spike2 script language* manual for details. You can set initial values from the script as long as you set the values after you create the new data file, but before you start sampling. Values set in this way take precedence over values set by the `VAR` directive.

The SET directive

The `SET` directive controls the interval between sequencer clock ticks and the DAC output units. Directives are not part of the sequence and do not occupy a step. This directive is allowed to occur anywhere in your script. However, most scripts include it as the first non-comment line.

```
SET    msPerStep, DACscale, DACoffset
```

The `msPerStep` field sets the milliseconds per step in the range 0.010 to 3000. The table shows the minimum step times and timing resolution for each type of 1401. To ensure accurate timing, `msPerStep` must be an integral multiple of Resolution or Spike2 will not sample. This is checked each time Spike2 samples data.

	Power	Micro mk II	micro1401	1401*plus*
Minimum step (ms)	.010	.010	.050	.050
Resolution (ms)	.001	.001	.004, .006, .010	.004, .006, .010
Time used per step (μs)	<1	~1	<8	<10
Cosine penalty/step (μs)	<1	~1	~4	~10

The code in the 1401 to manage the sequence takes a few μs per step. If you use the cosine output, there is an extra time penalty per step per cosine output. Time used by the sequencer is time that is not available for sampling or arbitrary waveform output. To make best use of the capabilities of your 1401 you should set the slowest sequencer step rate that is fast enough for your purposes.

If there is no `SET` directive, the sequence runs at 10 milliseconds per step, `DACScale` is 1 and `DACOffset` is 0. The sequencer expression functions `s()`, `ms()`, `us()`, `Hz()` and `VHz()` use the `msPerStep` value. If you use these functions the `SET` directive must occur first in the sequence.

DAC scaling

The DACs in the 1401 are implemented so that the full range maps onto the full range of 16-bit signed integer numbers (-32768 to +32767). If you have 16-bit DACs a change of 1 changes the output by the smallest step possible. With 12-bit DACs, the smallest step is a change of 16. For most purposes, it is easier to work in units such as Volts or millivolts rather than these DAC units. However, the 1401 works in DAC units and if you set DAC values using variables, the variable values are based on DAC units. In the `SET` directive:

```
SET    msPerStep, DACscale, DACoffset
```

the `DACscale` and `DACoffset` fields define the conversion from user units into DAC units. The standard values of 1.0 for the scale and 0.0 for the offset make the full scale DAC values run from –5 to +4.99985. As most systems have ±5 Volt analogue systems, the standard scale and offset let you work with the DACs in Volts.

`DACScale` The number of user units that correspond to an output change of 6553.6 DAC units (1 Volt for a ±5 Volt system, 2 Volts for a ±10 Volt system). The standard value 1.0 is used if you omit `DACScale`.

`DACOffset` The user units that correspond to 0 DAC units (0 Volt output). The standard value 0.0 is used if you omit `DACOffset`.

Please remember that `DACScale` and `DACOffset` do not change the DAC outputs in any way. They are a convenience to allow you to enter values in units that are appropriate to your application. The sequencer expression functions `VDAC16()` and `VDAC32()` use `DACScale` and `DACOffset`, so they must come after the `SET` directive.

As an example, consider the case where the DAC outputs are driving a patch clamp amplifier where a change of 2.5 Volts into the amplifier causes a 200 mV potential at the cell and 0 Volts into the amplifier is 0 mV at the cell. For a ±5 Volt system, 2.5 Volts is 16384 DAC units, so `DACScale` is `200 * 6553.6/16384`, which is 80. `DACOffset` is 0 as and output of 0 produces 0 mV at the cell.

Sequencer instruction reference

Each instruction below is followed by an example. The examples show the preferred instruction format, however the system is flexible. For example, a comma should separate arguments, but a space is also accepted. The patterns used for digital ports should be enclosed by square brackets, however you may omit the brackets if you wish.

DIGOUT

The `DIGOUT` instruction changes the state of digital output bits 15-8 (see page 5-4 for the connections). The output changes occur at the next tick of the output sequencer clock.

```
DIGOUT  [pattern]|Vn,OptLB
```

`pattern` This determines the new output state. You can set, reset or invert each output bit, or leave a bit in the previous state. The pattern is 8 characters long, one for each bit, with bit 15 at the left and bit 8 at the right. The characters can be “`0`”, “`1`”, “`i`” or “`.`” standing for *clear*, *set*, *invert* or *leave alone*. You may omit the square brackets, however the Format command will insert them.

```
DIGOUT  [....001i]  ;clear bits 3 and 2, set 1, invert 0
DIGOUT  [.......i]  ;invert 0 again to produce a pulse
DIGOUT  V10         ;use variable V10 to set the pattern
```

`Vn` With a variable the new output is: `(old output BAND Vn(7-0)) BXOR Vn(15-8)`. The variable equivalent of `[....001i]` is `241+256*3`, and of `[.......i]` is `255+256*1`. You can use the `VarValue` script in the `Scripts` folder to calculate variable values.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

This example produces ten 1 millisecond pulses 100 milliseconds apart.

```
        SET     1           ;run at 1 ms per step
        MOVI    V1,10       ;V1 holds the number of pulses
LOOP:   DIGOUT  [.......1]  ;bit 0 high          >Pulsing
        DIGOUT  [.......0]  ;bit 0 low           >Pulsing
        DELAY   ms(100)-4   ;4 inst in the loop >Pulsing
        DBNZ    V1,LOOP     ;count down          >Pulsing
        HALT                ;finished            >Done
```

DIGLOW

The `DIGLOW` instruction changes the state of digital output bits 7-0 of the Power1401 and Micro1401 (see page 5-4 for the connections). It has no effect on a 1401*plus*. Unlike `DIGOUT`, the output changes occur immediately, they do not wait for the next sequencer clock tick. You can take advantage of this to change all 16 digital outputs almost simultaneously (within a few microseconds) by using `DIGOUT` followed by `DIGLOW`.

```
DIGLOW  [pattern]|Vn,OptLB
```

`pattern` This determines the new output state. The pattern is 8 characters long, one for each bit, with bit 7 at the left and bit 0 at the right. The characters can be “`0`”, “`1`”, “`i`” or “`.`” standing for *clear*, *set*, *invert* or *leave alone*. You may omit the square brackets, however the Format command will insert them.

```
DIGLOW  [....001i]  ;clear bits 3 and 2, set 1, invert 0
DIGLOW  [.......i]  ;invert 0 again to produce a pulse
DIGLOW  V10         ;use variable V10 to set the pattern
```

`Vn` With a variable the new output is: `(old output BAND Vn(7-0)) BXOR Vn(15-8)`. The variable equivalent of `[....001i]` is `241+256*3`, and of `[.......i]` is `255+256*1`. You can use the `VarValue` script in the `Scripts` folder to calculate variable values.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

DIBEQ, DIBNE

These instructions test digital input bits 7-0 against a pattern (see page 5-4 for digital input connections). `DIBEQ` branches on a match. `DIBNE` branches if it does not match. Both instructions copy bits 7-0 of the digital input to `V56` (`VDigIn`), for use by `DISBEQ` and `DISBNE`. If you use the template-matched signal in the 1401*plus,* input bits 7-0 may be programmed as outputs and cannot be used as inputs.

```
DIBNE   [pattern]|Vn,LB
DIBEQ   [pattern]|Vn,LB
```

`pattern` This is 8 characters, one for each input bit. The characters can be "`0`", "`1`" and "`.`" meaning match 0 (TTL low), match 1 (TTL high) or match anything. The bit order in the pattern is `[76543210]`. You may omit the square brackets, however the Format command inserts them.

`Vn` With a variable the result is: `(input BAND Vn(7-0)) BXOR Vn(15-8).` A result of 0 is a match, not zero is not a match.

`LB` The destination of the branch if the input was a match (`DIBEQ`) or not a match (`DIBNE`). This label must exist in the sequence.

This example waits for a pulse sequence in which the falling edges of two consecutive pulses are less than 2*`V1`+2 sequencer clock ticks apart. It waits for a falling edge, waits for a rising edge with a timeout and then waits for the next falling edge with a timeout. If timed out, we start again. If the input signal has high states less than three ticks wide, or low states less than 2 ticks wide, this example may miss them.

```
WHI:     DIBNE   [.......1],WHI  ;wait until high       >Wait high
SETTO:   MOVI    V1,24           ;set 50 step timeout   >Wait low
WLO:     DIBNE   [.......0],WLO  ;wait for falling      >Wait low
TOHI:    DIBEQ   [.......1],TOLO ;wait for high         >Wait high
         DBNZ    V1,TOHI         ;loop if not timed out >Wait high
         JUMP    WHI             ;timed out, restart    >Restart
TOLO:    DIBEQ   [.......0],GOTIT;jump if found events  >Wait low
         DBNZ    V1,TOLO         ;loop if not timed out >Wait low
         JUMP    SETTO           ;timed out, restart    >Restart
GOTIT:   ...                     ;here for 2 close pulses
```

DISBEQ, DISBNE

These instructions test digital input bits 7-0 read by the last `DIBEQ`, `DIBNE` or `WAIT` against a pattern. `DISBEQ` branches on a match. `DISBNE` branches if it does not match.

```
DISBNE  [pattern]|Vn,LB
DISBEQ  [pattern]|Vn,LB
```

`pattern` This is 8 characters, one for each input bit. The characters can be "`0`", "`1`" and "`.`" meaning match 0 (TTL low), match 1 (TTL high) or match anything. The bit order in the pattern is `[76543210]`. You may omit the square brackets, however the Format command inserts them.

`Vn` With a variable the result is: `(input BAND Vn(7-0)) BXOR Vn(15-8).` A result of 0 is a match, not zero is not a match.

`LB` The destination of the branch if the input was a match (`DISBEQ`) or not a match (`DISBNE`). This label must exist in the sequence.

This example shows a typical use of this instruction. We want to run a different part of the sequence for three trial types signalled by external equipment that writes the trial type to digital input bits 1 and 0; 00 means no trial, 01, 10 and 11 select trial types 1, 2 and 3.

```
TRWAIT:'W DIBEQ   [......00],TRWAIT ;Wait for trial >Wait...
          DISBEQ  [......01],TRIAL1
          DISBEQ  [......10],TRIAL2
          DISBEQ  [......11],TRIAL3
```

WAIT

The `WAIT` instruction causes the sequence to wait until bits 7-0 of the 1401 digital input match a pattern. The digital input port is sampled once every sequencer clock tick until the pattern is found, or until the sequence is sent elsewhere by a keyboard command. `WAIT` copies bits 7-0 of the digital input to `V56` (`VDigIn`) for use by `DISBEQ` and `DISBNE`. It is usually a good idea to have a display message explaining what you are waiting for.

If you use the template-matched signal in the 1401*plus,* input bits 7-0 may be set as outputs and cannot be used as inputs.

```
WAIT    [pattern]|Vn
```

`pattern` This is the input condition to match before the sequence can continue. It is 8 characters long, one for each input bit. The characters can be "`1`", "`0`" or "`.`" indicating that the input bit in that position must be a one, a zero or don't care. The bits are given in the order `[76543210]`. You may omit the square brackets round the pattern if you wish; however the Format command will insert them.

```
WAIT    [.......1]  ;wait for bit 0 set>Wait for bit 0
WAIT    [0......0]  ;wait for 7 and 0 clear>Wait 7&0 low
```

`Vn` With a variable the result is: `(input BAND Vn(7-0)) BXOR Vn(15-8)`. A result of 0 is required to continue.

This instruction is shorthand for:

```
HERE:   DIBNE   [pattern]|Vn,HERE
```

DIGIN, BZERO, BNZERO

These instructions are obsolete and are provided for backwards compatibility. `DIBNE` and `DIBEQ` are more efficient. `DIGIN` reads digital input bits 7-0 (see page 5-4 for the digital input connections), compares them with a pattern and saves the result. The result can be tested with the two branching instructions `BZERO` and `BNZERO`. The result of the comparisons is preserved until the next `DIGIN`. If you use the template-matched signal in the 1401*plus,* input bits 7-0 may be set as outputs and cannot be used as inputs.

```
DIGIN  [pattern]|Vn     ;Test the digital input state
BZERO  LB               ;branch to LB if result is zero
BNZERO LB               ;branch to LB if result non-zero
```

`pattern` This is 8 characters, one for each input bit in the order `[76543210]`. The characters can be "`.`", "`0`", "`1`" and "`c`". For "`0`" or "`1`", the result for that bit is 0 if the input bit was the same, or 1 if it was different. "`c`" means copy the input bit to the result (this is the same as "`0`"). The result for "`.`" is always zero. You may omit the square brackets round the pattern, however the Format command will insert them.

`Vn` With a variable the result is: `(input BAND Vn(7-0)) BXOR Vn(15-8)`. The variable equivalent of `[0.c0110.]` is `190 + 256*12`. The `VarValue` script in the `Scripts` folder calculates variable values equivalent to patterns.

`LB` The destination of the branch if the last `DIGIN` produced a zero (`BZERO`) or non-zero (`BNZERO`) result. This label must exist in the sequence.

```
Loop:   DIGIN  [0.c0110.]       ;assume input is 01101011
        BZERO  GoOn             ;result is 00100110 so no branch
        BNZERO Loop             ;This will branch
GoOn:   ...
```

DAC outputs

The output sequencer supports up to 8 DAC (Digital to Analogue Converter) outputs. The 1401*plus* and Power1401 have four DACs and the micro1401 has two. However, the Power1401 can be expanded with up to 8 DAC outputs. The last value written to DACs 0-7 is stored in variables `V57`-`V64` (you can also refer to these variables as `VDAC0`-`VDAC7`). The values are stored as 32-bit numbers with the full 32-bit range corresponding to the full range of the DAC. This high resolution allows us to ramp the DACs smoothly.

Values written to the DACs are expressed in units of your choice. The `SET` directive determines the conversion between the numbers you supply and the DAC outputs. The standard settings for a system with ±5 Volts DACs is to set the DAC outputs in Volts.

If you write to a DAC that does not exist, the variable associated with the DAC is set as if the DAC were present. Output to 1401*plus* DACs 4 –7 is mapped back to DACs 0-3. For the micro1401 and Power1401, output to DACs that do not exist has no effect.

The Power1401 DAC 2 and 3 outputs are on the rear panel 37-way Cannon D type Analogue Expansion connector. DAC 2 is on pin 36 and DAC 3 is pin 37. Suitable grounds are on the adjacent pins 18 and 19. If you have a Power1401 top-box with additional DACs on the front panel, the two DAC output on the rear are mapped to be the two highest numbered DACs. For example, if you have a 2709 Spike2 Top Box, DACs 2 and 3 are available as BNC connections on the front panel, and the rear panel DACs become DAC 4 on pin 36 and DAC 5 on pin 37.

DAC, ADDAC

The `DAC` instruction can write a value to any of the 8 possible DAC outputs. The `ADDAC` instruction adds a value to the DAC output. The output value changes immediately unless the DAC is in use by the arbitrary waveform output, in which case the result is undefined.

```
DAC     n,expr|Vn,OptLB
ADDAC   n,expr|Vn,OptLb
```

`n` The DAC number, in the range 0-7. Variable 57+n is set to the new DAC value such that the full DAC range spans the full range of the 32-bit variable.

`expr` The value to write to the DAC or the change in the DAC value. The units of this value depend on the SET directive; the standard units are Volts. It is an error to give a value that exceeds the DAC output range.

`Vn` When a variable is used, the full range of the 32-bit variable corresponds to the full range of the DAC. You can use the `VDAC32()` function to load a variable using user-defined DAC units.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

This example sets DAC 2 to 0 Volts, then ramps it to 4.99 Volts in 1 second using steps of 0.01 Volts. The example also shows how to do the same ramp using variables:

```
            SET   1,1,0           ;1 ms per step, DAC scaled to Volts
         'R DAC   2,0             ;Ramp 0 to 5      >Ramping
            MOVI  V1,499          ;499 steps        >Ramping
RAMP:       ADDAC 2,0.01          ;0.01V increment  >Ramping
            DBNZ  V1,RAMP         ;count increments >Ramping
            HALT                  ;task finished    >Done

         'V MOVI  V3,VDAC32(0)    ;Use variables    >Ramping
            MOVI  V2,VDAC32(0.01) ;increment in V2  >Ramping
            MOVI  V1,499          ;499 steps        >Ramping
            DAC   2,V3            ;set initial value>Ramping
RAMP:       ADDAC 2,V2            ;add increment    >Ramping
            DBNZ  V1,RAMP         ;count increments >Ramping
            HALT                  ;task finished    >Done
```

It is a property of the signed integer numbers we use that if you add 1 to the maximum possible positive number, the result is the minimum possible negative number. If you use `ADDAC` repeatedly to add the same value, eventually you will run off the end of the DAC range and come back in at the other end.

DAC units run from -32768 to +32767. In a ±5 Volt system with 16-bit DACs, this is -5.0000 to +4.99985 Volts. The DAC unit value for +5 Volts is +32768, but this number does not exist in 16-bit signed integers and wraps around to -32878. Because it often happens that users want to set the DAC to full scale, for the `DAC` command used with `expr` (not with `Vn`), we detect a request to set the DAC to +32768 units, and we limit it to 32767.

The output value changes when the instruction is executed, not at the next sequencer clock tick. This means that the changes are not exactly clocked and may have a time jitter of a few µs.

DACn, ADDACn

These commands are provided for backwards compatibility with older versions of Spike2. The `DACn` and `ADDACn` commands (with `n = 0`, `1`, `2` or `3`) set and change the 1401 DAC outputs. The `expr` parameter is the new DAC output level, or change in level. Variable `57+n` is set to the new DAC value such that the full DAC range spans the full range of the 32-bit variable.

```
DACn      expr|Vn,OptLB
ADDACn    expr|Vn,OptLB
```

`expr` The value to assign to DAC n (`DACn` instruction) or to add to the output level (`ADDACn` instruction). The units of the DAC values are usually Volts, but can be changed by setting a scale factor with `SET`.

`Vn` With a variable, the values -32768 to 32767 correspond to the full range of the DAC. You can use the `VDAC16()` function to load a variable using user-defined DAC units. You can also use the `VarValue` script in the `Scripts` folder to calculate variable values. The value saved in `V57+n` is `Vn*65536`.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

The important difference between these commands and the `DAC` and `ADDAC` commands is for the case where a variable is used. With a variable, the bottom 16-bits of the variable are written to the DAC. In the case of the `DAC` and `ADDAC` commands, the upper 16 bits of the variable are written to the DAC.

Cosine output control instructions

The sequencer can output cosine waveforms of variable amplitude and frequency through DACs 1 (`Cxxx` instructions) and 0 (`Dxxx` instructions) with the micro1401 and the Power1401 and DACs 3 and 2 with the 1401*plus*. When enabled, the cosine value is computed and output every step. The time penalty per step per DAC is around 10 µs for the 1401*plus*, 4 µs for the micro1401 and less than 1 µs for the Power1401. The output is:

output in Volts = 5 A $Cos(\theta+\phi)$ + offset

where A is an amplitude scaling factor in the range 0 to 1
- θ is a phase angle in the range 0° to 360° that changes each step
- ϕ is a fixed phase angle in the range -360° to 360°
- offset A voltage offset defined by `COFF` or `DOFF`

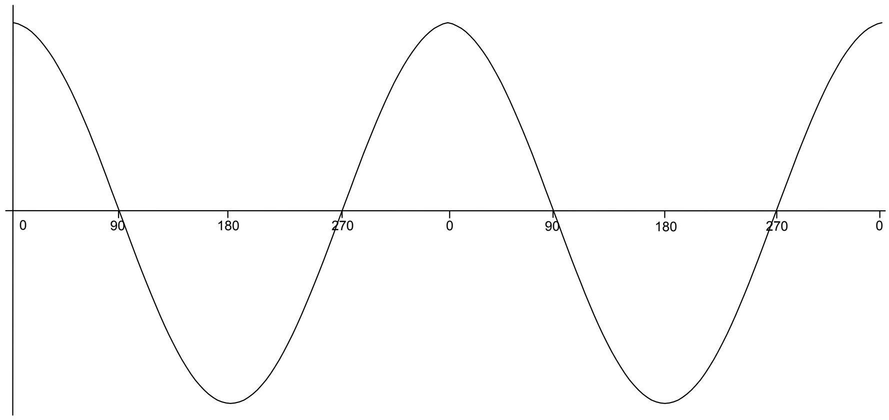
θ changes every step by dθ. A cycle of the cosine takes 360/dθ steps. You can change the angle increment immediately, or you can delay the change until the next time θ passes through 0°. You can set dθ in the range 0° up to 360° to an accuracy of about 0.0000001°. With the sequencer running at 1 kHz, you can output frequencies up to 500

Hz with a frequency resolution of around 0.00012 Hz. Ideally the output would be passed through a low pass filter with a corner frequency at one half of the sequencer step rate to smooth out the steps in the cosine wave.

By adjusting ϕ you control the output cosine phase where θ passes through zero. Unless you set the value (`CPHASE, DPHASE`), it is zero and the zero crossing occurs at the peak of the sinusoid. To have the output rising through 0, set the phase to –90.

Each time θ passes through zero a *new cycle* flag sets. The `CRATEW`, `CRINCW`, `CWAIT` and `CWCLR` instructions clear the flag for sinusoid C. The corresponding instructions for sinusoid D clear the sinusoid D flag.

Output as a function of $\theta+\phi$



In the descriptions, DAC numbers in brackets are for the 1401*plus*. The cosine output is implemented through these instructions:

CSZ, DSZ

These instructions set the waveform amplitude. If a wave is playing, the amplitude changes at the next sequencer step. The amplitude is set to 1.0 when sampling starts.

```
CSZ     expr|Vn,OptLB ;set DAC 1(3) scale
DSZ     expr|Vn,OptLB ;set DAC 0(2) scale
```

`expr` The cosine amplitude in the range 0 to 1. A cosine with amplitude 1.0 uses the full DAC range.

`Vn` Use variable values of 0 to 32768 for amplitude of 0.0 to 1.0, values outside the range 0 to 32768 cause undefined results.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

CSZINC, DSZINC

These instructions change the waveform amplitude. The change is added to the current amplitude. If the result exceeds 1.0, it is set to 1.0. If it is less than 0, the result is 0.

```
CSZINC  expr|Vn,OptLB ;alter DAC 1(3) scale
DSZINC  expr|Vn,OptLB ;alter DAC 0(2) scale
```

`expr` The change in the waveform scale in the range -1 to 1.

`Vn` A variable value of 32768 is a scale change of 1.0, -16384 is -0.5 and so on.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

You can gradually increase or decrease the wave amplitude. For example, the following increases the amplitude from zero to full scale (we assume that the waveform is playing):

```
        CSZ     0.0       ;start with zero size
        MOVI    V1,100    ;proceed in 1% increments
loop:   CSZINC  0.01      ;a 1% increase
        DELAY   ms(100)-2 ;show some of the waveform at this size
        DBNZ    V1,loop   ;loop 100 times
```

CRATE, DRATE

These set the angle increment in degrees per step, which sets the cosine frequency. You can stop the cosine output with a rate of 0. Stopping all cosine output removes the time penalty per step for calculating the cosine. Any non-zero value starts the cosine output.

```
        CRATE   expr|Vn,OptLB ;set DAC 3 cosine speed
        DRATE   expr|Vn,OptLB ;set DAC 2 cosine speed
```

`expr` The angle increment per step in the range 0.000 up to 180 degrees. The `Hz()` function calculates the increment required for a frequency.

`Vn` For a variable, the value 11930465 is an increment of 1 degree. The `VHz()` function can be used to set a variable value equivalent to an angle in degrees.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

`DRATE 0` stops DAC 0(2) output, but leaves DAC 1(3) running. `CRATE 0` stops all cyclical output unless DAC 0(2) is running, in which case it sets DAC 1(3) rate to 0, but leaves both cyclical outputs enabled. To stop all cosine output, use `DRATE 0` then `CRATE 0`. The angle increment (and change of frequency) applies to the next step.

This example starts cosine output at 10 Hz, runs for 10 seconds, and then stops it. This is then repeated using a variable to produce the same effect:

```
           SET 1,1,0          ;1 ms per step
        'C CRATE   Hz(10)     ;start output at 10 Hz
           DELAY   S(10)-1    ;delay for 10 seconds >Sine wave
X:      'S CRATE   0          ;stop output
           HALT                                    >Stopped
        'V MOVI    V1,VHz(10) ;set V1 equivalent of 10 Hz
           CRATE   V1         ;start at 10 Hz
           DELAY   S(10)-1,X  ;delay then goto exit >Sine wave
```

CRATEW, DRATEW

These instructions perform the same function as `CRATE` and `DRATE`, except that the change is postponed until the next time θ passes through 0 degrees. Unlike `CRATE` and `DRATE`, these instructions cannot start output; a sinusoid must already be running to pass phase 0. They can stop output, but they do not remove the overhead for using cosine output. This instruction clears the new cycle flag (see `CWAIT` and `DWAIT`).

```
        CRATEW   expr|Vn    ;delayed change of DAC 1(3) rate
        DRATEW   expr|Vn    ;delayed change of DAC 0(2) rate
```

`expr` The angle increment in the range 0.000 to 180 degrees. The `Hz()` built-in function calculates the increment required for a frequency.

`Vn` For a variable, the value 11930465 is an increment of 1 degree. The `VHz()` function can be used to set a variable value equivalent to an angle in degrees.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

This example starts cosine output at 10 Hz, runs for 1 cycle, changes to 11 Hz for one cycle, then stops:

```
        SET 1,1,0          ;1 ms per step
        CANGLE  0          ;make sure we are at phase 0
        CRATE   Hz(10)     ;start output at 10 Hz
```

```
         CRATEW  Hz(11)     ;request 11 Hz next time around
CYCLE10: CWAIT   CYCLE10    ;wait for the cycle>10Hz
CYCLE11: CWAIT   CYCLE11    ;wait for the cycle>11Hz
         CRATE   0          ;stop output
```

CANGLE, DANGLE

These change the cosine angular position. They take effect on the next instruction when the angle increment is added to the value set by this instruction and the result is output.

```
CANGLE  expr|Vn,OptLB  ;set phase angle for DAC 1(3)
DANGLE  expr|Vn,OptLB  ;set phase angle for DAC 0(2)
```

`expr` The phase angle to set in the range -360 up to +360.

`Vn` For a variable, the value 11930465 is a phase of 1 degree (to be precise, 4294967296/360 is a phase of 1 degree). You can use the `VAngle()` function to convert degrees into a suitable value for a variable.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

This example sets the phase angle to –90 degrees directly, and by using a variable. There is no need to use the `VAngle()` function; we could have set `V1` to –1073741824. However, `VAngle(-90)` is much easier to understand.

```
CANGLE  -90       ;set the angle directly
MOVI    V1,VAngle(-90)
CANGLE  V1        ;set using a variable
```

CPHASE, DPHASE

These change the relative phase of the cosine output for the next cosine output. They are usually used to change the output from a cosine (maximum value at phase zero) to sine (rising through zero at phase zero).

```
CPHASE  expr|Vn,OptLB  ;set relative phase for DAC 1(3)
DPHASE  expr|Vn,OptLB  ;set relative phase for DAC 0(2)
```

`expr` The relative phase angle to set in the range -360 up to +360. The relative phase is set to 0 when sampling starts. Set –90 for sinusoidal output.

`Vn` For a variable, the value 11930465 is a phase of 1 degree (to be precise, 4294967296/360 is a phase of 1 degree). You can use the `VAngle()` function to convert degrees into a suitable value for a variable.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

This example plays a 1 Hz sinusoidal output (assuming that the output is not running).

```
CPHASE  -90       ;set the angle directly
CANGLE  0         ;prepare to start as a sine wave
CRATE   Hz(1)     ;start the sinusoid
```

COFF, DOFF

These change the cosine output voltage offset for the next cosine output.

```
COFF    expr|Vn,OptLB  ;set voltage offset for DAC 1(3)
DOFF    expr|Vn,OptLB  ;set voltage offset for DAC 0(2)
```

`expr` The offset value for sinusoidal output. The units of this value depend on the SET directive; the standard units are Volts. It is an error to give a value that exceeds the DAC output range.

`Vn` When a variable is used, the full range of the 32-bit variable corresponds to the full range of the DAC. You can use the `VDAC32()` function to load a variable using user-defined DAC units.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

This example ramps DAC 0 from 0 to 1 Volt, the runs 5 cycles of a sine wave at 1 Hz, and finally ramps the data back to 0 Volts. If you have a 1401*plus* you should change all the references to DAC 0 to DAC 2.

```
            SET    1 1 0          ;1 millisecond per step
            DAC    0,0            ;(DAC 2 for 1401plus)
            DOFF   1.0            ;set DAC 0 (2) offset
            DSZ    0.2            ;1 V sinusoid
            DPHASE -90            ;Prepare sinusoid
            DANGLE 0              ;set start point
            MOVI   V1,500         ;ramp will take 1 s
RAMPUP:     ADDAC  0,0.002        ;(DAC 2 for plus)   >ramp up
            DBNZ   V1,rampup      ;count increments   >ramp up
            DRATE  HZ(1)          ;start sinusoid
            DELAY  S(4.9)         ;Sinusoid           >Sine
            DRATEW 0              ;stop at cycle end
END:        DWAIT  END            ;wait for end       >Wait end
            DRATE  0              ;stop now
            DAC    0,1.0          ;(DAC 2 for plus)
            MOVI   V1,500         ;ramp will take 1 s
RAMPDN:     ADDAC  0,-0.002       ;(DAC 2 for plus)   >ramp down
            DBNZ   V1,rampdn      ;count increments   >ramp down
            HALT
```

CWAIT, DWAIT

Each time the phase angle of a cosine passes through 0°, a new cycle flag sets. There is a separate flag for each sinusoid. This flag is cleared by `CWCLR` (`DWCLR`), `CRATEW` (`DRATEW`), `CRINCW` (`DRINCW`) and when tested by `CWAIT` (`DWAIT`).

```
CWAIT   LB        ;branch until DAC 1(3) passes phase 0
DWAIT   LB        ;branch until DAC 0(2) passes phase 0
```

`LB` A label to branch to if the new cycle flag is not set. If the flag is set, the sequencer will clear the flag and proceed to the next instruction.

These instructions can produce a pulse at the start (or at least a known time after) the start of each waveform cycle. The following sequence outputs 4 cycles of waveform at different rates on DAC 1(3), and changes the digital outputs for each cycle.

```
         CSZ     1.0        ;make sure full size
         CANGLE  0.0        ;make sure we start at phase 0
         CRATE   1.0        ;1 degree per step to start with
         DIGOUT  [00000001] ;so outside world knows
         CRATEW  1.2        ;next cycle faster, clear cycle flag
w1:      CWAIT   w1         ;wait for cycle >1 degree cycle
         DIGOUT  [00000010] ;announce another cycle
         CRATEW  1.4        ;next cycle a bit faster
w2:      CWAIT   w2         ;wait for cycle >1.2 degree cycle
         DIGOUT  [00000011] ;yet another one
         CRATEW  1.6        ;last cycle a bit faster
w3:      CWAIT   w3         ;wait for cycle >1.4 degree cycle
         DIGOUT  [00000100] ;last cycle number
w4:      CWAIT   w4         ;wait for end   >1.6 degree cycle
         CRATE   0.0        ;stop waveform
```

CRINC, CRINCW, DRINC, DRINCW

These instructions behave like `CRATE` and `CRATEW`, `DRATE` and `DRATEW` except that they *change* the output rate (angle increment per step) by their argument rather than set it. `CRINCW` and `DRINCW` clear the new cycle flags for each sinusoid.

```
CRINC   expr|Vn,OptLB ;change DAC 1(3) rate now
CRINCW  expr|Vn,OptLB ;change DAC 1(3) rate at phase 0
```

```
         DRINC   expr|Vn,OptLB ;change DAC 0(2) rate now
         DRINCW  expr|Vn,OptLB ;change DAC 0(2) rate at phase 0
```

`expr` The change in the angle increment per step. You can use the built-in `Hz()` function to express the change as a frequency.

`Vn` For a variable, the value 11930465 is a change of 1 degree. You can use the `VarValue` script in the `Scripts` folder to calculate variable values.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

This example starts cosine output at 10 Hz and lets you adjust it from the keyboard.

```
         SET    1,1,0     ;1 ms per step
         CRATE  Hz(10)    ;start output at 10 Hz
wt:      JUMP   wt        ;HALT stops all output>P=+1Hz, M=-1Hz
      'P CRINC  Hz(1),wt  ;1 Hz faster
      'M CRINC  Hz(-1),wt ;1 Hz slower
```

These instructions can be used to produce waveforms that change gradually in frequency. The following code generates a linear speed increase every two steps on DAC 1(3):

```
         CSZ     1.0      ;make sure full size
         CANGLE  0.0      ;make sure we start at phase 0
         CRATE   1.0      ;1 degree per step to start with
         LDCNT1  900      ;in 900 steps of...
loop:    CRINC   0.01     ;...1/100 degrees to...
         DBNZ1   loop     ;...10 degrees per step
```

The next example produces 90 cycles, each increasing by 0.1 degrees per step per cycle.

```
         CSZ     1.0      ;make sure full size
         CANGLE  0.0      ;make sure we start at phase 0
         CRATE   1.0      ;1 degree per step to start with
         LDCNT1  90       ;in 90 steps of...
loop:    CRINCW  0.1      ;...1/10 degrees to...
wait:    CWAIT   wait     ;...(wait for next cycle)...
         DBNZ1   loop     ;...10 degrees per step
```

CWCLR, DWCLR

These instructions clear the cosine output new cycle flag. If you have been running for several cycles and you want to stop the next time phase 0 is crossed use these instructions immediately before using `CWAIT` or `DWAIT`.

```
         CWCLR  OptLB          ;clear C sinusoid new cycle flag
         DWCLR  OptLB          ;clear D sinusoid new cycle flag
```

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

This example starts a sinusoid and then stops at the next phase 0 crossing after the user requests a stop. Because the sinusoid passes phase 0 in the `CWAIT` instruction and does another step in the `CRATE 0` instruction, we offset the phase by 2 steps. However, this would cause the start of the sinusoid to be 2 steps wrong, so we change the start angle to match.

```
     'G CPHASE  -2*Hz(2)       ; compenstate for ending
        CANGLE  2*Hz(2)        ; so we start in correct place
        CRATE   Hz(2)          ; 2 Hz output
HERE:   JUMP    HERE           ; output is running >Running
     'S CWCLR                  ; Stop output
WT:     CWAIT   WT             ; wait for cycle end>Waiting
        CRATE   0,HERE         ; stop and then idle
```

General control

These instructions do not change any outputs or read data from any inputs. They provide the framework of loops, branches and delays used by the other instructions.

DELAY

The `DELAY` instruction occupies one clock tick plus the number of extra ticks set by the argument. It produces simple delays of 1 to more than 4,000,000,000 sequencer steps.

```
DELAY   expr|Vn,OptLB
```

`expr` The number of extra sequencer clock ticks to delay in the range 0 to 4294967295. It is often more convenient to use the `s()`, `ms()` or `us()` built-in functions to convert a delay in seconds, milliseconds or microseconds into sequencer steps.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

This example uses display messages to tell the user what the sequence is doing.

```
SET     1.00,1,0 ;run with 1 millisecond clock ticks
DELAY   2999     ;wait 2999+1 milliseconds>3 second delay
DELAY   s(3)-1   ;3 seconds -1 tick delay >3 second delay
DELAY   V1       ;wait V1+1 milliseconds  >variable delay
```

DBNZ

`DBNZ` (`D`ecrement and `B`ranch if `N`ot `Z`ero) subtracts 1 from a variable and branches to a label unless the counter is zero. It is used for building loops.

```
DBNZ   Vn,LB
```

`Vn` The variable to decrement and test for zero.

`LB` Instruction to go to next if the result of the decrement is not zero.

`DBNZ` is often used with `MOVI` to set up loops, for example:

```
         MOVI    V2,1000    ;set times to loop
WT:      DIGOUT  [00000000] ;set all digital outputs low
         DIGOUT  [11111111] ;set them all high
         DBNZ    V2,WT      ;loop 1000 times
```

LDCNTn, DBNZn

These obsolete instructions use variables V33 to V36 as counters 1 to 4. New sequences should use the `MOVI` and `DBNZ` instructions that can use any variable as a counter. The `LDCNTn` instruction loads one of the four counters. `DBNZn` (`D`ecrement and `B`ranch if `N`ot `Z`ero) decrements a counter and branches to a label while the counter is not zero.

```
LDCNTn  expr|Vn
DBNZn   LB
```

`n` The counter number to load or decrement in the range 1 to 4. Counters 1 to 4 are emulated by variables `V33` to `V36`.

`expr` The 32-bit integer value to load into the counter.

`Vn` When a variable is used the counter is set to the 32-bit variable.

`LB` Instruction to go to next if the result of the decrement is not zero

CALL, CALLV, CALLn, RETURN

These instructions run a labelled part of a sequence and return. `CALL`, `CALLV` and `CALLn` save the next step number to a return stack and jump to the labelled instruction. The `RETURN` instruction removes the top step number from the return stack and jumps to it. `CALLV` also sets a variable to a constant. `CALLn` (`n`=1 to 4) is obsolete and sets the value of the variables that emulate the four counters `V33` to `V36`.

```
CALL   LB             ; use LB as a subroutine
CALLV  LB,Vn,expr     ; Vn = expr, then call LB
CALLn  LB,expr        ; V32+n = expr (n=1, 2, 3 or 4)
RETURN                ; return to step after last CALL
```

`LB` The next instruction to run. This section should end with a `RETURN`.

`Vn` `CALLV` copies the value of `expr` to this variable. `CALL1` sets `V33`, `CALL2` sets `V34` `CALL3` sets `V35` and `CALL4` sets `V36`.

`expr` A 32-bit integer constant that is copied to a variable. For `CALLn` only, if `expr` is 0, the variable is set to 256 to be compatible with previous versions.

You can use the `CALL` instruction inside a `CALL`ed subroutine. This is known as a *nested* `CALL`. If you call a subroutine from inside itself, this is known as a *recursive* `CALL`. The return stack has room for 64 return addresses. If you use more than this, the oldest return address is overwritten, so your sequence will not behave as you expect. Nesting calls this deep is usually the sign of an error (perhaps an unintentional recursive call).

In this example, we want to produce different pulse widths from DAC0. The sequence is written to be independent of the sequencer rate as far as is possible (the rate must be high enough so that the widths you want are possible). In this case the rate is set to 5 Khz. The example sets DAC0 to zero, then pulses for 20 milliseconds twice, once using `CALL` and once using `CALLV`. Then after a delay, there is a 50 millisecond pulse.

```
        SET    0.2,1,0        ; Run at 5 kHz, normal DAC scale
        DAC    0,0            ; make sure DAC0 is zero
        MOVI   V3,ms(20)-2    ; these two instructions...
        CALL   PUL            ; ...have the same effect as...
        CALLV  PUL,V3,ms(20)-2; ...this one. 20 ms pulse
        DELAY  s(1)-1         ; wait 1 second, then...
        CALLV  PUL,V3,ms(50)-2; ...a 50 ms pulse
        HALT                  ; So we don't fall into PUL routine
PUL:    DAC    0,1            ; set DAC value
        DELAY  V3             ; wait for time set
        DAC    0,0            ; set DAC back to zero
        RETURN                ; back to the caller
```

`CALL`/`CALLV` and `RETURN` let you reuse a block of instructions. This can make sequences much easier to understand and maintain. The disadvantage is the additional steps for the `CALL` and `RETURN`. If you need to set a variable, use `CALLV` and there is only the overhead of the `RETURN` instruction.

JUMP The `JUMP` instruction transfers control unconditionally to the instruction at the label. Many instructions allow the use of an optional label to set the next instruction, so you can often avoid the need for this instruction.

```
JUMP   LB             ; Jump to label
```

HALT The `HALT` instruction stops the output sequence and removes all overhead associated with it. It does not stop the sequencer clock, which continues to run. Any cosine output will stop, but will restart when the sequence restarts. To restart the sequencer, press a key associated with a sequence step or click a key in the sequencer control panel. If you associate a display string with this instruction, it appears in the sequencer control panel.

```
HALT                  >Press X when ready
```

NOP The `NOP` instruction (`NO OP`eration) does nothing except use up one sequencer clock tick. It can be thought of as the equivalent of `DELAY 0`.

Variable arithmetic

These instructions let you perform basic mathematical functions while the sequence is running. We have not included division as this takes too long in the 1401*plus*. However, we do allow division by powers of two. You can also compare variables and branch on the result.

Compare variable

These instructions compare two variables or a variable and a 32-bit expression and branch on the result of the comparison. All comparisons are as signed 32-bit integers.

```
Bxx      Vn,Vm,LB       ;compare with a variable
Bxx      Vn,expr,LB     ;compare with a constant
```

`xx` This is the branch condition. The `xx` stands for: `GT`=Greater Than, `GE`=Greater or Equal, `EQ`=Equal, `LE`=Less than or Equal, `LT`=Less Than, `NE`=Not Equal.

`Vn` The variable to compare with the next argument.

`Vm` A variable to compare `Vn` with.

`expr` A 32-bit integer constant to compare `Vn` with.

In this example, the sequence collects the latest data value from one channel 1 (assumed to be a waveform channel), waits for it to be in a defined range for 1 second, then outputs a pulse to a digital output bit.

```
START:  CHAN    V1,1            ; get channel 1 data
        BGT     V1,4000,START   ; if above upper limit, wait
        BLT     V1,0,START      ; if too low, wait
IN:     MOVI    V2,S(1)/4       ; timeout, 4 instructions/loop
INLOOP: CHAN    V1,1            ; to check if still inside
        BGT     V1,4000,START   ; if above upper limit, wait
        BLT     V1,0,START      ; if too low, wait
        DBNZ    V2,INLOOP       ; see if done yet
REWARD: DIGOUT  [.......1]      ; Task done OK
        DELAY   S(1)            ; leave bit set for 1 second
        DIGOUT  [.......0]      ; clear done bit
        ...                     ; next task...
```

We want the data to be in range for one second. There are 4 instructions in the loop that tests this, so we set to the loop to run for the number of steps in a second divided by 4. For this to work correctly, the sequencer must be running fast enough so that 4 steps are no longer than the sample interval for the waveform channel.

MOVI

This instruction moves an integer constant into a variable. `MOVIL` is an obsolete instruction that does exactly the same thing. The syntax is:

```
MOVI    Vn,expr,OptLB    ; Vn = expr
```

`Vn` A variable to hold the value of `expr`.

`expr` An expression that is evaluated as a 32-bit integer.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

`MOVI` is not the same as the `VAR` directive. The `VAR` directive sets the value of a variable when the sequence is copied to the 1401 and does not occupy a step. The `MOVI` instruction is part of the sequence and set the value of the variable each time the instruction is used.

MOV, NEG

The `MOV` instruction sets a variable to the value of another with the option of adding a 32-bit number and dividing by a power of two). The `NEG` instruction is identical to `MOV` except that the source variable is negated first. The syntax is:

```
MOV     Va,Vb,expr,shift   ; Va = (Vb + expr) >> shift
NEG     Va,Vb,expr,shift   ; Va = (-Vb + expr) >> shift
```

`Va` A variable to hold the result. It can be the same as `Vb`.

`Vb` A variable used to calculate the result. It is not changed unless it is the same variable as `Va`.

`expr` An optional expression that is evaluated as a 32-bit integer. If this argument is omitted, it is treated as 0.

`shift` An optional argument in the range 0 to 31, set to 0 if omitted, that sets the number of times to divide the result by 2.

The following examples assume that `V3` holds 1000:

```
VAR     V6,Result
MOV     V1,V3              ; set V1 to 1000
NEG     V1,V3              ; set V1 to -1000
MOV     V1,V3,-8           ; set V1 to 992
NEG     Result,V3,0,4      ; set V6 to -63
MOV     Result,V3,4,1      ; set V6 to 502
```

ADDI This instruction adds a 32-bit integer constant to a variable. `ADDIL` is an obsolete instruction that does exactly the same. There is no `SUBI` as you can add a negative number. The syntax is:

```
ADDI    Vn,expr,OptLB      ; Vn = Vn + expr
```

`Vn` A variable to hold the result of `Vn` + `expr`.

`expr` An expression that is evaluated as a 32-bit integer.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

The following examples assume that `V1` holds -1000:

```
VAR      V1,Result=-1000
ADDI    Result,1000        ; set V1 to 0
ADDI    V1,-4000           ; set V1 to -5000
```

ADD, SUB The `ADD` instruction adds one variable to another. The `SUB` instruction subtracts one variable from another. In both cases you can optionally add a 32-bit integer constant and optionally divide the result by a power of two. The syntax is:

```
ADD     Va,Vb,expr,shift ; Va = (Va + Vb + expr) >> shift
SUB     Va,Vb,expr,shift ; Va = (Va - Vb + expr) >> shift
```

`Va` A variable to hold the result. It can be the same as `Vb`.

`Vb` A variable to add or subtract.

`expr` An optional expression evaluated as a 32-bit integer. If omitted, 0 is used.

`shift` An optional argument in the range 0 to 31, set to 0 if omitted, that sets the number of times to divide the result by 2.

The following examples assume that `V1` holds -1000, `V3` holds 1000, `V6` holds 100:

```
VAR     V6,Result=100
ADD     V1,V3            ; V1 = 0 (-1000 + 1000 + 0)
SUB     V1,V3            ; V1 = -1000 (0 - 1000 + 0)
ADD     V1,V3,-8         ; V1 = -8 (-1000 + 1000 - 8)
SUB     Result,V3,0,2    ; V6 = -225 (100 - 1000 + 0)/4
ADD     Result,V3,4,1    ; V6 = 389 (-225 + 1000 + 4)/2
```

MUL, MULI `MUL` multiplies a variable by another variable, then optionally adds a 32-bit integer constant and divides the result by a power of two. `MULI` multiplies a variable by a 32-bit integer constant and divides the result by a power of 2.

```
MUL     Va,Vb,expr,shift ; Va = ((Va*Vb)+expr) >> shift
MULI    Va,expr,shift    ; Va = (Va*expr) >> shift
```

`Va` A variable to hold the result. It can be the same as `Vb`.

`Vb` A variable used to calculate the result.

`expr` An expression that is evaluated as a 32-bit integer. It is optional for `MUL` and required for `MULI`. If this argument is omitted, it is treated as 0.

`shift` An optional argument in the range 0 to 31, set to 0 if omitted, that sets the number of times to divide the result by 2.

The following examples assume that `V1` holds –10 and `V3` holds 10:

```
MULI    V1,10           ; V1 = -100 (-10 * 10)
MUL     V1,V3,-8        ; V1 = -992(-100 * 10 -8)
```

Access to data capture

Most activities in the sequencer are independent of the sampling process. However, there are times when you need to know the value of a channel to decide what to do next. The `CHAN` command gives you the latest waveform value or number of events on a channel. The `TICKS` command tells you the current time in terms of the sampling clock ticks.

The sequencer can also send information in the other direction. If the digital marker channel is active you can force it to record the current digital input state with `REPORT`, or you can force it to record a marker of your own choosing with `MARK`.

CHAN This instruction gives the output sequencer access to sampled data on a waveform channel and to the number of recorded events on all other channel types. You can also use this command to get the most recent value written to the DAC outputs. The variable value is 0 if the channel is not being sampled.

```
CHAN    Vn,chn          ; Vn = ChanData(chn)
```

`chn` The channel number is 1 to 100 for sampled channels or 0 to -3 for the last value on DAC 0 to 3. The result is the most recent data available. For a slow waveform channel this could be a long time in the past. In triggered sampling mode, waveform data is available between triggers.

Waveform and DAC data are treated as 16-bit signed values from –32768 to 32767 for version 3 compatibility. You also have access to DAC values as 32-bit data in variables `V67` to `V64` (`VDAC0` to `VDAC7`) without the need to use the `CHAN` instruction.

This example waits for a signal to cross 0.05 volts and produces a pulse. We assume that channel 1 is a waveform or the result will be very silly.

```
         SET     0.100 1 0        ;run at 10 kHz
         VAR     V1,level=VDAC16(0.05) ;level to cross
         VAR     V2,data          ;to hold the last data
         VAR     V3,low=0         ;some sort of hysteresis level
         DIGOUT  [00000000]       ;set all dig outs low
BELOW:   CHAN    data,1           ;read latest data   >wait below
         BGT     data,low,below   ;wait for below     >wait below
ABOVE:   CHAN    data,1           ;read latest data   >wait above
         BLE     data,level,above ;wait for above     >wait above
         DIGOUT  [.......1]       ;pulse output...
         DIGOUT  [.......0],below ;...wait for below
```

TICKS

This instruction sets a variable to the current sampling time in Spike2 time units (microseconds per time unit set in the sampling configuration) and adds an expression or 0 if `expr` is omitted. The `sTick()`, `msTick()` and `usTick()` expression functions can be used to make the sequence independent of the microseconds per time unit value.

```
TICKS   Vn,expr          ; Vn = Spike2 time + expr
```

This can be used with the `CHAN` command and variable related branches to check the timing of external pulses. The sequencer runs under interrupt, and competes for time with other interrupt driven processes in the 1401 interface. This causes some "jitter" in the timing. The jitter for a micro1401 or Power1401 is typically only a few microseconds. For a 1401*plus*, it can be a few tens of microseconds, depending on other 1401 activity.

REPORT, MARK

The `REPORT` instruction records a digital marker (if the digital marker channel is enabled) as if there was an external pulse on the E1 input. The `MARK` instruction does the same, except it takes the argument as the value to record. `REPORT` has no arguments.

```
REPORT  OptLB
MARK    expr|Vn,OptLB
```

`expr` The argument should have a value in the range 0 to 255. If a variable is used, the bottom 8 bits of the variable are used as the value.

`OptLB` If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

```
WAIT    [.......1]      >Waiting for bit 0
REPORT                  ;save a marker when this is set
MARK    12              ;set code 12 as a digital marker
```

Randomisation

These functions use a pseudo-random number generator. The generator is seeded by a number that is based on the length of time that the 1401 has been running since it was switched on.

BRAND

`BRAND` branches with a probability set by the argument or by a variable. This could be used when several different stimuli are required, but in a random sequence. `BRANDV` is an obsolete name for the same instruction.

```
        BRAND   LB,expr|Vn
```

`LB` Where to go if the branch is taken

`expr` This is the probability of branching in the range 0 up to (but not including) 1.

`Vn` When a variable is used for a branch, it is treated as a 32-bit unsigned number; 0 means a probability of zero and 4294967295 (the largest 32-bit unsigned number) means a probability of 0.9999999998.

```
        BRAND   LB,0.5     ;branch with 50% probability
```

To produce a multiple way random branch you use more `BRAND` instructions. A three way equal probability branch to `LA`, `LB` and `LC` can be coded:

```
        BRAND   LA,0.33333  ;Split the first route with p=1/3
        BRAND   LB,0.5      ;0.6667 to here * 0.5 is 0.3334 (1/3)
LC:     ...                 ;If neither of the above, comes here
```

The following shows the sequence for a five-way branch with equal probabilities:

```
        BRAND   LA,0.2  ;5 way, LA probability is 0.2 (1/5)
        BRAND   FX,0.5  ;Probability to here=0.8, so to FX=0.4
        BRAND   LB,0.5  ;Probability to here=0.4, so to LB=0.2
LC:     ...             ;Probability to here=0.2
FX:     BRAND   LD,0.5  ;Probability to here=0.4, so to LD=0.2
LE:     ...             ;Probability to here=0.2
```

The best technique is to reduce the branches to a power of two as soon as possible. Case 1 of the five-way branch is split off (probability of 0.2), leaving 4 ways. The 4 ways are split with a probability of 0.5 (0.4 for each division) then the last two routes are split, again with a probability of 0.5 (0.2 for each division).

Poisson process

In a Poisson process, the probability of something happening per time interval is constant. You can generate a delay with a Poisson statistic by:

```
POISSON: BRAND   POISSON,prob ; poisson delay
```

The probability is given by `prob = 1.0 - 1.0/(mDelay*S(1))`, where `mDelay` is the mean delay required in seconds and `S(1)` is the built in function that tells us how many steps there are per second. If you would rather express this in terms of a rate, then `prob = 1.0 - rate/S(1)`, where `rate` is the expected rate in Hz.

```
TENHZ:  BRAND   TENHZ,1.0-10/S(1) ;10 Hz mean rate
        DIGOUT  [.......1]        ;set output high
        DIGOUT  [.......0],TENHZ  ;set output low, goto TENHZ
```

This example generates a digital output that pulses to produce an approximation to a Poisson distributed pulse train with a mean frequency of 10 Hz. The approximation improves the shorter the step time. The mean interval between pulses is 100 milliseconds plus the time for 2 steps and the shortest gap between pulses is 3 sequencer steps.

Scripts and variables

From a script you can set sequencer variables as 32-bit signed integers. For the range 2147483648 to 4294967295 we must use negative numbers. This script example shows you how to convert a probability into a variable value and pass it to the sequencer:

```
Proc SetBrandVar(prob, v%) 'prob is probability, v% is variable
prob *= 4294967296.0;      'range 0-4294967296 is 0 to 1.0
if (prob > 4294967295.0 ) then prob := 4294967295.0 endif;
if prob > 2147483647 then prob -= 4294967296.0 endif;
if prob < -2147483647 then prob := -2147483647 endif;
SampleSeqVar(v%, prob);
end;
```

MOVRND This instruction generates a random number in the range 0 to a power of 2 minus 1, then adds an integer constant to it and stores the result in a variable.

```
MOVRND  Vn,bits,expr
```

`Vn` The variable to hold the result.

`bits` The number of random bits to generate in the range 1 to 32. The generated random bits fill the variable starting at the least significant bit. Bits above the highest numbered generated bit are set to 0.

`expr` An optional expression that evaluates to a 32-bit integer number, that is added to the random bits. If this is omitted, nothing is added.

Expressed in terms of the script language, the random number is one of the numbers in the range `expr` to `expr+Pow(2,bits)-1`. For example, `MOVRND V33,8,1` emulates the obsolete `LD1RAN` instruction that generates a random number in the range 1 to 256.

The following code fragment implements a random delay of between 1 and 2.024 seconds (assuming a 1 millisecond clock).

```
MOVRND  V1,10,998 ;load V1 0 with (0 to 1023) + 998
DELAY   V1        ;this uses 999 to 2023 steps
```

LD1RAN This obsolete instruction loads counter 1 (`V33`) with a pseudo-random number in the range 1 to 256. It is implemented as `MOVRND V33,8,1`.

Arbitrary waveform output

In addition to generating voltage pulses, ramps and cosine waves through the DACs, Spike2 can play arbitrary waveforms. The waveforms must be set up before sampling starts. You can play a wave, test the output and branch on the result, and stop a waveform playing or set one more output cycle. Each waveform has an associated code (often a keyboard character) and this code is used by the sequencer to identify the arbitrary wave to replay.

WAVEGO

The `WAVEGO` instruction starts output from a play wave area, or prepares the output for an external trigger. Starting output can take more time than we want to allocate to a sequencer step so `WAVEGO` sets a flag and output starts as soon as the 1401 has free time (usually within a millisecond). See `WAVEST` for a precisely timed start to output.

```
WAVEGO  code{,flags}
```

`code` This is either a single character standing for itself, or a two digit hexadecimal code. This is the code of the area to be played.

`flags` These are optional single character flags. The flags are not case sensitive. Use `T` for triggered waveform output. Use `W` to make the sequencer wait at this step until the 1401 has prepared the hardware to play. For example:

```
WAVEGO  X         ;area X, no wait, no trigger
WAVEGO  23,T      ;area coded hexadecimal 23, triggered
WAVEGO  0,WT      ;area 0, wait until trigger armed
WAVEGO  1,W       ;area 1, wait until play started
```

If you need to know when the output started, use the `WAVEBR T` option. If you need to know that the request to start the playing operation has been honoured, but you do not want to hang up the sequencer with the `W` option, use the `WAVEBR W` option.

If a waveform is playing when you use the `WAVEGO` command, the old output is cancelled just before the new area starts to play. If you want to use `WAVEBR` after the play request, unless you use the `WAVEGO` or the `WAVEBR W` option to be certain that the new area is active, the result of the `WAVEBR` may be based on the previous area, not the new one.

WAVEBR

The `WAVEBR` instruction tests the state of the waveform output and branches on the result. No branch occurs if there is no output running or requested.

```
WAVEBR  LB,flag
```

`LB` Label to branch to if the condition set by the flag is not met.

`flag` An optional single character flag to specify the branch condition:
- `W` branch until a `WAVEGO` request without the `W` flag is complete.
- `C` branch until the play wave area or cycle count changes.
- `A` branch until the play wave area changes.
- `S` branch until the current output stops.
- `T` branch until output started with `WAVEGO` begins to play.

The following sequence tracks the output when we have two play areas labelled 0 and 1. Area 0 is set to play 10 times and is linked to area 1. The sequence below will track the changes. Play wave DAC output happens one play wave clock tick after the output is changed, so the sequencer can know that a DAC output is about to change.

```
         WAVEGO  0,WT      ; area 0, wait for armed
         LDCNT1  5         ; load counter 1 with 5
WT:      WAVEBR  WT,T      ; wait for external trigger>Trigger?
W5:      WAVEBR  W5,C      ; wait for cycle   >Waiting for cycle
         DBNZ1   W5        ; do this 5 times  >Waiting for cycle
WA:      WAVEBR  WA,A      ; wait for area    >Wait for area
WE:      WAVEBR  WE,S      ; wait for end     >Wait for end
```

The WAVEGO command requests a triggered start and waits until the trigger is armed before moving on. It then waits for an external trigger at the WT label. Next the sequence tracks the end of 5 output cycles. At label WA the sequence waits for the area to change and finally the sequence waits for output to stop.

WAVEST The WAVEST instruction can start output that is waiting for a trigger and stop output that is playing, either instantly, or after the current cycle ends (see the *Sampling data* chapter for more details of triggering, including connections).

```
WAVEST  flag
```

flag This are optional single character flag and specifies the action to take:
- T Trigger a waveform that is waiting for a triggered start.
- S Stop output immediately, no link to the next area.
- C Play to the end of the current cycle, then end this area. If there is another area linked it will play.

The following code starts output with an internal trigger and then stops it

```
WAVEGO  X,TW          ; arm area X for trigger
WAVEST  T             ; trigger the area
DELAY   1000          ; wait
WAVEST  S             ; stop output now
```

Compatibility with previous versions

Although the Spike2 version 4 sequencer has many more features than previous versions, sequences from all previous versions still work, with the following provisos:

1. Variables and labels may not share names nor be the same as instruction codes.
2. DELAY 0 no longer hangs up for a long time. It now has the same effect as NOP.
3. If a sequence relied on looping back to the start after 256 steps, this no longer works as all sequences have a HALT instruction automatically added to the end.
4. BRAND used to set the probability to an accuracy of one part in 256. It now sets the probability to a very high accuracy. This might affect your results.
5. DACn and ADDACn now insist that any value you use lies within the range of the DAC.
6. ADDACn previously expressed your increment as a 16-bit integer. It now generates a much more accurate 32-bit integer. However, this means that ramps generated with ADDACn may have a different slope (and probably much closer to the intended slope).

If you want to write a sequence that will run on previous versions of Spike2, either refer to the manual for the previous version, or write the sequence using the previous version to be certain of complete compatibility.

Sequencer compiler error messages

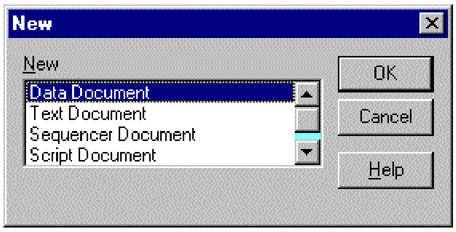
When you use the Format or Compile buttons in the sequence editor, Spike2 displays the result of the compilation or format operation in the message bar at the top of the window. The messages either indicate successful operation or indicate the cause of the problem.

File menu

The File menu is used for operations that are mainly associated with documents (opening, closing, importing and creating), configuration files and with printing. The final command in the File menu is also your route out of Spike2.

New File

This command creates a new Spike2 data file for sampling, an output sequence file, a script file, text file or an XY file. The command opens a new file dialog box in which you choose the type of file to create. You can activate this command from the menu and from the toolbar.



You can make files of five types: data, output sequencer (pulse) files, script files, text files and XY files. Select a file type and click OK, Spike2 will open a window of the specified type. Each file type has its own file name extension.

Data Document

A sampling window opens plus additional windows set by the sampling configuration(see the *Sampling data* chapter for details). Data documents are not stored in memory, unlike most new files, but are kept on disk. Until they are saved after sampling they are temporary files in the directory set in the Edit menu Preferences. The file name extension is `.smr`.

Sequencer Document

A new window opens in which you can type, edit and compile an output sequence (see the *Data output during sampling* chapter for details). The file name extension is `.pls`.

Text Document

Text files can be used to take notes, build reports and to cut and paste text between other windows and applications. The file name extension is `.txt`.

Script Document

A script editor window opens in which a new script can be written, run and debugged. The file name extension is `.s2s`.

XY Document

XY windows are used to draw user-defined graphs with a wide variety of line and point styles. The Windows file name extension is `.sxy`. They can be generated by scripts, or from the Analysis menu for trend plots.

Other types used by Spike2

In addition to the document types listed above, Spike2 also uses:

Result view files

These hold result views (file extension `.srf`). Result views are not created using the File menu New command. They are created as the result of an analysis, or from the script language.

Resource files

Spike2 creates files with the extension `.s2r`. These are associated with data files of the same name, but with the extension `.smr`. They hold configuration information so that Spike2 can restore the display. These files are not essential to Spike2 and if deleted, the associated data file is not damaged in any way.

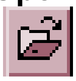
Configuration files

Spike2 stores sampling configuration information in these files. The file name extension is `.s2c`.

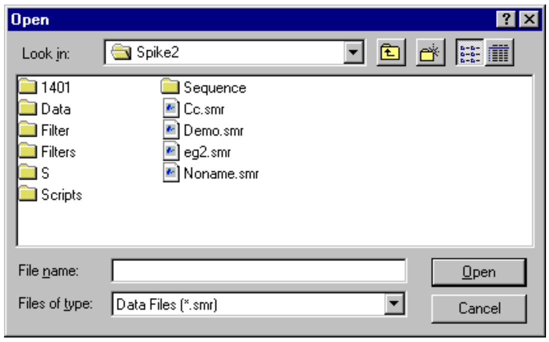
Filterbank files

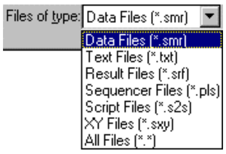
These files hold descriptions of digital filters and have the extension `.cfb`. They are used by the Analysis menu Digital filters command.

Open



You can activate this option from the menu or toolbar. It shows the standard file open dialog for you to select a file. You can open six file types with this command: a Spike2 data file, a Spike2 result view, a text file, a script file, an output sequence file or an XY file. The type of the file is selected with the Files of type field.





When you select a data file, Spike2 looks for a resource file of the same name, but with the file extension `.s2r`. If this is found, the new window displays the file in the same state and screen position as it was put away. Several windows may open if the file was closed with the `Ctrl` key held down. See the Close command, below, for more details.

If a text file is selected, a text window opens. This can be used as a notepad or as a repository for data copied from other parts of Spike2. If an output sequence or script or XY type is selected, an output sequence or script or XY window is created.

Import

Spike2 can translate data files from other formats into Spike2 data files. The import command leads to a standard file open dialog in which you select the file to convert. You then set the file name for the result; Spike2 will suggest the same name with the extension changed to `.smr`. The details of the conversion depend on the file type.

Supported formats include text files with data in columns, the CFS files used by CED programs such as Signal and SIGAVG, Spike2 for Macintosh files as well as data from several third party vendors. Spike2 searches the `import` folder in the Spike2 installation folder for CED File Converter DLLs. If you need to translate a file format that is not covered, please contact us and describe your requirements. The script language command to convert files is `FileConvert$()`.

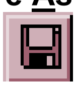
Close

This command closes the active window. If you use this command on an unsaved window, you are prompted to save it before closing the window. However, if the Edit menu Preferences option is set to not prompt to save modified result and XY views, you will not be prompted to save these views and they will be lost when you close them.

Close all associated windows

If you hold down the `Ctrl` key and close a time view, Spike2 closes the time view and all windows associated with it. This only works if the time view has been saved to a data file; it does not work with a newly sampled file. If the Edit menu Preferences option is set to not prompt to save result and XY views, associated result views are deleted. However, in addition to saving the state of the data file in a `.s2r` resource file (see page 6-1), Spike2 also saves the state and contents of any result view windows that belong to the data file in the same resource file. Next time you open the data file the result views are recreated from the resource file in the same state as they were closed.

Save and Save As



These commands are available when the current window is a text, script, output sequence, result or XY view or a time view holding newly sampled data that has not been saved. Save writes the file with its current name unless it is unnamed, in which case you are prompted for a name. Save As writes the file with a different name.

Time view files are kept on disk, not in memory, as they can be very large. Changes made to these files are permanent as they are made on disk. When you save a newly sampled data file, you give the file a name (replacing a temporary name). If you save it to a different drive from that set in the Edit menu Preferences, Spike2 copies the file to the new drive, then deletes the original. Moving a large file can take several seconds.

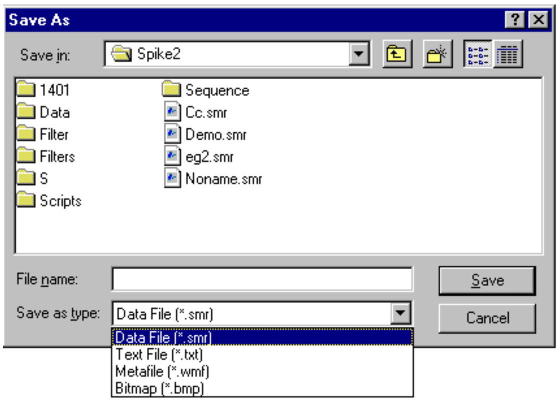
Sequence, text, result, XY and script files are held in memory. Changes made to these files are not permanent until the file is saved to disk.

Revert To Saved

You can use this command with a text file, a script file or an output sequence file. The file changes back to the state it was in at the last save.

Export As

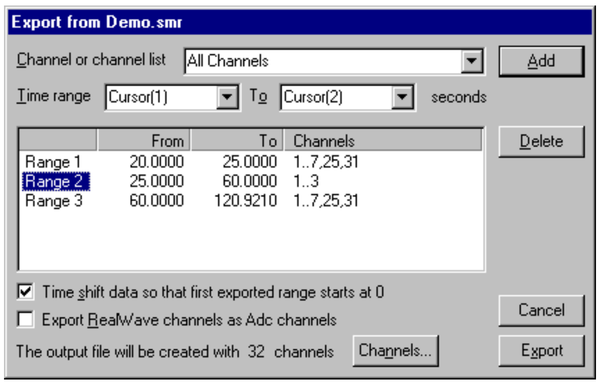
This menu item replaces Save As when the current window is a named time view. The dialog prompts you to choose a file for the output, and lets you select the output format. You can choose between: Data file (`*.smr`) to export as a new Spike2 file, Text file (`*.txt`), Metafile (`*.wmf`) for a scaleable image and Bitmap file (`*.bmp`) for a copy of the screen rectangle containing the window.



Select one of the formats and either select an existing file to overwrite or type a new file name, then use the Save button.

Data file

This opens a dialog in which you can create a new data file from selected channels and sections of the current file. You build a list of time ranges and channels using the top half of the dialog. Click Add to append the time range and selected channels to the list. Click Delete to remove a list entry. You can choose channels from the drop down list or type in a channel specification. You can export channels that exist in the original file plus memory channels; duplicated channels are not listed. Check the Time shift data… box to time shift the output data so that the Range 1 From time becomes 0.0 in the output file.



RealWave data was added at version 4.03. Older versions of Spike2 cannot read data files containing this data type. If you check Export RealWave channels as Adc Channels the channels are converted using the scale and offset value stored in the RealWave channel to produce a channel that older versions can read.

You can set the maximum number of channels in the output file with the Channels… button. Versions of Spike2 before 4.02 cannot read data files with other than 32 channels.

Once you have formed a suitable list of times and channels, click the Export button to write the selected time ranges and channels to the new data file. Channels are written in order of ascending channel number. Where possible, channels are copied to the same channel number in the output file. If this is not possible, for example for a memory channel, the channel is written to the lowest numbered unused channel in the output file.

Text file This is the same as the Edit menu Copy As Text command, but with the output sent to a text file and not the clipboard. See the Copy As Text command for details of the dialogs.

Bitmap file This option copies the screen area containing the window to a file. Make sure that the window is completely on the screen and that it is not covered by any other window. Use this option when the image you require is an exact copy of the screen. If you need to scale the image, or want to edit it, a Metafile copy is usually better.

Metafile This option copies the window as a Windows (Placeable) Metafile (`*.WMF`) or as an Enhanced Metafile (`*.EMF`). Enhanced Metafiles are theoretically better as they support the cubic spline and sonogram drawing modes. However, some graphics programs do not import Enhanced Metafiles very well. The default format is a Windows Metafile. To output in enhanced format, set the file extension to `.EMF`. For example, to save to the file `fred` as a Windows Metafile, set the file name to `fred` or `fred.wmf` but to save as an enhanced metafile set the file name to `fred.emf`.

These file formats can be scaled without losing resolution and are the preferred format for moving Spike2 images to drawing programs. The Edit menu Preferences… option lets you increase (or decrease) the output resolution. This can be important when saving time view and result view data as the number of vectors produced when drawing high resolution may stop some drawing programs from reading the file.

Load and Save Configuration

These commands manage Spike2 configuration documents. Configuration documents hold the colour setup, the channels, output sequence file and window arrangement required during sampling and the types of on-line analysis required.

The Load Configuration and Save Configuration commands transfer the Spike2 configuration between disk and the application. They both open an appropriate file dialog to select a file for loading or saving. You can read sampling parameters from a Spike2 data file (this is useful if you have lost the corresponding configuration file).

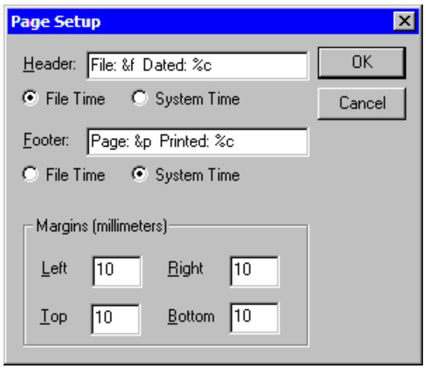
If you use the Load Configuration command and select a file to read the configuration from, the Sampling Configuration dialog will open to display the new configuration. If you wish to sample immediately, click the Run Now button.

You can also use the Sample Bar to load a configuration and start sampling using it with a single mouse click (see the *Sample menu* chapter for more information).

Default configuration files default.s2c last.s2c

If the configuration file `default.s2c` exists in the current directory, it is loaded when Spike2 starts. You can save this file yourself, or hold down the control key while activating the File menu and use the Save Default Configuration command to create this file automatically. The standard file `last.s2c` holds the last configuration that was used for sampling. If `default.s2c` cannot be found, and `last.s2c` exists, `last.s2c` is loaded. Spike2 saves `last.s2c` each time sampling stops.

Page Setup

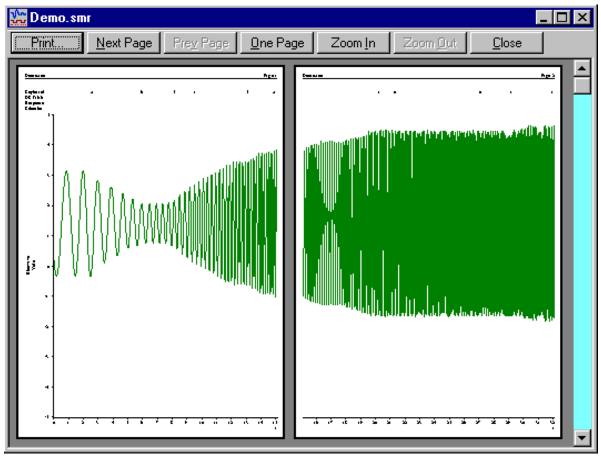


This opens the printer page setup dialog. The dialog varies between operating systems and printers. See your operating system documentation for more information.

For time, result and XY views the important options that are always present include the paper orientation (portrait or landscape), the paper source (if your printer has a choice), the printer margins and the choice of printer.

In text-based views, in addition to page margins you can set header and footer text to appear on each page. If you include `&f` in the text it is replaced by the file name, `&p` is replaced by the page number and `%c` is replaced by the file or system time.

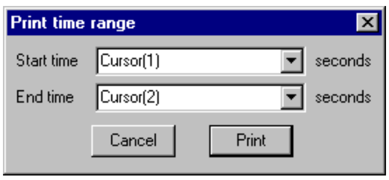
Print Preview



This option displays the current window as it would be printed by the Print option. You can preview time, result, XY and text based windows. You can zoom in and out, view a single or two pages, step through pages of multi-page documents and print the entire document using a toolbar at the top of the screen. Use the Close button to leave this mode without printing.

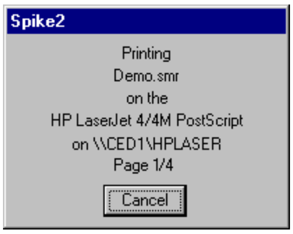
Print Visible, Print and Print Selection

These commands print time, result, XY and text-based windows. Any scroll bar at the bottom of the window is not printed. The Edit menu Preferences dialog sets the line widths used on screen and for printing. Print Selection prints the selected area of a text window. Print Visible prints the visible data in the current window. Print prints a specified region of a time, result or XY view; each printed page holds the x axis range of the window.



To print an entire data file, set the width of the time window holding the file to the page size required in the print, then select Print. Set the start time to zero and use the drop down list to set the end time to Maxtime(). Beware that Print could require several hundred miles of paper to complete the printing job in the worst case! If you displayed 10 milliseconds of data across the screen in a file that is 1000 seconds long, there are 100,000 pages to print.

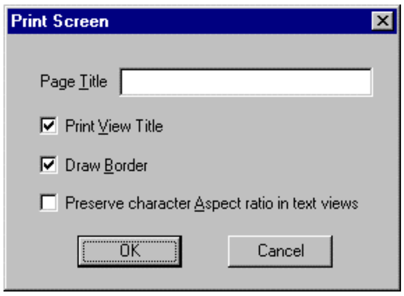
Both commands open the standard print dialog for your printer. You can set the print quality you require (in general, the better the quality, the longer the print takes) and you can also go to the Setup page for the printer. Once you have set the desired values, click the Print button to continue or the Cancel button to abandon the print operation.



During the print operation (which can take some time, particularly if you have selected a lot of data) a dialog box appears. If your output spans several pages, the dialog box indicates the number of pages, and the current page so you can gauge progress. If you decide that you didn't want to print, click the Cancel button.

Print Screen

The Print Screen command prints all time, result, XY and text based views to one printer page. The views are scaled to occupy the same proportional positions on the printed page as they do on the screen. The command opens a dialog in which you can set a page title and choose to print view titles and draw a box round each view. You can also preserve the aspect ratio of characters in text windows.



Exit

This command closes all open files and exits from Spike2. If there are any text or output sequence files open that have not been saved, you will be prompted to save them before the application terminates.

Send Mail

If your system has support for Mail installed (for example Microsoft Exchange), you can send documents from Spike2 to another linked computer. This option vanishes if you do not have compatible mail support.

Text-based documents and result views can be sent, even if they have not been saved to disk (Spike2 writes them to a temporary file if they have not been saved). You can send Spike2 data files, but only if they have been saved on disk.

Read-only files

You can open Spike2 data files that are read-only, but you are not allowed to write any changes back to the file. Read-only data files have [Read only] added to the window title. Unless the file is on a read-only medium (such as CD) you can usually clear the read-only state by opening the Property page of the file (right click on the file name or a list of selected files) and clear the Read-only check box. If you copy files from a read-only drive such as a CD, the copied file is marked read-only.

Read-only text-based files open normally, but you cannot edit them.

Edit menu

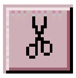
The majority of the Edit menu is associated with commands that move data to and from the clipboard. You can also use these commands to search for strings in a text window or to search for the currently selected text.

When the current window is a text window or an output sequence window, the Edit menu operates in the standard manner, allowing you to cut and paste text between text windows in Spike2 and other applications. When the current window holds a Spike2 data document or a result window, the behaviour is modified.

Undo and Redo

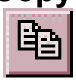
In a text, script or output sequence window this is used to undo or redo the last text edit operation. These windows support multiple Undo and Redo operations, however you cannot undo operations that have been saved to disk or text operations that were done by a script. In Spike2 data document windows, and in result windows, you can undo most operations that change the appearance of the data window.

Cut



You can cut editable text to the clipboard in any position in Spike2 where the text pointer is visible. You cannot use this command in Spike2 data document windows or in result windows.

Copy

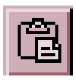


You can copy editable text to the clipboard plus selected fields from the Cursor Regions and Cursor Values dialogs. If you use this command in a time, result or XY window, the contents of the window, less the scroll bar, are copied to the clipboard as both a bitmap and as a metafile. See also Copy As Text.

Metafile output

To export an image to a drawing program for further annotation and manipulation, you can paste the image as either a bitmap or as a metafile. Metafiles are usually the preferred choice as you can treat the image as lines and text for further work. You can set the metafile scaling and if the image is saved as a Windows Metafile or as an enhanced Metafile in the Edit menu Preferences (see page 7-8 for more details).

Paste



You can paste text on the clipboard into a text, script or output sequence document. When you paste text, Spike2 checks that each pasted line ends with the correct end of line characters (for example, Macintosh and Windows use different sequences). The paste operation corrects incorrect sequences. To correct text that has come from a different operating system and that looks strange, select all the text, cut, then paste.

Delete

This command is used to delete the current text selection, or if there is no selection, it deletes the character to the right of the text caret. Do not confuse this with Clear, which in a text field is the equivalent of Select All followed by Delete.

Clear

When you are working with editable text, this command will delete it all. If you are in a result window, this command will set all the bins to zero and redraw the window contents. Clear removes everything; Delete removes the current selection.

Copy As Text (Result view)

This command is available in result views. It copies the bin values *in the window* as text to the clipboard. To copy all bins, double click the x axis and select Show All, then copy. The first output column holds the x value at the left of each bin. Each channel contributes one column of bin contents plus a second column of error bar sizes if error bars are enabled. The first line holds column titles. Duplicated channels are not dumped. The first example is from a result view holding two waveform averages:

```
"Time"    "Filtered"   "Sinewave"
0         0.84302088   3.2980477
0.01      0.27032173   2.2613753
0.02      -0.45484864  1.043198
0.03      -0.74969506  0.63429488
0.04      -0.45483295  1.0715468
```

The columns are separate by Tab characters. The second example is from the same data with error bars enabled and drawn in `SEM` mode.

```
"Time"    "Filtered"   "SEM"        "Sinewave"    "SEM"
0         0.84302088   0.038987886  3.2980477     0.032728175
0.01      0.27032173   0.021425654  2.2613753     0.036360926
0.02      -0.45484864  0.027970654  1.043198      0.056437311
0.03      -0.74969506  0.037386934  0.63429488    0.050863126
0.04      -0.45483295  0.035408324  1.0715468     0.051846368
```

Copy As Text (XY view)

This menu item is available in XY views. It copies the XY points for all visible channels to the clipboard. If the XY view holds a single channel or was generated by the measurement system with the All channels use same X option, the output is a rectangular table with the first line holding column titles. There is one column of x values followed by one column per channel for each y value. The columns are separated by a tab character:

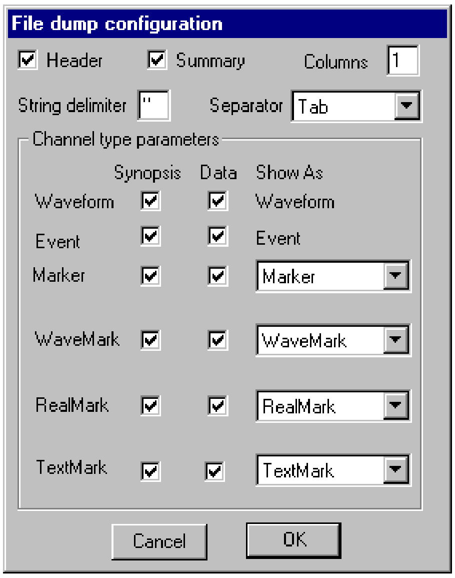
```
"X"       "Channel 1"  "Channel 2"
0         0.244140625  0.0170616301
2         3.122558594  0.0001053187043
4         2.211914063  0.0006319122258
6         1.889648438  -0.0005265935215
8         0.8642578125 -0.0005265935215
10        0.5712890625 -0.0005265935215
```

In all other cases, channels are output separately. For each channel, the first output line holds “`Channel : cc : nn`” where `cc` is the channel number and `nn` is the number of data points. The data points are output, one per line as the x value followed by the x value, separated by a tab character.

```
Channel : 1 : 6
0.0170616301      0.244140625
0.0001053187043   3.122558594
0.0006319122258   2.211914063
-0.0005265935215  1.889648438
-0.0005265935215  0.8642578125
-0.0005265935215  0.5712890625
Channel : 2 : 6
0                 0.0170616301
2                 0.0001053187043
4                 0.0006319122258
6                 -0.0005265935215
8                 -0.0005265935215
10                -0.0005265935215
```

Copy As Text... (Time view)

In a time view the command leads to dialogs in which you specify the time range of data to copy, how you want the data to be copied and the output format. Select the channels to copy in the time view before you use this option. If you select no channels, Spike2 copies all the channels. Text representations of sampled data can be very large and awkward to manipulate with the clipboard. Alternatively, you can write the text output to a file with the File menu Export As command.



The first dialog sets the format of the output. You can enable and disable a header section for the entire output and a summary section of all the channel information. You can also set the number of columns to be used when writing waveform and event times and the separators used between fields and to delimit text. For each type of channel you can enable the channel synopsis and data output. You can also choose to dump a channel type in its native format, or in any compatible format All output sections are preceded by a keyword. This keyword is provided so that other programs can parse the file easily.

Output fields, separators and string delimiters

All information is written in fields. Each field is either numeric or text. Between each field there is a separator, which can be set to be a Tab character, a blank or a comma in the Separator drop down list. Most programs that accept tabulated numbers will accept space, a Tab or a comma. The examples below use space as a separator.

A numeric field holds numbers only, either floating point numbers with a decimal point, or integer numbers. A text field is a sequence of characters that may include spaces. Text fields may hold numbers, but numeric fields cannot hold text. You can mark the start and end of a text field with a special character (usually ") so that a program reading the field can include blanks (spaces) and punctuation within a field without confusion. All keywords are treated as text fields. You can set a one character delimiter in the String delimiter field, or leave it blank for none. The examples use " as a delimiter.

Header

The first part of the output is a header that displays information about the data file. This is given after the keyword `INFORMATION`. The header consists of the file name followed by five lines of the file comment and a blank line. The header block is always 8 lines long (if present); blank file comment lines are not skipped.

```
"INFORMATION"
"demo.smr"
"These five lines"
"contain"
"the file comment"
"for this"
"data"
```

Summary

The summary section starts with the keyword SUMMARY followed by a summary of the channel information for each selected channel. The information given for the channel varies with the channel type. The first three fields are the same for all channels, being the channel number, the channel type and the channel title. The remaining fields are:

```
Waveform   Units, Ideal rate, Actual rate, Scale, Offset
WaveMark   Units, Ideal rate, Actual rate, Scale, Offset, Points, PreTrig
Event      Predicted mean rate
Marker     No other information
```

Here is some typical output. The SUMMARY section is terminated by a blank line.

```
"SUMMARY"
"1" "Waveform" "Sinusoid" "Volts" 100.0 100.0 1.0 0.0
"2" "Evt-" "Synch" 5.0
"4" "WaveMark" "Shapes" "Volts" 100.0 100.0 1.0 0.0 30 5
"31" "Marker" "untitled"
```

Channel information

For each channel selected an information section is displayed which is headed by the keyword CHANNEL, followed by the channel number. The channel number is a text field. The channel data type, comment and title are then given on the next three lines. The channel information block ends with a blank line.

```
"CHANNEL" "1"
"Waveform"
"Thandar signal generator"
"Sinusoid"
```

The information that follows this channel synopsis varies with the channel type:

Waveform

A channel containing waveform data has the channel synopsis followed by the data. The data starts with the channel units and ideal sample rate. The next line contains the keyword START followed by the start time of the data in seconds and the time increment per data point, also in seconds. It is possible for gaps to occur in the waveform data. A gap is shown by a line with the word GAP followed by the start time of the new section and the time increment. The data then follows as a list of waveform values.

```
"CHANNEL" "1"
"Waveform"
"Thandar signal generator"
"Sinusoid"

"Volts" 100.0000
"START" 0.00 0.01
-3.7208
-3.8356

"GAP" 4.23 0.01
-2.2901
-2.6075
-2.8858
```

Event

Event data comes in three types, rising edge, falling edge and both edge triggered (Evt-, Evt+ and Level). These three types are represented in basically the same way.

```
"CHANNEL" "2"
"Evt-"
"Synchronisation pulse from Signal generator"
"Synch"

0.2022
   .
   .
```

In the case of an event channel triggered on both edges, the state before the first transition is shown at the start of the data, by one of the words `HIGH` or `LOW`. `HIGH` means that the first time in the list represents a low to high transition, `LOW` means the reverse. If there are no times in the list, these labels are the opposite of the state in the period.

```
"HIGH"
0.2022
   .
   .
```

Marker

Marker data consists of a series of times and 4 bytes of marker information. The channel synopsis is as for other channels, and the data is displayed as a time followed by a string of 4 characters which represent the 4 bytes of information. If the bytes are non-printing characters a ? is shown instead. The 4 bytes are also displayed as numbers after the string. You can force marker data to be dumped as event data.

```
"CHANNEL" "16"
"Marker"
"No comment"
"untitled"

1.0906  "p???" 112  0  0  0
3.0336  "a???"  97  0  0  0
5.9802  "u???" 117  0  0  0
6.9018  "l???" 108  0  0  0
```

WaveMark

WaveMark data is structured as a marker plus a short section of waveform data. The channel synopsis is the same as for waveform data, but also has the number of waveform points associated with each event and the number of pre-trigger points. The dump of data starts with a line holding the channel units, the ideal sampling rate, the number of data points and the number of pre-trigger points:

```
Units, Ideal rate, Points, PreTrig
```

This is followed by blocks of data, one for each WaveMark event in the time range. The first line of the block holds the keyword `WAVMK`, followed by the time of the first data point in the event, the time interval between waveform points and the four marker bytes:

```
WAVMK,eventTime,adcInt,mark0,mark1,mark2,mark3
```

The following lines in the data block hold the waveform data values. This block is repeated for each event in the time range. A typical WaveMark channel begins:

```
"CHANNEL" "1"
"WaveMark"
"Channel comment for demonstration"
"Title"

"Volts" 100.0 30 5
WAVMK 0.0210 0.01 1 0 0 0
0.1318
.
.

"WAVMK" 0.8110 0.01 3 0 0 0
.
.
```

You can also force a WaveMark channel to be dumped as though it was a marker channel, as an event channel or as a waveform channel.

RealMark

The first line of RealMark data output holds the channel units followed by the number of reals attached to each event. This is followed by data blocks flagged by `REALMARK` followed by the event time and the four marker bytes. The following lines in the data block hold the real number. You can force RealMark data to be dumped as Marker or event data. Typical data follows:

```
"Units" 2
"REALMARK" 1.0906  112 0   0   0
1.8923
4.8567

"REALMARK" 1.8603  113 0   0   0
1.8224
4.123
```

TextMark

The first line holds the maximum number of characters allowed. This is followed by data blocks separated by blank lines, one for each TextMark. The first line of each block starts with `TEXTMARK`, followed by the marker time and the four marker bytes. The second holds the text. You can force TextMark data to be dumped as Marker or Event data.

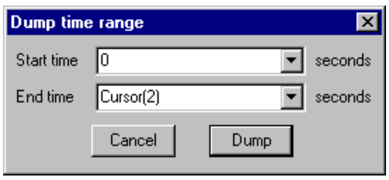
```
100
"TEXTMARK" 0.5234  112 0   0   0
"This is where I added the secret ingredient"

"TEXTMARK" 9.8603  113 0   0   0
"Control point 1"
```

You can also export this data type by double clicking any TextMark and copying the items from the TextMark dialog.

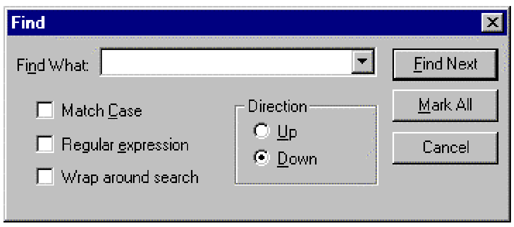
Time range

You select the time range of data to copy as text in this dialog. You can either type in the range, or use the drop down lists to select start and end times for the data output. When you are satisfied with the range, click on Cancel or OK to start the output.



Find
Find Again

These commands move the text cursor to the next occurrence of a string in a text window. The Mark All button sets a bookmark on all lines that match the search text. Searches are insensitive to the case of characters, unless you check Match Case.



The search starts at the cursor position and stops at the document end unless you check the Wrap around search box to search the entire file, regardless of where you start. Searches are done line by line (you cannot search for text that spans more than one line). You can also search for *regular expressions*. If you check this box, you can include pattern matching characters in your string. The simplest pattern matching characters are:

- `^` Start-of-line marker. This character must be at the start of the search text. The following search text will only be matched if it is found at the start of a line.
- `$` End-of-line marker. If you include this character it must come at the end of the search text. The preceding text will only be matched if it is found at the end of a line.
- `.` Matches any character.

To treat these special characters as normal characters with regular expressions enabled, put a backslash before them. A search for “`^\^.\.`” would find all lines with a “^” as the first character, anything as a second character and a period as a third character.

To search for a character list, enclose it in square brackets, for example "`[aeiou]`" will find any vowel. For a character range use a hyphen to link the start and end of the range. For example, to find any alphanumeric character use "`[a-zA-Z0-9]`". If the search is not case sensitive you can omit one of "`A-Z`" or "`a-z`". To include the "`-`" character in a search, place it first or last in the list. To search for any character that is not in a list by placing a "`^`" as the first character. To find a non-vowel character use "`[^aeiou]`".

There are three more special search characters that control how many times to find a particular character. These characters follow the character to search for:

`*` Match 0 or more of the previous character. So "`51*2`" matches `52`, `512`, `5112`, `51112` and so on. You can also match 0 or more or a character pattern, for example: "`h.*l`" matches `hl`, `hel`, `hail`, `horribl` and "`B[aeiou]*r`" matches `Br`, `Bear`, `Beer`, `Beeeaaaooor` and so on.

`+` Match 1 or more. The same as "*", but there must be at least one matching character.

`?` Match 0 or 1. The same as "*", but matches one character at most.

The Find Again option repeats the last search with the same options.

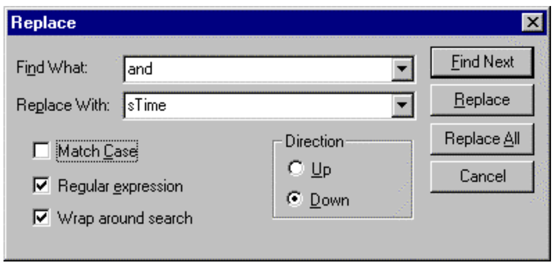
Edit toolbar

If you click the right mouse button on an empty area of a toolbar, or on the title bar of the Spike2 window, you can show (or hide) the Edit toolbar. This bar gives you access to the edit window bookmarks and short cuts to Find, Find next and Find previous. To find out the action of each button, move the mouse pointer over the button and leave it for a second or so and a "ToolTip" will reveal the button function.

The bookmark functions let you set/clear a bookmark on the current line, go to the next or previous bookmark and clear all bookmarks. You can also set bookmarks on all lines containing a search string with the Edit menu Find command.

Replace

This option searches for and replaces text in a text-based window. The search part of the dialog is identical to the Find command. Replace deletes the current match, replaces it, and searches for the next match. Replace All searches for all matches and replaces them.

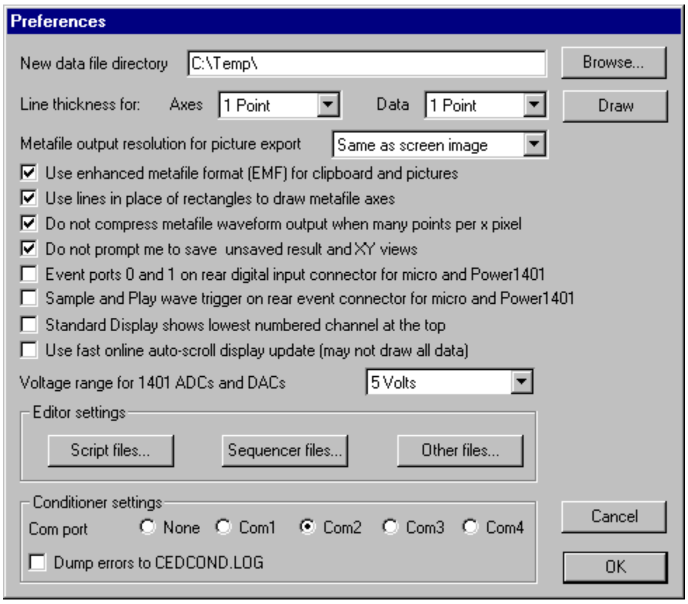


Select *All*

This command is available in all text-based windows and selects all the text, usually in preparation for a copy to the Clipboard command.

Preferences

The Edit menu Preferences dialog sets options for screen appearance, editor settings, metafile output, hardware connections and where to store data during sampling. The preferences are stored in the Windows registry and are user specific. If you have several different logons set for your computer, each logon has its own preferences.



New data file directory

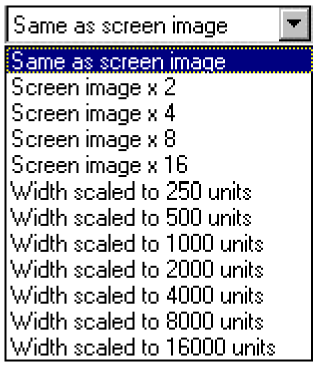
This is the directory/folder where Spike2 stores files data created by File menu New during sampling. If you do not set a directory, Spike2 uses the current directory, which may not be where you expect, so it is a good idea to set one. If you have more than one disk drive, choose a folder in your fastest drive. Do not use a networked drive. New data files in this folder do not have the `.SMR` file extension.

When you close a new data file, Spike2 prompts you for the file name. What happens next depends on where you choose to save the file. If the file is on the same volume (disk drive) as the directory/folder set in the preferences, Spike2 just renames the file. If the volume is not the same, Spike2 copies the file, and then deletes the original.

Line thickness for Axes and Data

Choose from Hairline (as thin as possible) and a list of point sizes (a point is 1/72nd of an inch, about one screen pixel); 1 point is a reasonable size for most purposes. Hairline is not a good idea for printing as the better your printer resolution, the thinner the line. If you find that coloured lines are not printing well to a monochrome printer, either make the lines thicker or use the View menu Use Black and White command. The Draw button redraws screen images to show the effect of any change.

Metafile output resolution



Spike2 saves time, result or XY views as pictures in either bitmap (a copy of the screen image) or metafile format. A metafile describes an image in terms of drawing operations. You can set the metafile drawing resolution; the higher the resolution, the more detail in the picture.

A problem for time and result views with many data points is that the higher the resolution, the more lines need to be drawn, and many drawing programs have limits on the number of lines they can cope with. You can use multiples of the screen resolution, or fixed resolutions. If you are not sure what setting to use, start with *Same as screen image* and adjust it as seems appropriate.

Use enhanced Metafile format

Spike2 supports two metafile formats: Windows Metafile (`WMF`) and Enhanced Metafile (`EMF`). `WMF` is a relic of 16-bit Windows and has limitations, but is widely supported. `EMF` is the standard for 32-bit programs and has many more features. However, some graphics packages do not support this fully (this was written in late 1998, but is still true in 2001).

If you use `EMF` format you can export waveforms drawn in cubic spline mode and sonogram data as part of the metafile. If you use the `WMF` format, waveforms drawn in cubic spline mode will be exported as plain lines and sonograms will be blank.

Use lines in place of rectangles…

This only affects metafile output. Some graphics programs cannot cope with axes drawn as rectangles; check this box to draw axes as lines. We use rectangles to make sure that axes drawn with pens of other than hairline thickness join up correctly.

Do not compress metafile waveform output

When Spike2 draws waveform data on the screen, it does not waste time drawing lines that make no visual difference to the output. If there are more than 3 data points per horizontal pixel, Spike2 draws one vertical line per horizontal pixel to give the same visual effect as drawing all the separate lines. However, when you export an image as a metafile this may not look correct as it relies on a perfect match between the vertical line width and the vertical line separation.

If you check this box, no compression is done when data is saved as a metafile or copied as a metafile to the clipboard. To get the most precise image, set the Metafile output resolution to 16000 units and check this box. When you import a metafile, most drawing programs will preserve the waveform as connected lines. Spike2 writes long sequences of data points in blocks of up to 4000 points. The last point of a block is at the same position as the first point of the next.

Prompting to save views

Several users pointed out that it was very irritating to be asked if you want to save data that is derived from other data, and that can easily be derived again. This is especially true when you are developing a new script application. If you check the *Do not prompt me to save unsaved result and XY views* box, Spike2 will close and throw away result and XY views without requiring a confirmation. As this is potentially destructive, we suggest that you don't use this option when you care about the result or XY data.

Event ports 0 and 1 connections

If you have a micro1401 or Power1401, you can source event ports 0 and 1 from the front panel or the digital input connector. Check the *Event 0 and 1 on back panel…* box to be pin compatible with the 1401*plus*. Leave the box unchecked to source your event signals from the front panel Event 0 and 1 inputs.

Sample and Play wave trigger connections

With a micro1401 or Power1401, you can trigger sampling and on-line waveform output with the front panel Trigger BNC, or on the rear panel Events connector pin 4 (GND is pins 9-15). Check the *Sample and Play wave trigger…* box to use the rear connector.

Channel order

You can change the order of time and result view channels by clicking and dragging channel numbers. The *Standard display shows the lowest numbered channels at the top* checkbox sets the order when you use the Standard display command or open a new window. If you do not check the box, lower numbered channels are at the bottom.

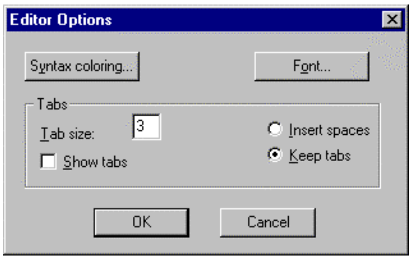
Use fast online auto-scroll display update

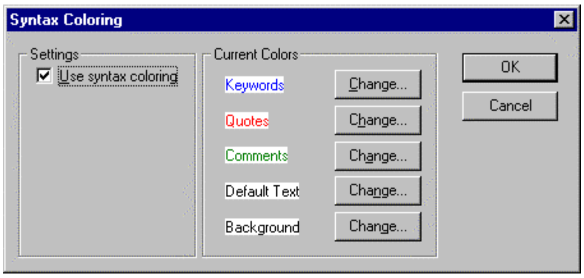
The online automatic scrolling of new data into the display during sampling can make the host computer feel unresponsive, especially in complex display modes at high sampling rates. If you check this box, Spike2 uses a faster (but incorrect) algorithm to decide how much of the screen to repaint each time the view scrolls during sampling.

In the faster mode, Spike2 does not properly allow for changes in already drawn data that depend on the newly sampled data that has just been scrolled into the display. This is particularly visible with sonogram displays and channels with channel processing options such as time shifts.

Editor settings

These buttons lead to dialogs for the standard settings for script, output sequence and text files and to control syntax colouring. The Font… button opens a font dialog that sets the font used when you open a file. You are restricted to fixed pitch fonts in the editor. The Tab size field sets how many spaces to move for each press of the Tab key. You can also choose to keep the tabs as tab characters, or have tabs converted to spaces.





The Syntax colouring… button opens a new dialog in which you can enable and disable syntax colouring for script and output sequencer windows. Enabling it for other windows has no effect. You can also change the colours used for syntax colouring and preview the effect.

Conditioner settings

Programmable signal conditioners (see the *Programmable signal conditioners* chapter) are controlled through communication (serial) ports. Check the *Dump errors…* box to write diagnostic messages to `CEDCOND.LOG` in the Spike2 folder.

Voltage range for 1401 ADC and DACs

Most 1401s have a ±5 Volt input range, but a small number have ±10 Volts. Spike2 can detect this setting automatically in recent 1401s, but you must set it manually for 1401*plus*. Choose from 5 Volt, 10 Volt and Last seen hardware. You are warned if Spike2 detects a conflict between user settings and installed hardware.

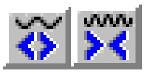
Changing to the 10 Volt setting affects the scaling in the sampling configuration and the DAC output values generated from the output sequencer. It has no effect on scale values in previously sampled data files.

View menu

This menu controls the appearance of the Spike2 time view, result and XY windows. The menu is divided into five regions. The first controls the visibility of the main toolbar and status bar. You can also control these by right clicking in the toolbar docking area. The second houses controls for zooming in and out in both the x and y directions. These controls are also available as short-cut keys, and through mouse actions.

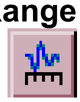
The third region controls the channels that are displayed on the screen, and the channel drawing modes. You can also use it to get information about a window or channel. The fourth region controls the fonts used when drawing the windows and the use of colour in time, result and XY windows. The final section allows you to simulate sampling for practice and demonstrations.

Enlarge View
Reduce View

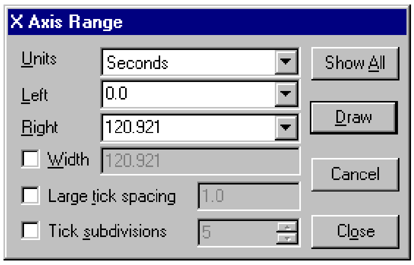


These two commands duplicate the two buttons at the lower left of time and result windows. The enlarge command, short cut `Ctrl+E`, doubles the x axis data region and the reduce command, `Ctrl+R`, halves the region. The left hand window edge is fixed unless enlarging would display data beyond the end of the data, in which case the displayed area is moved backwards. If enlarging would display more data than exists, all the data is displayed. Short cut keys `Ctrl+U` and `Ctrl+I` zoom about the screen centre. You can also change the view size by clicking and dragging the x axis numbers.

X Axis Range



This menu and toolbar command duplicates the action of double clicking on the x axis in a time, result or XY window. The Units field in a time window selects Seconds, hh:mm:ss (hours, minutes, seconds) or Time of day display mode. Time of day works for files created by Spike2 version 4.02 onwards; for older files with no saved creation time it is the same as hh:mm:ss.



The Time of day axis mode draws times as hh:mm:ss on the x axis that are the time of day at which the data was collected plus the time offset into the file. This is for display purposes only; all times used within Spike2 are times in seconds from the start of sampling. All times entered into dialogs are relative to the start of the file.

The Left and Right fields set the window start and end. You can type in new positions or use the drop down lists next to each field. The drop down list contains the initial field value, cursor positions, the minimum and maximum allowed values and the left and right edges of the window (`XLow()` and `XHigh()`). The Width field sets the window width if the Width box is checked. Click the Draw button to apply changes in these fields to the window. Show All expands the x axis to display all the data and closes the dialog.

In normal use, you will let Spike2 organise the x axis style. However, when preparing data for publication you may wish to set the spacing between the major tick marks and the number of tick subdivisions. If you prefer a scale bar to an axis, you can select this in the Show/Hide channel dialog.

You can control the Large tick spacing (this also sets the scale bar size) and the number of Tick subdivisions by ticking the boxes. Your settings are ignored if they would produce an illegible axis. Changes made to these fields take effect immediately; there is no need to use the Draw button.

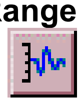
Cancel undoes any changes made with the dialog and closes it. The Close button closes the dialog.

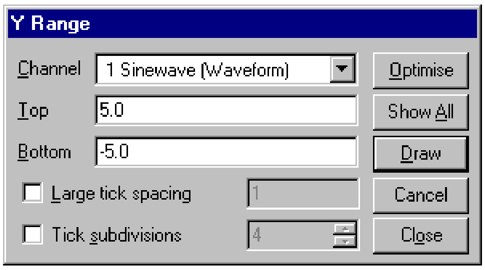
Short cut keys Short cut keys that control the x axis are: `Home` and `End` move to the start and end of the data, `Left` and `Right` arrow scroll by one pixel, `Shift+Left` and `Shift+Right` scroll by several pixels and `Ctrl+Left` and `Ctrl+Right` move by half the screen width.

Jump to event In a time view with cursor 0 in static mode, you can jump to the next or previous event by selecting the event channel, then using `Ctrl+Shift+Right/Left` arrow. Spike2 searches for the nearest event to the centre of the screen in the specified direction. If more than one event channel is selected, all channels are scanned for the nearest event. Spike2 beeps if no event is found. If cursor 0 is active, this key combination moves cursor 0.

You can jump to any TextMark from the TextMark dialog. Double-click any TextMark to open the dialog, click the >> button, select the marker from the list and click Show.

Y Axis Range





This dialog sets the range and style of the y axis for channels in time, result or XY views. The Channel field chooses one, all or selected channels (see the *Getting started* chapter for information about selecting channels). When you change the Channel field, the dialog contents update to display the settings for the selected channel. If multiple channels are selected, the settings are for the first channel in the list.

Click Optimise to draw the visible data at the highest magnification without clipping at the top and bottom. Click Show All to set the y axis to display the maximum possible range for waveform channels and from 0 to the estimated event rate for event channels drawn with a frequency axis. Both these buttons close the dialog. You can optimise without opening the dialog with the keyboard short-cut `Ctrl+Q` and by right-clicking on a channel and selecting the optimise option from the context menu.

The Draw button applies the y axis range set by the Top and Bottom fields to the channels set by the Channel field.

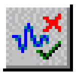
In normal use, you will let Spike2 organise the y axis style. However, when preparing data for publication you may wish to set the spacing between the major tick marks and the number of tick subdivisions. If you prefer a scale bar to an axis, you can select this in the Show/Hide channel dialog.

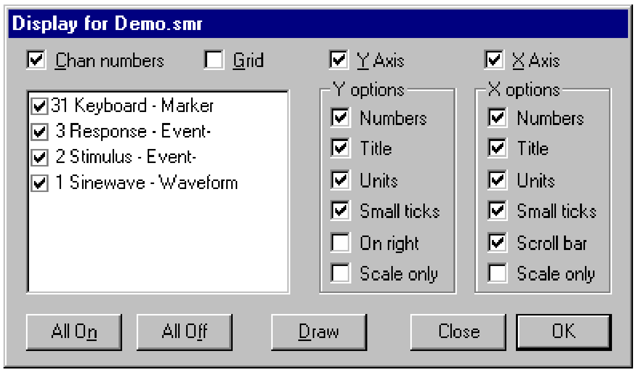
You can control the Large tick spacing (this also sets the scale bar size) and the number of Tick subdivisions by ticking the boxes. These check boxes will be grey if you choose more than one channel and these channels have different tick settings. You can set any tick spacing you like. However, your settings are ignored if they would produce an illegible axis. Changes made to these fields take effect immediately; there is no need to use the Draw button. Cancel undoes any changes and closes the dialog. The Close button closes the dialog.

Standard Display

This menu command sets the current time, result or XY view to a standard state. The x and y axes are displayed in a standard way with all user options for grids, tick spacing and special axis modes turned off. In time and result views it turns on all channels in a standard display mode and size and ordered as set in the Edit menu Preferences and all special channel colours are reset. In an XY view, all channels are made visible, the point display mode is set to dot at the standard size, the points are joined and the x and y axis range is set to span the range of the data.

Show/Hide Channel

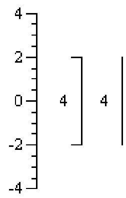




This dialog sets the channel list to display in time, result and XY views. It also controls the display of axes, grids and the horizontal scroll bar in time and result windows.

Check the Chan numbers box for channel numbers in time and result windows. The All On and All Off buttons select all or none of the channels, to save time when there is a long list. The Draw button updates the data window. You also have control over the x and y axis appearance. You can hide or display the grid, numbers on the axes, the big and small ticks and the axis title and axis units. You can also choose to show the y axis on the right of the data, rather than on the left.

For publication purposes, it is sometimes preferable to display axes as a scale bar. If you check the Scale only box, a scale bar replaces the selected axis. You can remove the end caps from the scale bar (leaving a line) by clearing the Small ticks check box. The size of the tick bar can set by the Large tick spacing option in the Y Axis Range or X Axis Range dialogs, or you can let Spike2 choose a suitable size for you.



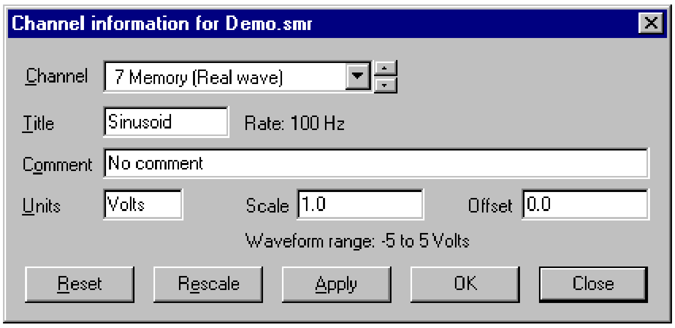
Info

This command displays information about the current result view window. In particular, it displays the number of sweeps that have been added into a PSTH, correlation or waveform average, the number of data blocks in a power spectrum, the number of cycles in a phase histogram and the number of intervals that have been processed to build an interval histogram (including intervals that fell outside the histogram).

File Information

This displays information about the current time view, including five lines of comments and the time and date it was created (if available). You can edit the comments unless the file is open in read only mode. You can cause this window to open automatically when sampling ends from the Automation tab of the Sampling Configuration dialog.

Channel Information



You can open this dialog in time window from the menu by a double-click on a channel title or from the channel context menu (right click on the channel). From this dialog you view and edit the channel information. Select the channel to view or edit with the Channel field. You can edit the Title and Comment for any channel. The remaining fields are hidden or displayed depending on the channel type. The Reset, Apply and OK buttons are disabled until you make a change to one of the fields. The Close button closes the dialog and does not apply any changes.

Changes made in this dialog have no effect on your data until you click the Apply button or OK (which is the same as Apply then Close). Changes to all fields except Title apply to all duplicates of the channel, to the original channel if the current channel is a duplicate and to the channel and its duplicates in any window duplicated from the current window. If you change channel without applying changes, any changes are lost.

Changes to the Title field are applied to duplicate channels if the duplicate has not been given its own title. If you set the title of a duplicate channel, it has no effect on any other channel. If you clear the title of a duplicate channel, it takes the title of the channel from which it was duplicated.

The Reset button restores any changes you have made to the channel settings unless you have used Apply. To undo applied changes, close the dialog and use the Edit menu Undo command (`Ctrl+Z`).

For waveform, RealWave or WaveMark channels, extra fields show the scale and offset values that convert between the 16-bit integer data used for waveform and WaveMark data and user units and that convert RealWave values to integer, when required.

Real value in user units = 16-bit value * scale / 6553.6 + offset

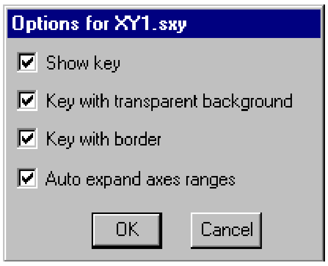
There is more about scaling in the documentation for the sampling configuration dialog. Both the scale and the offset must be smaller than 10 billion. The scale may not be set to 0 or very close to 0. If the channel scale or offset is edited into an illegal state, a warning message appears in the dialog and you cannot Apply the changes. If you want to calibrate a channel, it is often easier to use the Calibrate option of the Analysis menu. When the scale and offset fields are present, the range of 16-bit data is also displayed.

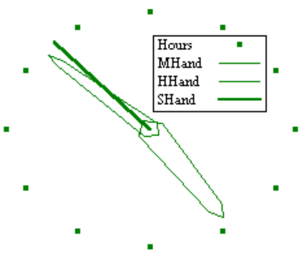
The Rescale button appears for RealWave channels. Click it to set the scale and offset fields so that the full range of data could be represented by 16-bit data. The offset is set to 0 if this does not lose too much precision. Some routines in Spike2 treat RealWave data as 16-bit integer, and the scale and offset you set determine how the conversion from 32-bit real to integer is done.

The Units field is present for waveform, RealWave, WaveMark and RealMark channels.

Options

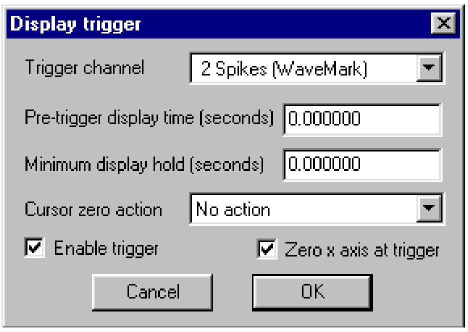
This command is for XY windows only and opens the XY options dialog. This dialog controls the XY window "key". The key is a small region to identify the data channels that you can drag around within the XY window. For each visible channel it display the channel name and draws the line and point style for the channel. This dialog also has a checkbox that controls the automatic expansion of the axes when new data is added. The equivalent script language command is `XYKey()`.





This example (made by the `clock.s2s` script in the `Scripts` folder) shows the key. You can choose to make the key background transparent or opaque and choose to draw a border around the key. If you move the mouse pointer over the key, the pointer changes to show that you can drag the key around the picture. If you double click the key, the Options dialog opens.

Display Trigger



The display trigger is used with Time views to provide an oscilloscope style trigger and paged display on-line, and a means of easily moving back and forward to the next or previous trigger both on-line and off-line. The script language equivalent of this command is `ViewTrigger()`.

The Trigger channel field sets an event, Marker, WaveMark, TextMark or RealMark channel to be used as the trigger. You can select No channel for a permanent trigger (for on-line paged displays).

The Pre-trigger display time field sets the time before the trigger to show each time a trigger occurs. This dialog does not set the width of the time view; that is set by the normal time view mechanisms. If you set the pre-trigger display time larger than the displayed time view width or negative, the trigger point will not be visible. Negative pre-trigger times move the trigger point off the screen to the left of the display.

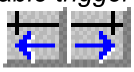
The Minimum display hold field is used on-line and sets the minimum time that data is displayed. This allows you to see individual data frames with a high frequency trigger.

The Cursor zero action field has three setting that control what happens to cursor 0 and any active cursors that depend on it when the view triggers:

No action	Cursor 0 state is unchanged.
Move to trigger	Cursor 0 moves to mark the trigger point, active cursors do not move.
Move and iterate	Cursor 0 moves to mark the trigger point, active cursors move.

The Zero x axis at trigger checkbox changes the x axis so that 0 lies at the trigger time. This is purely a visual convenience; all measurements are still in the original x axis units.

Enable trigger



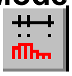
If you check the Enable trigger checkbox and click OK, two extra buttons appear in the area to the left of the x axis scroll bar. These buttons can be used on-line and off-line to move backwards and forwards by a trigger. With No channel set, they move by the time between the trigger point and the right hand screen edge.

On-line

To use the display trigger on-line, make sure you are at the end of the file. The display will scroll until a trigger is found when it will hold with the pre-trigger time shown before the trigger until another trigger occurs. The hold time will be at least as long as the minimum display hold time requested. If you move away from the end of the file with the scroll bar or the two trigger buttons, the automatic window update will cease until you drag the x axis scrollbar thumb back to the end of the file.

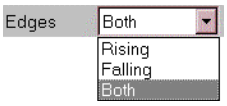
With No channel selected, the display is “paged”. Each time the right-hand screen edge is reached and the hold time has passed, a new sweep starts. The pre-trigger time sets the overlap between the sweeps.

Channel Draw Mode



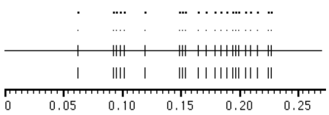
This menu item is available when the current window holds a Spike2 data document or result view and sets the data channels display mode. You can set the mode for a single channel, all channels, or selected channels. The dialog changes, depending on the channel type and display mode. The Draw button is common to all modes and updates the display without closing the dialog box.

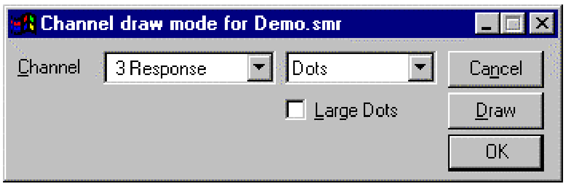
Time view drawing modes



In some modes the Edges field appears if you choose a channel of event level data. You can select Rising, Falling or Both edges of the data. If you select both edges, then the display modes that show frequency count both the rising and falling edges of the event signal in their rate calculations. You would normally count only one edge, so select the edge you prefer.

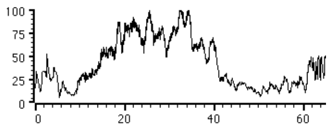
Dots and Lines mode

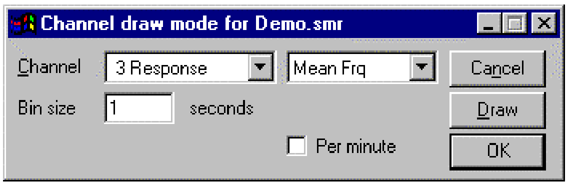




The simplest event channel draw method is dots. You can choose large or small dots (small dots can be very difficult to see). You can also select Lines in place of Dots. The picture to the left shows the result of both types of display. If you select lines for a marker channel, the display of markers is suppressed. You can also select Dots mode for a waveform channel. In Lines mode you have the option to suppress the central horizontal line.

Mean frequency





The mean frequency is calculated at each event by counting the number of events in the previous period set by Bin size. The result is measured in units of events per second unless the Per minute box is checked. The mean frequency at the current event time is given by:

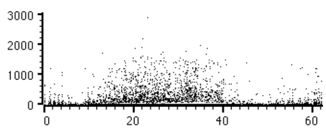
$$(n-1)/(t_e-t_1) \quad \text{if } (t_e-t_1) > t_b/2$$

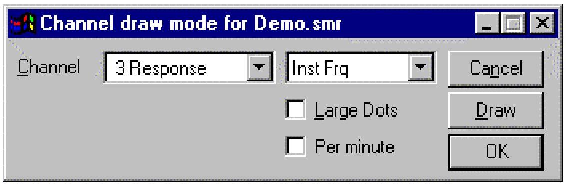
$$n/t_b \quad \text{if } (t_e-t_1) <= t_b/2$$

where: t_b is the bin size set; t_e is the time of the current event; t_1 time of the first event in the time range; n is the events in the time range

A constant input event rate produces a constant output until there are less than two events per time period. You should set a time period that would normally hold several events.

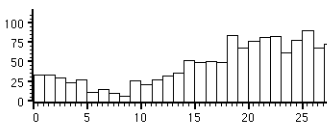
Instantaneous frequency

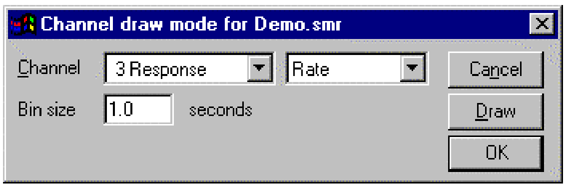




Instantaneous frequency is formed by plotting the inverse of the time interval between an event and the previous one on the same channel. You can display each event as a small or a large dot. You can also display the frequency as events per minute rather than per second.

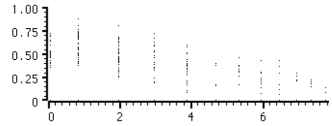
Rate histograms

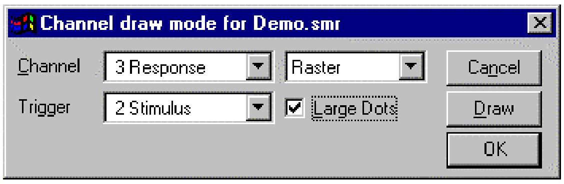




Rate mode counts how many events fall in each time period and displays the result as a histogram. The result is not divided by the bin width. This form of display is especially useful when the event rate before an operation is to be compared with the event rate afterwards.

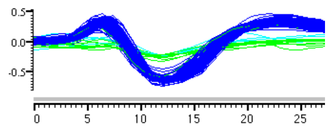
Raster display





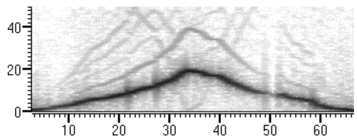
Raster mode shows the event pattern with respect to a trigger channel. Each trigger event defines time 0 in the y direction for the sweep. The Y Range dialog sets the time range to display in the y direction; negative times show pre-trigger events. For each trigger, events are drawn no further back in time than the previous trigger and no further forward in time than the next.

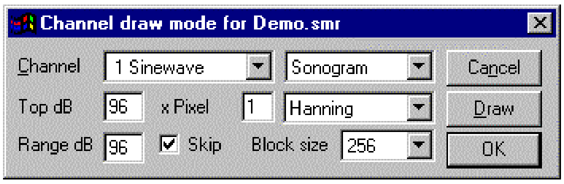
Overdraw WM



Overdraw WaveMark mode draws WaveMark data as superimposed waveforms. Channels drawn in this mode are moved to the top of the window and separated from the x axis (which does not apply to them) by a grey bar. 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). Click on the x axis scrollbar thumb to force an update.

Sonogram display





Sonogram mode shows how the frequency content of a waveform channel changes with time. The y axis units are Hz (frequency) and useful results are available for the frequency range 0 to one half of the sampling rate for the waveform channel. The intensity of the frequency content is indicated by a grey scale, the darker the image, the more intense the signal. You can set:

Top dB — This sets the signal intensity that corresponds with the darkest grey scale. dB means decibels, which is a logarithmic measure of ratio, usually of amplitudes or power. 20 dB is a factor of 10 in amplitude. Spike2 stores waveform data as 16-bit integers, and we measure the amplitude with respect to 1 bit, so 96 dB is the maximum possible level.

Range dB — This sets the range of data to display as a grey scale. Signals with an intensity of Top dB - Range dB (or less) are displayed as white. If you are unsure what dB values to set for a new signal, setting values of 96 dB for both Top dB and Range dB will display something in almost all cases!

x Pixel — You can speed up drawing, at the expense of resolution, by setting this field to values greater than 1. It sets the number of screen pixels in the x direction to calculate at a time. A value of 1 gives the best visual resolution (and the slowest calculation and drawing time).

Window — The sonogram is calculated using a Fast Fourier Transform. As explained in the analysis section for the Power spectrum, it is important to apply a “data window” to the signal before taking the power spectrum, otherwise the results are very difficult to interpret. We provide several different types of window: None, Hamming, Hanning, Kaiser 30dB, Kaiser 40dB, Kaiser 50dB, Kaiser 60dB, Kaiser 70dB, Kaiser 80dB and Kaiser 90dB.

All windows are a compromise between increasing the apparent width of a spectral peak and the ability to see small signal peaks in the presence of large ones. If you apply no window, you will get the sharpest resolution of a single peak. However, you will not be able to see any small peaks around it due to the “sidelobes” of the window. If you are not familiar with the use of windows, the Hanning window is a reasonable compromise. The Kaiser *xx*dB family of windows has the property that the largest sidelobe is *xx*dB below the peak. Of course, the larger the *xx*dB, the wider the peaks become.

Block size — This determines the number of data points used in the FFT, and thereby it determines the frequency resolution. Like the choice of data windows, this is also a compromise. The larger the block size, the better the frequency resolution, but the worse the time resolution. If you are looking for a short, localised burst of changing frequency, you will need to use a block size that is smaller than the duration of the episode you are looking for.

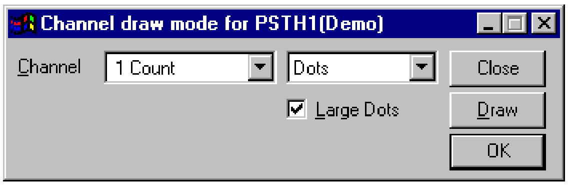
Skip — If you are analysing a lot of data, there can be many thousands (or even tens of thousands) of data points for each screen pixel. If you check this box, the sonogram will only analyse the first Block size points for each pixel (normally the sonogram analyses all the points). This can save a lot of time if you have a really large file. Of course, the result will only represent a "sampling" of the correct response.

You can export Sonograms as enhanced metafiles and as bitmaps. Sonogram printing is supported with Postscript compatible printers. With other printer types, the grey scale output may be quite coarse. You may obtain better output by saving the sonogram as a bitmap and printing from specialist bitmap editing programs.

Waveform, WaveMark and Cubic spline

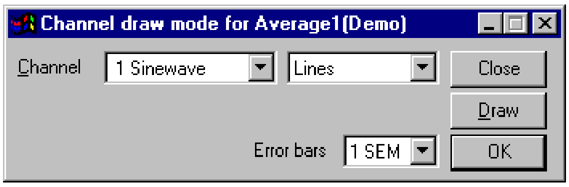
These modes apply to waveform and WaveMark channels. Waveform mode joins the data points with straight lines. Cubic spline mode joins the points with a smooth cubic spline that is probably a better guess at the data values between the data points than a straight line as long as the data was sampled fast enough. However, you must always remember that the only data values that you can rely on are those at the sample points. Channels drawn in Cubic spline mode will not appear in a Windows metafile output as this format is not supported. WaveMark is for WaveMark channels only and is the same as waveform mode but also draws the first marker code.

Result view drawing modes



There are six drawing modes for result views: Histogram, Line, Dots, SkyLine, Raster and Raster lines. The last two can be used if you checked the Raster data box when you created the result view. In Dots mode you can also choose to display large dots.

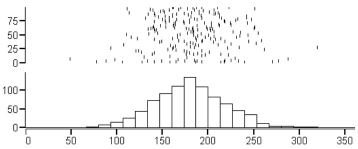
Error bars

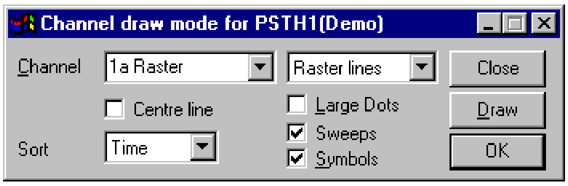


If your result view has associated error information, for example a waveform average with error bars enabled, you have an extra control. You can choose from None, 1 SEM, 2 SEM or SD. It should be emphasised that error bars only have meaning if the data points that contribute to the average have a normal distribution about the mean. Given this, then 1 SEM shows ±1 standard error of the mean, 2 SEM is ±2 standard errors of the mean and SD is ± 1 standard deviation.

If each point of your data can be modelled as a constant "real" value to which is added normally distributed noise with zero mean, then you would expect the measured mean value to lie within 1 *standard error of the mean* (SEM) 68% of the time, or within 2 SEM 95% of the time. The *standard deviation* represents the width of the normal distribution of the underlying data at each data point.

Result view raster





If a result view channel has associated raster data, there are two more drawing modes: Raster and Raster lines. Raster mode shows each event as a dot, Raster lines mode shows each event as a short vertical line. The Large Dots check box increases the dot size and vertical line length. In Raster lines mode you can check the Centre line box for a horizontal line through each sweep.

Extra features with auxiliary values

If you have set an auxiliary channel in the peri-stimulus histogram, event correlation or phase histograms setup dialog or set auxiliary values from the script language, you have a choice of the order in which the sweeps are presented. The Sort field can be set to Time, Aux 0 or Aux 1. Time presents the sweeps in the order that they were recorded. Aux 0 and Aux 1 sort the sweeps based on the value of auxiliary values 0 and 1. Auxiliary value 0 is set if you set an auxiliary channel in the result view setup dialog. The script language can set all auxiliary values with the `RasterAux()` command.

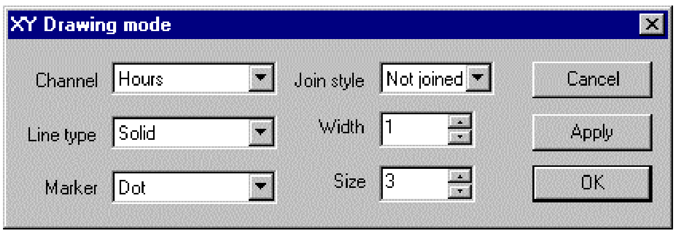
If you check Sweeps, the y axis is a sweep count and all sweeps are evenly spaced in the y direction, otherwise it is the value of the item selected by the Sort field and the sweeps are spaced out based on the value set in the Sort field. If you sort by Time, the sweep number is the order in which sweeps of data were added into the result view.

Each raster can store times in auxiliary values 2 to 5 that can be displayed as four symbols (circle, cross, square and triangle) on the raster. Symbols appear for times that lie within the x axis range. Auxiliary value 2 is set for you if you selected an event channel as the auxiliary channel in the result view setup dialog.

Interrupting drawing

Despite our best efforts to draw huge data files quickly, drawing can take a long time. You can interrupt it with the `Ctrl+Break` key combination. The `Break` key is usually at the top right of the keyboard and is often labelled `Pause` with `Break` on the front. If your system has sound enabled and you have selected a sound for “Exclamation”, Spike2 plays this sound to confirm that drawing has been abandoned for the current channel.

XY Draw Mode

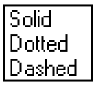


This command sets the drawing style of XY window data channels. OK makes changes and closes the dialog, Apply makes changes without closing the dialog. The Cancel button closes the window and ignores any changes made since the last Apply. The Channel field sets the channel to edit, or you can select All channels. If you change channel, the dialog remembers any alterations so there is no need to use the Apply button before changing channel unless you want an immediate update. The other fields are:

Join style

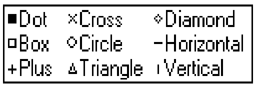
You can set three join styles for a channel. In Not joined style, no lines are drawn between data points. In Joined style, each point is linked to the next by a straight line. In Looped style, the points are joined and the final point is linked back to the first point. The Line type and Width fields set the kind of line that joins the points.

Line type and Width

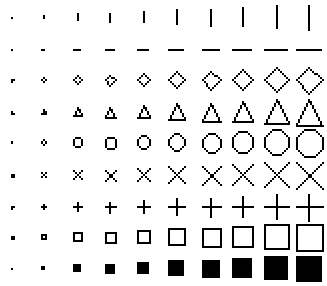


These two fields set the type of line joining data points. The Width field determines how wide the line is in units of half the data line width set by the Edit menu Preferences dialog. Set 2 for the normal line width, 1 for half this width. If the width in pixels is greater than 1, the Line type field is ignored and the line is drawn as a solid line.

Marker and Size

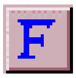


The Size field sets the marker size in units of Points (a point is 1/72nd of an inch or approximately one screen pixel). A size of 0 makes the markers invisible. You can set sizes up to 100 points. There is a wide range of marker styles to choose from.

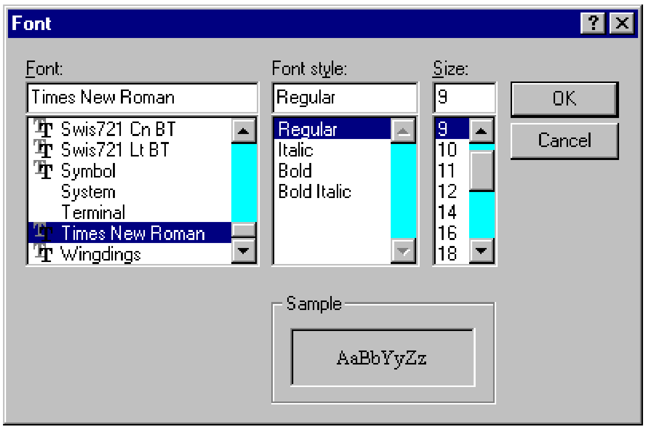


The picture to the right shows a screen dump of all the marker styles in sizes 1 to 10. If you need to tell them apart on screen, sizes below 3 should be avoided. If you have excellent eyesight and a high-resolution printer, size 1 is viable in printed output.

Font



You can select the font that is used for each window in Spike2. The font size changes the space allocated to data channels in a time, result or XY view. Smaller fonts give more space to the channels, however fonts need to be large enough to read! You can set different fonts in each data or text window.

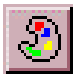


You can select only non-proportionally spaced fonts for text views as the text editor does not support proportionally spaced fonts. If you do manage to set a proportionally spaced font (for example through the script language `FontSet()` command), the result will be messy.

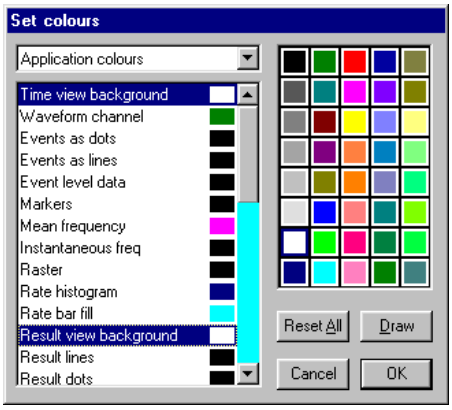
Use Colour and Use Black And White

If you have a monochrome monitor or if you have to use a printer which does not handle colour well, you can choose to display your data files in black and white. The Use Black And White menu item switches from colour to black and white displays. If you change to monochrome, the menu item changes to Use Colour. This option saves you from the tedious task of changing every colour in the colour palette manually.

Change Colours



You can choose the colours that are used for almost everything in Spike2. If you open the dialog with an active time, result or XY view, the dialog has multiple pages. Select a page with the drop-list at the top left. Pages are:



Application colours

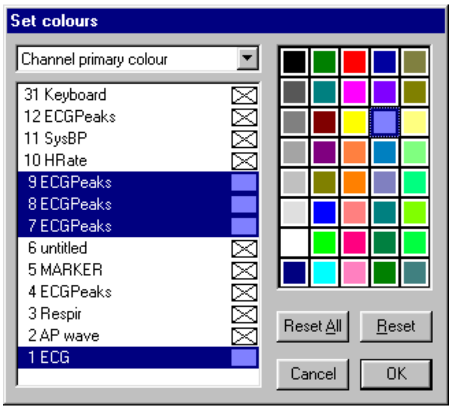
To change colours, select one or more items in the list on the left, and then click a colour in the palette on the right. You can check the result of your action with the Draw button. Cancel removes the window and undoes any changes. OK accepts changes and closes the dialog. The Reset All button returns the list and the palette to a standard set of colours. You can change the following:

Time view background	Instantaneous freq	Result skyline	Grid
Waveform channel	Raster	Result histogram	Axes
Events as dots	Rate histogram	Result bar fill	XY view background
Events as lines	Rate bar fill	WaveMark 0-8	Not saving to disk
Event level data	Result background	Text labels	Standard deviation/SEM
Markers	Result lines	Cursors	
Mean frequency	Result dots	Controls	

See the Edit menu Preferences for text view colours. The special colour, Not saving to disk, is used to draw data displayed in a sampling window that is not being written to disk. This colour is only used when the most recent data is drawn at the right hand edge of the window (when the scroll bar thumb is at the far right of the time view). Set this colour the same as Time view background to use the normal colours. WaveMark data that matches a template does not change colour.

Channel primary colour
Channel secondary colour

These two pages assign colours to data channels in the current time or result view and override the colours set for drawing modes. The primary colour is used as the drawing colour for lines, waveforms, events, and histogram outlines. The secondary colour is for filling histograms. An X in a box marks a channel with no colour override.

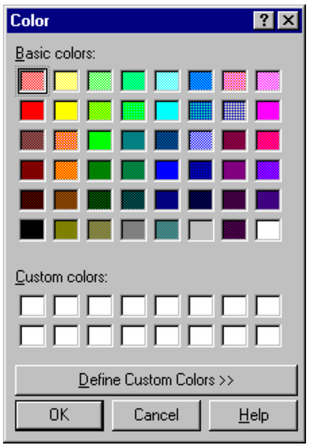


Changes made on this page are applied immediately, so there is no Draw button. You can Reset the selected channels back to the standard colours set in the Application colours page.

XY channel colours

This page lets you change the XY background and channel colours. Changes made on this page are applied immediately; there is no Draw button.

Changing the Palette

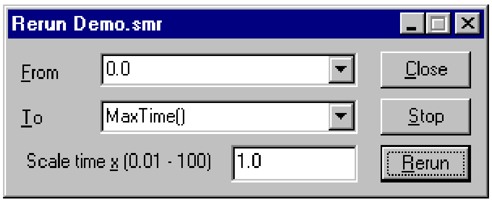


To change a palette colour, double click it to open the colour dialog and select a replacement colour. The first seven colours form a grey scale from black to white and cannot be changed.

You can replace the palette colour with any standard colour, or you can click the Define Custom Colours... button to select an arbitrary colour. Click OK or Cancel to exit.

The colour selection applies to all data files. It is stored with the Spike2 configuration in `default.s2c`, not in the data files. When you restore a Spike2 data file, the colours will be those that are currently active, not those in use when you last saved it.

ReRun



This option can be used in a Time window to simulate data sampling for demonstration and training purposes. The From and To fields define the time range to use. The Scale time field sets the number of seconds of data to rerun every second. The Rerun button starts the simulation; the Stop button cancels it. Unlike real sampling, there is no sample control window and the output sequencer and on-line waveform replay are not available.

When the update point of the document reaches the right-hand edge of the window, the window scrolls to keep the update point at the right-hand edge of the screen. You can use the scroll bar to move through the document, but only up to the current end. If you move the scroll bar fully to the right, the window will scroll automatically, if the scroll bar is not at the right, the window will not scroll, but the scroll bar position will change as time passes to show its relative position in the document. The document will keep running until it reaches the end or you use the Stop button.

In place of scrolled displays, you can have a paged update. Open the View menu Display Trigger dialog, select None as the trigger channel, check Enable trigger and click OK.

Analysis menu

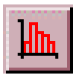
The Analysis menu is divided into several regions, all associated with the analysis of data or the processing of data into a different form. The first region holds commands that are associated with processing data from a time view into a result or XY view. The New Result View command opens a pop-up menu from which you can choose an analysis that generates a result window, for example a waveform average of one or more channels. The Measurements command is used to generate an XY graph from measurements based on cursor positions. The Process and Process Settings commands allow you to modify analyses created with the other two commands.

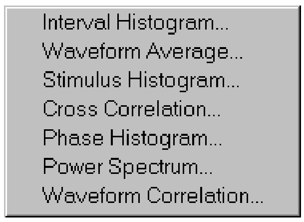
The second region manages time view memory buffers and general channel operations. A memory buffer is like any other data channel in a time view except that it is held in computer memory and will disappear when the data file closes. It can hold any type of data that can be stored in a time view. Data can be added to memory buffers interactively, or you can import sections of other channels. You can save memory channels as part of the data document. This section also holds commands to calibrate and process time view waveform and WaveMark channels, duplicate time and result view channels and delete data channels.

The third region holds commands to set the marker filter and process WaveMark data. The WaveMark commands are most useful when processing spike shapes, but they can be used for other tasks, such as detecting features (for example R waves in an ECG signal). The marker filter can also be used to isolate specific event markers, for example for conditional averaging.

The last region holds the digital filter command. You can both generate and apply Finite Impulse Response digital filters to time view waveform channels to generate new memory channels or permanent channels in the data file.

New Result View





This command is available when the current window is a time view. It is a pop-up menu from which you can select an analysis type. This leads into a dialog where you define the parameters to construct a result window.

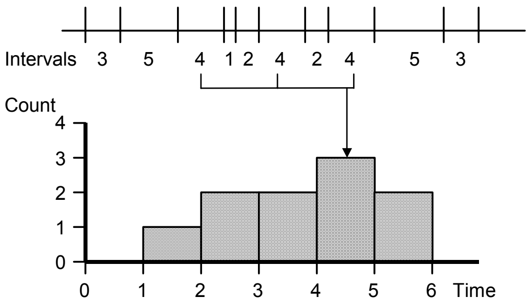
A result window holds arrays of data that can be drawn as a histograms or as waveforms. There is one data array for each channel you analyse. Some types of analysis can store additional data, for example the peri-stimulus histogram can store event times for a raster display and the waveform average can store extra data so that you can display error bars in the result.

Once you have set the required values in the dialog box, Spike2 creates a result window with all data values set to zero. A new dialog appears to prompt you to select a region of the original time window to analyse. The results of analysing different areas can be summed.

At the time of writing, the following analyses are implemented: interval histogram, waveform average, peri-stimulus time histogram, event time cross correlation, event phase histogram, power spectrum and waveform correlation.

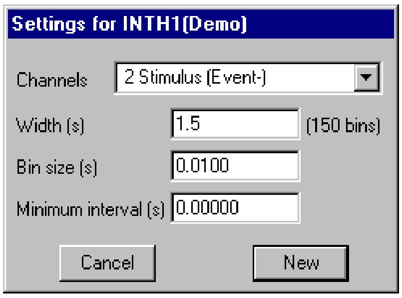
Interval histogram

An interval histogram (INTH) displays the frequency histogram of the intervals between events on a selected channel. All times in a Spike2 data document are expressed as a multiple of the basic time unit set when the data document was created. Therefore you cannot get better time resolution than the basic time unit (usually in the range 2 to 50 µs).



The picture illustrates how an interval histogram is formed. To make the diagram easy to understand, we have shown several events with all the intervals between them as a whole number of seconds. The interval histogram formed from these events has bins set with a width of one second each.

The interval between each pair of events is divided by the bin width to give the bin number to increment (fractional bin numbers are rounded down, the first bin is bin 0). In this case, the bin width is 1, so there is no rounding. In a more realistic case with an interval of 2.3 milliseconds and a bin width of 1 millisecond, bin number 2 (spanning the time period 2 to 3 milliseconds) would be incremented.



Event times are rounded down to the nearest base time unit, so an event time of t base units means that the actual time was greater than or equal to t and less than t+1 units. An interval of n units means that the real time interval between two events was greater than n-1 units and less than n+1 units. If you need to form interval histograms of very short periods, make sure that you have set the duration of the base time unit short enough to resolve the information you require.

The Channel field selects one or more event channels; type a list of channels or select from the drop down list. Each channel generates a histogram in the result view. If you type a channel list or use the selected channels, the order of the channels in the result view is the order of channels as you enter them.

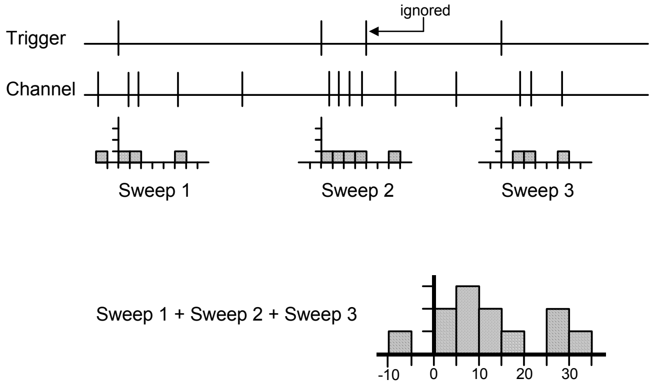
Width sets the time width of the histogram from left to right. The Minimum interval field is the smallest interval to put in the histogram and sets the left hand end of the x axis. It must be positive and is usually zero. Bin size is rounded to the nearest multiple of the underlying time units used in the time window. The number of bins (Width/Bin size) must be at least 1. The maximum size is limited only by available system memory.

The New button accepts the values in the dialog and creates a new result window. This dialog can also be activated by the Process Settings menu command, in which case the New button is replaced by a Change button. Change clears any previous results.

Click the New or Change button to open the Process dialog (see page 9-11) in which you set the time range of data to analyse. Only events that fall within the time range are considered, intervals that start or end outside the time range are ignored.

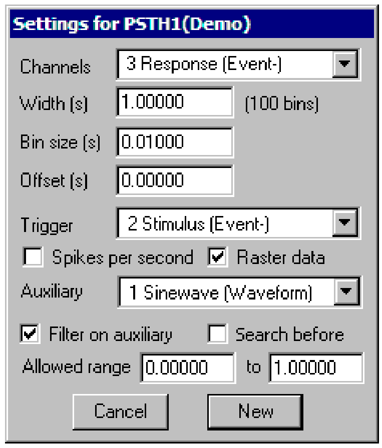
Peri-stimulus time histogram

A peri-stimulus time histogram (PSTH) forms a histogram of events on one channel around a stimulus event on another trigger channel. The drawing illustrates the general principle of the PSTH.



Each trigger event that does not lie within the previous sweep generates a new sweep. Trigger events that fall in the previous sweep are ignored. For overlapped sweeps, use the event correlation analysis. Sweeps are accumulated to produce a histogram. You can scale the result into spikes per second rather than event counts.

In the Settings dialog, the Channel field selects the channels to analyse. The Width and the Bin size fields set the number of bins in the histogram. The Offset field sets the pre-trigger time to display. If the time offset is 0.5, the histogram time axis starts at -0.5. All times are rounded to the nearest multiple of the basic time unit.



The Trigger field selects one data channel to use as the trigger, or you can select no trigger channel. If you do not select a trigger channel, the process command uses the start time as the trigger time for a single sweep. If you select a trigger channel, the process command defines a time range to search for trigger events.

Spikes per second

If you check the Spikes per Second box, the histogram shows equivalent spike rate rather than the number of events in each bin. To change from event count to rate, Spike2 divides the number of spikes in each bin by the number of sweeps to give the spikes per sweep, and then divides the result by the bin width to give spikes per second.

You can change between count and spikes per second after you have created the result view and analysed data. Click on the result view, open the Process settings dialog and change the state of the Spikes per second check box, then click Change.

Raster data

If you check the Raster data box, the new result view stores the times of all the events that are added into histograms and you have the option to draw the result as a raster display. When you create the result view, Spike2 automatically duplicates each result channel and draws the duplicated channel in raster drawing mode above the histogram. As each event uses memory and takes time to draw, the maximum number of events is limited by the memory in your system and your patience. This should not be an issue unless you work with hundreds of thousands of spikes with the result view raster enabled.

Once you have created the result window and analysed data you can delete the raster data by opening the Process settings dialog and clearing the Raster data check box. You cannot add raster data to an existing result view without clearing all the histograms.

You must enable Raster data if you want to take measurements from an Auxiliary channel.

Auxiliary measurements

Each raster sweep has six auxiliary values. From the Draw Mode dialog you can sort sweeps based on auxiliary values 0 and 1 and draw symbols at times in values 2 to 5. Set a channel with the Auxiliary field to enable auxiliary measurements. You can access all the auxiliary values with the `RasterAux()` script command.

If you choose an event or marker derived channel, auxiliary value 0 is set to the time between the trigger point of the sweep and the first event on the auxiliary channel after or before the trigger. The search direction is determined by the Search before check box. Times before the trigger are negative, times after it are positive. To allow you to show the event as a symbol, the event time is also saved as auxiliary value 2. To make sure that sweeps can be sorted, if no event is found, we create special events at time 0 and at the maximum time possible in the file. Special event times are not saved to value 2.

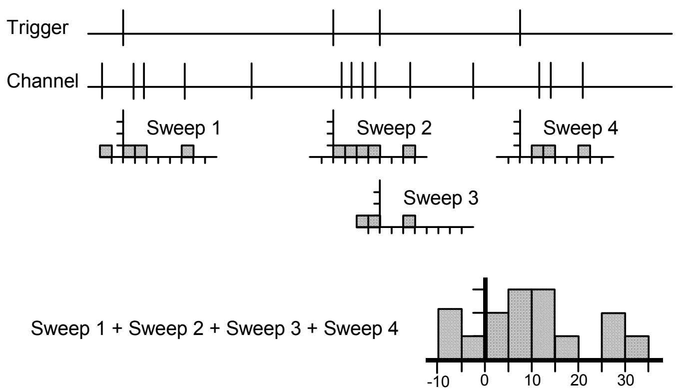
If the auxiliary channel is waveform data, the auxiliary value 0 measurement is the channel value at the trigger point. If there is no data, the value 0 is used for sorting.

Filter on auxiliary

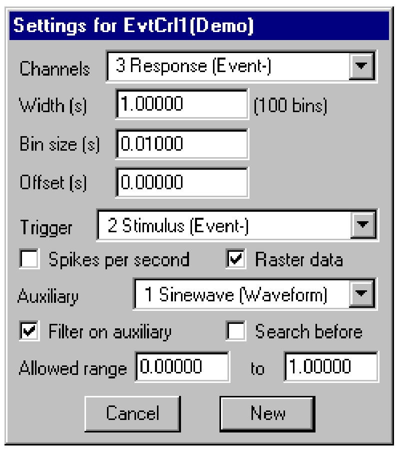
When auxiliary channel measurements are enabled you can also choose to filter sweeps based on the measured value. Sweeps with an auxiliary value that lies outside the values set by the Allowed range fields or for which no data is found are rejected; their data is not included in the raster display or the histogram.

Event correlation

This command performs event cross-correlations and event auto-correlations between a trigger channel and one or more other event channels. A cross-correlation produces a measure of the likelihood of an event on one channel occurring at a time before or after an event on another channel. The result can be displayed as a spike rate or as a count.



The drawing illustrates the general principle of an event correlation. Every trigger generates one sweep of analysis (unlike a PSTH where a trigger event that falls within a sweep is ignored). The zero times of each sweep are aligned, and then the sweeps are accumulated to form a histogram.



The Channel field sets a channel or a channel list to analyse. The Width and the Bin size fields set the number of bins in the histogram. The Offset field sets the pre-trigger time to display, so if the time offset is offset, the histogram time axis starts at a time of -offset. All times are rounded to the nearest multiple of the basic time unit.

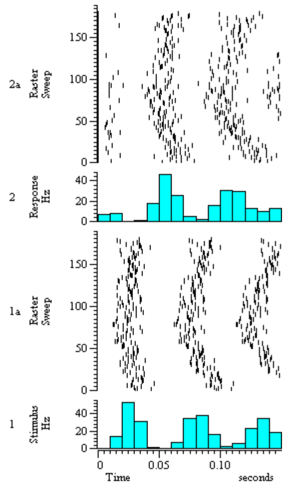
The Trigger field is a drop down list in which you can select a trigger channel or Manual mode. In Manual mode, the process command start time is the trigger time for one sweep. With a trigger channel, the process command sets a time range to search for trigger events.

Spikes per second

If you check the Spikes per Second box, the histogram shows equivalent spike rate rather than the number of events in each bin. To change from event count to rate, Spike2 divides the number of spikes in each bin by the number of sweeps to give the spikes per sweep, and then divides the result by the bin width to give spikes per second.

Raster data

If you check the Raster data box, the new result view stores the times of all the events that contributed to the histogram and you can use raster drawing mode for the result view channel. When you create the result view, Spike2 duplicates each channel and draws the duplicate in raster mode above the histogram.



As each event uses memory and takes time to draw, the maximum number of events is limited by the memory in your system and your patience. This should not be an issue unless you work with hundreds of thousands of spikes with raster enabled.

Auxiliary measurements

Select a channel with the Auxiliary field to enable auxiliary measurements. Each raster sweep has six auxiliary values numbered 0 to 5. From the Draw Mode dialog you can sort sweeps based on auxiliary values 0 and 1 and show symbols based on times held in auxiliary values 2 to 5. You can access all the auxiliary values with the `RasterAux()` script command.

If you choose an event or marker derived channel, the time between the trigger point of the sweep and the first event on the auxiliary channel after or before the trigger is stored in auxiliary value 0. The search direction is set by the Search before check box. Times before the trigger are negative, times after it are positive. To allow you to show the event as a symbol, the event time is also saved as auxiliary value 2. To make sure that sweeps can be sorted, if no event is found, we create special events at time 0 and at the maximum time possible in the file. Special event times are not saved to value 2.

If the auxiliary channel is waveform data, the auxiliary value 0 measurement is the channel value at the trigger point. If there is no data at the trigger point, the value 0 is stored for sorting.

Filter on auxiliary

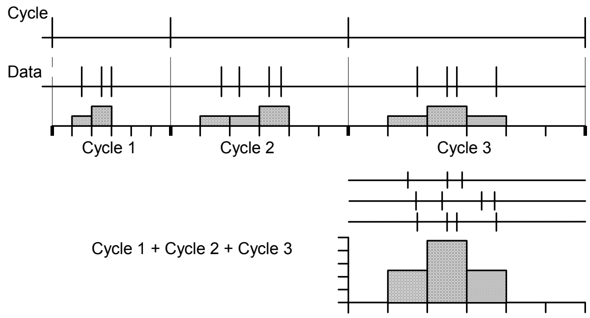
When auxiliary channel measurements are enabled you can also choose to filter sweeps based on the measured value. Sweeps with an auxiliary value that lies outside the values set by the Allowed range fields or for which no data is found are rejected; their data is not included in the raster display or the histogram. The allowed range units depend on the auxiliary channel type. When the auxiliary channel holds waveform or RealWave data the units of the range fields are the channel units, for all other types the units are seconds.

Auto-correlation

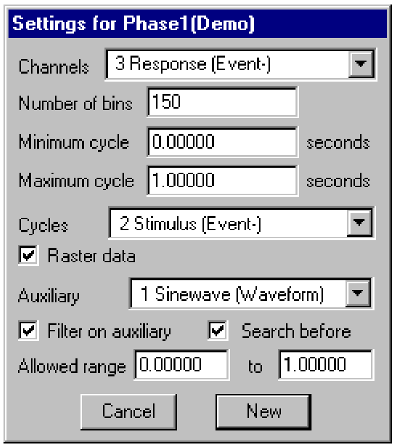
For an auto-correlation, set the Channel and Trigger fields to the same channel. The analysis for an auto-correlation has one difference from that for a cross-correlation. In an auto-correlation, the correlation of each event with itself at time 0 in the histogram is ignored.

Phase histogram

A phase histogram is used to show how events are distributed with respect to a cyclical process that may vary in cycle time. One event channel marks the start and end of cycles. The end of one cycle is the start of the next. Events on another channel are placed in bins depending upon their position within each cycle.



The Channels field sets the channels to analyse. The Cycle channel sets the channel with cycle markers. The Number of bins field sets the bins per cycle. The width of each bin depends on the duration of each cycle. The Minimum cycle time and Maximum cycle time fields exclude cycles that are too long or too short to belong to the data and are set in seconds. Check the Raster data box to save the times of all events and duplicate each result channel in raster mode.



In the Process command, the start and end times determine the time range of the data in the Cycle channel to search for cycle markers. Cycles that are longer than the maximum time or shorter than the minimum time are ignored. If no Cycle channel is supplied, the start and end times are used as the start and end of a single cycle.

Auxiliary measurements

Set the Auxiliary field to a channel to enable auxiliary measurements. Each raster sweep has six auxiliary values. You sort sweeps based on auxiliary values 0 and 1 and show symbols based on values 2 to 5 with the Draw Mode dialog. You can access all six values with the `RasterAux()` script command.

If you choose an event or marker derived channel, auxiliary value 0 is set to the cycle position of the first event on the auxiliary channel after or before the cycle start. The Search before check box sets the search direction. Positions before the cycle start are negative, positions after it are positive. The cycle start position is 0 degrees, the cycle end position is 360 degrees. If an event is at time *Te* and the cycle start and end times are *Cs* and *Ce*, the event position is 360*(*Te*-*Cs*)/(*Ce*-*Cs*).

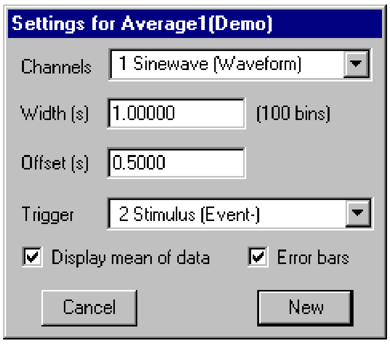
The time of the event is stored in auxiliary value 2 to allow you to draw it as a symbol. To make sure that sweeps can be sorted, if no event is found, we create special events at time 0 and at the maximum time possible in the file. These special events are not saved in auxiliary value 2.

If the auxiliary channel is waveform data, the auxiliary value 0 measurement is the channel value at the trigger point. If there is no data, the value 0 is used for sorting.

Filter on auxiliary

When auxiliary channel measurements are enabled you can also choose to filter sweeps based on the measured value. Sweeps with an auxiliary value that lies outside the values set by the Allowed range fields or for which no data is found are rejected; their data is not included in the raster display or the histogram. When the auxiliary channel holds events or markers, the allowed range is in degrees, otherwise it is in the auxiliary channel units.

Waveform average



This command averages waveform channels with respect to a trigger signal, or if no trigger is set, averages waveform channels with respect to user-defined trigger times. The Width field sets the time range spanned by the average. The Offset field sets the start time of each sweep with respect to the trigger. The Trigger field sets the source channel for trigger times or Manual trigger mode.

The Display mean of data chooses between a display of the mean of all the sweeps or a display of the sum of all the sweeps. If you return to this dialog after forming an average you can change this setting to redraw the result without needing to process the data again.

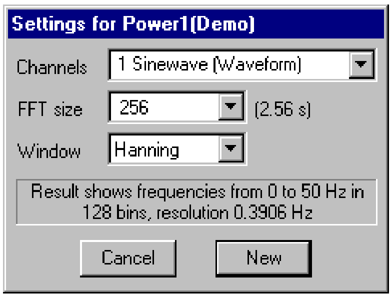
If you check the Error bars box, extra information is saved with the result so that you can display the standard deviation and standard error of the mean of the resulting data. The Channel Draw Mode command controls the display of the error information.

The New button (or Change when used from the Process Settings command) closes the dialog, creates a new result window and opens the Process dialog, described below. With a trigger channel, the events between the start and end times set in the process dialog are triggers. In Manual mode, the start time is the trigger for a single sweep.

The result window for an average keeps track of the number of data sweeps that have been added into the average, and it displays the mean waveform by accumulating the data and dividing by the number of sweeps.

The section of data to add to the average is found by subtracting the time offset from the trigger time, then taking the waveform section that starts at or within one waveform sample period of that time. If there is a gap in the waveform data, such that there is no data point that meets this condition, no data is added.

Power spectrum



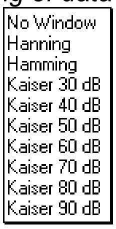
This command creates a result window that holds the power spectrum of a section, or sections of data. The result of the analysis is scaled to RMS power, so it can be converted to energy by multiplying by the time over which the transform was done. You can transform multiple channels, but they must have the same sample rate. Spike2 uses a Fast Fourier Transform (FFT) to convert the waveform data into a power spectrum.

The FFT is a mathematical device that transforms a block of data between a waveform and an equivalent representation as a set of cosine waves. The FFT that we use limits the size of the blocks to be transformed to a power of 2 points in the range 16 to 16384. You set the FFT block size from a drop down list. The result window ends up with half as many bins as the FFT block size. When you use the Process command, the selected area must hold at least as many points as the block size, otherwise no analysis is done.

The result window spans a frequency range from 0 to half the sampling rate of the waveform channel. The width of each bin is given by the waveform channel sampling rate divided by the FFT block size. Thus the resolution in frequency improves as you increase the block size. However, the resolution in time decreases as you increase the block size as the larger the block, the longer it lasts.

The mathematics behind the FFT assumes that the input waveform repeats cyclically. In most waveforms this is far from the case; if the block were spliced end to end there would be sharp discontinuities between the end of one block and the start of the next. Unless something is done to prevent it, these sharp discontinuities cause additional frequency components in the result.

Windowing of data

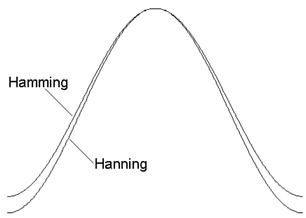


The standard solution to this problem is to taper the start and end of each data block to zero so they join smoothly. This is known as *windowing* and the mathematical function used to taper the data is the *window function*. The use of a window function causes smearing of the data, and also loss of power in the result.

You can find all sorts of windows discussed in the literature, each with its own advantages and disadvantages; windows shaped to have the smallest side-lobes spread the peak out the most. By reducing the side-lobes you decrease the certainty of where any frequency peak actually is (or the ability to separate two peaks that are close together). Spike2 implements the following windows:

No Window Use this if there is one sine wave, or if more than one, they all have similar amplitude. This has the sharpest spectral peaks, but the worst side-lobes.

Hanning This is a good, general purpose, reasonable compromise window. However, it does throw away a lot of the signal. It is sometimes called a “raised cosine” and is zero at the ends. If you are unsure about which window would be best for your application, try this one first.



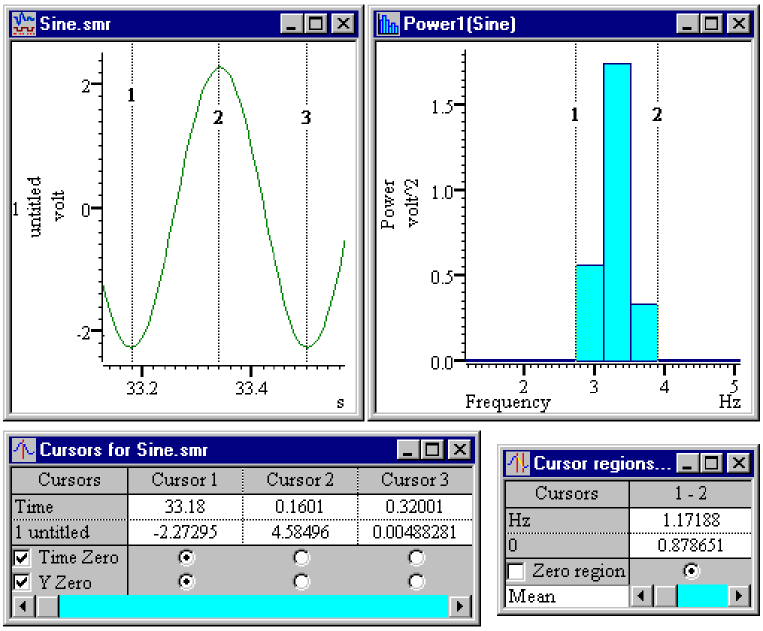
Hamming This preserves more of the original signal than a Hanning window, but at the price of unpleasant side-lobes.

Kaiser These are a family of windows calculated to have known maximum side-lobe amplitude relative to the peak. Of course, the smaller the side-lobe, the more signal is lost and the wider the peak. We provide a range of windows with side-lobes that are from 30 to 90 dB less than the peak.

Power spectrum of a sine wave

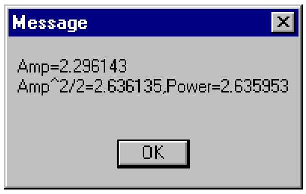
If you sample a pure sine wave of amplitude 1 Volt and take the power spectrum, you will not get all the power in a single bin. You will find data spread over three bins, and the sum of the three bins will be 0.5 $Volt^2$. The factor of 2 in the power is because we give the result as RMS (root mean square) power. This is illustrated by the example below where we have sampled a sine wave with amplitude 2.29248 Volts (peak to peak amplitude = 4.58496). We have formed the power spectrum of the signal using a 256 point transform and zoomed in around the bins where the result lies.

If the sampled data were a perfect sine wave we would predict a RMS power of 2.62773 $Volts^2$ from this waveform (2.29248^2 /2). The cursor analysis of the power shows a mean power of 0.878651, there are three bins, so the total power is 3 times this, which is 2.63595 $Volts^2$. This is about 0.3% above the predicted result for perfect data.



The predicted result is slightly low because the waveform samples used for the cursor measurements are unlikely to lie at the exact peak and trough of any particular cycle. Using the script language `Minmax()` function to find the maximum and minimum values over a wide time range gives a larger amplitude, and a much closer agreement:

You can find the Evaluate command in the Script menu if you want to try this. It gives a slightly larger value for the amplitude and now the power calculated from the amplitude and the measured power differ by 0.007%. For an explanation of the text in the Evaluate window see *The Spike2 script language* manual.

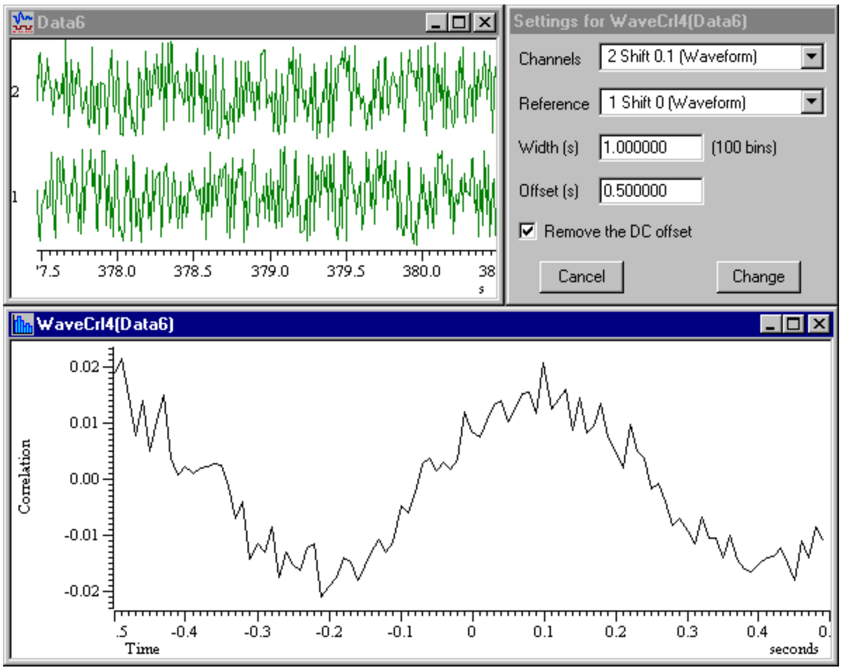


The duration of one cycle of the waveform (the time between cursor 1 and cursor 3) is approximately 0.320 seconds, a frequency of 3.125 Hz. Again, this is in agreement with the displayed power spectrum.

If you need access to the real and imaginary components of the FFT, you should consider using the `ArrFFT()` script language function.

Waveform correlation

This command creates a result view that holds the correlation between two waveform channels or the auto-correlation of a channel with itself. The correlation measures the similarity of two waveforms. The two waveforms must be sampled at the same rate. In the example below, the two waveforms hold random noise to which has been added a low amplitude 1.5 Hz sine wave. The sine wave in channel 2 is shifted forward by 0.1 seconds compared to channel 1.



The correlation is calculated by multiplying the two waveforms together, point by point, and summing the products. The sum is normalised to allow for waveform amplitudes and the number of points. This produces one result. The reference waveform moves one point to the right and the process is repeated to produce the next result. This is repeated for all the result bins. The results range between 1.0, meaning the waves are identical (except for amplitude) through 0 (un-correlated) to -1.0, meaning identical but inverted.

The bins in the output are the same width as the sampling interval of the two input waveforms. You can choose to remove the DC component from the signals before calculating the correlation. This can be important as DC offsets can dominate the result. The correlation is calculated in such a way that you can change the setting of the DC offset removal without recalculating the correlation. To do it, open the Process Settings... dialog again, click in the box and then click the Change button.

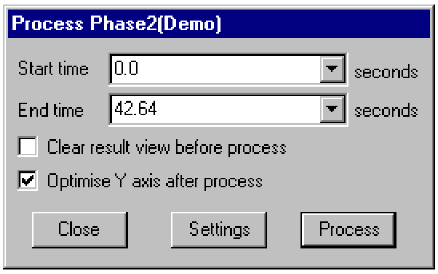
The Process dialog sets the time range of the reference waveform. The analysis is only done for regions of data in which both waveforms exist. If you calculate correlations over long sections of data, the calculations will take some considerable time! For this reason we do not recommend this as an on-line analysis (although there is nothing to stop you using it). The number of calculations (and hence the time taken) is proportional to the number of points in the result times the number of points in the reference waveform.

Process settings

This command opens the analysis setup dialog of the current result window. It is the same as the dialog that created the result window except the New button is now a Change button. The Change button accepts changed settings and clears the result.

Process

This command is available when the active window is a result view that is attached to a time view. The dialog prompts you to select a time range of the data document to process. You can choose to add the results of the analysis to data already in the result view, or you can clear the result view before analysis. You can also choose to optimise the y axis of the result view after the analysis is complete.



The simplest way to use this dialog is to type in time values for the start and end of the process region. You can also use the drop down list controls to select regions. All the items in the drop down list refer to times in the data document window. A typical list could hold the last numeric value in the window, cursor positions (for automatic cursor tracking) and the maximum time found in the data document.

Cursors and duplicated windows

If there is more than one time window associated with the data document because you have used the Window menu Duplicate command, then the list of possible positions includes the window number and the display in the dialog field becomes View(-2).Cursor(1), the -2 indicating the window to which the cursor belongs.

You can type in the cursor and maximum time specifiers directly if you wish. If you use the drop down list to select a cursor, only cursors that exist are listed.

The values in the start and end fields are evaluated when the Process button is used. The dialog window will remain on screen until removed with Cancel. This means that you can set the start time to Cursor(1) and the end time to Cursor(2) then select areas of the data document using these two cursors and click the Process button to analyse the selected area.

If you check the Clear result view before process box, the result window will be cleared of all data before the results of the processing are added.

If you check the Optimise Y axis after process box, the result window will be rescaled to best display the range of data visible in the window.

The Settings button closes the Process dialog and opens the Settings dialog for the result window.

If the type of analysis only requires a single time, not a range (for example if no trigger channel is given for a peri-stimulus time histogram), then the start time only is used. In the special case of a Phase histogram with no trigger channel, both the start and end time are used to mark the beginning and end of a single cycle.

Breaking out of Process

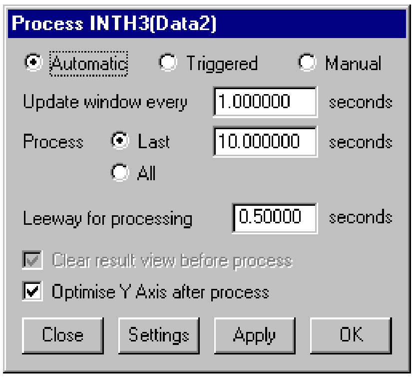
Processing operations can take quite a time, especially in large data documents. If Spike2 detects a lengthy process operation, it displays a progress dialog in which you can cancel the operation. You can also stop processing early with the `Esc` key.

Process command with a new file

This command is available when the current window is a result window from a new data file. The command activates a modified version of the process dialog. This dialog is also activated automatically when you create a result window from a new data file.

This dialog gives you control over when and how the result window is updated. You can select Automatic, Triggered or Manual updates. The dialog contents depend on the update mode. The two check boxes operate in the same way as in the Process dialog, described above. The Settings button takes you to the Process Settings dialog, Close removes the dialog, Apply and OK both apply the settings, but OK also closes the dialog.

Automatic mode



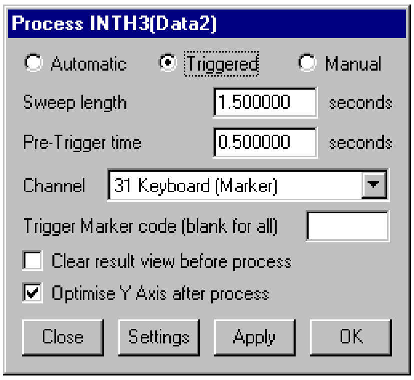
In Automatic mode, the result updates as close to a user-defined interval as possible. Set the interval to 0 for the most frequent updates.

The Clear result view before process check-box is grey as each of the two Process modes has only one useful setting. The modes are:

- **Last** Re-calculates the result for the time period set using the most recent data. Use this mode to follow changes. Clear result view is always checked.
- **All** The result of processing new data is added to the analysis. Clear result view is always unchecked.

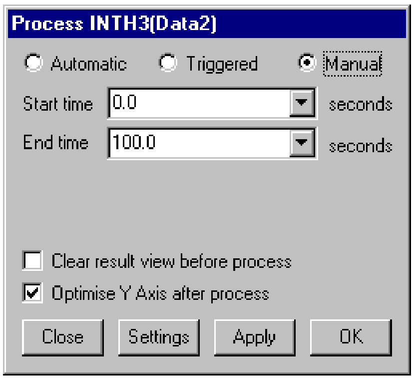
The Leeway for processing field is visible only when processing with active cursors to an XY view. It sets the time to allow after cursor 0 for other cursors and measurements. If your analysis does not generate any data points, check that this value is large enough.

Triggered mode



Triggered mode analyses data around a specific event. You specify the sweep length, the start point of the analysis region relative to the trigger point, the channel to use as a trigger point, and if this channel is a marker, the marker code to cause processing (see the *Sampling data* chapter for details of marker codes). To accept all marker codes set the Trigger Code box blank. If you do not want the results to accumulate for all analysis periods, check the Clear result view before process box in the Process dialog. Do not confuse this mode with the trigger channel in stimulus histograms and waveform averages.

Manual mode

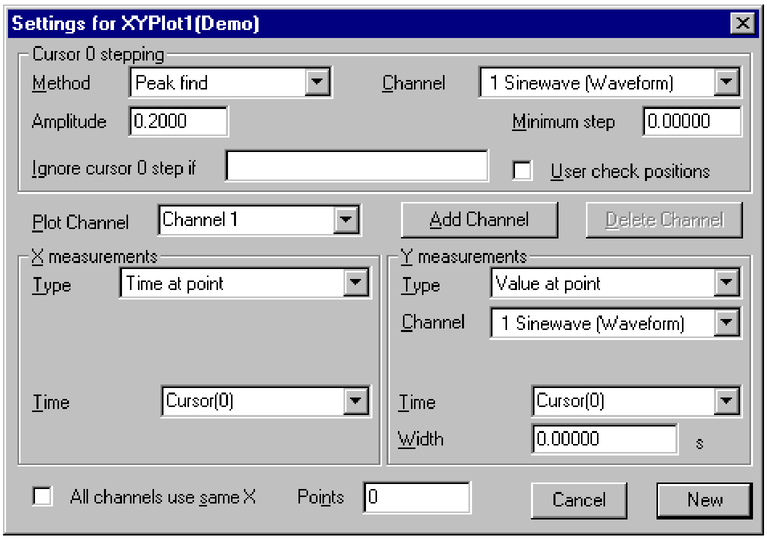


If you select Manual updates you must provide a start and end time for analysis. When you click Apply or OK, nothing will happen until the time range set for the start and end time is available in the data document.

At that point, the window contents will be calculated and the window will update. If you wish to process a different area, set a new time range and click the mouse on the Apply or OK button again.

Measurements

You can generate trend plots in an XY view both on-line in real time and off-line. Use the Measurements menu item and select XY view. This command generates a new XY window with one or more channels of data derived from the current time view.



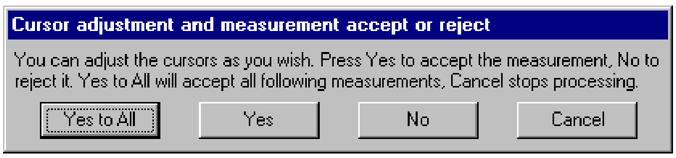
The basic idea is that cursor 0 steps through the data following a user-defined rule. This can be as simple as *move to the last cursor 0 position plus 1 second* or it can be a complex data-searching algorithm. For each cursor 0 position, we take an `x` and a `y` measurement and pass the (`x`,`y`) point to a channel in the XY view. Of course, each time cursor 0 moves, all active cursors reposition themselves. Using this mechanism, a very wide variety of measurements can be taken and added to the result.

A Process command sets the data region to search for cursor 0 step points, exactly as for the result view analysis commands. You can process data both on-line and off-line. Processing can take quite a long time if there are many cursor 0 steps to do. A progress dialog appears in a long process to give you the opportunity to stop processing early by clicking Cancel.

Cursor 0 stepping

This dialog area controls how cursor 0 moves through the data. If cursor 0 is already set to a suitable active cursor mode, this mode is copied to the dialog. You can set any active cursor mode that can iterate through data (see the *Cursor menu* chapter for details of all the active cursor modes). These are: Peak find, Trough find, Rising threshold, Falling threshold, Slope peak, Slope trough, Turning point, Data points and Expression. The other fields in this section depend on the stepping mode.

You can use the Ignore cursor 0 step field to reject the result of a cursor step operation and step again. If this field is not blank and evaluates without error, and the result is true (not zero), the current cursor step is skipped. Expressions usually involve cursor values, for example `Cursor(0)>Cursor(1)`. You will normally leave this field empty.



If you check the User check positions box, you are given the opportunity to adjust the cursor positions after each step and before each measurement. This is ignored if the data file is being sampled.

Plot Channel

The trend plot can generate up to 32 channels of data in the XY view. The dialog starts with one channel. You can set the channel title in the Plot Channel box and create additional channels with the Add Channel button. The Delete Channel button deletes the current channel; you cannot delete the last channel.

X and Y measurements These two areas have identical functionality. They generate the x and y values that are passed to the XY view for each point. The fields displayed in these areas depend on the measurement Type field. Most measurement types depend on times. For a useful result, you will set these to times relative to cursor 0, or to an active cursor that was positioned relative to cursor 0. The measurements are described in detail later in this chapter.

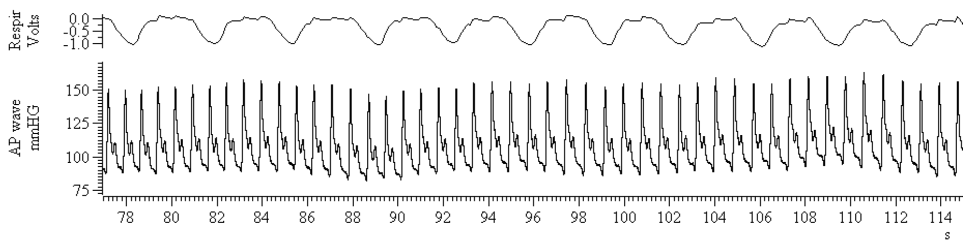
All channels use same X With multiple plot channels check this box to use the same x measurement for all channels. This is most commonly used when the x measurement is a time.

Points in plot If you set this field to a non-zero value, for example 20, the first 20 data points will appear, as you would expect. Subsequently, each added point causes the oldest point to be deleted. This is most useful on-line, for example to show pressure-volume curves or to display eye movements when you have horizontal and vertical input data.

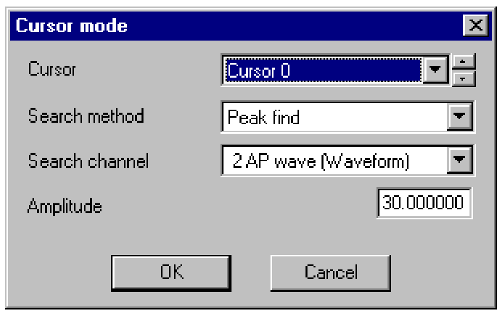
New and Cancel The New button creates the XY view and opens the Process dialog, ready to set a time range to step cursor 0 through to process the data. If you return to this dialog after creating the XY view, New becomes Change. Cancel closes the dialog.

Tabulated output You can tabulate the results in an XY view with the Edit menu Copy as Text command.

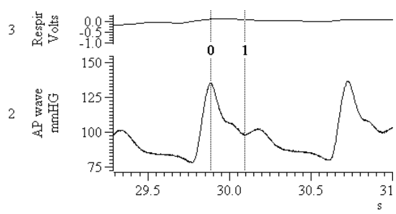
Example plot As an example of a trend plot, consider this data file containing an arterial blood pressure on channel 2 and a respiration signal on channel 3. Let us suppose that you are interested in the position of the dichotic notch (the small downward blip after the peak of the blood pressure) relative to the peak of the blood pressure.

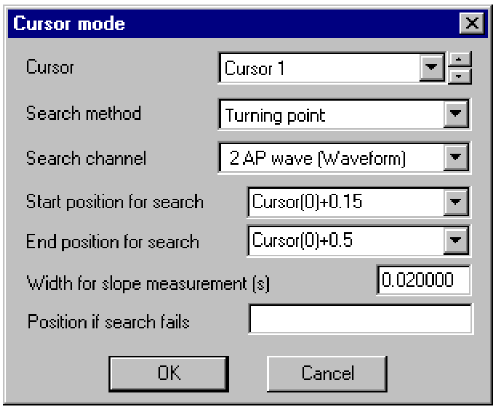


The first step is to decide how to step through the data. The obvious method is to locate the blood pressure peaks. I used the Cursor menu Active modes… command to select cursor 0, set the search method to Peak find, the search channel to 2 and Amplitude to 30. Any Amplitude from 5 to 40 would work as each cycle is at least 40 mmHg from peak to trough and the biggest wobble in between is around 5 mmHg. I checked that this was working with `Ctrl+Shift+right` and cursor 0 stepped from peak to peak.



The next step is to locate the notch. I added cursor 1 to the window with `Ctrl+1`, and then opened the cursor Active modes dialog for cursor 1. This time I set the method to Turning point, set channel 2, set the search range to start at `Cursor(0)+0.15` and end at `Cursor(0)+0.5` and set the Width for slope measurement to `0.02` seconds.

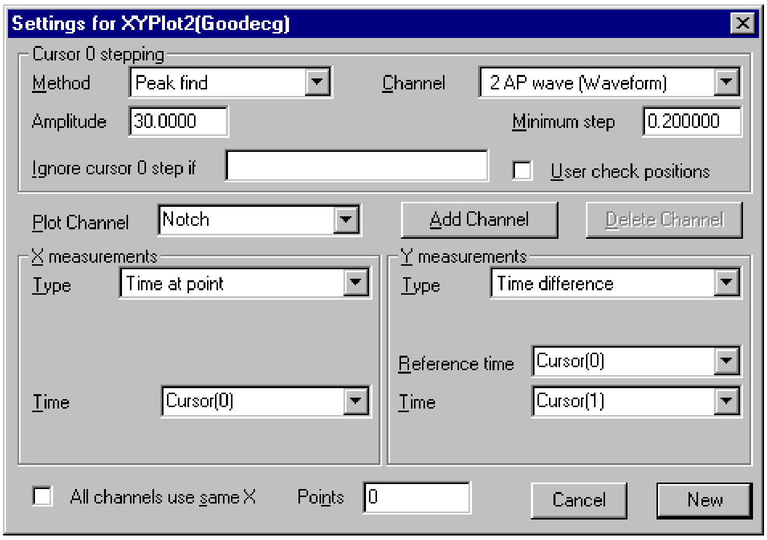




I used Turning point mode and not Trough because the amplitude of the peak after the notch may be very small. I started the search just after the peak because the peak is also a turning point (where the slope changes sign) and I wanted to exclude it. From a visual inspection of the data I could see that the notch was always more that 150 ms past the peak, so by starting the search at `Cursor(0)+0.15` I avoided the possibility of detecting the change of slope between the peak and the notch.

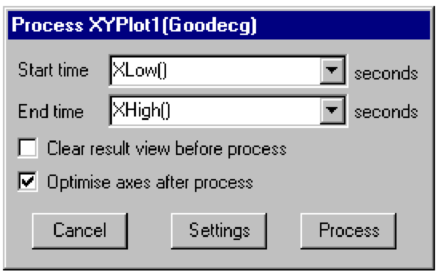
The end of the search is set so that any reasonable notch will be found. The hardest item to set is the Width for slope measurement. This needs to be big enough so that noise in the signal doesn’t cause false turning points, but small enough so we don't miss a small one. I checked it was working with `Ctrl+Shift+right` and now both cursors stepped along the data. Position if search fails is left blank so that if the turning point cannot be found, no measurement is taken.

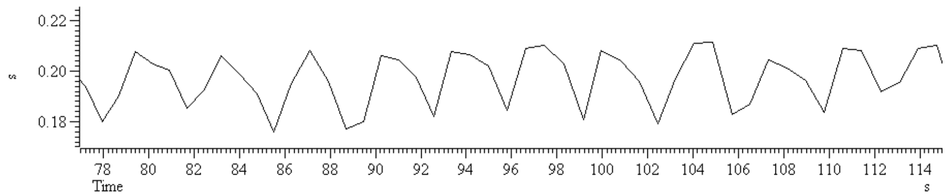
The final step is to generate the graph. I used the Analysis menu Measurements command and selected Trend plot. When the dialog opens it automatically picks up any active cursor setting from cursor 0.



I wanted to plot the notch position relative to the peak against the position of the peak. To do this I set the X measurement to be Time at point and selected Cursor(0) to be the point. I set the Y measurement to Time difference and set the Reference time to Cursor(0) and Time to Cursor(1) to form Cursor(1)-Cursor(0) as my measurement.

All that remained was to click New to create the XY window and open a Process dialog. I adjusted the time window so that it displayed the data I wanted to process, then set the start and end times to `XLow()` and `XHigh()` and clicked the Process button. The picture below shows the result. For each heart beat in the time range, we have a plot of the time delay from the peak to the dichotic notch.





Of course, we could have made a wide variety of measurements. The x axis need not be time, it could be a value derived from a different channel. For example, looking at the results, we can see that there seems to be a link between the notch position and the respiration signal, so we might have set the x axis to the value of the respiration channel at the cursor 0 time.

Measurement fields The X measurements and Y measurements areas determine the x and y values passed as a pair into each XY channel. The Type field sets the measurement method. The contents of the measurement area depend on the measurement type. The following fields are used:

Channel The data channel used for a measurement. Select the data channel from the list.

Time This is a time for use either as a value, or the time at which a measurement of the data channel is to be made. This field will usually be set to an expression that contains a cursor position, for example `Cursor(0)-1.3` or `(Cursor(0)+Cursor(1)/2)`. You can select expressions from the drop down list.

Expression This is used for Expression measurement mode only and is the expression to be used for a measurement. This will usually be a view expression that evaluates to a time.

Prompt This is used in User entered value measurement mode only, and is the prompt to use to request data values from the user.

Reference time This field is used in the same way as the Time field. It specifies a time to subtract from the time in the Time field for the Time difference measurement, or the time at which to measure the data value to subtract for the Value difference measurement.

Width This field is used to set the width of a point measurement around a time. If you set this to 0, the nearest data point is used, otherwise a mean value over the time range from Time-Width/2 to Time+Width/2 is used.

Start time, End time These two fields always occur as a pair and identify a time range in which to take a measurement. Both fields will accept an expression for a time and you can select likely expressions from the drop down lists.

Minimum step Cursor 0 does not use the Start time and End time fields. Instead, you set the minimum step from the last cursor 0 position. The search range for cursor 0 ends at the end of the file (or the start if you are searching backwards) and starts at the minimum step away from the last cursor 0 position.

Measurement types In the description of each measurement type we refer to waveforms and events. By waveform, we mean either a waveform channel, or a WaveMark channel drawn as a waveform. The measurements are:

Value at point This is the value of the nominated Channel at the specified Time. If Width is non-zero, the measurement is the mean value over the time range from Time-Width/2 to Time+Width/2. For a waveform or a channel drawn as a rate, mean frequency or instantaneous frequency, the result is in y axis units. For all other event drawing modes, the result is the time of the first event at or after Time.

Value difference This is the value of the nominated Channel at Time minus the value at Reference time. If Width is non-zero, the measurement is the mean value from -Width/2 to Width/2 around each time point. For a waveform or a channel drawn as a rate, mean frequency or instantaneous frequency, the result is in y axis units. For all other event drawing modes, the result is the time difference between the first event at or after Time and the Reference time.

Time at point This is the value of the expression in the Time field. In most cases, this will be a cursor position, for example: `Cursor(0)`.

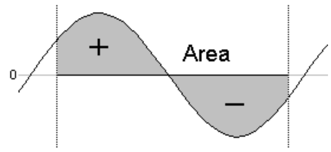
Time difference This is the value of the expression in the Time field minus the value of the expression in the Reference time field. In most cases, both these values will be cursor positions.

Fit coefficients This is reserved for a future release of Spike2. It will give you access to the result of curve fitting to the data.

Expression Type in an expression (usually one that involves cursor positions) and this is evaluated at each step to produce the value.

Use entered value You will be prompted to enter a value at each step of the cursor 0 position. If you enter a non-blank prompt, this is used when you are asked for a data value.

Area If the Channel is a waveform, the result of this measurement is the area between the waveform channel and the y axis zero over the time range set by Start time and End time. The area is positive for curve sections above zero and negative for sections below zero. For all other channel types this is the count of events in the time range.

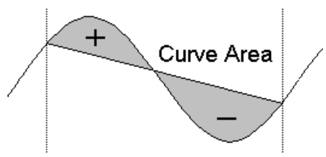


Mean For a waveform, the result is the mean value of the data points between Start time and End time. For all other channel types the value is the number of events in the time range divided by the time range. This could be thought of as the mean event rate in the time range.

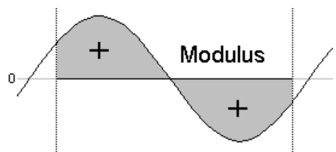
Slope This has meaning only for a waveform. The result is the slope of the best-fit straight line by the least squares method to the data between Start time and End time.

Sum For a waveform channel or a WaveMark channel drawn as a waveform this is the sum of the values of the data points. For all other channels it is the same as the Area measurement.

Curve area This measurement only has meaning for a waveform. It is the area between the straight line joining the ends of the waveform data in the time range and the data points. Sections of the curve above the straight line count as positive area, areas below count as negative area.

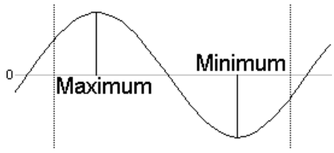


Modulus This has meaning only for a waveform. It is the same as the Area measurement except that all areas (above and below the zero line) count as positive. If you use this on an event channel no measurement is taken.



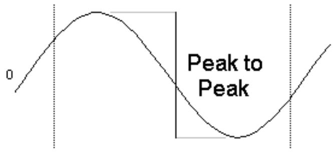
Maximum, Minimum, Extreme

These have meaning only for a waveform. They measure the maximum and minimum values in the time range and Extreme measures the larger of the maximum and minimum value ignoring the sign. For example if the maximum value was 6 and the minimum value was –7, the extreme value would be 7.



Peak to Peak

This measurement has a value for waveforms only. It is the difference between the maximum and minimum value of the channel in the time range Start time to End time.



RMS amp

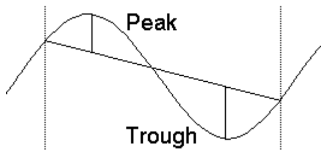
This measurement has a value for a waveform only. It is calculated by summing the square of each data point, dividing the sum by the number of data points and then taking the square root of the result.

SD

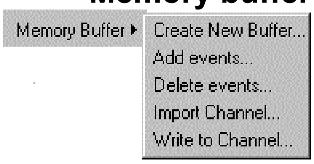
The standard deviation has a value only for a waveform. It is calculated by finding the mean of the data, then summing the squares of the differences between each data point and the mean, dividing the sum by the number of data points minus 1, and finally taking the square root of the result.

Peak and Trough

These measurements only have meaning for waveforms. These values are the maximum positive and negative distances between the waveform and a straight line joining the end points of the waveform in the time range Start time to End time. The Peak value is always greater than or equal to 0. The Trough value is always less than or equal to zero.



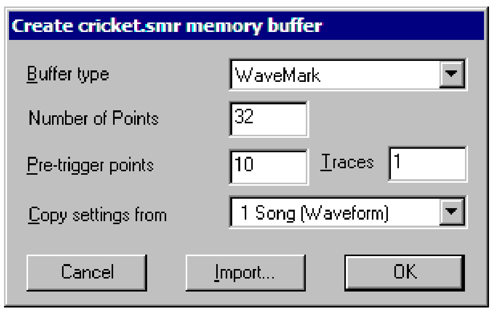
Memory buffer



Each data file may have up to 100 memory channels holding data copied from existing channels, derived from waveform channels or entered by hand. You can display the memory buffers and use them like any other channels. The memory buffers can also be written to the data document. If you do not write them, the memory channels are lost when you close the file. Each memory buffer expands as events are added. The size is limited by available memory.

Create New Buffer

This command creates a new memory buffer channel of any channel type. You can create up to 100 such channels, numbered 101 to 200. A new channel is given the lowest available number. Use the Analysis menu Delete Channel command to remove memory buffers. OK creates the channel, Import creates it and opens the Import channel dialog.



Channels created with this command are not permanent. The data is kept in memory and is lost when the file is closed. If you need to make the data permanent, you must write it to a permanent channel with the Write to Channel option.

The fields in the dialog change depending on the type of buffer you choose to create. The example shown is for a WaveMark channel. The fields are:

Buffer type

You can create a channel of any type: Event-, Event+, Level, Waveform, RealWave, Marker, TextMark, RealMark or WaveMark.

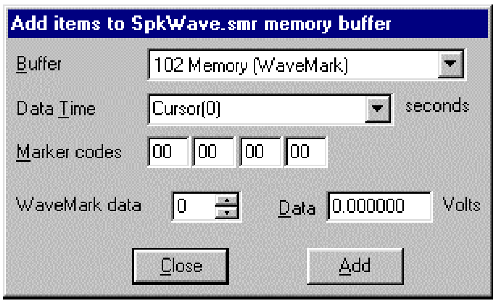
Number of … This field is present for TextMark data where it sets the maximum number of characters to store with each mark, RealMark data where its sets the number of real values to store with each mark and for WaveMark data where it sets the number of waveform points to store with each mark.

Pre-trigger points and traces These fields are present for WaveMark data only and set the number of waveform points before the trigger and the number of traces in each WaveMark item in the range 1 to 4.

Copy settings from When you create a waveform, RealWave or WaveMark buffer, Spike2 needs to know the spacing of the waveform points and the calibration settings. You use this field to indicate a channel that holds waveform, RealWave or WaveMark data and the buffer is given the same sampling rate and calibration. If you create a buffer from the script language you can choose a sample rate and calibration without reference to another channel.

Add items to memory buffer

This command adds a data item to a memory buffer. When you Add an event, a new event is placed in the buffer at the specified time. If an event already exists at exactly that time, it is replaced. Cancel closes the dialog.



You can add data for channels of any type except waveforms. The fields vary depending on the channel type:

Buffer The memory buffer channel to add events to.

Event time The time at which to add data. If you have cursors enabled in the time view you can add data at the time of the cursor (as in the example). This field is present for all channels.

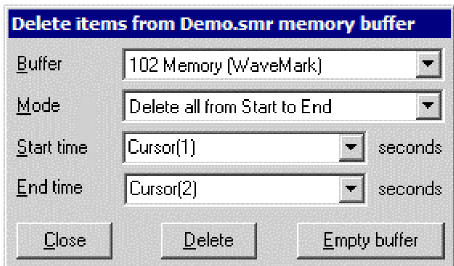
Marker These four fields are present for all Marker derived data types. You can set the marker codes appropriate for your data here.

WaveMark/RealMark data This field is present for WaveMark and RealMark channels. Enter the index of the data point that you wish to set in the Data field. There is also a Trace field for WaveMark buffers with multiple traces.

TextMark string This field is present for TextMark data only. Enter the text to appear in the new data item.

Delete items from memory buffer

This command opens a dialog in which you can delete one or more events or markers or a time range from a memory buffer. The Delete button removes one or more events as set in the dialog. The Empty buffer button deletes all data for this channel (it does not delete the channel). Close removes the dialog. The `MemDeleteTime()` script command has the same functions.



Buffer The memory buffer channel to use.

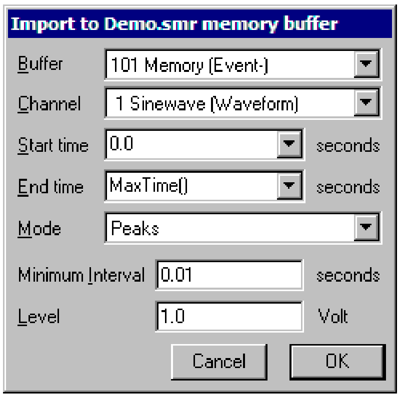
Mode There are four modes: Delete nearest to Start within Range, Delete all round Start within range, Delete first in range Start to End, Delete all from Start to End. The first two modes delete one or more data items around a time, the other two modes delete the first or all data items in a time range.

Start time The time range is Start time – Time range to Start time + Time range in the first two modes, and Start time to End time for the last two modes.

End time The end of the time range. This field appears in the second two modes.

Time range The time range around the Start time to search for data. This field appears in the first two modes. This field is usually set to a small value so that you can delete events close to the position of a cursor.

Import channel You can import data into a memory buffer selected by the Buffer field from a channel set by the Channel field. The Start time and End time fields set the time region to import data from. The Mode field is present if the source is a waveform channel and the destination is not, and sets the method to extract event times from the waveform.

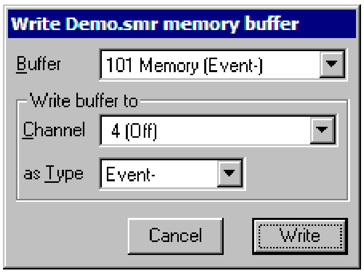


To import a waveform channel to a waveform or WaveMark channel the ADC sampling rates must match. Apart from this restriction, you can move data from any channel to any buffer. During data import, values that cannot be extracted from the source are set to 0. For example, when importing an event channel into a WaveMark channel, there are no marker codes or waveforms in the source, so these are set to 0.

Waveform channels There are four modes to extract events from waveforms: Peaks, Troughs, Data rising through level and Data falling through level. Detected events are added to the memory buffer as markers with codes 2 (Peaks), 3 (Troughs), 4 (rising data) or 5 (falling data).

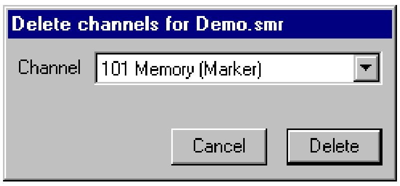
These modes use the Minimum interval field as the minimum separation of detected events, to filter out events caused by noise in the input waveform. Set this to 0.0 if you do not want to set a minimum period between detected events. The peak search mode looks for a peak in the signal followed by the signal falling by at least the contents of the Level field. The trough search mode looks for a minimum in the signal, followed by a rise of at least the contents of the Level field. The rising and falling level modes detect events when the signal crosses the level in the set direction.

Write to channel You can write a memory buffer to the data document. The Type field sets the format for the saved data. You can also Append the data to an existing channel. In Append mode, the data in the memory buffer must all occur later than data in the target channel. For modes other than Append, if you select a channel number that is already in use, you will be warned.



Delete channel

This removes a channel from a Spike2 data file permanently or deletes a channel created with New Buffer from the Memory Buffer command. If you delete a channel with duplicates, the duplicates are also deleted. Deleting a duplicated channel removes the duplicate only. You can also delete channels (except the last one) from XY views and duplicated result view channels.

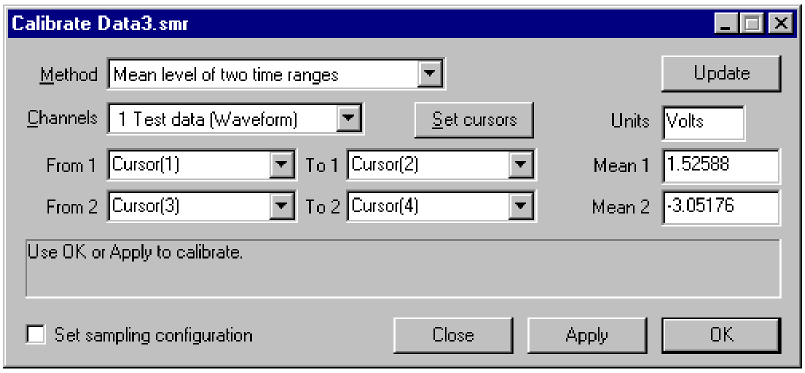


Duplicate Channels

This duplicates selected channels in the current time or result view. Duplicate channels share data, title and channel comment with the parent and inherit the channel settings. Once you have duplicated a channel you can change display mode and y axis range independently of the original. With time view marker data, you can change the marker filter. The channel number of a duplicated channel is displayed as the original channel number plus a letter. The first duplicate of a channel gets the letter a, the second b and so on up to z. Duplicate channels are deleted with the Analysis menu Delete Channel command. New duplicates of a channel get the lowest free letter available.

Calibrate

If a waveform, WaveMark or RealWave channel has sections with known amplitudes, offsets, slopes or areas, you can calibrate it. Spike2 supports a wide variety of calibration methods for these channels. You can calibrate a single channel, or all selected channels (as long as they have the same calibration values). If you make a mistake, you can Undo the calibration changes. You can also calibrate from a script.



To calibrate your data, click on a time window and open the calibrate dialog. If there are any suitable channels selected, the channel list in the dialog shows the selected channels, otherwise you must choose a channel for the drop down list. Once the dialog is open you can select channels in the time window; these are added to the selected list in the dialog.

Now choose a calibration method from the list. The dialog contents change depending on the selected method. All methods require you to select data areas with known values; this is most easily done with cursors. Click Set cursors to position the appropriate cursors in the time window and dialog. You can also type in the times and use expressions like (Cursor(1)+Cursor(2))/2. If the method requires two time ranges they must not overlap.

When you change the method or the channel or click the Update button, Spike2 collects the current calibration values from the (first) channel in the Channels field.

Once you have selected your data you must type in the calibration values, and also set the units of the data. A text box displays Use OK or Apply to calibrate or an explanation of any problem that prevents calibration. Click Apply to calibrate and leave the dialog open or OK to calibrate and close the dialog.

If you check the Set sampling configuration box, any calibration changes are passed through to the same channel number in the sampling configuration. If you calibrate with a data file that you are sampling, or that you have just sampled and not yet saved, this box is checked automatically when the dialog opens.

What calibration does

Waveforms stored on disk as 16-bit integers (range –32768 to 32767) are scaled into user units by a `scale` and `offset`: User units = 16-bit integer * `scale` / 6553.6 + `offset`

The factor 6553.6 is present so that if the input data range spans –5 to +5 Volts (as is usually the case with a CED 1401 interface), the relationship is:

User units = input in Volts * `scale` + `offset`

The `scale` and `offset` are the same values as you set in the sampling configuration dialog or in the channel information dialog in a time window. Calibration changes the `scale` and `offset` so that the displayed data matches your user units.

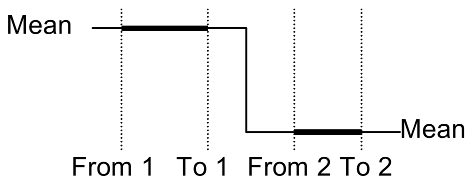
RealWave data is stored as floating point numbers on disk. Calibration rewrites this data and so may take noticeable time when working on very large data files. You cannot calibrate a RealWave data if any channel process is attached. You cannot calibrate any channel with an attached process that has changed the channel scale or offset value.

Calibration methods

You can select one of the following calibration methods using the Method field of the calibration dialog:

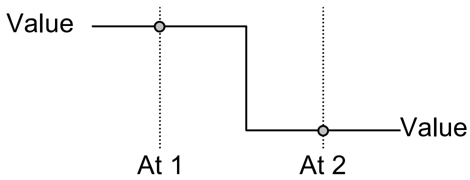
Mean level of two time ranges

You define two time ranges in your data (From 1 to To 1 and From 2 to To 2) and supply the mean levels of the two ranges (Mean 1 and Mean 2). The data in the time ranges must have significantly different levels and the two mean values you supply must not be the same.



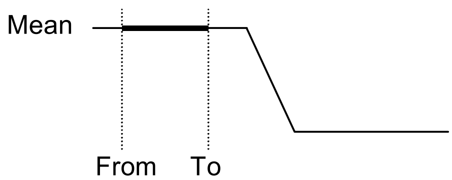
Values at two times

You define two times (At 1 and At 2) and the two values (Value 1 and Value 2) that correspond to the two data values. The two values must be different and the data at the two times must also be different.



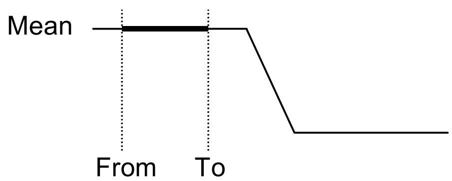
Set offset from mean of time range

If your data is calibrated, but suffers from baseline drift, you can use this method to redefine your base line. Set a time range (From and To) and the mean value of the data in the range (Mean). The data scaling is not changed.



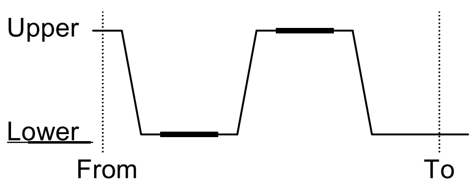
Set scale from mean of time range

This method is for data with a fixed offset (usually 0) and variable gain. Set a time range (From and To) and the mean value of the data in the range (Mean). The data offset is not changed. The mean value you set cannot be 0 and the mean level of the data before calibration cannot be 0.



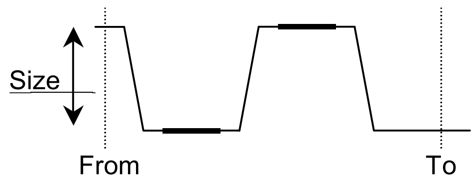
Square wave, upper and lower level

This method will calibrate a square wave with known Upper and Lower levels. The time range (From and To) must contain at least three transitions between levels. In this case Upper means the larger value before calibration. To calculate the mean value of these levels, Spike2 detects the transition points between high and low values. For each high or low section, the first and last 25% of the data is ignored. The upper and lower levels must not be the same.



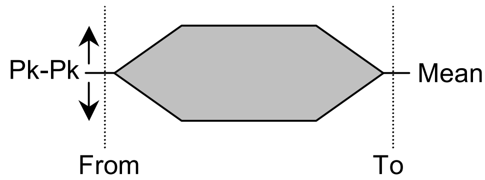
Square wave, amplitude (Size) only

This method detects a square wave in the same way as the *Square wave upper and lower level* method. In this case, you supply the Size (difference between the upper and lower levels) of the waveform. The offset of the data is not changed.



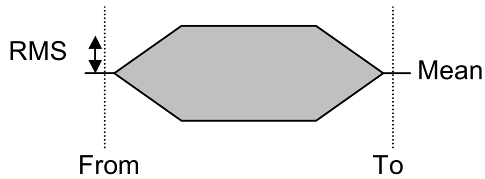
Peak to peak amplitude and mean

This method calibrates based on the peak to peak amplitude (Pk-Pk) and Mean value of the data in the time range (From and To). This could be used for a sinusoidal waveform of known amplitude. The Peak to Peak value must be non-zero.



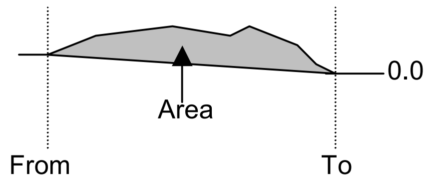
RMS amplitude about mean

This method calibrates based on the RMS (Root Mean Square) amplitude of the data in the time range (From and To) and the Mean level. The RMS value must be non-zero.



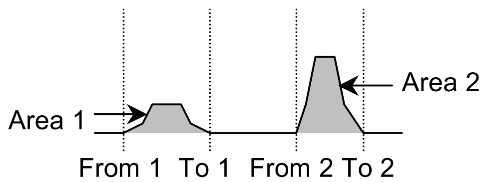
Area under curve, assume zero at end

You could use this method to calibrate a rate based on a known area (for example flow rate based on volume). The rate is assumed to be zero at each end of the time range (From and To) to allow for a drifting base-line. The area you set is in the units of the rate units times seconds. The rate value at the To time is set to zero (if the From value is different the base line is assumed to run linearly from From to To).



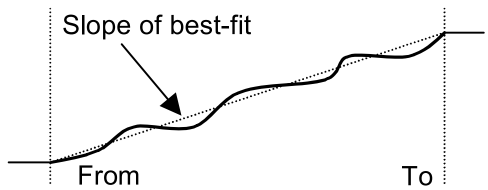
Areas under curve, two time ranges

Use this method when you know the area under the data for two sections (From 1 to To 1 and From 2 to To 2) of your data. The mean level of the two sections must be different. The area you set is in the units of the channel units times seconds.



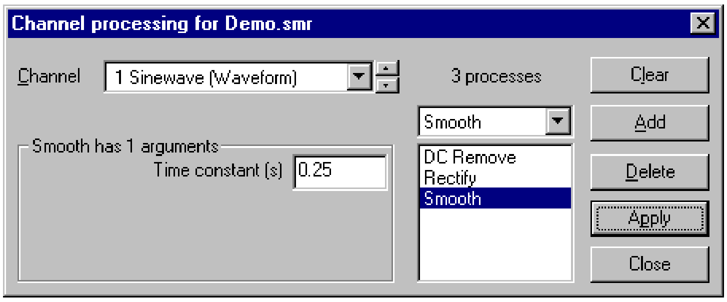
Set scale from slope (no offset change)

Use this method when you know the slope of a section of the data (for example calibrate position based on a known velocity, or velocity based on a known acceleration). The calibration is set so that the slope of the best-fit line to the data in the range matches the slope value you set. This method does not change the offset, so you may need to combine this method with the *Set offset from mean of time range* method for a full calibration. The units of the slope you set are the channel units per second.



Channel process

A channel process is an operation, for example rectification, applied dynamically to waveform or RealWave data channels. The original data is not changed in any way but all users of the channel see the channel data modified by the process. More than one process can be applied, for example DC removal could be followed by rectification followed by smoothing. Every time you refer to data in the channel, the processing is applied, so working with a processed channel is slower than working with raw data.



To add a process to a channel, select the Channel Process command in the analysis menu and select the Channel. The list to the left of the Apply button shows processes to apply in order. Any values that can be modified for the selected process are displayed in the box on the left of the dialog. The Clear button removes all processes from the channel. Add appends a new process set by the drop down list to the left of the Add button to the end of the list. Delete removes the selected process from the list and Apply copies the current values of the arguments to the process.

Some processes change the channel scale and offset (these are the values that translate between a 16-bit integer representation of a waveform and user units). Such changes do not affect the data on disk and are removed when the process is removed. The Calibrate dialog and the Channel Information dialog will not allow you to change the channel calibration if an attached process has changed the channel scale or offset. You are not allowed to calibrate a RealWave channel that has any attached channel process. You can add processes of the following types:

Rectify This replaces all negative input values with positive values of the same magnitude. The result of this operation may exceed the 16-bit range of a waveform channel if the channel offset is negative, in which case the output will be limited to the available 16-bit range. There are no additional arguments required to define this process.

Smooth This process has one argument, a time period in seconds, *p*. The output at time *t* is the average value of the input data points from time *t-p* to *t+p* seconds.

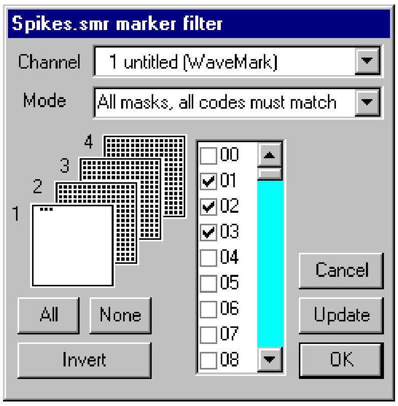
DC Remove This process has one argument, a time period in seconds, *p*. The output at time *t* is the input value at time *t* minus the average value of the input data points from time *t-p* to *t+p*. This process does not affect the channel scale, but the channel offset is set to zero.

Slope This process has one argument, a time period in seconds, *p*. The slope at time *t* is calculated using an equal weighting of the points from time *t-p* to *t+p*. If you apply this process to a channel, the channel scale, offset and units change. If the current channels units are no more than 3 characters long, `"/s"` is added to them, so units of `"V"` become units of `"V/s"`. If there is not sufficient space, the final character of the units becomes `"!"` to indicate that the units are no longer correct. The offset becomes 0, and the scale changes to generate the correct units.

Time shift This process has one argument, the time to shift the wave. A positive time shifts the wave into the future (to the right), a negative time shifts the wave into the past.

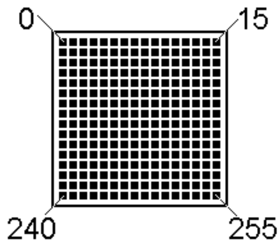
Down sample This process changes the sample rate of the wave by taking one point in `n`. There is one argument, prompted by Use one point in, which is the down sample ratio. You might want to use this command after filtering or smoothing a waveform. If the wave is already part of an analysis operation, such as a waveform average, a power spectrum or a waveform cross-correlation, adding this operation and continuing with the analysis will make nonsense of the results.

Marker Filter



You can filter any channel that holds marker, WaveMark, TextMark or RealMark data. You select the channel with the Channel field (which is also controlled by the Edit WaveMark and new WaveMark dialog channel field). Each marker channel has a *marker filter* that selects the data items to display and use in calculations. Each data item has four marker codes that are matched against the marker filter.

In the dialog, the marker filter is drawn as four masks numbered 1 to 4. Each mask has 256 elements in a 16x16 grid, one for each possible code value. The front-most mask is edited by the dialog; click on a mask to bring it to the front. The contents of the front mask are displayed in the scrolling list in the centre of the dialog.



The top row of each mask represents code values 0 to 15 (hexadecimal codes 00 to 0F), the second row is values 16 to 31 (10 to 1F) and so on down to the bottom row, which represents values 240 to 255 (F0 to FF). If a code value is included in the filter, the corresponding mask square is black. When values are excluded, the square is white. The mask gives a quick indication of the state of the filter. There are three buttons that act on the entire mask:

All This includes all code values in the filter
None This excludes all code values from the filter
Invert This excludes included values and includes excluded values

The list shows each the code value as either two hexadecimal digits, or the single character equivalent. Characters are appropriate for keyboard markers, hexadecimal is used for the non-printing codes. If you type a character, the list scrolls to the entry that starts with the typed character. To include marker codes, check the boxes. There are two modes in which to use the marker filter:

Mode 0: All masks, all codes must match

For a data item with marker codes *a*, *b*, *c* and *d* to be included, mask 1 must have code *a* checked, mask 2 must have code *b* checked, mask 3 must have code *c* checked and mask 4 must have code *d* checked. Most users of this mode set mask layers 2, 3 and 4 to All and use the first layer to select data values. You can think of this as the *and* mode; to accept data marker code 1 must be in the layer 1 *and* marker code 2 must be in the layer 2 *and* marker code 3 must be in the layer 3 *and* marker code 4 must be in layer 4.

Mode 1: One mask, any code can match

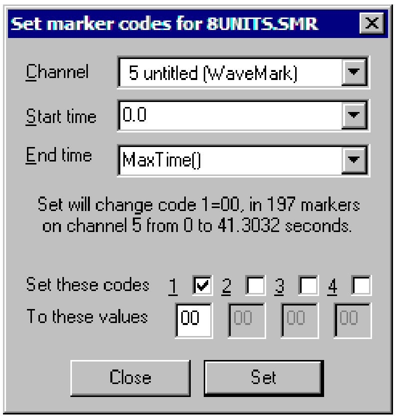
Only mask 1 is used, the rest are greyed out. For a data item with marker codes *a*, *b*, *c* and *d* to be included, mask 1 must have one or more of the codes *a*, *b*, *c* and *d* checked. You can think of this as the *or* mode; to accept data marker code 1 *or* marker code 2 *or* marker code 3 *or* marker code 4 must be in the layer. There is once exception; for marker codes 2 to 4, the code 00 is ignored. If you want to accept code 00 it must be the first marker code.

Mode 1 is often used when sorting spike shapes (WaveMark data) and you discover a WaveMark that is the result of a collision between two spikes. You can set the first marker code to the code for the first spike and the second to the code for the second (leaving the third and forth codes as 00), then the spike will appear on screen and in analyses when either of the codes are included in the mask.

The buttons on the right are:

Cancel Close dialog, cancelling changes since last update
Update Update the channel display to correspond with the new filter
OK Close dialog, accept new filter

Set Marker Codes



You can open this dialog from the Analysis menu, by right clicking on a marker channel and from the Edit WaveMark dialog. From this dialog you can set selected marker codes for marker, WaveMark, TextMark and RealMark channels.

Each data item has four marker codes that can be used to filter the data in the channel. Usually only the first marker code is used, see the Marker Filter command for more information.

The Channel field sets the channel to process and the Start time and End time fields identify the markers to be set. If a marker filter is set for the channel, only data items that are in the filter are changed by this command. The four check boxes 1 to 4 select the codes to change. You set the value for each code as either two hexadecimal digits or one printing character.

Example

To change all the marker items in a channel with a first marker code of `02` to have a first marker code of `1A` do the following:

1. Open the marker filter dialog for the channel and set the filter to display only marker code 02.
2. Open this dialog, select the channel and set the time range as 0.0 to Maxtime().
3. Check the `1` box to set the first marker code and edit the text under the check box to 1A. Make sure the other check boxes are clear.
4. Read the text in the centre of the dialog as a check that this is the action you wish to take. This action cannot be undone.
5. Click the Set button to change the marker codes.

All the code `02` items will disappear as they have been coded as `1A`, so you will need to change the marker filter settings if you want to see the result.

New WaveMark

This creates WaveMark data channels from waveform data or existing WaveMark data (see the *Spike shapes* chapter for details).

Edit WaveMark

This option reclassifies WaveMark data both manually and automatically (see the *Spike shapes* chapter for details).

Digital filters

This opens the Digital filtering dialog, which can create FIR filters and apply them to waveform channels (see the *Digital filtering* chapter for details).

Window menu

The Window menu controls the data and text windows that belong to the Spike2 application. It has commands to duplicate a time window, commands to hide and show windows, a command to close all windows and commands to arrange the windows within the application window. The remaining space at the bottom of the menu holds a list of all the windows that belong to the Spike2 application. If you select one of the windows in the list, the window is brought to the front and made the current window.

Duplicate window

This command is only available within a time window and creates a duplicate window with all the attributes (list of displayed channels, event display modes, colours, cursors and size) of the original window. Once you have created the new window, it is independent of the original. However, the data channels within it are the same data channels as in the original window, so any changes made to the data in one window will cause all duplicated windows to update. Duplicating a window allows you to have different views of the same data file with different scales and drawing modes and different sets of channels.

You can close all windows associated with a data document using the control key plus close (see the Close command in the *File menu* chapter). This will remember the position and state of all windows associated with the document.

Hide

This command makes a window invisible. This is often used with script windows and sometimes is used to hide data windows during sampling when only the result views are required. Closing the Log window is equivalent to hiding it as the Log window always exists.

Show

This option lists all hidden windows in a pop-up menu. Select a window from the list to make it visible.

Tile Horizontally

This command arranges all the visible screen windows so that no window is overlapped by any other. Iconised windows are arranged along the bottom edge of the window. The command attempts to arrange window so that they are wider than they are high. Tiling takes into account the space used for title bars of iconised windows, so to use the full application window area you should hide any iconised windows first. This command may have the same result as Tile Vertically, depending on the number of windows.

Tile Vertically

This command arranges all the visible screen windows so that no window is overlapped by any other. Iconised windows are arranged along the bottom edge of the window. The command attempts to arrange windows so that they are taller than they are wide. Tiling takes into account the space used for title bars of iconised windows, so to use the full application window area you should hide any iconised windows first.

Cascade

All windows are set to a standard size and are overlaid with their title bars visible. Any iconised windows are left in the iconised state, and they are arranged along the bottom edge of the window, as for the Arrange Icons option.

Arrange Icons

You can use this command to tidy up the windows that you have iconised in Spike2. The icons are lined up along the bottom edge of the application window.

Close All

This command closes all windows in the Spike2 application. You are asked if you want to save the contents of any text windows that have changed or any newly sampled data window. You can avoid being asked if you want to save modified result and XY windows with an option in the Edit menu Preferences. The positions of data windows and attached result view windows are saved.

Cursor menu

A cursor is a vertical line or horizontal drawn in a time or result view, to mark or obtain a position. The Cursor menu creates and destroys cursors, changes the display to make them visible, changes their labelling mode and obtains the values of channels where they cross the cursors and between the cursors. Up to 10 vertical and 4 horizontal cursors can be active in each time or result window. Cursors can be dragged over and past each other. Cursors in separate windows are independent of each other. When a time window is duplicated, the cursors are also duplicated.

Active cursors

In a time view, vertical cursors can be *active* or *static*. An active cursor can automatically seek to a position based on the position of other cursors and data in a channel. Active cursors can automate data analysis, leading to XY trend plots or tabulated output.

Cursor 0

Vertical cursor 0 is special. It always exists and cannot be deleted, but it can be hidden. It is used as the iterator for XY trend plots and a movement of cursor 0 causes all active cursors except cursor 0 to recalculate their positions. The calculation of active cursor positions is done in order of rising cursor number. To hide cursor 0, right click on it and select Hide cursor 0 in the context menu.

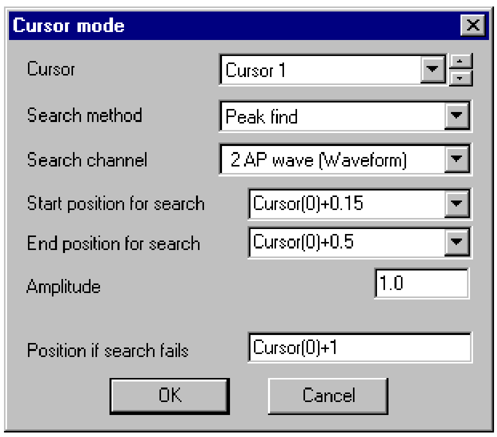
Valid and invalid cursors

Active cursor positions are either *valid* or *invalid*; invalid cursors have an exclamation mark at the start of the label. The cursor position is invalid if the search method 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 rejects 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 the search fails, and the Position if search fails expression is blank or there is an error evaluating it, or it uses an invalid cursor position, the cursor is not moved and the cursor position becomes invalid.

New Cursor

This command duplicates the cursor button at the bottom left of time and result windows. It is available in time, result or XY windows when not all cursors 1 to 9 are in use. It adds a vertical cursor with the current label style and lowest available cursor number (1 to 9) at the centre of the window. The keyboard short cut `Ctrl+n` where `n` is a cursor number (including 0 in a time window) creates cursor `n` if it doesn't exist and moves it to the centre of the window. New cursors are created in Static mode.

Active mode



This command opens the Cursor mode dialog. The contents of the dialog depend on the Search method field and the selected cursor. An active cursor has an associated Search channel and start and end positions that define the data to search to locate the new cursor position.

Cursor 0 does not have start and end positions. However, it does have a Minimum step; searches start at this distance from the cursor 0 position and continue to the end of the file for a forward search, or to the start of the file for a backward search. Cursor 0 has a restricted range of search methods: Peak find, Trough find, Rising threshold, Falling threshold, Peak slope, Trough slope, Inflection, Data points and Expression.

The search positions can be a fixed time, but more usually they will be expressions that involve the positions of other active cursors. Active cursor positions are evaluated in sequence from cursor 0 to cursor 9. For cursor `n`, an expression that refers to an active cursor less than `n` refers to the new position. An expression that refers to a cursor greater than or equal to `n` refers to the old position.

If the End position for search is less than the Start position for search, searches are backwards through the data. If a search is backwards, read *previous* for *next* and *last* for *first* in the descriptions of the cursor modes. Where search modes depend on data values, for example maximum and minimum, and the data channel does not have a y axis, then the value is taken as the interval between data items.

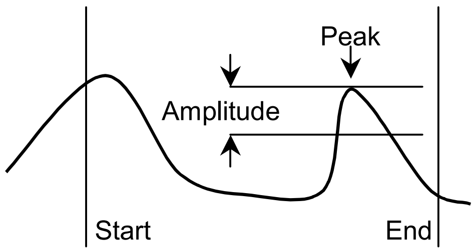
Several of these modes use the slope of a waveform. All modes with a slope have the value Width for slope measurement, which is in seconds. The data points from Width/2 before the current position to Width/2 after are used to calculate the slope unless there are more than 200 points, in which case 100 points before and 100 points after are used. The contribution of each point to the slope is proportional to the distance of the point from the current position. Because the slope at any point requires data around it, the slope within Width/2 of the ends of the data and any gaps is not to be relied on as the missing data is assumed to be zero.

Static

When you add a new cursor, it starts out with a mode of Static. In this state, the cursor stays where you put it; it is not changed by a change in the position of a lower numbered cursor. The cursor position is always valid.

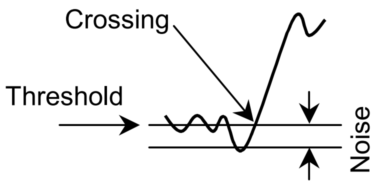
Peak find, Trough find

For this mode there is an extra field, Amplitude, which defines how much the data must rise before a peak and fall after it (or fall before a trough and rise after it), for it to be accepted as a peak. For waveform channels, the peak position is located by fitting a parabola through the highest point and the points on either side. In the diagram, which shows a peak search, the first peak is not detected because the data did not rise by Amplitude within the time range. The cursor position is valid if a peak is detected, invalid if not.



Rising threshold, Falling threshold,Threshold

In these modes there are two new fields, Threshold and Noise rejection (hysteresis). The data must cross Threshold from a level that is more than Noise rejection away from it. For the Rising threshold mode, the data must increase through the threshold, for Falling threshold mode it must fall through the threshold. In Threshold mode the crossing can be in either direction. The picture shows a rising threshold. For waveform channels, the crossing point is found by linear interpolation of the data points on either side of the threshold crossing. The cursor position is invalid if a crossing does not occur.

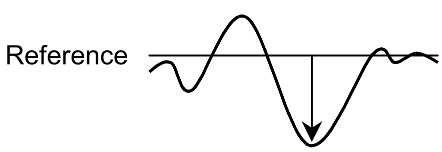


Maximum value, Minimum value

The result is the position of the maximum or minimum value. The result is invalid if there is no data in the search range.

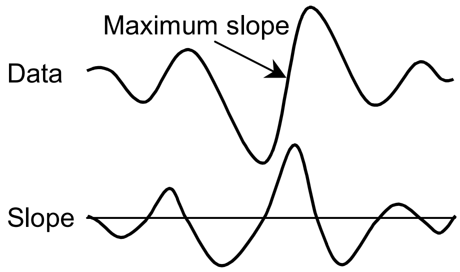
Maximum excursion

There is an extra field in this mode for the Reference level. The cursor is positioned at the point that is the maximum distance in the y direction away from the reference level. The result is invalid of there is no data in the search range.



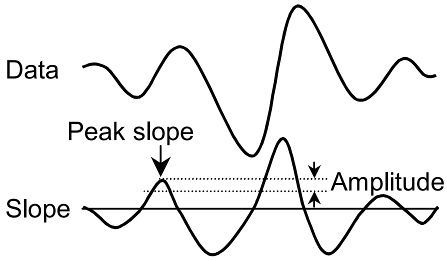
Steepest rising, Steepest falling, Steepest slope (+/-)

These modes are for waveform channels only. They have the extra field Width for slope measurement that sets length of data used to evaluate the slope at each data point. The result is the position of the maximum, minimum or maximum absolute value of the slope. The result is invalid if there is no data or not enough data to calculate a slope or if the channel is not waveform or WaveMark.



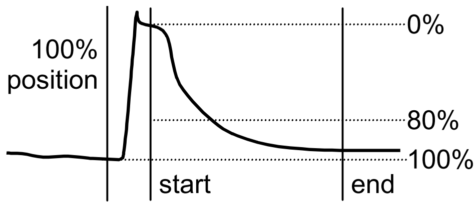
Slope peak, Slope trough

These two analysis modes calculate the slope of the data in the search range, and then search for the first peak or trough in the result that meets the Amplitude specification. The Width for slope measurement field sets the length of data used to evaluate the slope at each data point. The Amplitude field sets how much the slope must rise before a peak and fall after it (or fall before a trough and rise after it), for it to be accepted as a peak. The Amplitude units are y axis units per second.



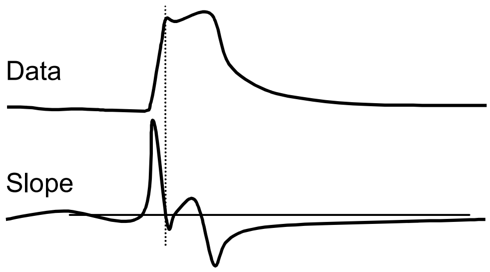
Repolarisation %

This mode finds the point at which a waveform returns a given percentage of the distance to a baseline inside the search range. The start of the search range defines the position of 0% repolarisation. The additional fields, 100% position and Width, identify the 100% level (this can lie outside the search range). The Repolarise % field (drawn at 80% in the picture) sets a threshold level in percent relative to the 0% and 100% levels. The position is the first point in the search range that crosses the threshold. The result is invalid if the threshold is not crossed.



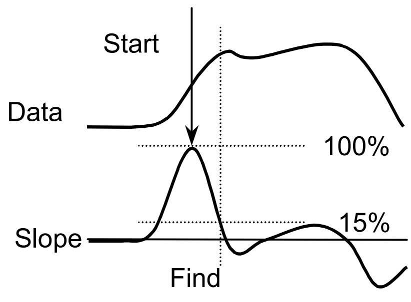
Inflection

This mode finds the first point in the search range where the slope of a waveform changes sign. Put another way, it finds a localised peak or trough in the waveform. The Width for slope measurement field sets the data range to calculate the slope. The picture shows this method used to find the top of a sharp rise where Maximum mode would get the wrong place. To use this you would probably set a cursor on the peak slope and start the search from that point. The result is invalid if no point of inflection is found.



Slope%

This method can be used to find the start and end of a fast up or down stroke in a waveform. The Width for slope measurement field sets the time width used to calculate the slope. The Slope% field sets the percentage of the slope at the start of the search area to search for. To use this mode you would set a cursor on the maximum or peak slope, then use that as the start point and search for the required percentage. A value of 15% usually works reasonably well. The result is valid if the slope value is located.



Data points The result is the position of the first data value in the search range on the referenced channel. For an event or marker channel, this is the time of the first event or marker. For a waveform channel this is the first waveform point in the time range. For a result channel, this is the next bin. The cursor position is invalid if there is no data item.

Expression The cursor position is obtained by evaluating the Position field. This field will normally hold an expression based on cursor positions.

Delete

The delete command activates a pop-up menu from which you can select a vertical cursor to remove, or you can delete all the cursors. The cursors are listed with their number and position as an aid to identification. Deleting a cursor removes it from the window; other cursors are not affected.

Fetch

This activates a drop down list from which you can select a vertical cursor to place in the centre of the x axis. The keyboard short cut `Ctrl+n` where `n` is a cursor number will fetch cursor `n` if it exists or will create it if it doesn't exist.

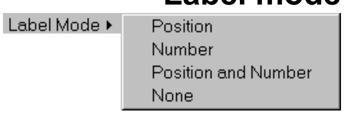
Move To

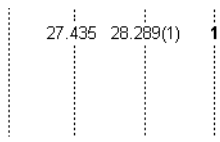
This command opens a pop-up menu from which you can select a vertical cursor to move to. The cursors are listed with their number and position as an aid to identification. The window is scrolled to display the nominated cursor in the centre of the screen, or as close to the centre as possible. This command does not change the x axis scaling. You can also use the keyboard short cut `Ctrl+Shift+n` where `n` is a cursor number.

Display All

This command has no effect if there are no active cursors. If there is a single cursor, the command behaves as though you had used the Move To command and selected it. When there are multiple cursors, the window is scrolled and scaled such that the earliest cursor is at the left-hand edge of the window and the latest is at the right-hand edge.

Label mode



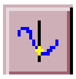


There are four cursor label styles: None, Position, Position and Number, and Number. You select the most appropriate for your application using the pop-up menu. To avoid confusion between the cursor number and the position, the number is displayed in bold type when it appears alone and bracketed with the position. The style applies to all the vertical cursors in the window. You can drag the cursor labels up and down the cursor with the mouse to suit the data.

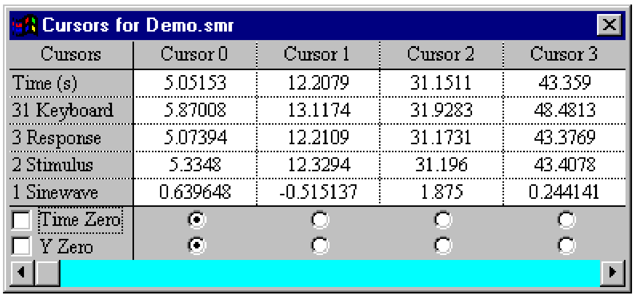
Renumber

When created, cursors take the lowest available cursor number. You can also drag cursors over each other. This command renumbers the vertical cursors, with cursor 1 on the left.

Display y values



This command opens a new window containing the values at any cursors in the current time or result window. Cells for cursors that are absent, or for which there is no data, are blank. The new window is a top level window and will never go behind another time or result view.



Cursors	Cursor 0	Cursor 1	Cursor 2	Cursor 3
Time (s)	5.05153	12.2079	31.1511	43.359
31 Keyboard	5.87008	13.1174	31.9283	48.4813
3 Response	5.07394	12.2109	31.1731	43.3769
2 Stimulus	5.3348	12.3294	31.196	43.4078
1 Sinewave	0.639648	-0.515137	1.875	0.244141
Time Zero	●	○	○	○
Y Zero	●	○	○	○

In a result window, the value displayed is the value to be found in the bin to the right of the cursor. If you position the cursor at the far right (where there is no bin to the right), a blank is displayed as the value.

In a time window, the values displayed depend on the channel display mode. There is an entry in the table for each channel displayed. The Waveform mode also applies to Cubic spline, Sonogram, and waveforms drawn as dots. The values are as follows:

Waveform — This applies to waveform data and also to WaveMark data displayed as a waveform or as WaveMark data. The value displayed is the value of the nearest data point that is within one sample period of the cursor position. If the cursor lies in a gap in the waveform, no value is returned and the field is blank.

Rate — The value displayed is the height of the rate bin that the cursor crosses. If the cursor lies precisely on the boundary between two rate bins, the cursor is considered to lie in the bin to the right.

Mean frequency — The value is the mean frequency at the cursor.

Instantaneous — The instantaneous frequency is calculated by taking the reciprocal of the time gap between the first event to the right of or at the cursor and the event before it. If there is no event to the right, or no event before the cursor, the value is blank.

Event — This applies to all remaining display types. The value shown is the time of the next event or marker at or to the right of the cursor.

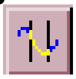
The Time zero check box enables relative cursor times. If checked, the cursor marked with the radio button is taken as the reference time, and the remaining cursor times are given relative to it. The reference cursor displays the absolute time, not 0. In the example above, cursor 1 has been set as the reference.

The Y zero check box enables relative cursor values. The radio buttons to the right of the check box select the reference cursor. The remaining channels display the difference between the values at the cursor and the values at the reference. The values for the reference cursor are not changed.

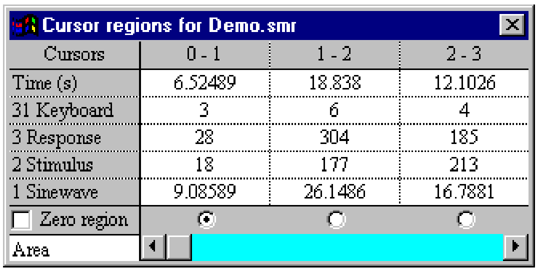
Selecting and copying data

You can select areas of this window and copy them to the clipboard by clicking on them with the mouse. Hold down the Shift key for extended selections. You can select entire rows and columns by clicking in the cursor and channel title fields. Use the Control key to select non-contiguous rows and columns.

Cursor regions



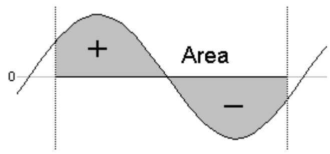
This command opens a cursor region window for the current time or result view. This window calculates values for data regions between the cursors. One column can be designated the Zero region by checking the box and selecting the column with a radio button. The value in this column is then subtracted from the values in the other columns (except in Area(scaled) mode which scales the zero region value to allow for column widths). The field at the bottom left indicates how to calculate the values; click on it for a full list. Cells for which there are no cursors or data are blank. You can interrupt long calculations in time windows with the `Ctrl+Break` key combination.



Time window waveform data and result window data

In a time window, waveform channels include WaveMark channels drawn as waveforms, WaveMark or Cubic Spline and waveform channels drawn as dots, Cubic spline or Sonogram. The region set by a pair of cursors is the data starting at the first cursor up to, but not including, the data at the second cursor. In a result window, the region set by a pair of cursors starts at the bin containing the left cursor and ends in the bin to the left of the bin containing the right cursor. The values for each mode are:

Area — The area between the data points and the y axis zero. Area is positive for curve sections above zero and negative for sections below zero. Use Modulus if you want areas below the y axis to be treated as positive.



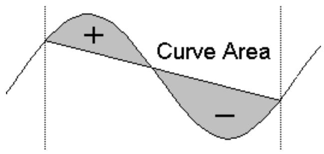
Mean — The sum of all the data points between the cursors divided by the number of data points between the cursors.

Slope — The slope of the least squares best fit line to the data between the cursors.

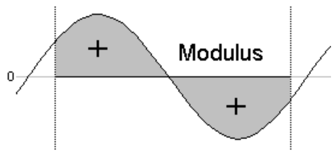
Sum — The sum of the data values between the cursors. If there are no samples between the cursors the field is blank.

Area (scaled) — The same as Area, but if a zero region is specified, the amount subtracted from the other regions is scaled by the relative width of the regions.

Curve area — Each data point makes a contribution to the area of its amplitude above a line joining the endpoints multiplied by the x axis distance between the data points. The picture makes this clearer.



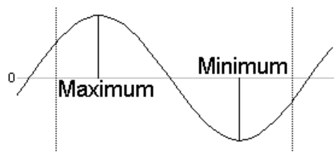
Modulus — Each data point makes a contribution to the area of its absolute amplitude value multiplied by the x axis difference between data points. This is equivalent to rectifying the data, then measuring the area. If a zero region is specified, the amount subtracted from the other regions is scaled by the relative width of the regions.

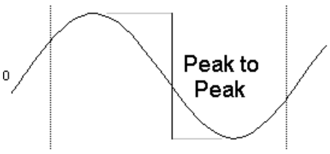


Maximum — The value shown is the maximum value found between the cursors.

Minimum — The value shown is the minimum value found between the cursors.

Extreme — The value shown is the maximum absolute value found between the cursors. If the maximum value was +1, and the minimum value was -1.5, then this mode would display 1.5.

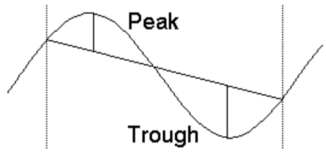




Peak to Peak The value shown is the difference between maximum and minimum values found between the cursors. The peak to peak value is always positive.

SD The standard deviation from the mean of the values between the cursors. If there are no values between the cursors the field is blank.

RMS The value shown is the RMS (Root Mean Square) of the values between the cursors. If there are no values between the cursors the field is blank.



Peak The value shown is the maximum value found between the cursors measured relative to a baseline formed by joining the two points where the cursors cross the data. This is always greater than or equal to 0.

Trough The value shown is the minimum value found between the cursors measured relative to a baseline formed by joining the two points where the cursors cross the data. This is always less than or equal to 0.

Event channels All measurement modes except those listed below produce a blank field in the window:

Mean The total number of events or markers between the cursors divided by the time difference between the cursors. This could be thought of as the mean event rate.

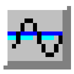
Area The total number of events or markers on the channel between the cursors.

Area (scaled) The same as Area, but if a zero region is specified, the amount subtracted from the other regions is scaled by the relative width of the regions.

Sum The same as Area.

Selecting and copying data Fields from this window can be copied to the clipboard. To do this, select the region to be copied and then click the right mouse in the window and use the Copy command. To select an entire row or column, click the mouse in the titles at the top and left of the window. To extend a selection, hold down the Shift key. To select non-contiguous rows or columns hold down the control key and select the rows and columns as required.

New Horizontal



The command is available when a time or result window is the current window, and there are less than four horizontal cursors already active. A new horizontal cursor is added to the top-most visible channel with a y axis. The cursor is given the lowest available horizontal cursor number and is labelled with the cursor label style for the window.

Delete Horizontal

The delete command activates a pop-up menu from which you can select a horizontal cursor to remove, or you can delete all the horizontal cursors. The cursors are listed with their number, channel number and position as an aid to identification. Deleting a cursor removes it from the window; other cursors are not affected.

Fetch Horizontal

This activates a drop down list from which you can select a horizontal cursor to place in the centre of the visible y axis range of the channel. This does not make a channel visible if the horizontal cursor is attached to an invisible channel.

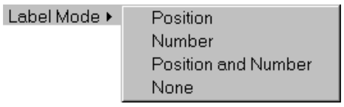
Move To Horizontal

This command opens a pop-up menu from which you can select a horizontal cursor to move to. The cursors are listed with their number and position as an aid to identification. The window is scrolled vertically to display the nominated cursor in the centre of the y axis range.

Display all Horizontal

This has no effect if there are no active cursors. If there is a single cursor, the command behaves as though you had used the Move To Horizontal command and selected it. When there are multiple cursors, the channel y axis range is adjusted such that the lowest cursor is at the bottom edge of the channel area and the highest is at the top edge.

Horizontal Label mode



There are four cursor label styles: no label, position, position and cursor number, and number alone. You select the most appropriate for your application using the pop-up menu. To avoid confusion between the cursor number and the position, the number is displayed in bold type when it appears alone and bracketed with the position. The style applies to all the horizontal cursors in the window. You can drag the cursor labels left and right along the cursor to suit the data.

Renumber Horizontal

When created, cursors take the lowest available cursor number. You can also drag cursors over each other. This command renumbers the cursors, with cursor 1 at the bottom.

Sample menu

The Sample menu is divided into several regions. The first region is used before sampling to configure the channels required for data capture and for waveform output and to configure the Sample Bar. The second region is for users with the optional 1401-18 discriminator card fitted to their 1401 or 1401*plus*, and for users with a serial line controlled signal conditioner, for example the CED 1902. The third region of the menu is used during sampling to start and end the sampling process, enable and disable data storage to disk and to display the sampling controls and the output sequence controls. The final command inserts timed comments into data files during sampling.

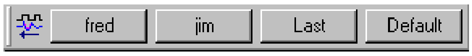
Sampling configuration

This command opens the Sampling Configuration dialog that sets the sampling parameters used when you select the File menu New command. You can load and save the configuration from the File menu Save Configuration and Load Configuration commands. The dialog is described in detail in the *Sampling data* chapter.

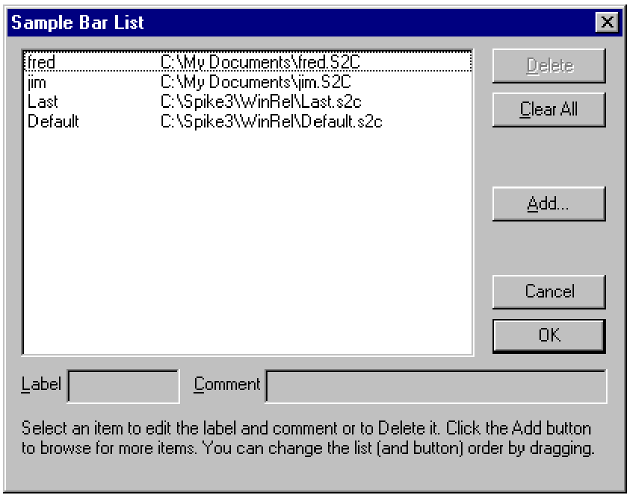
Clear configuration

This command deletes all sampling window positions, associated result views for on-line processing, display trigger settings and WaveMark templates. It preserves the channel lists and all values visible in the Sampling Configuration dialog.

Sample Bar



You can show and hide the Sample Bar and manage the Sample Bar contents from the Sample menu. The Sample Bar is a dockable toolbar with up to 20 user-defined buttons. Each button is linked to a Spike2 configuration file. When you click a button, the associated configuration file is loaded and a new data file is opened, ready for sampling. You can also show and hide the Sample Bar by clicking the right mouse button on any Spike2 toolbar or on the Spike2 background.



The Sample menu Sample Bar List… command opens the Sample List dialog from where you can control the contents of the Sample Bar.

The Add button opens a file dialog in which you can choose one or more Spike2 configuration files (`*.S2C`) to add to the bar. If a file holds a label or comment, it is used, otherwise the first 8 characters of the file name form the label and the comment is blank.

You can select an item in the list and edit the label and comment. This does not change the contents of the configuration file. You can re-order buttons in the bar by dragging items in the list. Delete removes the currently selected item. Clear All deletes all items. Spike2 saves the list of files in the Sample Bar in the registry. Each logon to Windows has its own registry settings, so if your system has three different users each has their own Sample Bar settings. Alternatively, you can have different experimental configurations by logging on with a different user name.

Discriminator Configuration

If your 1401*plus* has a CED 1401-18 event discriminator fitted, this command opens a dialog box in which you can configure it (see the *1401-18 Programmable Discriminator* chapter for a full description). The event discriminator converts waveform signals into digital data suitable for input as event and level data channels. This command is not available during sampling. There are script language commands that can be used during data capture to change the discriminator settings.

Conditioner Settings

Spike2 supports serial line controlled programmable signal conditioners. These devices amplify and filter waveform signals, and can provide other specialist functions. If a suitable conditioner is installed in your system, this menu command opens the conditioner dialog (see the *Programmable signal conditioners* chapter for a full description).

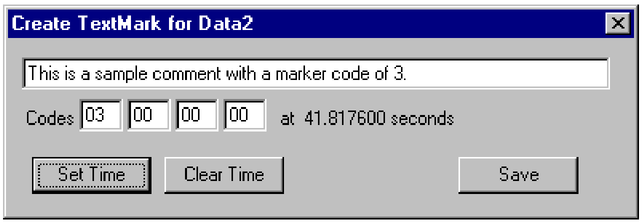
Sampling controls

This command hides and shows the Sample control toolbar. The controls within this toolbar are duplicated in this menu as the Start sampling, Write to disk, Abort sampling and Reset sampling commands (see the *Sampling data* chapter for details of the floating window and menu commands).

Sequencer controls

This command hides and shows the sequencer control panel that is available during sampling if an output sequence is active (see the *Data output during sampling* chapter for details of the panel).

Create a TextMark



This command opens the Create TextMark window. You can make text markers set marker codes while you sample data if you set a TextMark channel in the Sampling Configuration. If you enabled serial line input of TextMark data in the Sampling Configuration you can also add markers by hand.

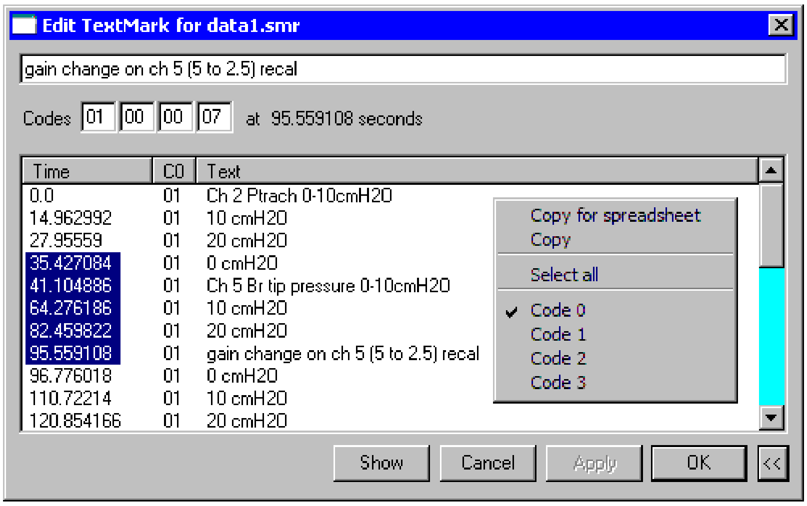
You commit the comment to the file when you press the Save button. The Set Time button fixes the time associated with the comment, the Clear Time button clears the time. If you have not set a time, the time of the comment is the time when it was saved.

If this channel is the trigger for Triggered sampling mode in the sampling configuration dialog, any times set by the Set Time button are ignored and each time you save the time, the 1401 is told to trigger and the trigger time is recorded with the text.

Comments are shown in the file as a small rectangle. The colour of the rectangle depends on the first marker code associated with the marker. If the code is 00, the rectangle is yellow. Otherwise, codes 01-08 use the same colours as WaveMark data, set by the View menu Change Colours command. The colours repeat every 8 codes.

You can see the text associated with a TextMark by moving the mouse pointer over the rectangle and waiting a moment. The text appears as a tool tip. Any movement of the mouse pointer hides the text.

To edit the comment text and marker codes, double click the rectangle to open the Edit TextMark dialog. You can also use this dialog to navigate to any marker or range of markers in the file. The Apply button saves any changes without closing the dialog. OK saves changes and closes the dialog.

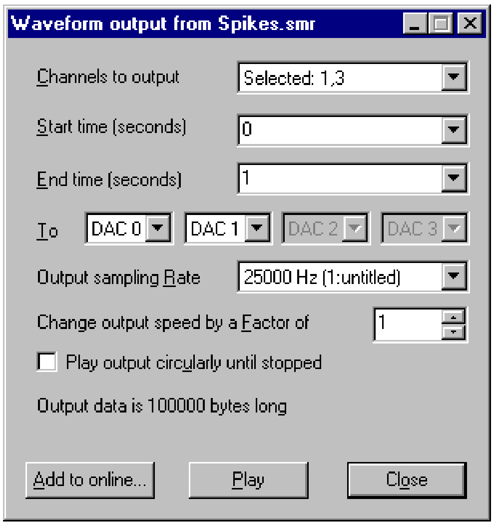


The list area of the dialog is hidden with the << button and restored with the >> button. Double-click an item in the list to edit it and save any changes to the item that was previously displayed. The Show button moves the time view display to the currently selected TextMark. If you have made multiple selections, the first marker is displayed at the left edge of the screen and the last one at the right edge.

Right-click in the list to control how many marker codes to display and to copy data to the clipboard. If you choose Copy for spreadsheet, text strings and marker codes are enclosed in quotation marks so that the data will import easily into spreadsheet programs. You can change the sort order of the data in the list by clicking the column titles. Initially, the items are sorted based on the Time column.

Offline waveform output

When a data file is active, the Sample menu Waveform Output… command opens the Waveform output dialog.



Channels to output

You can play up to 4 waveform or WaveMark channels through the 1401 DACs, or up to two channels through a sound card. To output multiple channels, select them in the time view, otherwise you can select single channels. You can change the channel selections after you open this dialog. Spike2 fills gaps in waveform data and gaps between WaveMark data items with zeros.

Time range

The Start time and End time fields set the file time range to play. You can link these to cursors in the time view. The values used are those set when you click Play.

To

You can select either a 1401 DAC for each channel, or if you select Sound in the first drop down list, the first (and second) channel is played through the sound card (mono for one channel, stereo for two channels).

Output sampling Rate You can choose the sampling rate of any of the channels for output. All channels are replayed at the same speed. If any channel has a different sample rate, Spike2 resamples the data to have the same rate as set by this field (using linear interpolation).

Change output speed by a Factor of You can speed up or slow down output by a factor of up to 4. This will change the pitch of sound or the repeat rate of a repetitive signal.

Play output circularly If you do not check this box, the waveform plays once only.

Play and Stop This button starts output (and changes to Stop, which stops it). The output pays no attention to the scale and offset values associated with the channel. The 16-bit data values recorded for the file are played out. If you have an 8-bit sound card and try to play a low level signal, the result may be disappointing (or just silence).

Add to online… This option adds the selected waveforms to the on-line waveform output list. As long as the list of waveforms is not full you can Add the waveform to the list. You can also Replace an existing waveform in the list. You can give each waveform a short name that labels the buttons for interactive replay of waveforms during data acquisition. There are more controls for the list of waveforms for replay in the Sampling Configuration dialog.

Waveforms can be saved as either a data file, time range and a list of channels or by converting the current channel list into the memory image that would be played through the output DACs on line. If a waveform has been converted the Source column in the list starts with "Memory". The advantages of converting are:

- The output is not affected if the original file is moved or deleted or changed
- The (usually small) time for the conversion is saved each time you sample data

However, there are also disadvantages to converting:

- It can take a lot of memory to hold the converted data which can slow Spike2 down and may cause your system to run out of memory
- If the sampling configuration is saved, the converted data is also saved, which makes the configuration files much larger and can slow down system operation.

To prevent the system running out of memory, there is a limit on the amount of memory that any one converted wave may use. If any channel in the play list is a WaveMark or a memory channel or a duplicate channel, Spike2 will always attempt to convert the data, as there is no guarantee that these channels will exist (or have the same marker filter) when sampling is requested.

If a channel has a channel process applied to it, you must either leave the data file open with the channel process applied or you must convert the wave. This is because a channel process only has effect while the data file is open in Spike2.

Script menu

The Script menu gives you access to the scripting system. From it you can compile a script, run a loaded script, evaluate a script command for immediate execution, record your actions as a script and convert a script from the format used in the DOS version of Spike2 to this version. You can find details of the script language and a description of the script window in the separate manual *The Spike2 script language* and in the on-line help. The menu commands are:

Compile Script

This option is enabled when the current window holds a script. It is equivalent to the compile button in the script window. Spike2 checks the syntax of the script, and if it is correct, it generates a compiled version of the script, ready to run.

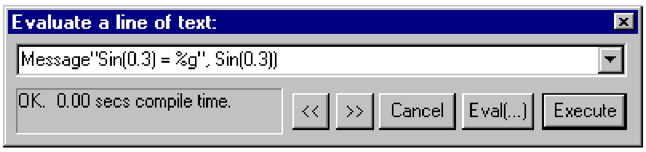
Run Script

This option pops-up a list of all the scripts that have been loaded so that you can select a script to run. Spike2 compiles the selected script and if there are no errors, the script runs. If you run a script twice in succession, Spike2 compiles it for the first run, and uses the compiled result for the second run, saving the compilation time. If a script stops with a run time error, the script window is brought to the front and the offending line is highlighted.

You can also select the Load and Run… option from which you can select a script to run. The script is hidden and run immediately (unless a syntax error is found in it).

Evaluate

This command opens the evaluate window where you can type in and run one line scripts. You can cycle round the last 10 script lines with the << and >> buttons. The Execute button compiles and runs your script, Eval() does the same, and also displays the value of the last expression on the line.



Turn Recording On/Off

You can record your actions into the equivalent script. Use this command to turn recording on and off. When you turn recording off, a new script window opens that holds the commands that are equivalent to your actions. When recording is enabled REC appears in the status bar.

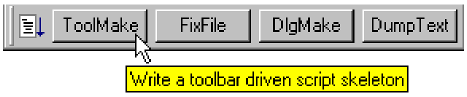
Debug bar

You can show and hide the debug bar from this menu when the current view is a script. You can also show and hide the debug bar by right clicking on the title bar of the Spike2 window, or on the application window.

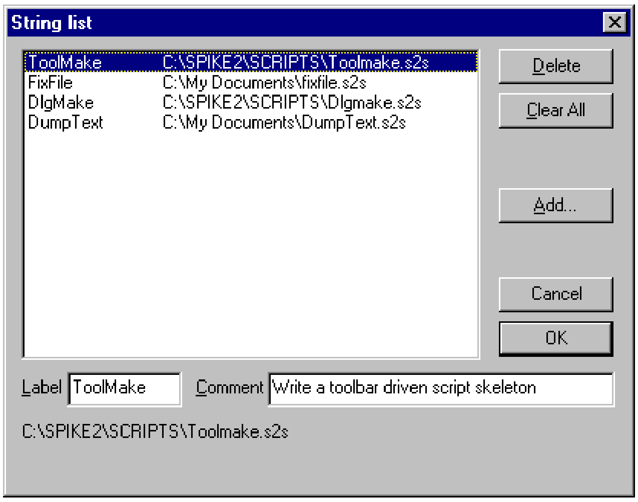
Convert DOS Script

This option is only available when a text window is open that is not empty. The command will attempt to convert a Spike2 for DOS script into the equivalent commands for this version of Spike2. This option is described in detail in the script manual.

Script Bar



You can show and hide the Script Bar and manage the Script Bar contents from the Script menu. You can also show and hide the Script Bar by clicking the right mouse button on any Spike2 toolbar or on the Spike2 background. The Script Bar is a dockable toolbar with up to 20 user-defined buttons. Each button is linked to a Spike2 script file. When you click a button, the associated script file is loaded and run. There is also a user-defined comment associated with each button that appears as a tool-tip when the mouse pointer lingers over a button.



The Script menu Script Bar List… command opens the Script List dialog from where you can control the contents of the Script Bar.

The Add buttons opens a file dialog in which you can choose one or more Spike2 script files (`*.S2S`) to add to the bar. If the first line of a script starts with a single quote followed by a dollar sign, the rest of the line is interpreted as a label and a comment, otherwise the first 8 characters of the file name form the label and the comment is blank. The label is separated from the comment by a vertical bar. The label can be up to 8 characters long and the comment up to 80 characters. A typical first line might be:

```
'$ToolMake|Write a toolbar driven script skeleton
```

You can select an item in the list and edit the label and comment. This does not change the contents of the script file. You can re-order buttons in the bar by dragging items in the list. The Delete button removes the selected item. Clear All removes all items from the list.

The list of files in the Script Bar is saved in the registry when Spike2 closes and is loaded when Spike2 opens. Each different logon to Windows has a different configuration in the registry, so if your system has three different users each has their own Script Bar settings. Alternatively, you can have different experimental configurations by logging on as a different user name.

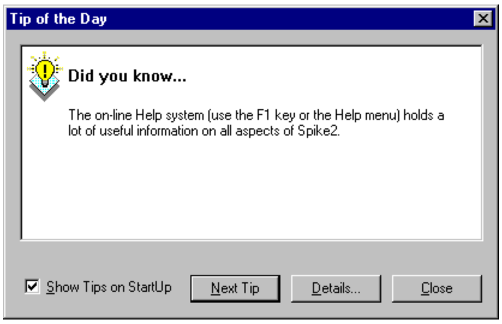
Using help

Spike2 supports context sensitive help that duplicates the contents of this manual and the script language manual. You can activate context sensitive help with the F1 key from most dialogs to get a description of each field in the dialog. You can use the help Search dialog to lookup topics that are not covered by the index.

In the script window you can obtain help by placing the cursor on any keyword in the script and pressing F1. To get help on a script function, type the function name followed by a left hand bracket, for example `MemChan(`, then make sure that the cursor lies to the left of the bracket and in the function name and press F1.

The help system has indexes, hypertext links, keyword searches, help history, bookmarks and annotations. If you are unsure of using help, once you have opened the help window, use the Help menu Using help command for detailed instructions.

Tip of the Day



You can show the Tip of the Day window when Spike2 starts by checking the box. Of course, if you hate this sort of thing, clear the check box and you never need see it again! The Details… button opens the on-line Help at a page that holds more information about the topic in the window. The tips are held in the hidden file `sonview.tip` in the same folder as Spike2. If you have a useful tip that you would like to share with others, send it to us and we will very likely include it in the next release.

View Web site

If you have an Internet browser installed in your system, this command will launch it and attempt to connect to the CED web site (`www.ced.co.uk`). The site contains down-loadable scripts, updates to Spike2 and information about CED products.

Getting started

This command runs a demonstration script that gives a quick tour of Spike2 basics. The option is disabled if the script is not present or if you are sampling data.

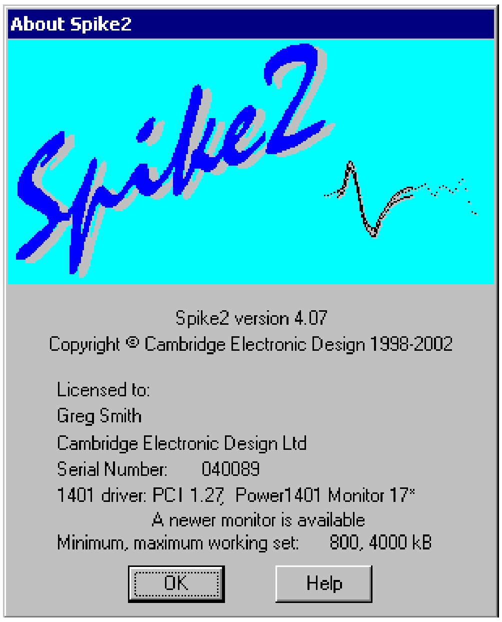
Other sources of help

If you are having trouble using Spike2, please do the following before contacting the CED Software Help Desk:

1. Read about the topic in the manual. Use the Index to search for related topics.
2. Try the help system for more information. Use Search to find related topics.
3. If the problem is in sampling data, run the Try1401 program provided as part of Spike2. This checks out the 1401, device driver and interface card.
4. If none of the above helps, FAX, email (`softhelp@ced.co.uk`) or call the CED Software Help Desk (see the front of this manual for details). Please include a problem description, Spike2 serial number and program version and a description of the circumstances leading to the problem. It would also help us to know the type of computer you use, how much memory it has and which version of your operating system you are running.

About Spike2

This command is found in the Help menu. It opens an information dialog that contains the serial number of your licensed copy of Spike2, plus your name and organisation. Please quote the serial number if you call us for software assistance.



1401 device driver

If there is a 1401 device driver installed, the driver revision is displayed. If the driver is older than Spike2 expects, you will be warned. Spike2 displays the type of 1401 and the monitor version if a 1401 is connected and powered up.

1401 Monitor revision

If the monitor is not the most recent at the time this version of Spike2 was released, an asterisk follows the version. If it is so old that it compromises data sampling, two asterisks follow the version. New monitor ROMs are available from CED for the 1401*plus* and the micro1401. You will need to open the 1401 case to replace them; we ship detailed instructions with the ROM. The Power1401 and Micro1401 mk II have firmware in flash memory. Flash updates can be obtained from the CED web site; you can update the flash firmware without opening the 1401 case.

Working set size

If you are running Windows NT, NT 2000 or Windows XP, there is information about the Working Set Size at the bottom of the About box. The two numbers describe the minimum and maximum physical memory that the operating system allows Spike2 to use. If you use Windows NT and suffer from error -544 when you sample data, these numbers are important. You will find more detailed information in the on-line help (look up "Working Set" in the on-line help index).

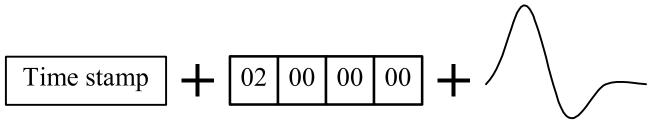
Free system resources

If you are running Windows 95, 98 or Me, the working set information is replaced by the Free 16-bit system resources for the GDI (graphic objects) and User (all other objects). The figures given are the percentage of free resources compared to the state when the system started up. If either of these figures gets less than 10% you will find that system performance is severely impacted, windows may not open and images may be missing from buttons. If free resources reach 0%, Windows tries to warn you; unfortunately, as resources have reached 0 the message box may not be legible.

The usual cause of running out of resources is to open many windows at the same time, usually from a script. On my Windows 98 system I can generate about 90 result windows before I run out, as long as Spike2 is the only open application. If system resources is a problem, consider changing to Windows NT, NT 2000 or XP where this is not an issue.

Spike shapes

Introduction



Spike2 stores spike shapes in the WaveMark channel type. Each spike shape has a time stamp, 4 identifying codes and up to 1000 waveform data points that define the spike (on-line you can use up to 126 points). The codes store the spike classification, obtained by matching the spike shape against shape templates; normally only the first code is used. The time stamp is the time of the first saved data point.

On-line, spikes can be sorted in real-time with up to 8 spike classes per channel and an optional digital output as each spike is classified. Spike2 captures up to 32 channels of spikes with a Power1401, up to 16 channels with a Micro1401 mk II and up to 8 channels with a 1401*plus* or a micro1401.

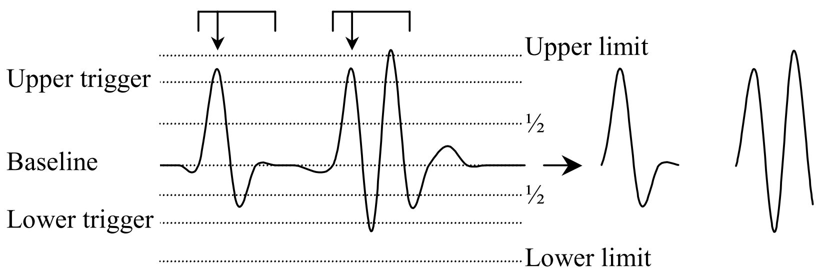
Off-line Spike2 can extract spikes from a waveform channel and resort and edit spikes extracted on-line or off-line; you can divide spikes into almost any number of classes.

The first part of this section is organised as reference information. This is followed by *Getting started with spike shapes and templates* which is more discursive. If you are a new user you may prefer to start there and come back to the reference material later.

Spike detection

Spikes are detected by the input signal crossing a trigger level. There are two trigger levels, one for positive-going and one for negative-going spikes. Captured spikes are aligned on the first positive or negative peak. To avoid problems due to baseline drift, the data capture routines pass the data through a high pass filter. The time constant of this filter can be adjusted to suit the data. You can define the number of data points captured before and after the peak. The spike detection algorithm is as follows:

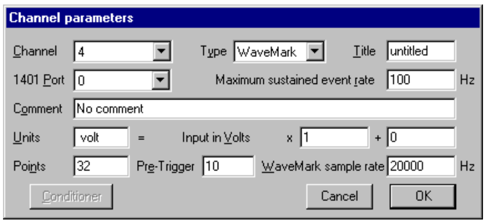
1. Wait for the signal to lie within half the trigger levels. When it does, go to step 2.
2. Wait for the signal to cross either trigger level. If it crosses the upper trigger level go to step 3. If it crosses the lower level, go to step 4.
3. Track the positive peak signal value. If the signal falls below the peak, see if we have sufficient post-peak points to define the spike. If so, go to 5. If the signal falls below half the upper trigger level, we ignore further peaks (as for the second spike in the diagram).
4. Track the negative peak signal value. If the signal rises above the peak, see if we have sufficient post-peak points to define the spike. If so, go to 5. If the signal rises above half the lower trigger level, we ignore further peaks.
5. Save the waveform and first data point time and go to step 1 for the next spike.



When working offline, you can choose to set two additional *limit* levels outside the trigger levels. Any detected spikes with a peak amplitude outside these limits are rejected. In this case the trigger and associated limit act as an amplitude window on the data. The limit applies to the initial peak amplitude, so the second spike in the diagram is acceptable, even though some of the data points lie outside the limits.

Sampling parameters

To sample spike shapes you must create a WaveMark channel in the Sampling configuration dialog. The Title, Port, Channel comment, Units and Scaling fields are the same as for Waveform data. The Maximum sustained event rate field is the total number of spikes (of all classes) that you expect per second on this channel.



The bottom line of the dialog box sets the number of points to save per spike in the range 10 to 126 (Points) and the number of points to display before the peak (Pre-trigger). You can alter the number of points before the peak and the total number of points in the template setup section, so it is not essential to get this exactly right.

The WaveMark sample rate field sets the ideal sampling rate for all WaveMark channels. A sampling rate of 20 kHz per channel is often used so that spikes (of typical duration 1 to 2 milliseconds) can be usefully discriminated. The actual sampling rate depends on the number of waveform and WaveMark channels. See the Sampling configuration Resolution tab for more information on setting waveform and WaveMark sample rates.

Template matched signal

You can use digital output bits 0-7 to signal that a spike shape has matched a template. There are two output formats (set in the on-line Template Setup). In One bit mode, a single bit pulses from low to high, then back to low for each of up to 8 templates. Bit 0 if for the first template, bit 1 for the second, and so on up to bit 7 for the eighth template. One bit mode is not useful when you have more than one channel of WaveMark data as you cannot tell which channel produced the output.

In Coded mode, pin 23 of the digital output port pulses for each matched template, and the digital output bits 3-7 hold the channel number-1 (0 to 31) and bits 0-2 hold the template number for that channel. The data is valid on both edges of the pulse. The pulse is short, less than 1 µs for the Power1401, more than 1 µs for other 1401s.

Digital output connections

Data bit	7	6	5	4	3	2	1	0	Gnd	Strobe
Output pin	5	18	6	19	7	20	8	21	13	23

1401*plus* digital bits 0-7 are used for both input and output. They are normally inputs. The state of bits used as outputs read back in the sequencer `DIGIN` and `WAIT` instructions. Coded mode uses all 8 bits, One bit mode uses one bit per template. The micro1401 and Power1401 have independent input and output bits 0-7, however you should use the output sequencer `DIGLOW` command with caution as it controls the same output bits.

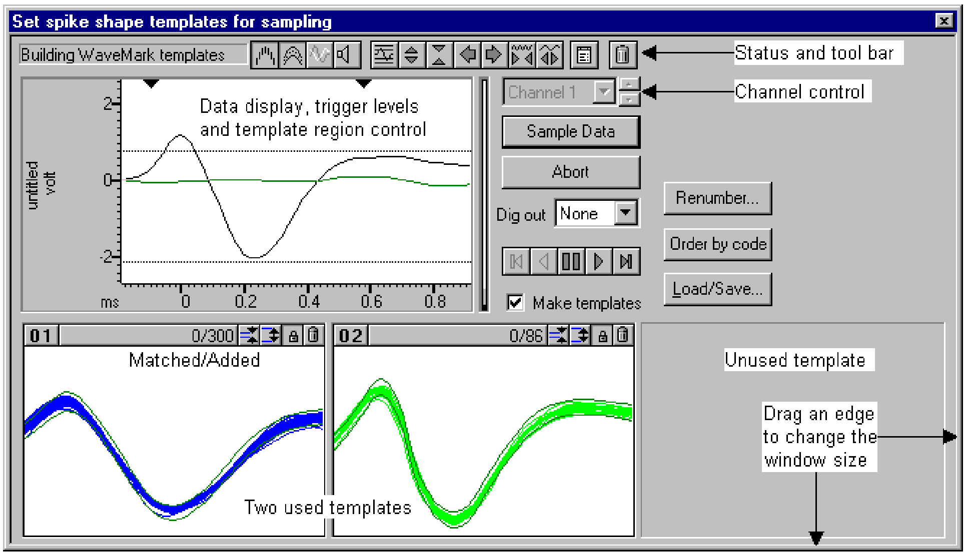
Timing of the digital output

Measurements with templates of 20 points show that the majority of spikes are matched within 2 milliseconds of the end of the spike (much less than this with a Power1401). However you cannot rely on this. Even with a PCI interface card, and a fast host, delays of 30 milliseconds or more may occur occasionally, especially when data is written to disk at high sampling rates. Longer delays will occur with slower hardware.

You can test the delays by feeding the output back into an event channel and correlating the Spike channel with the event channel. If you intend to use these outputs for a timing critical use it is important that you measure the typical time delays for your configuration.

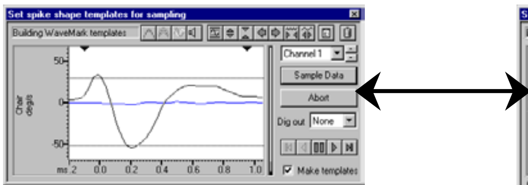
On-line template setup

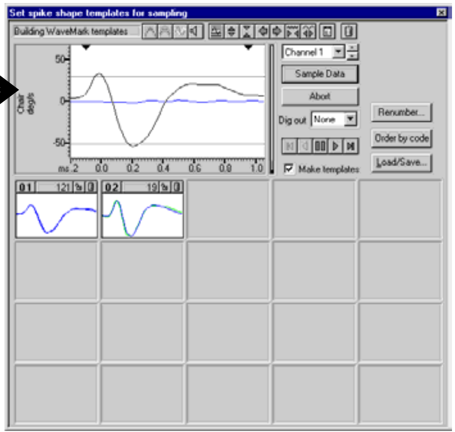
When you create a New file with WaveMark channels, Spike2 opens the template setup window. The window contains four main regions: a status and tool bar at the top, a data display area with the trigger levels, a control area on the right and a region with up to 20 templates at the bottom. The first 8 templates are used on-line.



The display area shows data from the channel set by the Channel control. Waveforms that meet the trigger criteria are drawn in the WaveMark 00 colour. Those that do not appear in the time view waveform colour. The horizontal lines are the trigger levels; drag them with the mouse to change the levels. The two black triangles at the top of this area mark the start and end of the region that is used to form templates. The vertical bar on the right of the data display shows the number of templates that the program is considering in grey, with black representing the number of confirmed templates.

You can change the window size by dragging any window edge. Double click the title bar of the window to maximise (or minimise if already maximised) the window.





The window size can be changed from very small (minimised), with no templates, to large (maximised) with all 20 possible templates. The templates can be either small or large. Double click on one of the used or unused template rectangles to change the template size.

Channel control

When there is more than one channel holding WaveMark data, you can change the channel either by dropping down the list of channels and selecting one, or by using the small spin box on the right of the channel control to move to the next channel in the list. Each channel has its own set of templates and template parameters.

Sample data

Once you have adjusted the trigger levels and set the templates for each channel, click this button to close the window and sample data.

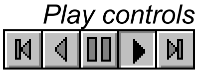
Abort

This button closes the window and does not sample data.

Dig out During sampling, the 1401 can flag template matches using bits 0-7 of the digital port. You can select between None (no output), One bit (single pulsed digital bit per template) or Coded (data channel and template coded in the 8 data bits).

Make templates With this box checked, detected spikes are offered to the template matching system and will create and modify templates. With this box unchecked, spikes are compared against templates but the templates do not change. When you open this window, this box is checked if there are no existing templates, otherwise it is clear.

Play controls



These five buttons control data replay. The centre button pauses replay and the buttons either side of it play the data forwards and backwards. The outer buttons step one spike forwards and backwards. You can only play or step backwards in the Edit WaveMark command. There are keyboard shortcuts for these controls. `B` steps backwards one spike, `M` steps forwards one spike, `n` runs forwards, `N` runs backwards and `V` pauses output.

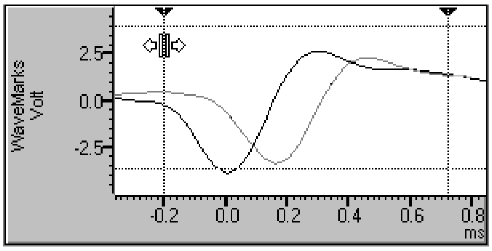
Renumber… This prompt you for a starting marker code then renumbers the templates with consecutive codes, starting with the code you set. The templates remain in the original positions and code order. If two or more templates share a marker code they will still share a code after renumbering. For example, templates coded 01, 04, 04, 07 and 03 renumbered to start at 01 would end up with codes 01, 03, 03, 04 and 02.

Order by code This button rearranges the templates so that they are in ascending marker code order. For example, templates with marker codes 01, 04, 04, 07 and 03 would be sorted into the order 01, 03, 04, 04 and 07. The sorting order is left to right and top to bottom.

Load/Save… This button leads to a new dialog in which you can load templates generated in previous spike sorting sessions and save your templates to resource files.

Selecting the area for a template

The two black triangles at the top of the large display area set the waveform region to use for template formation. To adjust this region, click and drag the triangles with the mouse. Vertical cursors appear as guides for the new template position. Old templates are deleted if the size of the template has changed.



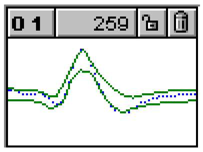
In many cases, the best information about the spike class is contained around the peaks and the start and end of the spike record holds mainly baseline noise. If you use the entire spike shape for template matching, you will be comparing baseline noise for some of the record, which is a waste of time and reduces the sensitivity of the template matching.

If you find that your spike only fills a small proportion of the spike trace you should reduce the number of points for the WaveMark channel with the button. This has several advantages: reduced disk space for storage, faster processing, quicker drawing of data and fewer records with a second spike overlapping the end.

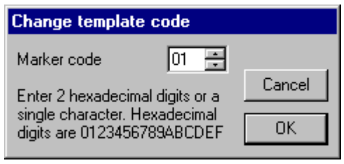
If you have too few points around the interesting region of your spike you will need to increase the sampling rate. Remember that the template routines interpolate between data points, so you only require enough points to define the spike shape.

You will get the best spike discrimination when the area selected shows the greatest variation between different classes of spikes. For example, if all your spike classes look very similar at the end, you will improve discrimination by excluding the end of the data.

The template area

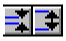


Templates appear at the bottom of the window. If this area is not visible, double click the window title bar or enlarge the window by dragging one of the edges. The templates can be small or large; double click a template to change the size. The template code is at the top left, with the number of spikes that matched the template in the centre. In large mode, the display also shows the number of spikes added to the template as: matched/added. When a template first appears, it has the lowest unused code.

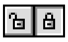


Double click the template code to open a dialog where you can change the code. Double-digit codes are read as hexadecimal marker codes, single characters are read as printing characters. If you have a spike that changes shape too much for one template to represent it, let it generate several templates and then edit the codes to be the same. You can use the code 00 to mark spikes or artefacts you are not interested in.

The display area holds the last spike that matched the template, the template itself and optionally all non-matching spikes. You control the drawing style of the matching and non-matching spikes and the template with buttons at the left-hand end of the toolbar. These buttons can appear at the top of each template area:

Change width 

These buttons are hidden in small template windows. They decrease and increase the width of the template.

Lock 

This button locks or unlocks a template. A locked template does not change when new spikes match it. An unlocked template adapts as each new spike is added. Templates can lock automatically if Auto Fix mode is set in the template settings dialog. When building templates, Spike2 cannot merge a provisional template with a locked template. Instead, Spike2 creates a new template with the same code as the one it would have merged with.

Clear 

This button erases the template. To erase all templates, click the bin icon in the toolbar.

Merging templates

You may decide that two templates are so similar that you would like to merge them into one. To do this, drag one of the templates over the one that you wish to merge it with and release the mouse button. The dragged template vanishes and the template over which you released it changes to take account of the new data.

The mouse pointer changes when it is over an area where you could drag or drop a spike shape or template. When over an area that you can pick up, the pointer is an open hand. If you hold down the mouse, the open hand changes to the closed hand, and you can drag the outline of the spike or template. When you drag over an area where you could drop, the closed hand has a plus sign on the back. You can compare templates by dragging, but not releasing the mouse. If you decide not to merge, make sure the mouse pointer is not the closed fist with the plus sign when you release the mouse button.

Manual template creation

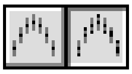
When paused, you can use drag and drop to create new templates and to classify spikes. Click the mouse in the data display area and drag the current spike to an existing template to set the marker code, or to an unused template to create a new template.

You can also use the keyboard to do the same thing: The keys `1` to `9` will match the current spike to the first template with the code 01 to 09, or if there is no template with a matching code, a new template will be created with that code. The `0` key can be used when editing spikes to give the current spike the code 00.

Toolbar controls

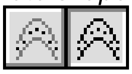
The toolbar buttons can be clicked with the mouse to control the template forming process. The leftmost group are three state buttons; click them to set the state, click them again to remove the state. The remaining buttons act immediately. The buttons are:

Overdraw



Normally, the templates display the last spike added. You can choose to show all spikes added to templates by overdrawing. This omits the step of erasing the previous spike before drawing a new one.

Template shape

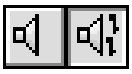


You can choose to display the mean template, or the upped and lower limits of the template. The template limits are useful when you display the last spike and need to see how well it fits within the template. The mean template can be useful when you are comparing template shapes.

Show non-matching spikes

In some circumstances you need to see every spike displayed over every template. With this button down, all non-matching spikes are drawn on top of the template in the "Not saving to disk" colour.

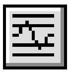
Sound on/off



In some circumstances it can be useful to have an audible indication of each spike as it is added to a template. If you turn this option on a tone is played through your sound card for each spike. A different tone is played for each template code; the tone is not related to the spike shape. A low pitched tone is played for a spike that matches no template, and a high-pitched tone for spikes that match templates numbered higher than 20.

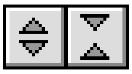
If you have no sound card the effect is operating system dependent (there may be no sound at all on some systems). With a sound card, there are limits on the maximum spike rate imposed by the operating system and the fact that sounds have to last much longer that the spikes for you to hear the tones! The best results are with Windows NT.

Automatic levels



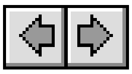
This option uses information collected from recent spikes to set the trigger and limit levels and to scale the data. This gives a good starting point for manual adjustment. The display scaling applies to both the data window and to the templates.

Scale data



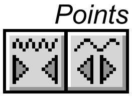
These two buttons increase and decrease the display size of the waveforms and the templates. If you click on a button, the scale changes once. If you hold one down, the scale changes repeatedly until you release the button. You can also scale the data by clicking and dragging the y axis.

Scroll data



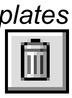
These buttons scroll the data in the large window sideways. This changes the pre-trigger time, and is equivalent to changing the Pre-Trigger field of the Channel Parameters dialog for the WaveMark channel. The two triangles marking the start and end of the region to use for templates are scrolled too, so the template region is unchanged. You can also click and drag the data x axis ticks to scroll the window.

Points



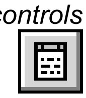
These buttons increase and decrease the number of points that will be saved for each WaveMark. This is the same as the Points field in the Channel Parameters dialog. You should minimise the number of points you save; the more points you save, the larger the data file. Click and drag the x axis numbers to scale the view around the 0 ms point.

Clear templates



This button clears all templates (and any tentative templates not yet confirmed). Think of throwing the templates in the bin. You can clear individual templates using the clear button on the top bar of each template.

Template controls

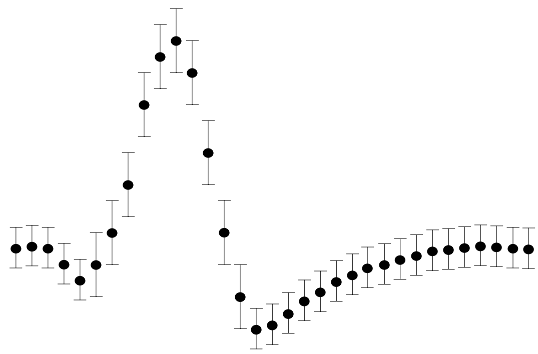


This button opens a dialog that gives you control over template generation. To get the best use of the items in the dialog you will need to understand how templates are constructed and generated, (see page 15-7, *Template formation* for more information).

Template formation

A template is stored as a series of data points. Each point of the template has a width and a minimum and maximum allowed width associated with it. The width represents the expected error in a signal that matches the template at that point. Spikes match a template if more than a certain percentage of the points in a spike fall within the template. It is easy to calculate an appropriate template width when you have a population of spikes of the same class. The problem comes when you have a single spike that starts a new template. What width do you give each point?

Our solution to this problem is to set the initial template width to a percentage of the maximum positive or negative amplitude of the spike. This value also sets the minimum allowed spike width. Experience shows that unless a minimum width is set, the template width tends to reduce due to random fluctuations in spike shapes. We also set a maximum width of 4 times the minimum width. The percentage to use depends on the data, 30% is a reasonable first guess.



The template width is twice the mean distance between the template and the spikes that created it, but is not allowed to become greater than the maximum width or less than the minimum. If the variation in spike shape around the mean were normally distributed, you would expect 80% of the data in a matching spike to lie within the template.

The template building algorithm works as follows:

1 The first spike forms the first template with the template width estimated. This is a *provisional* template. Provisional templates are not displayed.

2 Each new spike is compared against each template in turn to see if sufficient points fall within the template. If there are any *confirmed* templates, these are considered first, and if a match is found, the provisional templates are not considered.

3 If a spike could belong to more than 1 template, the spike is added to the template that is the minimum distance from the spike. Distance is defined as the sum of the differences between the spike and the template.

4 If a spike could only belong to one template, it is added to it.

5 If a spike belongs to no templates, a new provisional template is created.

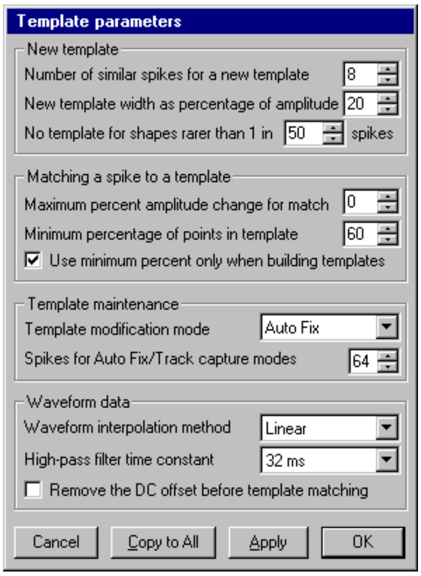
6 Adding a spike to a template changes the template shape and width. Once a pre-set number of spikes have been added to a template, it is checked against existing confirmed templates to prevent generation of two templates for the same data. If no match is found, the template is promoted to confirmed status. If a match is found, the provisional template is merged with the confirmed one unless the confirmed template is locked, in which case a new confirmed template is created *with the same code as the one it matched.* Each time a spike is added to a provisional template, the provisional template *decay count* is reset.

7 Each time a spike is added to a template, all provisional templates have their decay count reduced by 1. If any decay count reaches 0, that template has 1 spike removed from its spike count and the decay count is reset. If the number of spikes in a provisional template reaches 0, the provisional template is deleted.

Step 2 is complex. To compare a spike against a template we shift it up to 2 sample points in either direction to find the best alignment point. Shifts of a fraction of a point are done by interpolation. If you have enabled amplitude scaling, the spike is scaled to make the area under the spike equal to the area under the template (limited by the maximum scaling allowed). Amplitude scaling is only allowed for confirmed templates.

Template settings dialog

The dialog controls the formation of new spike templates and how spike match templates. The buttons at the bottom of the dialog are:



Copy to All

Use this button when you have multiple channels of WaveMark data and you wish to apply the current template parameters to all of the channels. The action is the same as Apply, but it sets the value for all channels, not just for the current channel.

Apply and OK

These do the same thing, but OK closes the dialog and Apply leaves it open. The settings you edit in the dialog make no difference to the template formation process unless you use one of the Copy to All, Apply or OK buttons. You cannot use these buttons if any field in the dialog holds invalid data.

Cancel

This button closes the dialog without applying any changes. Any changes made by previous use of the Apply or Copy to All buttons are preserved.

Each channel has an independent set of parameters. If you use the Channel control in the parent spike shape dialog to change data channel, the template parameters will also change to match the values for the new channel and any changes made will be lost unless you have used the Apply or Copy to All buttons to save the values.

The dialog has 4 areas:

New Template

This controls the creation of new templates.

Number of spikes for a new template

This sets the number of spikes at which a provisional template is promoted to a real one. We find that 8 is a reasonable starting value.

New template width as a percentage of amplitude

This is the percentage of the spike amplitude to give the initial (and minimum) template width. As spikes are added to the template, the width changes to represent the variation in amplitude of the spikes in the template. The maximum template width is set to 4 times the minimum width. A value of 30% is a reasonable starting value.

No template for shapes rarer than 1 in n spikes

If this is n, this is roughly the same as saying that you are not interested in spike classes which occur less often than once every n total spikes. If you want to keep all spikes as potential templates you should set this to a large number. We find that 50 is a reasonable starting value.

Matching a spike to a template

The items in this group are used when comparing a new spike with the existing templates to determine if the new spike is the same or should start a new provisional shape. As well as the conditions mentioned below there is always a limit to the error between a spike and a template (unless amplitude scaling is enabled) to prevent totally ridiculous results and to avoid wasting time interpolating spike shapes for data that could never fit a template.

Maximum percent amplitude change for match

Spike2 will scale spikes up or down by up to this percentage to make the area under the spike the same as the area under each target template. This is very useful if you have spikes that maintain shape but change amplitude. **Do not set this non-zero on-line unless you need to as it slows down the template matching process.** The maximum change permitted is 100%, which allows a spike to be doubled or halved in amplitude. Set this to 0 unless you need it. The value you set depends on the amplitude variability of your spikes; 25% is a reasonable starting value.

Minimum percentage of points in template

The percentage of a spike that must lie within a template for a match. If more than one template passes this test, spikes are matched to the template with the smallest error between the mean template and the (scaled) spike. 60% is a reasonable starting value. If you enable amplitude scaling you may want to increase this value to 70% or more.

Use minimum percentage only when building templates

The template width is most useful in the setup phase when we are looking carefully at differences between spikes. Once templates are established it can sometimes be better to match to the template with the smallest error and ignore the width. If you check this field, this sets the percentage of points that must lie in the template to 0% unless you are building templates.

Template maintenance

This group of fields determines how the shape of a template changes as more spikes are added to it. On-line, the template shape is fixed in all modes apart from Track. From our experience, Add All and Auto Fix are the most useful modes.

Template modification mode

Add All — All spikes that fit the template are added and modify the template. The effect of each spike becomes smaller as the number of matched spikes gets larger.

Auto Fix — The template is fixed once a set number of spikes have been added. If you have several similar spikes, using Auto Fix after a fairly small number of spikes can stop a template gradually changing shape and becoming the same as another template.

Track — The template shape tracks the spikes. The contribution of each spike to the template decays as more spikes are added. This is only really useful for slow changes in spike shape and always brings with it the danger than all your shapes will merge together. It also slows down on-line spike classification.

Spikes for Auto Fix/Track capture modes

This sets the number of spikes for the previous field in Auto Fix and Track modes. In Track mode, the smaller the number, the more rapidly the template shape changes.

Waveform data

The final group of fields control how the raw waveform data is processed into spikes:

Waveform interpolation method

Spike waveforms are shifted by fractions of a sampling interval to align them with templates using linear or parabolic interpolation. Parabolic should be slightly better, but is slower. The on-line 1401 code always uses linear interpolation for speed reasons.

High-pass filter time constant

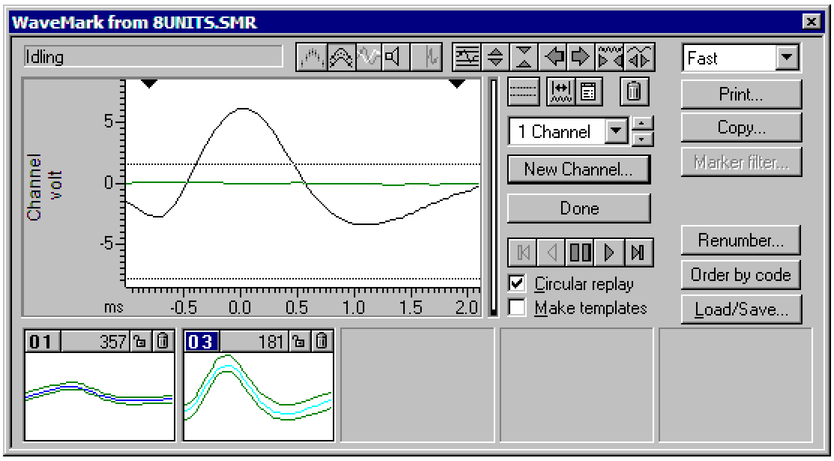
Use this to remove baseline drift. Set this to a few times the width of the spikes. Do not set the value too low or it will significantly change the spike shapes. If your signal has abrupt baseline changes, you may get better results with the DC offset option.

Remove the DC offset before template matching

Check this field to subtract the mean level from each spike before matching. This effects template formation and matching, not the saved data. Unless your baseline has sudden DC shifts, it is usually better to use the high-pass filter to follow signals that drift. There is a small time penalty for using DC offset removal on-line.

Off-line template formation

The Analysis menu New WaveMark... command extracts WaveMark data from waveform channels (and also from existing WaveMark data). If you wish to edit WaveMark data, see the *Off-line template editing* section.



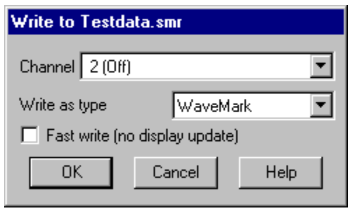
This window is almost identical to the on-line template setup window. There are a few more buttons in the control bar, some button labels have changed and there are new controls to set the spike replay rate and buttons to print and copy template data.

The program scans the data in the channel set by the Channel control (wrapping round to the start of the data when it reaches the end), and displays triggered and non-triggered events, as if the data were being sampled.

Replay rate

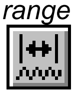
This control sets the maximum number of spikes per second to process when running continuously. With Fast selected, the spikes are processed as fast as possible; you can also set this by pressing the `F` key (`F` for Fast). With Real time selected, spikes are replayed no faster than real time; you can set this with `R` for `R`eal time. You can also select slower rates in Hz; the `S` for `S`low key selects a medium-slow rate.

New Channel...



Once you are satisfied with your templates, click the New Channel... button. A dialog opens in which you can select the channel to write to and the new data type. The Fast write checkbox suppresses display updates whilst producing the new channel, to save time. When you choose OK, the selected time range of data is analysed, and any events that cross the trigger levels are written to the new channel in the selected format. Options that change the number of points saved per WaveMark are disabled during analysis.

Time range



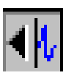
This button opens a dialog in which you set the time range of data to process. You can either type in the start and end times, or select values from the drop down list. When you open the window for the first time, the time range is set to the whole file.

Cursor 0

While this window is open, cursor 0 is added to the time view associated with the data file. This cursor marks the position of the current spike. When paused, you can drag this cursor to a new position. The special cursor does not appear in Cursor windows. The `Ctrl+A` key combination moves Cursor 0 to the start of the time range.

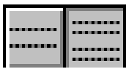
Circular replay

This check box enables circular continuous replay. If you clear this box, continuous replay stops when it reaches the end of the time range set for analysis.



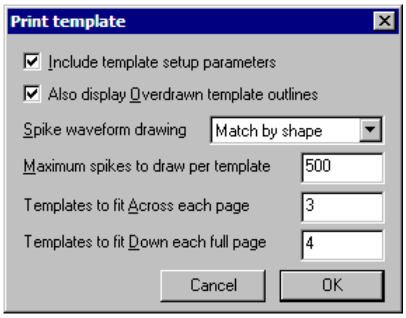
If you depress this button, the associated time window scrolls to keep the current spike centred, if this is possible.

Marker filter The Marker filter button is enabled when the source channel holds WaveMark data. Click the button to open the marker filter dialog, or bring it to the front if it is already open. Whenever you change channel to a channel holding WaveMark data with the Marker filter dialog open, the Marker filter dialog automatically changes channel to match.



Depress this button to display and use the limit cursors. Spikes with peak amplitudes at the trigger point that lie outside the limits are ignored.

Print… The Print button opens the Print template dialog. The output format depends on the printer page size (you can set the print margins in the File menu Page setup dialog). You can choose how many templates to draw across and down each page.

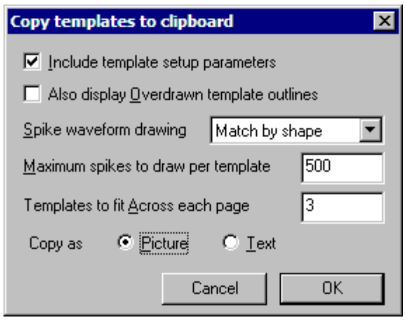


You can include the template setup parameters, and also an extra picture showing all of the template outlines overdrawn for easy comparison.

The Spike waveform drawing field chooses how spikes are included in the result. Set it to None for no spikes or to the method for selecting which spikes are drawn with a template. You can choose Match by shape or Match by code (only in the Edit WaveMark dialog). Spikes are drawn in the best-fit position to the template. You can limit the number of spikes to draw in each template; some printers cannot cope with huge numbers of lines in the printed output.

If you have set a time range to analyse, the overdraw spikes are taken from the time range. If you have not set a time range, the spikes are taken from the start of the file.

Copy… The Copy button opens the Copy templates to clipboard dialog, which is very similar to the Print templates dialog. There are radio buttons to choose between Picture and Text output.

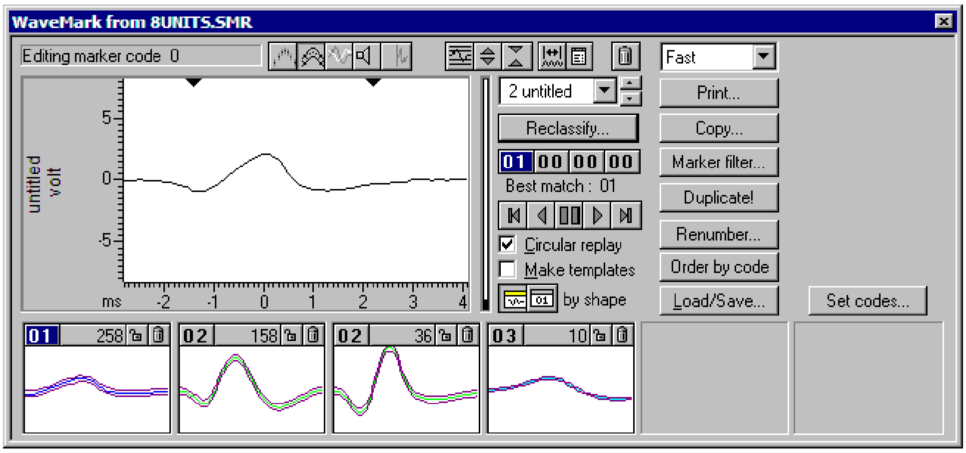


Picture output is in Enhanced Metafile format and can be pasted into documents in most drawing programs and many word processors. Many bitmap-editing programs will accept this format and convert it into a bitmap.

Text output optionally includes the setup parameters. The templates are output in vertical columns separated by Tab characters. There is one column or two columns per template, depending on the template shape display mode (mean templates or upper and lower limits). The first row of template output holds column titles, in quotation marks. The output is designed to paste easily into spreadsheet programs.

Off-line template editing

The Analysis menu Edit WaveMark... command opens a dialog in which you reclassify data in a WaveMark channel. If the selected channel has a marker filter set (see the *Analysis menu* chapter), you view and edit only those events that match the filter specification. The window controls are the same as for *Off-line template formation,* except that you cannot change the number of points in the main window and there are no horizontal cursors (use New WaveMark if you need trigger levels).



Marker codes
00 00 00 00

These four codes show the four marker codes of the current spike. Normally only the first code is used for spike classification and the others are 00. If you click on a code, it becomes highlighted and sets the code that is changed by reclassification. If you double-click a code a dialog box opens in which you can edit the code. The Best match code is the code of the template that best matches the current spike or 00 if no template matches. If you use this dialog on-line only the leftmost button is enabled.

Template formation method
by shape

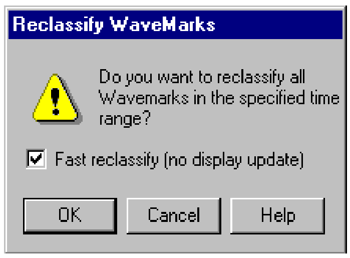
You can build templates by shape or by code. If you build by shape, templates are formed and spikes are sorted based on shapes; prior classification is ignored. If you build by code, the shapes are ignored, and the classification is purely by the code highlighted in the Marker codes. Templates are built automatically when the play controls are set to play forward or backward through the data and when the Make templates box is checked.

Manual reclassification

When replay is stopped, you can reclassify a spike by dragging the raw data and dropping it on a template or by double clicking a Marker code and editing. You can also reclassify by pressing keys 0 to 9 to give a spike template marker codes 00 though 09. If you use keys 1 through 9 and no template exists with the code, a new template with the code is created based on the current spike.

Automatic reclassification
Reclassify...

This option reclassifies all the events (or filtered events if a marker filter is set) on the channel in the time range by matching them to the templates. You can set fast reclassification (no display) or a slower mode where each event is drawn in the data display area as it is reclassified. You can stop the reclassification before all spikes have been scanned, but you cannot undo it.



Tip: remove any check mark from the Make templates box before reclassifying to stop Spike creating extra templates and modifying existing ones.

Marker filter

The Marker filter button is always enabled in this dialog. Click the button to open the marker filter dialog, or bring it to the front if it is already open. Whenever you change channel to a channel holding WaveMark data with the Marker filter dialog open, the Marker filter dialog automatically changes channel to match.

Duplicate! This button gives you a very fast and easy way to generate separate data channels in the time view linked to the dialog, one channel for each template code. It is disabled if the current channel is a duplicate channel. It does the following:

1. All duplicates of the current channel are deleted (with no warning). If you have duplicated the time view, the duplicate channels vanish from all the views. Any processing that depended on the duplicated channels is cancelled.
2. Spike2 counts how many different template codes you have defined. For example, if you have created four templates with codes 01, 02, 02 and 03, this is counted as 3 different codes.
3. For each template code, Spike2 generates a duplicate channel with the marker filter set to display only spikes that match the template code. The marker filter mask used is the same as the code highlighted in the Marker codes (usually the first mask is used). The channel title of each duplicate is set to `"Code nn"`, where `nn` is the template code. The new channel is positioned next to the original channel in the time window. It is placed above the original channel unless the Edit menu preferences have reversed the channel order, in which case it is placed below the original.

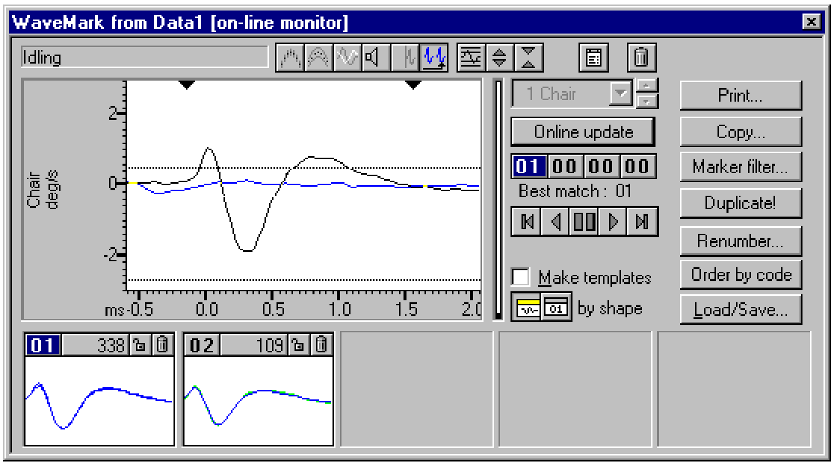
The assumption is that you have already, or are about to, classify the spikes in the channel based on the templates in the dialog. If you have duplicated the time view, the new duplicate channels exist in all views, but are only displayed in the view that is linked to the Edit WaveMark dialog. The original channel will remain, unchanged, in all the views.

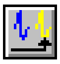
Set Codes This button is a short cut to the Analysis menu Set Marker Codes dialog. You can use this to give all the displayed spikes in a channel the same marker code. For example, if you want to change all the spikes on channel 2 with codes 04, 05 and 09 to have code 03:

1. Open the Marker Filter dialog and set the filter for channel 2 to display only codes 04, 05 and 09.
2. Click the Set Codes button and set the first code to 03 and click the Set button.

On-line template monitoring

The Analysis menu Edit WaveMark... command can be used during data sampling if there are any WaveMark channels. The window that opens is very similar to the off-line template editing window; the time range and reclassify buttons are hidden, only the first of the Marker code buttons is enabled, the Reclassify… button is renamed to Online update and there is a new button in the control bar.





When this button is down, all spikes are taken from the end of the file, the horizontal cursors that set the trigger levels appear and controls that cannot be used are hidden. In this state, you monitor the spikes as they are classified by the 1401. If there are no spikes, the data display shows recent waveform data so that you can adjust the trigger levels.

With the button up, the window behaves very similarly to the normal off-line template editing except that the Reclassify… button is not available.

Once sampling finishes the new button vanishes and the Reclassify… button reappears and the normal off-line editing behaviour is restored.

Online update

During sampling, the CED 1401 interface classifies spikes based on the templates and parameters that were last loaded to it. If you generate more spike templates, change the existing templates, or edit the template parameters this will make no difference to the 1401 unless you click the Online update button. This deletes all templates stored in the 1401 for the current channel and replaces them with the templates displayed in the window. The trigger levels are updated dynamically; you do not need to click the Online update button to change the trigger levels.

On-line and Off-line templates

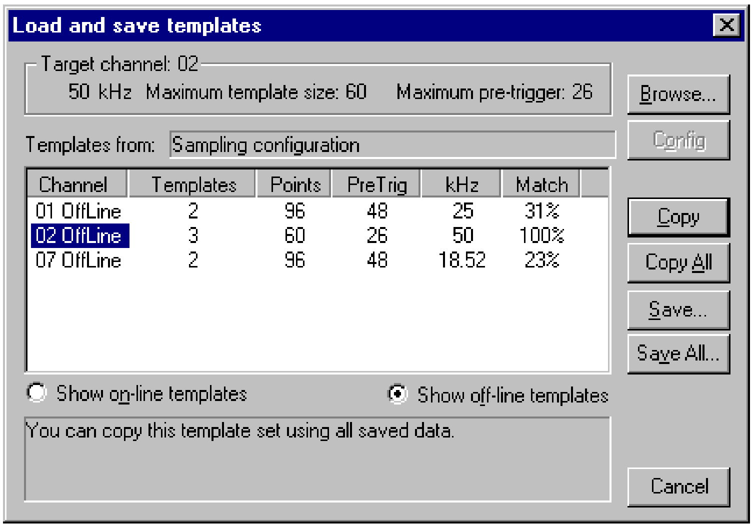
We save two types of template in the sampling configuration and in the `S2R` resource files associated with a data file: *on-line* templates and *off-line* templates. Spike2 saves on-line templates when you use the On-line template setup dialog and when you click the Online update button while monitoring spikes during sampling. Off-line templates are saved when you change channel or close the spike shape dialog.

When you change channel, Spike2 loads templates from the saved on-line templates if you are sampling and from the saved off-line templates if you are not sampling. This is so that the templates you see during sampling are as close as possible to the templates that the 1401 uses. Off-line templates are loaded from the resource file associated with the data file. If none are found or there is no resource file, off-line templates are loaded from the sampling configuration.

In summary, on-line templates are the last set of templates copied to the 1401 during sampling. Off-line templates are the last set of templates displayed in a template dialog.

Load/Save templates

You can save templates and load templates into the current channel or into all matching channels with the Load/Save… button. Spike2 saves templates in the sampling configuration and in the resource files associated with data files. Whenever you work on templates in a data file, for example `data1.smr`, both the associated resource file `data1.s2r` and the sampling configuration are updated to hold the latest set of templates. You can save the sampling configuration to disk in the File menu.



The box at the top of the dialog shows the sample rate and template size limits of the current target channel. This is the channel that is updated by the Copy command.

The box in the centre of the dialog lists sets of templates from the Sampling Configuration or loaded from a resource or configuration file. When you open the dialog, Spike2 shows you templates in the Sampling Configuration. The columns are:

Channel Identifies the channel from which the templates originated.

Templates The number of templates stored for this channel. Channels with no templates are not listed. All templates for a channel have the same sampling rate, points and number of pre-trigger points.

Points The number of data points in each template.

PreTrig The number of data points in each template that lie before the peak or trough used as a trigger. A negative value means that the template starts this many points after the trigger point.

kHz The sample rate of the channel from which the templates originated. This need not match the target channel; Spike2 uses cubic spline interpolation to compensate for sample rate differences. Templates generated before Spike2 version 3.15 did not include a sampling rate and 25 kHz is assumed.

Match This columns displays how much of each template shape would be used if this set of templates were copied to the target channel.

Browse… and Config

Use the Browse… button to list templates in resource (`.s2r`) or sampling configuration (`.s2c`) files. The Config button displays templates stored in the sampling configuration.

Show on/off-line templates

Templates copied to the 1401 during sampling (on-line templates) are saved separately from the templates displayed in the template dialogs (off-line templates); you choose which to list with Show on/off-line templates. The Copy All and Save All commands operate on the list selected by this option.

Copy

The Copy button copies the selected set of templates into the target channel and closes the dialog. You can copy a channel as long as Match is at least 10%. This overwrites any templates that already exist for the target channel.

Copy All For each channel in the channel control, Spike2 searches the list of templates for a matching channel number. If templates exist for the channel and they match the channel settings, they are copied from the list and become available for use.

Save This saves the selected set of templates in the list to a resource file (`.s2r` file extension). You are prompted to supply the name of a new file or to select an existing resource file. If the resource file already exists, any templates for the channel are replaced. If the resource file does not exist, the templates are written to a new file.

Save All This saves all the templates in the list to a resource file (`.s2r` file extension). You are prompted to supply the name of a new file or to select an existing resource file. If the resource file already exists, any existing templates for the channels in the list are replaced. If the resource file does not exist, the templates are written to a new file.

Cancel This button closes the dialog.

Template storage

You use templates to classify WaveMark data, but they are not stored in data files. Instead, Spike2 stores templates in the following places:

- In the sampling configuration. You can always load the last set of templates you used on-line or displayed off-line in the current session from here. You must save the configuration to make templates persist after you close Spike2.
- In saved configuration files with the file extension `.s2c`. For example `Last.s2c` automatically saves the last configuration used for sampling. However, changes made in the template dialogs after sampling ends are not saved automatically.
- In resource files associated with data files. Each time you use a template dialog, any templates you create or change are saved in a resource file with the same name as the data file, but with the file extension `.s2r`. During sampling, there is no associated resource file. In this case, when you save the data file, Spike2 creates a resource file and copies the templates in the sampling configuration to it.
- In resource files created from the Load and Save templates dialog.

When you open a new data file for sampling, the template dialogs load up the last set of on-line templates from the sampling configuration. These on-line templates are updated each time you click the Online update button in the Edit WaveMark dialog. The off-line templates are updated to match the templates displayed in the template dialog. When you stop sampling and save the file, the templates are written to both the resource file associated with the data file and to the sampling configuration.

When you work on a data file from disk and open a template dialog, the off-line templates are loaded from the resource file associated with the data file. If none are found, off-line templates are loaded from the sampling configuration. Any changes you make are saved to both the resource file and to the sampling configuration as off-line templates.

By saving templates to both the resource file and to the sampling configuration, Spike2 makes it easy for you to work on a sequence of data files. For example, you might break a sampling session into several data files. On-line templates created for the first data file are automatically used for the second and subsequent files.

Getting started with spike shapes and templates

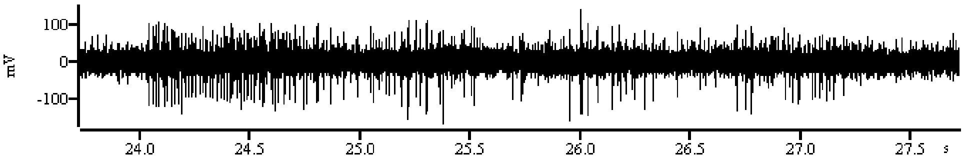
Sorting spikes is as much an art as a science; there is no substitute for the skill and experience of the investigator. Spike2 provides you with a toolbox of routines that will help you to discriminate various shapes, but there is no guarantee that two spikes are from the same source just because they have the same shape (or because any other measured parameter or set of parameters are the same). That said, given populations of spikes, observations of the relationships between them and experience of the system under study, useful results can be obtained.

Quality of the original signal

Probably the most important contributor to successful spike sorting is the quality of the original data. If you can improve the signal to noise ratio, or adjust the electrode position to increase the amplitude of your target spikes relative to the background, the time spent will be rewarded by time saved when classifying spikes.

The waveform sample rate used is also very important. For most spikes lasting 1 to 2 ms, a sample rate of 20 to 25 kHz is about right (some workers like higher rates than this). It is important that the data is frequency band limited to half the sample rate before it is sampled. For example, when sampling at 25 kHz there must be no frequency components above 12.5 kHz. If such components are present, they are aliased to lower frequencies and appear as noise that is very difficult to remove.

To check the quality of your signal it is a good idea to sample a section of data as a waveform first. Here is an example waveform sampled at 25 kHz from a tape recording.

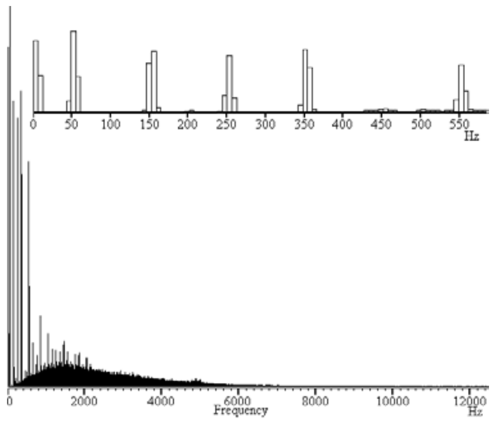


Sampling configuration for this data

Open the sampling configuration Channels tab and click Reset. Click New Channel and select a Waveform channel. Set Channel to 1 and 1401 Port to 0. Set the Ideal adc sampling rate to `25000`, and set Units `mV` = Input in Volts x `1000` + `0` and click OK. Now click the Resolution tab and set Optimise to Full. Finally click the Mode tab and select Continuous. You can now sample.

The example waveform amplitude is around ±100 mV, which is a bit small. The 1401 family usually expect a full-scale input range of ±5 Volts, and our spikes are using only 1/50th of the available range. The size of each digitisation step is about 2.5 mV for the micro1401 and 1401*plus* and 0.15 mV for the Power1401 or Micro1401 mk II. Although the system will work fine with signals of this amplitude, if your rig has more gain available, it is much better to make the signals around ±1000 mV in amplitude. This gives you plenty of headroom for large spikes and a good signal resolution.

It is worth checking the waveform power spectrum for unexpected frequencies (use the Analysis menu: New Result View: Power Spectrum command with a block size of 4096). Here we found narrow peaks at low frequencies and almost nothing above 6 kHz. Apart from a peak at 0 Hz caused by a small DC offset, the remaining peaks are all at odd multiples of 50 Hz. This data was recorded in England where the mains frequency is 50 Hz. Anything that reduces mains pickup (better shielding, removal of earth loops, cleaning up of connections) will make spike sorting easier.



Sharp power spectrum peaks often indicate signal contamination. If the peaks change frequency when you change the sample rate, they are caused by aliasing of frequencies above half the sampling rate and they may be removable with an external low-pass filter.

Setting up the WaveMark channel

Assuming that we have adjusted the rig to get the best signal to noise ratio and spike amplitude, we must now set up a suitable sampling configuration to capture our spikes. We could sample the data as a waveform, but unless you really need all the data in between your spikes, this approach is wasteful of disk space. For example, a single waveform channel sampled at 25 kHz consumes 50 kB of disk space per second, or 3 MB per minute, or 180 MB per hour.

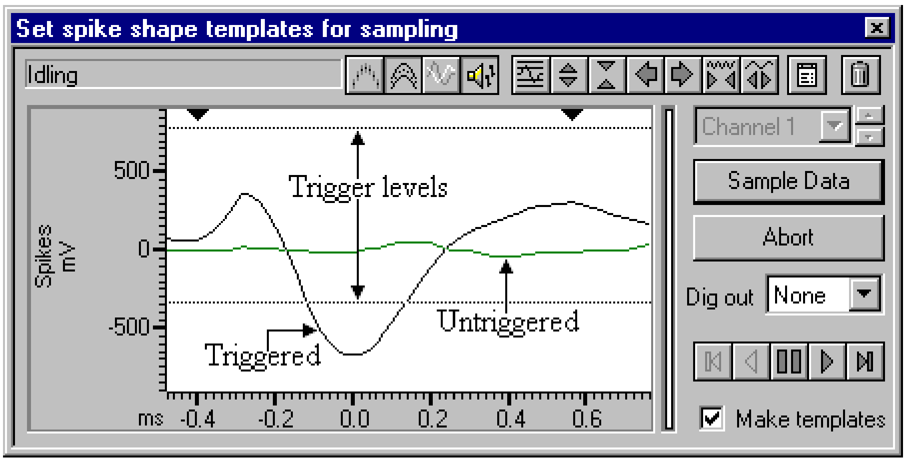
An alternative is to sample WaveMark data. In this case, each time the input signal crosses a positive or negative threshold we search for a peak or trough in the data and save the time and a small fragment of waveform. If we save 32 data points per spike, each spike costs us 72 bytes of data. As long as you set the threshold levels so that you get less than 700 spikes per second, this method uses less storage space. At a mean spike rate of 20 spikes per second, you would consume 86 kB per minute, or 5 MB per hour.

There is one further advantage to WaveMark data. As the data is captured, Spike2 can match it to templates and code each spike accordingly. This gives you further on-line analysis capability as you can look at the responses of individual spike types on-line. The on-line sorting is not the final word as you can revue and resort the data off-line.

Suggested configuration for WaveMark data

Open the sampling configuration Channels tab and click the Reset button. Then click New Channel and select a WaveMark channel. Channel should be `1` and 1401 Port should be `0`. Set the Maximum sustained event rate to `30`, and set Units `mV` = Input in Volts x `1000` + `0`. Set Points to `32` and Pre-trigger to `10` and WaveMark sample rate to `25000` and click OK. Now click the Resolution tab and set Optimise to Full. Finally click the Mode table and select Continuous. You can now sample.

Spike2 knows that you have a WaveMark channel in the sampling configuration, so when you click the Sample now! Button in the toolbar, it opens up a window for you to set trigger levels and adjust the number of sample points and to build templates.



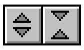
The picture shows the window after adjusting it to have the minimum size and setting up the trigger levels. When you open a window for the first time you are likely to have a more or less flat line across the centre of the display area and the trigger levels will not be in useful positions.

There are a lot of buttons and controls in this window. If you move the mouse pointer over one and leave it for a second or so, a line of text will "pop-up" with a short explanation. Alternatively, you can use the F1 key to activate the Help system.

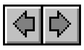
This button is one of the most useful when you are setting up for spike shape capture. Each time you click the button, Spike2 will make a guess at reasonable trigger levels (and limit levels for the New WaveMark command with limits enabled) and adjust the display gain so that you can see the data. Once a trigger level is crossed a second trigger cannot occur until the waveform falls below half the trigger level.

It is likely that you will only wish to trigger in one direction, so move the lines that you do not want to use to the edge of the window. To change a level, move the mouse pointer over the level, hold down the mouse button and drag to the new position.

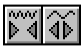
It is possible to use both trigger levels, but we strongly suggest that you do not to avoid the sideways shift you will get if a spike triggers on the opposite side from the one you intended. If your spikes are bi-phasic, that is they have a rising peak and a falling peak, it is best to trigger in the direction that has the clearer peak. If your spikes are tri-phasic, as in the example above, it is best to trigger in the direction that has only one peak to avoid problems caused by false triggers on the second peak. If both directions are equally clear, choose the direction with the narrower peak (usually the first one).



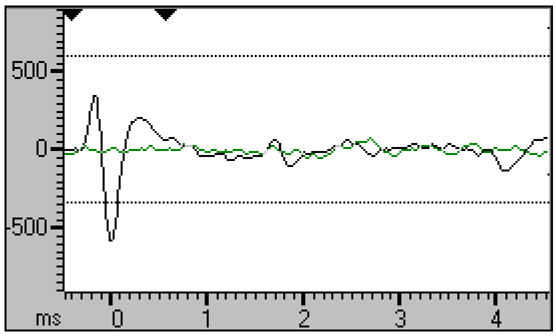
Once you have got more or less the desired display gain you can use these two buttons to change the display scaling manually. You may also want to reduce the display gain so you can drag one of the trigger levels off the screen to reduce visual clutter. As an alternative to using this button, move the mouse pointer over the y axis and click and drag to scale the data.



These buttons change the number of pre-trigger points. The trigger point is the first signal peak or trough after it crosses the trigger level. This point is labelled 0 on the horizontal axis. You cannot change the number of pre-trigger points after you start to sample. You can also scroll the data by clicking and dragging the tick marks of the x axis.

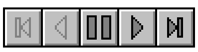


These buttons increase and decrease the data points that are displayed and written to disk for each spike. The number of pre-trigger points is preserved if this is possible, so you should set the pre-trigger points first. You can scale the x axis around the zero point by clicking and dragging the x axis numbers.



You need to capture sufficient points to enable accurate spike identification. You also want to capture as few points as possible to minimise the data storage. In this example we have captured about 5 ms of data, which is at least 3 ms more than we need. As well as leading to huge data files, sampling too many points also increases the chance of the data for one spike containing data for a second or even a third spike. Further, Spike2 will not trigger for the next spike until it has finished sampling the previous one. You cannot change the number of points once you start sampling.

The two black triangles mark the start and end of the region of the spike that is passed to the template matching system. Click on a triangle and drag to change the region. When you drag the markers, vertical cursors drop down to identify the selected area.



This is the play control. The buttons from left to right are: step backwards, run backwards, pause, run forwards and step forwards. Step and run backwards are disabled during setup. You can also use keyboard short cuts for these buttons: `b` or `B` steps backwards, `N` runs backwards, `v` or `V` pauses, `n` runs forwards and `m` or `M` steps forward.

Forming templates

By now you should be able to set up the system to capture spike shapes. The next step is to form templates from these spikes. The first step is to check that the template parameters are set to sensible values.

Click this button to open the template parameters. To get started, set these values:

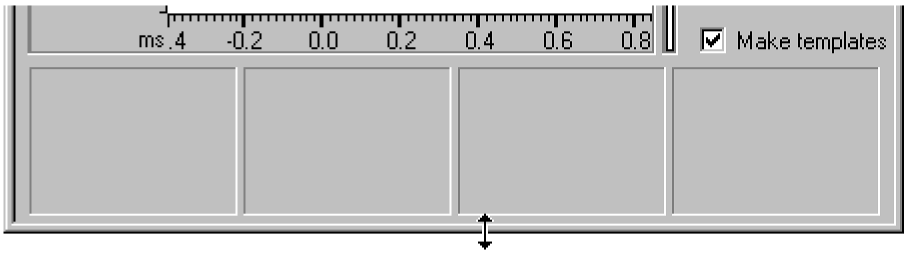
Suggested template parameters

In the New template section, set Number of similar spikes for a new template to `8`, New template width as a percentage of amplitude to `30`, No template for spikes rarer than 1 in to `50`. In the Matching a spike to a template section, set Maximum percent amplitude change for a match to `0`, Minimum percentage of points in template to `60` and check the box for Use minimum percent only when building templates. In the Template maintenance section, set Template modification mode to Add All. In the Waveform data section, set the Waveform interpolation method to Linear, High-pass filter time constant to 20 ms and leave the Remove DC offset before template match as unchecked. Finally click OK.

As you get more familiar with template matching you will probably change these values. However, this set is a useful starting position. We have turned off the amplitude scaling as this causes a large increase in on-line computation and so should not be used unless you really need it. Amplitude scaling comes into its own when your spikes change amplitude while preserving shape, for example during a burst.

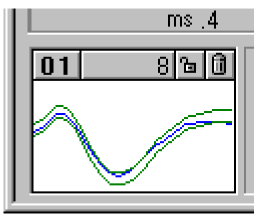
Spike templates are displayed below the data display area. If you cannot see the template area drag the bottom edge of the window downwards with the mouse. The template area may already contain templates. If it does, you can clear them all by clicking the bin button at the top right of the window. You can remove individual templates by clicking the small bin button in the template window.

Expanding the window to show the template area

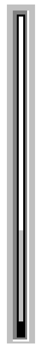


The template windows have two sizes: small (in the picture) and large. The large size is slightly more than twice the width and height of the small image. You change from small to large by double clicking in the template area. Spike2 remembers the last window size you used for both small and large template displays. You can also expand the window sideways by dragging the right-hand edge of the window, or you can grab one of the corners to expand in both directions. As a short-cut, you can maximise and minimise the size of the window by double-clicking in the title bar.

To build templates, make sure that the Make templates box is checked, the play control is set to run forwards and that spikes are crossing the trigger level you set in the data display area. If all is well, when 8 similar spikes have been detected (8 is the number you set in the template parameters) a new template should appear in the template area.



The four rectangles at the top of the template hold, from left to right, the code associated with the template (**01**), the number of spikes in the template (8), a button that indicates the locked state of the template and a button that can be used to delete the template. The vertical size of the displayed template is set by the vertical scale of the data display area.



As spikes are processed, the vertical bar on the right of the data display area indicates the number of templates that are being considered in grey and the number of confirmed templates in black. The first 20 confirmed templates appear in the template area.

The horizontal size of the template is set by the two black triangles in the data display area. For good spike sorting, it is important to select the region of the spike that shows the most variation between different classes of spikes. Do not include baseline areas before or after the spike. Spikes are matched to templates based on the errors between the mean template and each spike. Spikes are excluded from templates based on individual points falling outside the template boundaries.

You can control the appearance of the templates with these three latching buttons. If you click them once they become active, click them again and they become inactive. The leftmost button controls overdrawing of spikes in the template. With the button down, all spikes that match the template are over-drawn in the template. With the button up, only the last matching spike is drawn.

The centre button chooses between a display of the template boundary with the button down, and a display of the mean template with the button up. With the right-hand button up, spikes are displayed only in the template that they match. With the button down, spikes are also displayed (in grey) in all non-matching templates.

Spike code and template colour

Each template has a code in the range 1 to 255. When the templates are used for spike sorting, the code is used to label matching spikes. Code 0 is not used by templates, but spikes can be given a zero code, meaning that they are not coded.

The code is represented by two hexadecimal digits; 1 is represented by 01, 2 by 02 and so on up to 9 which is 09. In hexadecimal (base 16 numbers), the numbers ten to fifteen are represented by the letters A to F, so ten decimal is 0A, eleven is 0B through to fifteen which is 0F. Sixteen decimal is 10 in hexadecimal, seventeen is 11 and so on.

There is a further complication with the codes. We also allow the use of single character codes in place of the hexadecimal codes 20 to 7E. This came about because the keyboard marker channel uses the same coding system and the ASCII codes for the Roman printing characters lie in the range hexadecimal 20 to 7E. Although Spike2 can manage up to a hundred templates internally, you can display only the first twenty, so codes above 14 (decimal 20) are rarely used and single character codes are rarely a problem.

The code given to a template also determines the template drawing colour. In the View menu Change Colours… dialog you can assign colours for template codes 01 to 08. It is a good idea to set colours that contrast strongly with the background colour you have set for the time view as this colour also sets the background colour for the template windows. If you use higher numbered templates the colours repeat, so the colour for codes 09 to 10, 11 to 18, 19 to 1F and so on are the same as for 01 to 08.

New templates take the lowest available code, but you can change the template code by double clicking on it. This opens a dialog box in which you can edit the code. There is no restriction on the code you give, other than it must be in the range 01 to FF. In particular, you can have more than one template with the same code.

If you enlarge the window on the right, you will discover two more buttons. Renumber changes template codes to remove gaps caused by deleting or editing codes. Order by Code sorts the templates into ascending code order.

Automatic template creation

With the play control set to run and the Make templates box checked, Spike2 creates templates automatically by looking for similar spikes and following the rules set in the template parameters dialog. Each new spike is compared first with the existing confirmed templates (displayed in the template area). If it doesn't match any of them it is compared with the provisional templates. If it matches nothing, it forms a new provisional template.

When a spike matches a 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.

When a provisional template gets 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 is ready to become a confirmed template. If it matches a confirmed templates it is merged with the confirmed template. Otherwise, a new confirmed template is created.

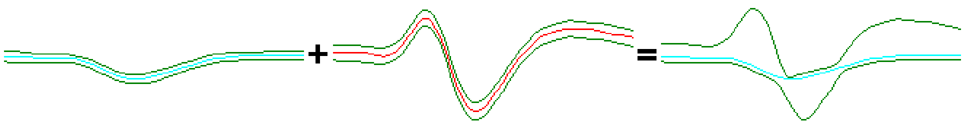
To avoid the system being swamped with provisional templates, there is a mechanism that makes them decay away. This is set by the No template for spikes rarer than 1 in field of the template parameters dialog. If this number of spikes are tested and a provisional template does not get one new spike added to it, the template spike count is reduced by one. If the spike count becomes 0, the provisional template is deleted.

Manual template creation and merging

Instead of letting the system create templates, you can build them yourself. Use the play control to step forwards one spike at a time until you find one that you want to turn into a template. Click the mouse in the middle of the display area and drag the spike to an unused window in the template area and release. If you drag a spike shape to an existing template, the spike is added to it.

If you decide that two templates are very similar, you can combine them by dragging one of the templates and dropping it on another. Spike2 will align the dragged template with the target and then merge the two shapes by combining the template limit.

Merging templates



This example shows the effect of merging two very dissimilar templates. You can see that the extremes of the template boundaries of the original templates are preserved in the result. Normally, you would merge two similar templates. You can test for template similarity by dragging a template over another without releasing.

Number of templates

You can create (and Spike2 will remember) up to 20 templates per channel. However, if you are preparing for on-line sampling, only the first 8 templates in the template area will be passed to the 1401 for use on-line. You can, of course, resort the data off-line using all 20 templates, and it is possible to sort spikes into many more classes than this by using the Analysis menu Marker Filter and Edit WaveMark commands.

Locking templates

Each time a spike matches a template, and the Make templates box is checked, the spike is added into the template unless the template is locked. The lock button in each template can be used to lock and unlock templates manually. You can also make the templates lock automatically after a set number of spikes have been accumulated by setting the Template modification mode in the template parameters dialog to AutoFix.

If you don't lock the templates, they change as more spikes are added. This means that the shape you end up with could be quite different from that with which you started. For example, if you create a template manually from a specific spike, you may want it to keep the exact shape so you would lock it.

Amplitude variation

If you find that the system keeps generating multiple templates for the same spike because the spike amplitude changes, there are several things to try. The obvious solution is to set the Maximum percent amplitude change for a match to a non-zero value. For example, if you set this to 30%, each spike can be increased or decreased in size by up to 30% before attempting to match it to the templates. Spike2 does this by computing the area between the spike and the zero baseline and changing the spike amplitude so that this area is the same as the area between the template and the baseline.

Beware that this is computationally expensive, which does not matter for off-line sorting, but it can be important on-line, especially if you don't have a Power1401. The more channels of spikes and the more templates you are matching against, the larger the time penalties you will impose. If you find that turning this option on leads to spikes not being classified when the spike rate is high, then you may need to consider other options.

Another method to cope with amplitude variation is to make the templates wider. You can do this by merging templates, or by increasing the New template width as a percentage of amplitude in the template parameters. However, wider templates leads to less discrimination between spike shapes.

Closely allied to making the templates wider is to reduce the Minimum percentage of points in template field. However, this also leads to less discrimination between spike shapes.

You might also consider checking the Use minimum percent only when building templates box. If you do this, when you run on-line and when reclassifying spikes, spikes are matched to the template that has the smallest error between the shape of the spike and the shape of the template, the template limits are ignored. This option is most useful when you know that a whole bunch of spikes are either type A or B but the data is very noisy, so you chose a good spike of each class to make two templates, then you check the box and sort on the basis of minimum error.

If you do not need to separate many shapes on-line, you can simply accept that it takes 2 or even 3 templates to cover the full range of amplitudes and give the 2 or 3 templates the same code (double click the code and edit it).

Using sound 

If your computer has a sound card, you can play a sound for each spike. Spikes that match a template play a sound with a pitch that rises as the code for the template increases. Spikes that don't match will play a lower pitch. How successful this is depends on the capabilities of your system! It works well on my computer when running under Window NT, but on the same computer running Windows 95 or 98 the result is not so good. Try it and see. Ideally, you should be able to distinguish different patterns for different spikes.

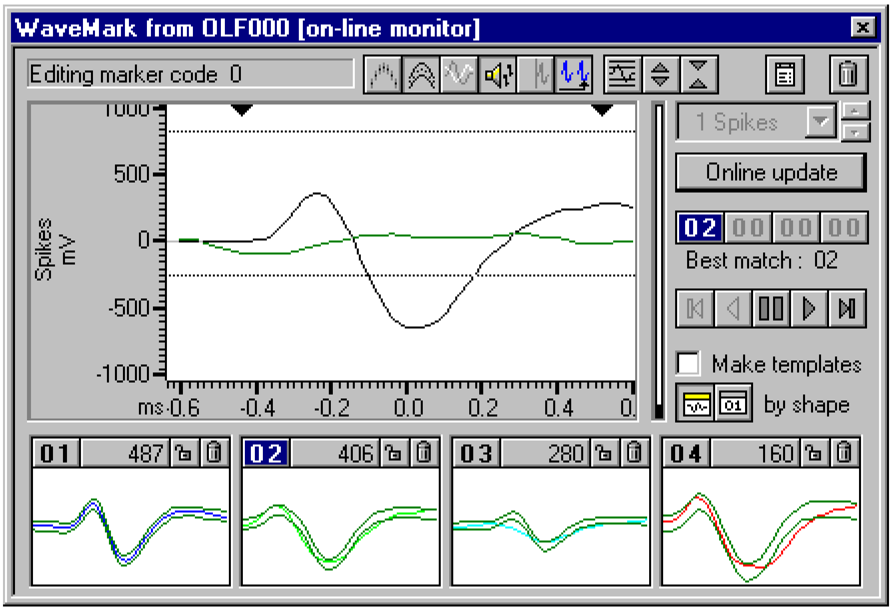
Sampling data

Although we have not discussed the details of every possible control you could have used, you should now have enough information to be able to set up your system with some templates and sample data. Click the Sample Data button and the dialog box will close and a new data file window opens. If you followed our sampling configuration settings, the window will contain a channel of WaveMark data and the Keyboard marker channel.

When you close the template setup dialog, you fix the number of points that will be captured for each spike, and the number of pre-trigger points.

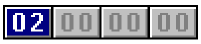
Monitoring spikes

You can adjust the trigger levels and even change the templates during sampling with the Analysis menu Edit WaveMark command. If you use this a window that is very similar to the template setup window opens:



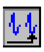
The displays area shows the last spike that crossed the trigger level for the channel, and when there are no spikes, it shows the latest data. You can adjust the trigger levels by dragging them, and *the level change is passed to the 1401 immediately*. If you make any other change, this has no effect on the 1401 unless you click Online update.

During sampling, the 1401 holds a set of templates that it uses to classify spikes. There is a separate set of templates held by Spike2 that start out as identical to the ones in the 1401. However, you can change them by adding more spikes, deleting templates or creating new ones. The templates in the 1401 do not change unless you click Online update to copy the latest templates to the 1401.



The leftmost of these 4 buttons shows the marker code assigned by the 1401; the other three buttons are not used on-line and are disabled. The Best match field displays the marker code of the template in the template area that is the best match to the spike in the display area. These will usually be the same. They can differ if you have edited the templates, or if Make templates is checked so that the template area is changing.

If you enable sound, the tones you hear depend on how the spikes match templates in the template area; the tones do not depend on the classification done by the 1401. This is to allow you to modify templates in preparation for Online update.

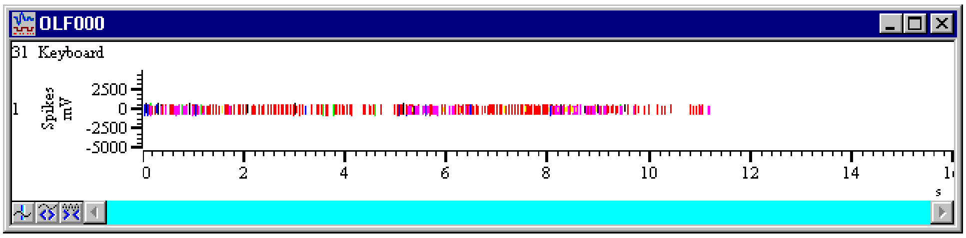


This is a latching button that is normally down for on-line use. Click the button to change the state between down and up. With this button down, the displayed spikes are always taken from the end of the data file, that is from the spikes that have just been sampled. If the spike rate is too high to show all the spikes, spikes are skipped.

With the button up, a new cursor appears in the time view to mark the point from which spikes are taken, all the play control buttons become enabled and the windows behaves almost identically to the off-line Edit WaveMark window which is described later. The marker code buttons still only allow the first marker code to be used; you can use all four codes when off-line.

Drawing modes for spikes

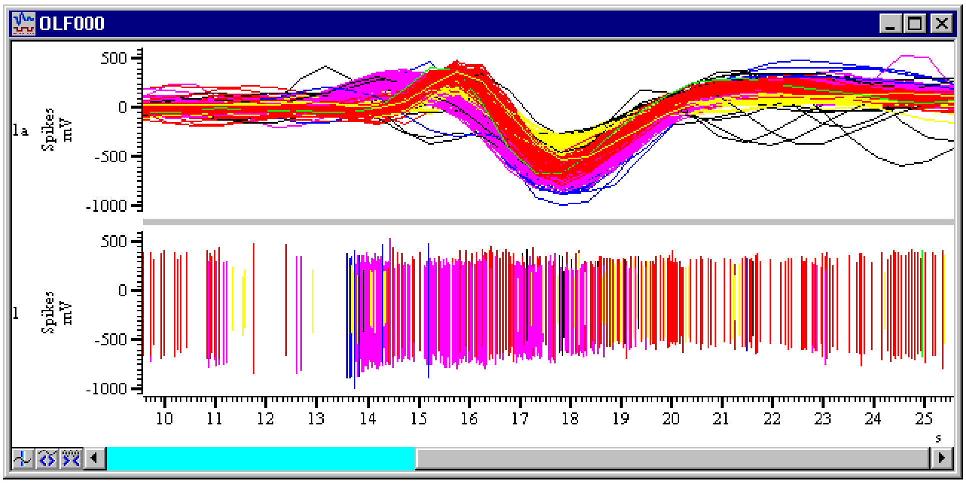
If you use the View menu Channel Draw Mode… command and select the WaveMark channel, you will find that you have a very wide choice of display modes. To start with, display the channel as Waveform. Arrange the x axis to show 10 to 20 seconds worth of data and click the Start button in the sampling control bar or use the Start Sampling command in the Sample menu. You should see something like the following:



If no spikes are appear, use the Analysis menu Marker Filter… command and select each of the four layer in turn and click All, then click OK. We will discuss this command in more detail later.

The spikes are colour coded for the template that they match, or are drawn in black if they don't match any template. These spikes are rather small, so the first thing to do is to double click the y axis and optimise the display. You can display the codes for the spikes as well as the waveforms by changing the drawing mode to WaveMark, but this is not very useful when you have a lot of spikes in the window as the codes will overwrite each other. Drawing in WaveMark mode on-line is much slower than drawing as Waveform, so we do not recommend this mode for on-line use.

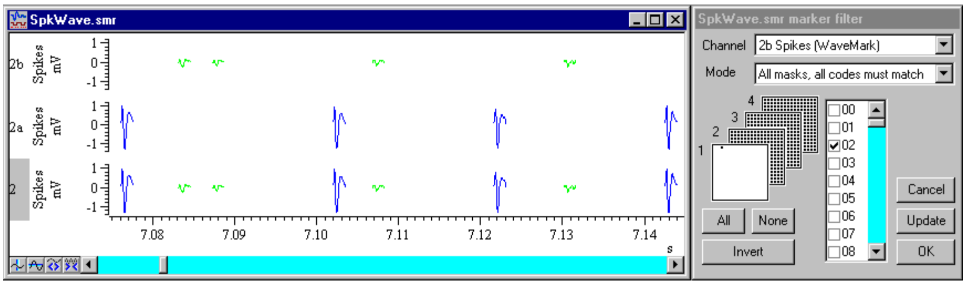
Another useful mode is Overdraw WM (overdraw WaveMark). This gives an immediate overview of the spike sorting. To see this mode in action, use the Analysis menu Duplicate Channels command to duplicate the WaveMark channel. Then use the View menu Channel Draw Mode… command to set the drawing mode for the duplicated channel to Overdraw WM. You may need to adjust the size of the data window to generate a useful image. In the next example we have hidden the keyboard channel.



In Overdraw WM mode, the full width of the window is used for each spike. There is a grey bar between the overdraw area and the rest of the window to show that the x axis does not apply. When the window scrolls, new spikes are added to the overdraw area, but old spikes are not removed as this would force the entire window to redraw, which could take a long time. If you want to force the overdraw area to redraw, click or drag the thumb in the scroll bar at the bottom of the window.

Using the Marker Filter

Although it is nice to see all the spikes you have collected, for most purposes, you are interested in the behaviour of one unit, or the relationship between one unit and another. If your templating and spike sorting has gone well, you will have isolated each unit as spikes with a particular code (or possibly more than one code). You now need a way to treat each unit as a separate channel of data.



In the example shown, channel 2 has two classes of spike, sorted into codes 1 and 2. before version 4.08 we generated channels 2a and 2b by duplicating channel 2 twice (right click on the channel and select Duplicate from the context menu). Then we right clicked on channel 2a and selected the Marker Filter command, cleared the check box next to 01 and clicked the Invert button. This cleared all check boxes except the one next to 01. Then we clicked Update and channel 2a then displayed only spikes with code 1. Next, we selected channel 2b in the channel list and repeated the procedure for code 02.

However, from version 4.08 onwards, there is a much easier way to do this. The Edit WaveMark dialog has a Duplicate! button that generates a duplicate channel with the marker filter set to display each template code.

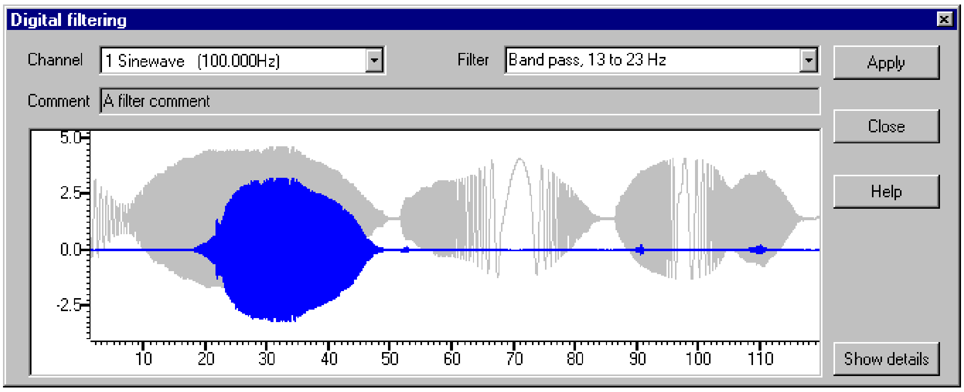
The result of this is that we have two new channels, each with spikes from a single unit. If your sorting resulted in one unit having more that one code, then you should select all the codes for that unit in the Marker Filter dialog; there is no need to sort spikes so that one template matches every single spike from the unit.

If you decide that spikes with several different codes should all have the same code, you can use the new Set Marker Codes command in the Analysis menu, or right click on a channel or use the Set Codes button in the Edit WaveMark dialog. Set the marker filter for the channel to display the codes that you wish to amalgamate then use the Set Marker Codes dialog to change all the spikes with these codes to the same code.

Digital filtering

Introduction

The Analysis menu Digital filters… command is available when you have a data file open that contains waveform channels. You can apply one of twelve stored digital filters to a waveform channel, or you can create your own digital filter. The program implements FIR filters (Finite Impulse Response) optimised to minimise the filter ripple in each filter band (see page 16-5 for more technical information on the filters).

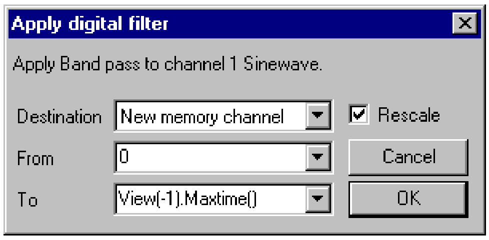


The Filter field of the dialog box selects the filter to apply and the Channel field sets the waveform channel to filter. The Comment field is for any purpose you wish; there is one comment per filter. The dialog shows the original waveform in grey, and a filtered version in the waveform colour.

The Close button shuts the dialog and will ask if you want to save any changed filter and the Help button opens the on-line Help at the digital filtering topic.

Apply

The Apply button opens a new dialog in which you set where to write the result of the filter operation on the data channel. You can write the result to a new memory channel, to an unused channel on disk, or to a memory channel that holds waveform data with the same sampling rate as the channel you are filtering.



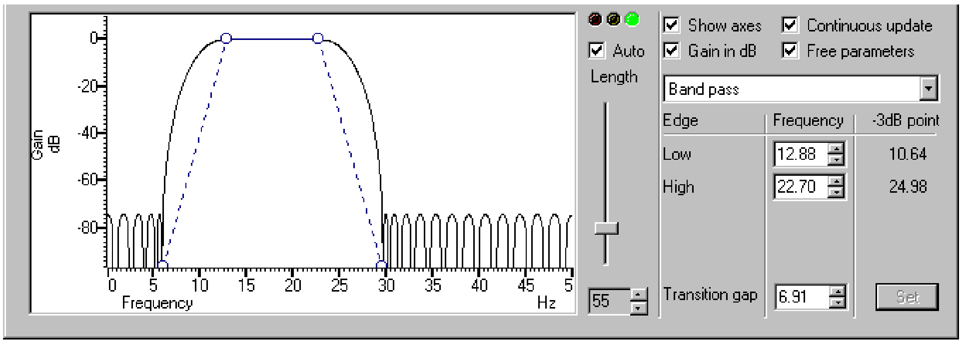
You can also set the time range of the output data. If the filter is of length n, then n/2 points around each input data point are used to produce each output point. When there is no input data available before or after a point, the filter uses a duplicate of the nearest input point as an estimate of the data value. This means that the n/2 output points next to any break in the input data should not be used for any critical purpose. For these purposes, the start and end of the file count as a break in the data.

If you check the Rescale box, the output data scale and offset is optimised to give the best possible representation of the waveform in the 16-bit numbers used to store waveform data in Spike2. To do this the program filters the data twice, once to find the range of the output, and once to store the result, so this doubles the time required to filter it. If you Rescale when you are adding data to an existing memory channel the range is optimised for both the added data and any remaining original data.

As applying a filter can be a lengthy process, a progress dialog appears with a Cancel button during the filtering operation.

Show details

The Show details button increases the dialog size to display a new area in which you can design and edit filters. Click this button again to hide the new dialog area.



The picture in the new area shows the frequency response of the filter. The Gain in dB check box sets the y axis scale to dB if checked, linear if not checked. The Show axes check box controls the axes of the raw and filtered data display. The frequency response display shows the ideal filter as solid lines for each defined band linked by dotted lines which mark each Transition gap between the bands. All transition gaps have the same width. The calculated frequency response is drawn as solid lines and is greyed when the filter specification has been changed and the response has not been calculated to match.

The circles can be dragged sideways to make the edges of the bands steeper or less steep or you can edit the band edges as numbers in the Frequency panel on the right. You can also drag the bands sideways and the band gaps. The mouse pointer changes to an appropriate symbol to indicate the feature you are dragging.

If you edit the numbers in the Frequency field, the Set button is enabled so you can force a recalculation of the filter.

The filters produced by the program are not defined in terms of -3dB corner frequencies and n dB per octave as is often the case for traditional analogue filters. The -3dB point column is present to help users who are more comfortable describing filter band edges in terms of the 3 dB point.

If you check the Continuous update box, the filter is updated while you drag the filter features around. If you have a slow computer and this feels ponderous you can clear the check box, in which case the filter is not recalculated until you stop changing features.

If you check the Free parameters box, dragged features are not limited by the next band and will push bands along horizontally. If you clear the box, the horizontal motion of a dragged feature is limited by the next moveable object.

To the right of the frequency response display is a slider that controls the number of filter coefficients. In general, the more coefficients, the better the filter. However, the more coefficients, the longer it takes to compute them and the longer to filter the data. If you check the Auto box, the program will adjust the number of coefficients for you to produce a useful filter. The “traffic light” display above the slider shows green if the filter is good, amber if the result is usable but not ideal, and red if the result is hopeless.

If you change a filter or create a new filter, you will be prompted to save the filter bank when you close the digital filter dialog.

Filter bank

A digital filter definition is complex and it would be tedious to specify all the properties of a filter each time you wanted to apply one to data. To avoid this, Spike2 contains a filter bank of 12 filter definitions. This filter bank is saved to the file `Filtbank.cfb` when you close Spike2 and reloaded when you open it. When you use the digital filter dialog, you specify which filter you want by the filter name. Script users identify the filter by an index number in the range 0 to 11.

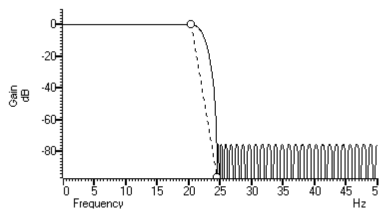
Filter types

The type of the filter is set by the drop down list to the right of the display. If you need a filter that is not in this list you can generate it from the script language. Users of the `FIRMake()` script language command should note that the bands referred to here are pass bands. In the script language there are additional stop bands between the pass bands. There are currently 12 different filter types:

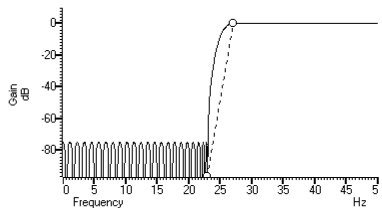
All pass This has no effect on your signal. This filter type covers the case where you apply a low pass filter designed for a higher sampling rate to a waveform with a much lower sampling rate, so that the pass band extends beyond half the sampling frequency of the new file.

All stop This removes any signal; the output is always zero. This filter type is provided to cover the case where you apply a high pass filter designed for a higher sampling rate to a waveform with a much lower sampling rate, so that the stop band extends beyond half the sampling frequency of the new file.

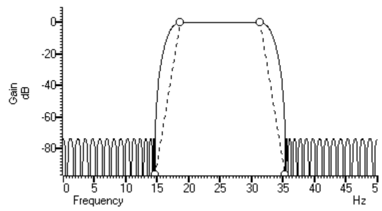
Low pass This filter attempts to remove the high frequencies from the input signal. The Frequency field holds one editable number, Low pass, the frequency of the upper edge of the pass band. The stop band starts at this frequency plus the value set by the Transition gap field.



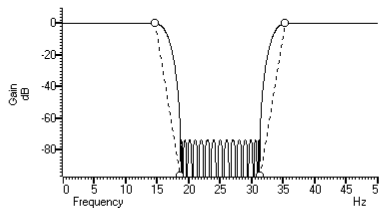
High pass A high pass filter removes low frequencies from the input signal. The Frequency field holds one editable number, High pass, the frequency of the lower edge of the pass band. The stop band starts at this frequency less the value set by the Transition gap field.



Band pass A band pass filter passes a range of frequencies and removes frequencies above and below this range. The Frequency field has two editable numbers, Low and High, which correspond to the two edges of the pass band. The stop band below runs up to Low-Transition gap, and the stop band above from High+Transition gap to one half the sampling rate.

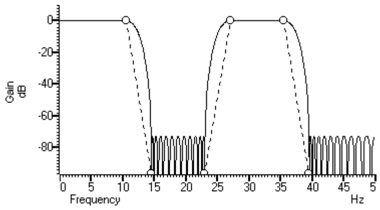


Band stop A band stop filter removes a range of frequencies. The Frequency field has two editable numbers, High (the upper edge of the first pass band) and Low (the lower edge of the upper pass band). The stop band below runs from High+Transition gap up to Low-Transition gap.



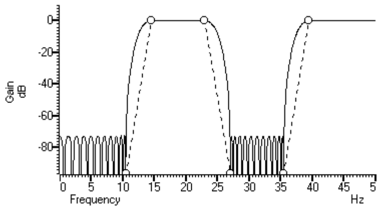
One and a half low pass

This filter has two pass bands, the first running from zero Hz and the second in the frequency space between the upper edge of the first pass band and one half the sampling rate. The Frequency field has three editable numbers: Band 1 high, Band 2 low and Band 2 high. These numbers correspond to the edges of the pass bands.



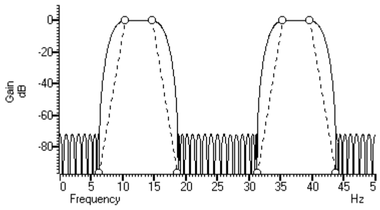
One and a half high pass

This filter has two pass bands. The second runs up to one half the sampling rate. The first band lies in the frequency space between 0 Hz and the lower edge of the second band. The Frequency field has three editable numbers: Band 1 low, Band 1 high and Band 2 low. These numbers correspond to the edges of the pass bands.



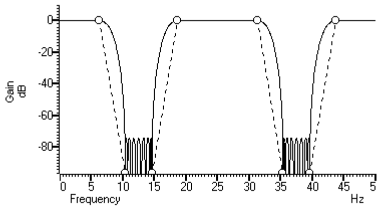
Two band pass

This filter passes two frequency ranges and rejects the remainder. Both 0 Hz and one half the sampling frequency are rejected. The Frequency field has 4 numeric fields: Band 1 low, Band 1 high, Band 2 low and Band 2 high. These fields correspond to the four edges of the two bands.



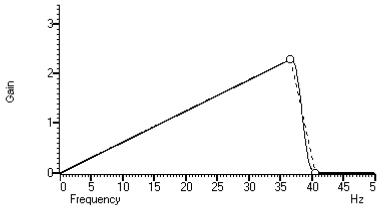
Two band stop

This filter passes three frequency ranges and rejects the remainder. Both 0 Hz and one half the sampling rate are passed. The Frequency field has 4 numeric fields: Band 1 high, Band 2 low, Band 2 high and Band 3 low. These fields correspond to the four edges of the three bands.



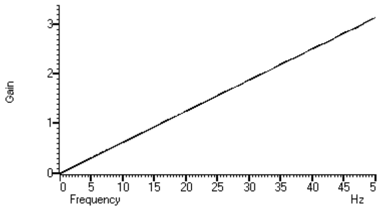
Low pass differentiator

This filter is a combination of a differentiator (that is the output is proportional to the rate of change of the input) and a low pass filter. The y axis scale is linear, rather than in dB (although you can display it in dB if you wish). There is one editable number in the Frequency field, Low pass, the end of the differential section of the filter.



Differentiator

The output of the filter is proportional to the rate of change of the input. The y axis scale is linear, rather than in dB (although you can display it in dB if you wish). The Frequency field is empty as there is only one band and it extends from 0 Hz to half the sampling rate.

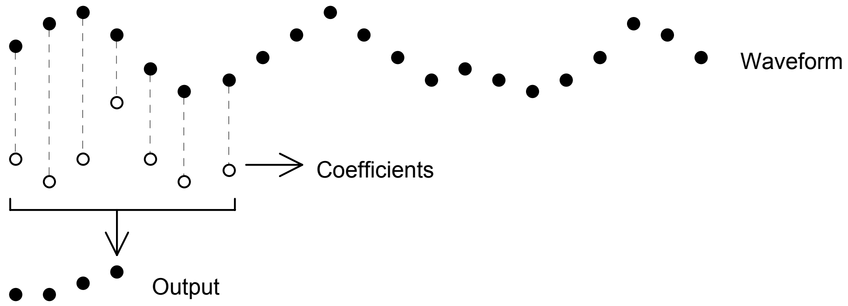


FIR filters

The `FIRMake()`, `FIRQuick()` and `FiltCalc()` script commands and the Analysis menu Digital filters… dialog generate FIR (Finite Impulse Response) filter coefficients suitable for a variety of filtering applications. The generated filters are optimal in the sense that they have the minimum ripple in each defined band. These filter coefficients are used to modify a sampled waveform, usually to remove unwanted frequency components. The algorithmic heart of the filter coefficient generation is based on the well-known FORTRAN program written by Jim McClellan of Rice University in 1973 that implements the *Remez exchange algorithm* to optimise the filter.

The theory of FIR filters is beyond the scope of this document. Readers who are interested in learning more about the subject should consult a suitable text book, for example *Theory and Application of Digital Signal Processing* by Rabiner and Gold published by Prentice-Hall, ISBN 0-13-914101.

FIR filtering



This diagram shows the general principle of the FIR filter. The hollow circles represent the filter coefficients, and the solid circles are the input and output waveforms. Each output point is generated by multiplying the waveform by the coefficients and summing the result. The coefficients are then moved one step to the right and the process repeats.

From this description, you can see that the filter coefficients (from right to left) are the *impulse response* of the filter. The impulse response is the output of a filter when the input signal is all zero except for one sample of unit amplitude. In the example above with 7 coefficients, there is no time shift caused by the filter. With an even number of coefficients, there is a time shift in the output of half a sample period.

Frequencies

The Analysis menu Digital filters… command deals with frequencies in Hz as this is comfortable for us to work with. However, if you calculate a FIR filter for one sampling rate, and apply the same coefficients to a waveform sampled at another rate, all the frequency properties of the filter are scaled by the relative sampling rates. That is, the frequency properties of an FIR filter are invariant when expressed as fractions of the sampling rate, not when expressed in Hz.

It is usually more convenient when dealing with real signals to describe filters in terms of Hz, but this means that each time a filter is applied to a waveform the sampling rate must be checked. If the rate is different from the rate for which the filter was last used, the coefficients must be recalculated. Unless you use the `FIRMake()` script command, Spike2 takes care of all the frequency scaling and recalculation for you. The remainder of this description is to help users of the `FIRMake()` script command, but the general principles apply to all the digital filtering commands in Spike2.

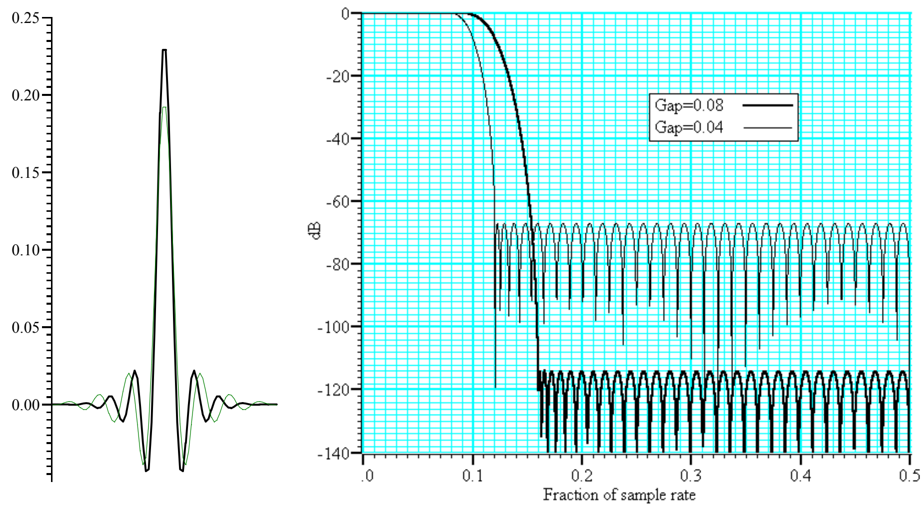
Users of the `FIRMake()` script command must specify frequencies in terms of fractions of the sample rate from 0 to 0.5. For example, if you were sampling at 10 kHz and you wanted to refer to a frequency of 500 Hz, you would call this 500/10000 or 0.05.

Example filter

The heavy lines in the next diagrams show the results obtained by `FIRMake()` when it designed a low pass filter with 80 coefficients with the specification that the frequency band from 0 to 0.08 should have no attenuation, and that the band from 0.16 to 0.5 should be removed. We can specify the relative weight to give to the ripple in each band. In this case, we said that it was 10 times more important that the *stop band* (0.16 to 0.5) should pass no signal than the *pass band* should be completely flat.

We have shown the coefficients as a waveform for interest as well as the frequency response of the filter. The shape shown below is typical for a band pass filter. One way of understanding the action of the FIR filter is to think of the output as the correlation of the waveform and the filter coefficients.

Coefficients and frequency response for low pass filters



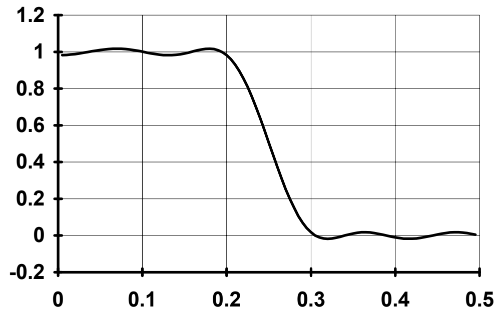
The frequency response is shown in dB, which is a logarithmic scale. A ratio r is represented by 20 $\log_{10}(r)$ dB. A change of 20 dB is a factor of 10 in amplitude, 6 dB is approximately a factor of 2 in amplitude. The graph shows that a frequency in the stop band is attenuated by over 110 dB (a factor of 300,000 in amplitude with respect to the signal before it was filtered).

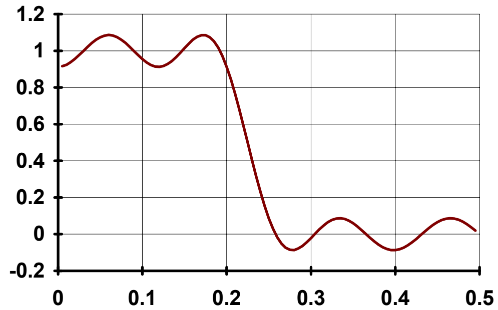
Because we didn't specify what happened between a frequency of 0.08 and 0.16 of the sampling rate, the optimisation pays no attention to this region. You might ask what happens if we make this transition gap smaller. The lighter line in the graph shows the result of halving the width of the gap by making the stop band run from 0.12 to 0.5. The filter is now much sharper. However, you don't get something for nothing. The attenuation in the stop band is reduced from 110 dB to around 70 dB. Although you cannot see it from the graph, the ripple in the pass band also increases by the same proportion (from 1 part in 30,000 to 1 part in 300).

We can restore the attenuation in the stop band by increasing the number of coefficients to around 120. However, there are limits to the number of coefficients it is worth having (apart from increasing the time it takes to calculate the filter and filter the data). Although the process used to calculate coefficients uses double precision floating point numbers, there are rounding errors and the larger the number of coefficients, the larger the numerical noise in the result.

Because the waveform channels are stored in 16-bit integers, there is no point designing filters that attenuate any more than 96 dB as this is a factor of 32768 (2^{15}). Attenuations greater than this would reduce any input to less than 1 bit. If you are targeting data stored in real numbers this restriction may not apply.

It is important that you leave gaps between your bands. The smaller the gap, the larger the ripple in the bands.





This is illustrated by these two graphs. They show the linear frequency response of two low pass filters, both designed with 18 coefficients (we have used so few coefficients so the ripple is obvious). Both have a pass band of 0 to 0.2, but the first has a gap between the pass band and the stop band of 0.1 and the second has a gap of 0.05. We have also given equal weighting to both the pass and the stop bands, so you can see that the ripple around the desired value is the same for each band.

As you can see, halving the gap has made a considerable increase in the ripple in both the pass band and the stop band. In the first case, the ripple is 1.76%, in the second it is 8.7%. Halving the transition region width increased the ripple by a factor of 5.

In case you were worrying about the negative amplitudes in the graphs, a negative amplitude means that a sine-wave input at that frequency would be inverted by the filter. The graphs with dB axes consider only the magnitude of the signals, not the sign.

FIRMake() filter types

`FIRMake()` can generate coefficients for four types of filter: Multiband, Differentiators, Hilbert transformer and a variation on multiband with 3 dB per octave frequency cut. The other routines can generate only Multiband filters and Differentiators.

Multiband filters

The filter required is defined in terms of frequency bands and the desired frequency response in each band (usually 1.0 or 0.0). Bands with a response of 1.0 are called *pass bands*, bands with a response of 0.0 are called *stop bands*. You can also set bands with intermediate responses, but this is unusual. The bands may not overlap, and there are gaps between the defined bands where the frequency response is undefined. You give a weighting to each band to specify how important it is that the band meets the specification. As a rule of thumb, you should make the weight in stop bands about ten times the weight in pass bands.

`FIRMake()` optimises the filter by making the ripple in each band times the weight for the band the same. The ripple is the maximum error between the desired and actual filter response in a band. The ripple is usually expressed in dB relative to the unfiltered signal. Thus the ripple in a stop band is the minimum attenuation found in that band. The ripple in a pass band is the variation of the frequency response from the desired response of unity. In some situations, for example audio filters, quite large ripples in the pass band are tolerable but the same ripple would be unacceptable in a stop band. For example, a ripple of -40 dB in a pass band (1%) is inaudible, but the same ripple in a stop band would allow easily audible signals to pass. By weighting bands you can increase the attenuation in one band at the expense of another to suit your application.

Differentiators

The output of a differentiator increases linearly with frequency and is zero at a frequency of 0. The differentiator is defined in terms of a frequency band and a slope. The frequency response at frequency *f* is *f* * *slope*. The slope is usually set so that the frequency response at the highest frequency is no more than 1.

The weight given to each frequency within a band is the weight for that band divided by the frequency. This gives a more accurate frequency response at low frequencies where the resultant amplitude will be the smallest.

Although you can define multiple bands for a differentiator, it is unusual to do so. Almost all differentiators define a single band that starts at 0. Occasionally a differentiator followed by a stop band is needed.

Hilbert transformers

A Hilbert transformer is a very specialised form of filter that causes a phase shift of $-\pi/2$ in a band, often used to separate a signal from a carrier. The theory and use of this form of filter is way beyond the scope of this document. Unless you know that you need this filter type you can ignore it.

Multiband with 3dB/octave cut

This is a variation on the multiband filter that can be used to filter white noise to produce band limited pink noise. The filter is identical to the band pass filter except that the attenuation increases by 3 dB per octave in the band (each doubling of frequency reduces the amplitude of the signal by a factor of the square root of 2). It is used in exactly the same way as the multiband filter.

Low pass filter example

A waveform is sampled at 1 kHz and we are interested only in frequencies below 100 Hz. We would like all frequencies above 150 Hz attenuated by at least 70 dB.

A low pass filter has two bands. The first band starts at 0 and ends at 100 Hz, the second band starts at 150 Hz and ends at half the sampling rate. Translated into fractions of the sampling rate, the two bands are 0-0.1 and 0.15 to 0.5. The first band has a gain of 1, the second band has a gain of 0. We will follow our own advice and give the stop band a weight of 10 and the pass band a weight of 1. We will try 40 coefficients to start with, so a possible script is:

```
var prm[5][2];       'Array for parameters
var coef[40];        'Array for the coefficients
'    band start        band end        function            weight
prm[0][0]:=0.00; prm[1][0]:=0.1; prm[2][0]:=1.0; prm[3][0]:= 1.0;
prm[0][1]:=0.15; prm[1][1]:=0.5; prm[2][1]:=0.0; prm[3][1]:=10.0;
FIRMake(1, prm[][], coef[]);
PrintLog("Pass Band ripple=%.1fdB Stop band attenuation=%.1f\n",
         prm[4][0], prm[4][1]);
```

If you run this, the log view output is:

```
Pass Band ripple=-28.8dB Stop band attenuation=-48.8
```

The attenuation in the stop band is only 48 dB, which is not enough. The ripple in the pass band is around 3% of the signal amplitude. We can increase the stop band attenuation in three ways: by increasing the number of coefficients, by giving the stop band more weight, or by making the gap larger between the bands.

We don't want to give the stop band more weight; this would increase the ripple in the pass band. We could probably reduce the width of the pass band a little as the attenuation of the signal tends to start slowly, but we will leave that adjustment to the end. The best way to improve the filter is to increase the number of coefficients. If we increase the size of `coef[]` to 80 coefficients and run again, the output now is:

```
Pass Band ripple=-58.7dB Stop band attenuation=-78.7
```

This is much closer to the filter we wanted. You might wonder if there is a formula that can predict the number of coefficients based on the filter specification. There is no exact relationship, but the following formula, worked out empirically by curve fitting, predicts the number of coefficients required to generate a filter with equal weighting in each of the bands and is usually accurate to within a couple of coefficients. The formula can be applied when there are more than two bands, but becomes less accurate as the number of bands increase.

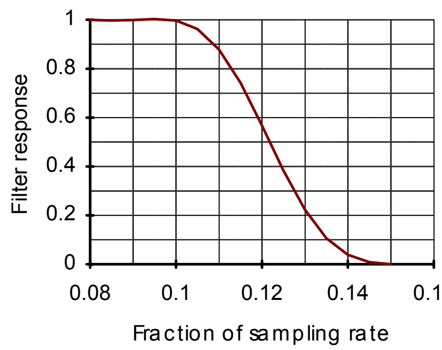
```
' dB     is the mean ripple/attenuation in dB of the bands
' deltaF is the width of the transition region between the bands
' return An estimate of the number of coefficients
Func NCoefMultiBand(dB, deltaF)
return (dB-23.9*deltaF-5.585)/(14.41*deltaF+0.0723);
end;
```

In our example we wanted at least 70 dB attenuation, and we weighted the stop band by a factor of 10 (20 dB). This causes a 10 dB improvement in the stop band at the expense of a 10 dB degradation of the pass band. Thus to achieve 70 dB in the stop band with the weighting, we need 60 dB without it. If we set these values in the formula (`dB` = 60, `deltaF`=0.05) , it predicts that 67.13 coefficients are needed. If we run our script with 67 coefficients, we get 70.9 dB attenuation, which is close enough!

A final finesse

If we look at the frequency response of our filter in the area between the pass band and the stop band, we see that the curve is quite gentle to start with. If you are used to using analogue filters, you will recall that the corner frequency for a low pass analogue filter is usually stated to be the frequency at which the filter response fell by 3 dB which is a factor of $\sqrt{2}$ in amplitude (when the response falls to 0.707 of the unfiltered amplitude).

If we use the analogue filter definition of corner frequency, we see that we have produced a filter that passes from 0 to 0.115 of the sampling rate, and we wanted from 0 to 0.1, so we can move the corner frequency back. This will increase the attenuation in the stop band, and reduce the filter ripple, as it widens the gap between the pass band and the stop band. If we move it back to 0.085, the attenuation in the stop band increases to 84 dB. Alternatively, we could move both edges back, keeping the width of the gap constant. This leaves the stop band attenuation more or less unchanged, but means that the start of the stop band is moved lower in frequency.

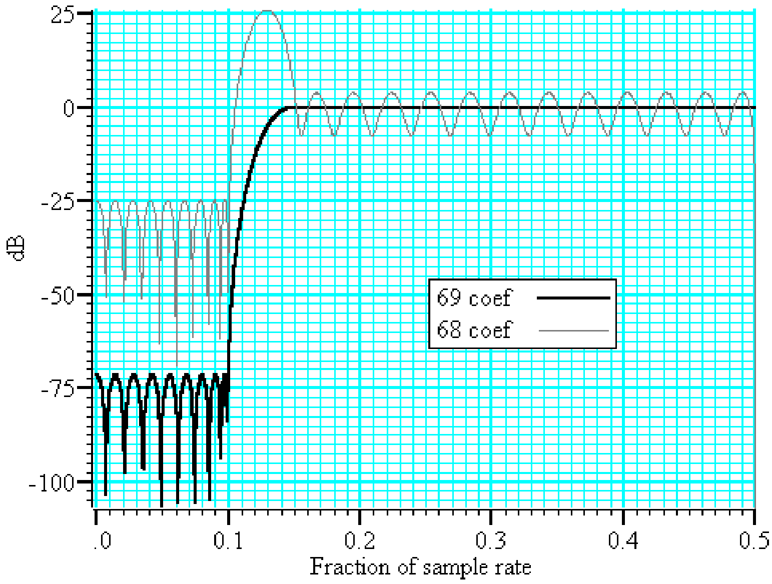


High pass filter

A high pass filter is the same idea as a low pass except that the first frequency band is a stop band and the second band is a pass band. All the discussion for a low pass filter applies, with the addition that **there must be an odd number of coefficients**. If you try to use an even number your filter will be very poor indeed. The example below shows a script for a high pass filter with the same bands and tolerances as for the low pass filter. We have added a little more code to draw the frequency response in a result view.

```
var prm[5][2];
var coef[69];
'    band start         band end         function            weight
prm[0][0]:=0.00; prm[1][0]:=0.1; prm[2][0]:=0.0; prm[3][0]:=10.0;
prm[0][1]:=0.15; prm[1][1]:=0.5; prm[2][1]:=1.0; prm[3][1]:= 1.0;
FIRMake(1, prm[][], coef[]);
const bins% := 1000;
var fr[bins%];
FIRResponse(fr[], coef[], 0);
SetResult(bins%, 0.5/(bins%-1), 0, "Fr Resp", "Fr","dB");
ArrConst([], fr[]);
Optimise(0);
WindowVisible(1);
```

Effect of odd and even coefficients



The graph shows the results of this high pass filter design with 69 coefficients, which gives a good result, and with 68 coefficients, which does not. In fact, if we had not given a factor of 10 weight (20 dB) to the stop band, the filter with 68 coefficients would not have achieved any cut in the stop band at all!

The reason for this unexpected result is that we have specified a non-zero response at the Nyquist frequency (half the sampling rate). If you imagine a sine wave with a frequency of half the sample rate, each cycle will contribute two samples. The samples will be 180° out of phase, so if one sample has amplitude *a*, the next will have amplitude -*a*, the next *a* and so on. The filter coefficients are mirror symmetrical about the centre point for a band pass filter, so with an even number of coefficients, the result when the input waveform is *a*, -*a*, *a*, -*a*... is 0. Another way of looking at this is to consider that a filter with an even number of coefficients produces half a sample delay. The output halfway between points that are alternately +*a* and -*a* must be 0.

You can use the formula given for the low pass filter to estimate the number of coefficients required, but you must round the result up to the next odd number.

General multiband filter

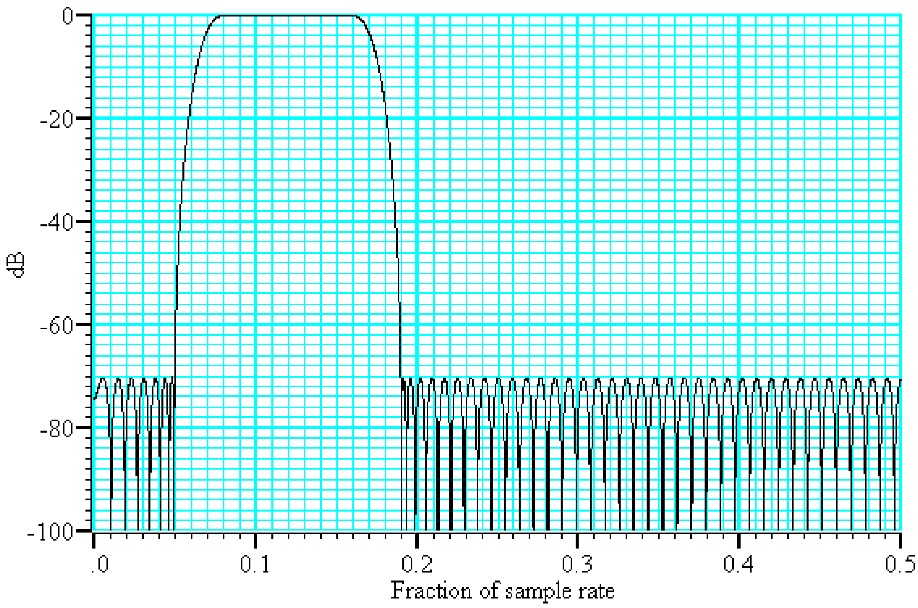
You can define up to 10 bands. However, it is unusual to need more than three. The most common cases with three bands are called band pass and band stop filters. In a band pass filter, you set a range of frequencies in which you want the signal passed unchanged, and set the frequency region below and above the band to pass zero. In a band stop filter you define a range to pass zero, and set the frequency ranges above and below to pass 1.

You must still allow transition bands between the defined bands, exactly as for the low and high pass filters, the only difference is that now you need two transition bands, not one. Also, if you want a non-zero response at the Nyquist frequency, you must have an odd number of coefficients.

For our example we will take the case of a signal sampled at 250 Hz. We want a filter that passes from 20 to 40 Hz (0.08 to 0.16) with transition regions of 7.5 Hz (0.03). If we say it is 10 times more important to have no signal in the stop band than ripple in the pass band, and we want 70 dB cut in the stop band we will get 50 dB ripple in the pass band (because a factor of 10 is 20 dB). To use the formula for the number of coefficients we need the mean attenuation/ripple in dB and the width of the transition region. The two stop bands have an attenuation of 70 dB and the pass band has a ripple of 50 dB, so the mean value is (70+50+70)/3 or 63.33 dB. We have two transition regions (both the same width). In the general case of transition regions of different sizes, use the smallest transition region in the formula. Plugging these values into the formula predicts 113 coefficients, however only 111 are needed to achieve 70 dB.

```
var prm[5][3];              ' 3 bands for band pass
var coef[111];              ' 111 coefficients needed
'    band start         band end        function           weight
prm[0][0]:=0.00; prm[1][0]:=0.05; prm[2][0]:=0.0; prm[3][0]:=10.0;
prm[0][1]:=0.08; prm[1][1]:=0.16; prm[2][1]:=1.0; prm[3][1]:= 1.0;
prm[0][2]:=0.19; prm[1][2]:=0.50; prm[2][2]:=0.0; prm[3][2]:=10.0;
FIRMake(1, prm[][], coef[]);
```

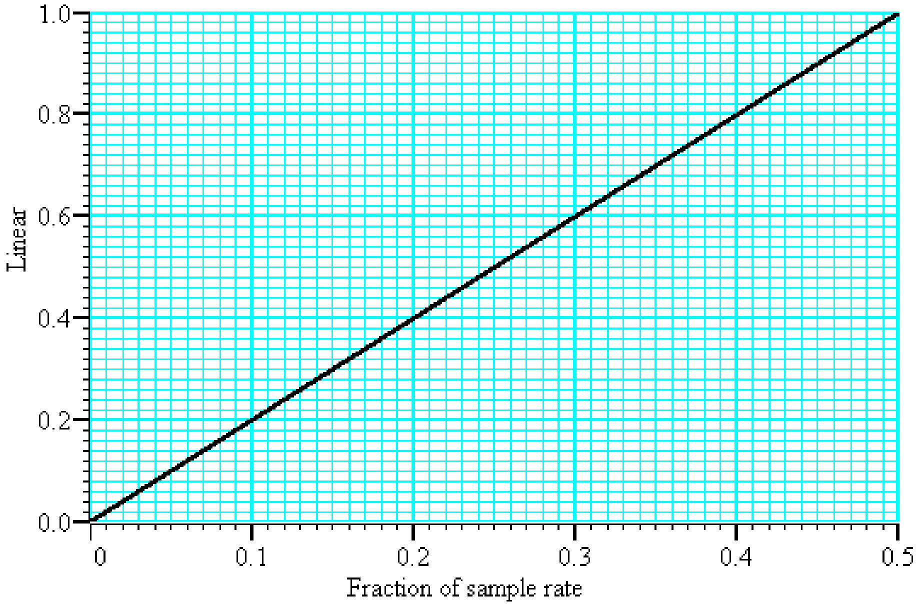
Band pass filter with 111 coefficients



Differentiators

A differentiator filter has a gain that increases linearly with frequency over the frequency band for which it is defined. There is also a phase change of 90° (π/2) between the input and the output.

Ideal differentiator with slope of 2.0



You define the differentiator by the number of coefficients, the frequency range of the band to differentiate and the slope. The example above has a slope of 2. Within each band (normally only 1 band is set) the program optimises the filter so that the amplitude of the ripple (error) is proportional to the response amplitude. A differentiator is normally defined to operate over a frequency band from zero up to some frequency `f`. If `f` is 0.5, or close to it, you must use an even number of coefficients, or the result is very poor. You can estimate the number of coefficients required with the following function:

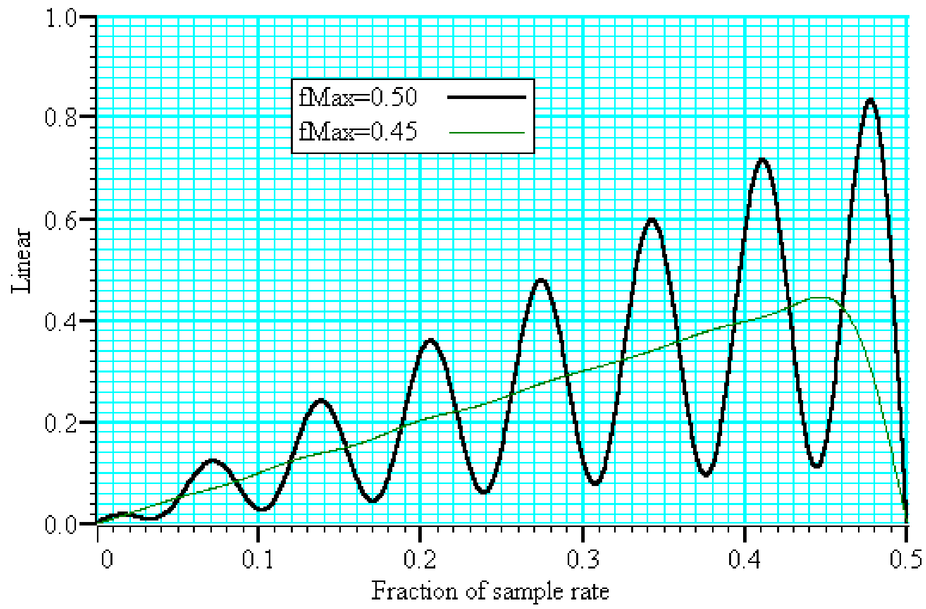
```
' dB    the proportional ripple expressed in dB
' f     the highest frequency set in the band
' even% Non-zero if you want an even number of coefficients
func NCoefDiff(dB, f, even%)
if (f<0) or (f>0.5) then return 0 endif;
f := 0.5-f;
var n%;
if (even%) then
   n% := (dB+43.837*f-35.547)/(0.22495+29.312*f);
   n% := (n%+1) band -2;   'next even number
else
   if f=0.0 then return 0 endif;
   n% := dB/(29.33*f);
   n% := n% bor 1;         'next odd number
endif;
return n%;
end
```

For an even number of coefficients this is unreliable when `f` is close to 0.5. For an odd number, no value of `n` works if `f` is close to 0.5.

These equations were obtained by curve fitting and should only be used as a guide. To make a differentiator that uses a small number of coefficients, use an even number of coefficients and don't try to span the entire frequency range. If you cannot tolerate the half point shift produced by using an even number of coefficients and must use an odd number, you must set a band that stops short of the 0.5 point. Remember, that by not specifying the remainder of the band you have no control over the effect of the filter in the unspecified region. However, for an odd number of points, the gain at the 0.5 point will be 0 whatever you specify for the frequency band.

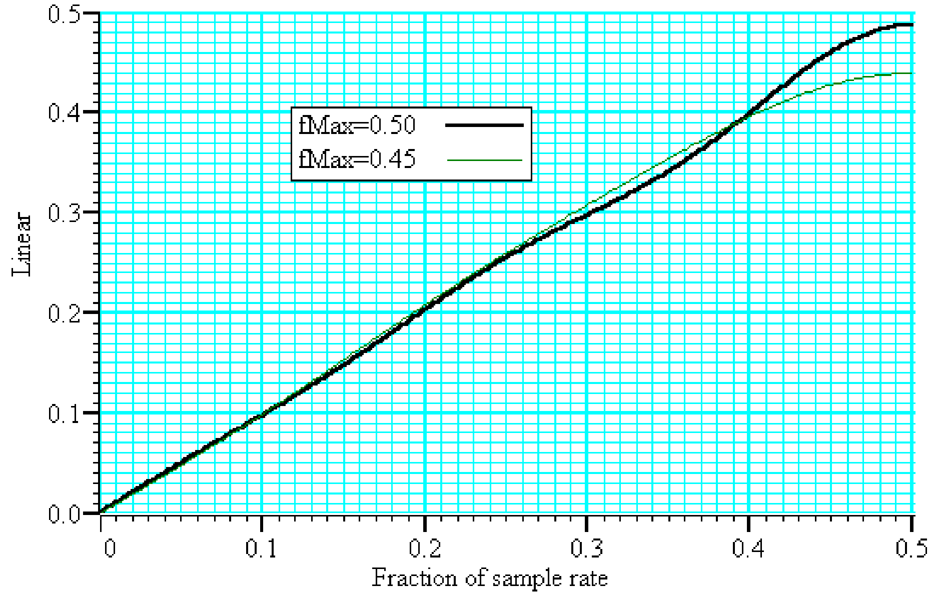
The graph below shows the effect of setting an odd number of coefficients when generating a differentiator that spans the full frequency range. The second curve shows the improvement when the maximum frequency is reduced to 0.45.

Differentiators with 31 coefficients



If you must span the full range, use an even number of coefficients. The graph below shows the improvement you get with an even number of coefficients. The ripple for the 0.45 case is about the same with 10 coefficients as for 31.

Differentiators with 10 coefficients




```
var prm[4][1];       ' 1 bands for differentiator
var coef[10];        ' 10 coefficients needed
'    band start         band end            slope              weight
prm[0][0]:=0.00; prm[1][0]:=0.45; prm[2][0]:=1.0; prm[3][0]:=1.0;
FIRMake(2, prm[][], coef[]);
```

Hilbert transformer

A Hilbert transformer phase shifts a band of frequencies from F_{low} to F_{high} by -π/2. The target magnitude response in the band is to leave the magnitude unchanged. F_{low} must be greater than 0 and for the minimum magnitude overshoot in the undefined regions, F_{high} should be 0.5-F_{low}. The magnitude response at 0 is 0, and if an odd number of coefficients is set, then the response at 0.5 is also 0. This means that if you want F_{high} to be 0.5 (or near to it), you must use an even number of coefficients.

There is a special case of the transformer where there is an odd number of coefficients and F_{high} = 0.5-F_{low}. In this case, every other coefficient is 0. This is no help to the MSF and MSF programs, but users who write their own software can use this fact to minimise the number of operations required to make a filter.

It is extremely unlikely that a Hilbert transformer will be of any practical use in the context of Spike2, so we do not consider them further. You can find more information about this type of filter in *Theory and Application of Digital Signal Processing* by Rabiner and Gold.

Programmable signal conditioners

Overview

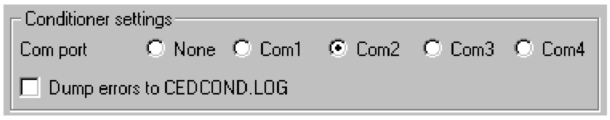
Spike2 supports the CED 1902 Mk III and the Axon Instruments CyberAmp signal conditioners and also the Power1401 gain option through the signal conditioner control panel. You can open the conditioner control panel from either the sampling configuration channel parameters dialog (when the channel type is waveform or WaveMark) or from the Sample menu. The 1902 and CyberAmp signal conditioners are controlled through a serial port which is set in the Edit menu Preferences… option.

What a signal conditioner does

A signal conditioner takes an input signal and amplifies, shifts and filters it so that the data acquisition unit can sample it effectively. Many input signals from experimental equipment are too small, or are masked by high and or low frequency noise, or are not voltages and cannot be connected directly to the 1401.

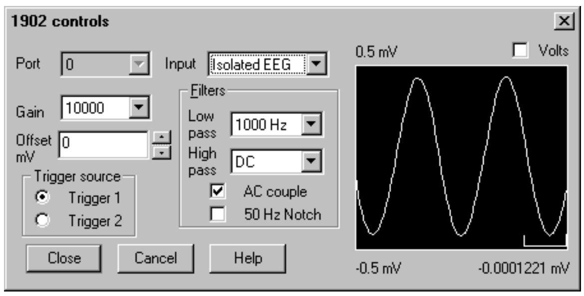
Signal conditioners may also have specialist functions, for example converting transducer inputs into a useful signal, or providing mains notch filters. The CED 1902 has options for isolated inputs and specialised front ends include ECG with lead selection, magnetic stimulation artefact clamps and EMG rectification filtering and filtering. The only option for the Power1401 is Gain. You should consult the documentation supplied with your signal conditioner to determine the full range of capabilities.

Serial ports

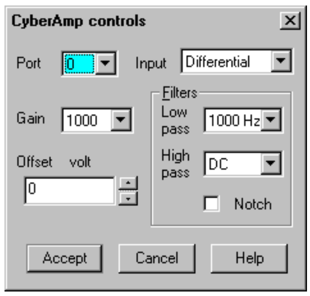


The basic communication parameters are set in the Edit menu Preferences dialog. Programmable signal conditioners are controlled through communication (serial) ports. No port is used if the conditioner support is not loaded or None is selected in the preferences or for the Power1401 gain option. Check the *Dump errors…* box to write diagnostic messages to `CEDCOND.LOG` in the current folder. Do not check the box unless you are having problems with a conditioner as it slows Spike2 down.

Control panel



The control panel is in two halves. The left-hand half holds the controls that change the conditioner settings, the right-hand half displays data from the conditioner. The right-hand half is omitted if Spike2 is sampling data, or if the 1401 is not available for any other reason.



If the right-hand half is present, the Volts check box causes the data to be displayed in Volts at the conditioner input in place of user units as defined by the Channel parameters dialog. The number at the bottom right is the mean level of the signal in the area marked above the number.

Signal conditioners differ in their capabilities. Not all the controls listed below may appear for all conditioners. This example is the CyberAmp control panel (with the right-hand half omitted). The controls are:

Port This is the physical 1401 port that the conditioner is attached to. If you open the conditioner dialog from the channel parameters dialog you cannot change the port.

Input If your signal conditioner has a choice of input options, you can select the input to use with this field. The choice of input may also affect the ranges of the other options.

Gain This field sets the gain for the signal selected by the Input field. Spike2 tracks changes of gain (and offset) and changed the channel gain and offset in the sampling configuration to preserve the y axis scale. You should adjust the gain so that the maximum input signal does not exceed the limits of the data displayed on the right of the control panel. When Spike2 is sampling, the gain and offset are fixed once you have written data to disk.

This is the only editable field for the Power1401 ADC Gain option.

Offset Some signals are biased away from zero and must be offset back to zero before they can be amplified. If you are not interested in the mean level of your signal, only in the fluctuations, you may find it much simpler to AC couple (1902) or high-pass filter (CyberAmp) the signal and leave the offset at zero. If Spike2 is sampling you cannot change the offset once data has been written to disk.

Low-pass filter A low-pass filter reduces high-frequency content in your signal. Filters are usually specified in terms of a corner frequency, which is the frequency at which they attenuate the power in the signal by a factor of two and a slope, which is how much they increase the attenuation for each doubling of frequency. Sampling theory tells us that you must sample a signal at a rate that is as least twice the highest frequency component present in the data. If you do not, the result may appear to contain signals at unexpected frequencies due to an effect called aliasing. As the highest frequency present will be above the corner frequency you should sample a channel at several times the filter corner frequency (probably between 3 and 10 times depending on the signal and the application).

You can choose a range of filter corner frequencies, or you can choose to have the data unfiltered (for use when the signal is already filtered due to the source).

High-pass filter A High-pass filter reduces low-frequency components of the input signal. High-pass filters area specified in the same way as low-pass filters in terms of a corner frequency and a slope, except that the slope is the attenuation increase for each halving of frequency. If you set a high-pass filter, a change in the mean level of the signal will cause a temporary change in the output, but the output will return to zero again after a time which depends on the corner frequency of the filter. The lower the corner frequency, the longer it takes for mean level change to decay to zero.

Notch filter A notch filter is designed to remove a single frequency, usually set to the local mains power supply (50 Hz or 60 Hz, depending on country).

The remaining options are for the 1902 only:

AC couple This is present for the 1902 only, and can be thought of as a high-pass filter with a corner frequency of 0.16 Hz. However, it differs from the high-pass filters as it is applied to the signal at the input; the high-pass filters in the 1902 are applied at the output.

Trigger source The 1902 provides two conditioned trigger inputs, and one output. This control selects which of the inputs is connected to the output.

Setting the channel gain and offset

If you change the gain or offset in the control panel, Spike2 will adjust the channel gain and offset in the sampling configuration to compensate so as to keep the y axis showing the same units. This means that if you change the gain, the signals will still appear in the same units in the file. However, the first time you calibrate the channel you must tell the system how to scale the signal into y axis units.

For example to set up the y axis scales in microvolts you do the following:

1. Open the Sampling Configuration dialog.
2. Select the waveform or WaveMark channel and open the channel parameters dialog.
3. Set the Units field of the Channel parameters to uV.
4. Set the Input in Volts x field to 1000000.
5. Press the Conditioner button to open the conditioner control panel.
6. Adjust the gain to give a reasonable signal.
7. Close the signal conditioner control panel.

You only need do steps 3 and 4 once. Any subsequent change to the conditioner gain will adjust the channel gain to leave the units in microvolts.

For the more general case where you have a transducer that measures some physical quantity (Newtons, for example) and it has an output of 152.5 Newtons per mV. If you wanted the Y axis scaled in Newtons, you would replace steps 6 and 7 above with:

3. Set the Units field of the Channel parameters to N.
4. Set the Input in Volts x field to 0.1525.

To work this out you must express the transducer calibration in terms of Units per Volt (in this case Newtons per Volt).

If you have set an offset in the conditioner, and you want to preserve the mean signal level, you should null it out by changing the offset in the Channel parameters dialog.

Conditioner connections

Spike2 usually expects the first channel of signal conditioning to be connected to 1401 ADC port 0, the second to channel 1 and so on. Connect the conditioner output BNC (labelled Amp Out on the 1902, and OUT on the CyberAmp) to the corresponding 1401 input. You can override this to start at another port, but in all cases the conditioner channel numbers must be contiguous and must match the ADC ports they connect to.

Signal conditioners connect to the computer with a serial line. You will have received a suitable cable with the unit. Basic communication and connection information is stored in the file `CEDCOND.INI` in the system directory of your computer. This file holds:

```
[General]
Port=COM1
```

The `Port` value sets the communications port to use. We would recommend you use the Edit menu Preferences… to change the port. If this file is missing, COM1 is used. Preferences… can also set a diagnostic option, enabled in the file by:

```
Dump=1
```

If this entry is included, the file `CECOND.LOG` holding diagnostic messages is written to the folder that Spike2 was run from.

As mentioned above, the first signal conditioner is usually connected to ADC port 0, the second to port 1, and so on. The program searches for signal conditioners based on this assumption. The search starts at signal conditioner channel 0 and continues until a channel does not respond. The search sequence can be changed by two additional lines that you can insert manually (not supported by Preferences…) into `CEDCOND.INI`:

```
First=1
Last=3
```

If you do not supply these values, they are assumed to be 0. How these values are used depends on the signal conditioner:

CED 1902 CED 1902s have unit numbers set by an internal switch pack; multiple units usually have the channel number as a label on the front panel. If you have multiple units, they must have different unit numbers. Each 1902 output should be connected to the 1401 ADC input with the same number as the 1902 unit.

`First` and `Last` set the range of unit numbers to search. The search continues beyond `Last` until a unit does not respond. The purpose of `Last` is to force the search to skip over missing conditioners. The purpose of `First` is to skip over missing lower-numbered conditioners to avoid time delays waiting for them to respond.

Normally you will have 1902s with consecutive unit numbers starting at 0, in which case you do not need to set `First` and `Last`. As an example of a more complicated situation, let us suppose you have unit numbers 4, 5, 6, 7, 12, 13, 14 and 15. In this case, you should set `First` to 4 and `Last` to 15.

CyberAmp CyberAmps have a device Address set by a rotary switch on the rear panel. If you have multiple units, they must have different addresses. There are two types of CyberAmp: 8-channel and 2-channel. If you have multiple units, they must all have the same number of channels, or all the 8-channel units must have lower device addresses than all the 2-channel units.

`First` and `Last` set the range of addresses to search. The search continues beyond `Last` until a device does not respond. The purpose of `Last` is to force the search to skip over missing devices. The purpose of `First` is to skip over missing lower-numbered devices to avoid time delays waiting for them to respond.

The range of ADC channels supported by each CyberAmp is set by the device address (`dev`) and the number of channels in the first device detected (`num`). The first ADC channel supported by the device at address `dev` is `dev*num`. Each CyberAmp support 8 or 2 consecutive ADC channels.

Normally you will have one 8-channel CyberAmp, and you will give it address 0 to support ADC channels 0 to 7. In this case you do not need to set `First` and `Last`. If you want to connect it to channels 8-15, you would set both First and the device address to 1 and then you could connect it to channels 8-15.

Here is a more complex example with three CyberAmps, two with 8-channels and one with 2-channels. The table shows some possible configurations (assuming you have 32 ADC inputs to choose from):

`CEDCOND.INI`		8-channel unit 1		8-channel unit 2		2-channel unit 3	
`First`	`Last`	Switch	ADC	Switch	ADC	Switch	ADC
0	2	0	0-7	1	8-15	2	16-17
1	3	1	8-15	2	16-23	3	24-25
2	3	2	16-23	3	24-31	-	-

Spike2 can control a CED 1401-18 programmable discriminator card fitted in a standard 1401 or 1401*plus*. The Sample menu Discriminator Configuration… command opens the discriminator setup tabbed dialog. The dialog has sections that control the signal routing through the card, the mode of discrimination, the trigger levels and the waveform channel used as a signal monitor.

What the 1401-18 does

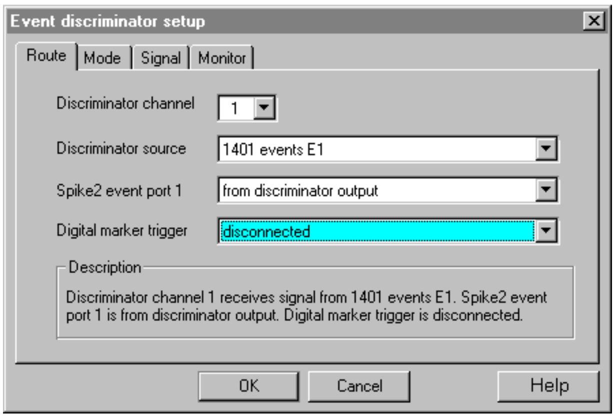
The 1401-18 event discriminator card detects waveform events using two trigger levels and an optional time window. The output from the detector is a digital signal suitable for input to the Spike2 event channels as event and level event data.

The card contains eight discriminator channels numbered 0 to 7. Channels 0 to 5 can take their input from either a front panel event input BNC or a digital input. Channels 6 and 7 take their input only from the digital input port. The input voltage range is ±5 Volts. The output of the channels is normally routed to the Spike2 event channels, however channel 1 can be routed to the digital marker trigger and channel 3 to the start sampling trigger.

Spike2 saves the discriminator setup as part of the sampling configuration. The dialog is used before sampling starts to prepare the 1401-18; it cannot be used during sampling. However, you can use the script language discriminator support during sampling.

Signal route

The Route tab in the Event discriminator setup dialog sets the source and destination of the discriminator, Spike2 event port, digital marker trigger and sampling trigger data. All eight channels are independent of each other.



The text Description field gives a synopsis of the settings for the channel. The fields are:

Discriminator channel

This selects the discriminator channel to adjust. It also sets the channel for the Mode and Signal tabs. You can also change channel in the Signal tab.

Discriminator source

This selects the input to take waveform data from. All channels can take data from the digital input connector, and channels 0 to 4 can use one of the front panel event inputs. Channel 5 can take its input from the ADC Ext front panel input. You can also set the source as Not used in which case the signals for this channel behave as if the discriminator were not installed. When a front panel input is used as a source, that input cannot be used as the source of any other signal.

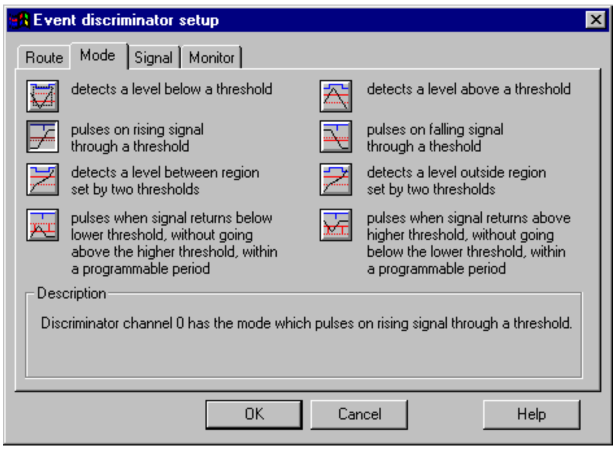
Spike2 event port n

This field sets where Spike2 takes the data used for Event and Level data from for port n. This can come from the digital input, or from the discriminator output. You can also disconnect it (in which case you will not be able to collect data from this port).

Digital marker trigger
Sampling trigger

The Digital marker trigger field is present for channel 1 only, the Sampling trigger field is present for channel 3 only. You can choose to take these signals either from the front panel E1 (Digital marker trigger) or E3 (Sampling trigger) inputs, from the discriminator output, or to leave the trigger disconnected.

Discrimination mode



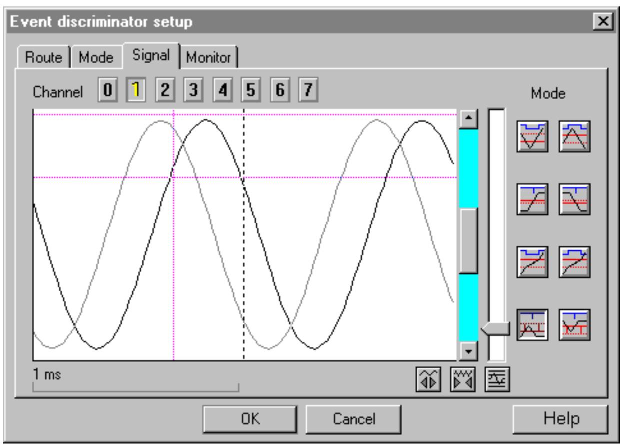
The Mode tab in the Event discriminator setup dialog sets the discrimination mode for the current channel and gives you a description of each mode. You can change the channel in the Route and Signal tabs. You can also set the mode from the Signal tab.

Set the mode by clicking one of the buttons. There are eight buttons, but only four modes; the right-hand column modes are the same as the left-hand column but with the active signal in the opposite direction.

In the pictures on the buttons, the main threshold is shown as a solid line. The dotted line is a second threshold. The four modes are:

1. The first mode detects the signal above or below a band defined by the two levels. When the signal lies between the levels, the last level crossed determines the state of the output. The output is suitable for use in a Level data channel.
2. The second mode produces a 1 µs pulse when the signal crosses the main threshold level. The data must cross the second threshold before another pulse can occur. The output is suitable for an Event data channel.
3. The third mode detects when the signal lies between the two thresholds (or conversely, lies outside the levels). The output is suitable for use in a Level data channel.
4. The last mode produces a 1 µs pulse when the signal crosses the main threshold and falls below it again without crossing the second threshold within a set time period. The output is suitable for an Event data channel.

View data



The Signal tab shows the raw and discriminated waveform on a selected channel so you can set the threshold levels. The buttons along the top of the window let you swap channel quickly and the buttons on the right swap modes (see the Mode tab for descriptions of each mode).

The vertical scroll bar moves the data vertically in the window. The slide bar to the right of the scroll bar changes the vertical magnification of the displayed wave. If your signal looks very small on the screen even with the gain slider at the top, your signal needs more amplification before it can be usefully discriminated by the 1401-18. At maximum gain, the threshold levels are accurate within a few pixels in the vertical direction.

The two buttons below the vertical scroll bar change the time width of the displayed data. The button under the slide bar optimises the display and threshold levels based on the current trigger mode and recent data.

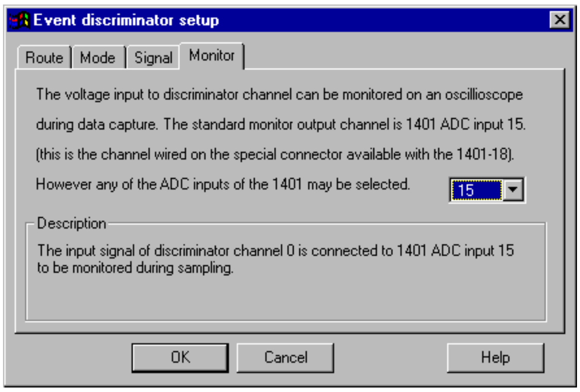
To set up a channel, follow these steps:

1. Select the channel with the buttons above the displayed data.
2. Select the discrimination mode with the buttons on the right.
3. Use the width controls and the optimise button to display data and cursors.
4. Adjust the trigger levels (and the time window if enabled) by dragging the trigger levels (and time marker) with the mouse.
5. Repeat for all channels and click OK when you have finished.

The data area shows the last event that caused a trigger and the current input (if there is no triggering event). During setup, the computer may not be fast enough to show you every event that causes a trigger. However, during sampling, all such events are captured.

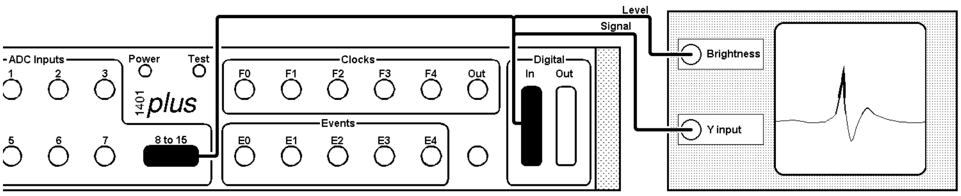
You may sometimes see trigger events that do not appear to cross a threshold. This can happen when the data contains spikes of very short duration. The displayed data is sampled, and it can happen that very short spikes fall between samples, so are invisible on the screen. This is usually an indication that the input data contains high-frequency noise and should be passed through a filter.

Monitor channel



Spike2 uses a waveform input channel to monitor the data entering the discriminator. The cables supplied with the 1401-18 card connect the monitor output to ADC channel 15. It is unusual to change the monitor channel. The supplied cables also have labelled BNC connectors for an oscilloscope. The oscilloscope outputs are active when Spike2 captures data (when the Signal option cannot be used) and can be used to monitor the discriminator action on the current channel.

Cable connections with the 1401-18



It is important to fit this cable (or make the connections with your own wiring) so that the Signal option can display the current discriminator channel. If you connect your input signals to the 1401 through the digital inputs, you must open the 25-way connector on the cable and wire your signals into it.

Digital input connections

The functions of the digital input connector are changed with the 1401-18 card present. The pins of the digital inputs that previously were the Spike2 event inputs (see the *Sampling data* chapter) now become discriminator inputs when selected in the Route tab. The discriminator outputs are buffered and connected to the pins that were previously the digital marker inputs (TTL output pin in the table below). You can still use digital markers as the digital marker data bit inputs are duplicated on the same pin numbers on the digital output port. However, the analogue ground uses pins 23 and 24, so the optional strobe and h/s signals are not available if you have the discriminator fitted.

Digital input pins for discriminator channels

Discriminator channel	7	6	5	4	3	2	1	0
Data input pin	1	14	2	15	3	16	4	17
TTL output pin	5	18	6	19	7	20	8	21

Other digital input signal pin numbers

Analogue ground	22, 23 and 24		
Level Monitor output	10	Signal Monitor output	11
Digital ground	13	+5 Volts	25

The Signal Monitor output is the raw waveform data for the current discriminator channel. The Level Monitor output signal is a 3-level signal that can be used for 'scope brightness controls, indicating where the input lies with respect to the two threshold levels.

Electrical information

All 1401-18 waveform inputs have a working range of ±15 Volts. There is a 20 Volt over-voltage protection on all inputs, even on power-off. When the event discrimination is by-passed, the card has a maximum impedance of 50 Ohms. When a channel is set for conditioning, the impedance is 100 kOhms, with 100 nA bias current and up to 2 mV offset.

The two thresholds have a ±5 Volt programmable range, with a 12-bit resolution. Linearity is to ±2 bits, the offset accuracy is ±5 bits. There are no adjustments.

There is a fixed 20 mV hysteresis in all modes built-in to the hardware. In modes that use the second threshold level to program hysteresis the software allows for the extra 20 mV. This hysteresis is present to stop noise on the input causing spurious, high frequency triggering.

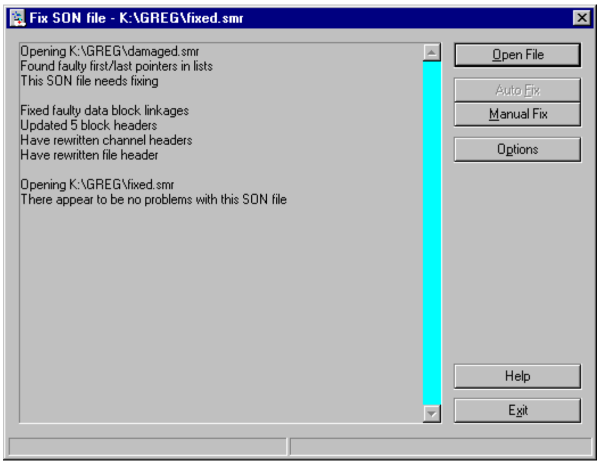
Input pulse widths should be at least 1.2 µs; pulses narrower than this may not be detected. Pulse outputs are approximately 1 µs in length.

The time window is programmable from 20 µs to 0.65535 seconds in steps of 10 µs.

The two monitor output signals can drive loads as low as 600 Ohms. The Level Monitor output signal has the following voltage levels:

2 to 5 Volts	Signal above both thresholds
1 to -1 Volts	Signal between thresholds
-2 to -5 Volts	Signal below both thresholds

The SonFix data recovery utility



The `SonFix` utility program recovers damaged Spike2 data files. A Spike2 data file has a header followed by a channel description list, followed by data blocks. Each data block has links to the previous and next data block of the same channel. If a file becomes damaged, these links can be broken, rendering the file useless. The `SonFix` program scans the file, rebuilds the links for each channel and corrects the file information in the file header. There are two common types of damage that occur to Spike2 data files:

1. The file is damaged as it is sampled. This can happen if your system loses power. The next time you run Spike2 it will tell you where the data can be found and recommend that you run `SonFix` to repair the file.
2. The file is damaged due to some other disk accident (such as a disk crash or when archiving and restoring files to tape or some other removable media). As long as the damage is not at the start of the file, the file can usually be fixed.

This version of `SonFix` relies on the header and channel list being intact. However, if the program does not recognise the file as a Spike2 data file, don't despair. It is possible to patch the file header with a header from a similar file, then recover the data. Contact the CED Help desk if you are in this situation. Future versions of `SonFix` may add this facility to the program. From version 3.05 of Spike2 onwards, you can set a file flush period in the Sampling Configuration Automation page. If you sample large files, we recommend that you use this feature to make file recovery much more certain.

Using SonFix

If you have a disaster, check that your disk drive is not damaged. Writing more data to a damaged drive is likely to cause further problems and may render your data unrecoverable. If the hard disk is damaged you must repair it with a suitable disk utility program or find someone who can do this for you.

Next, find the data file. If you have to use a disk repair utility, you may find it useful to know that the first two bytes of a data file hold the file revision followed by the ASCII characters "`(C) CED 87`". If sampling did not complete properly, Spike2 will tell you where to look for the damaged file the next time you run the program.

In most cases there will be no disk damage and the file will be present with a non-zero size. **If your file is very valuable**, make certain you do not fix it in place by saving the fixed file with the same name as the damaged file. Run `SonFix` by double clicking its icon (it is in the same folder as the Spike2 program). Click the Open File button and select the file to repair and `SonFix` will tell you if it can repair the file. Click the Auto Fix button to carry out the repair.

What cannot be recovered

Data that was not written to the physical disk cannot be recovered. To keep the system responsive and to make high speed sampling efficient, Spike2 buffers data in memory as much as it can. The operating system also buffers data in memory before writing it to disk. However, with data in memory, a power loss or system crash will lose data.

If you use the Flush data to disk option in the sampling configuration, you can protect yourself against disaster at the cost of some loss of performance. If you do not use the flush data option you increase the risk of data loss if your system goes down.

Batch Processing

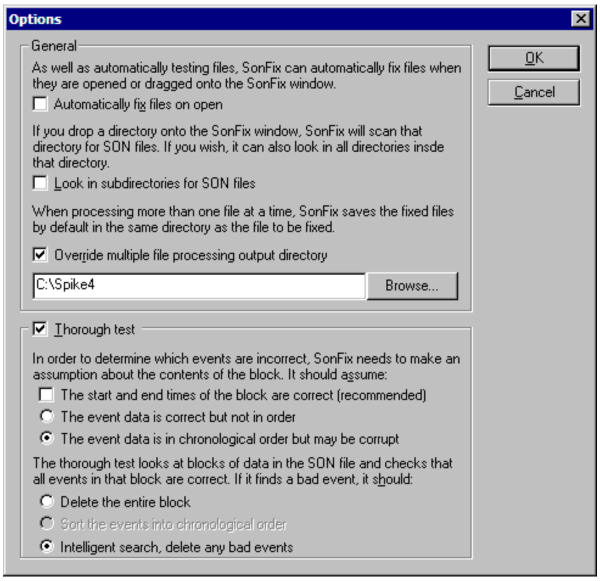
There are two ways to test or fix more than one file at a time:

1. Use the `Open File` button on the main window to show a standard File Open dialog box in which you can select as many SON files as you wish to test or fix.
2. Drag files from Windows Explorer or My Computer and drop them on the SonFix window is the other way of testing or fixing several files at the same time. You can also drag a directory onto the `SonFix` window, and it will test or fix all the SON files (files with a `.smr` extension) in that directory, and optionally all the SON files in any sub-directories (this setting can be changed from the Options window).

If you want files to be fixed automatically, you can specify this in the Options window. Fixed files have `Fixed` added on to the start of the file name. The fixed files will, by default, be placed in the same directory as the damaged files, but this can be changed from the Options window.

After `SonFix` finishes testing and optionally fixing your data files, it displays a list of the files that needed fixing, and a summary of the number of files tested and fixed.

SonFix Options



To customise `SonFix`, select `Options` from the main `SonFix` window. As well as the options described above, there is also a section in the options dialog box entitled `Thorough Test`. If selected, the thorough test is performed on files after the main test, and checks all events in the data file to make sure that they are in the correct order. If they are not, it is able to fix them in one of three ways:

- by deleting the block containing the corrupt events (a good idea for heavily damaged files where you care more about the waveform data than event or marker data)
- by sorting the events into numerical order (for example for a file containing RealMark, WaveMark or TextMark channels where you care more about the data they contain than the times they are stored at)
- by doing an intelligent search for bad events and deleting the minimum number of events to leave the data in order. This is usually the best option.

Manual Fix

The Manual Fix button leads to a new dialog in which you can inspect the lists of used blocks, deleted blocks and lost blocks for each channel. You can move blocks between these three lists for each channel. Unless you are certain that you know what you are doing it is best to use Auto Fix.

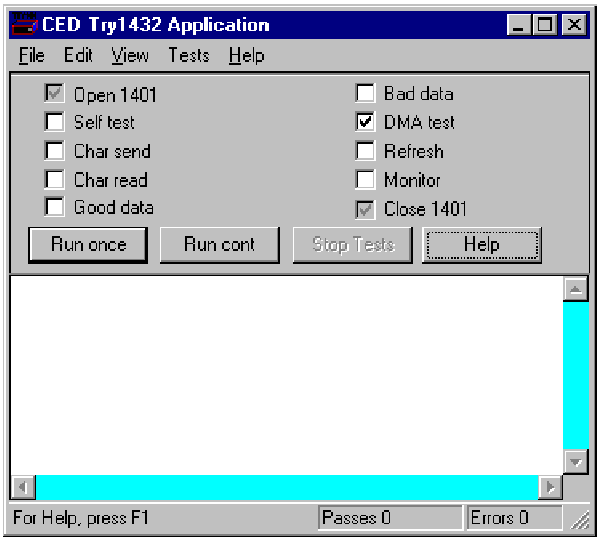
Try1401 test program

If you are having problems sampling data, and you suspect that the problem lies with the 1401, interface card or device driver, then it is well worth running the Try1401 program, found in the same folder as Spike2 (the program file is called `Try1432.exe` on Windows systems).

The Try1401 program exercises the 1401 in much the same way as Spike2 does, but it also checks every byte of data transferred and reports errors in a way that is useful to CED engineers, particularly when there are data transfer problems. Unfortunately, the sternest test of data transfer we know of is the Spike2 program, so it is possible for Try1401 to pass a system that fails in Spike2 (but this is most unusual).

The Try1401 program can also run the 1401 self-test and decode any errors. These can vary from simple problems with the calibration of the 1401 inputs and outputs (usually not a serious problem), to serious internal errors associated with damaged components. You should select the self-test option if the 1401 flashes the Test LED on power up when no other external equipment is connected to the 1401 inputs.

Running Try1401



Double click the program icon to start the program. You have a choice of test options and of running once or continuously. Errors are written to the text window, and can be copied, cut and pasted. If a single pass of the test doesn't show a problem, run the test continuously and leave it for several hours.

Most of the tests are to check the data path between the host computer and the 1401. If you have a problem with your 1401, the CED engineer who helps you to sort it out will most likely ask you to run this program and email the results. For a confidence check, the only option that is needed is the DMA test. The remaining tests help CED engineers to narrow down problems. The individual options are:

Self test

This runs the internal 1401 self test and interprets the results. Please remove all connections from the 1401 except the power cord and the data interface cable as other connections can cause self-test failures.

Char send

This checks the general data path from the host computer to the 1401. If you have a USB connection, problems indicate some sort of installation failure. For the other interface cards data corruption problems are rare, and indicate either a damaged data cable, or bent pins on the 1401 or the host interface card. A timeout error in this test usually indicates an interrupt problem.

Char read

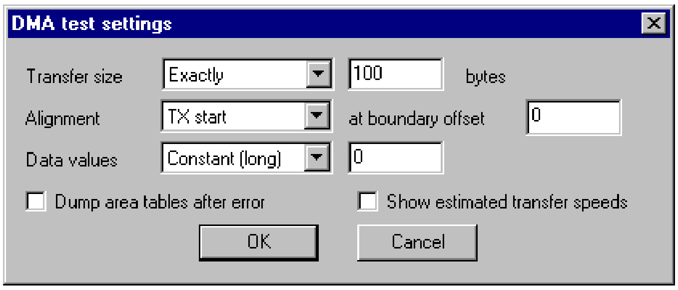
This is similar to *Char send*. It checks the general data path with an emphasis on reading back data.

Good data/Bad data

These are further variations on the *Char send* and *Char read* tests. You can usually skip these tests unless a CED engineer asks you to run them.

DMA test

This test checks out high-speed block data transfer between the 1401 and your computer. There is no point running this test if the *Char send* or *Char read* tests have failed. This test transfers data blocks of various sizes and alignments and checks that the data transferred correctly. Unfortunately, Spike2 is the sternest test of data transfers that we know of, but this test will usually detect any transfer problems. There are additional options in the Tests menu DMA settings… command:



You will not normally need to set any of the values in this dialog unless a CED engineer needs additional data to diagnose a data transfer problem. The fields in the dialog are:

Transfer size can be set to: Unconstrained, Exactly, Greater than or Less than. For all sizes except Unconstrained you must provide a byte count. For all except Exactly mode, Spike2 chooses random sizes within the constraint you have imposed.

The data for transfer is held in memory. This memory lies in a continuous block of virtual address space. However, the physical memory that makes up this address space may be mapped anywhere in computer memory. This usually means that a data transfer is broken up into sub-blocks of contiguous physical memory. Alignment relates to the position of data blocks relative to starts and ends of these sub-blocks. This field only applies to ISA and PCI interface cards using DMA transfers; we have no knowledge of alignment for the USB interface. You can choose from Unconstrained, TX start, TX end, RX start and RX end. TX = transmit to 1401, RX = receive from 1401. The at boundary offset field sets the relative position of the transfer start or end to sub-block boundaries. You can set from –4095 to +4095 (but useful values are usually in the range ± 100.

Data values can be set to be Random, or you can choose to set values based on bytes, words (2 bytes) or longs (4 bytes) and the values can be a constant, an upward ramp or a downward ramp.

Check the Dump area tables after error box to print a lot of extra info about any failing DMA transfer. This can help a CED engineer to diagnose a fault.

Show estimated transfer speeds prints the approximate number of kB per second for each transfer.

File menu

There are options in the File menu to update the Power1401 flash memory. These are described in the Power1401 Owner's manual. There is also the 1401 info… command. This prints out information about the 1401 device driver and the 1401, for example:

```
1401 type        = Micro1401
Monitor revision is 20.14
1 megabyte base memory
Micro1401 main card is 2501-01 C-211
Block transfers use DMA, multiple transfer areas
ADC channel sequencer is FIFO at 4 MHz
Supports up to 256 channels, 3uS ADC block
No extra ROM in spare slot
```

CED file formats

CED uses two data filing systems. The SON system is used by Spike2 for DOS, Windows and the Macintosh, the VS system and CHART and is well suited to a multi-channel asynchronous data stream with a mixture of events, markers and waveform data with data access keyed by the precise time of the data. Full documentation of the library as a PDF file is included with the Spike2 distribution. To get a copy, select Custom install during Spike2 version 4 installation and then check the Extra Documentation box. The additional documentation is copied to `ExtraDoc` in the folder you select for Spike2.

The CFS system is used by Signal, SIGAVG, Patch and VClamp, CHART, MassRAM, EEG and EEGER and is more suited to synchronous episodic waveform data capture, and can handle a wide range of data types. CED has placed CFS in the public domain to encourage its widespread use. Contact CED for more information.

`C2S.EXE` and `S2C.EXE` are DOS programs that convert between the two formats so Spike2 can exchange data with applications that use CFS.

Spike2 version 4 includes routines to import a wide range of data files, including CFS.

C2S - CFS to SON file conversion

`C2S.EXE` transfers data from CFS files to an equivalent SON file. The data sections in the source CFS file are converted into a single SON file with multiple data sections placed end-to-end, separated by a small time gap with the start and end of each section marked on the keyboard channel (channel 31) with marker codes 0 and 1. The data types accepted are equal-spaced channels containing 16 bit integer data and matrix data channels in a CFS marker format which are converted into Spike2 waveform data and marker data respectively. To use the program:

```
C2S source {dest} {-txxx}
```

`source` Specifies the source CFS file with default extension of .CFS

`dest` Specifies the destination SON file, default extension `.SMR`. If you omit `dest` the `source` name is used with the `.SMR` extension.

`-t` In CFS files, waveform channels are sampled at any rate whatsoever. In Spike2 files, waveform channels are sampled at a multiple of some fundamental interval. The `-t` option specifies the fundamental timing interval in µs. If the option is omitted C2S calculates the interval. You would use this option if you wished to work at a higher time resolution in the file than that implied by the waveform sampling rates.

S2C - SON to CFS file conversion

S2C.EXE converts Spike2 data to an equivalent CFS file. Each section of contiguous data is converted into a CFS data section, with the relative times of the various sections preserved. The data types accepted are waveform data and up to two marker channels. The first marker channel, if it appears to act as a section marker as used by CHART and Spike2, is used to indicate the start and end of the data sections. WaveMark channels are translated into marker channels. To use the program:

```
S2C source {dest}
```

`source` specifies the source SON file with default extension of .SMR

`dest` specifies the destination CFS file, default extension `.CFS`. If omitted the source name will be used with `.CFS` extension

MSF Modify Son File

`MSF.EXE` is a DOS utility program that reads one or more Spike2 files and creates a new, modified file. The modifications range from data extraction and channel deletion to waveform rectification, digital filtering and file amalgamation. Some of these features, such as digital filtering, already exist in Spike2 version 4.

Command line options

The MSF program is command line driven; the information required to use it is supplied on the DOS command line. The command line is (items in curly brackets are optional):

```
MSF source {sourc2} dest -op {-ttimes} {-cchans} {-cchans2}
```

`source` The source Spike2 file for the data.

`sourc2` This is only used in the amalgamate operation, and specifies the second source file for the amalgamate operation.

`dest` The output SON file that will hold the result of the MSF operation.

`-op` The operation to be carried out. `op` is a character defining the operation, plus any additional information:

A	Amalgamate channels from `source` and `sourc2`
D	Delete listed channels, copy the remainder
F	Filter waveform channels and optionally down sample
H	High pass filter (AC couple) with optional time constant
K	Recalibrate waveform channels
M	Make WaveMark channel using event channel as trigger
R	Rectify waveform channels
S	Stretch the timing (slow down) a file by a factor
U	Unstretch timing (speed up) a file by a factor
X	Extract data from source (the default operation)

`-ttimes` A list of time sections separated by commas: `-t10-40,100-200` selects two time sections, one from 10 to 40 seconds, and one from 100 to 200 seconds. A time section specified as `10-` selects data from 10 seconds to the end of the file. Only data occurring within these time sections is copied or processed. If the time specifier is omitted, the full time range of the source file is used.

`-cchans` A list of source file channels: `-c1,3,5,18` specifies channels 1, 3, 5 and 18. If no list is supplied all channels are selected. The `-cchans2` option specifies a separate set of channels in the `sourc2` file for the amalgamate operation.

Amalgamate files

The `-A` option copies specified channels from two source files to a destination file. The -t option limits the time range to copy. For example:

```
msf file1 file2 dest -a -c1,2,3 -c17,18
```

copies channels 1, 2 and 3 from `file1.smr`, plus channels 17 and 18 from `file2.smr` to `dest.smr`, using the full time range in each source file. The two channel specifiers select channels from the first and second source file. They are normally given the same channel number in the destination as they had in the source file. If a duplication of the channel number occurs, then the channel from the second source file is renumbered to use a free channel number. Marker data is forced to use channel numbers above 15.

Delete channels

The `-D` option deletes specified channels from `source` and copies the rest, unchanged, into `dest`. The `-t` option, if supplied, limits the time range to be copied. For example:

```
msf file1 dest -d -c1,2,3 -t0-100
```

copies all of the channels except channels 1, 2 and 3 from `file1.smr` to `dest.smr`, using only the data from the time range 0 to 100 seconds. The delete option requires a channel list. The Spike2 delete channel command marks a channel as deleted but leaves the disk space used; this command completely removes the channel in the destination file.

Digital filter waveform

The `-Fname{.txt}{,n}` option copies all channels from `source` to `dest` and filters specified waveforms with a Finite Impulse Response filter. The `-f` option is followed by the name of a text file holding filter coefficients. If no file extension is given, `.txt` is assumed. The output file can also be a down-sampled version of the source if the `,n` optional parameter is used (meaning output every n^{th} point). If `n` is omitted, all points are output. The `-t` option, if supplied, limits the time range to copy and filter. For example:

```
msf file1 dest -fmycoeff,4 -c1,2,3 -t10-50
```

copies all of the channels in `file1.smr` to `dest.smr`, using the full time range and filtering waveform data in channels 1, 2 and 3 using the coefficient data read from the file `mycoeff.txt`. The sampling rate of the output is reduced by a factor of 4. If any of these specified channels do not hold waveform data then they are copied unchanged.

This option is for readers who are familiar with digital filters and have access to software (such as FIRMAKE, available from CED) to calculate filter coefficients.

Filter coefficient file

The filter coefficient file is a text file. Lines in the filter coefficient file that start with a semicolon are comments and are ignored. It has the following format:

Line	Contents
1	The gain factor for the filter. This is a real number that is used to rescale the data after the filter operation, normally set to 1.0. The scale factor in the Spike2 file for the channel is divided by this number. It has no other effect.
2	The units for the filtered data. If non-blank, this will be used to set the units for all filtered channels.
3	The channel title for the filtered data. If non-blank, this will be used to set the channel title for all filtered channels.
4	The channel comment for the filtered data. If non-blank, this will be used to set the channel comment for all filtered channels.
5	The number of coefficients to follow.
6	The first coefficient, a real number in the range 1.0 to -1.0.
7	The second coefficient.
5+n	The n^{th} coefficient.

Example filter files

We supply example filter coefficient files with the Spike2 system in the `filter` directory. We have implemented low pass, high pass and notch filters and differentiators.

The low pass and high pass filters have two frequency bands, the pass band and the stop band. In the pass band (a band is a range of frequencies) the signal passes through the filter with little or no change in amplitude (but there can be changes in phase). In the stop band the signal is attenuated (the filters provided have a minimum attenuation of 40 dB in the stop band, which is a factor of 100). Between the pass band and the stop band is a transition region where the signal changes between the zero attenuation of the pass band and the 40 dB attenuation of the stop band.

The notch filters pass all frequencies except a narrow range around a centre frequency. These filters are sometimes used to reduce mains interference.

We specify the frequency ranges to be used to define the bands as a fraction of the waveform channel sampling rate. You can find the sampling rate of a channel in a file by using the File menu Read Configuration option, then opening the Sample menu Sampling configuration dialog. The file names used for the coefficient files have the form: `Lpppsss.TXT`, `Hsssppp.TXT` and `Nssswww.TXT`. The initial character indicates the type of the filter (L = Low pass, H = High pass, N = Notch). The remaining letters indicate the band edges of the filters in thousandths of the sampling rate. There cannot be frequencies above 0.5 of the sampling rate present in a data file. For example:

`L150200.TXT` A low pass filter with a pass band from 0 to 0.150 of the sampling rate and a stop band from 0.200 to 0.5 of the sampling rate.

`H050100.TXT` A high pass filter which rejects data from DC to 0.050 of the sampling rate and passes data from 0.100 of the sampling rate upwards.

`N200050.TXT` A notch filter centred at 0.200 of the sampling rate with a width of 0.050 of the sampling rate.

The choice of 40 dB as the attenuation in the stop band is entirely arbitrary, but should be useful for general purpose applications. Much higher attenuation can be achieved at the cost of more coefficients and slower filtering performance.

There are also coefficient files for differentiating a signal. These files start with the letter D and are followed by a number indicating the number of data points used to generate each output point. So `D50.TXT` indicates a differentiator with 50 coefficients.

NOTCH filter generation

The program `NOTCH.EXE` (in the `FILTERS` directory) calculates coefficient files for notch filters. A notch filter is normally used to remove single unwanted frequencies (these are usually due to mains interference). The input to the program is the notch frequency as a fraction of the sampling rate, and the name of the output file:

```
NOTCH freq fname
```

`freq` The fraction of the sampling frequency in the range 0.001 to 0.5. Values outside this range cause the program to stop.

`fname` The name of the output file to hold the coefficients.

As an example, assume that we have sampled some data at 1000 Hz on channel 1 of a file and that this data is polluted by 50 Hz mains interference (or 60 Hz if you are in America or parts of Japan). The fraction of the sampling rate required is 50/1000 (60/1000) which is 0.050 (0.060). To attenuate this frequency:

```
NOTCH 0.05 n050.txt
MSF datafile newfile -fn050 -c1
```

Assuming that the old data was in `DATAFILE.SMR`, the result is now in the new file `NEWFILE.SMR`.

The program produces between 101 and 159 coefficients and the width of the notch is of order 1% of the sampling rate. Thus in our example above, the notch will affect frequencies in the range 40-60 Hz (50-70 Hz).

If mains interference is your problem, use the power spectrum analysis to discover if you are also suffering from 3rd and 5th harmonics of the mains frequency. If you are you may find that creating additional notch filters at these frequencies helps.

High pass filter

The `-H{tc}` option applies a high pass filter with an optional time constant (sometimes known as AC coupling) to waveform channels specified by the `-c` option, or to all waveform channels if `-c` is not used. If no time constant is supplied, a one second time constant is used. The results are written to the new file. For example:

```
msf waves new -h0.5 -c1-3
```

This copies the contents of file `wave.smr` to `new.smr` and applies a high pass filter with a 0.5 second time constant to waveform channels 1, 2 and 3.

Calibrate waveform

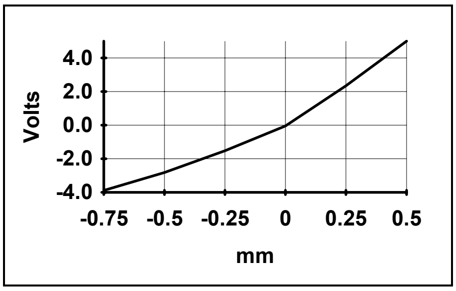
The `-Kfile{.txt}` option calibrates (linearises) waveform channels selected by the -c option based on a calibration curve in a text file. The text file name is given after the `-K` with no intervening spaces. If no extension is given for the file name, `.txt` is assumed.

The calibration file specifies the units and range of values after calibration, plus pairs of input values and their corresponding output values. The input and output values specify the calibration curve to use. Linear interpolation is used to calculate outputs for input values between the calibration points. Input values that lie outside the calibration are limited to the ends of the calibration. The calibration file holds the following information:

Line	Contents
1	The calibration file title. It can be blank, but it must be present.
2	Set blank to calibrate all channels in the -c list. If non-blank, it holds the input units. Only calibrate channels with matching units (ignoring case).
3	The units for the calibrated data. If blank, the original units are retained.
4	Two numbers: the first is the minimum output value, the second is the maximum. This sets the mapping between the integer numbers in the file and the values displayed. Setting too wide a range can result in loss of accuracy.
5	Channel comment for the calibrated data. If blank, the original is used.
6	The number of calibration points that follow.
7	The first calibration point as two numbers separated by a space or spaces. The first number is the input value, the second is the output value.
8	The second calibration point.
6+n	The n^{th} calibration point.

Lines in the file that start with a semicolon are comments and are ignored. An example may make this clearer. Consider a rather non-linear strain gauge whose output we have sampled as Volts. We also have a table of Volts and a displacement in mm:

Volts	-3.88	-2.81	-1.52	-0.043	2.351	5.0
mm	-0.75	-0.50	-0.25	0.00	0.25	0.50



We want to convert the data from Volts, into millimetres. We know that the output range will only span the range of the calibration, so we can set this to -0.75 to 0.50. A suitable calibration file, assuming that the input units were set to Volt, would be as follows:

```
;Strain gauge calibrate
gauge 11a23 calibration
Volt
mm
-0.75 0.5
Voltage converted to mm
6
-3.88  -0.75
-2.81  -0.5
-1.52  -0.25
-0.043  0.0
2.351  0.25
5.00  0.50
```

Any input values below -3.88 Volts are treated as though they were -0.75 mm. Similarly; inputs above 5.00 Volts are limited to 0.50 mm. Both source and destination units are specified, so only channels with matching units are calibrated.

If there are problems in the format or contents of the file, MSF will not use it. Do not set input values that exceed the possible input data range or calibration values that exceed the output range set in line 4.

Make WaveMark channels

The `-Me,p,b` option converts waveform channels into WaveMark channels. The command line tells the program which channel to extract the times from, how many data points to use for each WaveMark event, and how many pre-trigger points are wanted. The characters `e`, `p` and `b` stand for:

`e` The event channel number to use as the trigger time for each WaveMark event. This channel must exist, and must be an existing event, marker or WaveMark channel.

`p` The number of waveform points required in each WaveMark event. This must be in the range 1 to 2000.

`b` The number of pre-trigger data points in the range 0 to `p`-1.

If you have given a channel list, all the waveform channels in the list are converted. If you do not give a list, all waveform channels in the file are converted. Any channels that are not converted are copied to the destination file.

```
msf waves wavemark -m3,32,12 -c1
```

This example changes channel 1 (which must be a waveform channel) into a WaveMark channel. The trigger points are held in channel 3 as events or markers. The output is a WaveMark channel with 32 points per trigger, with 12 points before the trigger. All other channels are copied.

Rectify waveforms

The `-R` option copies all channels from `source` into `dest`, rectifying specified waveform channels. The `-t` option, if used, limits the time range to copy or rectify. For example:

```
msf file1 dest -r -c1,2,3
```

copies all of the channels in `file1.smr` to `dest.smr`, using the full time range and rectifying waveform data in channels 1, 2 and 3. If any of these channels do not hold waveform data then they are copied unchanged.

Time compress or expand

The `-Sfactor` and `-Ufactor` options stretch and unstretch a file. Unlike the other options, if no destination file is given the source file is patched in place. These options change the meaning of the basic clock tick that is used as the basis of all timing in the file. The `factor` is an integer number that sets the ratio between the original time and the new time for the file. The use of `-c` or `-t` is not meaningful in this option.

The stretch option is normally used to slow down high-speed replay from a tape recorder. For example, tapes are often used to record EEG data from ambulatory subjects over many hours. Suppose there were four data channels to sample at 100 Hz each, a total of 400 Hz. Spike2 can sample at many times this rate. If the tape recorder could replay at 64 times real time and Spike2 was set to sample at 6400 Hz per channel, 24 hours of data could be transcribed into a Spike2 file in 23 minutes (as long as you had 70 MB of disk space free). The final step would be to run:

```
msf data -s64
```

to change the file time base back to real time. There is a limit to how much you can stretch a time base, depending on the current value of the microseconds per unit time set in the file. The new value of microseconds per time unit (which is simply the old value multiplied by the factor) must be less than 32768. Because of this, and also because this value sets the time resolution of the file, it is a good idea to set this value to the smallest value possible (compatible with the length of recording desired) when sampling the data.

The unstretch option does the reverse of the stretch option, and is rarely used. There is the limitation that the current microseconds per time unit of the file must be divisible by the scaling factor with no remainder.

Extract data

The `-X` option extracts specified channels from `source` and copies them, unchanged, into `dest`. The `-t` option, if supplied, limits the time range for the data to copy. For example:

```
msf file1 dest -x -c1,2,3
```

copies channels 1, 2 and 3 only from `file1.smr` to `dest.smr`, using the full time range. The File menu Export option in Spike2 for Windows has a more powerful version of this command.

Index

—D—

—T—