Signal for Windows

Version 4

Version 4.00	October 2007
Version 4.01	April 2008
Version 4.02	May 2008
Version 4.03	October 2008
Version 4.04	December 2008
Version 4.05	March 2009
Version 4.06	November 2009
Version 4.07	January 2010

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

Curve fitting procedures are based on routines in Numerical Recipes: The Art of Scientific Computing, published by Cambridge University Press and are used by permission.

Table of Contents

1 Signal for Windows

Introduction

The Signal software running under Windows together with one of the CED 1401 family of interfaces gives a PC the power to capture and analyse multi-channel waveform and keyboard time marker data.

Signal is designed to let you manipulate your data using the familiar Windows idioms. You can arrange the windows to display the data within them to best advantage and copy and paste the results to other applications. Alternatively, you can obtain printer hard copy directly from the application. When you close a data file, Signal saves the screen format and analysis window setup associated with it. When you open a file, Signal 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 frame by frame automated analyses such as waveform averaging and power spectrum. More ambitious users can further automate both data capture and analysis with scripts. The script language is described in *The Signal script language* manual.

New features in version 4

Version 4 of Signal is completely compatible with earlier versions; it will read data files and sampling configurations created by version 2 or 3 without problems. Scripts that ran in version 3 should work without modification (with a very few exceptions). The main new features in Version 4 are:

- Gap-free sampling records sweeps continuously with no missing points.
- Variable number of points can be sampled in different sweeps to match experimenters' requirements.
- New script editor with options including intelligent hiding of blocks of script, auto formatting, auto complete and call tips.
- Script code extended to share and re-use include files.
- Multiple or averaged measurements, generated using active cursor iteration across data frames, can be placed in an XY display.
- Single channel (SCAN) analysis provides incremental processing of single channel data to produce an idealised trace by reverse convolution fitting of the amplifier's step response function with the raw data
- Interactive IIR filtering and improvements to the FIR filtering dialog.
- Extended multiple states options to automate states sequencing.
- Directly control Magstim transcranial magnetic stimulator during data acquisition, including adjusting stimulator amplitude and timing and checks on stimulator health.
- Enhanced Power-Spectrum analysis, up to 16384 point FFT and windowing.
- Direct access to the 1401 data acquisition system using the script language.
- Fill areas within an XY plot.
- User defined keyboard shortcuts for use with single channel analysis.
- Extended header and footer customisation for printed output.
- Automatic generation of file names based upon the current date.
- Cursor iteration and feature detection through data frames.

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 releases of Signal version 4 from our web site `www.ced.co.uk` as they become available.

Hardware required

The absolute minimum requirement to run the program is a Pentium II with 256 MB of memory running Windows 98SE. The more memory you have and the faster the processor, the better Signal will run. A graphics accelerator will greatly improve drawing and scrolling speeds.

To sample data, you will need a CED Power1401 mk I or II, Micro1401 mk I or II, 1401*plus* or a standard 1401. Pulse output during sampling and many other features are not available with a standard 1401, gap-free sampling and some output options are not available with a 1401*plus*. Unless the exact type of 1401 is specified, when the terms Micro1401 and Power1401 are used in this manual they refer to all versions of these types of 1401. USB operation requires Windows 98SE, Me, 2000, XP, XP64 or Vista.

Installation

Your installation disk is serialised to personalise them to you. Please do not allow others to install unlicensed copies of Signal.

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 `Signal4` on the CD-ROM, then the `disk1` folder and running `setup.exe`.

During installation

You must select a suitable drive and folder for Signal and personalise your copy with your name and organisation. You can have earlier versions of Signal on the same system as long as they are in different folders. If you have a previous version on your system, make sure you install to a different folder. The installation program copies the Signal program plus demonstration, help, tutorial and example files. It also copies and installs all required 1401 support (device drivers and control panels). In rare cases you may need to install drivers manually; the installation program will tell you if this is the case and point you at detailed instructions. Your system may require a restart after installation to get all 1401 device drivers up to date.

Custom install

To install without 1401 support, or to install extra information and documentation (which includes the latest copies of the software manuals) choose **`Custom`** installation.

After installation

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

Updating Signal

You can update your copy of Signal to the latest version 4 release free of charge from our web site: `http://www.ced.co.uk` in the `Downloads` section. You can only update a correctly installed copy of Signal version 4. There are full instructions for downloading the update on the web site, you can also sign up for email notification of updates on the web site.

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

Removing Signal

You can remove Signal from your system: open the system Control Panel, select Add/Remove Programs, select CED Signal for Windows version 4 and click Remove. This removes files installed with Signal; you will not lose data and other files that you created.

File icons

The various file types in Signal have icons so that you can easily recognise them when you minimise their windows. The icons will automatically be used for the relevant files by programs such as Windows Explorer. All the icons have a set of waveforms to remind you of the application to which they belong. The icon to the left is the Signal application icon that you double-click to launch Signal from the Signal program group.

These icons are for Signal data files. The icon on the left represents Signal CFS data files or file views. The icon on the right represents an XY data file: a saved XY view. If you double-click on one of these in Windows Explorer it will launch the Signal application (if it is not already running) and open the data file.

These icons are for text-type Signal documents and files. The icon on the left represents a text file, which can hold any textual data. The centre icon represents a Signal script file. Script files hold script programs that execute within Signal to automate analysis or to customise Signal behaviour in some way. The icon on the right represents a Signal sequencer file. These files store sequences of output pulses for use during sampling.

These icons are for other files created by Signal. The icon on the left represents a Signal sampling configuration, while the one on the right represents Signal resources. A resources file is a file with the same name as a data or XY file (but with a .SGR filename extension) that holds extra information used with the data file. This information can be anything from how the data is to be drawn to single-channel idealised trace data.

Using this manual

The first section of this manual introduces you to Signal by suggesting a few tasks you might undertake to familiarise yourself with the system. We have supplied an example data file for you to experiment with, so there is no need to have your own data available at this time. A second chapter introduces sampling new data. Subsequent chapters describe using pulse outputs during sampling and multiple frame states.

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

We do not explain standard Windows 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 manual *The Signal script language*. The online help system duplicates all the information in the manuals and is often the fastest way to look up a topic.

Forthcoming attractions

This manual covers version 4.00 of the Signal software. Later versions of the software will have further enhancements which will be described in the revision history found by selecting the Index from the Help menu.

Direct access to the raw data

Some users may wish to write their own applications that manipulate Signal data files directly. A C library: *The CFS library* is available from CED. The library includes all functions necessary to read or write Signal data files from either DOS or Windows programs. This library is available, along with complete documentation in PDF format, from the CED web site (http://www.ced.co.uk).

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 Signal… 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.

2 Getting started

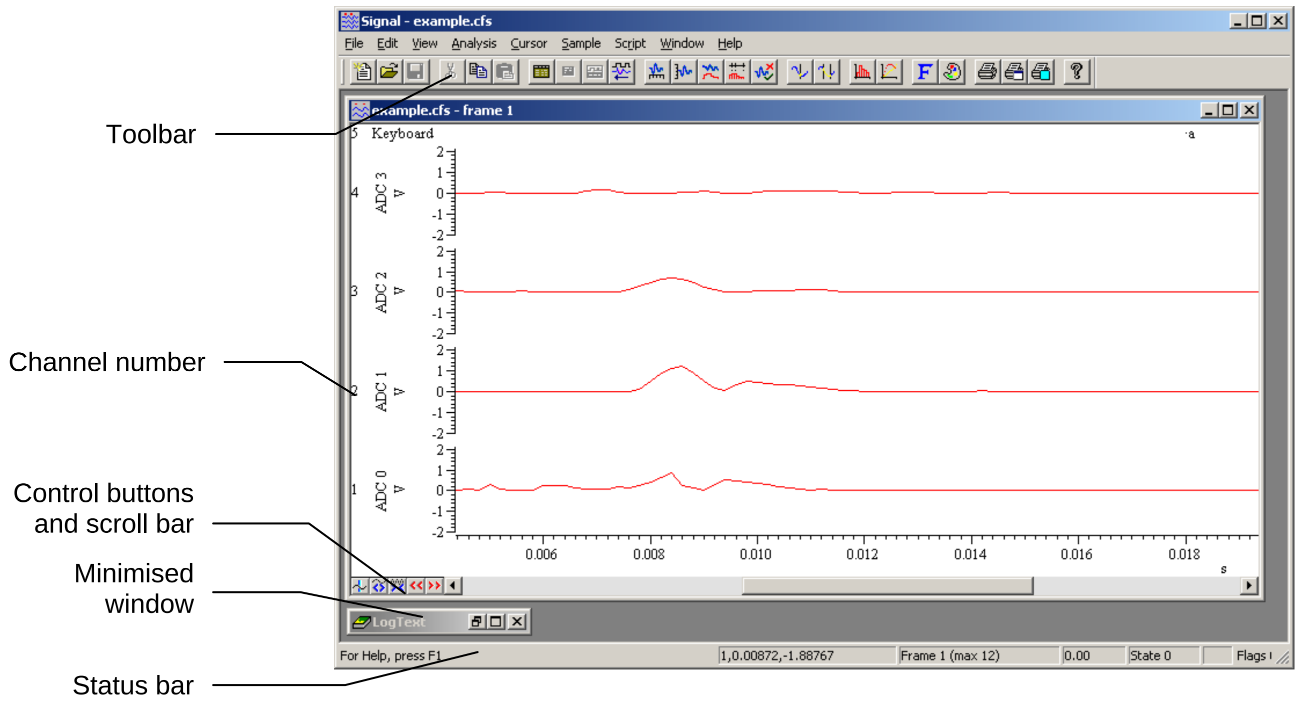
Introduction

In this section you will open a Signal data file, manipulate the contents and familiarise yourself with the basic display and analysis 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 Signal files. We use a sample file, `example.cfs`. The Help menu Index command *Getting started* topic duplicates this chapter. Once you have started Signal you may prefer to follow the remainder of the chapter from the on-screen help.

☞ **From the Start button choose Programs, then Signal, then example.**



You could also have started Signal by selecting the Signal icon and then opened `example.cfs` with the File menu Open command. Signal displays the file as it was last saved (using information in the file `example.sgr`, if it can be found). The picture shows the state as shipped. You are looking at the raw data in the file, we call this a *file view*. This view displays a single *frame* of the data; the axis at the bottom is in seconds.

There are five *data channels* displayed in the window. Channels 1 to 4 hold waveform information and channel 5 holds markers logged from the keyboard. Signal arranges channels so that the lowest numbered ones are at the top by default, this can be changed using the preferences dialog available through the Edit menu.

Below the file view window there is a minimised window with the title LogText. This is the log view; a text document that is always open in Signal. If you try to close the log view, it is only hidden and can be re-displayed using the Window menu Show command.

Selecting channels

Click on the channel number to select a particular channel. Signal highlights the channel number when it is selected. Hold down the `Shift` key and click on a channel to select all visible channels between it and the last selection. Hold down `Ctrl` to select discontinuous channels; clicking on a channel number will toggle its state between selected and de-selected. Click on the blank rectangular area below the y axes and to the left of the x axis to de-select all channels. Many commands and operations can operate on the selected channels (for example y axis optimisation).

Toolbar and Status bar

The Toolbar is at the top of the Signal window; just below the menus, it is a shortcut to commonly used menu items. Each toolbar button matches an action from the menu. The buttons are shown in this manual with the menu items. To find out what a toolbar button does, leave the mouse pointer over the button for a few seconds.

The Status bar is at the bottom of the Signal window, it provides information about the current view. The status bar holds a number of panes plus an open area on the left. This leftmost portion shows prompts from menu items; as you move the mouse pointer over the menus, the panes each show a particular item of information. If the space available is too small for all of the panes, panes disappear starting at the right hand side. From the left, the status bar panes are:

Cur Pos	If the mouse pointer is over part of a data or XY view that has axes this pane displays the pointer X and Y positions, the first figure is the channel number from which the Y value is taken. For a text-type view it displays the current text cursor position in lines and columns.
Frame	This pane shows the current frame number for the current view and the maximum frame number in the view. If the current view is not a data document then this pane is blank. See below for a discussion on frames.
Start	This pane, adjacent to the frame number, shows the absolute start time for the frame. For files collected by Signal version 1.00, this is always zero.
State	This pane shows the state code for the current frame in the current view as a decimal number. It is blank if the current view is not a data document.
Tag	If the current view is a data document and the current frame is tagged (described below), this pane holds TAG, otherwise it is blank.
Flags	This pane shows the flags for the current frame in the current view as an 8 digit hexadecimal number (hexadecimal format makes the individual flag states visible), each digit shows the state of four flags with the highest on the left. This pane is blank if the current view is not a data document view.
Caps	If the keyboard Caps lock is on, this pane displays the text CAPS.
Num	If the keyboard Num lock is on, this pane displays the text NUM.
Record	This pane displays REC if Signal is recording user actions into a script.

You can hide and show the toolbar and status bar by using the View menu. You can also drag the toolbar and “stick” it to any of the 4 sides of the application window. This is known as docking the toolbar.

A discussion about frames

Frames are a central concept within the Signal software. A CFS file or Signal data document consists of a number of sections or frames, each frame corresponds to a sweep of data. A data document view normally displays data from a single frame at a time, this is the current frame for that view. You can have duplicate views of the same data document, each view can have a different current frame so you can examine separate parts of the file simultaneously.

Each frame in a Signal data document has the same number and type of channels, but may have varying frame start and end times. Each frame holds channel data from the various channels in the source data file. Usually, all of the waveform channels will have the same number of data points, while the number of markers can vary. In addition to this channel data there are a number of other data items attached to each frame:

Comment	a line of text of up to 72 characters that can be read or written using Signal.

State	a 32-bit number that can have any value from 0 to over 4 billion, it is intended for use in conditional analysis where each state value corresponds to a separate condition in the experiment, but may be used for any purpose. Signal can sample using multiple frame states logged from external equipment or generated internally to control 1401 outputs, see the chapter *Sampling with multiple states*.
Start	a floating-point number holding the absolute start time for the frame. This value is the time for the frame x axis zero relative to the start of sampling.
Tag	each frame is tagged or not tagged. Tagging is intended for any purpose in which specific frames need to be marked for analysis or attention.
Flags	a set of 32 flags available for any user-defined purpose. They are accessible from the Signal script language.
Variables	16 floating point numbers that can be read or written using Signal scripts. Their meaning is user defined.

The entire data in the current frame for the view is held in memory, this memory data is discarded when the view switches to a different frame. You may find that Signal's performance when handling large frames is improved by installing more memory in your computer. Changes made to the current frame must be saved before a new frame is loaded or the changes will be lost. You can write the changed data back into the file using the File menu Save command, or you can use the File:Update mode dialog to select what happens if the frame is changed while data is unsaved. Changes made to non-channel data such as the frame state or flags are always saved.

Scroll bars and buttons

If you resize the window, the same data is redrawn to fit the window. The scroll bar controls movement through the data within the current frame, while the buttons allow stepping from frame to frame, changing the x axis width and adding a cursor.

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

Click these buttons to move to the previous or next frame in the file. CFS files contain frames which hold similar data; you can use these buttons to move from one frame to another. These buttons correspond to the View menu Previous frame and Next frame commands (`PgUp` and `PgDn` keys), there is also a Goto frame command.

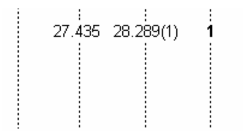
Click this button to halve the displayed x axis range (zoom in). The left hand edge of the display remains fixed. You can zoom in until the ratio between the total length of the frame and the width of one screen pixel reaches about 2 billion. In practice this means you can zoom in as far as you like. This button corresponds to the View menu Reduce View command (`Ctrl+Left`).

Click this button to double the displayed x axis range (zoom out). The left hand edge of the window does not change unless the start plus the new width exceeds the length of the frame, in which case the left edge moves back. If the new width would exceed the total length of the frame, the entire frame is displayed. This button corresponds to the View menu Enlarge View command (`Ctrl+Right`).

Click these buttons to add cursors to the display. Up to 10 cursors can be present in a window, plus up to 9 horizontal cursors. A vertical cursor is a vertical dashed line used to mark positions, a horizontal cursor marks levels in a channel. You can remove cursors by using the Cursor menu Delete command. You can add a cursor in three ways:

1. Click on the relevant button, or right click on the channel data area.
2. Use the Cursor menu New cursor command, or its shortcut `Ctrl+|`.
3. Use the shortcuts `Ctrl+0` to `Ctrl+9` to create or fetch a specific vertical cursor.

Click the cursor button so that at least one cursor is visible. Drag the cursor and observe how the mouse pointer changes. Use the Cursor menu Label mode command.



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 using the Cursor menu Label mode command. To avoid confusion between the cursor number and the cursor position, Signal draws the number in **bold** type when it appears alone, and in brackets when shown with the position.

The mouse pointer changes when it is over a cursor into one of three possible shapes to indicate the 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.

If you position the mouse pointer over the centre of the cursor label, the pointer changes to a four-headed arrow to indicate that you can drag both the label and the cursor. This can be very useful when you are preparing an image for publication and you need the cursor label to be clear of the data. If you move the pointer to one side, or hold down the shift key, the pointer changes to a two-headed vertical arrow and you can drag the label but not the cursor.

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

Controlling the display

There are many ways to use Signal to adjust or customise the display or to control the data that is displayed.

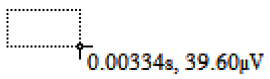
Move the mouse pointer to a waveform channel, clear of any cursor. Click and drag a rectangle round a waveform feature, then release the button.

This action zooms the display so that the area within the rectangle expands to fill the entire view. 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 channel area to show that you are about to drag a rectangle or line to magnify the data.

If you hold down the `Ctrl` key before you hold the mouse button down, the mouse pointer changes to the un-magnify symbol. If you drag a rectangle, the data in the view shrinks to cover an area the same size as the rectangle you have dragged, making this the inverse of the effect without the `Ctrl` key.

Whichever method used to scale the data, you can return to the previous display using the Edit menu Undo command or the keyboard short-cut `Ctrl+Z`. If you decide not to expand the display after starting to drag, return the mouse pointer to the original click position (making the rectangle have zero width and height). The rectangle will vanish and you can release the button without changing the display.



If you hold down the `Alt` key before you click and drag, Signal displays the size of the dragged rectangle next to the mouse pointer and does not zoom the display.

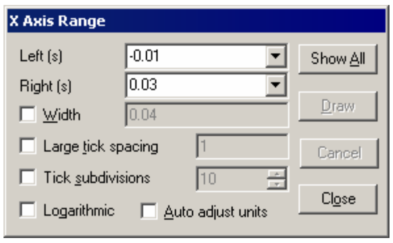
Move the mouse pointer over the x and y axes and experiment with clicking and dragging the axes. Try it with the `Ctrl` key held down.

When the cursor is over the tick marks of an axis, you can drag the axis. This maintains the current axis scaling and the y-offset changes to keep pace with the mouse pointer. You can do this with most x and y axes in Signal. 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 file view, memory view (see page 2-10), or XY view, you can drag the y axis so as to invert it. You are not allowed to invert the x axis.

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.


X Axis Range

Left (s): -0.01
Right (s): 0.03
[ ] Width: 0.04
[ ] Large tick spacing: 1
[ ] Tick subdivisions: 10
[ ] Logarithmic
[ ] Auto adjust units

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 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.

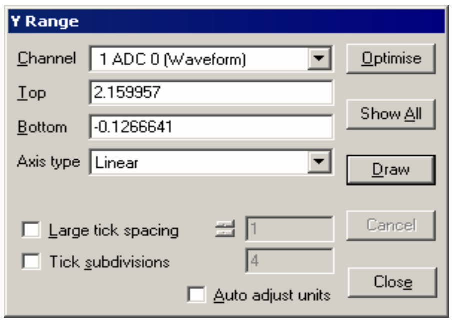
In addition to typing times, or selecting a time from the drop-down list, you can type in expressions using the maths symbols `+` (add), `-` (subtract), `*` (multiply) and `/` (divide). You can also use round brackets. For example, to display from 1 second before cursor 1 to one second past cursor 1 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. The Auto adjust units option will cause the units displayed on the axis to switch to multiples of powers of 10 in order to keep the figures sensible when zoomed well in or well out. This option affects only the axis; the units used by the cursors etc will still be the same. Checking the Logarithmic option will switch the axis from linear to logarithmic; modifying the displayed range only if it included negative values. In logarithmic mode another check box: Show powers will appear. This allows the big ticks to be labelled with powers of the big tick spacing.

Specifying times

Often in Signal you will need to specify time points within the frame. For example, specifying the X axis limits, the start and end times for file export, or the search limits for an active cursor. Signal provides a standard control that allows you to enter a time directly or to select the frame limits (Mintime() and Maxtime()), the current display limits (XLow() and XHigh()) or the position of any cursors. You can also use an offset value along with the built-in values, for example “Mintime() + 0.75” or “Cursor(1) - 0.1”. Note that any numerical values entered always use the currently selected time units.

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


Y Range

Channel: 1 ADC 0 (Waveform)

Top: 2.159957

Bottom: -0.1266641

Axis type: Linear

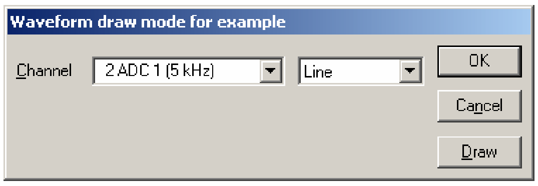
☐ Large tick spacing: 1

☐ Tick subdivisions: 4

☐ Auto adjust units

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, all channels with y axes, selected channels, or all visible channels. You can also type a list of channel numbers and ranges such as `1,4,6..10`. You can either Optimise the display, which makes sure that all the data in the window on the specified channel(s) fits in the y axis range, use Show All, which adjusts the display to the data limits if possible, or you can type in the y axis limits. You can also control the other features, as for the x axis with the addition of the option of a square root axis type.

Open the View menu Draw Mode dialog and experiment with different drawing modes for the channels.

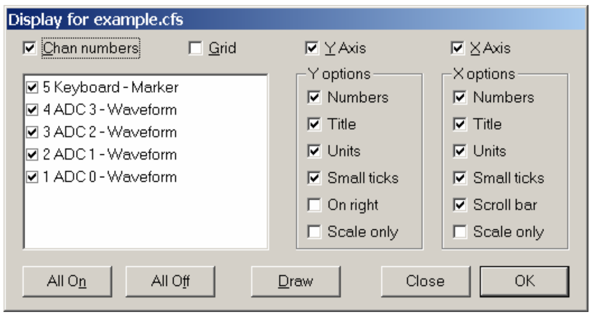

Waveform draw mode for example

Channel: 2 ADC 1 (5 kHz) — Line

Signal data files hold two basic data channel types: waveform and marker. Waveform data channels hold values that are the amplitude of the waveform at equal time intervals. Marker data channels hold the times at which something happened. If a waveform has any error information associated with it then the display of this information may also be manipulated here. There are several different ways to display waveform data (click the Draw button to cause an update without closing the dialog).

The most common draw mode for waveform data is Line mode. In Line mode successive data points are joined with straight lines. You can also select Histogram, Skyline, Dots or Cubic Spline modes. In Dots mode, you can choose large or small dots (small dots can be very difficult to see on some displays). See the *View menu* chapter for a complete description of waveform and marker draw modes.

Open the View menu Customise display dialog. Experiment with the channels, axes and grid.


Display for example.cfs

☑ Chan numbers ☐ Grid ☑ Y Axis ☑ X Axis

☑ 5 Keyboard - Marker
☑ 4 ADC 3 - Waveform
☑ 3 ADC 2 - Waveform
☑ 2 ADC 1 - Waveform
☑ 1 ADC 0 - Waveform

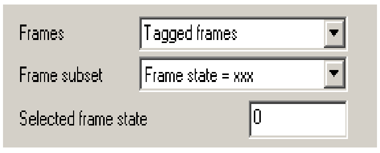
Y options: ☑ Numbers ☑ Title ☑ Units ☑ Small ticks ☐ On right ☐ Scale only

X options: ☑ Numbers ☑ Title ☑ Units ☑ Small ticks ☑ Scroll bar ☐ Scale only

This dialog sets the channels to display in your window. A Signal data file can hold up to 100 channels, so having the ability to choose the channels to display is 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 also show or 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.

Open the View menu Frame display list dialog. Specify frames 1..4 as the frame list and click OK. Use the View menu Overdraw frame list command to turn overdrawing on and off. Experiment with selecting frames and with the colour cycling display mode.



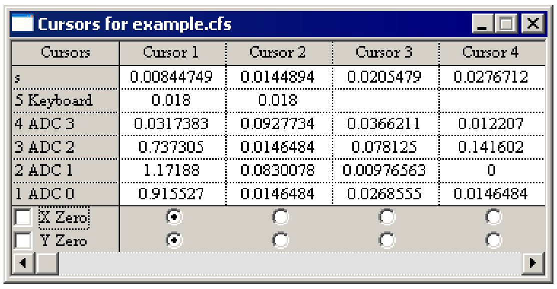
Frames	Tagged frames
Frame subset	Frame state = xxx
Selected frame state	0

The frame display list is a list of frames to show in addition to the current frame if overdraw mode is enabled. It is also possible to overdraw the buffer (a special frame held in memory). You turn overdraw mode on or off with the View menu Overdraw frame list command. In colour cycling mode, each overdrawn frame draws in a different colour, otherwise all the display list frames draw in the colour set by the Frame list traces item in the colour setup dialog. The current frame draws in its usual colour if it is not in the frame list. All the standard mechanisms for selecting frames are available, see the *General information* chapter for details of these. More information on the buffer can be found in the *Analysis menu* chapter.

Cursor measurements

You can use the cursors to take measurements at or between cursor positions.

Make sure you have some cursors in the window, then open the Cursor menu Display Y Values window. Experiment with changing cursor positions and channel display types.



Cursors for example.cfs

Cursors	Cursor 1	Cursor 2	Cursor 3	Cursor 4
s	0.00844749	0.0144894	0.0205479	0.0276712
5 Keyboard	0.018	0.018		
4 ADC 3	0.0317383	0.0927734	0.0366211	0.012207
3 ADC 2	0.737305	0.0146484	0.078125	0.141602
2 ADC 1	1.17188	0.0830078	0.00976563	0
1 ADC 0	0.915527	0.0146484	0.0268555	0.0146484
☐ X Zero	●	○	○	○
☐ Y Zero	●	○	○	○

The columns show the cursor positions and the value at the cursor positions for waveform channels. Marker channels displayed as Rate also show the value at the cursor position, marker channels in other draw modes show the time of the next marker after the cursor.

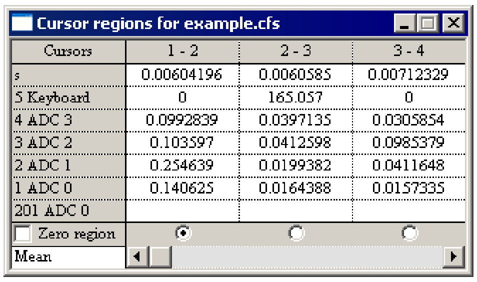
To measure the difference between cursor values, use the X zero and Y zero check-boxes. The radio buttons below each column choose the cursor to make the reference. The values for the reference cursor are shown unchanged; the values for the other cursors have the value at the reference cursor subtracted. You can use this feature to show how data values have changed from a reference point.

If you move the cursors, change frame or change the channel display mode, the values in the window update to reflect the change of position. Likewise, if you show or hide data channels in the display, the cursor window display changes to match.

You can select fields in this window 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 `Ctrl` key and click at the top or left hand edge for non-contiguous selection of rows or columns.

You can also print, copy to the clipboard; change the font or copy to the log window using the right mouse button menu.

Now open the Cursor menu Cursor Regions window. Experiment with changing cursor positions and measurement modes.



Cursor regions for example.cfs

Cursors	1 - 2	2 - 3	3 - 4
s	0.00604196	0.0060585	0.00712329
5 Keyboard	0	165.057	0
4 ADC 3	0.0992839	0.0397135	0.0305854
3 ADC 2	0.103597	0.0412598	0.0985379
2 ADC 1	0.254639	0.0199382	0.0411648
1 ADC 0	0.140625	0.0164388	0.0157335
201 ADC 0			
☐ Zero region	●	○	○
Mean			

The regions window looks at the data values between cursors. There are many measurement modes including Area, Mean, Slope, Peak, Sum, Modulus, Maximum, Minimum, Amplitude, SD and RMS. You can select the mode with the popup menu at the bottom left corner of the window. Click on the rectangle showing Mean to see the menu. If you want to make measurements relative to one of the regions, check the Zero region box and choose a reference region with the radio buttons.

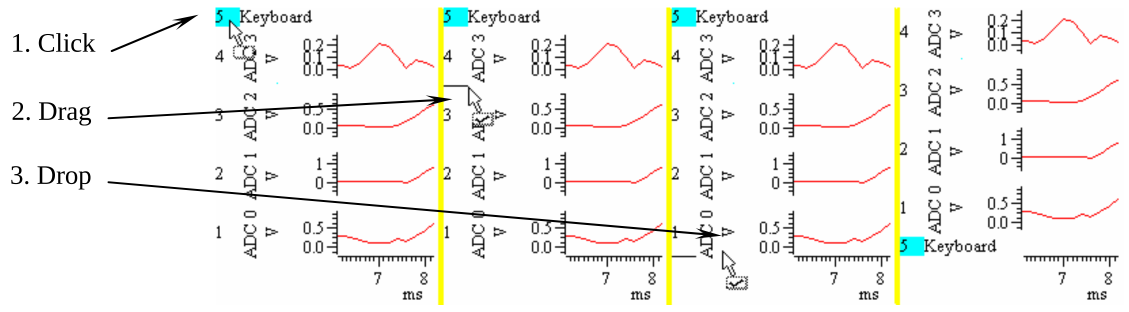
For a waveform channel or markers as rate, Area is the area between the waveform trace and the line joining the intersection points between the cursors and the trace, Mean is the mean level of the signal, Slope is the gradient of the least-squares best fit line to the data, Area/0 (read this as 'area over zero') is the area between the waveform trace and the y zero level, Sum is the sum of all data points and Modulus is the area over zero, but with all amplitudes considered positive - the 'rectified area'.

For a marker channel in other (not Rate) draw modes, Sum is the number of markers in the region. Mean is the count of markers divided by the width of the region. Slope has no meaning for a marker channel, neither does Area, Area/0, Modulus or the others.

Channel arrangement

You can control the order in which channels are displayed or overlay them on top of one another.

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



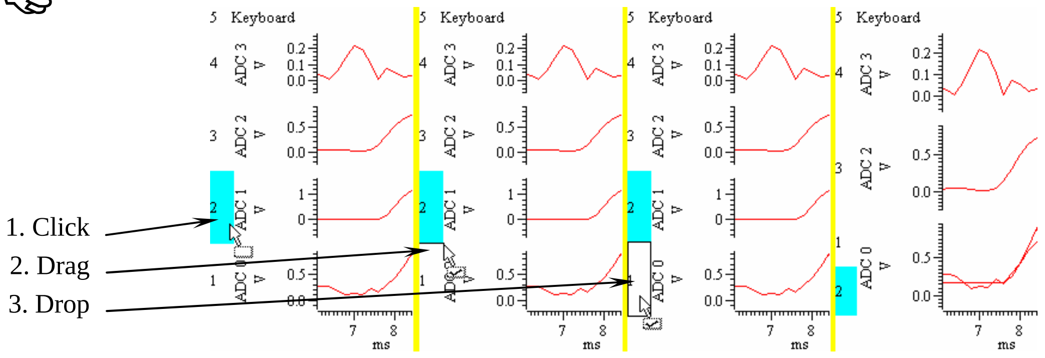
As the mouse pointer passes over each channel, a horizontal line appears above or below the channel. This horizontal line shows where the selected channel will be dropped. Drag until you have a horizontal line below channel 1 and release the mouse button. The Keyboard channel will now move to the bottom 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. Signal 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 default channel order is with lowest numbered channels at the top of the display. If you prefer the reverse order, open the Edit menu Preferences and un-check Standard

Display shows lowest numbered channel at the top, then use the View menu Standard Display command.

Click the "2" of channel 2 and drag it on top of Channel 1 and release.

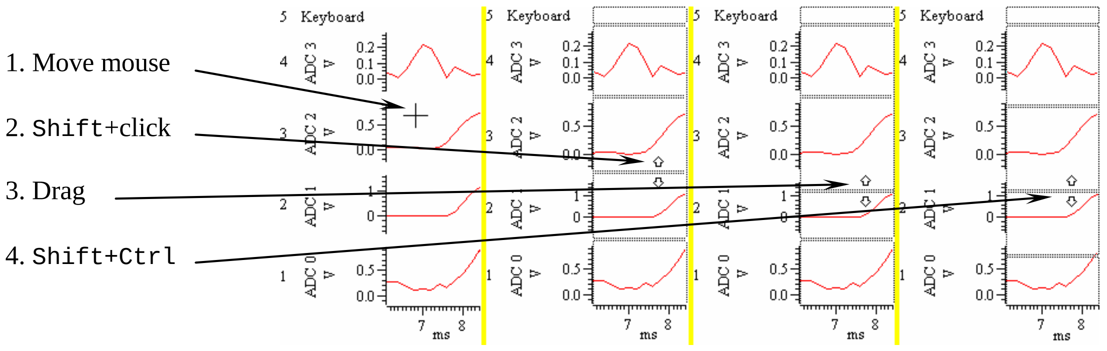


The channels now share the same space with the channel numbers stacked up next to the y axis. The visible y axis is for the top channel number in the stack. To move a stacked channel to the top, double-click the channel number. Stacked channels keep their own y axes and scaling. To remove a channel, 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 marker channel drawn as rate mode, put it at the bottom of the stack, as it will mask channels below it in the stack.

Hold down the `Shift` key and move the mouse over the data area for channel 2. 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.

Add `Ctrl` to scale all channels with a y axis. If there are no channels with a y axis, 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.

Memory views

So far, you have been looking at windows holding data read from a disk file. We call these *File views*. There is another type of data window, called a *Memory view*, which holds data created by the Signal program that is held in memory. This data is usually the result of some sort of analysis. When a memory view is saved to disk and then re-loaded, it has then become a file view; the two types of view are very similar. A simple way to create a memory view is by analysing file view data. There are two steps in the analysis:

1. You set the type of analysis, the channels to analyse, the width (or number of bins) of the analysis result and any other parameters required. This creates a new, empty, memory view with the appropriate frame width and channels.
2. You define the frames from the file view that are to be analysed and Signal carries out the analysis and adds the result into the memory view.

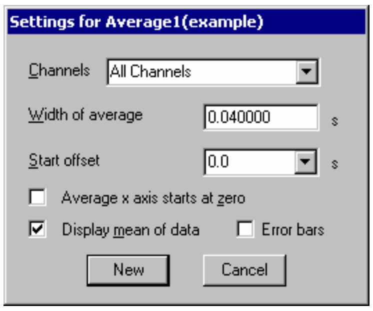
You may repeat step two as many times as required to accumulate results from different sets of frames of data.

You can use the Analysis menu to add additional frames to the memory view. Each frame can hold the result of analysis of different frames from the original file. One way of using this would be to separate averages for each frame state in the source file.

Processed memory views will be automatically re-processed if appropriate. For example, if a memory view holds the average of all tagged frames, and a frame in the source document is tagged or untagged, then the memory view data will be automatically regenerated using the new frames.

The new window behaves like a file view containing one or more frames of data. The simplest way to get a feel for this is to try it, so:

☞ **Make the original file 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 Memory View command to select a Waveform average.**


Settings for Average1(example)
Channels: All Channels
Width of average: 0.040000 s
Start offset: 0.0 s
☐ Average x axis starts at zero
☑ Display mean of data ☐ Error bars

The Settings dialog prompts for information to define the new window. There are three fields that define the waveform average. The Channels field selects the channels to analyse. You can select any channel or list of channels that holds waveform data. The channel list in the pop-up menu only includes suitable channels for analysis.

The Width of average field sets the width of the result, in units set by the source data x axis units. You can choose any width you like, limited only by the width of the source frame.

The Start offset field sets the start point within the frame of the data that is included in the average. This is specified as the offset from the start of each frame to the start of the data included, so an offset of zero will use data from the beginning of each frame. Again, this value is in source x axis units.

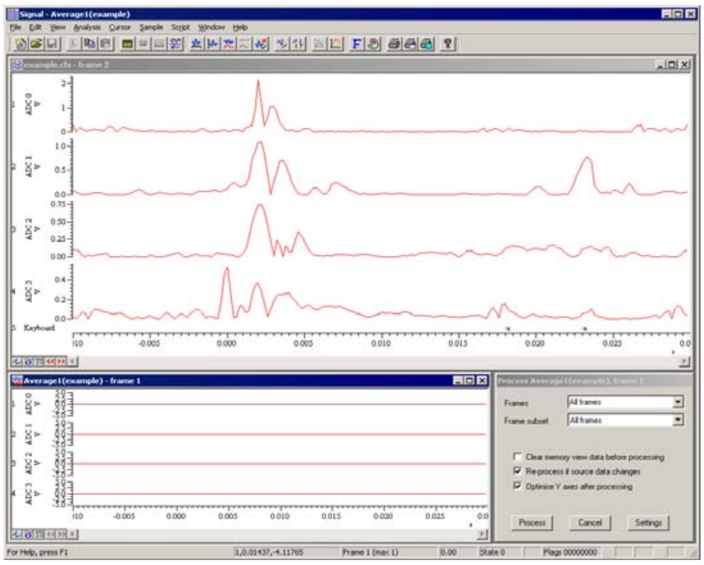
Below these fields are three checkboxes used to enable various options. The first checkbox is used to force the start time of the memory view data to zero, if this is clear then the memory view data x axis start will be copied from the first data added to the average. The second checkbox selects display of the data as a mean value, if this is clear then the sum of the data is displayed. The third causes error values to be calculated and displayed in the memory view.

The first thing to do is to select the channels to analyse. For this example we use All waveform channels, so select this option using the pop-up menu. Set the other fields as they are in the picture above; 0.04 seconds width and a start offset of 0.

☞ **Once you have set these values, click the New button to generate the new memory view. Now set the data frames to analyse.**

When you click the New button several things happen. Signal creates a new memory window ready to display the result of the analysis, the Settings dialog vanishes and the Process dialog appears. You must now set frames from the data document to analyse – choose All frames in both selectors.

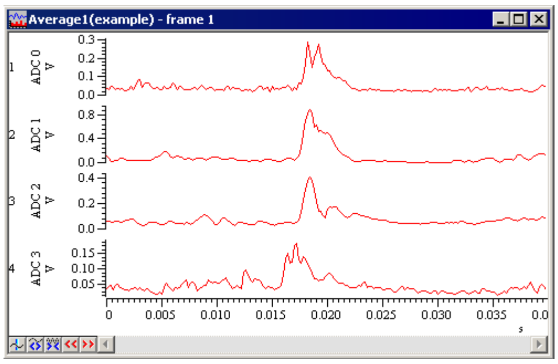
The three check boxes determine how to treat the result of the analysis. You can choose to clear the memory window before you analyse the data, otherwise each new average is added to the previous one. Signal can also re-create the average if the source data changes, and optimise the display after each analysis so that the full data range is visible. You can also click Settings to go back to the Settings dialog.


Signal - Average1(example)
File Edit View Analysis Cursor Sample Script Window Help
example.cfs - frame 2
ADC 0
ADC 1
ADC 2
ADC 3
Keyboard
Average1(example) - frame 1
Process Average1(example), frame 1
Process
Cancel
Settings
For Help, press F1
Frame 1 (max 1)
State 0
Flags 00000000

Frames: All frames
Frame subset: All frames

- [ ] Clear memory view data before processing
- [✓] Re-process if source data changes
- [✓] Optimise Y axes after processing

☞ **When you have set the frames to analyse, click the Process button.**



The dialog closes and the memory window shows the analysed data. You can recall the dialog by selecting the Process command from the Analysis menu. Do this now and click the Process button again. The data in the memory window will not change because this is an average. The count of sweeps displayed by using the View menu Info command will double (as long as you have not checked the Clear memory view before process checkbox).

☞ **Experiment with this new window.**

You will find that the new memory view behaves just like the original file view but has only one frame. This is a good time to experiment with manipulating the data in the memory view without worries about overwriting file data. Use the Analysis menu Modify channel command and try out the options; note that most of the options have keyboard commands assigned.

Saving and reloading memory views

You can save memory view data to disk, as a CFS file. When the CFS file is reloaded into Signal it appears in a file view.

☞ **Now select the File menu Save As command.**

This displays a standard Save As dialog to allow you to select a name for the file to hold the memory data. The memory view data will be saved as a CFS data file. Once you have entered a suitable name and saved the memory view, close it using the File menu Close command. You can open the file holding the memory view data by using the File menu Open command, it is now opened as a file view with a single frame.

XY views

In addition to file and memory views, Signal also uses XY views. These hold multiple data channels (up to 256) 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 have a wide range of uses, ranging from user-defined graphing to drawing pictures. XY views can be used from the script language. Signal can also create XY views holding data taken from measurements from data files.

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

Signal will load and run this script, which generates an analogue clock in an XY view. You can stop the script running (and regain control of Signal) by clicking on the OK button at the upper right hand side of the Signal window.

Now use the analysis menu New XY view command to select Trend Plot analysis. This opens a dialog for the trend plot settings and leads to a process dialog, which is used to select the frames from which measurements are taken. As for memory view processing, trend plot generation can be saved as part of a sampling configuration.

You can manipulate the XY view using the Signal menus. Most of the Signal commands (for example, Show/Hide channels) act on XY views in the same way as for data views. You will find that the view menu contains new items; Options and XY Draw mode, for XY views and the analysis menu is extended to include Delete channel. The Change colours dialog is also different for XY views. You can read more about XY views in the *Edit menu*, *View menu* and *Analysis menu* chapters and in the script language manual.

Summary

If you have followed this chapter, you are familiar with the basic actions required to use Signal. The next chapter tells you how to configure the system to sample your own data. The remainder of the manual covers the menu commands in the system, copying data to other applications and printing, digital filtering, external hardware and signal conditioner support.

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

3 General information

This chapter gathers together information that would otherwise be scattered and repeated throughout this manual. Channel lists and frame specifications 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 Signal; they are gathered together here. The command line lets you control Signal from other programs; script users can even launch another copy of Signal.

Channel lists

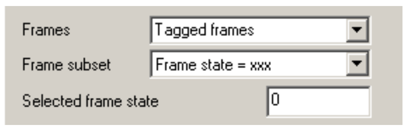
In most places where Signal prompts you for a data file channel you can select a channel or group of channels from a drop down list or you can type in a channel list directly. A channel list is a list of channel numbers or channel ranges separated by commas, while a channel range is two channel numbers separated by two periods or a hyphen. For example 4..7 means channels 4, 5, 6 and 7 while the range 7-4 is equivalent to channels 7, 6, 5 and 4. In nearly all situations the order of channels is not significant and these two ranges are treated the same. The channel list 1,3..5,7 therefore means channel numbers 1, 3, 4, 5 and 7. Virtual channels can also be specified by using the vn channel numbers shown by Signal, for example 1..3,v1,v3..v5.

In most cases, Signal checks channel lists and removes channels that are not suitable for the operation. For example, if you open the `example.cfs` file supplied with Signal, select a waveform average and type in a channel list of 1..32 and then click on another field in the settings dialog, Signal will reformat the list as 1..4 as these are the only suitable channels in the data file. 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.

You can also specify groups of channels in other ways, for example dialogs that require you to enter a channel list will also, if any channels are currently selected, offer you a Selected channels option. Many of these dialogs will also allow you to select and de-select channels while the dialog is visible. You will normally also be offered All channels and All visible channels as other alternatives.

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.

Frame lists



Frames	Tagged frames
Frame subset	Frame state = xxx
Selected frame state	0

In many places, you will be asked to specify the frame or frames to be used for an operation. Dialogs that do this generally provide you with a pair of frame selectors plus an associated numeric value, for example in the Overdraw settings dialog. Other dialogs behave in a very similar manner to this one though the precise options available may vary, the description below only mostly matches the display frame list setup.

You can use the upper Frames selector to choose from All frames, Current frame, Buffer, Tagged frames, Untagged frames, Frames state = xxx (with a separate field for entering the state value) and Last n frames (again with an extra field for entering n). When used online an additional field All sampled frames is available and the text shown for All frames changes to All filed frames. In addition to these options you can also directly enter frame numbers or a frame list such as 1..50,60,61,70..80. Direct entry can also be used to specify wanted or unwanted states; ST:<list> will select all frames whose state value is in the list, while !ST:<list> selects all frames whose state is not in the list. For example ST:1..8 will select all frames with state values from 1 to 8, while !ST:3,6 selects all frames whose state value is not 3 or 6. So this main selector already gives you a wide range of possibilities.

When the main Frames selector is set to All frames, Tagged frames, Untagged frames or a numerical frame list the Frame subset selector is shown. This allows you to select an extra criterion such as Frame state = xxx to the frames that are wanted. Using the two selectors together allows selection of, for example, all untagged frames with a certain state code, giving a wide range of possibilities. These mechanisms are also available in the Signal script language.

Dialog expressions

Many dialogs in Signal 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 `)`. 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 operator precedence level.

View-based expressions

These expressions follows the rules for numeric expressions and allow references to 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 can use the following:

`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.
`MaxTime(n)`	The time of the last data item on channel `n` in a time view.
`AbsTime()`	The frame start time; the time relative to the start of sampling of the trigger time (generally zero on the X axis) for the frame.
`FO`	This is shorthand for `AbsTime()`.

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

Times are normally entered in the preferred X axis units, as set in the preferences. However, where a time is typed into a dialog field with a drop-down for preset strings such as `XLow()` 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 values. We also allow a number with no colons to be followed by s, `ms` or `us` to force Signal to interpret the time entered in seconds in milliseconds or microseconds. So the following are all equivalent: `1.6`, `1.6s`, `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 use of `C0` to `C9`, `H1` to `H9` and `F0` as shorthand for `Cursor(0)` to `Cursor(9)`, `HCursor(1)` to `HCursor(9)` and `AbsTime()` and the use of colons, `ms` and `us` to denote times. If you use these in a script you will get syntax errors. However, you can use these constructs in strings passed as expressions to `CursorActive()` or `MeasureToXY()`.

Data view keyboard shortcuts

The following shortcut key combinations can generally be used in file, memory or XY views, except where otherwise specified.

Key	Operation
`Left arrow`	Scroll display left.
`Right arrow`	Scroll display right.
`Ctrl+Left`	Decrease X axis range.
`Ctrl+Right`	Increase X axis range.
`Ctrl+Home`	Show all X range.
`Ctrl+X`	Provide the X Axis range dialog.
`Down arrow`	Shift data down (scroll Y axis).
`Up arrow`	Shift data up (scroll Y axis).
`Ctrl+Down`	Increase Y range (data shown smaller).
`Ctrl+Up`	Decrease Y range (data shown bigger).
`Home`	Show all Y range.
`End`	Optimise Y range.
`Ctrl+Y`	Provide the Y Axis range dialog.
`Double-click`	File and memory views only. Zoom or un-zoom a channel.
`Del`	Hide the selected channels.
`Ctrl+Del`	Provide the customise display dialog.
`Ctrl+B`	File and memory views only. Toggle displaying the frame buffer.
`Ctrl+D`	File and memory views only. Toggle frame display list overdrawing
`Ctrl+Shift+D`	File and memory views only. Provide the frame display list dialog.
`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 file and memory views.
`Ctrl+Shift+ Left/Right`	File and memory views only. If cursor 0 is active, search for the next/previous feature and scroll the screen to make it visible.
`PgUp`	File and memory views only. Next frame.
`PgDn`	File and memory views only. Previous frame.
`Ctrl+PgUp`	File and memory views only. Last frame.
`Ctrl+PgDn`	File and memory views only. First frame.
`Ctrl+G`	File and memory views only. Provide the Goto frame dialog.
`Ctrl+C`	Copy the image of the view, and data as text to the clipboard.
`Ctrl+N`	Open a new data file.
`Ctrl+O`	Open the file open dialog.
`Ctrl+E`	File and memory views only. Export data as.
`Ctrl+P`	Print the current data file.
`Ctrl+L`	Open the Evaluate window to run single line script commands.
`Ctrl+T`	File and memory views only. Toggle the frame tag.
`Ctrl+Z`	Undo the last undoable operation.
`Ctrl+Break`	Break out of long drawing or calculation operations.

There are also a large number of keyboard shortcuts for various analysis and data manipulation operations for file and memory views only. These are documented separately at the end of the *Analysis menu* chapter.

Text view keyboard shortcuts

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

Text caret control

The text caret is a flashing vertical bar that indicates the current position. Do not confuse this with the I-beam mouse pointer which does not flash and which indicates the mouse position. Each time you click and release the left mouse button (we assume you haven't swapped the mouse buttons), the caret moves 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.
`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.
`Ctrl+]` (`[`)	Start of next (previous) paragraph (after empty line)
`Ctrl+\` (`/`)	Word part right (left).
`Insert`	Swap between insert mode caret | and overtype caret _

Cut, Copy, Paste, Delete, Undo and Redo

Some of these operations are also available from the Edit menu and the main toolbar.

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+Shift+T`	Copy the current line to the clipboard.
`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+D`	Duplicate the selection.
`Ctrl+X`	Cut the selected text and copy it to the clipboard.
`Shift+Del`	Cut the selected text and copy it to the clipboard.
`BackSpace`	Delete the selection or the character to the left of the text caret.
`Del`	Delete the selection or the character to the right of the text caret.
`Ctrl+Del`	Delete word right. Add `Shift` to delete to the end of the line.
`Ctrl+D`	Duplicate the selection.
`Ctrl+Shift+L`	Delete the current line.
`Ctrl+Z,Alt+ Backspace`	Undo the last interactive text operation. The editor supports more or less unlimited levels of Undo.
`Shift+Ctrl+Z`	Redo the immediately previous Undo operation.

Miscellaneous

These commands do not fit into any other category!

Key	Operation
`Ctrl+U`	Convert the selection to upper case. Add `Shift` for lower case
`Ctrl+Add,Sub`	Change font size (`Add` and `Sub` are numeric keypad `+` and `-`).

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 mark 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 Signal application title bar and using the pop-up context menu that appears.

Indent and Outdent

The structure of Signal 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 (note that the Auto Format command is capable of setting the indentations for you). The tab size is set in the Edit menu Preferences option.

Key	Operation
`Tab`	If there is a multi-line 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 multi-line 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 Signal and between Signal and other applications that support it (for example the Signal Help system). Signal 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 and drag. The mouse pointer will 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, click and drag. A 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.

Virtual space

Prior to Signal version 4.06, the text caret could only be positioned between or next to existing characters in a text line. From version 4.06 onwards, you can position the caret beyond the end of the text in a line by clicking in a blank area with the mouse, or using the cursor right key. You cannot position the text caret below the last line of text. When the caret is beyond the end of the line, it is said to be in *virtual* space. If you type with the caret in virtual space, space characters will be added to fill in the virtual space up to the text you type.

There are two main uses for virtual space: to add comments without having to space along to the required column, to make rectangular selections without having strange visual effects due to short lines.

If you use the script language to manipulate the text caret and make selections, virtual space is ignored; a caret in virtual space will be treated as if it is at the end of the line. There are no script commands that will move the caret into virtual space. If there is a requirement for the script to report or use virtual space, we will extend the script in a compatible way to incorporate it.

Multiple selections

From Signal version 4.06 onwards, you can make multiple selections in a text view. To do this, hold down the `Ctrl` key and click and drag. Each time you make a new selection in this way it becomes the current selection; all previous selections are shown with a different selection colour. When you have a multiple selection, each selected area has a flashing text caret at the insertion point. If you type characters, these will appear at all insertion points, replacing all the selected text. If you use the delete or backspace key, all selected text will vanish. Note that pasting into a multiple selection will clear all the selected text, then insert the pasted text at the current (last made) selection.

Multiple selection can be useful when you want to move several non-consecutive script functions to make them consecutive.

Select rectangular text area

You can select, cut, paste and drag rectangular selections of text within Signal. 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. You can also paste such text into other applications as plain text.

A rectangular selection will paste within Signal as a rectangular selection. Beware that `Ctrl+X` (cut) on a rectangular selection followed by `Ctrl+V` (paste) will not leave the text unchanged (unless you select the text from bottom to top). The cut operation will leave a vertical flashing line (assuming a fixed pitch font) with the insertion point marked by a more visually obvious caret. The paste operation will paste the cut text as a rectangular block at the insertion point.

A rectangular selection is a multiple selection.

The Signal command line

When Signal starts, it checks the command line for option switches and for files to load. If there is no command line, Signal looks in the folder that it ran from for a script called `startup.sgs` and, if it exists runs it. If `startup.sgs` is found and run or if the command line loads a file, any start up messages that wait for a user response are suppressed.

The command line holds options and file names separated by white space characters (space and tab). If a file name contains spaces, you must surround the file name with quotation marks. Options start with / or – followed by a character to identify the option.

`/M` When you start Signal, it checks if there is already a running copy. If there is, the new one quits. This option removes the check, allowing multiple copies to run on a single system. You need a Signal licence for each copy except when using multiple synchronised 1401s to capture related data on one computer under the control of a single operator, when one licence is sufficient. To do this, you must set separate file names for each 1401 in the Automation tab of the Sampling Configuration dialog.

`/Un` `n` is 1-8 to select a 1401 when you have more than 1. The default is `/U0`, which uses the lowest-numbered unused 1401. You set a device number in the CED 1401 device settings in the Device Manager (My Computer->Properties->Hardware->Device Manager).

`/Q` Quiet startup. Suppress all message boxes and the Signal "splash screen".

The remaining items in the command line are assumed to be file names. Signal 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, Signal runs it before continuing with the remainder of the command line.

As an example, suppose we want to launch Signal so that it automatically opens a data file called `example.cfs` and runs `doit.sgs` to process it. Follow these steps:

1. Create a short cut to `cfsview.exe` (this is the Signal program).
2. Right-click on the new short cut and select Properties and open the Shortcut tab.
3. Add `example.cfs doit.sgs` 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.

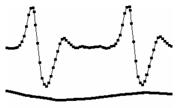
4 Sampling data

If you have worked through the previous section you already have most of the skills needed to work with a new data document created by sampling; a *sampling document*. A sampling document is much the same as an old document, except that the sampling document grows by adding frames to the end. The sampling document also has an extra frame, frame zero, that contains transitory data retrieved as it is sampled.

Types of channel

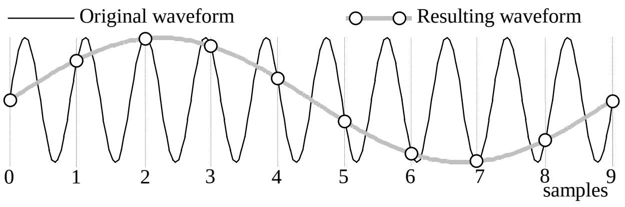
Before we discuss the sampling configuration dialog, we need to provide some background on the types of data channel that Signal can sample. Signal handles two types of channel: waveform and marker.

Waveform channels



The waveforms that Signal records and displays are continuously changing voltages. Signal stores waveforms as a list of numbers that represent the waveform amplitude at equally spaced time intervals. These numbers are 16-bit integers. They are scaled using calibration values to produce the floating point data values that Signal uses and displays. Signal can also use and create waveform channels where the underlying data are floating point values; these are indistinguishable from channels using integer data in nearly all circumstances. Floating point data can be stored more accurately though it requires twice as much disk space to do so. 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. A set of samples taken at regular intervals is referred to as a *sweep*.

Minimum sample rate



The sample rate for a waveform must be high enough to represent the data correctly. You must sample at a rate at least double, and preferably 2.5 to 5 times, 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. The dots in the diagram represent samples; the lines show the original waveform. On the other hand, you want to sample at the lowest frequency possible, otherwise your disk will soon be full.

Voltage range

The 1401 ADC (Analogue to Digital Converter) measure varying voltage signals in the range of ±5 volts, these can be optionally changed to ±10 volts if required - usually this requires the unit to be returned to CED. The Power1401 mkII and Micro1401-3 incorporate software control of the ADC input range - you can switch these units between 5 and 10 volts using the 1401 options dialog in the Try1401 utility installed with Signal

Use of filters

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

Input connections

To sample waveforms, connect your waveform signals to the 1401 ADC input ports. Ports 0-7 (0-3 for an unexpanded Micro1401) are the labelled BNC connectors on the front of the 1401. For the original standard 1401 and 1401*plus* ports 8-15 are on the 15 way Cannon connector on the front of the 1401, the connections are:

ADC port	8	9	10	11	12	13	14	15	Ground
Pin number	1	2	3	4	5	6	7	8	9-15

For early-model Power1401s without an ADC expansion box ports 8-15 are on the rear panel 37-way "D-type" connector labelled Analogue Expansion. The connections are:

ADC port	15	14	13	12	11	10	9	8	Ground
Pin number	35	34	33	32	31	30	29	28	1-19

For late-model Power1401s and Power1401 mk IIs the Analogue expansion socket is a high-density 44-way "D-type" connector. On this connector each signal has its own separate ground return. The connections are:

ADC port	8	9	10	11	12	13	14	15
Pin number	33	34	35	36	37	38	39	40
Ground return	3	4	5	6	7	8	9	10

For port numbers above 15 you will require a 32-channel expansion card (for the standard 1401 or 1401plus) or expansion topboxes for Micro or Power1401s. If you have a Micro1401 ADC expansion box installed, ports 4 to 15 are BNC connectors on the expansion box. If you install a Power1401 expansion box, it adds new ADC ports starting at number 8 and the port numbers on the rear expansion connector are adjusted to start after the expansion box ports, so if you add an ADC-16 topbox ports 0-7 are on the main unit, ports 8-23 are on the topbox and ports 24-31 are on the rear connector. If you try to sample using a port above the number available, Signal will generate an error message.

TTL compatible signals

In several places in this manual we refer to TTL compatible signals. TTL stands for Transistor-Transistor Logic and is a method of passing logical (High/Low) 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.

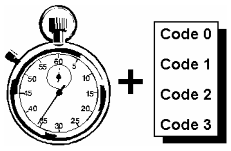
The TTL inputs and outputs on the 1401 are the digital inputs and outputs, the event inputs, the clock F external frequency inputs, the ADC external convert input, the clock output, the DAC Bri output and the micro1401, Micro1401 mk II or Power1401 trigger input. On Micro1401s and Power1401s, the event inputs are not actually TTL but can be treated as such.

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 some 0.8 mA to pull them into the Low TTL state. Alternatively, you can connect them to ground to pull them low (useful for the Event inputs).

See the *Owners handbook* of your 1401 interface for full details of each input port.

Marker channels



Signal can sample two types of Marker data: keyboard and digital. Signal treats both marker types identically once the data has been captured; they differ only in their source.

A Marker is a 32 bit time value, in units of the sample interval on waveform channels for keyboard markers and of the output resolution for digital markers. In addition to the time a marker has 4 bytes of marker data. The first of these 4 data bytes is the ASCII code of the keyboard character pressed by the user (for a keyboard marker) or an 8 bit digital code read by the 1401 (for a digital marker). The remaining 3 bytes are normally zero.

Keyboard markers

Keyboard markers time events to an accuracy of, at best, around 0.1 second, you should use digital markers if you require precise timing. The upper and lower case characters a-z and the numbers 0-9 are logged, but *only when the new document window or the sampling control panel is the current window*. The keyboard marker channel, if created, is the first channel after the waveform channels.

Digital markers

These are not available for the Standard 1401. Digital markers are timed as accurately as the outputs and record 8 bits of TTL data. These can be used as 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 0 to 7 of the 1401 digital input.

Digital marker connections

The digital marker data is read from the 1401 digital inputs bits 0 to 7. These inputs are found on the 1401 Digital inputs connector; a 25-way ‘D-type’ plug located on the front of the 1401*plus*, and on the rear of Micro1401s and Power1401s. In addition to the data lines a TTL pulse is required on the digital inputs Data Available input to log a digital marker.

Digital input: bits 0 to 7

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

Digital input: other signals

Signal	1401*plus*	All others
Data Available	pin 24	pin 23

To log a digital marker, apply a low going TTL pulse at least 1 µs wide to the Data Available. When the 1401 detects the falling edge of the input, it latches the input data on Power1401s and Micro1401s. If you have a 1401*plus* you must keep the digital input data signals stable for 50 microseconds after the low going data available edge.

Marker codes

When Signal displays marker data from keyboard marker or digital marker channels, it shows the code of the first of the four markers as well as the marker time.

Marker codes have values from 0 to 255. This is the same range of numbers that the 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 Signal displays a marker code that has the same value as the ASCII code of a printable character, it displays the code as a character, otherwise it displays the marker code as a two digit hexadecimal number. 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.

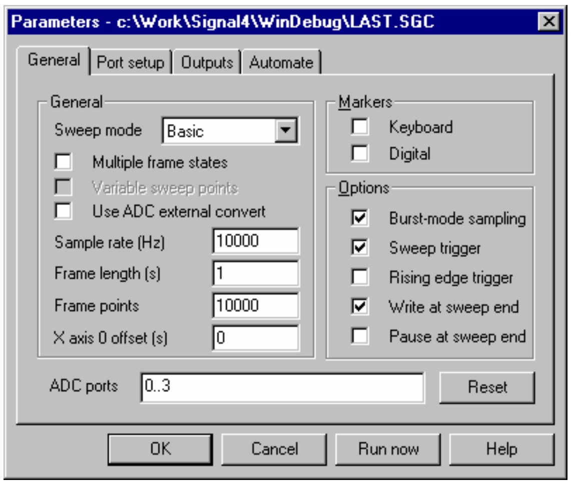
The printable characters (as far as Signal is concerned) span the hexadecimal range 20 to 7e (32 to 126 decimal) and are as shown in the table:

+	**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	{	|	}	~	

To find the hexadecimal code of a printable character, add the number above the character to the number to the left of the character. 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`.

Sampling configuration

Before you start to sample data with Signal you must set the sampling configuration. This is done through the Sample menu Sampling Configuration dialog, which is also available by using a toolbar button. The sampling configuration dialog is a tabbed dialog; containing a number of tabs for selecting different sections of the parameters. Click on a tab to display the corresponding section of the dialog. The sections always available are General, Port setup, Outputs and Automate. There are other sections, Peri-trigger, States and Clamp that hold extra information not relevant to all sampling configurations. The Peri-trigger tab appears only when the sampling mode selector in the General section is set to Peri-trigger, the States tab only appears when the General section Multiple frame states item is checked. The Clamp section is only available if clamping experiments are enabled using the Edit menu Preferences dialog.


Parameters - c:\Work\Signal4\WinDebug\LAST.SGC

General | Port setup | Outputs | Automate

General
Sweep mode: Basic
☐ Multiple frame states
☐ Variable sweep points
☐ Use ADC external convert
Sample rate (Hz): 10000
Frame length (s): 1
Frame points: 10000
X axis 0 offset (s): 0

Markers
☐ Keyboard
☐ Digital

Options
☑ Burst-mode sampling
☑ Sweep trigger
☐ Rising edge trigger
☑ Write at sweep end
☐ Pause at sweep end

ADC ports: 0..3

General configuration

The General section holds a selector for the sweep mode, the multiple frame states checkbox, fields to define the waveform sampling rate and the frame width or points, checkboxes to control the creation of marker channels and various other options, plus a list of ADC ports to sample.

Sweep mode

The Sweep mode selector defines how sweeps of data are taken and triggered and how sampling sweeps relate to the outputs system. The modes available are:

Basic	the trigger for a sweep of data is a TTL pulse at the start of the sweep, and pulse outputs start and finish at the same time as a sampling sweep.
Peri-trigger	the trigger point can be before the start of the sweep, at the start of the sweep or at any point within the sweep. Pulse outputs start at the trigger point and finish at the end of the sampling sweep. This mode allows a wide variety of triggers including threshold crossings on a sampled waveform channel. The trigger point and type of trigger are set in a separate peri-trigger configuration page.
Outputs frame	the pulse outputs are triggered rather than the sweep, output pulses can occur both before the sweep starts and after the sweep is over, and the sampling sweep is started by the outputs; the start time of the sampling sweep is set along with the outputs in the pulses dialog.
Fixed interval	similar to Outputs frame, but the sweeps are internally timed so that they occur at a specified interval; a random variation in the interval can be generated. Both the sweep interval and any required random variation are set using the pulses dialog that is used to define the outputs. External sweep triggers are not used in this mode. The fixed interval must be longer than the overall sweep length so that Signal has enough time to get things ready for the next sweep. The amount of time required varies a lot with the sampling and computer system, 100 to 200 milliseconds is generally sufficient.

Fast triggers	like Basic mode except that multiple frame states and incremental pulsing are not available. This keeps the inter-sweep interval to a minimum – less than 100 microseconds usually.
Fast fixed int	has the same limitations as Fast triggers but uses a fixed interval between sweeps rather than requiring an external trigger; in this mode no random variation in the interval is available - the interval is set using the pulses dialog. The fixed interval still has to be longer than the overall sweep length, but now only by a millisecond or two.
Gap-free	is like Fast triggers mode but only the first frame is triggered; subsequent frames start immediately after the previous frame finishes with no loss of data, change in sampling interval between points or any variation in sample timing. It is used when you want to sample continuously but are happy to break up the data into frames. If you want full-blown continuous sampling then you should consider using the CED Spike2 software.

Useful extra information can be found in the *Pulse outputs during sampling* chapter – the pulses setup dialog is used to set the fixed interval sweep timing and other parameters.

Multiple frame states

This checkbox enables sampling with multiple frame states. With multiple frame states disabled, all sampling sweeps are the same, the same pulse outputs are generated and the new data frames are set to state zero. With multiple frame states enabled, each sampling sweep can be different from other sweeps in a number of ways and the data frame states are different to indicate what happened during sampling. This can be used to achieve a variety of useful effects.

The use of multiple frame states is a complex topic which is covered in the Sampling with multiple states chapter of this manual.

Variable sweep points

This checkbox is only available with Outputs frame and Fixed interval sweep modes with multiple frame states enabled; it allows different sampled sweeps (with different states) to have different numbers of ADC data points. The number of data points for each state is set in the pulses outputs dialog, the sweep points set in this page sets the upper limit to sweep data points.

When variable sweep points data is displayed, the allowed X range is set by the maximum number of sweep points which is set here, so you may have frames of data that do not reach the limits of the X axis.

Use ADC external convert

ADC sampling is normally done on a clocked interval basis. This means that each sample point in the sweep is separated by the same time period. With this box checked each point is triggered by a pulse supplied by external hardware. On the 1401*plus* this trigger is connected to the Ext BNC connector on the front panel. On more modern 1401s you should use pin 6 of the Events socket on the back of the 1401. The pins are numbered from right to left with pin 6 being the 6th hole along on the top row.

Sample rate

The Sample rate field sets the sampling rate for all waveform channels, in Hz. The rate displayed will not always be the preferred rate that was entered; it shows the closest rate achievable given the 1401 clocks and the number of ADC ports to be sampled. The overall sampling rate in the 1401 is the Sample rate times the number of ADC ports.

With a Power1401 625, the maximum overall sampling rate is 625 kHz. A Micro1401 mk II will sample at up to 500kHz, while a Power1401 mkII will go up to 1MHz. With a micro1401 or a 1401*plus* (with modern 12 bit ADC hardware), the maximum overall sampling rate is 333 kHz. A 1401*plus* with a 16 bit ADC can sample at 400 kHz. With a standard 1401 or 1401*plus* with older ADC hardware the maximum rate is 82.5 kHz.

The sampling configuration dialog does not apply hardware-specific limits to the sample rates that you enter. If you use a sampling configuration with an overall sampling rate beyond that achievable a 1401 sampling error will occur and be reported by Signal. If Signal detects a Power1401 (or other more modern 1401 type) during program startup, it enables a higher timing resolution. You can force Signal to allow this higher timing resolution by checking the Assume Power1401 hardware box in the Edit menu preferences dialog.

Frame length and points The Frame length and Frame points fields set the length of the sampled frame. The frame length is always shown in the appropriate time units. Changes made to one of these fields automatically cause an appropriate change in the other. The Frame length field also updates whenever the sampling rate changes.

The maximum frame length possible varies with the model of 1401 and the 1401 memory installed; each sampled point requires two bytes of memory. For a standard 1401 the maximum number of points (points per frame times number of channels) is about 28,000, for a micro1401 or unexpanded 1401*plus* the limit is about 480,000 while for an expanded 1401*plus* or Power1401 the limit depends upon the amount of extra memory installed but is at least 15 million. For a Micro1401 mk II the limit is either about 480,000 or 1 million depending on the amount of memory the unit was built with. The sampling configuration dialog does not apply any limits to the frame length that you enter, when sampling starts the 1401 memory required is checked against the memory that is available.

X axis zero offset Normally the X axis zero appears at the start of the frame, or at the trigger time for peri-triggered sampling mode (see below). This position can be moved by entering a non-zero value in this field. This does not alter how and when sampling takes place, only the way in which times are displayed on the x-axis.

ADC ports This field sets the ADC ports to sample, up to 80 ports can be specified. You can enter individual ADC ports separated by commas or spaces or a range of ports such as 0..7 or both (for example `"0,7,1..6"`). Port numbers between 0 and 127 are accepted. Each sampled ADC port creates a separate waveform channel in the resulting data document. The ADC ports are sampled in the order specified and the data document will have all waveform channels first, so the first ADC port provides data for channel 1, the second for channel 2, and so forth. Duplicate port numbers are allowed and will be sampled (see page 4.1 for a discussion of waveform channels).

Keyboard marker The Keyboard marker checkbox enables the keyboard marker channel and logging of keyboard markers. If the keyboard marker channel is enabled then it is the first channel in the data document after the waveform channels.

Digital marker The Digital marker checkbox enables the digital marker channel and logging of digital markers. If this channel is enabled it is the first channel after the keyboard marker channel or the waveform channels if the keyboard marker channel is not present. (See page 4-3 for a discussion of marker channels).

Burst mode Check this box for burst mode sampling, leave it clear for equal interval sampling. In equal interval sampling the waveform data points are sampled individually in turn. The interval between samples is 1/(Sample rate * number of ADC ports). In burst mode sampling all the ADC ports are sampled in a burst, as close together as possible, the interval between bursts is 1/Sample rate. Equal interval sampling has some advantages with the standard 1401 as it loads the 1401 system more evenly, while burst mode ensures that the interval between samples on adjacent ADC ports is kept to a minimum. With a 1401*plus*, Power1401 or Micro1401 there is no performance penalty with burst mode. Burst mode is generally recommended because it allows greater accuracy in matching the sampling rate used to that required.

If the first two ADC ports sampled are ports 0 and 7 (or 0 and 3 for a micro1401 or Micro1401 mk II), then the second sample and hold circuit optionally fitted to 1401s is enabled. If fitted this option causes the sampling on ports 0 and 7(3) to be exactly simultaneous. If the 1401 has the 1401-32 multiple sample and hold card fitted, then burst mode sampling will be exactly simultaneous on all channels. Second sample and hold is not currently available Power1401s.

With Peri-triggered sampling burst mode is always used for efficiency reasons.

Sweep trigger

This checkbox sets the initial state of the Sweep trigger checkbox in the sampling control panel enable and disable sweep triggers. With sweep triggers enabled, a sampling sweep will not occur until a trigger has been detected, the sampling configuration determines what a trigger is. With sweep triggers disabled, a sampling sweep starts immediately. For Outputs frame sweeps, the sweep trigger starts the outputs rather than the sampling sweep. For Basic or Outputs frame sweeps, the sweep trigger is a TTL pulse that is applied to the 1401 and 1401*plus* event 0 inputs, or to the Micro1401 or Power1401 Trigger input. For Peri-triggered sweeps the trigger can be any of a number of signals.

There is a small (~10 microseconds) delay between the time of the sweep trigger and start of sampling. This is affected by the outputs synchronisation controls in the Outputs configuration section. When using Basic sweep mode, it is possible to start sampling at exactly the time of the sweep trigger by providing the trigger pulse to both the 1401 E0 and E4 inputs. This mechanism is only available when the synchronised sampling option in the Outputs configuration is disabled. For the micro1401, Micro1401 mk II and Power1401, the trigger input is automatically routed to both E0 and E4 internally if appropriate, thus guaranteeing a precise start of sampling relative to the trigger.

Rising edge trigger

Sweep triggers are normally on a falling edge of a TTL pulse. Check this box to make them occur on the rising edge.

Write sweep to disk

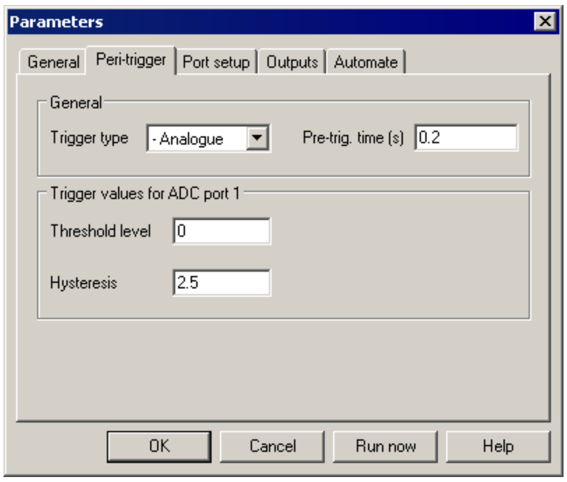
This checkbox sets the initial state of the Write to disk at sweep end checkbox in the sampling control panel. When this is set, sampled sweeps are automatically written to disk when the sweep finishes.

Pause at sweep end

This checkbox sets the initial state of the Pause at sweep end checkbox in the sampling control panel. When this is set, Signal waits at the end of a sweep instead of immediately starting the next sampling sweep.

Peri-trigger configuration

The Peri-trigger section holds information that is specific to the Peri-trigger sampling mode. It is only available when Peri-triggered sweeps are selected in the General section.



General
Trigger type: - Analogue
Pre-trig. time (s): 0.2

Trigger values for ADC port 1
Threshold level: 0
Hysteresis: 2.5

At the top of the dialog is a selector for the type of trigger and a field for the pre-trigger points. Below this is a section holding details of the trigger parameters. The contents of this section changes with the type of trigger, the individual fields will be described along with the various types of trigger.

Trigger type

This can be set to one of +Analogue, -Analogue, =Analogue, Digital or Event. The three analogue types monitor the last ADC port in the sampled ADC ports list for a trigger. The trigger levels are shown with the sampled data as a pair of cursors which can be moved, without stopping the sampling, to alter the levels. The Digital trigger waits for a specified state on a bit in the 1401 digital inputs, while the Event trigger is a TTL pulse just as for the Basic sample mode triggers. Each form of trigger has different parameters:

+Analogue
Trigger on a positive-going level transition. The parameters are Threshold level and Hysteresis, both in units set by the channel calibration. The trigger process first waits for the sampled data to go below (Threshold - Hysteresis) and then triggers when the sampled data value rises above Threshold. The hysteresis acts to prevent false triggering by noise as the sampled data passes downwards through the threshold level, triggering can only occur after the sampled data has clearly been below the threshold. If you find that you are having problems with false triggers due to noise, increase the Hysteresis value.

-Analogue
Trigger on a negative-going level transition. This is identical to +Analogue, but in the opposite direction. The trigger process first waits for the sampled data to go above (Threshold + Hysteresis) and then triggers when the sampled data value falls below Threshold.

=Analogue
Trigger on signal moving outside a pair of levels. The parameters are Upper threshold and Lower threshold. The trigger process first waits for the sampled data to go between the thresholds. It then monitors the sampled data and triggers when the sampled data value is above the upper level or below the lower level.

Digital
Trigger on a digital input bit state. The parameters set the digital input bit, from 8 to 15 and select triggering on a high bit or on a low bit. The trigger occurs when the bit is in the correct state. There is no requirement for the bit to be in the other state first. The digital inputs are found on the 1401 digital inputs connector, the pins for the digital bits are:

Digital input bit	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	GND
Digital input pin	1	14	2	15	3	16	4	17	13

Event
Trigger on a TTL pulse. There are no parameters; the trigger occurs when a TTL pulse is detected on the standard 1401 or 1401*plus* Event 0 input, or the Trigger input on a micro1401 or Power1401.

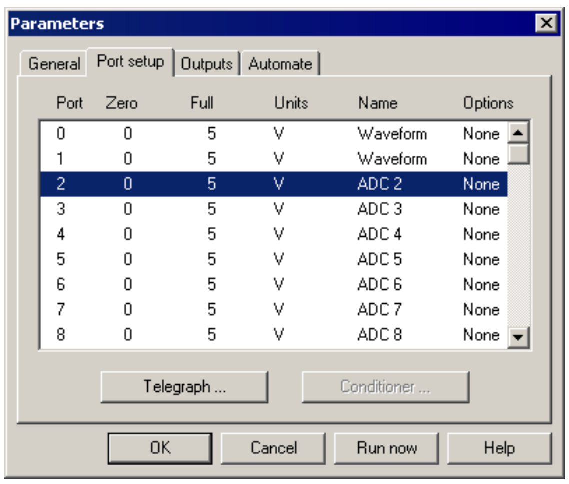
Pre-trig. time

This parameter sets the number of points in the frame before the point at which the trigger occurred. This can have any value from -(1,000,000 * sample interval) to the length of the frame -(2 * sample interval). If the value is negative, this means that points sampled after the trigger occurs are discarded before the first point in the frame is kept. If the value is positive, then the specified time must have elapsed before the search for a trigger begins and the resulting frame contains points sampled before the trigger occurred.

When a non-zero pre-trigger time is specified the resulting data x axis adjusts to start at -pre-trig. time. Thus a negative value gives an x axis starting at some positive value because the first point in the frame was sampled some time after the trigger. Similarly, a positive value gives an x axis starting at a negative value as some points sampled before the trigger are shown.

Ports configuration

The port setup section defines the individual ADC ports. You can set the scaling and units for data sampled from a port, the name of a data channel taken from a port, and specify online processing options for data from a port.



Port	Zero	Full	Units	Name	Options
0	0	5	V	Waveform	None
1	0	5	V	Waveform	None
2	0	5	V	ADC 2	None
3	0	5	V	ADC 3	None
4	0	5	V	ADC 4	None
5	0	5	V	ADC 5	None
6	0	5	V	ADC 6	None
7	0	5	V	ADC 7	None
8	0	5	V	ADC 8	None

The main dialog displays the current settings for all of the available ADC ports. Double-click on the entry for a particular port to open the parameters dialog for that port. The entries for each port (both in the main dialog and in the parameters dialog) are:

Zero	The value (in the specified units) corresponding to a zero volt reading from the ADC. This value, along with Full, is used to convert ADC data into the floating-point values used by Signal.
Full	The value corresponding to the full scale reading from the ADC. To scale the data in volts, this will be 5 for a 5 volt 1401 and 10 for a 10 volt 1401.
Units	The units for calibrated data. This is a string from 1 to 6 characters long. If you set the first character of the units as a space this allows future versions of Signal the option of automatically adjusting the units by replacing the space with a character representing a factor of a 1000 such as µ or k.
Name	The port name. This is a string from 1 to 19 characters long, it sets the title of the waveform data channel sampled from this port.
Options	This is a string of 0 to 8 characters that holds online processing options for data from the port. Characters corresponding to various processing options can be entered into this dialog. Currently, only one processing option is available; enter an 'R' character to cause online rectification of sampled data.

The Conditioner… button opens the signal conditioner setup dialog if a signal conditioner has been found (see the *Programmable Signal Conditioners* chapter).

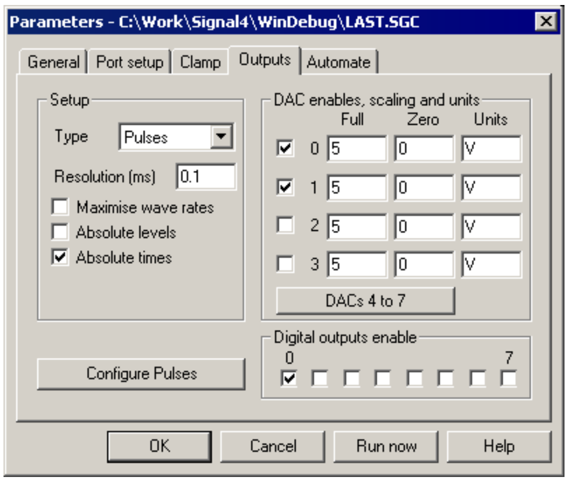
The Telegraph… button on the bottom left opens a dialog that allows you to configure amplifier telegraph support. Amplifier telegraphs are signals, usually analogue outputs, from an amplifier that indicate amplifier settings such as gain and offsets. By collecting and interpreting the telegraph signals, Signal can automatically adjust for changes in the amplifier settings. Signal supports a standard telegraph mechanism using analogue voltages sampled using the 1401 or it can use a custom DLL to support alternative mechanisms. If support for an alternative telegraph system has been installed then the label on this button will change. For information on setting up and using amplifier telegraphs, see the *Amplifier Telegraphs* chapter. Only a few types of amplifier provide telegraph signals; if you do not have such an amplifier you can ignore this option.

Clamp configuration

This section of the sampling configuration is only available when clamping support is enabled in the preferences. It is described in the *Sampling with clamp support* chapter.

Outputs configuration

The outputs section is used to set the outputs required during sampling, which DACs and digital outputs are available for use and to set the DAC units and scaling. The leftmost area is used to configure the type of outputs. It contains a selector for the type of outputs required plus items specific to the currently selected type of outputs. The right-hand areas enable and set up the output ports.



Parameters - C:\Work\Signal4\WinDebug\LAST.SGC

Setup

Type: Pulses

Resolution (ms): 0.1

☐ Maximise wave rates

☐ Absolute levels

☑ Absolute times

DAC enables, scaling and units

		Full	Zero	Units
☑	0	5	0	V
☑	1	5	0	V
☐	2	5	0	V
☐	3	5	0	V

Digital outputs enable

0							7
☑	☐	☐	☐	☐	☐	☐	☐

Outputs type

This control selects the type of outputs to use from either None, Pulses or Sequence. The controls in the area below the selector vary according to the selection.

Outputs type: None

This disables outputs during sampling. When selected, only a single control is shown:

Timer period (ms)

This item sets the period of the internal timer used to measure the absolute frame start time and to time digital markers. A value of 1 to 10 ms is usually appropriate for these purposes. Values from 0.1 microseconds to 250 ms can be entered and they are rounded to the nearest 0.1 microseconds.

Outputs type: Pulses

This selects pulse outputs during sampling. The pulses can be controlled by the script language or interactively using a dialog. The details of configuring and using pulse outputs are covered separately in the *Pulse outputs during sampling* chapter. When pulse outputs are in use, a number of controls to configure the pulses are shown:

Resolution (ms)

This sets the timing resolution of the output pulses in milliseconds or microseconds and also sets the period of the internal timer used to measure the absolute frame start time and to time digital markers. Values are rounded to the nearest 0.1 microseconds. The practical limit to the resolution depends upon the type of 1401 in use; for a 1401*plus* the recommended limit is 3 ms, for the micro1401 values down to 0.1 ms can be used, for the Micro1401 mk II 25 microseconds, while for Power1401s you can go down as far as 6 to 10 microseconds.

Maximise wave rates

The design of the 1401 is such that it is possible to maximise either the timing precision with which the pulses are generated or the highest possible arbitrary waveform output rates, but not both. Normally you should leave this checkbox clear but if you want high waveform rates and are getting errors indicating the rate in use is too high, setting this checkbox may help.

Absolute levels This selects between absolute and relative pulse levels. With absolute pulse levels, the pulse amplitude sets the level directly, with relative levels the pulse amplitude is added to the level before the pulse to get the actual pulse level.

Absolute times This selects between absolute and relative pulse times. With absolute times, the pulse dialog allows you to enter the pulse start time directly; with relative times you use the delay since the start of the previous pulse. This control only affects the way in which the pulse dialog handles pulse start times, not the other times shown in the dialog, the underlying pulse data or the generation of pulses.

Configure pulses Press this button to configure the output pulses using the pulse configuration dialog. Details of doing this are covered in the *Pulse outputs during sampling* chapter.

DAC enables, scaling and units This section contains four sets of controls, one for each DAC (users of micro1401s and Micro1401 mk IIs should ignore DACs 2 and 3). These control if a DAC is available for use and set the scaling and units with which DAC values are defined.

Enable These checkboxes enable the DACs for use. Set a checkbox to use a DAC, leave it clear otherwise. The fewer DACs are enabled for output the more space is available for the display of each DAC in the pulse dialog.

Zero The value (in calibrated units) corresponding to a zero value output from the DAC. This value, along with Full, is used to convert the floating-point values used by Signal into the integer quantities actually used by the DAC hardware. This conversion process occurs when generating pulse outputs, when waveform data is pasted into an arbitrary waveform pulse and when compiling pulse sequences.

Full The value corresponding to the full scale output from the DAC. For DACs calibrated in volts set this to 5 for a 5 volt 1401 and to 10 for a 10 volt 1401.

If your 1401 has a patch clamp scaling card fitted, this scales DAC 3 in a 1401*plus* and DAC 1 in a Power1401 or Micro1401. To calibrate the DAC in volts, set Full to 2.048 or 10.24 depending on the scaling card setting.

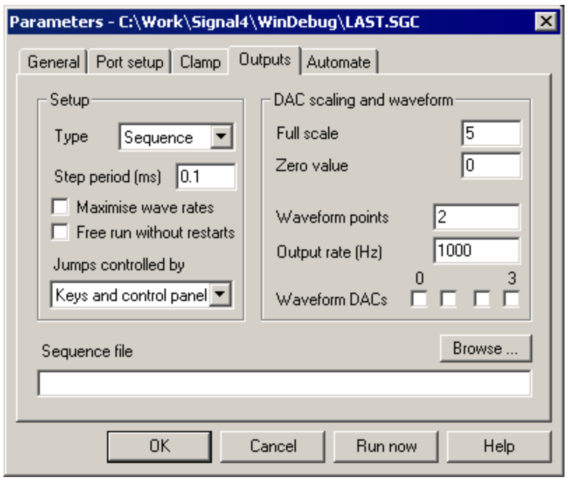
Units The units with which the DAC output scaling is specified. This is a string from 1 to 6 characters long.

On the Power1401 DACs 2 and 3 are available on pins 36 and 37 respectively of the rear-panel analogue connector. If a Signal top-box is fitted, DACs 2 to 5 will be available on the top box front panel, with DACs 6 and 7 available on pins 36 and 37 respectively of the rear-panel analogue connector. If a Spike2 top-box is fitted then DACs 2 and 3 will be available on the top box front panel, with DAC's 4 and 5 available on pins 36 and 37 respectively of the rear-panel analogue connector.

Digital outputs enable This section contains a set of checkboxes to enable and disable the individual digital outputs for use. Set the checkbox to use this digital output port, leave it clear otherwise. The fewer digital outputs are enabled for output, the more space is available for the display of each output in the pulse configuration dialog. See the Pulse outputs during sampling chapter for details of the digital outputs.

Outputs type: Sequence

This option generates pulses and other outputs using a list of sequencer instructions that are executed inside the 1401 at a specified rate. Each instruction carries out a simple function such as setting a DAC to a given value, waiting for a specified time or looping. The sequencer instructions are generated using a output sequence file; a form of text document that is edited and viewed within Signal using a Sequence view.



Setup
Type: Sequence
Step period (ms): 0.1
☐ Maximise wave rates
☐ Free run without restarts
Jumps controlled by: Keys and control panel

DAC scaling and waveform
Full scale: 5
Zero value: 0
Waveform points: 2
Output rate (Hz): 1000
Waveform DACs: 0 ☐ ☐ ☐ ☐ 3

Sequence file:

The sequencer includes 256 variables that can hold values and a table of data that can be quickly read and updated by scripts running in Signal.

The details of the sequencer language and instructions are covered separately in the *Sequencer outputs during sampling* chapter. When Sequencer outputs are selected, a number of sequencer controls are shown:

Step period (ms) This item sets the clock interval for sequencer instruction execution and thus the rate at which sequencer instructions are executed. It functions identically to the Pulses outputs Resolution (ms) control, it has the same hard limits of 0.1 microseconds to 250 milliseconds, and the recommended limits for the various types of 1401 are the same as documented for Pulses outputs. Note however that some instructions will have a speed penalty such that they will start to limit just how fast you can run the sequencer on modern 1401's. `DIV` and `RECIP` in particular may take twice as long to execute.

Maximise wave rates The design of the 1401 is such that it is possible to maximise either the timing precision with which the sequencer executes or the highest possible arbitrary waveform output rates, but not both. Normally you should leave this checkbox clear but if you want high waveform rates and are getting errors indicating the rate in use is too high, setting this checkbox may help.

Free run without restarts If this item is left clear, sequencer execution will be restarted at the first instruction at the start of each sampling sweep (specifically, at the time that the data point at time zero is sampled). This allows you to easily produce sequencer outputs at a particular time in the sampled data, but the sequencer is halted between sampling sweeps. If this item is checked, the sequencer starts running at the time that sampling starts, before the first sampling sweep is started, and continues to run until sampling is stopped.

Jumps controlled by Sometimes you may want to stop users activating sequence sections with the keyboard or from the sequencer control panel, for example when an inadvertent change in a DAC output controlling a force feedback device might hurt the subject. This item allows you to do this. The script language `SampleKey()` command can always activate sequencer sections.

Sequence file This control defines the file holding the instruction sequence to be used. You can either enter a file name directly or you can use the Browse button to select the sequence file directly.

DAC scaling and waveform

This section contains controls that define how DAC outputs are calibrated for sequencer and arbitrary waveform output. It also sets the arbitrary waveform output rate, length and DACs used. Both the sequencer and arbitrary waveform output assume that all the DACs have the same scale factor and zero setting.

Full scale	This defines the full scale output level of the DACs in the units that you wish to use, corresponding to a full scale output from the DAC. This item, along with the Zero value, is used to convert from the user units entered into the sequence into actual DAC values.
Zero value	This defines the value in your preferred units corresponding to a zero-volt output from the DACs. This value is usually 0.
Waveform points	This defines the length of the arbitrary waveform storage area, in points. Values from 2 to 10 million can be entered.
Output rate (Hz)	This defines the rate at which the arbitrary waveform is played out through the DACs. In conjunction with Waveform points, this sets the maximum duration of waveform replay. Values from 1 to 1 million can be entered. The maximum achievable rate depends on the type of 1401 and the number of DACs used.
Waveform DACs	These checkboxes set which DACs will be used for waveform output. If one DAC is selected, the waveform data consists of a list of values; for more than one DAC the data for the DACs is interleaved. Only DACs 0 to 3 can be used for arbitrary waveform output.

Pulses or sequencer?

You can define the outputs that Signal will generate during sampling in two different ways; by defining pulses using a graphical editor and by generating a sequence, a text file listing the operations that will be carried out in the form of a simple program. Pulses output is easier to get started with and supports multiple states directly, this form of outputs will be suitable for most users of Signal. You should consider using the sequencer only if you are unable to achieve the effect you require with Pulses output. The table below summarises the main differences between these two forms of output.

	Graphical sequence	Text sequence
Edited with	Built-in graphical editor	Built-in text editor
Visualise output	Yes	No
Stored as	Part of sampling configuration	Output sequence `.PLS` files
Implemented by	Drag and drop editing	Machine code like language
Ease of use	Very easy to learn and use	Takes time to learn
Flexibility	Uses pre-set building blocks	All features available
User interaction	Pulse editor while sampling	Buttons trigger jumps
Script interaction	Add, delete and modify pulses	Variables and table data
Arbitrary waveform	Data in sampling configuration	Data loaded by script
Sweeps	Locked to sampling sweeps	Can be sweep independent
Timing	Several instructions per item	One instruction per text line

Though the way in which the required outputs are defined in Pulses and Sequencer outputs are very different, the actual outputs generation is identical. When you use Pulses outputs, the pulses information is used to build sequencer data that is loaded into the 1401 and executed, so all timing requirements and limits are identical.

The following table summarises the use of the various output methods in the different sweep modes. More details can be found in the Pulse outputs during sampling and Sequencer outputs during sampling chapters.

	Pulses	**Sequence (triggered)**	**Free-run sequence**
Basic	Sweep and outputs triggered or free-run together and are of same length.	Sweep and sequencer triggered (or just start) together so that sequence repeatedly restarts.	Sequencer starts immediately and runs throughout. Can trigger sweeps or react to sweep progress.
Peri-trigger	Pulse output triggered when sweep trigger recognised, same length as remaining sweep.	Sequencer triggered when sweep trigger is recognised, so sequence restarts at trigger point.	Sequencer starts immediately, runs throughout. Can trigger sweeps via outputs or react to sweep progress.
Outputs frame	Pulse outputs triggered or free-runs. Outputs start sampling sweep at the required point.	Sequencer triggered (or just starts) and triggers the sampling sweep at the required point.	Sequencer starts immediately and triggers sweeps as required. No external trigger available.
Fixed interval	Outputs, which are triggered by internal timer, starts sweep at specified time.	N/A, simulate with Outputs frame mode with free running sequence.	N/A, simulate with Outputs frame mode with free running sequence.
Fast triggers	Single set of outputs, same length as sampling sweep. Sweep and outputs triggered or free run together.	Sweep and sequencer triggered together (or just started) so that sequence repeatedly restarts.	Sequencer starts immediately, runs throughout. Can trigger sweeps itself or react to sweep progress.
Fast fixed interval	Single set of outputs, same length as sampling sweep. Sweep and outputs triggered together by internal timer.	N/A, simulate using Fast triggers mode with free running sequence.	N/A, simulate using Fast triggers mode with free running sequence.
Gap-free	Single set of outputs, same length as sampling sweep. First sweep only can be triggered.	Sweep and sequencer triggered together for each sweep so that sequence repeatedly restarts.	Sequencer starts immediately, runs throughout. Can react to sweep progress.

DAC outputs

The 1401 DACs (Digital to Analogue Converters) produce varying voltage outputs in the range ±5 volts, these can be optionally scaled to ±10 volts if required. DACs can be used to generate pulses with arbitrary initial values and amplitudes (as long as they lie within the DAC output voltage range), they can also generate ramps, sine waves and arbitrary waveforms.

To generate complex DAC outputs and particularly for arbitrary waveform output, the DACs must update repeatedly with new output values. The faster this is done, the more accurate the output pulses or waveforms. However, very high DAC output rates may interfere with data acquisition. Signal pulse output is limited to a time resolution of 100 microseconds, which is not fast enough to cause any interference with data acquisition.

The 1401*plus* and Power1401 have four DACs, numbered 0 to 3, while the micro1401 has two (0 and 1). Both the 1401*plus* and micro1401 have the DAC outputs available on BNC connectors on the left-hand side of the 1401 front panel. The Power1401 has DACs 0 and 1 on the front panel and has DACs 2 and 3 on pins 36 and 37 respectively of the rear-panel analogue connector.

Digital outputs

The 1401 digital outputs are TTL-compatible and can be set high or low. When high they generate a voltage in the range 2.6 to 5 volts; the usual lightly loaded level is about 4.5 volts. When low they generate a voltage between 0 and 0.6 volts.

The Signal pulse output system and the output sequencer `DIGOUT` instruction control 1401 digital output bits 8 to 15 of the 16-bit digital output port. If you use the output sequencer, you can also set bits 0 to 7 with the `DIGLOW` instruction. Signal refers to both sets of outputs as digital outputs bits 0 to 7, relying upon the context to make the set in question clear. The 1401 digital outputs use a 25-way ‘D-type’ socket. This is on the right-hand side of the 1401*plus* front panel and on the rear of the Power1401 and micro1401. The Power1401 and micro1401 digital output bits 8 and 9 are also on front panel BNC sockets labelled 0 and 1 (the labels match Signal usage). If the Spike2 digital I/O expansion top box is fitted, it has front panel BNC sockets for digital output bits 10 to 15 labelled 2 to 7 (also matching Signal usage).

Digital output connections

Signal output bit number	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	GND
Pulses and `DIGOUT` pins	1	14	2	15	3	16	4	17	13
`DIGLOW` pins	5	18	6	19	7	20	8	21	13

Digital inputs

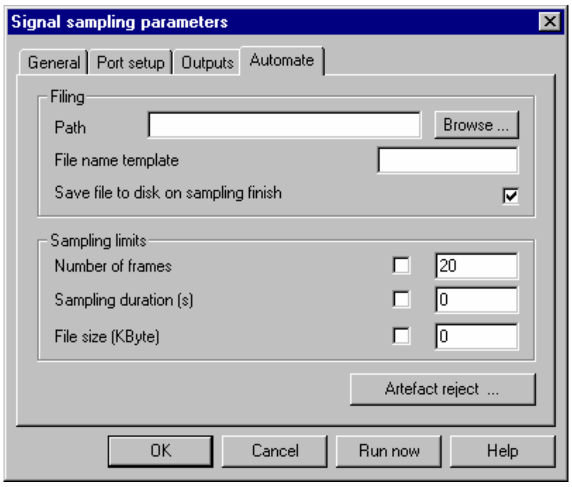
Digital input bits 8 to 15 are used in External digital states mode and in Peri-trigger sampling. Digital input bits 0 to 7 can be read by the output sequencer and are used to log digital markers. As these two sets of 8 bits are used independently, Signal refers to both sets of inputs as bits 0 to 7 and relies on the context to make the input bit set clear. The 1401 digital inputs are on a 25-way ‘D-type’ plug; this is on the right-hand side of the 1401 and 1401*plus* front panel and on the rear of the Power1401 and micro1401. The Power1401 and micro1401 digital inputs 8 and 9 are also on front panel BNC sockets, labelled as event inputs 0 and 1 (matching Spike2 usage). If the Spike2 digital I/O expansion top box is fitted, it has digital inputs 10 to 15 on front panel BNC sockets labelled as events 2 to 7 (also matching Spike2 usage).

Digital input connections

Signal input bit number	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	GND
External digital/peri-trig pin	1	14	2	15	3	16	4	17	13
Sequencer/Marker pins	5	18	6	19	7	20	8	21	13

Automation configuration

This section of the sampling configuration dialog controls the Signal automation features. There are two areas of the dialog: Filing, which controls automatic file name generation and automatic filing, and Sampling limits, which can be used to restrict the amount of data sampled or filed. There is also a button used to access the artefact rejection dialog.


Signal sampling parameters

General | Port setup | Outputs | Automate

Filing
Path: (blank) [Browse ...]
File name template: (blank)
Save file to disk on sampling finish: ✓

Sampling limits
Number of frames: ☐ 20
Sampling duration (s): ☐ 0
File size (KByte): ☐ 0

Path This sets the directory where the automatic file naming looks to produce a unique file name and where new files are saved when sampling has finished. This is different from the directory for new files set in the Preferences dialog, which sets the location for the temporary files used while sampling. If this field is blank, automatic file name generation and file saving use the current directory. You can enter a directory path directly, or use the Browse button to select a directory.

File name template This sets the template for automatic file name generation, it can be up to 22 characters long. If this is blank, automatic file name generation is disabled and normal document names (`Data1`, `Data2`, …) are used for sampled data. If a template is provided, it generates a sequence of unique file names based on a numeric code. If the template ends with one or more digits, these set the length and initial value of the numeric code. If the template does not end with digits, Signal adds "`000`" before using it. Signal increments the code until it finds an unused name in the directory set by the Path field. Thus, a template of "`testdat`" generates "`testdat000`" to "`testdat999`", while "`testdat10`" generates "`testdat10`" to "`testdat99`".

You can insert fields holding the time and date that the data was sampled by using % followed by a character code. The button marked >> to the right of the template text can be used to insert these codes, which is easier than remembering them. The codes are:

`%S`	Seconds as number (00-59)	`%M`	Minute as number (00-59)
`%H`	Hour in 24 hour format (00-23)	`%d`	Day of month (01-31)
`%m`	Month as number (01-12)	`%y`	Year without century (00-99)
`%Y`	Year with century, e.g. 2007		

If you use one of these codes at the end of the template, Signal will add a '_' so that the unique filename generation does not change the date or time information. If a file name template is set, the generated name is used automatically when the data file is saved without the user being prompted to confirm the name. A different name can be specified using the File Save As command.

Save file to disk If this checkbox is set, the new data file will automatically be saved to disk when sampling finishes. If automatic filename generation is in use, the generated filename is used, otherwise the usual prompts for a file name from the user are generated.

Sampling limits This part of the dialog controls three limits; Number of Frames, Sampling duration and File size. Each option has a checkbox to enable the limit plus a field for entry of the limit value. If the checkbox for a particular limit is clear, or if the corresponding limit value is set to zero, then that limit is disabled. Note that all of these limits cause sampling to stop, not finish, sampling can still be continued after a limit is reached.

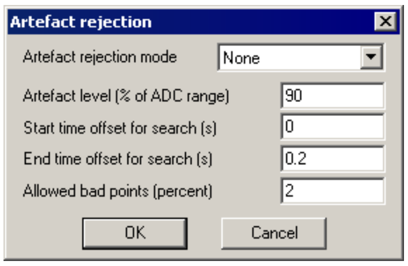
In addition to these user-defined limits, Signal has a built-in sampling limit based on the available free disk space. If the available free disk space drops below 0.5 Mbytes, sampling stops automatically. This limit cannot be disabled as it is important not to allow hard disks to get completely full, both because this can slow down file accesses considerably and also because of the trouble a full disk gives to an operating system such as Windows 95 that uses spare disk space for virtual memory management. The user-defined limits are:

Number of frames	If this limit is enabled, sampling stops when the set number of frames have been written to the data file.
Sampling duration	If this limit is enabled, sampling stops when the set time has passed since sampling was started.
File size (Kbyte)	If this limit is enabled, sampling stops automatically when the file size reaches or exceeds the set size.

The Artefact reject … button opens a dialog that allows you to configure artefact rejection. This provides mechanisms for automatically rejecting or tagging sampled frames if they contain an artefact, normally caused by stimulation.

Artefact rejection

The artefact rejection dialog can be used to configure Signal to automatically examine newly sampled data and, if the data has reached the ADC limits, to reject or tag the new frame. Artefact rejection is important when generating averaged evoked responses, particularly from the EEG, where artefacts often occur and where the mathematical rigor of the averaging process will be affected by the presence of signals at the ADC limits.


Artefact rejection

Artefact rejection mode: None

Artefact level (% of ADC range): 90

Start time offset for search (s): 0

End time offset for search (s): 0.2

Allowed bad points (percent): 2

The Artefact rejection mode item controls what form of artefact rejection to apply. It can be set to None for no artefact rejection, Tag frames to label frames with artefacts, or Reject frames, to discard frames with artefacts. The Artefact level is the percentage of the ADC range outside which data values will be regarded as an artefact.

The next two items set the time range for the search for artefacts; these are specified as offsets from the start of the data rather than as absolute frame times. The final item, Allowed bad points (percent), sets the limit before the frame is rejected or tagged, allowing you to avoid rejecting frames with a trivial amount of bad data.

Unless the mode is None, each sampled frame of data is scanned for artefacts. All waveform channels are scanned over the time range specified and the number of points that exceed the Artefact level are counted. If the number of bad points, expressed as a percentage of the total points scanned, exceeds the limit for allowed bad points then that frame is automatically rejected or tagged as appropriate.

Creating a new document

Once you have set up the sampling configuration you can create a new sampling data document. Either click the Run Now button in the sampling configuration dialog or the Run Now button on the Toolbar, or Select New from the File menu, then Data Document for the file type.

The exact appearance of the new document view varies, depending on the configuration. The name for the new document will either be `Data1`, `Data2`, and so on or a name produced from the file name template if one is available. You can customise the view by adjusting the x axis, y axes, channels and other aspects of the view.

Frame zero

A sampling document is different from other types of data documents because it starts at frame zero, while all other types start at frame 1. Frame zero is a special frame that holds the transitory data for the sweep currently being sampled; frames 1 onward hold data that has been written to the new file. Sampling is a cyclical process of collecting a new sweep of data into frame 0, deciding if it is to be written to disk and writing it if necessary, clearing frame 0 and then starting off the next sweep. This process continues until enough frames have been written to disk or until sampling is stopped.

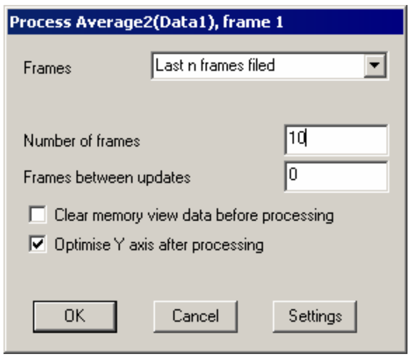
Data is shown in frame zero for as long as it is the most recent data. Frame zero is cleared when pulse outputs for the next frame start or when sampling of the next frame begins. For Basic and Peri-trigger sweeps, this means that data is displayed right up until data from the next sweep starts to be drawn. For Outputs frame or Fixed interval sweeps, the frame zero data will be cleared when the pulse output for the next frame starts and the new data starts to be drawn when the sampling is triggered. There may be a period when no data is shown, which provides the user with a clear indication that the next sweep has started. This is valuable because once the pulse output or sampling has started, the Accept/Reject button in the control panel cannot be used on the previous sweep of data.

For fast sweep modes (Fast triggers, Fast fixed int or Gap-free) the most recently sampled sweep is displayed, starting when the first data points for the frame are sampled.

The small floating window is the *Sampling control panel*. It contains buttons and other controls to interact with the sampling. Sampling will not start until you click Start in this control panel (the Sample menu duplicates the panel controls). If the Event 1 start box is checked, sampling waits for an external signal after the start button is pressed.

Online analysis

You can create result and XY views to hold the result of analysis during data acquisition in exactly the same way as when working offline; using the Analysis menu New Memory View and New XY View commands. The dialogs through which you specify how the analysis is to be performed are exactly the same as when offline, however the Process dialog that specifies what data is to be processed is different from it's offline equivalent.


Process Average2(Data1), frame 1

Frames: Last n frames filed

Number of frames: 10

Frames between updates: 0

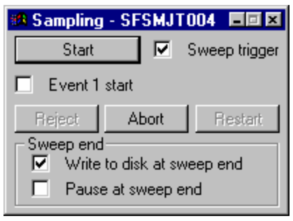
☐ Clear memory view data before processing

☑ Optimise Y axis after processing

This dialog controls which frames to include in the processing and how to update the memory view holding the analysis result (see the *Analysis menu* chapter for a full description of this dialog). The most common mode is All filed frames, with an update every 0 frames (every frame). This dialog closes when you select either OK or Cancel. You can recall it with the Process command in the Analysis menu or by right-clicking in the result view.

Sampling control panel

The sampling control panel holds several buttons and checkboxes used to control and interact with data sampling. When the control panel appears it looks like the picture to the right. Click Start to begin sampling, or Abort to give up immediately. Once sampling has started the buttons change but most checkboxes are always present and can be changed at any time during sampling.



The Event 1 start checkbox allows you to trigger the entire sampling process externally. If this checkbox is set, clicking on Start will not start sampling directly, it enables the start of sampling on an event 1 pulse. This allows precise control of the time of the start of sampling; the time from which the frame absolute start times are measured. While Signal is waiting for an E1 start pulse, the Start button will flash 'Waiting for E1'. The E1 input is on the 1401*plus* front panel and is pin 2 of the rear panel Events connector for Power1401 and Micro1401 with a suitable ground on pin 9. Connect the E1 input to ground (or pull it below 0.6 V) to start sampling.

The Sweep trigger checkbox is initially set from the Sweep trigger item in the sampling configuration. It enables or disables triggered sweeps, while Signal is waiting for a sweep trigger the checkbox text will show 'Waiting (TR)'.

The checkboxes in the Sweep end section of the dialog control what happens at the end of a sweep of sampling and how data is written to disk. If the Write to disk at sweep end checkbox is set, then each frame zero is written to the disk file when the sweep finishes. If the checkbox is clear, then the frame is not written. You can override this behaviour for individual sweeps using the Accept/Reject button in the sampling control panel.

The Pause at sweep end checkbox controls whether a new sweep is started automatically once the previous sweep has ended. If the checkbox is set then sampling will pause until the Continue button is pressed, or until the checkbox is cleared again, allowing the user to pause for a while or to inspect the data and accept or reject each frame. If the checkbox is clear then the next sweep starts immediately.

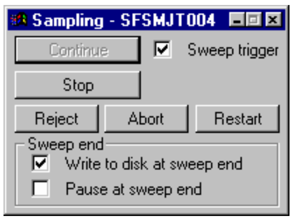
Sweep end checkbox combinations

Write to disk	Pause at	Effect
No	No	Continuous sweeps for signal monitoring
Yes	No	Continuous sweeps written to disk
No	Yes	Interactive sweeps written to disk if accepted
Yes	Yes	Interactive sweeps removed from disk if rejected

The sampling control panel can be hidden or shown using the Sampling menu, the Signal toolbar or the pop-up menu provided by right-clicking the mouse on unused parts of the Signal window. It can also be hidden by clicking on the **x** button at the top right of the control panel, or docked at any edge of the Signal window.

During sampling

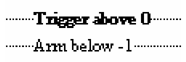
Once sampling has started, the sampling control panel changes to show buttons suitable for the sampling process. If Event 1 start was checked before you click Start, the word Waiting flashes until a suitable signal is applied to the E1 input to enable sampling. This is distinct from the Sweep trigger. Use this method to synchronise the start of sampling with an external event. Sampling starts within 1 or 2 µs of the external event signal. The buttons available are:



Continue	This is labelled Start before data capture starts. It is only enabled when sampling is paused at the end of a sweep. When you click on the button, sampling of the next sweep is enabled. If the Sweep trigger box is checked, the 1401 system waits for the sweep trigger before starting to collect data.
Stop	This is displayed after data capture starts. If you click on this button, the data capture stops, in the same manner as if one of the Automate limits was reached. Once the data capture has been stopped, no more sweeps will be collected, but it is possible to resume sampling again.
Accept	Clicking on this button writes unwritten frame 0 data to disk. If frame zero is still being collected, it overrides the Write to disk at sweep end checkbox so that the frame is written to disk at sweep end; it does not affect subsequent sweeps. If the frame has been collected and sampling is paused, this writes the frame to disk immediately. The button acts upon a sweep up until the point that the next sweep is triggered or pulse outputs for the next sweep begins.
Reject	If Write to disk at sweep end is checked, or sampling is paused at the end of the sweep and frame 0 has been written, then the label on the Accept button changes to Reject. Clicking on Reject either overrides the Write to disk at sweep end checkbox to cause the currently sampling frame not to be written automatically, or removes the current frame 0 data from the end of the data file, as appropriate. This button acts on a frame until the next frame is triggered or pulse outputs for the next frame starts.
Abort	This button abandons sampling and discards the file. You can use this button before sampling starts or while sampling. You are warned if this will lose saved data.
Restart	This button is available once sampling starts. It stops sampling, discards any saved data, then waits for you to start sampling again with the same document. You are warned if this will lose saved data.

Other interaction with sampling

If a keyboard marker channel is being sampled, you can insert markers into the sampled data by pressing keys on the keyboard. You can do this if the new data view is the current view, or if the sampling control panel is current. The current view or window has a highlighted title bar, you can make a view current by clicking on it. If the sampling control panel is current, you should be careful about pressing the space bar, which is equivalent to pressing whichever button is currently highlighted, or pressing Enter, which is equivalent to pressing the Start/Continue button.



If you are using Peri-triggered sampling in any of the analogue trigger modes, frame zero of the sampling document displays a pair of horizontal cursors that indicate the positions of the two trigger thresholds. These cursors are displayed on the highest-numbered waveform channel (which is the last port in the ADC ports field), as this is the trigger channel. For +Analogue and -Analogue trigger modes, the cursors show the trigger level and the trigger level minus (or plus) the hysteresis, this latter level is shown as the Arm level. For =Analogue trigger mode the two separate trigger levels are shown. During sampling you can adjust the trigger levels used by moving the cursors; they cannot be moved to a different channel as the trigger channel is fixed.

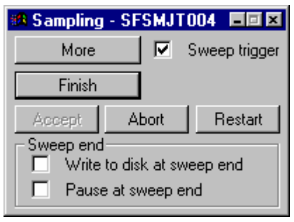
If you use the keyboard command `Ctrl+PgUp`, Signal will switch to displaying the last frame that was saved to disk.

If you are using a programmable signal conditioner such as the CED 1902 programmable amplifier then you can use the signal conditioner control panel to adjust the amplifier settings – see the *Programmable Signal Conditioners* chapter for more details of these. Note that, if you alter the amplifier gain or offset settings, the channel calibration will be adjusted in such a way to keep the incoming data correctly calibrated – cancelling out the gain and offset changes – so you will not normally see any change in the displayed data. The Edit menu Preferences dialog has a setting (in the Sampling section) to allow the display range shown to be adjusted, if required, to reflect changes in the ADC range caused by signal conditioner changes.

You can tag data frames as they are sampled by pressing `Ctrl+T`, or frames can be automatically tagged by the artefact detection system.

Most standard display manipulation mechanisms work in exactly the same manner online, but frame overdraw mode works differently on frame zero; it never erases old data but just redraws it in grey and the frame display list is ignored. This provides a very nice ‘storage oscilloscope’ style display, but if any part of the view is redrawn the previous traces are lost. You can use the Edit menu Clear command to erase all the previous traces.

Stopping sampling



Sampling can be stopped by clicking on the Stop button or by the sampling limits being reached. When sampling is stopped, Signal is in between sampling and finishing sampling. The sampling control panel is still present, but the buttons have changed:

More This is the button previously labelled as Start or Continue. Click it to resume sampling as if the Stop button had not been pressed. If sampling stopped due to the frame count reaching a limit then pressing More resets the frame count and sampling runs until the count again reaches the limit, otherwise the limit is disabled and sampling continues indefinitely.

Finish This is the button previously labelled as Stop. If you click on this button, the data capture is terminated and the sampling control panel disappears.

Finishing sampling

Click on the Finish button to end sampling. If the sampling configuration has automatic saving to disk enabled, the new data file will be saved at this point, using either the automatically generated filename or a name entered by the user as appropriate. You will also be prompted for a comment for the new file, if this feature is enabled. Once the sampling has actually shut-down, the sampling control panel is removed. In the new data view, frame zero of the data document disappears and the view changes to show frame 1 if it was previously showing frame zero. If there are no saved frames, the data document and view are destroyed, giving the same effect as pressing Abort.

Saving new data

A sampling document that has stopped sampling is essentially the same as a document loaded from disk. However, unless you are using automatic saving, the data has not yet been saved in a permanent disk file, though it is stored on disk. To keep the data, you must save the new data using the File menu Save command. If you try to close the document view without saving the data Signal will check that you really want to do this.

Data documents 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 use the File menu New command, Signal creates a temporary file in the directory specified in the Edit menu Preferences dialog. If you do not specify a directory, the location of the temporary file is system dependent. When you save a new data document after sampling, Signal moves it to the directory you specify.

Saving configurations

To avoid setting the sampling, analysis and screen configuration each time you sample, you can save and load sampling configurations from the File menu. The configuration includes:

The basic sampling parameters in full, including port information for all ports.

The position of all windows associated with the new file (including duplicated windows and the various control panels).

The displayed channels and display modes of the channels in the windows.

The outputs to be generated during sampling.

The multiple states information, including the protocols.

The processing parameters and update modes of all memory views.

If sampling ends without failing or being aborted, then Signal saves the configuration as the file `last.sgc`. The saved configuration is used the next time data is sampled. When Signal starts, it searches for and loads the configuration file `default.sgc`. If this cannot be found, it uses `last.sgc` so generally you will get the same sampling setup as was used last time. These files are kept in the directory from which Signal ran. If sampling fails or is aborted, the sampling configuration switches back to the configuration in use before you started sampling.

It is a good idea not to rely on `last.sgc` as this may have been changed by someone else doing some sampling, instead you should save important sampling configurations to disk using suitable names and reload them before starting an experiment. You can make this much easier by using the Sample Bar (available from the Sample menu) which holds sampling configurations so that they can be quickly loaded using a single mouse click.

Sequence of operations to set and save the configuration

This describes a sequence of operations that build a new sampling configuration from scratch. You will find that once you have built a few configurations, it is 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 using the Sampling menu or toolbar and set the sampling configuration, limits and port information.

2 Press Run Now from the configuration dialog or press OK and select the New command in the File menu and choose Data Document.

3 Arrange the new file view as you require and add or remove duplicate windows.

4 Use the Analysis menu New Memory view and New XY view commands to add result views as required and set their update mode and position on screen.

5 You can use the File menu Save Configuration As command to save the configuration to disk at this point.

6 Sample, adjusting any positions, display configurations and so forth as required.

7 If sampling has finished without being aborted or failing, the sampling configuration is held in memory for use the next time you sample. You can use the File menu Save Configuration As command to save the configuration used to disk.

You can re-use a saved configurations by loading it with the File menu Load Configuration command before you use the File menu New command to create a sampling document.

5 Sampling with clamp support

Introduction

Signal incorporates a number of specialised features directly supporting whole-cell and single-channel clamping experiments. In order to avoid confusing users who are not interested in this type of experiment, these features can be hidden using a checkbox in the Edit menu Preferences dialog (Clamp page), the initial state of this option is set up by Signal when starting if it detects that it has not already been set. The features controlled by this option are the online clamping support as described below, leak subtraction analysis and the generation and analysis of idealised single-channel traces.

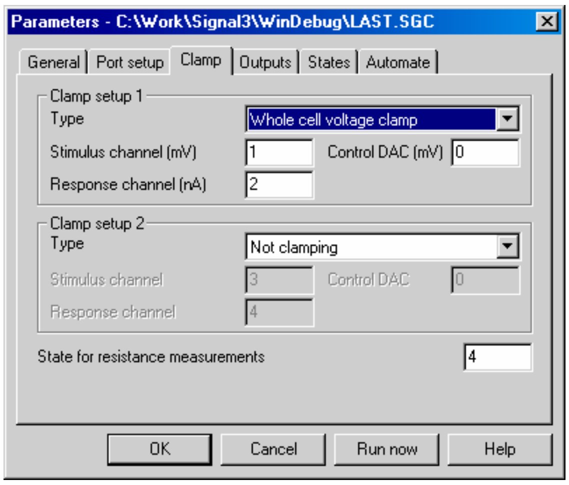
Unlike many applications used for clamping experiments, Signal is a very general-purpose program that can be used for many other types of experiment. This can make it harder for new users setting up clamping experiments to find the aspects of Signal that they need. In addition to this chapter (and general familiarity with Signal), clamping researchers should look at the amplifier telegraph system (*Amplifier telegraphs* chapter), the pulses configuration dialog (*Pulse outputs while sampling*) and dynamic outputs multiple states (*Sampling with multiple states*). In addition to the clamp specific analysis mechanisms the general *Analysis menu* documentation should be useful as all of the general analysis features in Signal can be used on clamp data. The trend plot, measurement generation and curve fitting sections of the analysis menu may be of particular interest. Specialised data acquisition and analysis can be carried out using the Signal script language.

Online clamp support features

The online clamp support within Signal consists of a specialised page within the sampling configuration which defines clamp-specific information plus mechanisms for online analysis and experiment manipulation that make use of this information to provide automatic resistance measurements, holding-potential (or current) controls and a membrane analysis display. There are a number of limitations to the types of experiments that can be used with the clamping support which are described below.

Sampling configuration

The clamp page in the sampling configuration is used to configure Signal online clamping support. This page allows two clamp setups to be defined along with a state (set of pulse outputs) to be used for membrane analysis. A clamp setup defines data file channels that will hold the current and voltage data (so that analysis can find the correct data) and also checks the units for these channels (so that current and voltage values can be scaled correctly to amps and volts). In addition a setup defines the DAC used to control the membrane current or voltage (so that the holding potential and pulses can be manipulated).


Parameters - C:\Work\Signal3\WinDebug\LAST.SGC

Clamp setup 1
Type: Whole cell voltage clamp
Stimulus channel (mV): 1
Control DAC (mV): 0
Response channel (nA): 2

Clamp setup 2
Type: Not clamping
Stimulus channel: 3
Control DAC: 0
Response channel: 4

State for resistance measurements: 4

The State for resistance measurements item sets a state number that will be used for resistance measurements and by the membrane analysis (see below).

The Experiment type item can be set to Not clamping to disable use of that clamp setup or to Whole-cell or Single channel, each either voltage clamp or current clamp. If both setups are set to Not clamping then all of the online clamping support will be disabled.

The Stimulus channel and Response channel items set the data file channels that will hold the corresponding data. For voltage-clamp experiments the stimulus channel is voltage and the response channel is current, for current-clamp it is the other way around. These items can be set to any valid channel number from 1 to 80. If the channel specified does not exist in the sampling configuration then the text for the incorrect item will be shown in red. If the channel does exist then the channel calibration will be checked for valid units. The units for a voltage channel must contain the character `V` and start with either `V` or one of `M`, `U`, `N`, `P` or `F` (lower-case characters are also allowed) implying milli-, micro-, nano-, pico- and femto-volts respectively. Plausible legal voltage units would thus include `'V'`, `'mVolts'` or `'uV'`. The initial character is used to scale the measured values to actual volts during online analysis. Similarly the units for a current channel must contain the character `A` and start with either `A` or one of `M`, `U`, `N`, `P` or `F`, `'nA'` and `'pAmp'` are obvious examples that would be acceptable.

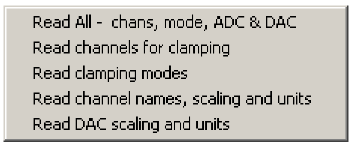
The Control DAC item sets the DAC number that is used to control the stimulus. The DAC specified must be from 0 to 7 and be in use in the pulse outputs. The units set for the DAC must be appropriate for voltage or current according to the experiment type, or the DAC text will be shown in red.

The State for resistance measurements item specifies a state (a set of pulse outputs) to be used for measurement of resistance and other membrane analyses. This value is not checked within the dialog, but it must either be zero (if multiple states are not being used) or from zero to the maximum state number in use. Whenever this state is used during sampling, Signal will automatically calculate and display the membrane resistance. In addition, this state will be automatically selected when the membrane analysis dialog is used. So, with state 4 set as the state for resistance measurements as in the picture above, whenever state 4 is used during sampling the membrane resistance will be recalculated, displayed and saved in the data file. Signal states are quite a complex topic; for more information on them, see the chapter *Sampling with multiple states*. The clamp support expects that you will be using Dynamic outputs multiple states so you can ignore the extra complexity of the other multiple states modes.

A suitable stimulus pulse for resistance measurements should be defined for the control DAC in the specified state, this pulse can be added to the pulse outputs or modified during sampling should this prove necessary. If there is more than one pulse for this state the pulse to be used for measurements should be labelled by setting the pulse ID to 'RM', otherwise the first pulse in the outputs will be used. Currently, a square pulse (constant or varying amplitude) or a square pulse train should be used. Varying duration pulses, sine waves or ramps and arbitrary waveforms are not currently supported.

Multiclamp 700 integration

If you have installed support for the MultiClamp 700 amplifier telegraphs, a button will be available at the bottom of the clamp page to read the current MC700 settings and use them to set up clamping information according to the MC700 setup. This option can read the ADC ports set in the MC700 configuration page and use them to set the channels used in the clamping configuration, the clamping mode set in the MC700 as well as the relevant ADC and DAC calibrations - the only thing you have to set yourself (apart from the MC700 settings in the ports configuration page) is the DAC that is connected to the amplifier - so it should be much easier to get everything set up correctly. These options are only available when the MC700 commander software is also running, as Signal needs to be able to read back the amplifier settings from the commander software.


Read All - chans, mode, ADC & DAC
Read channels for clamping
Read clamping modes
Read channel names, scaling and units
Read DAC scaling and units

The options available from the MC700 button are `Read channels for clamping`, which sets the stimulus and response channel items up according to the current MC700 settings (note that the ADC ports specified must be used in the sampling configuration for Signal to be able to set the channels), `Read clamping modes` which retrieves the clamping mode used in the relevant sections of the MC700, `Read channel names, scaling and units` which retrieves this information for the waveform channels generated from the relevant ADC ports and `Read DAC scaling and units` which sets up the specified DAC output. An extra option, `Read All`, reads all of this information at once.

These options (apart from `Read channels for clamping`, which is extra) match the reading & use of MC700 settings which can be carried out when sampling starts and as well as allowing quick & correct setup, can be used to check what the current amplifier settings are and, because the DAC scalings can be quickly set up, makes it easier to set up the output pulses that you want to use.

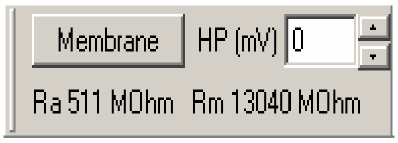
Other sampling configuration considerations

As mentioned above, because Signal is a very general-purpose program it is possible to produce a sampling configuration that does not allow the clamping features to operate correctly, though we have tried hard to make the software as flexible as possible. A checklist of aspects of the sampling configuration that you need to ensure are correct is:

- Any sweep mode can be used but as the two fast sweep modes do not allow multiple states or changes to the pulse outputs they are rather limiting and will not work well. Gap-free sampling can be used to avoid lost data, particularly for single channel experiments, but again only one fixed set of pulse outputs are available.
- External convert sampling cannot be used as this prevents Signal from determining the time-course of events.
- The stimulus and response channels specified must exist (be generated by the configuration) and the ports used to log the channels must use appropriate units as described above. The channel calibration scaling factors must be correctly set up for the amplifier gains, and any telegraph settings correct, or data analysis will give incorrect results.
- Pulses outputs must be used with absolute levels turned off if you want the holding potential controls to work as expected. Sequencer output can be used for specialised experiments but it will not work with the built-in clamping support.
- All the DACs specified for stimulus control must be enabled in the outputs and calibrated using suitable units. The DAC scaling factors must be correctly set for the amplifier control inputs or the stimuli generated will be incorrect.
- If multiple states are used (which is almost always necessary), these should use dynamic outputs variation.
- The state specified for resistance measurements must exist. State zero always exists, even if multiple states are not used.

Signal checks the sampling configuration for incompatibilities with the requirements of clamping at the start of sampling and will generate an error message if problems are detected.

Running a clamping experiment



When sampling with at least one enabled clamping setup the clamping toolbar is created in addition to the standard Signal sampling toolbars. The clamping toolbar contains separate controls and information for each enabled clamping setup and can either be floating or docked to any edge of the Signal window. The clamp toolbar contains three items; controls for the holding potential, a text display area and a button marked Membrane used to provide the membrane analysis dialog.

The holding potential (or holding current, for current clamp) controls are used to adjust the baseline level of the stimulus DAC output (as set in the sampling configuration). The initial value of the holding potential is taken from the initial (baseline) level of the appropriate DAC in state zero. Whenever the holding potential is changed, the baseline level for the DAC is set to the new value in all states, the holding potential is also initialised for all states at the start of sampling. Note that, when editing the holding potential directly, it is necessary to accept the new value by pressing tab or otherwise moving away from the edit field for the change to take effect.

You can step the holding potential up and down by clicking on the spinner to the right of the edit field, by using the keyboard up- and down-arrow keys when the text cursor is in the edit field and by rotating the mouse wheel when the text cursor is in the edit field. All of these changes take place immediately. For all of these methods, the step size is 1 millivolt normally, 5 millivolts if the Shift key is held down and 20 millivolts if the Ctrl key is down (0.5, 2.5 and 10 pA for current clamp).

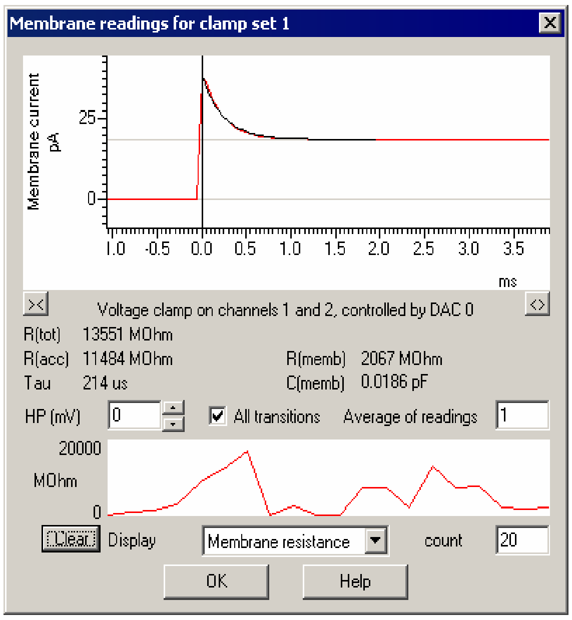
At the bottom of the clamping toolbar is a text field where the electrode access and membrane resistances are displayed. This field is automatically updated whenever the state specified in the sampling configuration is used, you can select the state manually or it can be used as part of normal states cycling or protocol execution. Resistance measurements are carried out regardless of whether sampled data is being written to disk.

Membrane analysis

The `Membrane` button is used to provide the membrane analysis dialog; this dialog uses the state for resistance measurements specified in the sampling configuration to carry out a more complete analysis of membrane properties. When it is used, sampling is automatically switched to the state specified for resistance measurements and writing to disk is turned off.

The membrane analysis measures and displays the total resistance, access and membrane resistances, the time constant for the decay of the capacitance transient and the membrane capacitance (see below for details of the analysis).

At the top of the dialog is the waveform display area. This shows a slice of waveform from the response channel at the time of the stimulus pulse. If the analysis has been successful the fitted curve corresponding to the capacitance decay will be drawn on top of the waveform data. The two buttons labelled >< and <> (below the waveform display area) can be used to increase and decrease the time range shown.



Voltage clamp on channels 1 and 2, controlled by DAC 0

R(tot) 13551 MOhm
R(acc) 11484 MOhm R(memb) 2067 MOhm
Tau 214 us C(memb) 0.0186 pF

HP (mV) 0 ✓ All transitions Average of readings 1

Display Membrane resistance count 20

Immediately below the waveform display area is an area used to show the results of the most recent analysis. These results are updated after every frame, the results can optionally be the average of all stimulus transitions or just one and can also be averaged over a number of sweeps if desired.

The holding potential controls adjust the membrane holding potential (or current) in exactly the same way as the equivalent controls in the clamping control toolbar.

The All transitions checkbox, when checked, causes the response to all of the edges of the selected stimulus pulse to be analysed rather than just the first edge (the one shown in the waveform display). If more than one edge is analysed, the results shown in the text area are based upon the averaged results for all edges. This control also affects the results shown in the graphical display of a measurement over time.

In addition to averaging across all stimulus transitions, you can average the analysis results that are displayed over a number of sweeps by using the Average of readings control. This can be set to any value from 1 to 1000.

The area below these controls provides a display of a selected measurement over time. The Display selector selects the measurement to be displayed from those available, while the `count` control sets the number of measurements shown over the width of the display area, from 10 to 1000. The values displayed in this graph are updated with data from every sweep (the Average of readings control has no effect, but the All transitions checkbox operates). The Clear button deletes all saved measurements and restarts the display – this can be particularly useful if an aberrant value causes the automatic Y scaling to show too large a data range.

When the membrane analysis dialog is closed the sampling state sequencing system and writing to disk are both restored to their previous state at the time that the dialog was displayed.

Changing pulses

The pulses dialog can be used to directly edit the output pulses, including the pulse used for resistance measurements, at any point while the experiment is in progress. Any changes made to the pulses while the membrane analysis dialog is in use will not cause the membrane analysis to fail in a destructive fashion, but deleting the pulse for resistance measurements or changing it to a type that is not usable for resistance measurements will prevent the membrane analysis from being carried out.

Resistance data

The total, access and membrane resistance values, along with the membrane capacitance, are stored in the new data file as frame variables. The names of the variables are `'RTotn'`, `'RAccn'`, `'RMembn'` and `'CMembn'` respectively, where n is 1 for the first clamp set and 2 for the second. The values stored will be zero for data where the relevant clamp set is not being used, 1.0 when a resistance measurement has not yet been taken and the relevant value (in MOhms or pF) generated by the last resistance measurement otherwise. These values can be retrieved using the Signal script language, later versions of Signal will be able to make use of them directly.

Analysis methods

Resistance and other membrane measurements are generated by analysis of a transition; a step change in stimulation (clamped voltage or current) level with the new level being maintained for a reasonable period. For a simple square pulse there are two transitions; one at the start of the pulse and one at the end, for a pulse train there are a sequence of transitions. Analysis is carried out on the first suitable pulse found in the outputs for the specified state; you can force a particular pulse to be used by setting the pulse ID to 'RM'.

To measure total resistance, the post-transition level is measured during the final quarter of the stimulus pulse (for transitions at the end of a pulse the previous pulse width is assumed). The delay of ¾ of the pulse width is intended to allow the capacitive transient to settle and can be adjusted by altering the stimulus pulse width. The baseline (pre-transition) level is measured over the same amount of time just before the transition. Measurements are taken from both the stimulus and response channels allowing the resistance to be easily calculated by using Ohms law.

After the stable levels before and after the transition have been measured, voltage clamp data can be analysed further. The decay phase of the overshoot is analysed by fitting a double exponential curve to the current data. The data used for the fitting is 3 milliseconds in duration, starting at the point of maximum overshoot. If the double-exponential fit is successful the shorter time constant found (corresponding to the neuron body) is used, otherwise a single-exponential fit is tried. Having got a time constant for the decay, the area underneath the overshoot is also measured to give the total charge.

Having measured the total charge and the amplitude of the voltage step, the membrane capacitance can be calculated as charge/delta volts. We are then able to estimate the access resistance by Rs = time constant/capacitance, which allows us to separate out the membrane and access components of the total resistance.

For current clamp experiments, only the overall resistance is measured.

When all transitions are being analysed, the results of analysis of each transition are averaged to give a single set of measurements for the sweep.

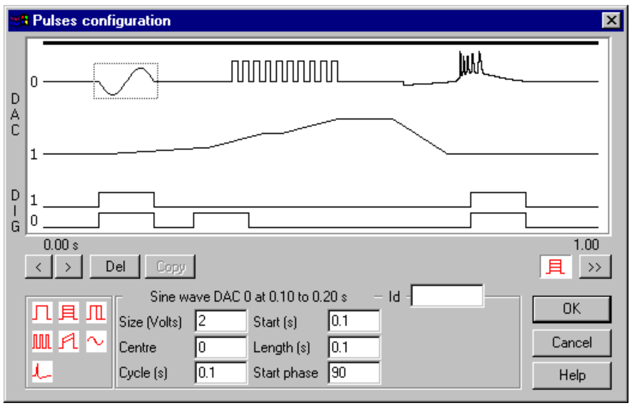
6 Pulse outputs during sampling

Introduction

In the *Sampling data* chapter you will have encountered pulses as an option within the outputs section of a Signal sampling configuration and details of the connections for these outputs. Signal can generate outputs from the *1401plus*, Power1401 and micro1401 using the DACs and the digital outputs. This chapter describes the pulses configuration dialog and its use during sampling.

Signal pulse output, like Signal data acquisition, is arranged as fixed-length frames. Depending upon the sampling sweep mode, the pulse outputs frame may be the same length as the sampling sweep, longer or shorter. In all circumstances, the pulses output are fixed in time relative to the sampling sweep. In Basic mode and the three fast-trigger modes the pulse output frame starts at the same time as the sweep (triggered or un-triggered) and is of the same length. In Peri-triggered mode, the pulse output frame starts at the time of the trigger (which can be before or after the start of the sampled data, depending upon pre-trigger points), and again runs to the end of the sampling sweep. In Outputs frame and Fixed interval modes, the pulse output frame can be set to any length greater than or equal to the sampling sweep, and the sampling sweep starts at a defined point after the start of the pulse output frame.

Pulses dialog



Sine wave DAC 0 at 0.10 to 0.20 s — Id

Size (Volts)	2	Start (s)	0.1
Centre	0	Length (s)	0.1
Cycle (s)	0.1	Start phase	90

If you press the Configure Pulses button in the outputs page in the sampling configuration, Signal will display the Pulses dialog which allows you to view and edit the pulses.

This dialog can also be used to control and adjust the pulses while Signal is sampling, in which case it is accessed using the sampling menu, the Signal toolbar or by using the right mouse button popup menu.

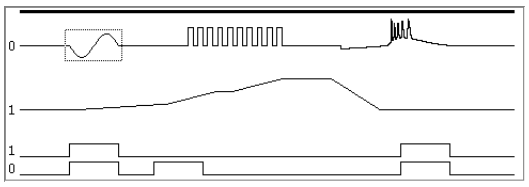
The Pulses configuration dialog can be used to define square pulses and pulse trains, ramps, sine waves and arbitrary waveforms which will be output during sampling. Many of these pulses can automatically vary by incrementing their amplitude or duration by fixed amounts. This provides a straightforward way of generating a repeating set of pulses from one definition. If you are using Outputs frame or Fixed interval sweeps, this dialog is also used to set and control the length of the pulse outputs frame, the start of the sampling sweep within the outputs frame, the sampling sweep ADC points if Variable points are enabled, and the fixed interval sweep interval and interval variation.

You can resize the Pulses dialog by dragging the bottom right-hand corner of the dialog.

Pulses dialog layout

The dialog is divided into four sections:

Display



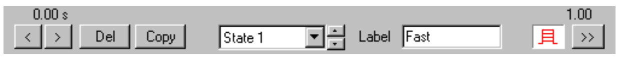
This occupies the upper part of the dialog and shows the currently defined outputs as graphical traces. At the top of the display is a solid line that shows the portion of the pulse output frame period that is covered by the sampling sweep. If you are using Outputs frame or Fixed interval sweeps, this line is thin where no sampling is taking place.

The outputs themselves are displayed starting with DAC zero at the top and finishing with the digital outputs at the bottom, with the higher-numbered digital outputs first. Only traces for enabled outputs are displayed, they are labelled with the DAC or digital output bit number. If you double-click on a trace in the display area it will expand to use all of the space available for traces, hiding all other outputs. Double-clicking on the display again will restore the display of all traces.

The display also includes a dotted rectangle drawn around one of the pulses, or around the initial level of an output. This indicates the currently selected pulse whose numerical parameters are displayed at the bottom of the dialog (see the *Values* section below). You can select a pulse by clicking on it with the mouse. Alternatively you can toggle the selection through the various pulses on an output using buttons in the *Controls* section. You can change the start time of a pulse by dragging it around the display area.

Experiment with clicking on pulses to get a feel for how pulse selection behaves. Note how the display of pulse parameters changes as you select different pulses.

Controls This is the central area of the dialog, which contains a number of controls used to interact with the dialog.



At the top of the controls area at the left-hand side is text such as "0.00 s" indicating the time for the start of the pulse output frame. The right-hand side holds similar text showing the time for the end of the pulse frame. When a pulse is being dragged about the display, the current pulse position is shown level with these numbers. All of these times are shown in the currently selected time units. Below these time indicators are a number of controls, from the left these are:

These buttons are used to toggle through the pulses on the current output trace in forwards or reverse time order, selecting each pulse available in turn.

This button is used to delete the currently selected pulse.

This button is only enabled when multiple states controlling dynamic outputs are in use (see the *Sampling with multiple states* chapter). It is used to copy sets of pulse data from the currently displayed state to other states. See below for more details on using this.

This pair of controls is only visible when multiple states controlling dynamic outputs are in use (see the *Sampling with multiple states* chapter). They are used to select the state shown by the dialog; you can display and edit pulses from only one state at a time.

This control is also only visible when multiple states controlling dynamic outputs are in use (see the *Sampling with multiple states* chapter). It is used to set a label for the currently displayed state; a descriptive name for the state up to 10 characters long that will be used in the state control bar buttons and elsewhere within Signal. If you leave this field blank the standard state names `"Basic 0, State 1 .."` will be used.

Click on this control to see the effect of pulse variations. While you hold down the mouse button on this control, the limits to pulse variations plus three intermediate values are displayed.

Click on this button to animate the display to show the effect of pulse variations. The display updates to show the effect of each variation in turn. Click again on the button to turn animation off.

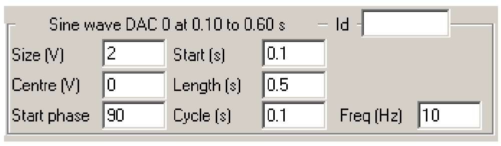
Pulses

This is the area at the bottom left of the dialog. It holds pulse icons which can be dragged into the display area to add a new pulse into the outputs. Each icon represents a different type of pulse:

a square pulse without variations, digital or DAC.
a square pulse with varying amplitude, DACs only.
a square pulse with varying duration, digital or DAC.
a train of square pulses without variations, digital or DAC.
a ramp pulse with varying amplitude at start, finish or both, DACs only.
a sine-wave without variations, DACs only.
an arbitrary waveform on multiple DACs, one only of these is allowed.

Values

This is the area at the bottom right of the dialog. It holds the parameters defining the currently selected pulse. The pulse can be changed by editing the parameter values. The parameters shown vary according to the type of pulse, the box title shows the type of pulse and the start and end times for the pulse, again using the currently selected time units. The details of the parameters are covered in the *Editing the pulse parameters* section below.



Sine wave DAC 0 at 0.10 to 0.60 s — Id

Size (V)	2	Start (s)	0.1		
Centre (V)	0	Length (s)	0.5		
Start phase	90	Cycle (s)	0.1	Freq (Hz)	10

Dragging and dropping

A number of operations in the pulse configuration dialog are carried out by dragging and dropping. A major advantage of drag and drop is its graphical, visual nature. For this reason, it is increasingly commonly found as a way of carrying out operations in Windows software.

To drag and drop a screen object place the mouse pointer on top of the object, then press and hold down the left mouse button. The mouse pointer may change to indicate the start of a drag operation and an icon may be attached to the pointer to indicate that the object has been ‘grabbed’. Still holding the mouse button down, move the mouse pointer to drag the object to its required destination. The mouse pointer may change while you do this, a common effect is for the pointer to change to a no entry sign (a circle with a diagonal line through it) to indicate that you cannot drop the object at this point. When the object is at the required destination, release the mouse button to drop it in place.

Adding a new pulse

To add a new pulse into the outputs, select the pulse required from the pulses area of the dialog. Drag the new pulse from the pulses area into the display, the mouse pointer changes to a (a hand with a plus sign) to indicate that you are adding a new pulse. As you move the mouse pointer about the display, a vertical line indicates where the new pulse will go and the drop time is displayed in the control area. Once the pulse is correctly positioned, drop it into place by releasing the mouse button.

If you cannot get the display area to accept the new pulse (the mouse pointer is always No Entry), this could be because of the following reasons:

- You are trying to drag a DAC-only pulse such as a ramp, sine wave or waveform into the digital outputs.
- You are trying to drag an arbitrary waveform item into a set of outputs that already contains an arbitrary waveform item, only one of these is allowed.

Moving a pulse

To move a pulse, select the pulse in the display area. Then drag the pulse into the new position required (you are allowed to drag a pulse onto a different output as long as you do not change from DAC to digital or vice-versa). The drop position is shown as for adding a new pulse, the mouse pointer changes to (a hand) to indicate taking hold of something without addition or removal..

If the `Ctrl` key is held down during the drag operation the mouse pointer will change to a hand with a + in the centre to indicate addition and the pulse will be copied instead of moved.

To avoid problems with precise positioning of the mouse Signal does not recognise a drag-and-drop operation until the mouse has moved a certain amount away from the initial position. The mouse pointer shows this by only including the pulse icon and showing the drop position when the amount of movement is sufficient. You can give up on a move by returning the pointer close to the initial position. If you want to move a pulse a small amount, but find that Signal will not recognise a drag operation that short, move the mouse pointer a larger amount vertically to convince Signal that you mean it and smaller horizontal movements will be accepted.

If you cannot find or click on the pulse to start dragging it (usually because it is too short or completely hidden by another pulse), see *Finding a pulse* below for how to select it. Once the pulse is selected, you can change the start time directly by editing it in the values area.

Removing a pulse

To remove a pulse, select it in the display area, then drag the pulse out of the display area completely. The mouse pointer changes to (a hand with a minus sign) to indicate a remove operation once you are outside the display area. Drop the pulse to complete the removal.

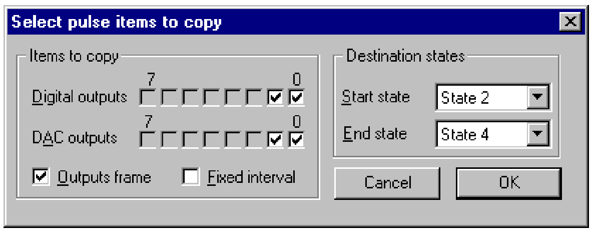
Alternatively, once the pulse has been selected, you can remove it by clicking the Del button in the controls area. If you cannot find or click on the pulse to start dragging it, see *Finding a pulse* below for how to select it.

Finding a pulse

It may be difficult to see a pulse or to click on it because the pulse is too short to see or because it is hidden by another pulse. Click on the appropriate output trace, then use the ‹ › buttons to toggle through all of the pulses on that output. At the appropriate point, your hidden pulse will be selected and can then be edited or deleted directly.

☞ **Experiment with adding, moving and removing pulses to get a feel for how the dialog behaves. Note the different behaviour of overlapped pulses with absolute or relative pulse levels.**

Copying pulses



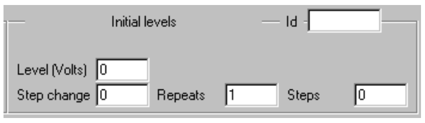
Start state: State 2
End state: State 4
[✓ Outputs frame] Fixed interval

When using multiple states with multiple sets of pulses it may be useful to copy a set of pulses or settings to create duplicated data for other states. The Copy button opens a dialog where you can specify items within the currently selected state to be copied and a range of destination states to copy them to. The left-hand side of the dialog sets the items to copy, the right hand side sets the destination. If you clear the Outputs frame checkbox, the outputs duration, any variable points setting, and the trigger time within the outputs are not copied, otherwise the target states are set to the length, points and trigger time of the current state. Similarly, checking the Fixed interval checkbox allows the fixed interval and variation values to be copied. For the outputs selected, all existing pulses in the destination states are deleted before the new data is copied in.

Editing pulse parameters

In addition to changing the pulse start times by dragging and dropping, all of the pulse parameters can be edited by using the values area of the dialog. This area shows all the pulse parameters, it is different for each type of pulse.

Initial level



Level (Volts) 0
Step change 0 Repeats 1 Steps 0

This specifies the state that the outputs are set to at the beginning of the pulse outputs frame, this item is always present and cannot be deleted or moved. The initial level has a single parameter; Level, that sets the level that the DAC is initially set to. The level entered is scaled before use as defined by the DAC settings in the output page.

The three other parameters at the bottom of the values area; Step change, Repeats and Steps, can be used to define a built-in variation in initial level. Built-in variations are described under *Pulses with variations*, below.

For digital outputs, only the initial level parameter is needed, set 1 for high and 0 for low.

The Id field in the values area boundary displays the name for the currently selected pulse, which can be edited as desired. The main reason for giving a pulse a name is to make it easy to access the pulse from the script language.

Non-output parameters

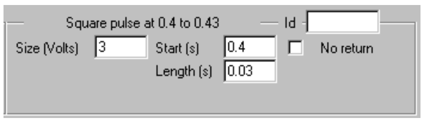
The initial level information is also used to display other parameters that are not part of the actual outputs generated.

With Outputs frame and Fixed interval sweep modes, the Frame parameter sets the length of the outputs frame, while the Trigger parameter sets the start time of the sampling sweep.

With Fixed interval mode two more parameters; Interval (s), and Vary (%) appear (with Fast fixed int mode the Interval field is available but not Vary). These set the timed interval between output frames and the percentage random variation in the interval. If the variation is non-zero, the frame interval used while sampling will vary randomly from Interval-Vary% to Interval+Vary%.

If variable sweep points are enabled (only available with Outputs Frame and Fixed interval modes), yet another field is provided to set the number of points sampled per channel for the selected state.

Simple square pulse



Square pulse at 0.4 to 0.43 — Id

Size (Volts) 3 Start (s) 0.4 ☐ No return

Length (s) 0.03

This specifies a square pulse without any variations. This is the simplest type of pulse and is available for DACs and for the digital outputs. Many of the parameters used for other types of pulse are also used for this type, to save space these common parameters are described in detail once only.

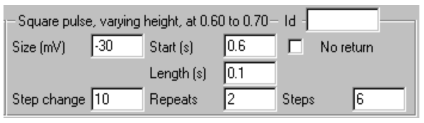
The Size parameter sets the pulse size, in calibrated units as defined in the outputs page. If absolute pulse levels are in use, the item changes to Level, and is the level that the pulse goes to, for relative levels the size is added to the level prior to the pulse to get the pulse level. Either positive or negative values can be used.

For digital pulses, the Size parameter disappears; for outputs with a low initial value a high-going pulse is produced and vice-versa. The use or otherwise of absolute pulse levels does not affect digital pulses and overlapping pulses do not invert each other.

The Start parameter sets the start time for the pulse. If absolute times are not in use, this parameter changes to Delay and sets the delay from the start of the previous pulse to the start of this pulse. The Length parameter sets the length of the pulse. Both of these fields will use the time units selected in the Preferences dialog.

If the No return checkbox is checked, the pulse does not return back to the initial level when it ends but just stays at the pulse level, giving us a single step-change. In this circumstance the length parameter has no effect.

Varying amplitude pulse



Square pulse, varying height, at 0.60 to 0.70 — Id

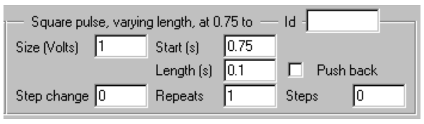
Size (mV) -30 Start (s) 0.6 ☐ No return

Length (s) 0.1

Step change 10 Repeats 2 Steps 6

This specifies a square pulse whose amplitude varies as it is used. The Size, Start and Length parameters are exactly the same as for the simple pulse. In addition there are three parameters controlling the built-in variation, the use of this variation is described under *Pulses with variations*, below. This type of pulse is not available for digital outputs.

Varying duration pulse



Square pulse, varying length, at 0.75 to — Id

Size (Volts) 1 Start (s) 0.75

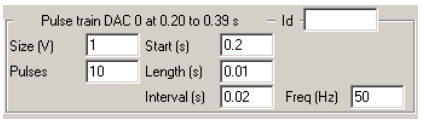
Length (s) 0.1 ☐ Push back

Step change 0 Repeats 1 Steps 0

This specifies a pulse whose amplitude is constant but the pulse length changes as it is used. This is the only pulse with built-in variation that is available for the digital outputs.

The Size, Start and Length parameters are the same as for the simple pulse. The other parameters control the variation, with the exception of the Push back checkbox. If this is checked, increases in the pulse duration delay the start of following pulses and decreases in the pulse duration move following pulses earlier. If the checkbox is clear, changes in the pulse duration do not affect the time of following pulses.

Square pulse train



Pulse train DAC 0 at 0.20 to 0.39 s — Id

Size (V) 1 Start (s) 0.2

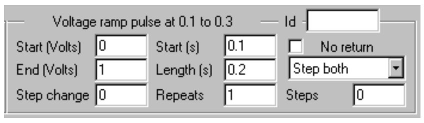
Pulses 10 Length (s) 0.01

Interval (s) 0.02 Freq (Hz) 50

This specifies a series of non-varying square pulses. This type of pulse is available for DACs and for the digital outputs.

The Size, Start and Length parameters are all exactly the same as for the simple square pulse. The extra parameters are Pulses, which sets the number of pulses in the train, Interval which sets the pulse spacing – this should not be less than Length, and Freq (Hz) which allows you to set the pulse spacing in terms of pulses per second.

Ramp with varying amplitudes

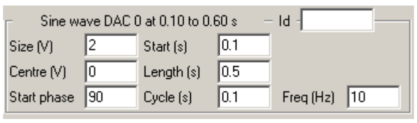

Voltage ramp pulse at 0.1 to 0.3 — Id

Start (Volts)	0	Start (s)	0.1	☐ No return	
End (Volts)	1	Length (s)	0.2	Step both	
Step change	0	Repeats	1	Steps	0

This item specifies a pulse with different start and end amplitudes so that the top of the pulse can be sloping. The variation in amplitude, if this is used, can be applied to either the pulse start or end, or both.

The Start and Length parameters are exactly the same as for the simple pulse, the size parameter is also called Start, but shows the DAC units to avoid confusion. The extra parameters are End, which sets the pulse level at the end of the pulse and a selector for the variation which can be set to Step both, Step start or Step end. This is an extremely versatile form of pulse; for example by setting the start size to zero you can produce a ramp running from one level to another.

Sine wave

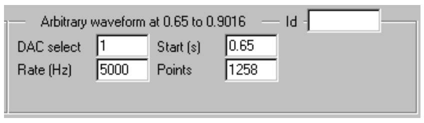

Sine wave DAC 0 at 0.10 to 0.60 s — Id

Size (V)	2	Start (s)	0.1		
Centre (V)	0	Length (s)	0.5		
Start phase	90	Cycle (s)	0.1	Freq (Hz)	10

This item specifies a cosine wave output of fixed duration, amplitude and frequency for output on a DAC. For Signal version 3, sine wave output can be generated on all DACs. In earlier versions only DACs 0 and 1 could be used.

The Size parameter sets the amplitude of the cosine wave (the distance from the mid-point to extreme). The other non-standard parameters are Centre, which sets the level about which the cosine oscillates, Start phase, which sets the initial phase in degrees, Cycle (s), which sets the duration of one complete cycle and Freq (Hz) which allows you to set the cycle period in terms of cycles per second. The Centre value is an absolute or relative voltage level as required. Because the output is actually a cosine wave an initial phase of 90 degrees will start the output off at the centre level.

Arbitrary waveform


Arbitrary waveform at 0.65 to 0.9016 — Id

DAC select	1	Start (s)	0.65
Rate (Hz)	5000	Points	1258

This item specifies arrays of data to be output to one or more DACs at a specified rate. Output to each DAC starts simultaneously and consists of the same number of points, so the output also finishes synchronously. The arrays of data can be changed in two ways; by using the Signal script language functions and by copying and pasting Signal data using the Windows clipboard.

The DAC select parameter specifies which DACs are used. Its value is made up by summing codes for each DAC wanted, the code values are shown in the table. Thus for DACs 0 and 1, enter the value 3. The Rate (Hz) parameter sets the output rate for the data points for each DAC and the Points parameter sets the number of data points for each DAC.

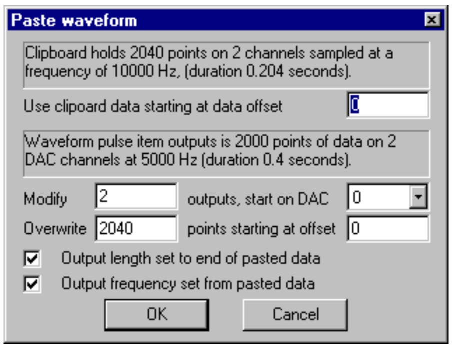
DAC	Code
0	1
1	2
2	4
3	8

The maximum output rates possible vary according to many factors such as the ADC rate and the pulse output timing resolution. Tests carried out at CED with two channels of ADC data sampled at 10 KHz show that waveform rates of 70KHz are possible using a 1401*plus*, and rates of up to 275KHz can be achieved with a micro1401.

Setting a waveform

The Signal data view copy operation places data onto the clipboard in a private format which can be used by the pulse dialog. To place suitable data on the clipboard, open a data file and adjust the display so that the required time range and channels are visible, then use the Edit menu Copy command to place the data onto the clipboard.

If you then open the pulse outputs configuration dialog, select or add a waveform output item and then press `Ctrl+V` (the Paste command shortcut), Signal recognises this as an attempt to paste data into the waveform buffer. It is difficult to ensure that you have copied exactly the right data onto the clipboard, so Signal provides a dialog to control the paste:


Paste waveform

Clipboard holds 2040 points on 2 channels sampled at a frequency of 10000 Hz, (duration 0.204 seconds).

Use clipoard data starting at data offset: 0

Waveform pulse item outputs is 2000 points of data on 2 DAC channels at 5000 Hz (duration 0.4 seconds).

Modify 2 outputs, start on DAC 0

Overwrite 2040 points starting at offset 0

☑ Output length set to end of pasted data

☑ Output frequency set from pasted data

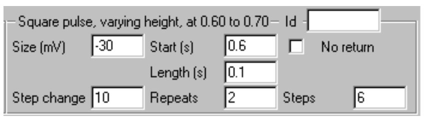
OK Cancel

The upper part of the dialog describes the data on the clipboard and provides control of the first data point taken from the clipboard. An offset of zero takes data starting with the first point on the clipboard, larger offsets cause points at the start of the clipboard data to be skipped. The same offset is used for all DACs.

The lower part of the dialog describes the current waveform output parameters, and provides control over what waveform data is changed and how the waveform output is modified to match the clipboard data. The upper pair of controls, making up the line Modify x outputs, start on DAC n, set which channels of the waveform are modified. If changing more than one channel, channels in use starting with the DAC specified are changed. The lower pair, making up the line Overwrite x points, starting at offset y, sets the amount of data pasted and where in the waveform buffer it is put.

The Output length set to end of pasted data and Output frequency set from pasted data checkboxes act as their titles imply. If all data offsets are set to zero, the points to overwrite is set to the points on the clipboard and these two checkboxes are checked, the paste operation copies all the clipboard data and changes the waveform output parameters to match. Click OK to paste the data, or Cancel to do nothing.

Pulses with variations


Square pulse, varying height, at 0.60 to 0.70 Id

Size (mV) -30 Start (s) 0.6 ☐ No return

Length (s) 0.1

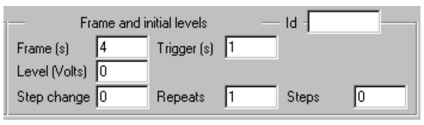
Step change 10 Repeats 2 Steps 6

Some types of pulse can be set up so that they vary automatically. The pulse types that support this are initial level, square pulse with varying amplitude, square pulse with varying duration and the ramp pulse. All of these use the same three parameters to control the variation, only the varying aspect depends upon the type of pulse. The pattern of variation used is: the pulse is generated a number of times without variation, then a number of times with one ‘step change’ added, then two steps and so on. This repeats until the maximum number of changes has been reached, at which point the cycle restarts with the pulse with no step changes.

The Step change parameter sets the amount by which the varied aspect (the pulse amplitude for example) changes at a time. The Repeats parameter sets the number of times each step is repeated before moving on to the next step and the Steps parameter sets the maximum number of changes to be added. This arrangement gives a final value of Initial + (Steps * Step change). The total number of pulse forms generated is one more than Steps as the variation includes the basic pulse without any step changes.

In the example shown above, seven pulses will be generated with amplitudes of -30, -20, -10, 0, 10, 20 and 30 mV. Each pulse will be generated twice in the order shown; the entire sequence will repeat after 14 pulses. If you wanted a sequence that ran 30, 20, 10, 0, -10, -20,and -30, you would set the pulse amplitude to 30 and the step change to -10.

Outputs frame and Fixed interval sampling modes

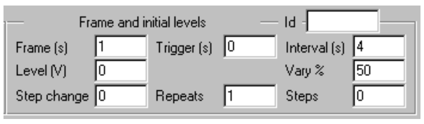


Frame and initial levels — Id

Frame (s)	4	Trigger (s)	1		
Level (Volts)	0				
Step change	0	Repeats	1	Steps	0

With Basic, Peri-triggered, Fast triggers and Fast fixed int sweep modes, the length of the pulse outputs is set by the sampled data. In Basic, Fast triggers and Fast fixed int modes, the outputs frame length is the same as the data frame length, in Peri-triggered mode the outputs length is the data frame length less any pre-trigger points. With Outputs frame and Fixed interval sweep modes, the outputs frame length is set independently of the data frame and data acquisition starts at a preset time within the outputs. Because these values can vary with the state used, they are set in the pulse outputs dialog for convenience.

The outputs length and sweep time are set by extra parameters shown with the Initial level data for all outputs. The Frame parameter sets the length of the outputs frame, this value must be at least as long as the sampling sweep. The Trigger parameter sets the time, relative to the start of the pulse outputs, of the start of the sampling sweep.

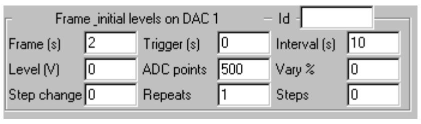


Frame and initial levels — Id

Frame (s)	1	Trigger (s)	0	Interval (s)	4
Level (V)	0			Vary %	50
Step change	0	Repeats	1	Steps	0

With Fixed interval mode two more parameters; Interval (s), and Vary (%) appear (with Fast fixed int mode the Interval field is available but not Vary). These set the timed interval between output frames (which cannot be less than the pulse output frame length) and the percentage random variation in the interval, from 0 to 100, for the selected state. If the variation is non-zero, the frame interval used while sampling will vary randomly from Interval-Vary% to Interval+Vary%.

Variable sweep points



Frame _initial levels on DAC 1 — Id

Frame (s)	2	Trigger (s)	0	Interval (s)	10
Level (V)	0	ADC points	500	Vary %	0
Step change	0	Repeats	1	Steps	0

If variable sweep points are enabled (only available with Outputs Frame and Fixed interval modes), yet another field is provided in the pulse initial levels to set the number of points sampled per channel for the selected state. This ADC points field can be set to any value less than or equal to the sweep points value set in the General section of the sampling configuration.

Controlling pulse outputs during sampling

While sampling is in progress, the pulses configuration dialog can be obtained by using the Sampling menu, by clicking on the relevant toolbar button or by right-clicking on spare application areas to get a popup menu. All three mechanisms provide the pulse configuration dialog in the same form as used offline. The only difference is that the OK button has been renamed Apply and the Cancel button Close and that pressing Apply doesn’t cause the dialog to disappear.

You can use the pulse configuration dialog freely during sampling to change the pulses that are output, the only restriction being that you cannot increase the size of any arbitrary waveform items, or add an arbitrary waveform item where one was not present beforehand. All changes made will be copied into the sampling configuration if the sampling completes successfully so that they can be used next time.

Timing warnings

With Fixed interval or Fast fixed int mode in use, the use of external sweep triggers is disabled as the interval timer does all of the triggering. If the delay between the end of one pulse frame and the start of the next is too short, the internal trigger may be missed. If this occurs a warning message is generated at the end of data acquisition.

7 Sequencer outputs during sampling

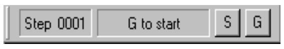
Overview

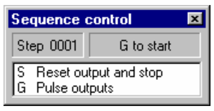
An output sequence is a list of up to 8191 instructions (2047 for older 1401 types) that are executed by the 1401 at a constant, user-defined rate to produce the outputs required. Sequences are defined by a sequence file, which is a text file with a `.pls` file extension. The Signal sequencer has the following features:

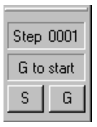
- It controls digital output bits 15-8 to produce precisely timed digital pulse sequences. In the Power1401 and Micro1401 it can also control digital output bits 7-0.
- It controls the 1401 DACs (Digital to Analogue Converters) to produce voltage pulses and ramps.
- It can play cosine waves at variable speed and amplitude through the DACs.
- It can test digital input bits 7-0 and branch on the result.
- It supports loops and branches and can randomise delays and stimuli.
- It has 256 variables (`V1` to `V256`) that can be read and set by on-line scripts. The 1401*plus* allows only 64 variables.
- It supports a user-defined table of values for fast information transfer from a script.
- It can read the latest value from a waveform channel and, using this information, provide real-time (fractions of a millisecond) responses to input data changes.
- It can set or get the sweep state code, get the start time of the sweep, wait until a specified time within the sweep or trigger a sweep.
- It can control and monitor the arbitrary waveform output.

You write sequences with a text sequence editor; each text line generates one instruction.

Sequencer control panel







While sampling is in progress, you can interact with sequencer execution using the sequencer control panel. The control panel displays the next sequence step 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. If there are any keyboard-controlled entry points in the sequence, these are also shown in the 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 Signal application window. When undocked, the control panel displays 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 is always displayed if you start sampling using Sequencer outputs from a Signal 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.

Sequencer technical information

The sequencer clock starts within a microsecond of recording time zero, the time at which the Start button is pressed, and is time locked to the 1401 sweep 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 stop 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 interval, the approximate time per step and the extra time used for cosine and ramp output for each 1401. Notice that the first item is in units of milliseconds (to match Signal usage) and the remainder are in microseconds.

	Power 2	Power	Micro 3	Micro 2	micro	*plus*
Minimum tick (ms)	0.006	0.010	0.025	0.025	0.10	3.0
Time used per tick (us)	<1	<1	~1	~1	<8	<10
Cosine penalty/tick (us)	0.3	0.55	~1	~1	~5	~10
Ramp penalty/tick (us)	0.25	0.5	0.7	0.7	~3	~10
DIV/RECIP penalty (us)	<1	<1	<3	<3	<10	<5

The Minimum tick is the shortest interval we recommend that you 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. The Ramp penalty/tick is the extra time taken per ramped DAC, the DIV/RECIP penalty indicates the extra time taken up by these instructions. Time used by the sequencer is time that is not available for sampling, transferring data back to the host 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, Signal will warn you of this when sampling finishes. If the 1401 is extremely overloaded by the combination of sampling and sequencer output, sampling will fail with an appropriate error message.

The clock interval that you set is rounded to the nearest 4 microseconds to generate the actual sequencer tick interval. For modern 1401s (the Power1401s and Micro1401 mk II & later), intervals less than 5 milliseconds are rounded to the nearest 0.1 microsecond. All subsequent timing is based upon the rounded interval so that full accuracy is maintained.

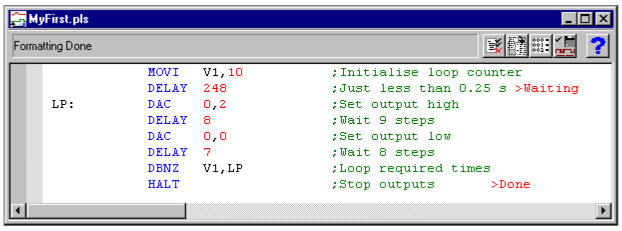
1401-specific limitations

On the 1401*plus*, the sequencer is limited to 64 variables, all other 1401 types allow 256.

On the 1401*plus*, micro1401 and Micro mk II, the limit to the number of sequencer instructions is 2047, other types allow 8191.

The sequence editor

The output sequence is stored as a text file with the extension `.PLS`. You can open existing Signal output sequence files or create new ones with File menu New. There are five buttons at the top of the window:



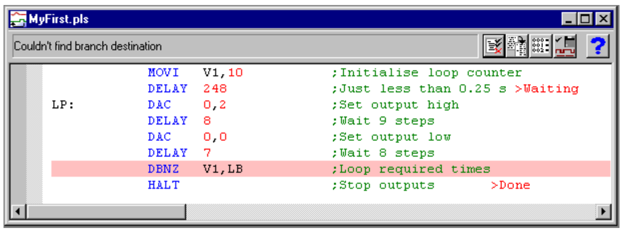
MyFirst.pls

Formatting Done

```
        MOVI   V1,10          ;Initialise loop counter
        DELAY  248            ;Just less than 0.25 s >Waiting
LP:     DAC    0,2            ;Set output high
        DELAY  8              ;Wait 9 steps
        DAC    0,0            ;Set output low
        DELAY  7              ;Wait 8 steps
        DBNZ   V1,LP          ;Loop required times
        HALT                  ;Stop outputs      >Done
```

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 `LB` in the seventh line should be `LP`). The line in error is marked and an explanatory message is shown at the top of the window.



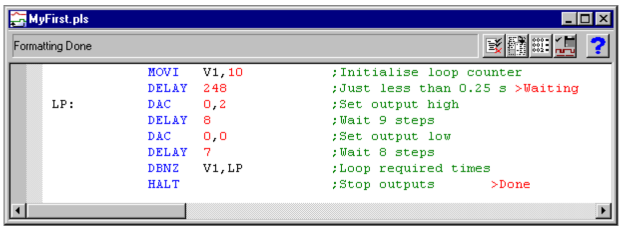
MyFirst.pls

Couldn't find branch destination

```
        MOVI   V1,10          ;Initialise loop counter
        DELAY  248            ;Just less than 0.25 s >Waiting
LP:     DAC    0,2            ;Set output high
        DELAY  8              ;Wait 9 steps
        DAC    0,0            ;Set output low
        DELAY  7              ;Wait 8 steps
        DBNZ   V1,LB          ;Loop required times
        HALT                  ;Stop outputs      >Done
```

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.



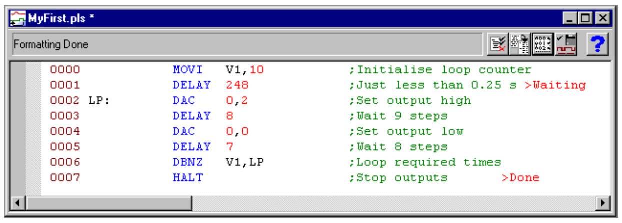
MyFirst.pls

Formatting Done

```
        MOVI   V1,10          ;Initialise loop counter
        DELAY  248            ;Just less than 0.25 s >Waiting
LP:     DAC    0,2            ;Set output high
        DELAY  8              ;Wait 9 steps
        DAC    0,0            ;Set output low
        DELAY  7              ;Wait 8 steps
        DBNZ   V1,LP          ;Loop required times
        HALT                  ;Stop outputs      >Done
```

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.



MyFirst.pls *

Formatting Done

```
0000        MOVI   V1,10          ;Initialise loop counter
0001        DELAY  248            ;Just less than 0.25 s >Waiting
0002 LP:    DAC    0,2            ;Set output high
0003        DELAY  8              ;Wait 9 steps
0004        DAC    0,0            ;Set output low
0005        DELAY  7              ;Wait 8 steps
0006        DBNZ   V1,LP          ;Loop required times
0007        HALT                  ;Stop outputs      >Done
```

Current

This compiles the sequence and, if the sequence 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.

Help

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

Loading sequence files for sampling

The name of the output sequence 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 250 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, Signal searches for the output sequence file named in the sampling configuration and will generate an error message if it cannot be found.

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

Getting started

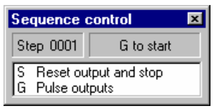
The sequencer runs instructions in order unless told to branch, starting with the first instruction. The sequence can either be restarted at the start of each sweep, or it can free run throughout data acquisition. Sequencer execution can be re-routed during data acquisition by associating an instruction 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. Signal records keys pressed during sampling in the keyboard marker channel, so you can have a record of where in the sampling you switched to a new portion of the sequence.

Simple sequence, restarted each sweep

Here is a simple sequence that pulses DAC 0 high for 10 milliseconds ten times starting at 0.25 seconds into the sweep. It for use with sequencer execution restarting each sweep. You can start and stop the pulses with keys or by clicking buttons. You will find this in `Signal3\Sequence\MyFirst.pls` to save typing it in.

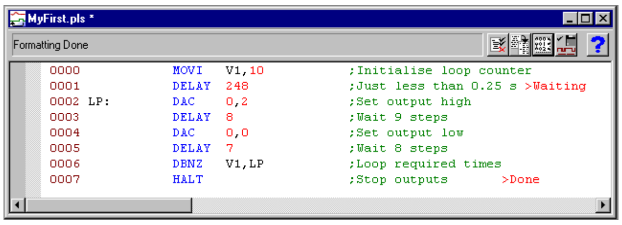
```
0000            MOVI   V1,10       ;Initialise loop counter
0001            DELAY  248         ;Just less than 0.25 s >Waiting
0002 LP:        DAC    0,2         ;Set output high
0003            DELAY  8           ;Wait 9 steps
0004            DAC    0,0         ;Set output low
0005            DELAY  7           ;Wait 8 steps
0006            DBNZ   V1,LP       ;Loop required times
0007            HALT               ;Stop outputs       >Done
```

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 sampled channel on ADC port 0, a sample rate of at least 10 KHz and a sweep length of a second. On the Outputs page, make sure the sequencer clock rate is set to the default of 1 millisecond, and that the Free run without restarts box is not checked. To record the output pulses you must connect the DAC 0 output to the ADC 0 input. This is easily done with a BNC cable on the 1401 front panel. You do not need to make the connection to follow this description.



```
Sequence control
Step 0001     G to start
S  Reset output and stop
G  Pulse outputs
```

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 0000 in the top left window and a blank area to the right. If the control panel is not docked there is also an area for comment display, currently clear. If you start sampling, you will see that the control panel displays the text after the > when the relevant steps are being executed, if you are sampling the output you will see the ten pulses being generated.



```
MyFirst.pls *
Formatting Done
0000            MOVI   V1,10       ;Initialise loop counter
0001            DELAY  248         ;Just less than 0.25 s >Waiting
0002 LP:        DAC    0,2         ;Set output high
0003            DELAY  8           ;Wait 9 steps
0004            DAC    0,0         ;Set output low
0005            DELAY  7           ;Wait 8 steps
0006            DBNZ   V1,LP       ;Loop required times
0007            HALT               ;Stop outputs       >Done
```

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 of the lines get a step number starting at zero, this is the step number displayed in the control panel. You can remove step numbers with the Format button.

Step 0 is executed at time zero in each sweep. The `MOVI V1,10` instruction on this line sets the value of variable 1 (`V1`) to 10. This variable is used as a loop counter in the sequence, so this statement sets the number of times that execution will go round the loop. The remainder of the line is a comment.

The next line, step 1, holds the instruction `DELAY 248`. This causes sequencer execution to wait at this step for the specified number of sequencer ticks into the sampling sweep, plus one more for the delay instruction itself (this could be changed, but we have kept it this way to match the behaviour of the Spike2 sequencer, for those who use both programs). The sequencer control panel displays the text to the right of the > for the current step, so while this step is executing, the text "Waiting" is displayed. Note that we wait for 249 ticks (0.249 seconds). This means that the next step will execute at precisely 250 steps into the sweep, or 0.25 seconds. Note also that we could have written `DELAY ms(249)-1` to achieve the same effect and that this would have the advantage of being immune to changes in the sequencer tick interval.

Step 2 is `DAC 0,2`, which sets the output of DAC zero to two volts. This step has a label at the start of the instruction text, `LB:`, so we can branch back here from elsewhere.

Step 3 is `DELAY 8`, which causes sequencer execution to wait at this step for nine sequencer clock ticks, thus setting the width of the pulse. The width will include the time taken by the DAC instruction as well, giving ten ticks (0.01 seconds) overall. Note that, as for step 1, we could have written `DELAY ms(9)-1` to achieve the same effect.

Steps 4 and 5 are the same as steps 2 and 3; they set the DAC output low again and wait for 8 sequencer ticks. We only wait for 8 ticks because step 6 adds an extra step into this part of the sequence, so we get 10 ticks overall for the low phase too.

Step 6 is `DBNZ V1,LP`, which causes the sequencer to decrement the value of variable number 1 and, if it has not reached zero, branch to the label specified so that the next instruction executed will be step 2. As we have put the value 10 into variable 1 at the start of the sequence, this will cause the loop that creates a pulse to execute ten times.

Finally step 7 is `HALT`, which causes sequencer execution to stop until the start of the next sweep, where it will be automatically restarted at step 0. This step displays the text "Done" in the sequencer control panel while it is executing.

Free-running sequence

The simple way to use the sequencer is with restarts, so that the sequence restarts at the first step at the start of each sweep. However, this mode of operation may prove insufficiently flexible for some situations. If you check the Free run without restarts box in the Outputs page, you get a different style of sequencer operation. You will find a sequence that operates in this manner in `Signal3\Sequence\MySecond.pls`; this also demonstrates some other aspects of sequencer use:

```
0000 FL:     'F DAC    0,2         ;Start fast pulse   >S for slow
0001            DELAY  ms(9)-1     ;Wait 9 ms          >S for slow
0002            DAC    0,0         ;Set output high    >S for slow
0003            DELAY  ms(8)-1     ;Wait 8 ms          >S for slow
0004            JUMP   FL          ;Continue           >S for slow

0005 SL:     'S DAC    0,2         ;Set slow pulse     >F for fast
0006            DELAY  ms(99)-1    ;Wait 99 ms         >F for fast
0007            DAC    0,0         ;Set output low     >F for fast
0008            DELAY  ms(98)-1    ;Wait 98 ms         >F for fast
0009            JUMP   SL          ;Continue           >F for fast
```

Most of this sequence is similar to `MyFirst.pls`. The differences of interest are:

Steps 0 and 5 have a keyboard character defined as well as a label for looping. The sequencer control panel will show buttons corresponding to each character defined; pressing the button or the appropriate keyboard character will cause sequencer execution to jump to the relevant instruction.

A `JUMP` instruction is used instead of `DBNZ` so that the pulse outputs continue forever.

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 *Signal Training Course* manual.

Instructions

These are the output sequencer instructions available in Signal.

Digital (TTL compatible) input and output (see page 7-12)	`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
DAC (waveform/voltage) outputs (see page 7-14)	`DAC`	Set DAC value (for DACs 0-7)
	`ADDAC`	Increment DAC by a value
	`RAMP`	Automatic DAC ramping to a target value
Sinusoidal waveform output (see page 7-16)	`SZ`	Set the cosine output amplitude
	`SZINC`	Change the cosine output amplitude
	`RATE`	Set the cosine angular increment per step
	`RATEW`	As `RATE`, but waits for phase 0
	`ANGLE`	Set cosine angle for the next step
	`WAITC`	Branch until cosine phase 0
	`RINC`	Change the cosine angle increment per step
	`RINCW`	As `RINC` but waits for phase 0
	`PHASE`	Defines what phase 0 means
	`OFFSET`	Offset for sinusoidal output
	`CLRC`	Clear the new cycle flag
General control (see page 7-21)	`DELAY`	Do nothing for a set number of steps
	`DBNZ`	Decrement a variable and 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`	Like `CALL`, but load a variable with a value before call
	`RETURN`	Branch to instruction after last `CALL` or `CALLV`
	`JUMP`	Unconditional branch to a label
	`HALT`	Stops the sequencer and waits to be re-routed
	`NOP`	This does nothing for one step (`NO` `OP`eration)
Variable arithmetic (see page 7-23)	`ADD, SUB`	Addition and subtraction of variables
	`ADDI`	Add a constant value to a variable
	`DIV, RECIP`	Division and reciprocal of variables
	`MOV`	Copy one variable to another
	`MOVI`	Move a constant value into a variable
	`MUL, MULI`	Multiply two variables, multiply by a constant
	`NEG, ABS`	Negate variable, absolute value of variable
Variable logic (see page 7-25)	`AND, ANDI`	Bitwise AND of variables, variable & constant
	`OR, ORI`	Bitwise OR of variables, variable & constant
	`XOR, XORI`	Bitwise XOR of variables, variable & constant
Table support (see page 7-26)	`TABLD, TABST`	Load a register from the table and store a register to the table
	`TABADD, TABSUB`	Add or subtract table value from variable
	`TABINC`	Increment a register and branch while within the table
Access to data capture (see page 7-27)	`CHAN`	Get the latest waveform value from a channel
	`POINTS`	Get the number of points sampled in the current sweep
	`REPORT, MARK`	Forces a digital marker, force marker with set code
	`STATE, SETS`	Get and set the current sweep state code
	`SWEEP, TICKS`	Load variable with the sweep start time, current time
	`WSWP`	Wait until a given time within the sweep
	`TRIG`	Trigger the start of a sampling sweep
Randomisation (see page 7-30)	`BRAND`	Random branch with a probability
	`MOVRND`	Load a variable with a random number
Arbitrary waveform output (see page 7-31)	`WAVE`	Start arbitrary waveform output
	`WAVEBR`	Test arbitrary waveform output and branch on the result

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. If an argument can be represented in different ways, they are separated by vertical bars (read as "or"), for example: `expr|Vn|[Vn+off]`. In this case, the argument can be an expression, a variable or a table reference.
`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. It cannot include a variable.

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 sequence compiler evaluates expressions as real numbers, so `3/2` has the value `1.5`. If `expr` is used as an integer, for example `DELAY expr`, it is rounded to the nearest integer. Values in the range 3.5 to 4.49999… are treated as 4.

Sequencer expression functions

These functions can be used as part of expressions to give you access to Signal 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.
`Hz(expr)`	The angle change in degrees per step for a cosine output of `expr` Hz. For a 2 Hz cosine, use `RATE Hz(2)`. To slow by 0.1 degrees per step use `RINC Hz(-0.1)`. Use in `RATE`, `RATEW`, `RINC` and `RINCW`.
`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 `RATE 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 `expr` user DAC units so that the full DAC range spans the full range of a 16-bit integer.
`VDAC32(expr)`	Converts `expr` user DAC units into a 32-bit integer value such that the full DAC range spans the 32-bit integer range. Use this to load variables for use with the `DAC` and `ADDAC` instructions.
`ASz(expr)`	Converts expr user DAC units into a value suitable for use with the SZ and SZINC commands.
`VSz(expr)`	Converts expr user DAC units into a variable value suitable for use with the SZ and SZINC commands.
`TabPos()`	The number of table data items defined at this point by TABDAT.
`DRange()`	The DAC output range in volts, 5.0 or 10.0.

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 256 variables, `V1` to `V256`, 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 and `V56` holds the last bit pattern read from the digital inputs with the `DIBxx` or `DIGIN` instructions.

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 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. Signal 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 degrees. 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 sequencer variable values by using the `SampleSeqVar()` script command. See *The Signal 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.

Constants

It is often useful to define a numeric value as a named constant. For example, when referencing a table value `[V1+Slope]` is easier to understand than `[V1+23]`. It also helps to maintain code; if you need to change a numeric value that is used often make it a named constant and just change the constant once. The = directive does not generate any instructions, it just makes a name equivalent to a number, and the name can be used anywhere a numeric expression is valid. The syntax is:

```
name = expr            ;comment
```

The `name` must start with an alphabetic character and can contain alphabetic characters and the digits 0 to 9. The name can be up to 8 characters long. Numeric constant names are not case sensitive and must not clash with label, instruction, variable or built-in function names. Examples of use:

```
Wait1  = ms(100)
Wait2  = Wait1*1.3
       DELAY  Wait1-1  ;Use constant in an instruction
```

The expression values are calculated and stored as floating point numbers. If they are used in a context that requires an integer value, the fractional part of the number is ignored.

The = directive was added at version 4.06.

Table of values

The Signal sequencer supports a table of 32-bit values when used with the 1401*plus,* Micro1401s or Power1401s.

Declaring the table

You declare that a table exists with the `TABSZ` directive, which normally occurs at the start of your sequence:

```
       TABSZ  expr
```

Where `expr` is an expression that sets the number of items in the table. The table size must evaluate to a number in the range 1 to 1000000. Each table item is a 32-bit integer and uses 4 bytes of 1401 memory. The maximum size in a 1401 with 1MB of memory, and assuming that there is no arbitrary waveform output, is around 150000 items. The first table item has an index number of 0, the second item is index 1, and so on.

Setting table data

From the script language you can move data between an integer array and the table with the `SampleSeqTable()` function. You can also preset table data from the sequence with the `TABDAT` directive, which must come after the `TABSZ` directive:

```
       TABDAT expr
       TABDAT expr,expr,expr...
```

Where `expr` is an expression that evaluates to a 32-bit integer. Each `TABDAT` directive adds data to the table, starting at the beginning. The sequencer compiler will flag an error if you define more data that will fit in the table. Table data declared in this way is stored separately from the sequence and is transferred to the 1401 when you create a new data file to sample. If you do not set the table data with the `TABDAT` directive or from a script, the values in the table are undefined.

Accessing table data

Although you can move data between one of the variables and the table with the `TABLD` and `TABST` instructions, many instructions access the table directly. It takes slightly more time to use a table than to use a variable.

All references to the table use the contents of one of the variables as an index into the table plus an optional offset as: `[Vn]` or `[Vn+off]` or `[Vn-off]`. The offset `off` is an expression that evaluates to a number in the range –100000 to 1000000. For example, if `V1` holds 10, `[V1]` refers to the contents of index 10, `[V1-10]` refers to index 0 and `[V1+10]` refers to index 20. Out of range table indices read 0 and are non-destructive.

The `TABINC` instruction makes it easy to increment a variable used as a table index and branch until the increment generates a value outside the table size. The following example generates five DAC outputs at 5 different intervals:

```
            TABSZ  10            ; table of 10 items
            TABDAT ms(1000)-3,VDac32(1) ; 1000 ms, 1 volt
            TABDAT ms(100)-3,VDac32(2)  ;  100 ms, 2 volts
            TABDAT ms(50)-3,VDac32(3)   ;   50 ms  3 volts
            TABDAT ms(500)-3,VDac32(-1) ;  500 ms -1 volt
            TABDAT ms(200)-3,VDac32(0)  ;  200 ms  0 volts

            MOVI   V1,0          ; use V1 as table index, set 0
TLOOP:      DELAY  [V1]          ; programmed delay
            DAC    0,[V1+1]      ; set DAC 0 to the value
            TABINC V1,2,TLOOP    ; add 2 to V1, branch if in table
```

Long data sequences

If you have a very long data sequence, you should consider using the table as a buffer. The basic idea is to divide the table into two halves and use a script to transfer new data into the half of the table that the sequence is not using. To find out where the sequence has reached, look at the value of the variable used as an index with `SampleSeqVar()`. Set a large enough table size so that the time taken to use half the table is several seconds.

Include files

There are times when you will want to reuse definitions or sequence code sections in multiple projects. You can do this by pasting the text into your sequence, but it can be more convenient to use the `#include` command to include files into a sequence. A file that is included can also include further files. We call these nested include files. Only the first `#include` of a file has any effect. Subsequent `#include` commands that refer to the same file are ignored. This prevents problems with output sequence files that include each other and stops multiple definitions when two files include a common file. A `#include` command must be the first non-white space item on a line. There are two forms of the command:

```
#include "filename" ;optional comment
#include <filename> ;optional comment
```

where filename is either an absolute path name (starting with a `\` or `/` or containing a `:`), for example `C:\Sequences\MyInclude.pls`, or is a relative path name, for example `include.pls`. The difference between the two command forms lies in how relative path names are treated. The search order for the first form starts at item 1 in the following list. The search for the second form starts at item 3.

1. Search the folder where the file with the `#include` command lives. If this fails...
2. Search the folder of the file that included that file until we reach the top of the list of nested include files. If this fails...
3. Search any `\include` folder in the folder in which Signal is installed. If this fails...

4. Search the current folder.

Included files are always read from disk, even if they are already open. If you have set the Edit menu Preferences option to save modified scripts and sequences before running, modified include files are automatically saved when you compile. If this option is not set, the output sequence compiler will stop the compilation with an error if it finds a modified include file. You must save the included file to compile your sequence.

There are no restrictions on what can be in an included file. However, they will normally contain constant and variable definitions and possibly user-defined code. It is usually a good idea to have all your `#include` commands at the start of a sequence file so that anyone reading the source is aware of the scope of the sequence.

The `#include` command for output sequences was added to Signal at version 4.06 and is not recognised by any version before this. A typical file using `#include` might start with:

```
#include <sysinc.pls>           ; my system specific includes
#include "include/proginc.pls" ; search script relative folder
  .....                         ; start of my code...
```

Opening included files

If you right-click on a line that holds an include command, and Signal can locate the included file, the context menu will hold an entry to open the file. The search for the file follows that described above, except that it omits step 2.

Errors in included files

If an error is detected during the compilation of an included file, an error message is displayed at the top of the original window indicating which included file has a problem, and the included file is opened (if it can be found) and the offending line is highlighted.

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.

Many of these instructions allow you to use a variable or a table entry in place of an argument. In this case, the alternatives are separated by a vertical bar, for example:

```
DELAY   expr|Vn|[Vn+off],OptLB
```

This means that the first argument can be an expression, a variable or a table entry. There is no explicit documentation for the use of the table, except in `TABLD` and `TABST`. Where table use is allowed it is written as `[Vn+off]`. If you use a table value in an instruction, the effect is exactly the same as using a variable with the same value as the table entry.

`OptLab` in instructions is an optional label that sets the next instruction to run. If it is omitted, the next sequential instruction runs.

Digital I/O

These instructions give you control over the digital output bits and allow you to read and test the state of digital input bits 7-0.

DIGOUT

The `DIGOUT` instruction changes the state of digital output bits 15-8 (see the *Sampling data* chapter for the connections). The output changes occur at the next tick of the output sequencer clock, so you need to use this instruction one tick early!

```
DIGOUT  [pattern]|Vn|[Vn±off],OptLab
```

`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`. If you use a table value, set the same value in the table that you would use for a variable. You can use the `VarValue` script in the `Scripts` folder to calculate variable or table values.

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

This example produces ten 1 millisecond pulses 100 milliseconds apart.

```
         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 the *Sampling data* chapter for the connections). It has no effect on a standard 1401 or 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|[Vn+off],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`. If you use a table value, set the same value in the table that you would use for a variable. You can use the `VarValue` script in the `Scripts` folder to calculate variable or table values.

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

This example produces ten 1 millisecond pulses 100 milliseconds apart.

DIBEQ, DIBNE These instructions test digital input bits 7-0 against a pattern (see the *Sampling data* chapter for connections). These are the same inputs that will be used for digital markers in the future. `DIBEQ` branches on a match. `DIBNE` branches on a non-match. Both instructions copy digital input bits 7-0 to `V56` (`VDigIn`), for use by `DISBEQ` and `DISBNE`.

```
DIBNE   [pattern]|Vn|[Vn+off],LB
DIBEQ   [pattern]|Vn|[Vn+off],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|[Vn+off],LB
DISBEQ  [pattern]|Vn|[Vn+off],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
```

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 to 8 DAC outputs. The last value written to DACs 0-7 is stored in variables `V57`-`V64` (which you can also refer to 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 DAC scaling set in the sampling configuration Outputs page determines the conversion between the numbers you supply and the DAC outputs. All DACs are scaled identically. 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 pins 36 and 37 of the rear panel 37-way Cannon D type Analogue Expansion connector. Suitable grounds are on the adjacent pins 18 and 19. With a Power1401 top-box with additional front panel DACs, the rear panel DAC outputs are mapped to the two highest numbered DACs. For example, with 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 writes a value to any of the 8 possible DAC outputs. `ADDAC` 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|[Vn+off],OptLB
ADDAC   n,expr|Vn|[Vn+off],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 are as set in the outputs page of the sampling configuration dialog. 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.

This example ramps DAC 2 from 0 to 4.99 volts in 1 second in steps of 0.01 volts using values and then using variables. You can also use the `RAMP` instruction to ramp a DAC.

```
      'R DAC   2,0             ;Ramp 0 to 5      >Ramping
         MOVI  V1,499          ;499 steps        >Ramping
RAMP1:   ADDAC 2,0.01          ;0.01V increment  >Ramping
         DBNZ  V1,RAMP1        ;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
RAMP2:   ADDAC 2,V2            ;add increment    >Ramping
         DBNZ  V1,RAMP2        ;count increments >Ramping
         HALT                  ;task finished    >Done
```

It is a property of signed integers that adding 1 to the maximum positive number yields the minimum negative number. If you use `ADDAC` repeatedly with 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. Users often want to set the DAC to full scale, so for the `DAC` command used with `expr` (not with `Vn`), we change requests to set +32768 units to set 32767 units.

Unlike the digital outputs, the DAC output changes when the instruction runs, not at the next sequencer clock tick; the changes may have a time jitter of a few microseconds.

RAMP This command starts a DAC ramping with updates every sequencer step. If the DAC was generating a cosine, the cosine output stops. The DAC ramps from the current value until it reaches a target value, when the DAC cycle flag sets. You can use `WAITC` to test for the end of the ramp. The `RATE` instruction stops a ramp before it reaches the target value.

```
RAMP     n,target|Vn,slope|Vs|[Vs+off]
```

`n`	DAC number in the range 0-7 (available DACs depend on the 1401 type).
`target`	This is the DAC value at which to end the ramp. The units of the DAC values are those set in the outputs page of the sampling control dialog.
`Vn`	When a variable is used for the target, 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.
`slope`	This expression sets the DAC increment per sequencer step. The sign of the value you set here is ignored as the sequencer works out if it must ramp upwards or downwards to achieve the desired target value. If your DAC is calibrated in volts, for a slope of 1 volt per second, use `1.0/s(1.0)`.
`Vs`	You can also set the slope from a variable or by reading it from the table. In this case, the full range of the 32-bit value represents the full range of the DAC. The DAC changes by the absolute value of this 32-bit value on each step. For a slope of 1 user unit per second, use `VDAC32(1.0)/s(1.0)`.

This example ramps DAC 1 from its current level to 1 volt in 3 seconds, waits 1 second, then ramps it to 0 volts in 5 seconds. See the `OFFSET` command for another example.

```
         RAMP    1,1.0,1.0/S(3) ;start with zero size
         ...                    ;other instructions during ramp
WT1:     WAITC   1,WT1          ;wait for ramp to end >Ramp to 1
         DELAY   S(1)-1         ;wait for a second    >Wait 1 sec
         RAMP    1,0,1.0/S(5)   ;ramp down
WT2:     WAITC   1,WT2          ;wait for ramp        >Ramp to 0
```

Cosine output control instructions

The sequencer can output cosine waveforms of variable amplitude and frequency through Micro1401 DACs 0 and 1, Power1401 DACs 0 to 7 and through 1401*plus* DACs 0 to 1. If you attempt to set cosine output for an unsupported DAC, the instruction is treated as a NOP. When enabled, the cosine value is computed and output every step. The time penalty per step per DAC is around 10 us for the 1401*plus*, 4 us for the micro1401 and about 1 us for the Power1401 and Micro1401 mk II. The output is:

output in volts = 5 A *Cos*(Theta+Phi) + offset

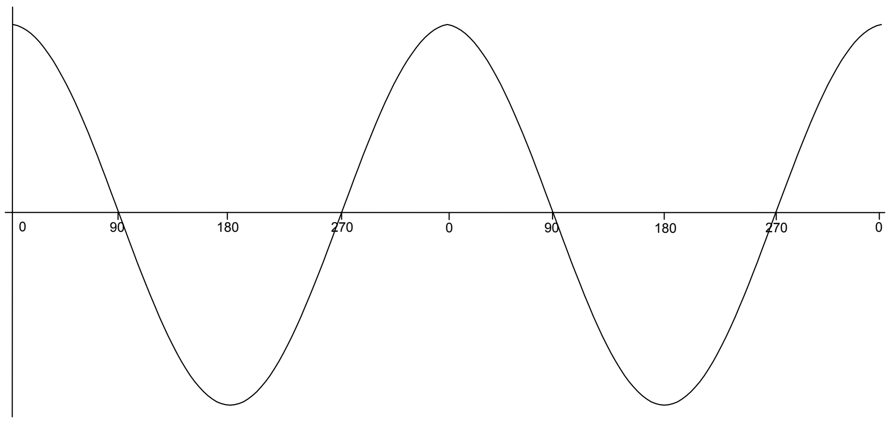
where	A	is an amplitude scaling factor in the range 0 to 1
	Theta	an angle in the range 0° to 360° that changes each step (set by `ANGLE`)
	Phi	is a fixed phase angle in the range -360° to 360° (set by `PHASE`)
	offset	A voltage offset defined by the `OFFSET` command

Theta changes every step by dTheta. A cycle of the cosine takes 360/dTheta steps. You can change the angle increment immediately, or you can delay the change until the next time Theta passes through 0°. You can set dTheta 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 Phi you control the output cosine phase where Theta passes through zero. Unless you set the value (`PHASE`), 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 Theta passes through zero a *new cycle* flag sets. The `RAMP`, `RATEW`, `RINCW`, `WAITC` and `CLRC` instructions clear the flag.

Output as a function of Theta+Phi



SZ This instruction sets 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.

```
SZ      n,expr|Vn|[Vn+off],OptLB    ;DAC n
```

`n`	DAC number in the range 0-7 (available DACs depend on the 1401 type).
`expr`	The cosine amplitude in the range 0 to 1. A cosine with amplitude 1.0 uses the full DAC range.

`Vn` Variable values 0 to 32768 correspond to amplitudes 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.

SZINC This instruction changes 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.

```
          SZINC  n,expr|Vn|[Vn+off],OptLB     ;DAC n
```

`n` DAC number in the range 0-7 (available DACs depend on the 1401 type).

`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.

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):

```
          SZ      0,0.0      ;start with zero size
          MOVI    V1,100     ;proceed in 1% increments
loop:     SZINC   0,0.01     ;a 1% increase
          DELAY   ms(100)-2  ;show some of the waveform at this size
          DBNZ    V1,loop    ;loop 100 times
```

RATE This sets the angle increment in degrees per step, which sets the cosine frequency. If the nominated DAC was ramping, this cancels the ramp. You can stop the cosine output with a rate of 0. Any non-zero value starts the cosine output.

```
          RATE    n,expr|Vn|[Vn+off],OptLB    ;DAC n
```

`n` DAC number in the range 0-7 (available DACs depend on the 1401 type).

`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.

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:

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

RATEW This instruction performs the same function as `RATE`, except that the change is postponed until the next time Theta passes through 0 degrees. `RATEW` cannot start output; a sinusoid must already be running to pass phase 0. It can stop output, but does not remove the overhead for using cosine output. This instruction clears the new cycle flag (see `WAITC`).

```
          RATEW    n,expr|Vn|[Vn+off]         ;DAC n
```

`n` DAC number in the range 0-7 (available DACs depend on the 1401 type).

`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.

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

```
         ANGLE   0,0        ;make sure we are at phase 0
         RATE    0,Hz(10)   ;start output at 10 Hz
         RATEW   0,Hz(11)   ;request 11 Hz next time around
CYCLE10: WAITC   0,CYCLE10  ;wait for the cycle>10Hz
CYCLE11: WAITC   0,CYCLE11  ;wait for the cycle>11Hz
         RATE    0,0        ;stop output
```

ANGLE This changes the cosine angular position. It takes effect on the next instruction when the angle increment is added to the value set by this instruction and the result is output.

```
ANGLE   n,expr|Vn|[Vn+off],OptLB    ;DAC n
```

`n`	DAC number in the range 0-7 (available DACs depend on the 1401 type).
`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.

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.

```
ANGLE   1,-90       ;set the DAC 1 cosine angle directly
MOVI    V1,VAngle(-90)
ANGLE   1,V1        ;set using a variable
```

PHASE This changes the relative phase of the cosine output for the next cosine output. A common use is to change the output from a cosine (maximum value at phase zero) to sine (rising through zero at phase zero).

```
PHASE   n,expr|Vn|[Vn+off],OptLB    ;DAC n
```

`n`	DAC number in the range 0-7 (available DACs depend on the 1401 type).
`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.

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

```
PHASE   2,-90     ;set the DAC 2 phase angle directly
ANGLE   2,0       ;prepare to start as a sine wave
RATE    2,Hz(1)   ;start the sinusoid
```

OFFSET This changes the cosine output voltage offset for the next cosine output.

```
OFFSET  n,expr|Vn|[Vn+off],OptLB    ;DAC n
```

`n`	DAC number in the range 0-7 (available DACs depend on the 1401 type).
`expr`	The offset value for sinusoidal output. The units of this value are as set in the outputs page of the sampling configuration dialog. 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.

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. This example does not work with a 1401*plus*.

```
            DAC    0,0              ;use DAC 0 for all output
            OFFSET 0,1.0            ;set DAC 0 offset
            SZ     0,0.2            ;1 V sinusoid
            PHASE  0,-90            ;Prepare sinusoid
            ANGLE  0,0              ;set start point
            RAMP   0,1.0,1.0/s(1)   ;ramp to 1 volt in 1 sec
RAMPUP:     WAITC  0,RAMPUP         ;wait for ramp      >Ramp up
            RATE   0,HZ(1)          ;start sinusoid
            DELAY  S(4.9)           ;Sinusoid           >Sine
            RATEW  0,0              ;stop at cycle end
END:        WAITC  0,END            ;wait for end       >Wait end
            RATE   0,0              ;stop now
            RAMP   0,0.0,1.0/S(1)   ;ramp to 0 volt in 1 sec
RAMPDN:     WAITC  0,RAMPDN         ;wait               >Ramp down
            HALT
```

WAITC Each time the phase angle of a cosine passes through 0°, a new cycle flag sets. This flag is also set when a ramp terminates. There is a separate flag for each DAC. This flag is cleared by `CLRC`, `RATEW`, `RINCW` and when tested by `WAITC`.

```
WAITC   n,LB                        ;DAC n
```

`n`	DAC number in the range 0-7 (available DACs depend on the 1401 type).
`LB`	A label to branch to if the new cycle flag is not set. If the flag is set, the sequencer clears the flag and does not branch.

This instruction 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, and changes the digital outputs for each cycle.

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

RINC, RINCW These instructions behave like `RATE` and `RATEW` except that they *change* the output rate (angle increment per step) by their argument rather than set it. `RINCW` clears the new cycle flag.

```
RINC     n,expr|Vn|[Vn+off],OptLB    ;DAC n
RINCW    n,expr|Vn|[Vn+off],OptLB    ;DAC n
```

`n`	DAC number in the range 0-7 (available DACs depend on the 1401 type).
`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.

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

```
         RATE   1,Hz(10)  ;start output at 10 Hz
wt:      JUMP   wt        ;HALT stops all output>P=+1Hz, M=-1Hz
      'P RINC   1,Hz(1),wt;1 Hz faster
      'M RINC   1,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:

```
         SZ      1,1.0    ;make sure full size
         ANGLE   1,0.0    ;make sure we start at phase 0
         RATE    1,1.0    ;1 degree per step to start with
         MOVI    V1, 900  ;in 900 steps of...
loop:    CRINC   0.01     ;...1/100 degrees to...
         DBNZ    V1, loop ;...10 degrees per step
```

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

```
         SZ      1,1.0    ;make sure full size
         ANGLE   1,0.0    ;make sure we start at phase 0
         RATE    1,1.0    ;1 degree per step to start with
         MOVI    V1, 90   ;in 90 steps of...
loop:    RINCW   1,0.1    ;...1/10 degrees to...
wait:    WAITC   1,wait   ;...(wait for next cycle)...
         DBNZ    V1, loop ;...10 degrees per step
```

CLRC This instruction clears 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 this instruction immediately before using `WAITC`.

```
CLRC   n,OptLB                      ;DAC n
```

`n`	DAC number in the range 0-7 (available DACs depend on the 1401 type).
`OptLB`	If this optional label is present it sets the next instruction to run.

This example starts a sinusoid and stops it at the next phase 0 crossing after a user stop requests. Because the sinusoid passes phase 0 in the `WAITC` instruction and does another step in the `RATE 1,0` 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 PHASE   1,-2*Hz(2)     ; compenstate for ending
        ANGLE   1,2*Hz(2)      ; so we start in correct place
        RATE    1,Hz(2)        ; 2 Hz output
HERE:   JUMP    HERE           ; output is running >Running
     'S CLRC    1              ; Stop output
WT:     WAITC   1,WT           ; wait for cycle end>Waiting
        RATE    1,0,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|[Vn+off],OptLB
```

`expr`	The extra sequencer clock ticks to delay in the range 0 to 4294967295. The `s()`, `ms()` and `us()` built-in functions convert a delay in seconds, milliseconds or microseconds into sequencer steps.
`Vn`	Variable or table index from which to read the number of extra clock ticks.
`OptLB`	If this optional label is present it sets the next instruction to run.

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,LB    ;wait V1+1 ms, branch    >variable delay
DELAY   [V1+9]   ;V1+9 is table index     >table 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 variable 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
```

CALL, CALLV, RETURN

These instructions run a labelled part of a sequence and return. `CALL` and `CALLV` 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.

```
CALL   LB              ; use LB as a subroutine
CALLV  LB,Vn,expr      ; Vn = expr, then call LB
RETURN                 ; return to step after last CALL
```

`LB`	The next instruction to run. The `CALL`ed 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.

You can use `CALL` 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.

This example generates different pulse widths from DAC 0. The sequence is written to be independent of the sequencer rate, However, it must be high enough so that the widths are possible. In this case a sequencer Step period of 1 millisecond (set in the Outputs tab of the sampling configuration) would be fine. The example sets DAC 0 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.

```
        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. You can also jump using the contents of a register as the destination, or relative to a label (`LB`):

```
        JUMP   LB              ; Jump to label
        JUMP   (Vn),OptLB      ; Jump to instruction Vn
        JUMP   LB(Vn),OptLB    ; Jump to instruction LB+Vn
```

`(Vn)` The value of variable `Vn` sets the instruction number to jump to.

`LB(Vn)` Jump to the instruction given by label `LB` plus the contents of `Vn`.

`OptLB` An optional label to jump to if `(Vn)` or `LB(Vn)` is not an instruction number. The first instruction is 0, the last depends on the size of the sequence.

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 perform basic mathematical functions while a sequence runs. 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
Bxx     Vn,[Vm+off],LB ;compare with a table entry
```

`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 or table index variable.
`expr`	A 32-bit integer constant to compare `Vn` with.

This example collects the latest data value from channel 1 (assumed to be a waveform), waits for it to be in a set 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. 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.

`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, ABS

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, `ABS` is the same, but it takes the absolute value. The syntax is:

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

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
```

DIV, RECIP `DIV` and `RECIP` divide variables. They take around 1 µs in Power1401s, 3 in a micro1401 mk II, 10 in a micro1401 mk 1 and 5 in a 1401*plus*.

```
DIV     Va,Vb           ; Va = Va / Vb
RECIP   Va,expr         ; Va = expr / Vb
```

If the numerator is 0, the result is 0. If the denominator is 0, the result is 2147483647 if the numerator is greater than 0 and –2147483648 if it is negative. The 1401*plus* truncates all results downwards, all other 1401s truncate towards 0. So, 7/3 or –7/–3 is 2 in all 1401s, but –7/3 or 7/–3 is –2 except in a 1401*plus*, where it is –3.

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)
```

Variable logic

These instructions perform bitwise logical functions.

AND, ANDI

These instructions bitwise `AND` a variable with a variable or a 32-bit expression. A bitwise `AND` means that each bit of the 32-bit result is 1 if both corresponding source bits are 1, otherwise the result bit is 0. For example, `3 AND 1` is 1, `0x55 AND 0xAA` is 0.

```
AND     Va,Vb       ; Va = Va AND Vb
ANDI    Va,Vb,expr  ; Va = Vb AND expr
```

`Va`	The variable to hold the result.
`Vb`	A variable to `AND` with `Va` or with the expression.
`expr`	A 32-bit integer constant to `AND` with `Va`.

This example waits for the digital input to have bit 4 set, then branches based on the digital input value (placed in `VDigIn` or `V56` by `DIBNE`).

```
WT:     DIBNE   [...1....],WT   ; wait for bit 4 set >Wait 4 Bit 4
        ANDI    V1,VDigIn,0x0f  ; isolate bits 0..3 (value 0-15)
        JUMP    ACTION(V1)      ; branch based on the result
ACTION: JUMP    ACT0            ; action for value 0
        JUMP    ACT1            ; action for value 1
        ...
        JUMP    ACT15           ; action for value 15
```

OR, ORI

These instructions bitwise `OR` a variable with a variable or a 32-bit expression. A bitwise `OR` means that each bit of the 32-bit result is 1 if either corresponding source bit is 1, otherwise the result bit is 0. For example, 3 `OR` 1 is 3, 0x55 `OR` 0xAA is 0xff.

```
OR      Va,Vb       ; Va = Va OR Vb
ORI     Va,Vb,expr  ; Va = Vb OR expr
```

`Va`	The variable to hold the result.
`Vb`	A variable to `OR` with `Va` or with the expression.
`expr`	A 32-bit integer constant to `OR` with `Va`.

XOR, XORI These instructions bitwise exclusive OR a variable with a variable or a 32-bit expression. A bitwise exclusive OR means that each bit of the 32-bit result is 1 if the corresponding source bits differ, otherwise the result bit is 0. For example, 3 XOR 1 is 2, 0x55 XOR 0xAA is 0xff.

```
XOR    Va,Vb       ; Va = Va XOR Vb
XORI   Va,Vb,expr  ; Va = Vb XOR expr
```

Va	The variable to hold the result.
Vb	A variable to XOR with Va or with the expression.
expr	A 32-bit integer constant to XOR with Va.

Table access

Tables are declared with the TABSZ directive and can be populated with data using the TABDAT directive. Most access to tables is through the [Vn+off] method, but there are also instructions for loading and storing a register in a table and for incrementing or decrementing a register used as a pointer into the table.

TABLD, TABST These two instructions load a register from the table and store a register into the table. Many instructions can load arguments from the table, so TABLD is not often required.

```
TABLD   Vm,[Vn+off],OptLB  ; load Vm from the table
TABST   Vm,[Vn+off],OptLB  ; store Vm into the table
```

Vm	The variable to load from the table or store into the table.
+off	An optional expression that evaluates to an integer in the range –1000000 to 1000000. If omitted, the value 0 is used.
Vn	Any offset in the off expression is added to the contents of this variable and the result is used as an index into the table. If the index lies in the table, Vm is loaded from the table or stored in the table at the index. If the index is outside the table, TABLD copies 0 to Vm and TABST does nothing.
OptLB	If this optional label is present it sets the next instruction to run.

TABADD, TABSUB These two instructions add a table value to a variable or subtract a table value from a variable. These instructions were added at version 4.06.

```
TABADD   Vm,[Vn+off],OptLB  ; add table value to Vm
TABSUB   Vm,[Vn+off],OptLB  ; subtract value from Vm
```

Vm	The variable to add data to or subtract it from.
+off	An optional expression that evaluates to an integer in the range –1000000 to 1000000. If omitted, 0 is used.
Vn	The variable value plus the offset is used as a table index. If the index lies in the table, Vm is changed. If the index is not in the table, the instruction does nothing.
OptLB	If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

TABINC This instruction adds a constant to a variable and detects if the result is a valid table index. If it is a valid index, the instruction branches. If it is not, the result is reduced by the table size if it is positive and is increased by the table size if it is negative, and the instruction does not branch. This gives you an efficient way to work through the table.

```
         TABINC   Vn,expr,OptLB
```

`Vn`	This variable is assumed to hold a valid table index.
`expr`	This expression evaluates to a positive or negative number that is added to `Vn`.
`OptLB`	If this optional label is present it sets the next instruction to run.

For example, the following codes plays pulses through DAC 0 based on data in the table. The table data holds groups of three items, holding the time for the DAC to stay at 0, the DAC amplitude and the time to stay at the amplitude. Some example table data is given, but the data could also be set with the `SampleSeqTable()` script command.

```
         TABSZ   12                ;4 sets of 3 items
         TABDAT  ms(50)-2,VDAC32(1),ms(50)-3
         TABDAT  ms(100)-2,VDAC32(1.3),ms(70)-3
         TABDAT  ms(200)-2,VDAC32(1.5),ms(90)-3
         TABDAT  ms(400)-2,VDAC32(1.9),ms(110)-3
      'G MOVI    V1,0              ;use V1 as the table pointer
LOOP:    DAC     0,0               ;strt with the DAC low
         DELAY   [V1]              ;wait for first period>Low
         DAC     0,[V1+1]          ;get the DAC value
         DELAY   [V1+2]            ;wait for second period>High
         TABINC  V1,3,LOOP
         DAC     0,0               ;tidy up the dac
```

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.

CHAN This instruction gives the output sequencer access to sampled data on a waveform channel. You can also use this command to get the most recent value written to the DAC outputs. The variable value is set to 0 if the channel is not being sampled. This instruction is not available for the 1401*plus*.

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

`chn`	The channel number is 1 to 80 for sampled channels or 0 to -7 for the last value on DACs 0 to 7. The result is the most recent data available.

Waveform and DAC data are treated as 16-bit signed values from –32768 to 32767. 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 instruction takes rather longer to execute than other sequencer instructions (it incurs the DIV/RECIP time penalty), and may cause timing problems if used in circumstances when the 1401 is heavily loaded.

This example waits for a sampled signal to cross 0.05 volts and produces a pulse.

```
         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
```

POINTS This instruction sets a variable to the current number of points sampled in the current sweep. The variable value is set to 0 if the sampling sweep has not started. This instruction is not available for the 1401*plus*.

```
POINTS   Vn,OptLB         ; Vn = Sweep points
```

`Vn`	This variable is updated with the sweep point count.
`OptLB`	If present, the instruction branches to this label.

This can be used instead of the `WSWP` command to wait until a specific point in the sweep, but based on points rather than time. This instruction takes rather longer to execute than other sequencer instructions (it incurs the DIV/RECIP time penalty), and may cause timing problems if used in circumstances when the 1401 is heavily loaded.

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 “data available” input (see the *Sampling data* chapter for the input to use). 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|[Vn+off],OptLB
```

`expr`	The argument should have a value in the range 0 to 255. If a variable or table is used, the bottom 8 bits of the value are used.
`OptLB`	If this optional label is present it sets the next instruction to run, otherwise the next sequential instruction runs.

```
LB:     DIBNE    [.......1],LB >Waiting for bit 0
        REPORT                 ;save a marker when this is set
        MARK    12             ;set code 12 as a digital marker
```

STATE This instruction sets a variable to the current sweep state code.

```
STATE   Vn,OptLb         ; Vn = Sweep state code
```

`Vn`	This variable is updated with the current sweep state code.
`OptLb`	If present, the instruction branches to this label.

This can be used in conjunction with the Signal multiple states mechanism (in Static or Dynamic outputs mode only) to produce a sequence that behaves in a different fashion according to the sweep state.

SETS This instruction sets the state code for the current sweep.

```
SETS   expr|Vn|[Vn+off],OptLB    ; Sweep state = expr
```

`expr`	This is the state code, from 0 to 255 normally.
`Vn`	When a variable or table entry is used to set the state, the value sets the sweep state.
`OptLb`	If present, the instruction branches to this label.

This can be used instead of the normal Signal multiple states mechanisms to generate data files with separate state codes attached to the data sweeps. This instruction should not be used with a sampling configuration which uses multiple states as the two mechanisms will be in conflict; use it in a sampling configuration with multiple states turned off. To match the built-in multiple states mechanism, you should restrict yourself to state codes from 0 to 255. However, you can use other state code values if you wish.

SWEEP This sets a variable to the time of the current sweep in sequencer clock ticks start plus an optional expression.

```
SWEEP   Vn,expr          ; Vn = Sweep start time + expr
```

`Vn`	This variable is updated with the start time of the current sweep.
`expr`	This optional expression will be added to the time.

This can be used to find the start time of the current sweep, or a time within the current sweep. If there is no sweep in progress, the start time of the previous sweep is used.

WSWP This instruction waits until a given time (in sequencer ticks) within the sampling sweep.

```
WSWP   expr|Vn|[Vn+off],OptLB   ; Wait till expr in sweep
```

`expr` This is time within the sweep, in sequencer ticks. Values of `expr` from 1 to the sweep duration specify a time within a sweep, if you specify a time greater than or equal to the sweep duration the sequencer will wait forever. The following values of `expr` have special meanings:

0	Wait until a sampling sweep is in progress
-1	Wait until a sampling sweep is not in progress
-2	Wait until the sampling sweep is armed; the 1401 is ready to accept a trigger but has not been triggered yet

`Vn` When a variable or table entry is used, the value is the time within the sweep in sequencer ticks or one of the special values above.

`OptLb` If present, the instruction branches to this label.

This can be used to pause sequencer execution until a required sweep time is reached; it is the easiest way of synchronising the sequencer with sampling.

TRIG This instruction causes a trigger to start a sampling sweep. If the Free run without restarts box is checked, you can use this in Basic, Outputs frame and Fast triggers. If the box is clear, you can use it in Outputs frame mode only. Using this in other circumstances may cause sampling problems.

```
TRIG            ; Trigger a sweep
```

In Outputs frame sampling mode, you must use `TRIG` to trigger sweeps as external triggers are disabled.

TICKS This instruction sets a variable to the current sampling time in sequencer clock ticks and adds an expression or 0 if `expr` is omitted. The `s()`, `ms()` and `us()` expression functions can be used to make the sequence independent of the clock rate.

```
TICKS   Vn,expr          ; Vn = Signal time + expr
```

`Vn`	This variable is updated with the current time.
`expr`	This optional expression will be added to the time.

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.

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 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.

```
BRAND   LB,expr|Vn|[Vn+off]
```

`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 or table entry is used for a branch, the value 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` 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
```

Arbitrary waveform output

In addition to generating voltage pulses, ramps and cosine waves through the DACs, Signal can play arbitrary waveforms. The sequencer can start waveform output and test it and branch on the result. The sampling configuration sets the size of the area reserved for waveform storage, though not all of that area need be used. The waveforms are stored in 1401 memory and can be updated just before and during sampling with the `SampleSeqWave()` script command.

WAVE The `WAVE` instruction starts arbitrary waveform output from the waveform area.

```
WAVE                ; Start arbitrary waveform output
```

The waveform output area used by the sequencer can only be loaded up with data by using the `SampleSeqWave()` script language function, it can be reloaded during sampling as required.

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 met.
`flag`	An optional single character flag to specify the branch condition: `S` branch if arbitrary waveform output is stopped. `G` branch if arbitrary waveform output is going.

The following (fairly trivial) sequence plays the output waveform 5 times.

```
        MOVI    V1,5       ; load variable 1 with 5
WL:     WAVE               ; start output
WT:     WAVEBR  WT,G       ; wait here while output is going
        DBNZ    V1,WL      ; do this 5 times  >Waiting for cycle
```

The `WAVE` command starts output, it then waits for the output to stop at the `WT` label. Next the sequence repeats this process 5 times.

Sequencer compiler error messages

When you use the Format or Compile buttons in the sequence editor, Signal displays the result of the compilation or format operation in the message bar at the top of the window. The messages report either successful operation or the cause of the problem.

8 Sampling with multiple states

Introduction

In the *Getting started* chapter we met the frame state code; a value from 0 to 256 that is attached to each frame in a Signal data file. The value indicates a condition or classification of the frame and can be used to select data file frames for analysis. This chapter describes the uses and control of multiple states in Signal data acquisition.

What does multiple states sampling do?

Multiple states sampling can do a lot of things, by far the most common (and straightforward) usage is to allow a sampling configuration to have a number of different sets of output pulses (one per state) available and to switch between these outputs during sampling. So, for example, you could have one state that generates a single stimulation pulse on a DAC, another state that has two stimulation pulses separated by 20 milliseconds, and another that has two pulses separated by 40 milliseconds and Signal could then switch between them randomly or in a preset order during sampling. This form of multiple states is known as Dynamic outputs, we will concentrate on this type of multiple states usage because it is the most useful and general-purpose.

State numbers, idling and state 0

With multiple states sampling disabled, all sampling uses the only state available, which is number zero. With multiple states in use you gain a number of extra states, which are numbered from one upwards. The design of Signal expects (though it does not require) that state zero will be reserved for passive or idle outputs rather than for outputs that will form part of the main experimental data – so for example state zero might generate no stimulus at all or maybe a sub-threshold stimulus that allows the health of the preparation to be checked but does not generate any useful data.

This design choice is most visible in the ‘idling’ behavior of Signal states sequencing. Idling means that Signal will stop states sequencing, switch to state zero, and turn off writing to disk. Controls in the states configuration allow you to specify when states sequencing will automatically idle; there is also an Idle button in the states control bar. The built-in states sequencing within Signal also expects this arrangement - for example if you use numeric sequencing with three extra states Signal will run through states 1, 2 and 3 but will not use state zero until the sequencing has finished.

Auxiliary states hardware

In addition to controlling the 1401 outputs, Signal multiple states can be used to control external hardware, for example a MagStim transcranial magnetic stimulator. Such hardware can be set up differently for each state. This is done using specialised software for each type of supported hardware and is documented separately in chapter 20; *Auxiliary states devices.*

What else can multiple states do?

In addition to Dynamic outputs there are two other more specialised forms of multiple states: Static outputs and External digital. These are described at the end of this chapter.

Static outputs mode replaces multiple sets of pulse outputs with multiple sets of unchanging outputs, one for each state. It is not much different from what one could achieve using Dynamic outputs without any output pulses defined so you could only set the initial levels of the outputs. The key extra feature of Static outputs states is that the outputs are set up before the sampling sweep begins, at the time the sweep is enabled (soon after the previous sweep finishes). So Static outputs can be useful if you want to feed controlling information to external equipment such as a stimulator to get it ready to deliver different stimulations – setting the outputs earlier allows the stimulator time to read the outputs and become ready before triggering the sampling sweep.

External digital mode is very different. In this style of operation the 1401 digital inputs are read by Signal at the end of each sampling sweep to retrieve data generated by external stimulation equipment such as a visual stimulator. This digital input data is used to generate the state code value for the frame just sampled. So this form of multiple states is suitable with external equipment which determines the stimulation to be used independently of Signal and outputs digital data to indicate what it has done.

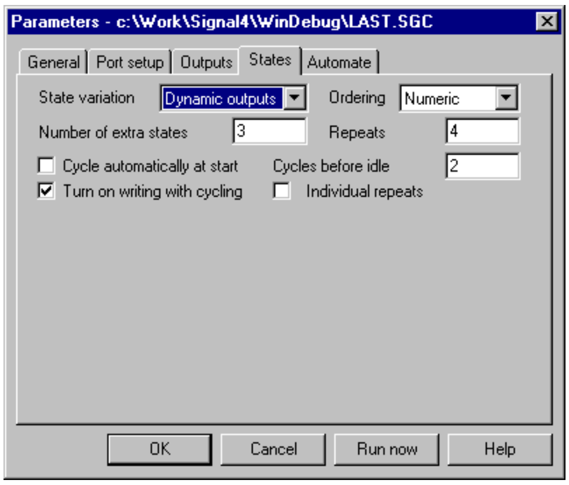
Enabling multiple states

The General page in the sampling configuration dialog contains a checkbox labelled Multiple frame states that enables multiple states in data acquisition. With this checkbox clear, sampling does not use multiple states and all sampled data frames are set to state zero, with it checked, sampling will use multiple states and set the data frames to the appropriate state value. The checkbox is not available with fast triggers, fast fixed interval or gap-free sweep modes as these do not allow for adjustment of the 1401 behaviour between sweeps.

When multiple frame states are enabled, the sampling configuration dialog gains another page labelled States holding controls used to configure multiple states.

Defining multiple states

All configuration of multiple states is carried out using the States page in the sampling configuration dialog.


Parameters - c:\Work\Signal4\WinDebug\LAST.SGC

State variation: Dynamic outputs
Ordering: Numeric
Number of extra states: 3
Repeats: 4
☐ Cycle automatically at start
Cycles before idle: 2
☑ Turn on writing with cycling
☐ Individual repeats

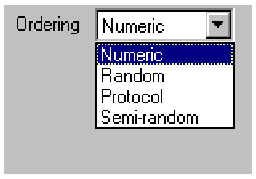
The State variation selector at the top left of the States page selects the type of multiple states to use from External digital, Static outputs and Dynamic outputs. The controls shown in the dialog change dramatically with the type of multiple states that is selected. Unless you are sure that you want to use another form of multiple states you should select Dynamic outputs.

Dynamic outputs states

With Dynamic outputs each state uses a different set of output pulses. The actual digital and DAC outputs for each state (along with some other information such as the state label) are set up using the Pulses dialog available from the Outputs tab of the sampling configuration. The states page is purely concerned with defining how many states there are and how they are sequenced – how and when Signal will switch from state to state during sampling.

The Number of extra states item can be set to any value from 1 to 256, note that this sets the number of states in addition to state zero. Thus in the example shown above there are 4 states possible, with codes running from 0 to 3. In many circumstances Signal will only make use of the extra states and reserves state zero as a background or idle state. This item also controls the states available in the pulses configuration dialog; you should set the number of extra states that you want here and afterwards set up the pulse outputs.

State sequencing



Because Signal controls the states, we need to be able to specify which state is used at what point during sampling. The simplest way to do this is to control the state manually using the states control bar (see below), but we will often want some form of automatic sequencing – its easier, faster and less error-prone. Signal provides two basic forms of states sequencing – numeric or protocol. The three numeric modes provide simple numeric or randomised states sequencing, while using protocol mode allows more complex sequences of states to be generated.

Numeric (non-protocol) sequencing modes

In the numeric states sequencing modes the user specifies how many of each state are to be used, how many times they are to be used overall, how and whether the ordering is to be randomised and what happens when state sequencing starts.

The Repeats item in the dialog sets how many of each state are to be used to make up one cycle of the states sequence. It can be set to any number from 1 to 1000. In the example shown above with a Repeats value of 4 each state from 1 to 3 will be used 4 times in one cycle of the states sequencing.

The Cycles before idle item sets how many cycles (a cycle being states*repeats as described above) of sequencing are wanted before the sequencing stops and Signal switches automatically to idling in state 0. Set this item to the number of cycles you want or to 0 if you want states sequencing to repeat forever until stopped manually.

The Cycle automatically at start checkbox, if set, causes states sequencing to begin automatically when sampling starts (normally sampling starts with Signal idling in state zero, though writing to disk can be turned on). The Turn on writing with cycling checkbox causes writing to disk to be enabled whenever states sequencing is started – this can be quite useful as otherwise this is hard to do neatly and easy to forget.

The Ordering selector sets the precise type of sequencing to be used, this can be set to:

Numeric — In this mode the states are used in numerical order with each state being used the number of times that is set by Repeats. First state 1 is used the specified number of times, then state 2 and so forth. Once the last state has been done, one sequencing cycle has been completed. For the example shown above, with 3 states and 4 repeats, Numeric sequencing would give:

```
1  1  1  1  2  2  2  2  3  3  3  3
```

in one cycle of sequencing.

Random — In this mode, one cycle of the sequencing again uses each state the number of times specified by Repeats, but the order of the states within a cycle is randomised. So, again for the example shown above, Random sequencing might give:

```
2  3  2  1  3  3  1  2  1  3  1  2
```

in one cycle of sequencing (but of course what you actually get will vary). When the sequencing starts another cycle, the states order is re-randomised.

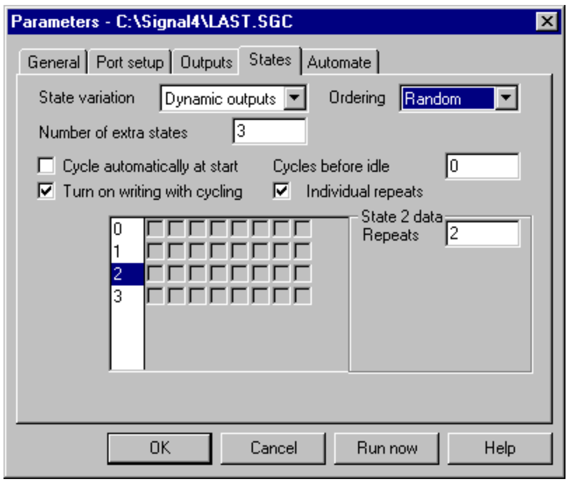
Semi-random This is a slight modification of Random mode where the states are not all randomised across a cycle but instead randomised within one set of states. For the example settings the first 3 frames will always include one of each state (in random order), as will the next 3 and so forth, but one cycle of sequencing still consists of (states * repeats) frames. So you might get:

2 1 3 3 1 2 3 2 1 1 2 3

in a sequencing cycle for the example above. As you will have realised, this mode achieves exactly the same as setting the number of repeats to 1 in Random mode, but a sequencing cycle still consists of each state being used the number of times specified by Repeats.

Protocol This is a non-numeric mode, and is described below. When Protocol mode is selected, all of the checkboxes and controls for repeats and cycles are hidden and replaced by a Protocols… button.

Individual repeats The descriptions above all assume that each state is used Repeats times in one sequencing cycle. The Individual repeats checkbox allows you to set a different repeat count for each state. If it is clear, the Repeats item is used to set how many times each state is repeated. If the checkbox is set, different controls are used to set the repeats wanted for each state separately.


Parameters - C:\Signal4\LAST.SGC

State variation: Dynamic outputs
Ordering: Random
Number of extra states: 3
☐ Cycle automatically at start
Cycles before idle: 0
☑ Turn on writing with cycling
☑ Individual repeats
State 2 data
Repeats: 2

With Individual repeats selected, the overall Repeats item is hidden and the dialog instead shows a state selector and repeat count so that you can set repeats for each state. The state selector (on the left) is used to select a state by clicking on the state number; then you can edit the Repeats value that is shown in the state data box on the right.

Individual repeat counts are used in much the same way as an overall repeat value only each state has a separate count. In Numeric ordering each state is repeated in turn by the specified number of times, while with Randomised ordering each state is repeated the set number of times but the order within each sequencing cycle is randomised. Semi-random ordering does not work very well with individual repeats enabled, as towards the end of a cycle the states with a lower repeat count will be left out, but maybe this behaviour will be useful in some circumstances.

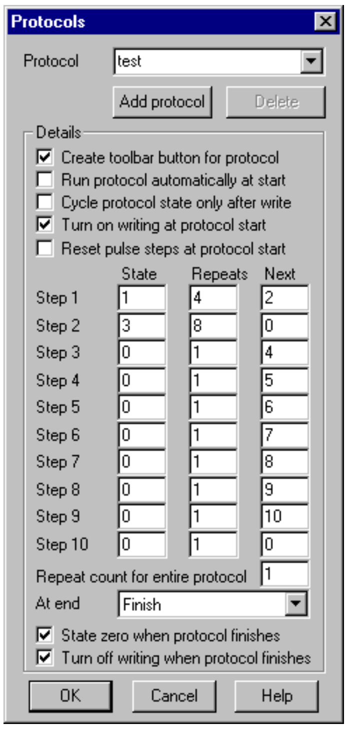
State sequencing using protocols

When Protocol states ordering is selected all of the states page controls apart from the number of extra states are hidden and a Protocol… button is provided that allows you to create, view and edit the protocols.

A protocol consists of a list of steps; each step defines a state that will be used, the number of times it will be used and the step to go to next. There are also controls defining what happens when a protocol starts, and when it finishes. Each protocol can use up to ten steps and protocols can loop and be chained together to produce longer sequences of states. Up to 50 protocols can be defined in a single sampling configuration.

Protocols are defined by using the protocols dialog which is obtained by pressing the Protocol… button on the states page. The dialog has a selector at the top that is used to select a protocol for viewing and editing. To create a new protocol press the Add Protocol button, while the currently selected protocol can be deleted using the Delete button.

The protocol name can be changed by directly editing it in the protocol selector. Blank protocol names are not allowed and will be rejected.



Protocols

Protocol: test

Add protocol | Delete

Details

- [✓] Create toolbar button for protocol
- [ ] Run protocol automatically at start
- [ ] Cycle protocol state only after write
- [✓] Turn on writing at protocol start
- [ ] Reset pulse steps at protocol start

	State	Repeats	Next
Step 1	1	4	2
Step 2	3	8	0
Step 3	0	1	4
Step 4	0	1	5
Step 5	0	1	6
Step 6	0	1	7
Step 7	0	1	8
Step 8	0	1	9
Step 9	0	1	10
Step 10	0	1	0

Repeat count for entire protocol: 1

At end: Finish

- [✓] State zero when protocol finishes
- [✓] Turn off writing when protocol finishes

OK | Cancel | Help

The checkboxes at the top of the protocol details set general options and define what happens at the start of protocol execution – some of them perform similar functions to checkboxes available for numeric sequencing modes. The Create toolbar button for protocol item provides a separate button for this protocol in the states control bar – see the section *Controlling multiple states online*, below. Run protocol automatically at start does what it says; sets this protocol to be run when sampling starts. A protocol that is to be run automatically at the start is indicated by having a '*' character appended to the protocol name. Cycle protocol states only after write controls the sequencing of protocol steps; if it is checked then the protocol sequence does not advance unless data file frames are written to disk, if the data is not written or a sweep is rejected then the same state is repeated. Turn on writing at protocol start causes writing of sampled sweeps to disk to be turned on when the protocol starts execution. Reset pulse steps at protocol start causes any varying pulses that are defined in the output pulses to be reset to their initial state when the protocol starts.

The table below the checkboxes defines the protocol steps. There are ten steps in a protocol, each one with a State, a Repeats count and a Next step. These specify the state to be used, the number of times to repeat this state and the step to go to (from 1 to 10) when this step is done. If you set the next step to zero then this step is the end of the protocol.

When protocol execution begins it starts with step 1. The state set by the State field in step 1 is set and is used the number of times set in the Repeats field. Following this the protocol switches to the step that is set by the Next field. This process continues until it is ended by encountering a Next item of zero. In the example shown above, step 1 repeats state 1 four times and then goes to step 2. This repeats state 3 eight times, after which the protocol ends. If the Next item for step 2 were set to 1, the protocol would run forever or it could be set to 3 for a more complex sequence.

The Repeat count for entire protocol item below the step table controls the number of times that the protocol repeats before it ends. Set this to 1 for a protocol that goes through the steps only once (as in the example above) or set it to a larger number for a protocol that repeats a set number of times before ending. For example, if you set the repeat count to three in the example above, the protocol would run for 36 frames before ending. If you set this item to 0 the protocol will repeat forever until stopped manually.

The items below the repeat count control what happens when protocol execution ends. The At end selector selects either Finish or a protocol that will be ‘chained-to’, chaining to a protocol allows more complex sequences than is possible with just ten steps. If the At end selector is set to Finish protocol execution finishes, otherwise execution switches to the protocol selected and carries on. When a protocol is chained-to, it starts off completely normally, so the Turn on writing and Reset pulse steps checkboxes take effect. These checkboxes do not take effect if the last step in the protocol has a Next item of step 1 or when an entire protocol repeats, so it is meaningful to allow a protocol to chain to itself.

The checkboxes at the bottom of the dialog control what happens when a protocol actually finishes; they are disabled if the protocol chains to another protocol. State zero when protocol finishes enables automatic switching to state zero at the end, if it is clear then the last state set by the protocol will continue to be used. Turn off writing when protocol finishes controls disabling of writing sampled data to disc, both of these checkboxes on together give standard idling behaviour when the protocol finishes.

When the test protocol shown above is executed writing to disk would first be turned on, then the following sequence of states would be generated:

```
1  1  1  1  3  3  3  3  3  3  3  3
```

and finally Signal would idle in state zero, with writing to disk turned off.

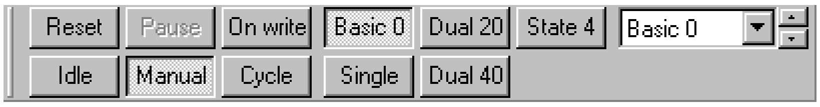
Controlling multiple states online

When sampling using multiple states Signal provides a states control bar to allow you to control the sampling state and states sequencing. The control bar can be docked at the edges of the Signal application or be left floating anywhere within the application. The states control bar will be initially visible when sampling starts, it can be hidden or shown by using the sample menu or a popup menu that is displayed if you right-click on an unused part of the Signal application window.

The layout of controls in the states control bar is depends upon whether a numeric or protocol sequencing style is being used; numeric modes have extra buttons for controlling states sequencing whilst protocol mode has extra items to select and run a protocol and optional buttons to run individual protocols.

Non-protocol ordering

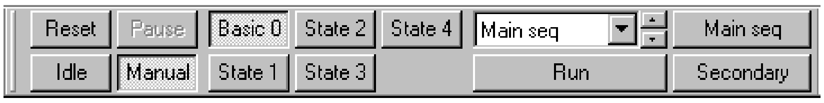
The states control bar contains a number of buttons and controls:



Reset	Press this button to reset any states sequencing in operation (so that the state sequence restarts at the beginning) and also to reset any varying pulses to their initial state.
Idle	Press this button to force states sequencing to idle. It switches to manual control of states, sets state zero and turns off writing to disk.
Pause	Press this button to pause any automatic state sequencing that is in progress. This option does not pause the sampling itself which continues to run using the current state (use the sampling control panel if you want to pause the sampling). While states sequencing is paused this button is shown depressed; press it again to resume sequencing.
Manual	Press this button to terminate any automatic sequencing in progress and to switch to manual control of states. With manual control, the frame state is controlled interactively by the user, sampling begins with manual control selected unless the Cycle automatically at start option is used.
On write	Press this button for automatic states sequencing with the states changing only if the previous data was written to disk. The states sequencer will have control of the frame states and will move on to the next stage after a sampling sweep only if the sweep data was saved to disk. This allows for artefact rejection and interactive sweep accept/rejection without missing out states from the sequence.
Cycle	Press this button to start automatic states sequencing with the states always changing. The state sequencer will have control of the frame states and the sequencer will always move on to the next stage after a sampling sweep regardless of whether the sweep data is written to disk or not.
Basic 0 …	Press these buttons when in manual mode to switch to a state. Buttons for unused states are hidden, as are buttons for states numbers greater than 9. If you have created labels for the states in the pulses configuration dialog these labels will be used in the buttons instead of the standard state names. During automatic states sequencing the state buttons are disabled; Signal depresses them automatically to show the state currently in use.
State n	To the right of the individual state buttons is a state selector and spinner that can be used when in manual mode to choose any state from those available. This selector is most useful when there are more than 9 states, the limit for individual state buttons. As for the buttons, if you have set a label for a state this is shown instead of the simple state name. Selecting a state uses it immediately. During automatic states sequencing the current state is shown.

Protocol ordering

When multiple states are used with protocol ordering some controls in the states control bar are hidden and others are displayed instead. Those that are retained behave in the same way as described above (the On write and Cycle buttons are hidden because the individual protocols select between these two styles of operation). If the number of states is less than ten, so that they can all be represented by the individual state buttons, the main state selector is also hidden to save space. The extra items provided are the protocol selector, the run protocol button and the individual protocol buttons:



Protocol	To the right of the state buttons (or the state selector if it is visible) is a protocol selector and spinner that can be used to select a protocol from those available. Unlike the state selector, selecting a protocol does not execute the protocol. During execution of a protocol the current protocol in use is shown so you can see what is going on if you use chained protocols. If all of the protocols have individual buttons then the protocol selector is not shown.
Run	This button is shown below the protocol selector. Pressing it causes Signal to start executing the selected protocol. If all of the protocols have individual buttons then this is also not shown.
Buttons	These buttons are created for the individual protocols as required, they are arranged to the right of (or below) the protocol selector and are labelled with the protocol name. A protocol button will be created if the Create toolbar button for protocol checkbox in the protocol definition is set and there are not too many protocol buttons – the limit is 20. Pressing one of these buttons causes the relevant protocol to be executed immediately.

Starting a protocol

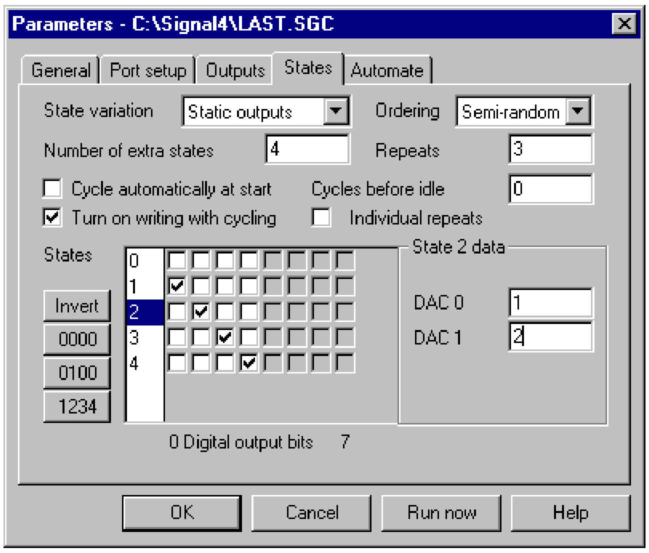
Execution of a protocol can be begun by the user pressing the Run button with that protocol selected (or the button for an individual protocol), or by another protocol chaining to it, or by means of the script language.

Stopping a protocol

Execution of a protocol is stopped when the protocol finishes, by the user pressing the Idle or Manual buttons in the states control bar or by another protocol being started by whatever means.

Static outputs states

Static outputs states are very similar to Dynamic outputs mode, the difference is that instead of having different sets of output pulses it sets up unchanging outputs that are present throughout the sampling sweep. When a sampling sweep is primed, Signal writes values to the 1401 DACs and digital outputs using data for the current state of the experiment. These outputs could select the stimulus to be used or have other effects.


Parameters - C:\Signal4\LAST.SGC

General | Port setup | Outputs | States | Automate

State variation: Static outputs — Ordering: Semi-random
Number of extra states: 4 — Repeats: 3
☐ Cycle automatically at start — Cycles before idle: 0
☑ Turn on writing with cycling — ☐ Individual repeats

States: Invert, 0000, 0100, 1234
State 2 data: DAC 0: 1, DAC 1: 2

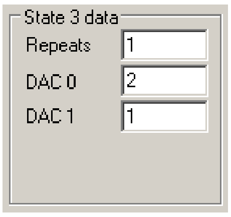
0 Digital output bits 7

The States box to the bottom left of the dialog is used to select a state and to defines the values to be written to the digital outputs for each sweep. The digital output bits available are set in the Outputs page of the sampling configuration, check on a checkbox to set that output high for the relevant state or leave it clear for a low output. The actual digital outputs and connector pins used are the same as for pulse outputs (see the *Pulse outputs during sampling* chapter).

The buttons to the left of the States box set or modify the digital bit patterns in various useful ways, they are intended to allow simple patterns to be set up quickly and to help to produce more complex ones:

Invert	This inverts all of the bits for all states. This is useful for converting all zeroes to all ones and a walking one into a walking zero.
0000	This sets all of the bits for all states to zero.
0100	This sets most bits to zero with a walking 1. This leaves state zero all clear, sets bit 0 for state 1, bit 1 for state 2 and so forth. The pattern is not adjusted to skip disabled digital outputs.
1234	This sets the bit values to count the states using binary code. Thus state 1 has just bit 0 set, state 2 has bit 1 only and state 3 has both bits 0 and 1. Once again the pattern is not adjusted to skip disabled digital outputs.

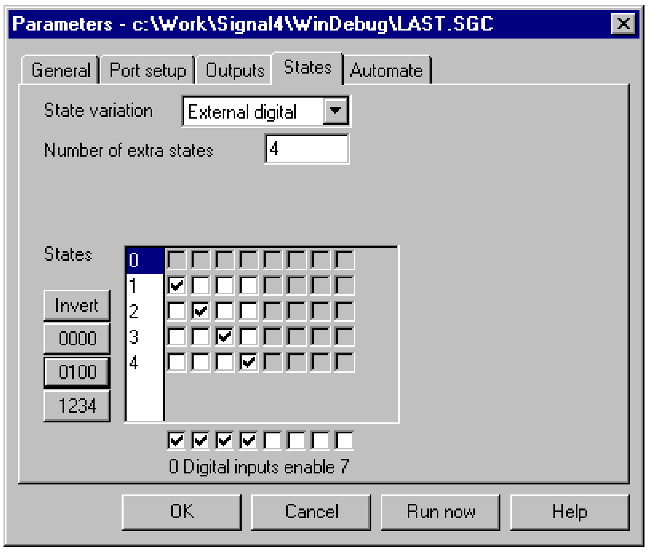
The State data box to the right of the States box sets the DAC outputs for a selected state. You can select the state for which values are shown by clicking on the digital outputs line for that state. The DAC outputs used (0 to 3 only are available) are enabled and disabled in the Outputs page of the configuration. If individual repeats are enabled then the repeat count for a state is set here too.


State 3 data
Repeats: 1
DAC 0: 2
DAC 1: 1

Static output states can only be used with the outputs type set to None in the Outputs page, as otherwise the state values would conflict with other outputs that are generated. When sampling using Static output states, controls for the state and states sequencing are provided in exactly the same manner as for Dynamic outputs.

External digital states

External digital states are very different from the other: forms of multiple states. When using this mode, external equipment generates up to 8 bits of digital code corresponding to the current experiment state. Signal reads this code from the 1401 digital inputs at the end of each sampling sweep and uses this data to set the state for each sampled data frame. There is no internal control over the states so all of the state sequencing controls are hidden.


Parameters - c:\Work\Signal4\WinDebug\LAST.SGC

General | Port setup | Outputs | States | Automate

State variation: External digital

Number of extra states: 4

States

0	☐	☐	☐	☐	☐	☐	☐	☐
1	✓	☐	☐	☐	☐	☐	☐	☐
2	☐	✓	☐	☐	☐	☐	☐	☐
3	☐	☐	✓	☐	☐	☐	☐	☐
4	☐	☐	☐	✓	☐	☐	☐	☐
	✓	✓	✓	✓	☐	☐	☐	☐

0 Digital inputs enable 7

The States box on the bottom left of the dialog is used to define the input bit patterns that correspond to individual states. Input bits 0 to 7 are shown, with 0 on the left, the Digital inputs enable checkboxes below control which inputs are used; disabled inputs are ignored. An unchecked bit corresponds to a zero bit (low or 0 volts) while a checked bit selects a one bit (high or 5 volts). The buttons to the left of the States box modify the bits in various useful ways as documented for Static outputs mode.

During sampling Signal will read the digital inputs at the end of each sampling sweep. The bits read are then checked against the bits for each of the states starting with state 1 and the state for the new frame is set to the first one that matches. If no match is found then the frame state is left set to zero. The bits for state zero are shown disabled as they are ignored.

External digital states uses digital input bits 8 to 15; see the *Sampling data* chapter for details of the digital input connections. Please note that if a digital input is not connected it will read as high; it is thus important to disable any unconnected inputs.

When sampling using External digital states, Signal behaves much as it does with multiple states disabled and no states control bar is shown. The only difference is that the state code value for sampled data frames varies according to the digital inputs.

Auxiliary state hardware

In addition to the standard multiple states facilities that control the 1401 outputs it is possible to install support for auxiliary states hardware. Auxiliary states hardware is separate, external hardware (most often a type of stimulator that cannot be adequately controlled using the 1401 outputs) that is set up in a different way for each state.

Auxiliary states hardware support is provided by means of a DLL that is copied to the Signal installation directory; a separate DLL is supplied for each type of hardware. When auxiliary states hardware is installed an extra button (labelled with the external hardware name) is provided in the States page to allow the hardware settings for each state to be defined. In addition, the `SampleAuxStateParam()` and `SampleAuxStateValue()` script language functions can be used to read and write the auxiliary hardware settings.

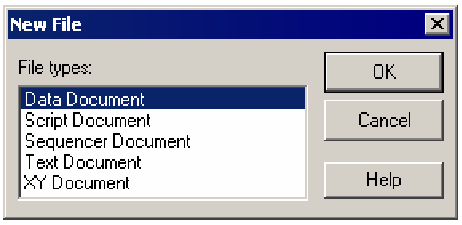
See chapter 22 *'Auxiliary states devices'* for details of the supported auxiliary states hardware.

9 File menu

The File menu is used for operations that are mainly associated with documents (opening, closing and creating), sampling configuration files and with printing. The final command in the File menu is your route out of Signal.

New

This command creates a new Signal file. This can be a sampling document, an XY file, or a new text-based file. You can activate this command with the `Ctrl+N` shortcut key or from the menu or toolbar.



The command opens the New File dialog, in which you select the type of document to create. You can create five types of document: Signal data documents, script documents, sequencer documents, `text` documents and XY documents. Select the type of document, click OK and Signal will open a new window holding an empty document of the specified type.

Data Document

A sampling document window opens plus additional windows as set by the sampling configuration (see the *Sampling data* chapter for details). You cannot create a new sampling document if sampling is already in progress. Sampling documents are not initially stored in memory, like most new files, but are kept on disk. Until they are saved after sampling these are temporary files, without any extension, stored in the directory set by the Filing path in the Automation page of the sampling configuration or if that is blank, the new data directory set in the Signal preferences. When they have been saved, the file name extension is `.cfs` and the files hold CFS (CED Filing System) format files.

Text Document

Text documents can be used to take notes, build reports and to cut and paste text between other windows and applications. The Log view is a specialised type of text document which is always present. The file name extension is `.txt`.

Script Document

A script editor window opens in which a new script can be written, run and debugged. A script document is a specialised form of text document. The file name extension is `.sgs`.

Sequencer Document

A new window opens in which you can type, edit and compile an output sequence (see the *Sequencer output during sampling* chapter for details). The file name extension is `.pls`.

XY Document

XY windows are used to draw user-defined graphs with a wide variety of line and point styles. Although these views can be created interactively, their major use is from the script language. They are also created when generating a trend plot or measurements to an XY view. The file name extension is `.sxy`.

Other file types used by Signal

In addition to the document types listed above, Signal also uses a number of other types of file:

Resource files

Signal creates resource files with the extension `.sgr`. Each resource file is associated with a data file of the same name but with the extension `.cfs`. The resource files hold configuration information about the data file display so that Signal can restore the display on loading. These files are not essential to Signal and if you delete them the associated data file is not damaged in any way, a new default display configuration will be used.

Configuration files

Signal stores sampling configuration information in files with the extension `.sgc`. These store all of the information needed to carry out sampling: the sampling parameters, the position of the sampling control panel and any other windows, the position and display configuration of the data document window (including any duplicate windows) plus any online processing required, the online processing parameters and the position and display configuration of the memory views showing the results of online processing.

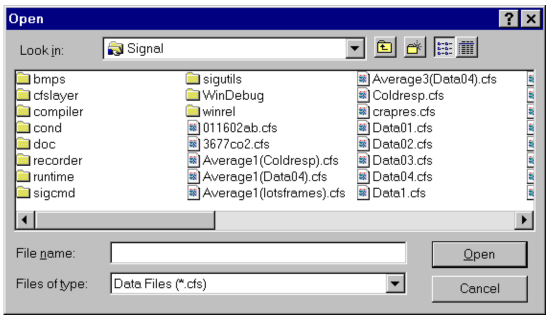
Application preferences

Signal stores some of its preferences in a file called `cfsview.sgp`. This file holds the position of the application window on the screen, the colour palette and 'use colour' switch. The main preferences information from the preferences dialog is now largely stored in the Windows registry. Signal initialises itself with data from this file whenever it is started and saves the current state of the software into this file when it exits.

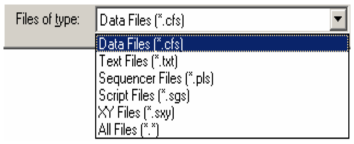
Filterbank files

These files hold descriptions of digital filters and have the extension `.cfb`. They are used by the Analysis menu Digital filters command.

Open



This command opens a file into a Signal document of any type. You can activate this command with the `Ctrl+O` shortcut key or from the menu or toolbar. It shows the standard file open dialog for you to select a file. You can open five types of file with this command: a Signal data file with the standard extension `.cfs`, a text file with the extension `.txt`, a sequencer file with the extension `.pls`, a script file with the extension `.sgs` or an XY data file with the extension `.sxy`. The type of the file is selected with the Files of type field, if this is set to All files(*.*), Signal will try to open the file selected as a CFS file whatever its extension.



When you select a CFS data file, Signal also looks for a file of the same name, but with the file extension `.sgr`. 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 Close All command. See the Close and Close All commands, below, for more details.

If a read-only CFS data file is opened, you will be warned that the file cannot be modified.

If a text file is opened, a simple text edit window is opened. This facility can be used as a notepad or as a repository for data copied from other parts of Signal. If a script, sequencer or XY file is opened, a window of the appropriate type is created for it.

Import data

Signal can translate data files from other formats into Signal data files. The import data command leads to a standard file open dialog in which you select the imported file format and the file to convert. You then set the file name for the result; Signal will suggest the same name with the extension changed to `.cfs`. The details of the conversion depend upon the imported file type.

Supported formats include the SON files used by CED programs such as Spike2 and Spike2 for Macintosh as well as data from many third party vendors. Signal searches the `import` folder in the Signal 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 used to import files is `FileConvert$()`.

Import Op/Cl

Signal can import idealised traces from the CED Patch software for analysis, these are files of the type `*.res` or `*.r??` generated by the Pulsed Analysis or Continuous Analysis programs. You can also import idealised traces from David Colquhoun's SCAN analysis software, these are files of type `*.scn`. To use this feature, first open the relevant `cfs` data file, then import the idealised trace data – the trace data associated with the current data file will be used. Once imported the idealised trace will be stored in the resource file associated with the data file (i.e. the `*.sgr` file) and can be processed using the Analysis menu (see the *Single channel analysis* chapter for more details).

Close

This command is used to close the active window. It is equivalent to double-clicking the control menu icon at the top left of the window (in windows that have one) or pressing the right-hand top corner **X** button. If you use this command on a new memory or XY view, a newly sampled data document or a text-based view with text that has not been saved, a dialog will ask if you wish to save the text before closing the window. This behaviour can be disabled for memory and XY views using an option in the Preferences dialog.

Close All

This command closes the current window and all windows associated with the file. In addition to saving the state of the data file in a `.sgr` resource file, Signal offers to save the state and contents of any memory view windows that belong to the data file. Next time you open the data file, all the data view windows and their contents will be restored. If you open saved memory view data then that will be opened as a file view and restored to its previous state as well.

Save, Save As

Save will save the current document under its current name, unless it is unnamed, in which case you are prompted for a name before it is saved. Save As is used to save the document with a different name, leaving the original file intact. The Save command is not available for a data document unless the data has been changed since the last save.

Data documents

Data files are kept on disk, not in memory, as they can be very large. Changes made to a data file are permanent as they are made on disk. When you save newly sampled data, you are giving the data file on disk a name (replacing a temporary name). If you save it to a different drive from that set in the Preferences, Signal copies the file to the new drive and deletes the original. If the file is large this operation can take a noticeable time.

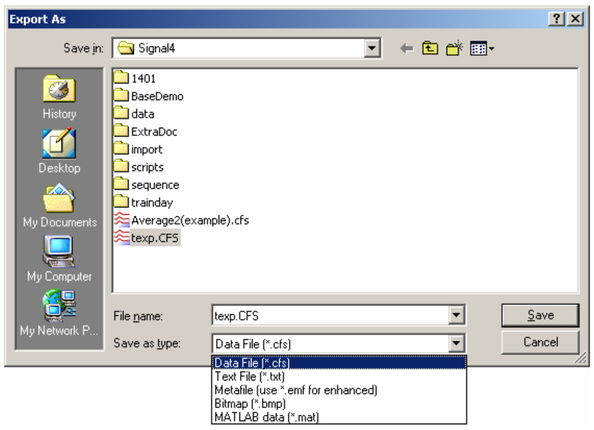
When you are working with a file view, the current frame for the view is held in memory and will be discarded when the view switches to a different frame. Any changes made to the frame data while it is held in memory must be saved before the new frame is loaded or the changes will be lost. You can write changed data back into the file using the Save command. The Preferences dialog allows you to select what happens if the frame is changed while data is unsaved, the default action is to query the user. Changes made to non-channel frame data (such as the frame state or flags) are always saved. Memory views hold all frames in memory, changes are not saved until the document is saved.

Other documents

Text, script and XY documents are held in memory. Changes made to them are not permanent until the document is saved to disk.

Export As

This menu item, available only when the current window is a data or XY view, is used to save some or all of the view data to a new file in one of a number of formats. The dialog prompts you to choose a file name for the output, and lets you select the output format. You can choose between: Data file (*.cfs) to export as a new Signal data file, Text file (*.txt), Metafile (*.wmf, *.emf) for a scaleable image and Bitmap file (*.bmp) for a copy of the screen rectangle containing the window. More file types will be listed if you have installed external exporters. Select one of the formats and set a file name, then use the Save button.



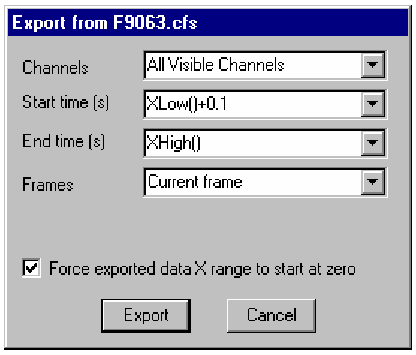
File name: texp.CFS

Save as type: Data File (*.cfs)

Data file

This option opens a dialog in which you can create a new data file from a time range and selected channels and frames of the current file.

Use the dialog to select the channels, time range within the frames, and the frames to be exported. You can choose all the channels or all visible channels, individual channels, selected channels or enter a list of channel numbers directly. You can export all frames, the current frame, all tagged or un-tagged frames, frames with a given state, or you can enter a list of frames. The check box Force exported data X range to start at zero shifts the times of data points in the new file so that the time for the first point in each frame is forced to be zero.


Export from F9063.cfs

Channels: All Visible Channels
Start time (s): XLow()+0.1
End time (s): XHigh()
Frames: Current frame

[✓] Force exported data X range to start at zero

Once you have selected the channels, frames and a time range, click the Export button to write the data to the new data file or Cancel to quit.

XY file This option, available for XY views only, saves all the XY view data to a new 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 that will be used.

Bitmap file This 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. You should only 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 Windows Metafile copy is usually better.

Metafile This copies the window as a Windows Metafile or enhanced Metafile, use the wmf file extension for a normal metafile and emf for an enhanced metafile. This file format can be scaled without losing any resolution and is usually the preferred format for moving Signal images to drawing programs or into reports. Note however that some drawing programs can have problems using metafiles, particularly if you are using XY views with channels that are drawn filled.

External exporters These use the same dialog as for exporting to a data file to select the channels, frames and time range. There may be additional dialogs depending on the target format. See the Export folder for additional documentation for external exporters. The only external exporter currently defined is for MatLab files.

Backup file.sgr

You can use this command with any type of data file. Information relating to the current file is stored in an associated `.sgr` file, this includes things like the draw modes of the channels but more significantly it contains any idealised trace data associated with the file. In a lengthy idealised trace editing session you may wish to backup the `.sgr` file to ensure edits are not lost. This option creates a new file called `filename.backup.sgr` where filename is the stem of the original data file name, this file will contain all of the idealised trace data at the time of backup. In the unlikely event of needing to retrieve the `.sgr` file you should first make sure that the original data file is not open and then copy the backup `.sgr` file to `filename.sgr`, it will be automatically picked up when the data file is opened.

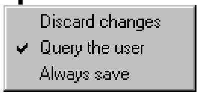
Revert To Saved

You can use this command with a text file, sequencer file or a script file. The file changes back to the state it was in at the time of the last save to disk.

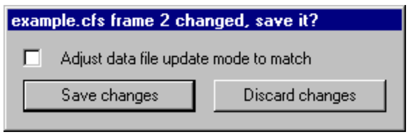
Send Mail

If your system has support for email installed, you can use this command to send any document from Signal to another linked computer. The command vanishes if you do not have mail support on your PC. Text-based documents, XY views and memory views can be sent even if they have not been saved to disk (Signal writes them to a temporary file if they have not been saved). Signal data files can be sent as long as they have already been saved to disk.

Data update mode



This command controls how and if changed file view data is written back to the CFS data file. Signal holds the data for the current frame in memory where it can be accessed and modified by script commands or by the channel data manipulation commands. When the file is closed, or the view switches to another frame, the data update mode determines what happens if there is changed data in memory. Changes to the frame data can be written back to the disk, or they can be discarded, or the user can be asked to choose what is to be done. The Edit menu Preferences dialog sets the default data update mode for all files, this command changes the mode for the file attached to the current view.


example.cfs frame 2 changed, save it?
☐ Adjust data file update mode to match
Save changes
Discard changes

If Query the user mode is selected, the Save changes dialog will appear when required. This allows changed frame data to be interactively discarded or saved, check the Adjust data file update mode to match checkbox if you don't want to see the dialog again for this file.

Load and Save Configuration As

These commands manage Signal configuration files. These hold the sampling parameters, the window arrangement required during sampling, the output setup and the types of on-line analysis required. Signal always has one configuration loaded, this is the configuration used for sampling and the sampling configuration dialogs.

The Load Configuration and Save Configuration As commands transfer the Signal configuration between disk file and the application. They both open an appropriate file dialog to select a file for loading or saving to.

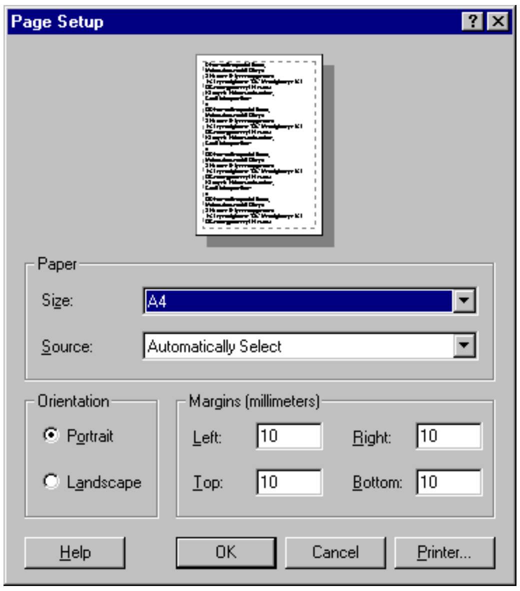
The Save Default Configuration command saves the current configuration as the default configuration which will be automatically loaded whenever Signal is started.

Default configuration files
default.sgc
last.sgc

If the configuration file `default.sgc` exists in the Signal program directory, it is loaded when Signal starts. You can save the current configuration in this file by using the Save Default Configuration command. There is also the standard file `last.sgc` that holds the last configuration that was successfully used for sampling. If `default.sgc` cannot be found, and `last.sgc` exists, `last.sgc` is loaded. Signal saves the current configuration in `last.sgc` each time sampling finishes successfully.

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. 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.


Page Setup

Paper
Size: A4
Source: Automatically Select

Orientation
[✓ Portrait] Landscape

Margins (millimeters)
Left: 10 Right: 10
Top: 10 Bottom: 10

The Orientation option (portrait or landscape) applies to all output except the Print Screen command, which has it's own selector for the orientation.

The printer margins will appear in inches or millimetres, depending upon the locale set for your computer. These margins are used for all printed output. The left and right margins are applied to everything including headers and footers. The top and bottom margins apply to everything except headers and footers which have their own top and bottom margin settings (see the Page Headers command). The top and bottom margins set here are further modified if a header or footer would collide with them.

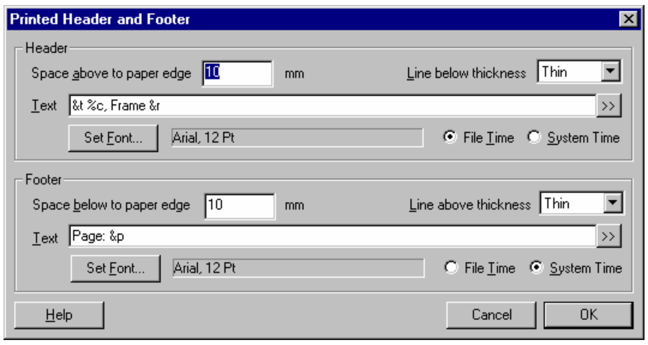
Most printers have an unprintable area near the paper edge. If you set margins that are smaller than this unprintable area, the margins are increased so that all output is visible. If you set margins that reduce the printable area too much, the margins are ignored.

Registry use

Signal saves the printer margins in units of 0.01 mm in the system registry. You can find them in the `HKEY_CURRENT_USER\Software\CED\Signal\PageSetup` folder as `REG_DWORD` values: `LeftMargin`, `RightMargin`, `TopMargin` and `BottomMargin`.

Page Headers

You can apply headers and footers to all printed output. The headers set in this dialog are used with all File, Memory, XY and text-based views. Other printed output (for example from the Print Screen command) uses all of these settings except for the text, which each command provides separately.


Printed Header and Footer

Header
Space above to paper edge 10 mm
Line below thickness Thin
Text &t %c, Frame &r
Set Font... Arial, 12 Pt
[✓ File Time] System Time

Footer
Space below to paper edge 10 mm
Line above thickness Thin
Text Page: &p
Set Font... Arial, 12 Pt
File Time [✓ System Time]

Header and footer positions

The horizontal position is set by the left and right margins in the Page Setup dialog. The vertical positions are set by the space above the header and the space below the footer fields in inches or millimetres, depending on the locale. If your header or footer encroaches on the top and bottom margins set in the Page Setup dialog, the top and bottom values are adjusted to keep the output clear of the header and footer.

Line thickness

You can choose between: None, Hairline, Thin, Medium and Thick. A Hairline is the thinnest line possible on the output device. The other settings should be self-explanatory.

The header line is drawn below any header text, the footer line is drawn above any footer text. If there is no header or footer text, no line is drawn.

Text This is the text to display as the header or footer. If this field is empty, the header or footer is omitted (including any line). You can split the text into left-justified, centred and right-justified sections with the vertical bar character. You can also insert codes that are replaced by document and time information. The simplest way to do this is by clicking the >> button to the right of the text and choosing an option.

Set Font Click this button to choose a font for the header and the footer. Font sizes are limited to the range 2 to 30 points.

File/System Time You can insert times into both the header and the footer. However, you have to choose between the current time and the file time. This allows you to display the file time in the header and the current time in the footer, or vice versa.

Document information The codes shown below are replaced by the relevant document information. Except where an alternate effect is shown for upper case, upper and lower case are treated alike. A single ampersand '`&`' character can be inserted using a double ampersand '`&&`' code.

`&f`	File and path	`&F`	Upper case file and path
`&t`	Document title	`&T`	Upper case document title
`&p`	Page number	`&n`	Total number of pages
`&a`	Absolute frame time	`&s`	Frame state code
`&g`	Frame flags	`&c`	Frame comment
`&r`	Current frame number	`&l`	Overdrawn frame list

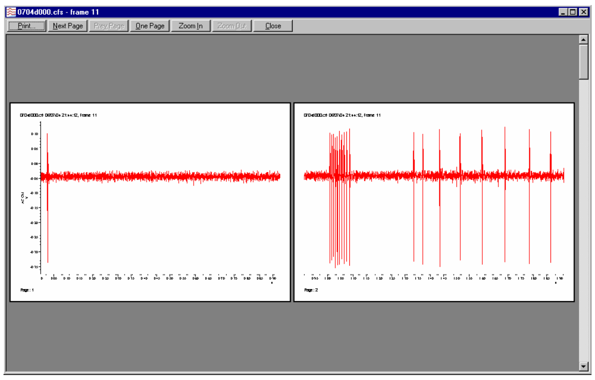
Time and date codes You can insert times and dates using % followed by a character code. The combination is replaced as described in the table. You can also use `%#c` and `%#x` for a long version of dates and times. You can remove leading zeros from numbers with `#`, for example `%#j`.

`%a`	Short day of week (Mon)	`%p`	Indicator for A.M or P.M.
`%A`	Long day of week (Monday)	`%S`	Seconds as number (00-59)
`%b`	Short month name (Jan)	`%U`	Week of year, Sunday based (00-53)
`%B`	Long month name (January)	`%W`	Week of year, Monday based (00-53)
`%c`	Date and time for locale	`%w`	Sunday based weekday (0-6)
`%d`	Day of month (01-31)	`%x`	Date formatted for locale
`%H`	Hour in 24 hour format (00-23)	`%X`	Time formatted for locale
`%I`	Hour in 12 hour format (01-12)	`%y`	Year without century (00-99)
`%j`	Day of year (001-366)	`%Y`	Year with century, e.g. 2004
`%m`	Month as number (01-12)	`%z/Z`	Time zone name
`%M`	Minute as number (00-59)	`%%`	The percent sign

Registry use Signal saves the header and footer settings in the system registry. You can find them in the `HKEY_CURRENT_USER\Software\CED\Signal\PageSetup` folder as strings: `Header`, `Header info`, `Footer` and `Footer info`. The `Header` and `Footer` items hold the text strings. The two `info` items are strings that code up the font name, point size, bold and italic settings, distance to the paper edge in units of 0.01 mm, line thickness (0=none, 1=Hairline, 2=Thin, 3=Medium, 4=Thick), and File time (0) or System time (1).

Print Preview

This option displays the current window as it would be printed by the Print option. You can preview file, memory, XY and text based windows. You can zoom in and out, view 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.

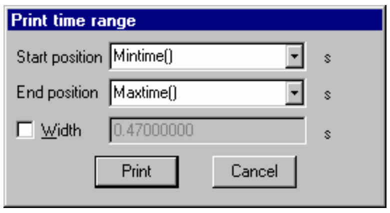


In previous version of Signal, the print preview window took over the entire program and no other commands were available apart from those on the preview toolbar. From Signal version 4, the preview takes place inside the frame of the time, result or XY view. You can now access the File menu to change the headers and footers and print margins while within preview. If you use the menus to make changes to the contents of the displayed data, the screen may not display the changes until you exit from Print Preview mode.

Print visible, Print and Print selection

These commands print data views, XY views and text-based windows. The scroll bar at the bottom of the window is not printed. Print selection prints the selected area of a cursor or text window. Print visible prints the visible data in the current window. Print or its `Ctrl+P` shortcut prints a specified region of a data view at the scaling in the window (one page of paper holds the same x axis range as the current display, the output spans as many printer pages as are required to show the data selected). You must set the print range in a dialog, either by typing the start and end times directly, or by selecting them from the pop-up menus.

To print an entire frame, move to the frame required, set the visible width of the view to the x axis range per page required in the print, then select Print. Select Mintime() for the start position and Maxtime() for the end position. Print doesn't print multiple frames, this may be provided in later versions of Signal.

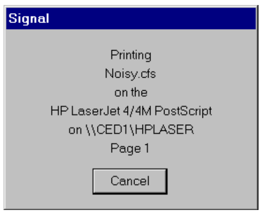

Print time range

Start position	Mintime()	s
End position	Maxtime()	s
☐ Width	0.47000000	s

Print Cancel

All of the print commands open the standard print dialog for your printer. You can set the print quality you require (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.

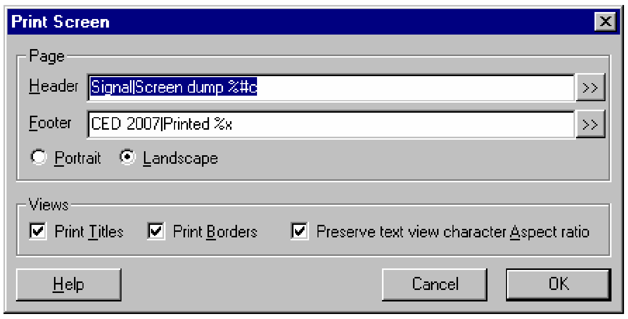

Signal

Printing
Noisy.cfs
on the
HP LaserJet 4/4M PostScript
on \\CED1\HPLASER
Page 1

Cancel

Print screen

The Print Screen command prints all file, memory, 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 page margins are those set by the Page Setup and Page Headers dialogs.


Print Screen

Page

Header Signal|Screen dump %#c

Footer CED 2007|Printed %x

Portrait [✓ Landscape]

Views

[✓] Print Titles [✓] Print Borders [✓] Preserve text view character Aspect ratio

The command opens a dialog with two regions: Page and Views. In the Page region you can set a page header and footer and choose to print in Landscape or Portrait mode. The header text is printed with an underline across the width of the page. The footer text is printed with a line over it across the page width. The fonts used and line thicknesses are as set in the Page Headers dialog If the header or footer is blank, both it and the associated line are omitted.

You can divide the header and footer into a left justified, centred and right justified sections with the vertical bar character, for example: Left|Centered|Right. You can include the current date and time in the header or footer by including, for example `%c`, as described for the Page Headers dialog. The >> button can be used to insert the time and date and vertical bar separators into the header and footer.

In the Views region you can choose to print view titles and draw a box round each view. You can also ask Signal to attempt to preserve the aspect ratio of characters in text windows. Without this, characters are scaled in an attempt to match the printed output to the text displayed in the view. However, this may not produce a very legible output.

Registry use

Signal saves the Print Screen settings in the system registry. You can find them in the `HKEY_CURRENT_USER\Software\CED\Signal\PageSetup` folder. The text strings are saved as `PSHeader` and `PSFooter`. The remaining values are saved as `REG_DWORD` values of 0 (not selected) and 1(selected): `PSViewTitle`, `PSBorder`, `PSScaleText` and `PSLandscape`.

Exit

This command will close all open files and exit from Signal. If there are any data documents or text-based files open containing changes that have not been saved, you will be prompted to save them before the application terminates.

10 Edit menu

This menu holds the standard Edit menu functions that all Windows programs provide. The majority of the 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-based view the Edit menu operates in the standard manner; you can cut and paste text between Signal text windows and other applications. When the current window holds a Signal data document, the behaviour is modified.

Undo and Redo

In a text-based view this is used to undo or redo the last text edit operation. In Signal data views you can undo display scaling operations (for example when you drag a rectangle over a channel to zoom in or out or use the X or Y axis dialogs) and most other operations that change the appearance of the data window. Changes made to an idealised trace formed from patch clamp data may also be undone. A multiple-level undo list is maintained.

Cut

You can cut selected portions of editable text to the clipboard from any position in Signal where the text pointer is visible. This includes any text or numeric fields in dialogs and all text-based views. You cannot use this command in Signal data document windows.

Copy

You can copy selected portions of editable text to the clipboard. If you use this command from a data document window, the window contents, less the scroll bar, are copied to the clipboard as text, a bitmap and also as a metafile. It is also copied as binary data that can be pasted into an XY view or the pulses dialog. See also Copy As Text.

Text output

The visible data is copied to the clipboard as text. The text format used is the same as was last used in Export As with text or the Copy as Text command. See the Copy As Text command for details of the text output format.

Bitmap output

Make sure that the window is completely on the screen and that it is not covered by any other window before copying. Use this option when the required image is an exact copy of the screen. If you need to scale or edit the image Metafile format is usually better.

Metafile output

The screen display is copied to the clipboard as a Windows Metafile. You can read an exported image into a drawing program as either a bitmap or as a metafile. Metafiles are often the preferred choice as you can treat the image as lines and text for editing. Use Paste Special and select Picture in the target application to select the metafile image.

CED binary format

The visible waveform data is copied to the clipboard as binary (numbers) using a private CED format. This binary data can be pasted into the arbitrary waveform buffer used by the pulses dialog (see the *Pulse outputs during sampling* chapter), into another data view or into an XY view. This private clipboard format is not usable by other applications.

Paste

You can paste text on the clipboard into any text-based document, or any text or numeric field in a dialog with the `Ctrl+V` key combination. Clipboard data using the private CED binary format can be pasted into compatible data views, an XY view or into the arbitrary waveforms output by the pulse dialog. Data is pasted into views within the displayed limits, if the clipboard holds more points or channels that the visible data, only the visible data is modified. This is not true for XY views, where all of the clipboard data is inserted as new channels. If the binary data holds fewer points or channels than the displayed data, only part of the visible data is modified. Pasting from the clipboard does not affect the data stored on the clipboard.

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 memory data view, this command will set all the bins in all channels to zero and redraw the window contents. In an XY view whose data is created by processing, this command removes all data points. If you are looking at frame zero in sampled data, Clear erases any overdrawn traces. Clear removes everything, Delete removes the current selection.

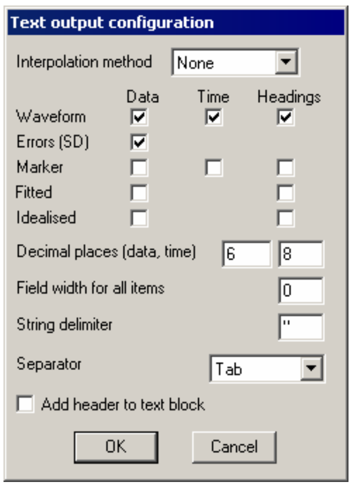
Select All

This command is available in text-based views and selects all the text, usually in preparation for a copy to the Clipboard command.

Copy As Text ... (Data view)

This command copies data views to the clipboard as text. Text representations of Signal data can be very large and awkward to manipulate with the clipboard; as an alternative you can write the text output to a file with the File menu Export As command. The command leads to dialogs that set the text output format and the data to copy:

Text output configuration



Text output configuration

Interpolation method: None

	Data	Time	Headings
Waveform	✓	✓	✓
Errors (SD)	✓		
Marker	☐	☐	☐
Fitted	☐		☐
Idealised	☐		☐

Decimal places (data, time): 6 8
Field width for all items: 0
String delimiter: "
Separator: Tab
☐ Add header to text block
OK Cancel

This dialog sets the text output format. The first section sets an interpolation method for Waveform channels. This is because the Waveform channels will be output in columns with each row showing data sampled at a particular time. If the data is not sampled in burst mode then the data points in the file will not all be sampled at the same time. This will result in a small error as data will appear shifted slightly to bring all the points into line. This can be overcome by using interpolation to estimate where the waveform on a given channel would have been at the time a point on another channel was sampled. Linear and Cubic Spline interpolations are available.

In the next section you can choose whether to output Headings, Data values and Time values (where appropriate) for each of Waveform, Errors, Marker, Fitted and Idealised trace data by checking the boxes. Error data is always the standard deviation and is not generated unless waveform data is dumped.

Output field types All output is written in fields that are either numeric or text. A text field is a sequence of characters that may include spaces. Text fields may hold numbers, but numeric fields cannot hold text.

Separator Signal outputs a separator between each field. It can be set to `Tab`, space or comma with the Separator selector. The examples below use `Tab` as a separator.

Decimal places You can set the number of decimal places to use for both data and time output.

Field width for all items This field sets the minimum width of each output field, in characters, leave this set to zero for the default field width.

String delimiter You can mark the start and end of a text field with special characters (usually ") so that a program reading the field can include spaces and punctuation within a field without confusion. Use this field to set a one or two character delimiter, or leave it blank. The examples use " as a delimiter.

Add header to text block

If you check this box, Signal outputs a header of the file name followed by the frame number before the text output. This header is normally disabled as it may interfere with reading exported Signal data into spreadsheets.

```
"Noisy.cfs"    "Frame 4"
```

Data output

All waveform data is output together, with the first column holding the time values, if enabled, and then one column per waveform channel. Waveform headings are a single line holding a title for each text column. Following this are multiple lines with the waveform data times and values:

```
"s"        "ADC 0"    "ADC 1"    "ADC 2"    "ADC 3"
0.23675    1.01074    4.61670    0.46875    0.23438
0.23700    1.37451    4.61182    0.44678    0.27832
...
0.33350    -1.69922   -0.18799   0.43945    0.21729
```

Each Marker channel is output separately. The output starts with the channel headings, if these are enabled. The data follows in two columns: a time followed by a character.

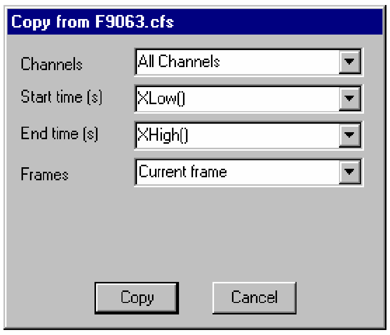
```
"s"            "Keyboard"
20.20608       "a"
21.24544       "n"
```

Fitted data is output with two lines of headings, if enabled, followed by the data.

```
"Single exponential fit on channel 1"
"Amplitude 1"  "Time constant 1"  "Offset"
0.27061        0.00575            -0.20811
```

Copy data selection

Once you have defined the output format, you have to select the data to copy. This is done using the same dialog that is used to select the data to be exported to a CFS or text file. You can specify the channels to use, the time range for the data within each frame and the frames to be used, if you select Frames with state = xxx a separate field appears to set the state in the usual manner.


Copy from F9063.cfs
Channels: All Channels
Start time (s): XLow()
End time (s): XHigh()
Frames: Current frame

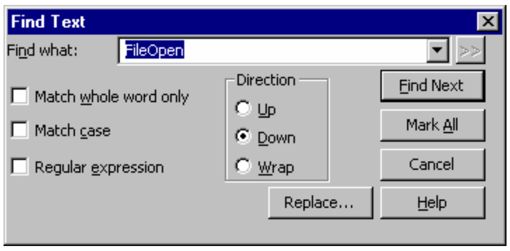
When you are satisfied with the selection, click on Cancel to quit or Copy to start the copy process.

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. The first line of output for a channel 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 Y value, separated by a tab character.

Find
Find Again
Find Last

The Edit menu Find command opens the Find Text dialog. The dialog is shared between all text-based views. It is closely linked to the Find and Replace Text dialog and shares all its fields with it; opening this dialog closes the Find and Replace dialog. Click Replace... to swap to the Find and


Find Text
Find what: FileOpen
☐ Match whole word only
☐ Match case
☐ Regular expression
Direction: ○ Up [✓ Down] ○ Wrap

Replace dialog. A successful search moves the text selection to the next matching string. Searches are line by line; you cannot search for text that spans more than one line.

Find Next starts a search for the text in the Find what field. The Mark All button sets a bookmark on all lines that match the search text, but does not move the selection. Searches are insensitive to the case of characters, unless you check Match Case. Check the Match whole word only box to restrict the search so that the first search character must be the start of a word and the last search character must be the end of a word.

Search direction

Select `Up` to search backwards, towards the start of the text. Select `Down` to search forwards towards the end of the text. Select `Wrap` to search forwards to the end, then wrap around to the start and stop when you reach the current position. Searches do not include the currently selected text.

Regular expression

Check this box to search for *regular expressions*. This disables Match whole word only as regular expressions have their own way to match word starts and word ends. It also disables Up searches; regular expression searches are forwards only. It enables the >> button, which displays a list of regular expressions to insert into the search string. The simplest pattern matching characters are:

`^`	Start-of-line marker. Must be at the start of the search text or it just matches itself. The following search text will only be matched if it is found at the start of a line.
`$`	End-of-line marker. Must be at the end of the search text or it just matches itself. 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 the third character.

You can use `\a`, `\b`, `\f`, `\n`, `\r`, `\t` and `\v` to match the ASCII characters BELL, BS (backspace), FF (form feed), LF (line feed), CR (carriage return), TAB and VT (vertical tab). Searching for `\n` and `\r` will not normally match anything as `\n` or `\r\n` mark line ends and the search is of complete lines ignoring end of line markers. `\xnn` can be used to search for an ASCII character with hexadecimal code `nn`.

To search for one of a list of alternative characters, enclose the list 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, `[a-zA-Z0-9]` matches any alphanumeric character. To include the `-` character in a search, place it first or last. To include `]` in the list, place it first. To search for any character that is not in a list, place a `^` as the first character. For example, `[^aeiou]` finds any non-vowel character.

You can search for the start and end of a word with `\<` and `\>`. Word characters are the set `[a-zA-Z0-9]`, or `[a-zA-Z0-9%$]` in script views. For example, the regular expression `\<[a-zA-Z]+\>` will match a text word, but not if it contains numbers. You can also use `\w` to match a word character and `\W` to match not a non-word character. Likewise, `\d` matches a decimal number and `\D` to matches a non-number character and `\s` matches white space (space, TAB, FF, LF and VT) and `\S` matches non-white space. You can use `\w`, `\W`, `\d`, `\D`, `\s` and `\S` both inside and outside square brackets.

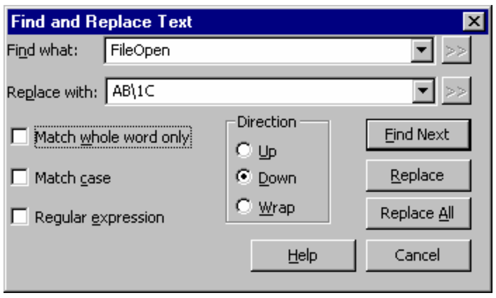
There are search characters that control how many times to find a particular character. These characters follow the character to search for:

`*`	Match 0 or more of the previous character. So `51*2` matches `52`, `512`, `5112`, `51112` and `h.*l` matches `hl`, `hel`, `hail` and `B[aeiou]*r` matches `Br`, `Bear` and `Beer`.
`+`	Match 1 or more. The same as "*", but there must be at least one matching character.

You can also tag sections of matched text by wrapping it in `\(` and `\)`. You can then insert the tagged text later in the regular expression (or in the replace text in the Find and Replace dialog) using `\n`, where `n` is 1 for the first remembered text, up to `9` for the ninth. For example, `\(foo\)-\1` matches `foo-foo`. More interestingly, the regular expression `\(\<[a-zA-Z]+\>\)-\1` matches `Jim-Jim`, `plum-plum` and the like.

Find Again and Find Last repeat the previous search forwards or backwards.

Replace



Find what: FileOpen

Replace with: AB\1C

Match whole word only [ ]
Match case [ ]
Regular expression [ ]
Direction: Up ( ) [✓ Down] Wrap ( )

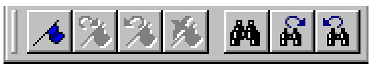
The Edit menu Replace command is available when the current view is text-based. It opens the Find and Replace Text dialog in which you can search for text matching a pattern and optionally replace it. The search part of the dialog is identical to the Find Text dialog. The search pattern set by the `Find what` field can be a simple match, or can be a regular expression. In regular expression searches, the replacement text can refer back to tagged matches in the search text. See the Find Dialog for details of regular expressions and tagged matches. The >> buttons are also enabled in regular expression mode and let you insert expressions into the search and replace text.

Replace with This field holds the text to replace the matched search text. In a regular expression search, you can include tagged matches from the search text using `\1` to `\9` as described for the Find dialog. For example, suppose you have variables named `fred0` to `fred17` that you want to convert into an array `fred[0]` to `fred[17]`. You can do this by setting the Find what field to `\<fred\(\d+\)\>` and the Replace with field to `fred[\1]`.

Replace The Replace button tests if the current selection matches the Find what field and if it does, the field is replaced by the Replace with text field and the selection is moved to the next match. If the selection does not match, Replace is equivalent to Find Next.

Replace All This button searches for all matches in a forward direction starting from the beginning of the text to the end, and replaces them.

Edit toolbar

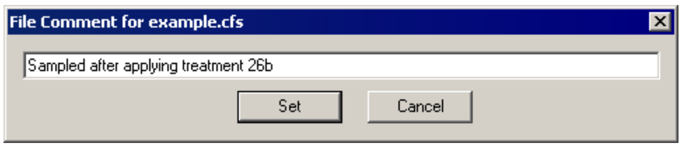


The Edit toolbar gives you access to the edit window bookmarks and short cuts to the Find, Find next, Find previous and Replace commands. If you are unsure of the action of a button, move the mouse pointer over the button and leave it for a second or so; a “ToolTip” will reveal the button function and any short-cut key associated with it.

The four bookmark functions toggle a bookmark on the current line, go to the next or previous bookmark and clear all bookmarks. You can set bookmarks on all lines containing a search string with the Edit menu Find command.

If the Edit toolbar is not visible, click the right mouse button on an empty area of a toolbar or of the Signal main window. Then select Edit bar in the pop-up menu. You can use the same procedure to hide it. The bar can be docked to any side of the application main window or dragged off the application main window as a floating bar.

File comment


File Comment for example.cfs

Sampled after applying treatment 26b

This command is available with file and memory views, it allows you to see and edit the file comment. The file comment is a single line of text attached to the data file; it can be entered at the end of sampling or by using this command.

Frame comment

This command is available with file and memory views, it allows you to see and edit the frame comment. The frame comment is a single line of text attached to each frame in the data file that is available for any purpose.

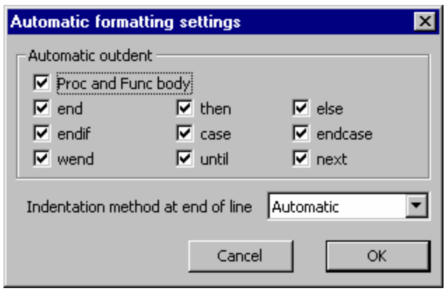
Auto Format

Automatic formatting is available for script views only and applies standard formatting while you type and lets you reformat entire scripts or selected script regions. Formatting is done by indenting lines of text based on the script keywords. An indented line is one that starts with white space. The indenting is in units of the Tab size set for the view in the Script Editor Settings dialog. Indentation is done with Tab characters if you have chosen to keep Tabs in the view, otherwise indentation is done with space characters. There are two sub-commands:

Apply Formatting

If any text is selected in the current view, all lines included in the selection are formatted. If there is no selection, the entire document is formatted. If text is selected, you can right-click in the text view and choose Auto Format Selection from the pop-up menu to activate this option.

Settings...


Automatic formatting settings

Automatic outdent

☑ Proc and Func body

☑ end	☑ then	☑ else
☑ endif	☑ case	☑ endcase
☑ wend	☑ until	☑ next

Indentation method at end of line: Automatic

The Auto Format Settings dialog controls how text is formatted. Formatting is based on the same scheme that is used for folding text. Each script keyword that starts a block construction (`proc`, `func`, `if`, `docase`, `while`, `repeat`, `for`) increases the indent, and the keywords that complete blocks decrease the indent. However, many users prefer the look of the text with some extra outdents.

The standard CED formatting is to have all the boxes checked, however, it makes no difference to the script operation so, what you choose is a matter of personal taste. It is a good idea to use consistent indenting as it helps you to understand the structure of the script.

Proc and Func body

If this box is not checked, all text within a function or procedure is indented. Check the box to outdent the text between the `Proc` or `Func` and the `end`.

end...next

You can choose to outdent any line starting with one of the keywords `end`, `then`, `else`, `endif`, `case`, `endcase`, `wend`, `until` and `next`.

Indentation method at end of line

This field determines what happens to the indentation of the current line and the new line when you type the `Enter` key. The settings are:

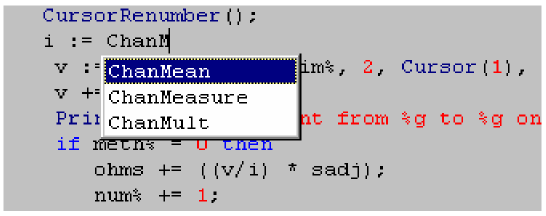
None	No automatic formatting is applied. The new line is not indented.

Maintain	The new line is indented to match the indentation of the current line.
Automatic	The indentation of both the current line and the new line is adjusted based on the Auto Format settings.

Toggle Comments

This Edit menu command is available in script and text sequencer views, and is also available in the context menu when there is a selection. It adds or removes a comment marker at the start of the line for all lines in the current selection. It decides what to do based upon the first character of the first line in the selection.

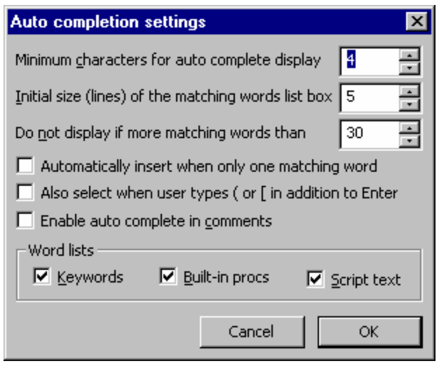
Auto Complete



```
CursorRenumber();
i := ChanM
 v :=ChanMean      im%, 2, Cursor(1),
 v +=ChanMeasure
 PriiChanMult      nt from %g to %g on
 if meth% = 0 then
    ohms += ((v/i) * sadj);
    num% += 1;
```

In any script, you will find that you are typing the same text items repeatedly. The editor can save you some time by popping up lists of known words that match your typing. The matching is done by looking for a word break in the text before the text caret, then matching your typing against various categories of words known to the script. Matched words are displayed in alphabetical order and the current word is highlighted.

You can use the up and down arrow keys to choose an item in the list and the `Enter` key to select an item or double click an item to enter it. The `Esc` key cancels the list. Alternatively, you can just keep on typing, which will narrow the choice of words to match. The Edit menu Auto Complete Setup dialog gives you some control over the words that are matched and when the auto-completion lists appears.



Auto completion settings

Minimum characters for auto complete display: 4

Initial size (lines) of the matching words list box: 5

Do not display if more matching words than: 30

☐ Automatically insert when only one matching word

☐ Also select when user types (or [in addition to Enter

☐ Enable auto complete in comments

Word lists: ☑ Keywords ☑ Built-in procs ☑ Script text

The Word lists section of the dialog lets you choose the categories of words to match your typing against. If you do not want to display auto-completion lists, clear all the categories.

You can choose from script keywords (`Proc`, `Func`, `EndCase`, ...), built in functions and procedures (`SampleSequencer`, `NextTime`, ...), and words that already exist in the script. You may wish to disable the Script text option if you are working with a huge file and you notice an appreciable delay after typing before the list appears.

Minimum characters for auto complete display	This sets the number of characters in a name that must be typed before the list will appear. You can set this to 1 to 12 characters. Setting a low value can cause the pop-up display to become annoying. You need to choose a count that gives you enough help, but not too much. Try a value of 3 to start with.
Initial size (lines) of the matching words list box	This field sets the maximum number of words to display at a time in the range 1 to 40. If there are more matching words, the list contains a scroll bar. After the list appears, you can re-size it by clicking and dragging on the horizontal edge furthest away from the matched text.
Do not display if more matching words than	If there are more matching words than you set here, the list is not displayed. You can set this value in the range 1 to 200 words.

Automatically insert when only one matching word

Because of the design of the script language, apart from variable declarations, all words you type in are likely to have been defined already. If you set this option, once your typing has reduced the list size to 1, the word is automatically inserted. You may want to increase the Minimum characters for auto complete display if you enable this option.

Also select when user types (or [

Normally, the list text is inserted when you type the `Enter` key or double-click the list text. If you check this option, the text is also inserted when you type an opening round or square bracket, followed by the bracket.

Enable auto complete in comments

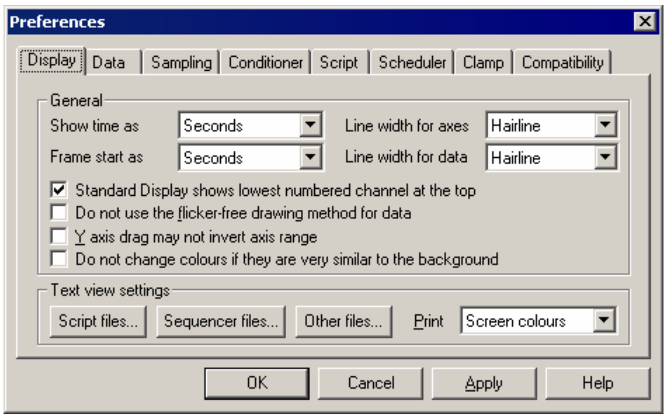
Normally, automatic word completion is disabled in a comment. However, if refer to script functions in your code documentation you may find it useful to check this box.

Preferences

The Edit menu Preferences dialog has separate tabs for display options, data options, sampling options, script options, signal conditioner setup, time scheduling and clamping options. The preferences are stored in the Windows registry and are user specific. If you have several different logons set for your computer, each logon will have its own preferences. It is possible to change the preferences from a script; see the `Profile()` command for details.

Display

This tab contains options relating to the way data is shown on the screen.



Show time as: Seconds
Frame start as: Seconds
Line width for axes: Hairline
Line width for data: Hairline
[✓] Standard Display shows lowest numbered channel at the top
[ ] Do not use the flicker-free drawing method for data
[ ] Y axis drag may not invert axis range
[ ] Do not change colours if they are very similar to the background
Print: Screen colours

Show time as

The Show time as selector controls the time units used within Signal, you can select Seconds, Milliseconds or Microseconds. The selected units will be used in the sampling configuration, for all data files with a time-based X axis including cursors and cursor windows and for all appropriate dialogs including the various process settings dialogs. The main area of Signal not affected by this setting is the script language, which will always see and use values in seconds, though script programs can read the current settings and adjust their behaviour as required. This setting does not affect any data stored by Signal, just the way time is displayed to the user, so you can switch settings without causing problems with data collected using another setting. Some time values are saved by Signal as strings, particularly the parameters for memory view and XY view processing online and active cursors, and these may be misinterpreted after the time units are changed.

Frame start as

The frame start time shown in the status bar and in printouts may be set to be the time since sampling started in seconds; the same value shown as hours, minutes and seconds or the absolute time of day also shown as hours minutes as seconds.

Line widths

The two Line width items set the line width in points for drawing data and axes respectively. These are relatively unimportant for displays on the screen, where most reasonable settings will only select between lines one or two pixels wide and many different settings will produce the same display. The line width controls are particularly valuable for printing, where the lines drawn can get unsatisfactorily fine. At CED we find that values of 0.75 pt for data and 1.0 pt for axes look pleasant. The line widths also control various other drawing operations; for example the axis width controls borders drawn around views and the data width sets the basic size of drawn dots and XY view lines and symbols.

Channel order You can change the order of data and memory 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.

Do not use the flicker-free drawing method Version 3.04 implemented a new buffered drawing method that reduces screen flicker on updates. However, this may impact the display speed. Check the box for the old method.

Y axis drag may not invert axis range Set this checkbox to prevent a Y axis being inadvertently inverted by dragging it too far.

Do not change colours Signal attempts to avoid your data being invisible by overriding your colour choices if they are very similar to the view background colour. Set this checkbox if you want your selected colours to be used regardless.

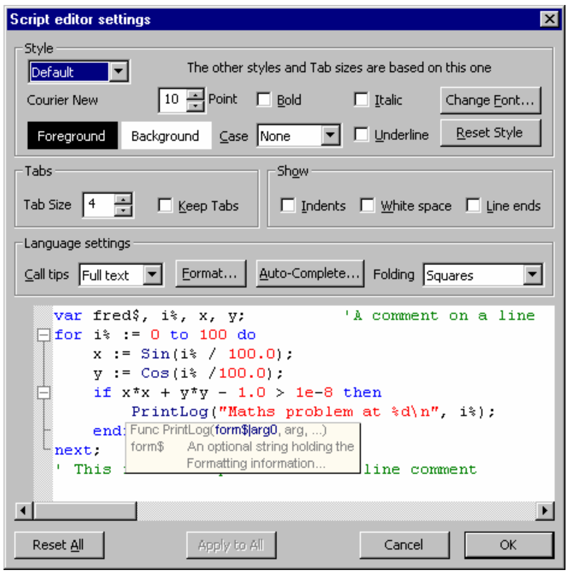
Text view settings This area of the dialog holds controls that let you configure the appearance of text-based views (script, text sequencer, log views and text views created by a script) on screen and when printing. The dialogs accessed by the Script, Sequencer and Other files buttons will apply any changes globally (to all open files of the selected type and to all future files). The same dialogs can be used from the View menu Font commands to make changes to the current view (there is then an extra button to apply changes to all views).

Print This sets how to print coloured text from a text view. The choices are:

Screen colours	Use the displayed screen colours. This is very wasteful of ink or toner if the background is not white.
Invert light	If you display your text as a light colour on a dark background, this setting prints on a white background and inverts the text colours.
Black on White	Prints black text on a white background.
Colour on White	Prints in screen colours on a white background.

Script files... Open the editor settings dialog from the Edit menu Preferences General tab to change settings for all views of a given type, or from the View menu Font command to change the current view only. When used for the current view, an extra button Apply to All appears at the bottom of the dialog to apply changes to all views. The Reset All button returns all the dialog values to standard settings.

The bottom half of the dialog holds example text suitable for the view type to show the effect of any changes.



```
var fred$, i%, x, y;          'A comment on a line
for i% := 0 to 100 do
    x := Sin(i% / 100.0);
    y := Cos(i% /100.0);
    if x*x + y*y - 1.0 > 1e-8 then
        PrintLog("Maths problem at %d\n", i%);
    end; Func PrintLog(form$|arg0, arg, ...)
next;    form$     An optional string holding the
' This             Formatting information...        line comment
```

Style Each view type supports one or more text styles. For each style you can set a font (including proportionally spaced fonts). The font includes the size in points (in the range 2-256) and bold and italic settings. You can also choose to force upper or lower case and underlines for all text in the style. You can set a foreground and background colour for the style by clicking on the Foreground and Background rectangles.

All views have the Default style that the remaining styles are based on; it is used for everything that is not covered by one of the other styles. The Tab size is based on the width of a space character in this style. If you change any aspect of this style, all the other styles that match that aspect will also change.

In a Script view, you can control the appearance of many different aspects of the display, based on the syntax of the language. There are settings for:

White space	Style used when drawing spaces and tabs and control characters.
Comment	The style used when displaying comments.
Numbers	The style to use when displaying numbers.
Keywords	The style to use for script keywords, such as `for`, `next` and `end`.
String	The style for literal strings, such as `"This is a string"`.
Function	Used for built-in script functions, such as `PrintLog()`.
Operator	The style for script language operators, such as `+`, `-` and `+=`.
Identifier	The style for function, procedure and variable names created in a script.
Call tips	The style for pop-up call tips. This style has an extra Tip highlight colour field, used to highlight the current argument in a function. This replaces the Bold, Italic, Case and Underline fields, which are not displayed.

The Reset Style button reverts the current style to standard settings.

Tabs This dialog region controls the size of tabs (set in units of the width of a space character in the Default style) and if Tabs are implemented by saving a Tab character in the text (check Keep Tabs) or are implemented as multiple spaces (clear the Keep Tabs checkbox). If you use a fixed pitch font, such as `Courier New`, then it does not matter too much if you choose to keep Tab characters or not. If you use a proportional font for anything except comments, it is better to keep the Tab characters. When automatically formatting a script, the Keep Tabs setting determines if indents are generated with Tab characters or spaces.

Show In addition to displaying the text, you can also choose to display information about the white space in your file. The settings are:

Indents	It can be useful when manually lining up indented loops in a script to see the indent level. Check this box to display vertical lines at each tap stop in the leading white space of each line.
White space	Check to display spaces with a centred dot and tabs as right arrows.
Line ends	Check this box to see the Carriage Return (CR) and Line Feed (LF) characters that mark the end of a line. This can be helpful to understand what is happening when working with scripts that move the caret around.

Language settings This area of the dialog is used by script and output sequencer views. It sets your preferences for call tips, automatic formatting, automatic text completion and folding. The Format and Auto Complete buttons duplicate Edit menu commands and are described separately.

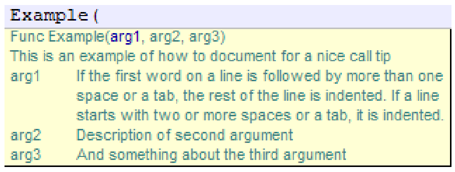
Call tips A Call tip is a block of text that pops up in a script view when you type the opening brace of a function of procedure name. The call tip text contains a synopsis of the command and a list of the command arguments, with the current argument highlighted.

You can choose from None (no call tips), Single line (just the command name and arguments) and Full text (includes a command synopsis).

For built-in functions, the call tip text holds the command name and arguments plus one sentence of description. For user-defined `Proc` and `Func` items, the arguments are taken from the line containing the `Proc` or `Func` keyword (expected to be the first item on the line). If the line before the `Proc` or `Func` is a comment, up to 10 lines of comment are used as a description. If the line before is blank, and the line after is a comment, up to 10 lines following are used as a description. If you document your functions and procedures as in this example, they will generate tidy call tips (and fold away to a single line):

```
Func Example(arg1, arg2, arg3)
'This is an example of how to document for a nice call tip
'arg1  If the first word on a line is followed by more than one
'      space or a tab, the rest of the line is indented. If a line
'      starts with two or more spaces or a tab, it is indented.
'arg2  Description of second argument
'arg3  And something about the third argument
var x,y;  'the start of the code
...
```

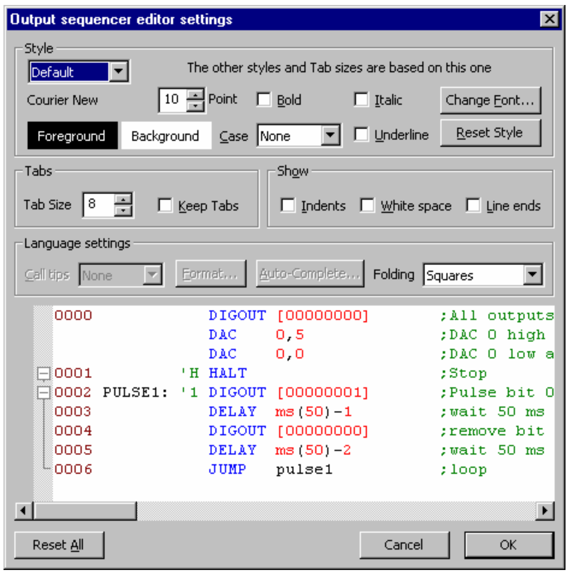
The example text above would produce a call tip looking like this. Comment markers at the start of each line are removed and multiple spaces after the first word or at the start of each line are replaced with Tab characters.



```
Example(
Func Example(arg1, arg2, arg3)
This is an example of how to document for a nice call tip
arg1	If the first word on a line is followed by more than one
	space or a tab, the rest of the line is indented. If a line
	starts with two or more spaces or a tab, it is indented.
arg2	Description of second argument
arg3	And something about the third argument
```

Folding Script views and output sequence views can display a folding margin, and allow you to fold the code by clicking in the margin, from the View menu Folding command, and by right clicking in the view and selecting Expand All Folds, Collapse All Folds or Toggle all folds (which finds the first fold point in the document, reads its folded or unfolded state and then sets all folds to the opposite state). In a script view, fold points are based on a lexical analysis of the script text. You can choose from one of four folding styles, or have no folding margin.

Sequencer files... The editor settings dialog for output sequencer files is very similar to that for script files. The example text in the lower half of the dialog displays some typical sequencer code and there is no support for Call tips, auto formatting or automatic completion.



Output sequencer editor settings

Style: Default — The other styles and Tab sizes are based on this one
Courier New, 10 Point, Bold [ ], Italic [ ], Underline [ ], Case: None
Tabs: Tab Size 8, Keep Tabs [ ]
Show: Indents [ ], White space [ ], Line ends [ ]
Language settings: Call tips None, Folding Squares

```
0000                DIGOUT [00000000]          ;All outputs
                    DAC    0,5                 ;DAC 0 high
                    DAC    0,0                 ;DAC 0 low a
0001             'H HALT                       ;Stop
0002 PULSE1:     '1 DIGOUT [00000001]          ;Pulse bit 0
0003                DELAY  ms(50)-1            ;wait 50 ms
0004                DIGOUT [00000000]          ;remove bit
0005                DELAY  ms(50)-2            ;wait 50 ms
0006                JUMP   pulse1              ;loop
```

There is a list of styles that you can apply to the different elements of the sequencer text. The Default style is used in exactly the same way as for script views as the basis for all other styles and as the font that is measured to set the Tab size. The remaining styles are:

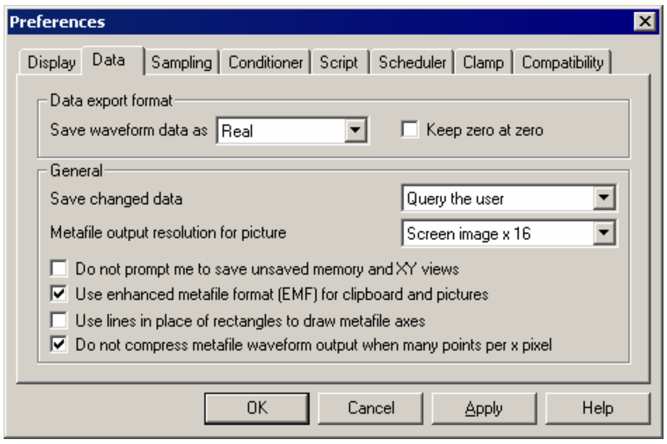
White space	Style used when drawing spaces and tabs and control characters.
Comment	The style used when displaying comments.
Numbers	The style to use when displaying numbers and digital i/o expressions such as `[...0101]`.

Keywords	The style to use for standard output sequencer instructions.
Deprecated	The style to use for outmoded output sequencer instructions like `DAC0` that have been replaced and may not be supported in future revisions.
Display	Text introduced by > for display on screen during sampling.
Directives	Items such as `VAR` that do not generate output instructions.
Operator	The style for mathematical operators like +, -, * and /.
Identifier	The style for user-defined variable names and labels.
Step	The style for the optional step numbers at the start of an instruction line.
Key	The style for keyboard links introduced by a single quote, for example `'H`
Functions	The style for built-in conversion functions like `ms()`.

Folding support The output sequencer editor supports code folding based on the keyboard link entry points. That is, you can fold up all code from a line holding a keyboard link up to the next line holding a link.

Other files... The editor settings dialog for all views except script and sequencer views is the same as the dialog for script views, except that there are no language settings and there is only the Default text style.

Data The Data tab controls the way data is stored and exported.


Preferences

Display | Data | Sampling | Conditioner | Script | Scheduler | Clamp | Compatibility

Data export format
Save waveform data as: Real
☐ Keep zero at zero

General
Save changed data: Query the user
Metafile output resolution for picture: Screen image x 16
☐ Do not prompt me to save unsaved memory and XY views
☑ Use enhanced metafile format (EMF) for clipboard and pictures
☐ Use lines in place of rectangles to draw metafile axes
☑ Do not compress metafile waveform output when many points per x pixel

OK | Cancel | Apply | Help

Data export format The data export format items control decisions on how waveform data is written to CFS files when no other information is available to help Signal make this decision. The Save waveform data as field sets the preferred format for waveform data on disk. Waveform data in CFS files can be stored either as floating point numbers or as scaled integers. Scaled integers are the format of data sampled by the 1401, occupy less space on disk and are the only format that can be read by the DOS SIGAVG software, but can be less accurate and can overflow (when a data value outside the possible integer range is required). Floating point numbers are more bulky on disk, but are more accurate and cannot overflow. In most circumstances, the format used for waveform data is derived from the data source or set by the destination file, but when creating a new CFS data file by other mechanisms (saving a some types of memory view in particular), the format used is controlled by this field. Select Real for maximum data accuracy, Integer to produce smaller data files or for SIGAVG compatibility.

Keep zero at zero This item controls how scaling factors for integer data are calculated. Signal tries to use the same integer scaling factors as the previous values for the frame (or the previous frame where appropriate), but may need to recalculate the factors if the data values become too large for the existing scaling or too small to represent accurately. If this checkbox is set, Signal will always calculate scaling factors that keep zero in the integer data corresponding to zero in the scaled values, which can be convenient in some circumstances.

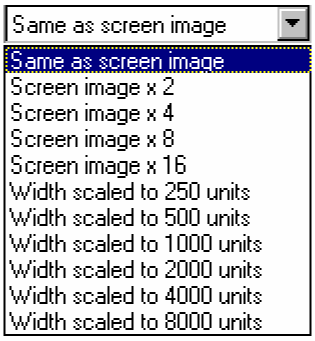
Default data update

The Save changed data selector sets the default action for file data that has been changed in memory. For example, the script language could have been used to differentiate the data in a channel. You can choose to write changes back always, to only write changes after querying the user or to always discard changed data. This option sets the default initial state for all data files opened, use the File menu Data update mode command to control data write-back for a single file.

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, Signal 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.

Metafile output resolution


Same as screen image
Same as screen image
Screen image x 2
Screen image x 4
Screen image x 8
Screen image x 16
Width scaled to 250 units
Width scaled to 500 units
Width scaled to 1000 units
Width scaled to 2000 units
Width scaled to 4000 units
Width scaled to 8000 units

Signal saves file, memory or XY views as pictures in either bitmap (screen image) or metafile format. A metafile describes a picture in terms of lines and text based on a grid of points. Metafiles have the advantage that they can be scaled without losing resolution.

You can choose the density of the grid. The higher the density, the more detail in the picture. The problem for time and result views with a lot of data points is that the higher the grid density, the more lines need to be drawn, and many drawing programs have limits on the number of lines they can cope with.

You can set a grid based on the screen resolution, or a grid such that the width of the image is a fixed number of points. If you are not sure what setting to use, start with *Same as screen image* and adjust it as seems appropriate for your use.

Use enhanced Metafile format

Signal 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 2007).

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.

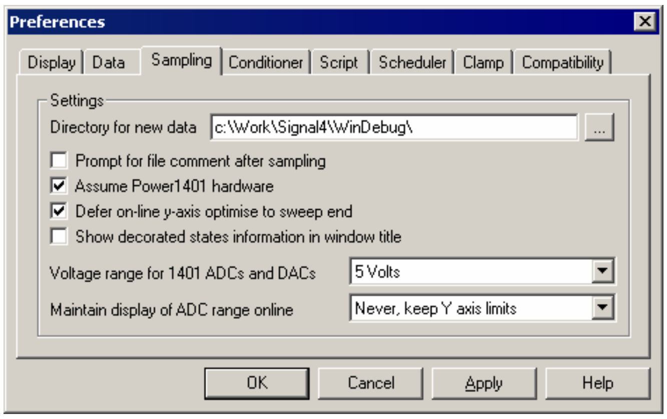
Do not compress metafile waveform output ...

When generating metafiles (and also when drawing to the screen, though that is not affected by this option) Signal can save space and time by drawing waveforms with a large number of points per pixel as a series of vertical lines, one pixel apart. This does not result in drawing errors because there are already many waveform points drawn on top of each other.

When drawing to a metafile this can cause problems, as there is not the same match between the vertical line width and vertical line spacing. If you check this box, no such compression is done when you generate a metafile. For the most precise metafile data, set the Metafile output resolution to screen*16 and check this box.

Sampling The Sampling tab contains preferences for the sampling of data.

Directory for new data When Signal samples data, the new data is stored in a temporary file while it is being collected and only stored in a final CFS file when the file is saved. The Directory for new data field sets the directory in which Signal puts this temporary file. If this field is blank, Signal will use the current directory (which may not be where you expect), so it is a good idea to set one. If you want to save over a network, for example, or store new data files on media that is slow to write, you can use this field to ensure that the temporary file directory (which is where all the real-time writing occurs) is on a fast hard disk.


Preferences

Settings

Directory for new data: c:\Work\Signal4\WinDebug\

- [ ] Prompt for file comment after sampling
- [x] Assume Power1401 hardware
- [x] Defer on-line y-axis optimise to sweep end
- [ ] Show decorated states information in window title

Voltage range for 1401 ADCs and DACs: 5 Volts

Maintain display of ADC range online: Never, keep Y axis limits

When you save a new data file, Signal 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 temporary file, Signal just renames the file. If the volume is not the same, Signal copies the file, then deletes the original.

Prompt for file comment The Prompt for file comment after sampling item is used to encourage entry of a file comment when a new data document has been created by automatically popping-up the file comment entry dialog.

Assume Power1401 hardware In order to correctly show the sampling rates attainable and limits to the pulse output resolution, Signal needs to know if Power1401 hardware is in use. Signal tries to detect the 1401 type automatically when it starts up, this checkbox sets whether Signal will assume the presence of a Power1401 if it cannot detect the 1401 type directly.

Defer on-line y-axis optimise to sweep end If a channel display is optimised in the y-direction and the Defer on-line y-axis optimise to sweep end box is checked, Signal will wait until the sweep finishes before scanning the data collected in order to perform the optimise. If this box is unchecked then the optimisation will be done only with the data collected so far in that sweep. This may mean no data has been collected and axes will be set to default limits which may well not be ideal.

Show decorated states information in window title When sampling using multiple states, Signal will display the frame state number (with optional state label) in the window title bar for frame zero. This is intended to make it easier to see what is going on during data acquisition. Signal can also display extra information showing the progress of the states sequencing or, if an auxiliary states device is in use, the auxiliary states device settings. This extra information is only shown if this checkbox is ticked, otherwise the state number and label only are shown.

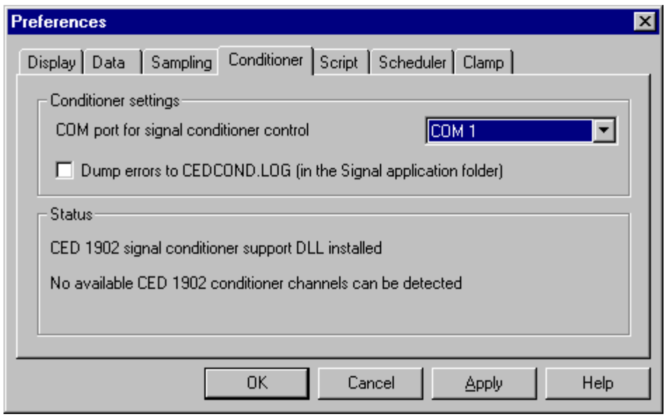
Voltage range for 1401 ADC and DACs Most 1401s have a ±5 Volt input range, but some are set up for ±10 Volts; more modern 1401 hardware can be switched between these voltage ranges using the Try1401 application. Signal detects the voltage range in use automatically in recent 1401s, but you must set it manually for the 1401plus. Choose from 5 Volt, 10 Volt and Last seen hardware. You will be warned if Signal detects a conflict between user settings and installed hardware. The voltage range setting affects scaling in the sampling configuration and DAC output values in the output sequencer. It has no effect on scale values in previously sampled data files. If a sampling configuration generated with the voltage range set to 5 volts is read into a copy of Signal set up for 10 volts or vice versa, the scalings in the sampling configuration are automatically adjusted to keep things correct.

Maintain display of ADC range online

When a programmable signal conditioner settings are changed or a telegraph voltage changes during sampling, the display y-range for the relevant channel can do one of three things. Never, keep the y-axis limits will keep the y-axis unchanged. In this case there may be no visible indication that the amplifier gain has changed although the resolution of the data may change. Maintain it if Showing All range will alter the y-axis when the gain changes to show the new full range of the ADC provided the full range was previously displayed. Maintain ADC range percentage will maintain the ADC range previously displayed but show the new values corresponding to the amplifier settings. In all cases you should note that, for example, data drawn at 3 mV would continue to be drawn at 3 mV. It is only the axis limits that may change.

Conditioner

This tab lets you set the port used by a signal conditioner (see the *Programmable signal conditioners* chapter). Check the *Dump errors…* box to write diagnostic messages to `CEDCOND.LOG` in the Signal application folder. If you wish to change the type of conditioner you must run the Signal install program; you will be offered a list of conditioners to choose from.


Preferences

Conditioner settings

COM port for signal conditioner control: COM 1

☐ Dump errors to CEDCOND.LOG (in the Signal application folder)

Status

CED 1902 signal conditioner support DLL installed

No available CED 1902 conditioner channels can be detected

If the currently installed signal conditioner uses a serial line port, you can select any port in the range COM 1 to COM 8. Each time you change the port, Signal searches for any conditioner attached to it and will update the status panel.

The Status panel gives information on the type of signal conditioner support that is installed. If it can detect a signal conditioner, it also holds information about the channels that have conditioner support.

Script

This tab items regarding the script editor and script behaviour.

Save modified scripts and sequences before they run

Save modified scripts and sequences before they run can be checked so that any changes to a script will be automatically committed to disk each time the script is run. This checkbox also affects automatic saving of modified sequence text before sampling.

Enter the debugger

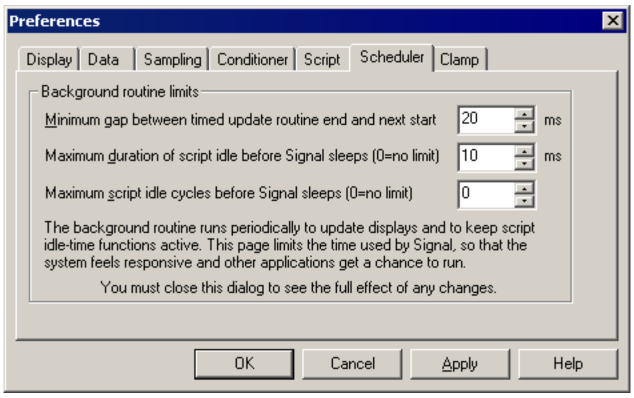
If Enter the debugger when a script has a runtime error is set then if a runtime error in a script happens then the script view will be displayed with the debugger running. This allows the user to check on the values stored in the variables as well as looking at the call stack.

Maximum lines in Log window

You can use this field to prevent huge amounts of text accumulating in the Log view. If there are more lines of text in the Log view than set here, and a script writes more, the oldest lines are deleted to leave room. Set to 0 for no limit.

Scheduler

This tab limits the processor time consumed by the Signal user thread while sampling and idling with a script running. If Signal takes too much time, the system feels unresponsive. It does not affect the time-critical thread that writes sampled data to disk. You can read more about threads below. Signal runs a background routine when it has idle time and also periodically on a timer. The routine handles the following tasks:



Background routine limits

Minimum gap between timed update routine end and next start: 20 ms

Maximum duration of script idle before Signal sleeps (0=no limit): 10 ms

Maximum script idle cycles before Signal sleeps (0=no limit): 0

The background routine runs periodically to update displays and to keep script idle-time functions active. This page limits the time used by Signal, so that the system feels responsive and other applications get a chance to run.

You must close this dialog to see the full effect of any changes.

- When sampling, it gives windows a chance to detect that the maximum time has changed, which may cause windows to update and processing to occur. Any invalidated windows will update the next time Signal gets idle time.
- If a script is running, it gives any "idle function" in the script a chance to run.
- In automatic file naming mode it starts the next file running.

There are three fields that limit the time used in the background routine. They have no effect on the time used when Signal runs a script that does not idle (see the `Yield()` or `YieldSystem()` script commands for this). The standard values work for most cases.

Minimum gap between timed update routine end and next start

If the time interval set by this field passes without the background routine running, it is scheduled to run as soon as possible. You can use values in the range 1 to 200 milliseconds. The standard value is 20 milliseconds. The lower the value, the more time Signal will spend on background processing relative to other applications. This field also limits the time that Signal will sleep for (see the discussion of threads, below).

Maximum duration of script idle before Signal sleeps

When Signal gets idle time (see the discussion of threads, below), you can limit the time it uses before Signal goes to sleep. Signal uses idle time to run script idle routines, such as those created by `ToolbarSet(0,...)`. You can set from 0 to 200 milliseconds (0 means no time limit). The standard value is 10 milliseconds. The larger the value, the more the script idle routine runs at the expense of other applications.

Maximum script idle cycles before Signal sleeps

As an alternative to limiting the script idle routine by elapsed time, you can limit it by the number of times it is called. You can set from 0 to 65535 times (0 means no limit). The standard value is 0. Setting both this and the *Maximum duration...* field to 0 is unkind to other applications. Setting this to 1 is the most generous to other applications.

About threads

A *thread* is the basic unit of program execution; a thread performs a list of actions, one at a time, in order. To give you the impression that a system with one processor can run multiple tasks simultaneously, the system scheduler hands out time-slices of around 10 milliseconds to the highest priority thread capable of running. Tasks at the same priority level share time-slices on a round robin basis. Lower priority tasks rely on higher priority tasks "going to sleep" when they have nothing to do or when they are blocked (for example, waiting for a disk read). If this did not happen, low priority tasks would not run.

When Signal gets a chance to run, it processes pending messages such as button clicks, keyboard commands, mouse actions and timer events and then updates invalid screen areas. Finally, Signal is given idle time until it says it does not need any or new messages occur. If Signal needs no more idle time it sleeps until a new message appears in the input queue. The *Minimum gap...* timer wakes up Signal if nothing else happens.

Clamp

This tab is currently used only for enabling the clamping features in Signal. If you are not involved with patch/voltage clamping work then you may want to use this option to turn off and conceal clamping features to avoid confusion.

Compatibility

This tab contains options for reverting Signal behaviour to that of earlier versions. At the moment there is just one option here.

If showing frame zero, switch to frame 1 when sampling finishes

Signal changed in version 4.03 to showing the last sampled frame when sampling finishes – it is after all the most recent data. Checking this box will cause Signal to revert to showing frame 1.

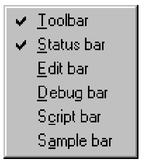
11 View menu

The view menu is used to control the appearance of the data and XY views, it contains commands to show and hide the application toolbar, edit bar and status bar, to move the display from one frame to another within the data file, to control frame overdrawing, for zooming in and out in both the x and y directions, controlling the channels and other items that are displayed on the screen and the drawing mode and the fonts used and the use of colour.

Toolbar
Edit bar
Status bar

These three commands enable and disable the display of the application toolbar, the edit toolbar and the status bar. The toolbar is the array of buttons normally displayed below the application menu bar. The edit bar contains buttons for functions available in text views. The status bar is always at the bottom of the application window and displays information about the current (highlighted) view. These items display a checkmark if the corresponding bar is displayed, click on the item to toggle the display state.

These bars can also be shown and hidden by right-clicking on an unused area of the application window to open a pop-up menu. This menu can show and hide other types of toolbar. The script and sample bars are described in the *Script menu* and *Sample menu* chapters. The debug bar is described in the script manual.



Next frame
Previous frame

These commands, together with the buttons at the bottom left of the data window and the shortcut keys `PgUp` and `PgDn`, change the current frame to the next or previous frames in the data document. If the current frame is the first or last frame in the document then the corresponding menu item and button are greyed out. The `Ctrl+PgUp` and `Ctrl+PgDn` shortcuts change to the last or first frame in the document. When sampling `Ctrl+PgUp` switches to always showing the last frame filed until the frame is manually changed.

Goto frame

This command (shortcut `Ctrl+G`) opens a dialog to allow you to enter a frame number to move to directly.

Show buffer

This command (shortcut `Ctrl+B`) toggles between showing the frame buffer and the current frame. When the buffer is shown, the menu item is shown checked, and the view title is modified to show that the buffer is visible. The frame buffer is a separate frame of data 'behind' each open CFS file which is provided by Signal. See the Analysis menu chapter for more details of the frame buffer.

Overdraw frames

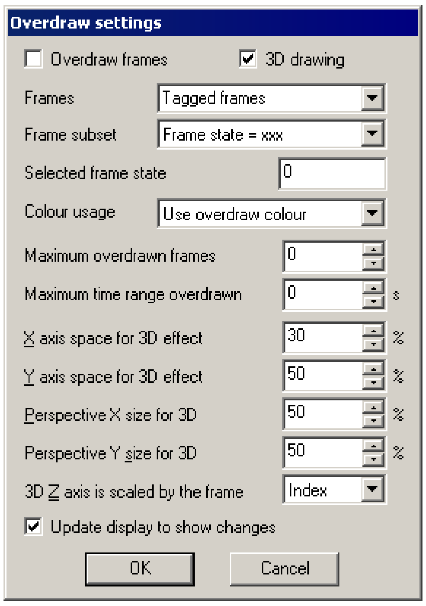
This command (shortcut `Ctrl+D`) switches the display between normal mode, displaying the current frame only, and overdraw mode, which also displays all of the frames in the frame display list, either behind one another or using a 3-dimensional representation. When overdraw mode is enabled, this menu item is shown checked.

The frame display list defines frames to display in addition to the current frame, it is set using the Overdraw settings option, below. These additional frames can also be drawn in a 3D style plot.

Add frame to list

This adds the currently displayed frame to the frame display list that will be displayed along with the current frame if frame overdrawing is enabled. If the current frame is in the display list, this command becomes Remove frame from list. Adding or removing a frame from the display list in this way will destroy any dynamic behaviour of the list. For example: if you have requested to overdraw all tagged frames and then add another frame using this menu item, the display list will then remain fixed even if you subsequently tag or un-tag a frame.

Overdraw settings


Overdraw settings

[ ] Overdraw frames [✓] 3D drawing

Frames	Tagged frames
Frame subset	Frame state = xxx
Selected frame state	0
Colour usage	Use overdraw colour
Maximum overdrawn frames	0
Maximum time range overdrawn	0 s
X axis space for 3D effect	30 %
Y axis space for 3D effect	50 %
Perspective X size for 3D	50 %
Perspective Y size for 3D	50 %
3D Z axis is scaled by the frame	Index

[✓] Update display to show changes

This command (shortcut `Ctrl+Shift+D`) opens a dialog that sets the frame display list and also controls the manner in which overdrawn frames are displayed. The upper part of the dialog is used to enable frame list overdrawing and to define the frames to be overdrawn, the lower part controls the optional 3D overdraw mode.

At the top of the dialog there are checkboxes to turn on frame overdrawing and to enable 3D overdrawing. There is another checkbox at the bottom of the dialog which causes immediate display updates as you change fields in the dialog; very useful for getting a display just right.

Frame selection

The next items select the data frames to be overdrawn. Set the frames either by entering a list of frame numbers directly or by using the drop-down list to select a category of frames from: All frames, Current frame, Buffer, Tagged frames, Un-tagged frames, Frame state = xxx and Last n frames. A subsidiary field below allows a subset of these frames to be defined.

If you select Frame state = xxx the Selected frame state field appears, use this to select the state number to be displayed. Similarly if you select Last n frames an extra field appears for the number of frames wanted, this selects the number of frames before the current frame when used offline, and the number of frames most recently filed to disk when used online.

The frame list is dynamic so that, for example, if you request all tagged frames and subsequently tag a frame it will be added to the overdrawn frames.

Colour usage

This item selects the colours used for overdrawn frames. You can choose between the following options:

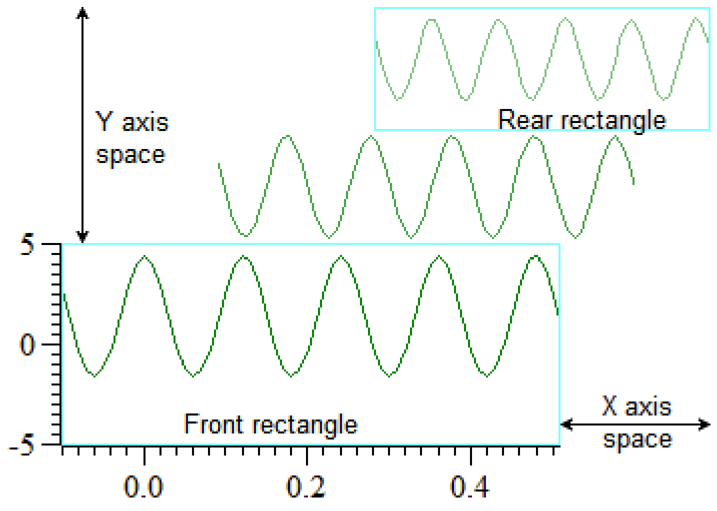
Use overdraw colour	Draw all overdrawn (non-current) frames using the Overdraw colour defined in the colour settings.
Use colour cycling	Draw overdrawn frames using an internal table of 18 colours. Each frame is drawn using the next colour in the table.
Draw at half intensity	Draw overdrawn frames using a colour halfway between the channel trace colour and the channel background.
No colour variation	Draw overdrawn frames using the same colour as the current frame – the standard trace colour.
Fade to background	Draw overdrawn frames in a colour that fades from the trace colour to the channel background colour.

Fade to overdraw colour	Draw overdrawn frames in a colour that fades from the trace colour to the defined overdraw colour.
Fade to secondary colour	Draw overdrawn frames in a colour that fades from the trace colour to the channel secondary colour.

Frame limits The next two items, labelled Maximum overdrawn frames and Maximum time range overdrawn, can be used to set limits to prevent overdrawing from taking too long. Leave these fields set to zero if you do not want to apply such limits.

3-D overdrawing

If you enable 3D overdrawing using the checkbox at the top of the dialog, data is drawn as a 3-dimensional ‘stack’ of frames, with the current frame at the front and older (lower-numbered) frames behind (you cannot show frames after the current frame – we found it made things impossibly confusing). To create the 3D effect, each frame is drawn inside a rectangle, and the position and size of the rectangle depends on a notional z axis. The z axis can be based on the frame’s position in the frame display list, the frame number or the trigger time of the frame.



3D drawing rectangles

In this example, there are three frames of data. The *Front* rectangle, used for the current frame, is always positioned at the bottom left of the channel area. The *Rear* rectangle, used for the oldest (lowest-numbered), is always positioned at the top right of the channel area. The middle frame also has a rectangle that is calculated by a linear interpolation between the front and rear rectangles based on the z axis position of the frame.

The dialog fields concerned with 3-D overdrawing control the positioning of the front and rear rectangles:

X axis space for 3D effect This field sets the percentage of the width of the channel area to use for the 3D effect. The larger the value, the smaller the width of the front rectangle. Set this and the Perspective X size fields to 0 to make all frames draw vertically aligned.

Y axis space for 3D effect This field sets the percentage of the entire view vertical space to share between all the channels to generate the 3D effect. All channels are given exactly the same space so that the 3D effect is the same for all channels. The larger the value, the smaller the height of the front rectangle.

Perspective X size This sets the width of the rear rectangle as a percentage of the width of the front rectangle in the range 0 to 100. Setting a value less than 100 gives a perspective effect.

Perspective Y size This sets the height of the rear rectangle as a percentage of the height of the front rectangle in the range 0 to 100. Setting a value less than 100 gives a perspective effect.

Z axis is scaled by the frame The position of each frame of data (between the *Front* rectangle and the *Rear* rectangle) can be set by the frame index within the display list (giving equally spaced frames), the frame number within the file, or by the frame trigger time (as shown for the current frame in the status bar). Choose from Index, Number or Time.

Notes The 3D display mode behaves in exactly the same way as ordinary overdrawing except that the overdrawn frames are offset and scaled. There is one other difference; only frames from earlier in the file than the current frame will be overdrawn, ensuring that the current frame is at the front and the oldest overdrawn frame is at the back.

The 3D display mode makes no difference to any measurements you may make with cursors, the X and Y axes that are displayed match the current frame that is displayed at the front.

Overdrawing online When 3D overdrawing data while it is being sampled, the special frame zero behaves as if it is after all other frames in the file – where it will be if it is written to disk.

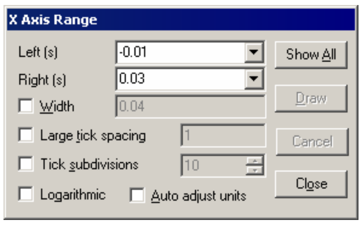
The All sampled frames option is only available overdrawing online; it enables special behaviour where frame data is never erased so that the display accumulates sampled traces. However if the display redraws for any reason then all the overdrawn data is lost. The Edit menu Clear option can be used to erase the data. The Last n frames option is renamed Last n frames filed when used online. Options that will cause a lot of frame overdrawing (such as All frames) should be avoided online unless you use the frame limits; the time taken to redraw many frames may impact on sampling performance.

Enlarge view
Reduce view

These commands, the two buttons at the lower left of data windows and the keyboard shortcuts `Ctrl+Right` and `Ctrl+Left` expand and contract the displayed x axis area. The enlarge command zooms out by doubling the data region spanned by the x axis. The reduce command zooms in by halving the data region. The left hand edge of the screen is fixed unless the expand operation would display data beyond the end of the frame, in which case the start of the displayed area is moved backwards. If the result of expanding would display more data than exists in the frame, all of the frame data is displayed.

X Axis range

This menu command, double-clicking the x axis of a data or XY view and the shortcut key `Ctrl+X` open the x range dialog, which sets the region of the view to display. The dialog also gives you the option of setting the x axis tick spacing.


X Axis Range

Left (s) -0.01
Right (s) 0.03
☐ Width 0.04
☐ Large tick spacing 1
☐ Tick subdivisions 10
☐ Logarithmic ☐ Auto adjust units

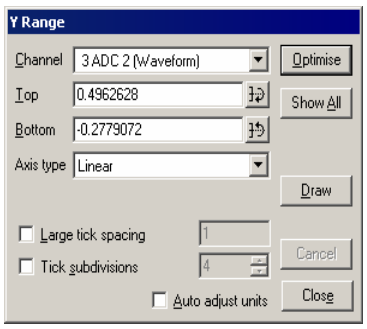
The Left and Right fields set the window start and end. You can type in new positions or select values from a drop down list. Each drop down list contains the initial field value, cursor positions, the minimum and maximum allowed values and the left and right edges of the window (XLow() and XHigh()). The Width field sets the window width if the box is checked. Click the Draw button to apply changes without closing the dialog. Show All expands the x axis to display all the data and closes the dialog. Cancel undoes changes made with the dialog and closes it. Close accepts any changes and closes the dialog.

Normally, you let Signal organise the x axis style. However, when preparing data for publication you may wish to set the axis tick spacing. 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 to these fields take effect immediately; there is no need to use the Draw button.

The Auto adjust units option will cause the units displayed on the axis to switch to multiples of powers of 10 in order to keep the figures sensible when zoomed well in or well out. This option affects only the axis; the units used by the cursors etc will still be the same. Checking the Logarithmic option will switch the axis from linear to logarithmic; modifying the displayed range only if it included negative values. In logarithmic mode another check box: Show powers will appear. This allows the big ticks to be labelled with powers of the big tick spacing. As with the tick spacing options these changes take place immediately with no need to press Draw.

Y Axis Range

This command, double-clicking a y axis or using the `Ctrl+Y` shortcut open the Y Range dialog. This sets the y axis range and style for data or XY view channels. The Channel field chooses one, all or selected channels. If more than one channel is selected, the settings shown are from the first channel.



Channel: 3 ADC 2 (Waveform)
Top: 0.4962628
Bottom: -0.2779072
Axis type: Linear
☐ Large tick spacing: 1
☐ Tick subdivisions: 4
☐ Auto adjust units

Optimise draws the visible data scaled to fill the display. Show All sets the y axis to display the maximum possible range for waveform channels, for channels without a range limit the available data range is shown. Both buttons close the dialog. To optimise without opening the dialog, right-click a channel and select the optimise option from the context menu.

The Top and Bottom fields are used to directly enter the limits to the displayed data. The buttons to the right of these fields are used to set up a symmetrical axis - clicking on one copies the current setting, negated, into the other field.

Lock axes and Group offset are visible when the current channel shares a y axis with other channels. They are enabled when the channel is the first in the group. If you check Lock axes, the grouped channels not only share the same space, they also share the y axis of the first channel in the group. The Group offset field sets a per-channel vertical display offset for each locked channel to space out channels with the same mean level.

Draw applies changes to the Top, Bottom, Lock axes and Group offset fields. The axis type can be selected to be Linear, Logarithmic or Square root. Cancel undoes any changes and closes the dialog. Close accepts changes and closes the dialog. The remaining options are identical to those already described under X Axis Range (above).

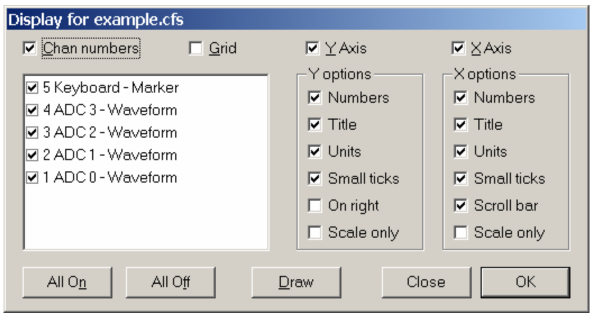
Standard Display

This command sets the current data, memory, XY or text view to a standard state. In file and memory views it turns on all channels in a standard display mode and size, ordered as set in the Edit menu Preferences, all special channel colours are reset, x and y axes turned on and all special axis modes and grids off. 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.

In a text-based view it removes any maximum line limit except in the Log view, where it applies the line limit set in the Edit menu Preferences option. It also hides line numbers, displays the gutter and the folding margin (for views that support folding). All the text styles are set to the default values (equivalent to opening the Font dialog and using Reset All) and any zooming is removed.

Customise display

This command opens a dialog that controls the channels to display in a data or XY view, the display of x and y axes, grids and of the horizontal scroll bar.



Display for example.cfs

☑ Chan numbers ☐ Grid ☑ Y Axis ☑ X Axis

☑ 5 Keyboard - Marker
☑ 4 ADC 3 - Waveform
☑ 3 ADC 2 - Waveform
☑ 2 ADC 1 - Waveform
☑ 1 ADC 0 - Waveform

Y options: ☑ Numbers ☑ Title ☑ Units ☑ Small ticks ☐ On right ☐ Scale only

X options: ☑ Numbers ☑ Title ☑ Units ☑ Small ticks ☑ Scroll bar ☐ Scale only

Check the Chan numbers box to show channel numbers in file and memory views. All On and All Off select all or none of the channels to be shown, or you can show and hide individual channels using the checkboxes. Draw updates the data window. You also have control over the x and y axes. 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 Signal choose a suitable size for you.

Channel Information

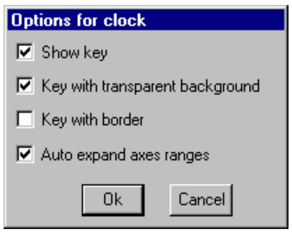
Use this dialog to view and edit data view channel information. You can open it by double-clicking a channel title, from the View menu or by right-clicking the channel to open the context menu. You can edit the Title of the channel set by the Channel field. The remaining fields are hidden or displayed depending on the channel type. For waveform channels the Units may be changed. The Full scale and Zero values may also be edited. These are the same as the values set in the sampling configuration when the file was sampled originally and can be changed to re-calibrate the data. For an idealised trace channel the Units may also be changed as well as the –3 dB frequency used for drawing the convolution and fitting the trace to the raw data. 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.

File Information

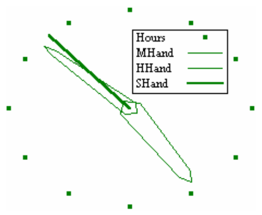
This command (shortcut `Ctrl+I`) displays information about the current data document. Currently, it displays the number of sweeps that have been added into a waveform average; the number of data blocks in a power spectrum or the number of events from an idealised trace added. For an amplitude histogram the number of points included is displayed. Equivalent information is also shown for the frame buffer.

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 displays the channel name and draws the line and point style for the channel. The 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()`.


Options for clock

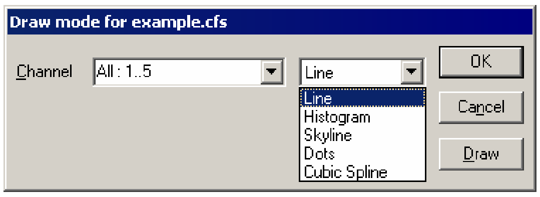
- [✓] Show key
- [✓] Key with transparent background
- [ ] Key with border
- [✓] Auto expand axes ranges



This example (made by the `clock.sgs` 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. Double-click the key to open the Options dialog.

Draw mode

The Draw mode dialog sets the display mode for channels. You can set the mode for a single channel, all channels or any subset of the channels.


Draw mode for example.cfs

Channel: All : 1..5

Line

The Channel field sets the channels to change. The next field sets the draw mode to use. Click Draw to change the draw mode for the selected channels without closing the dialog, click OK to change the draw mode and close the dialog.

Waveform draw modes

There are five drawing modes available for waveform channels: Line, Histogram, Skyline, Dots and Cubic Spline. Line joins the data points with straight lines. Skyline joins points with horizontal and vertical lines. Cubic Spline joins the points with smooth curves based on the assumption that the first and second derivatives of the data are continuous at the data points. However, you must always remember that the only data values that you can rely on are those at the sample points. Cubic Spline mode becomes Line mode in Windows metafile output.

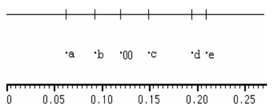
If your view has associated error information, for example a waveform average with error bars enabled, you can set the error display mode as: None, 1 SEM, 2 SEM or SD. Error bars only have meaning if the data points that contribute to the average have a normal distribution about the mean. Given this, 1 SEM shows ±1 standard error of the mean, 2 SEM is ±2 standard errors of the mean and SD is ± 1 standard deviation.

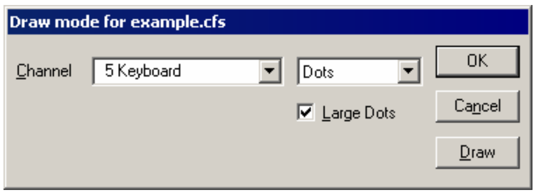
If each point of your data can be modelled as a constant "real" value to which is added normally distributed noise with zero mean, then you would expect the measured mean value to lie within 1 standard error of the mean (SEM) 68% of the time, or within 2 SEM 95% of the time. The standard deviation represents the width of the normal distribution of the underlying data at each data point.

Marker draw modes

There are three drawing modes available for marker channels; Dots, Lines and Rate.

Dots and Lines mode



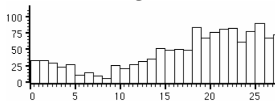

Draw mode for example.cfs

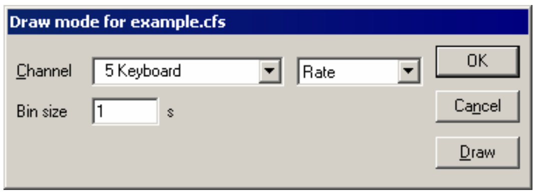
Channel: 5 Keyboard | Dots

☑ Large Dots

The simplest method is to draw the marker channel as dots. You can choose large or small dots (small dots can be very difficult to see on some displays). You can also select Lines in place of Dots. The picture above left shows the result of both types of display. If you select lines mode the display of marker values is suppressed.

Rate histogram mode



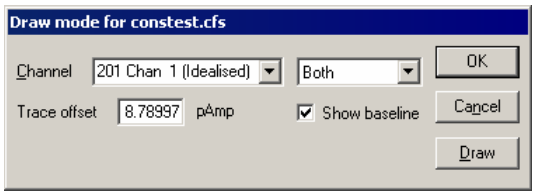

Draw mode for example.cfs

Channel: 5 Keyboard | Rate

Bin size: 1 s

The rate display mode counts how many markers fall in each time period and displays the result as a histogram. The result is not divided by the width of the bin, so strictly speaking it is a count histogram, not a rate. This form of display is especially useful when the marker rate before an operation is to be compared with the rate afterwards.

Idealised trace draw modes


Draw mode for constest.cfs

Channel: 201 Chan 1 (Idealised) | Both

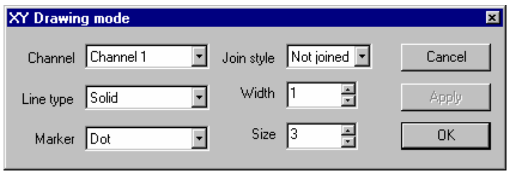
Trace offset: 8.78997 pAmp

☑ Show baseline

There are three drawing modes for idealised traces: Basic, Convolution and Both. The Basic mode draws the trace as horizontal lines representing event amplitudes separated by vertical lines for the transitions. Closed states are drawn in a different colour. There is an optional offset to allow the trace to be drawn below or above the raw data when displayed with a shared locked axis. The Convolution is a continuous curve formed by the convolution of the Basic idealised trace and the step response function. The step response function is the error function erf() defined as the integral of the Gaussian function. This is because it is assumed that the raw data was filtered using a close approximation to a Gaussian filter. The cut-off frequency for this filter can be changed by using the channel information dialog. For an idealised trace generated using threshold crossing the cut-off frequency will be set to the Nyquist frequency by default. There is also an optional baseline indicator which will show the level of the baseline for a selected event. Both shows both the Basic and Convolution at the same time. It is possible to "break" the drawing of an idealised trace if it is taking too long; hit `Ctrl+Break` and the drawing will be abandoned.

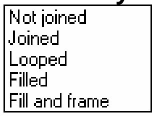
XY Draw Mode

This command is used in XY windows to set the drawing style of the XY data channels. Click OK to make changes and close the dialog, click Apply to make changes without closing the dialog. Cancel closes the window and ignores any changes made since the last Apply. The Channel field sets the channel to edit. If you change the channel, the dialog remembers any changes you have made so there is no need to use the Apply button before changing channel unless you want to see the change immediately. The settings available are:


XY Drawing mode

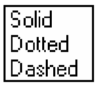
Channel: Channel 1
Join style: Not joined
Line type: Solid
Width: 1
Marker: Dot
Size: 3

Join style


Not joined
Joined
Looped
Filled
Fill and frame

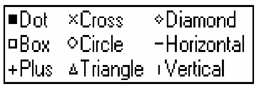
You can set five join styles for a channel. In Not joined style, no lines are drawn between data points. In Joined style, each point is linked to the next by a straight line. In Looped style, the points are joined and the final point is linked back to the first point. In Filled style, the data points are not joined, but they are filled with the XY channel fill colour. In Fill and frame style, the first and last points are joined and linked by a line and the channel is filled. The Line type and Width fields set the kind of line that joins the points.

Line type and Width

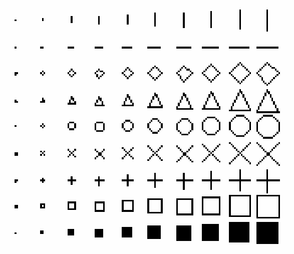

Solid
Dotted
Dashed

These two fields set the type of line used to join data points. The Width field determines how wide the line is, in units of half the data line width set in the Signal Preferences dialog. If you set a Width of other than 1, the Line type field is ignored and a solid line is drawn.

Marker and Size


■Dot ×Cross ◇Diamond
□Box ○Circle −Horizontal
+Plus △Triangle ı Vertical

The Size field sets how far, in points, the markers extend around their screen position. A size of 0 makes the markers invisible. There is a wide range of marker styles to choose from, including boxes, circles, triangles and vertical and horizontal bars.

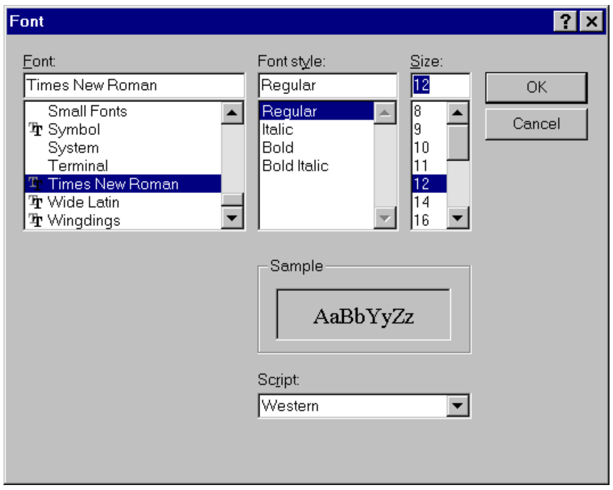


The picture to the right shows a screen dump of all the marker styles in sizes 1 to 10. If you need to tell them apart on screen, sizes below 3 should be avoided. If you have excellent eyesight and a high-resolution printer, size 1 is viable in printed output.

Font

This command sets the font that is used for each window. The font selection dialog is generated by the operating system, and varies with the version of Windows.

The font size changes the space allocated to data channels in a data view. Smaller fonts give more space to the channels, however fonts need to be large enough to read easily. You can set different fonts in each data or text window.


Font

Font: Times New Roman
Small Fonts
Symbol
System
Terminal
Times New Roman
Wide Latin
Wingdings

Font style: Regular
Regular
Italic
Bold
Bold Italic

Size: 12
8
9
10
11
12
14
16

Sample
AaBbYyZz

Script: Western

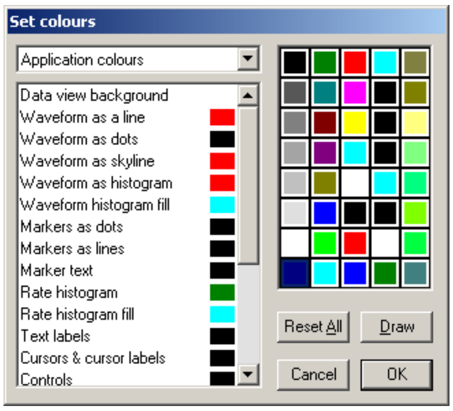
In a text-based view, this command opens the editor settings dialog for the current text view type where you can set fonts and text styles for view. This is described in the Edit menu Preferences dialog under the Display tab.

Use Colour and Use Black And White

If you have a colour monitor, you can choose to display your data files in colour. The Use Colour menu item switches from black and white data displays to colour. If you change to colour, the menu item changes to Use To Black And White. You may prefer to work in monochrome if you have to print the end result in black and white.

Change Colours

You can choose the colours for almost everything in Signal. If you open the dialog with an active file, memory or XY view, the dialog has multiple pages. Select a page with the drop-down list at the top. The pages are:



Set colours

Application colours

Data view background	
Waveform as a line	🟥
Waveform as dots	⬛
Waveform as skyline	🟥
Waveform as histogram	🟥
Waveform histogram fill	🟦
Markers as dots	⬛
Markers as lines	⬛
Marker text	⬛
Rate histogram	🟩
Rate histogram fill	🟦
Text labels	⬛
Cursors & cursor labels	⬛
Controls	⬛

Reset All · Draw · Cancel · OK

Application colours

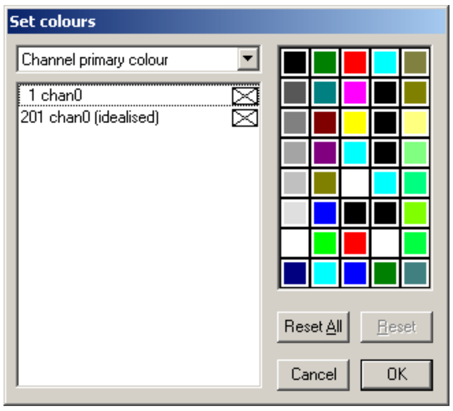
To change colours, select one or more items in the list on the left, and then click a colour in the palette on the right. You can check the result of your action with the Draw button. Cancel removes the window and undoes any changes. OK accepts changes and closes the dialog. The Reset All button returns the list and the palette to a standard set of colours. You can change the following:

Data view background	Marker text	Tagged frames background
Waveform as line	Rate histogram	Frame list traces
Waveform as dots	Rate histogram fill	XY view background
Waveform as skyline	Text labels	Standard deviation/SEM
Waveform as histogram	Cursors and cursor labels	Fitted data
Waveform histogram fill	Controls	Convoluted trace
Markers as dots	Data display grid	Closed state
Markers as lines	Axis markings & text labels	

See the Edit menu Preferences for text view colours.

Channel primary colour
Channel secondary colour
Channel background colour

These three pages assign colours to channels in the current file or memory view and override the application colours set for drawing modes. The primary colour sets the drawing colour for lines, waveforms, markers, and histogram outlines. The secondary colour is for filling histograms and for drawing the SEM and SD in data views. The channel background colour overrides the view background for the area occupied by the channel data. An X in a box marks a channel with no colour override.



Set colours

Channel primary colour

1 chan0	☒
201 chan0 (idealised)	☒

Reset All · Reset · Cancel · OK

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. The equivalent script command is `ChanColour()`.

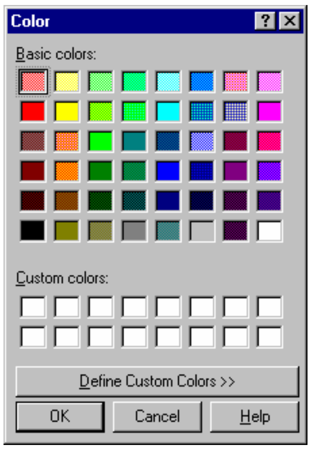
XY channel colours

This page lets you change the XY channel line colours and the fill colours. Changes made on this page are applied immediately; there is no Draw button. The equivalent script command is `XYColour()`

View colours

This page allows you to override the application colours set for the current view. At the moment you can only override the view background colour. The equivalent script command for this is `ViewColour()`.

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 Signal preferences in `Signal.sgp`, not in the data files. When you restore a Signal data file, the colours will be those that are currently active, not those in use when you last saved it.

Folding

This item, which is available for script and sequencer views, allows you to select the style of code folding margin (including disabling code folding), while the Expand All Folds command unfolds all fold points, Collapse All Folds folds all fold points, and Toggle All Folds finds the first fold point in the document, reads its folded or unfolded state and then sets all folds in the document to the opposite state.

Show Gutter

The gutter is an area to the left of the text in a text view, usually with a grey background, that can be used to select lines of text, except in a script view where clicking on it sets break points. The gutter is also used in script views to display the current line pointer. This menu command shows and hides the gutter, equivalent to the `Gutter()` script command. The gutter shares the same display background colour as line numbers. The gutter is normally visible.

Show Line numbers

You can choose to display line numbers in any text view. Line numbers usually have space for up to 5 digits, but if you need more than 99,999 lines of text you can use the `ViewLineNumbers()` script command to increase the displayed digits. This menu command makes the line number area visible if it is invisible and hides it if it is visible. You can change the appearance of the line number region with the View menu Font command. Select the Line number style, and a line number will appear in the example window so you can preview any changes.

Keyboard display control

Windows software is usually orientated towards control by means of the mouse and menus, but it is often convenient to use the keyboard instead. For interactive adjustments of the data or display, keyboard control can also be much faster. With this in mind, Signal includes keyboard shortcuts to handle most display manipulation requirements:

Scroll data down / up	`Cursor Down` / `Cursor Up`
Decrease Y range / increase range	`Ctrl+Down` / `Ctrl+Up`
Optimise Y range	`End`
Show all Y range	`Home`
Y axis dialog	`Ctrl+Y`
Scroll left / right	`Cursor Left` / `Right`
Decrease X range / increase range	`Ctrl+Left` / `Ctrl+Right`
Show all X range	`Ctrl+Home`
X axis range dialog	`Ctrl+X`
Next frame / previous frame	`PgUp` / `PgDn`
First frame / last frame	`Ctrl+PgDn` / `Ctrl+PgUp`
Zoom / un-zoom channel	Double-click
Hide selected channels	`Del`
Customise display	`Ctrl+Del`

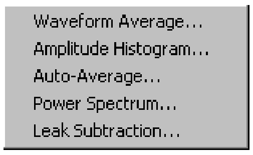
Some of these shortcuts are documented with the appropriate menu command, others do not have an equivalent command. All display shortcuts are listed here for convenience. There are more shortcuts provided for data manipulation; see the Analysis menu chapter.

12 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 two regions contain commands that are associated with processing data from a data or memory view into a memory or XY view. The New Memory View command opens a pop-up menu from which you can choose an analysis that generates a memory data window, for example a waveform average of one or more channels. The New XY View commands are 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 these commands. The Fit Data command opens a dialog from which you can fit mathematical functions to sections of data in time, result or XY views.

The third region holds commands to append new data frames to documents and to delete appended frames. The fourth region holds commands that use the frame buffer, this buffer is a separate frame of data, not part of a data file, that can be used in a number of ways. This region also includes the multiple frames dialog, which carries out operations on many frames. The final regions holds channel data modification commands, control frame tagging and provide FIR and IIR digital filtering, which are described separately in the *Digital filtering* chapter.

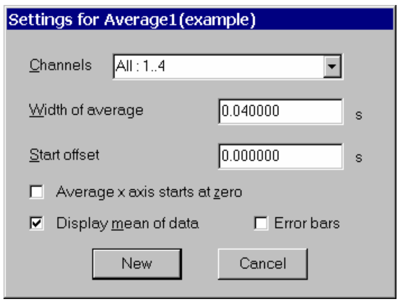
New Memory View



This command is enabled when a file view is selected. It opens a pop-up menu in which you can select an analysis type from: Waveform Average, Amplitude histogram, Auto-Average, Power Spectrum and Leak Subtraction (this last if you have clamping enabled). Selecting an analysis opens a Settings dialog where you set the analysis parameters and other information needed to construct a memory view to hold the analysis results. This dialog can be recalled from the memory view to change the analysis parameters.

Click New in the Settings dialog box to create a memory view with all data values set to zero and to open the Process dialog, in which you select the source data frames to analyse. The results of analysing different sets of frames can be summed by repeatedly using the Process dialog to select different frames.

Waveform Average


Settings for Average1 (example)

Channels: All : 1..4

Width of average: 0.040000 s

Start offset: 0.000000 s

☐ Average x axis starts at zero

☑ Display mean of data

☐ Error bars

This analysis averages waveform data channels across multiple frames. The Channels field sets the waveform channels to average. The Width of average field sets the width of the new memory view. The Offset field sets the start time for the data as an offset from the start of the frame. An offset of zero selects data from the start of the frame, regardless of the frame start time. The data from each frame to be analysed starts at Frame start + Offset and runs up to Frame start + Offset + Width. If this data range extends beyond the end of the frame data for any sweep, the sweep is not added into the average and an error message is generated.

If you check Average x axis starts at zero, the X axis of the memory view created by this analysis starts at zero, otherwise the x axis starts at the start time of the first section of data added into the average. This is Frame start + Offset for the first frame analysed.

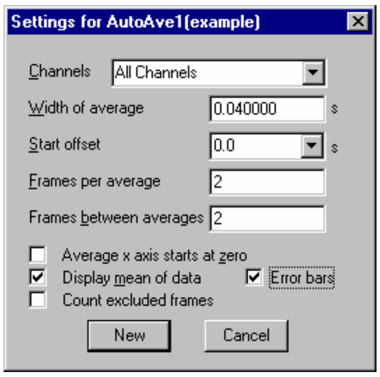
The Display mean of data checkbox selects between displaying the mean data value or the sum of all sweeps added into the average. It can be changed after the data has been generated.

If you check the Error bars checkbox, extra information is saved with the averaged data so that Signal can display the standard deviation and standard error of the mean of the resulting data. The Waveform Draw Mode dialog controls the display of the error information.

The New button (or Change if the memory view already exists) closes the dialog, creates the new memory view and opens the Process dialog, described below.

Auto-Average

This form of analysis averages waveform channels in the same manner as standard waveform average processing, but automatically produces a memory view with multiple frames, each frame holding a separate average. Each memory view frame is generated from a preset number of source frames, the groups of frames used can overlap, be adjacent, or can have gaps of unused data between them.


Settings for AutoAve1(example)

Channels	All Channels
Width of average	0.040000 s
Start offset	0.0 s
Frames per average	2
Frames between averages	2

☐ Average x axis starts at zero
☑ Display mean of data ☑ Error bars
☐ Count excluded frames

The Settings dialog holds fields for the channels, the width of the average, the data start offset and the number of frames and frame separation per average. The Channels, Width of average and Offset fields are all the same as for the standard waveform average processing described above, as are the Average x axis starts at zero, Display mean of data and Error bars checkboxes.

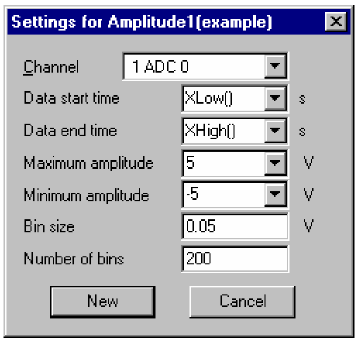
The Frames per average item sets the number of source frames that are used to make up each average. The Frames between averages item sets the number of frames in the source between the start of one average and the start of the next.

Thus if you want 4 frames to make up each average you would set Frames per average to 4. If you set Frames between averages to 4, Signal will use frames 1, 2, 3 and 4 for the first average, 5, 6, 7 and 8 for the second average and so on. Setting Frames between averages to 6 would mean that frames 1, 2, 3 and 4 make average frame 1 as before, but now frames 7, 8, 9 and 10 are used for the next average and frames 5 and 6 are not used. Then again, if you set Frames between averages to less than Frames per average then the data for each average will overlap, with some frames being used to make more than one average.

The Count excluded frames checkbox controls whether the frames used are determined only on frame numbers or if they take into account the subset of frames selected for processing. For example, with 4 frames per average, 4 frames between averages and processing tagged frames only, if Count excluded frames is not checked then the first 4 tagged frames will make up the first average, the next 4 tagged frames the next average and so on. Alternatively if Count excluded frames is checked then those of frames 1 to 4 that are tagged will make up the first average, any tagged frames from frames 5 to 8 make up the second average and so on. In this case some of the averages would be formed from fewer than four frames, any averages formed from no frames are left set to zero.

The New button (or Change if from the Process Settings command) closes the dialog, creates the new memory view and opens the standard Process dialog, described below.

Amplitude Histogram



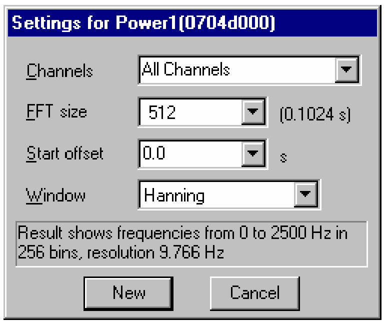
Settings for Amplitude1(example)

Channel	1 ADC 0	
Data start time	XLow()	s
Data end time	XHigh()	s
Maximum amplitude	5	V
Minimum amplitude	-5	V
Bin size	0.05	V
Number of bins	200	

This analysis creates a memory view showing a histogram of the number of points of a waveform channel that lie within particular amplitude ranges. The option will not appear for log-binned data. The processing reads data for the channel being analysed that covers a set time range, and for each data point read the waveform level is used to calculate the corresponding histogram bin. If this bin number exists within the histogram then the bin data is incremented. This process is repeated for every data point read from every frame, the resulting histogram shows the relative frequency of occurrence of the different waveform levels. For example, if your data waveforms were generally at one of two levels then the amplitude histogram of the data would show two peaks with (probably) Gaussian distributions.

The Channel item sets the waveform channel to be analysed, only a single channel can be selected. The Data start time and Data end time items define the time range over which the source data will be analysed. The Maximum amplitude and Minimum amplitude items set the range of X values that the histogram will cover. These items are linked to the Bin size and Number of bins items, both of which set the number of histogram bins. If either of the amplitude range items are changed the Bin size will be adjusted to fill the new range and keep the Number of bins constant. If the Bin size is changed then the Number of bins is adjusted to keep the amplitude range constant. Finally if the Number of bins is changed then the Bin size changes, again keeping the amplitude range constant.

Power Spectrum



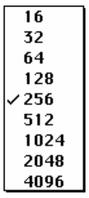
Settings for Power1(0704d000)

Channels	All Channels	
FFT size	512	(0.1024 s)
Start offset	0.0	s
Window	Hanning	

Result shows frequencies from 0 to 2500 Hz in 256 bins, resolution 9.766 Hz

This analysis creates a memory view that holds the power spectrum of a section or sections of waveform data. The option will not appear for log-binned data. If multiple sections are processed the result is an averaged power spectrum. The results of the analysis are scaled to RMS power (where power is amplitude squared), so they can be converted to energy by multiplying by the time over which the transform was done. Signal uses a Fast Fourier Transform (FFT) to convert the waveform data into a power spectrum.

The channels and offset items in the settings dialog behave the same way as they do in the Waveform Average settings dialog.

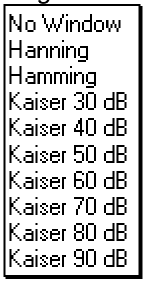


16
32
64
128
✓256
512
1024
2048
4096

The FFT is a mathematical device that transforms data between a waveform and an equivalent representation as a set of cosine waves each with an amplitude and relative phase angle. The version of 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 in the dialog, the dialog will automatically update to show the time range that this block size covers. The way the maths works out, the resulting memory view data ends up with half as many bins as the FFT block size. As for waveform averaging, if the block of data starting at the offset specified runs past the end of the frame the sweep is discarded and no analysis is done.

The data in the memory view spans a frequency range from 0 to half the sampling rate of the source 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.

Windowing of data



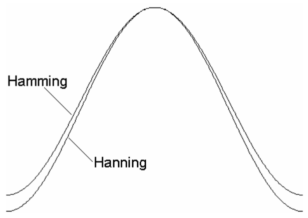
The mathematics behind the FFT assumes that the input waveform repeats cyclically. This means that the maths treats the block of data as though it was taken from an input consisting only of that block, repeated over and over again. In most waveforms this is far from the case; if the block was spliced 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.

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). Signal implements the following windows:

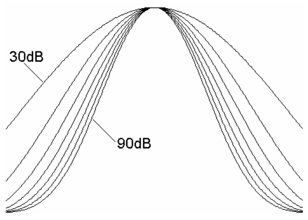
No Window Use this if there is one sine wave, or if more than one, they all have similar amplitude. This has the sharpest spectral peaks, but the worst side-lobes.

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.

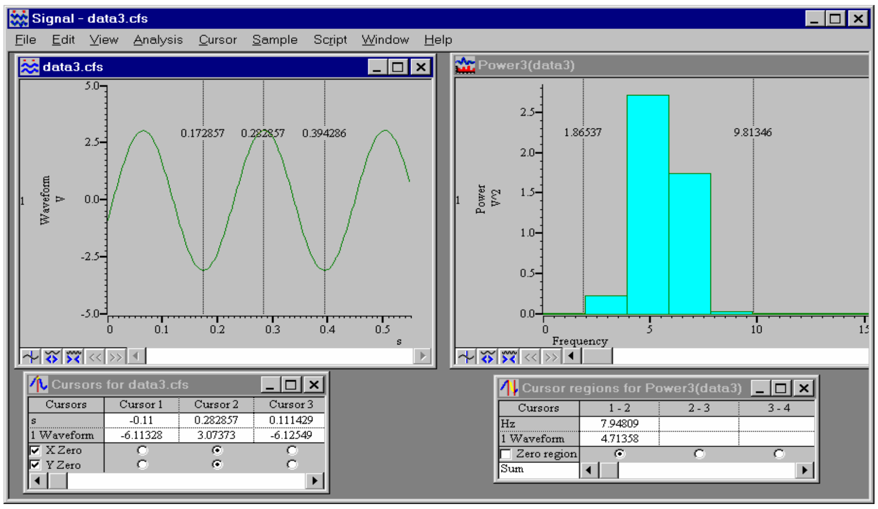
Kaiser These are a family of windows calculated to have known maximum side-lobe amplitude relative to the peak. Of course, the smaller the side-lobe, the more signal is lost and the wider the peak. We provide a range of windows with side-lobes that are from 30 to 90 dB less than the peak.



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 volts^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 3.06 volts (peak to peak amplitude = 6.12). 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 waveform was a perfect sine wave we would predict a RMS power of 4.6818 volts^2 from this waveform ($3.06^2/2$). The cursor analysis of the power shows a total power of 4.7135 volts^2. This is about 0.7% above the predicted result for a perfect waveform.



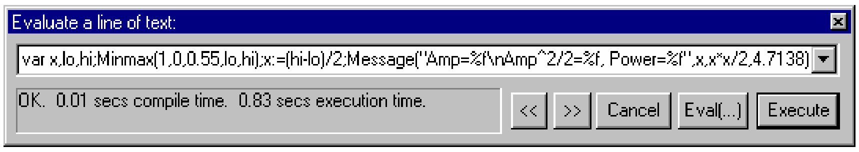
Cursors for data3.cfs

Cursors	Cursor 1	Cursor 2	Cursor 3
s	-0.11	0.282857	0.111429
1 Waveform	-6.11328	3.07373	-6.12549
✓ X Zero	○	●	○
✓ Y Zero	○	●	○

Cursor regions for Power3(data3)

Cursors	1 - 2	2 - 3	3 - 4
Hz	7.94809		
1 Waveform	4.71358		
☐ Zero region	●	○	○
Sum			

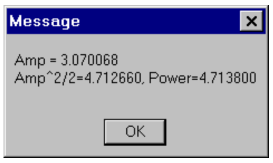
The predicted result is slightly low because the waveform samples used for the cursor measurements are unlikely to lie at the exact peak and trough of any particular cycle. Using the script language `Minmax()` function on a waveform channel to find the maximum and minimum values over a wide time range gives a slightly larger amplitude, and a much closer agreement:



```
var x,lo,hi;Minmax(1,0,0.55,lo,hi);x:=(hi-lo)/2;Message("Amp=%f\nAmp^2/2=%f, Power=%f",x,x*x/2,4.7138)
```

```
OK.  0.01 secs compile time.  0.83 secs execution time.
```

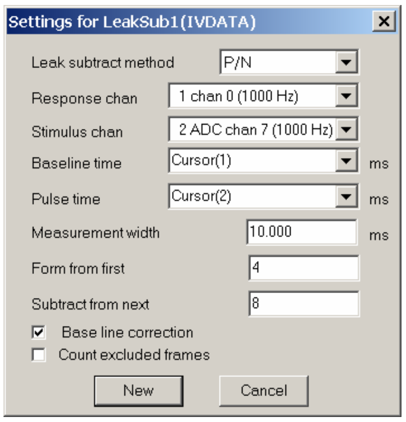
You can use the Evaluate command in the Script menu if you want to try this. It gives a slightly larger value for the amplitude and now the power calculated from the amplitude and the measured power differ by 0.025%. For an explanation of the text in the Evaluate window see *The Signal script language* manual.



```
Amp = 3.070068
Amp^2/2=4.712660, Power=4.713800
```

The duration of one cycle of the waveform (the time between cursor 1 and cursor 3) is approximately 0.2214 seconds, a frequency of 4.52 Hz. Again, this is in agreement with the displayed power spectrum.

Leak Subtraction



Settings for LeakSub1(IVDATA)

Leak subtract method	P/N	
Response chan	1 chan 0 (1000 Hz)	
Stimulus chan	2 ADC chan 7 (1000 Hz)	
Baseline time	Cursor(1)	ms
Pulse time	Cursor(2)	ms
Measurement width	10.000	ms
Form from first	4	
Subtract from next	8	

[✓] Base line correction
[ ] Count excluded frames

This analysis creates a multi-frame memory view by carrying out a leak subtraction analysis on the source data. Leak subtraction is a specialised analysis used by voltage and patch clamp researchers, this analysis will not be available unless clamping support has been enabled in the Edit menu Preferences dialog. The basic technique is to apply a small stimulus, one that does not cause the cell membrane ion channels to turn on, and to measure the current flow through the membrane (made up from resistive and capacitive components) caused by this stimulus. This 'leak measurement' is then scaled by the stimulus amplitude to give the expected leak conductance generated by larger stimuli and this scaled leak is subtracted from the recorded traces to leave only the ion-channel effects. Normally an average leak trace is generated from a number of small pulses to minimise the effects of noise. Leak subtraction makes special use of two channels; the stimulus channel which is used to measure the amplitude of the stimulus pulse so that the leak can be scaled (but which is not modified by the analysis) and the response channel which is the only channel modified by the leak subtraction process. All other channels are ignored and copied into the memory view unchanged.

The Settings dialog holds fields for the leak subtraction mode and to define the channels for the stimulus and the response signals. Baseline time defines a time within the sweep where there will be no stimulus and the Pulse time is a time where there is a stimulus. These two times are used to measure the stimulus amplitude from the relevant channel; the measurements are averaged over the Measurement width (using the time range from time-width/2 to time+width/2).

The following two edit fields are used to specify the frames used to generate the leak measurement and the frames from which the leak is subtracted. These fields are interpreted in different ways depending on the leak subtraction method (see below). If Base line correction is on, the corrected response will also have a DC offset removed so that at the baseline time the response level will be unchanged.

The Leak subtraction method can be set to Basic, P/N, or States. These three modes are very similar, the only real difference being how frames used to generate the leak trace are selected:

Basic here the leak measurement is generated from a fixed contiguous set of frames regardless of which frames are being processed, these frames are set using the First frame for leak and Last frame for leak items. All of the frames processed use this single leak measurement, frames contributing to the leak data are automatically skipped during processing.

P/N here the leak measurements are extracted from the frames that are being processed. The first n frames processed are used to make the leak data, then the next m frames are processed using this leak data. Then the next n frames are used to make a new leak data set and the following m frames are processed using this new leak data, this cycle continues throughout the frames being processed. The values for n and m are entered in the settings dialog using the Form from first and Subtract from next items respectively.

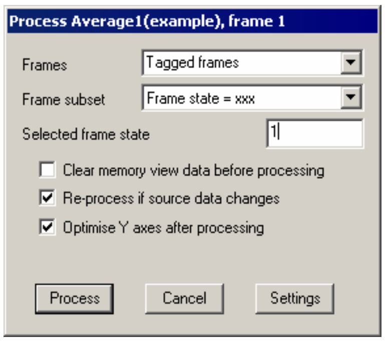
States here the leak data is assembled from all frames within the file with a given state, the state number is entered in the settings dialog. All the frames processed use this set of leak data, frames contributing to the leak data are automatically skipped.

As in Auto-Average processing if, for instance, we only process un-tagged frames and there are some tagged frames in the file, the process will continue to search for un-tagged frames until the required number have been found to complete the formation or subtraction of the leak. If Count excluded frames is turned on, however, then even frames that are excluded from the process will still be counted as part of the frames used.

The New button (or Change if this is used from the Process Settings command) closes the dialog, creates the new memory view and opens the Process dialog, described below. Leak subtraction processing uses the same process dialog, but because leak subtraction creates a set of frames in the memory view the behaviour is subtly different; the Clear bins checkbox, if set, clears out the entire memory view (deleting all frames apart from the first one) and if not set does not accumulate more data into the existing memory view frames but rather appends more frames to the view. For similar reasons the Analysis menu Append Frame command does not create a second set of process parameters; all frames use the same process parameters.

Process…

This command is available when a memory view created using the New Memory View command or an XY created using New XY view is the current view. When you use it a dialog prompts you to select the frames of the source data document to be processed. The Process dialog is also provided automatically when you use the New or Change button from the Process Settings dialog to create or rebuild a memory view.


Process Average1(example), frame 1

Frames: Tagged frames
Frame subset: Frame state = xxx
Selected frame state: 1

☐ Clear memory view data before processing
☑ Re-process if source data changes
☑ Optimise Y axes after processing

The dialog contains a number of items used to select the source data frames that will be processed, plus checkboxes controlling what happens when data is processed.

The Frames and Frame subset items are the main controls used to select source frames. The simplest way to use these is to type in a frame list directly. You can also select the current frame, all frames, tagged or untagged frames or frames with a given state code. If you choose the state code option, the dialog also displays a field into which you can enter the state code to use. The Frame subset selector can be used to further qualify which frames are wanted.

The frame list values are evaluated when the Process button is used, then the frames are processed and the results added into the memory view data.

If you check the Clear memory view before process checkbox, the memory view data is cleared before the results of the processing are added. The Reprocess if source data changes checkbox enables automatic re-processing. Automatic re-processing is optimised to try to prevent unnecessary work, but can still slow Signal significantly on occasion, particularly with large files. If you check the Optimise Y axis after process checkbox, the memory view y axes will be re-scaled after the new results are added into the memory view to best display the data. If you are processing data to generate points in an XY view, an extra checkbox provides automatic X axis optimisation.

If the Settings button is pressed, the process dialog is removed and replaced by the settings dialog.

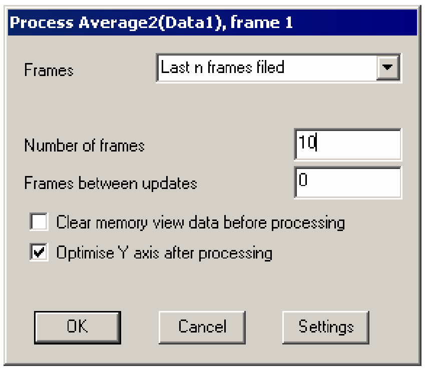
Breaking out of Process

Processing operations can take quite a time, especially in large data documents. You can stop a processing operation early with the `Esc` key.

Multiple frame processes

For Waveform average, Amplitude histogram and Power spectrum processing, you can use the Append frame command (below) to add more frames to the memory view created by processing. This new frame will have the same process settings as the original frame, but will have its own separate Process dialog so that you can analyse a different set of source frames for each memory view frame.

Process command with a new file


Process Average2(Data1), frame 1

Frames: Last n frames filed

Number of frames: 10

Frames between updates: 0

☐ Clear memory view data before processing

☑ Optimise Y axis after processing

The Process dialog is slightly different when the current window is a memory view derived from a sampling document. The dialog is also activated automatically when you create a new memory view from a sampling document or when you press the Change button in the Process settings dialog for a similar memory view.

This form of the Process dialog gives you control over when and how the memory view is updated during sampling. The Frames field contains extra items that are suitable for processing sampled data: Sampled frames and Last n frames filed. The contents of the dialog change depending upon which frame option is selected. In addition there is a new field: Frames between updates.

Sampled frames	all frames that are sampled will be processed. This option is not available in Fast triggers, Fast fixed int or Gap free sampling modes.
All filed frames	all frames saved to disk are processed.
Last n frames filed	process the most recent frames saved to disk; the dialog displays a field in which you can enter the number of frames required. The Clear memory view checkbox is ignored as the memory view is always cleared before processing.

The Frames between updates field sets how often the processing of filed frames occurs. Set this to zero to process as often as possible. If you are processing Sampled frames, then this field is ignored and the memory view is updated for each frame.

Process settings…

This menu command re-opens the analysis settings dialog for the current memory or XY view. This is the same dialog as the one used to define and create the new view except that the New button is now a Change button. The Change button accepts the changed settings, if the changes are significant enough to make any previous analysis incompatible the destination memory view will be cleared and re-initialised.

New XY View

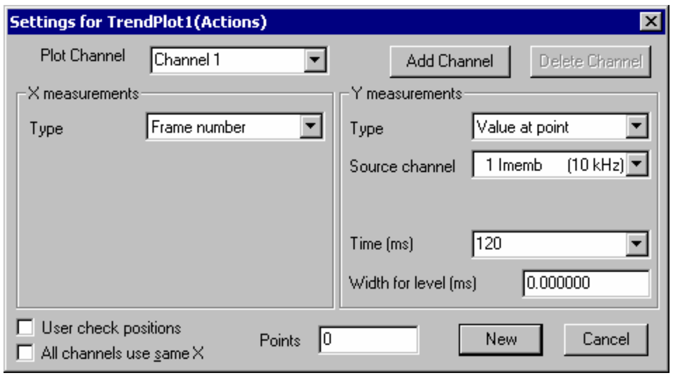
This command, analogous to the New Memory View command, is available when a file or memory view is selected. It provides a pop-up menu from which you can select an analysis type, Trend plot or Measurements analysis are available.

Analyses that generate XY view data can be used online in the same way as the memory view analyses to take measurements from the sampled data as it is collected. In addition, these analyses can also be carried out online on memory views that are themselves being generated by analysis of the sampled data. In this circumstance, the exact manner in which the XY measurements are generated depends upon the form of analysis generating the memory view data. For Auto Average analysis, the XY measurements are taken when a new averaged frame is completed - when the required number of source frames have been averaged into the frame. For Leak Subtraction analysis, the XY measurements are taken whenever a new frame is added into the leak subtracted data. For all other analyses all of the available memory view frames are re-processed in a manner very similar to offline analysis whenever any of the memory view data is changed, so the XY view data automatically updates to reflect the changing memory view contents.

Trend plot

A trend plot consists of sets of measurements taken from a source data document and plotted into an XY view. Trend plot analysis generates a single measurement from each source data frame, each measurement having an X and a Y part which generate the relevant parts of the XY data. An XY view holds lists of XY points, one per channel. Trend plot analysis can create data points using a wide variety of measurements for both the X and Y values, up to 32 separate channels of XY data can be created.

Selecting Trend plot analysis provides a Settings dialog where you define the analysis parameters. This dialog is also available later on to change the analysis parameters.


Settings for TrendPlot1(Actions)

Plot Channel: Channel 1

X measurements
Type: Frame number

Y measurements
Type: Value at point
Source channel: 1 Imemb (10 kHz)
Time (ms): 120
Width for level (ms): 0.000000

User check positions
All channels use same X
Points: 0

The dialog consists of four regions. At the top is a control to select the currently viewed channel (in the XY view) and buttons to add and delete channels. In the middle are two very similar regions that set the parameters for the X and Y measurements and at the bottom are the standard New and Cancel buttons plus controls that affect the XY view data.

You can set the XY view channel title by typing into the Plot Channel selector and create additional channels by using the Add Channel button. The Delete Channel button deletes the current channel; you cannot delete the last remaining channel.

The X measurements and Y measurements sections are the same as each other. Both hold a selector for the type of measurement plus a variety of other parameters used by the measurement process; a channel to take measurements from, items for the one or two time values needed, a measurement width and a fit coefficient number. These additional parameters are shown and hidden according to the type of measurement selected. The types of measurement available are:

Value at point	the value on the channel specified at the time specified, measured using the set width. A non-zero Width will give a measurement averaged over Time–Width/2 to Time+Width/2,

	a zero width reads the value of the nearest available waveform data point.
Value difference	the difference between the value on the channel at the time specified and the value at the reference time, both measurements using Width as for Value at point.
Value above baseline	the difference between the channel value at the time specified and the value at the reference time. Only the reference time measurement uses the specified width, the other is always a single-point measurement.
Value ratio	the channel value at the time specified divided by the value at the reference time, both using the specified width.
Value product	the product of the channel value at the time specified and the value at the reference time, both using the specified width.
Time at point	the time specified. This can be a cursor position; if that cursor's mode is set to move to a feature, it measures the feature position in each frame.
Time difference	the difference between the time specified and the reference time. Either or both of these can be a cursor position that can vary.
Frame number	the frame number. This is often used as the X measurement to give a plot of ‘measurement against frame’.
Absolute frame time	the absolute start time of the frame. Often used as an alternative to the frame number to give a ‘measurement against time’.
Frame state value	the frame state value for the frame in question.
Fit coefficients	a coefficient resulting from a fit on the specified channel. The coefficient index from zero upwards should be specified.
User entered value	a value entered by the user. A dialog opens to read the value.
Expression	a string entered by the user is evaluated and the result used.
Cursor regions	any measurement available in the cursor regions window can be used (see the *Cursor menu* chapter).

The channel selector and time entry fields are as standard for Signal and should be familiar to you and easy to use. Don’t forget that, in addition to entering a time value directly or selecting an item such as “`Cursor(2)`” or “`XLow()`”, you can apply an offset to these selected value allowing you to enter “`Cursor(2)-0.1`” or “`XLow()+1`”.

The User check positions checkbox below the X measurements will, when checked, cause Signal to pause on each frame after positioning any active cursors. This allows you to check that the cursor positions are correct and to skip individual measurements or cancel all further measurements.

The All channels use same X checkbox below disables the X measurement for all XY view channels apart from the first one, then all the XY channels created will use the same X value. This makes the trend plot setup easier and when the XY data is converted to text only a single column of X values is created, which makes the text easier to handle in spreadsheets.

The Points item below the X and Y measurements allows you to specify the number of points the currently selected XY channel can contain before old points are deleted to make way for new data. Set this field to zero if you want all data points to be retained.

The New button (or Change if this is used from the Process Settings command) closes the dialog, creates the new XY view and opens the Process dialog, described above. Processing for trend plots is very similar to standard memory view processing; the frames specified are used to generate measurements which are added to the view data. If the Clear XY view data before processing checkbox is checked, all of the data points in the XY view will be deleted first. In addition to the Y axis optimisation control, there is also an extra checkbox for view X axis optimisation after processing.

Trend plots and active cursors

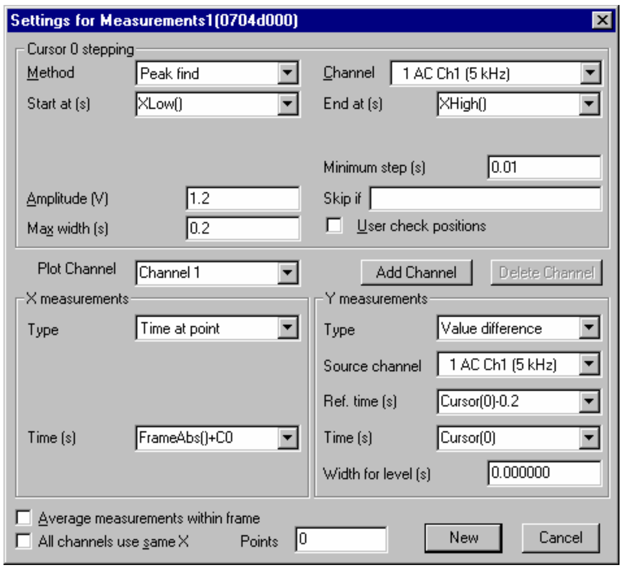
Trend plot analysis can be made a great deal more powerful by the use of active cursors. Active cursors can be set up to move to features in the data such as a maximum or a threshold crossing, so that they are automatically aligned to significant features in each data frame. By using the cursor position to specify a time or time range for a measurement, many useful effects can be achieved. For example, with cursor 1 set to find the maximum value between a preset time range within the frame, you could have the X measurement being the frame number and the Y measurement being Time at point. With the time field set to “Cursor(1)”, you would get a graph of time for the maximum versus the frame number. See the *Cursor menu* chapter for complete details of active cursors.

Measurements

Measurements analysis is very similar indeed to trend plot analysis, the basic difference is that multiple measurements can be taken from each frame. This is achieved by using a special cursor, cursor 0, which is used by Measurements analysis to iterate through each frame searching for features. A separate XY data point is generated for each feature found by the cursor 0 search. Even more than Trend plot analysis, Measurements analysis depends strongly upon the use of active cursors, both as a mechanism for controlling cursor 0 feature iteration and to cause other cursors to position themselves around each cursor 0 position found so that each feature is separately analysed.

When Measurements analysis is used Signal provides a Settings dialog used to define the various analysis parameters. This is very similar to the Trend plot settings.

The main and highly visible difference is the addition of the Cursor 0 stepping section at the top of the dialog, an extra measurement type and an option for averaging measurements are also available. Everything else is identical to the trend plot settings.


Settings for Measurements1(0704d000)

Cursor 0 stepping
Method: Peak find
Channel: 1 AC Ch1 (5 kHz)
Start at (s): XLow()
End at (s): XHigh()
Minimum step (s): 0.01
Amplitude (V): 1.2
Skip if:
Max width (s): 0.2
☐ User check positions

Plot Channel: Channel 1

X measurements
Type: Time at point
Time (s): FrameAbs()+C0

Y measurements
Type: Value difference
Source channel: 1 AC Ch1 (5 kHz)
Ref. time (s): Cursor(0)-0.2
Time (s): Cursor(0)
Width for level (s): 0.000000

☐ Average measurements within frame
☐ All channels use same X
Points: 0

The Cursor 0 stepping section of the dialog contains items that define how cursor zero searches through each frame that is processed to find features in the data. When a frame is processed, cursor zero is initially positioned at the Start at time. Then (except for Expression mode, where a measurement is generated at the initial position) the feature is searched-for, other cursors are positioned and a measurement is taken if the search has been successful and the other active cursors have also succeeded in their searches (are valid). This search-and-measure process is repeated until the search fails or the End at time is reached. An XY point is generated for each feature found; there may be no features found in a given frame of data and in that case no measurements will be generated for that frame. The available cursor 0 stepping methods are:

Peak find	local maxima in the data are found. The data must rise and fall again by the set amplitude, a maximum allowed width for the peak can be set.
Trough find	local minima in the data are found. The data must fall and then rise again by the set amplitude, a maximum allowed width can be set.
Rising threshold	upwards crossings through a threshold level are found. The data must first fall far enough below the threshold level as set by a hysteresis value, a delay value sets how long it must remain above the threshold.
Falling threshold	downwards crossings through a threshold level are found. The data must first rise far enough above the threshold level as set by a hysteresis value, a delay value sets how long it must remain below the threshold.
Outside dual thresholds	transitions from within a pair of levels to outside the levels are found.. The data must first of all lie sufficiently far within the levels as set by a hysteresis value, a delay value sets how long it must remain outside the levels.
Within dual thresholds	transitions from outside a pair of levels to within the levels are found. The data must first of all lie sufficiently far outside the levels as set by a hysteresis value, a delay value sets how long it must remain within the levels.
Slope peak	local maxima in the slope of the data (steeply rising data values) are found. The slope measured must first rise and then fall again by the set amplitude, a width value sets the time range over which the slope is measured.
Slope trough	local minima in the slope of the data (steeply falling data values) are found. The slope measured must first fall and then rise again by the set amplitude, a width value sets the time range over which the slope is measured.
+ve slope threshold	upwards transitions of the slope through a threshold level are found. The slope value must first be sufficiently below the threshold level as set by a hysteresis value, a width value sets the time range over which the slope is measured.
-ve slope threshold	downwards transitions of the slope through a threshold level are found. The slope value must first be sufficiently above the threshold level as set by a hysteresis value, a width value sets the time range over which the slope is measured.
Turning point	points at which the slope changes sign in either direction (peaks and troughs in the data) are found. A width value sets the time range over which the slope is measured.
Data points	a specified number of data points or marker items.
Expression	a string specifying a time such as “Cursor(0)+0.1” is evaluated and cursor 0 set to the position generated. This is most useful for stepping cursor 0 through the data by a fixed amount, but other effects are possible. Iteration stops when the cursor passes the End at location, if the expression used does not move cursor zero onwards through the data then the analysis will hang.

These stepping methods and nearly all of the parameters controlling them are identical to the active cursor modes for cursor zero, see the active cursor documentation in the *Cursor menu* chapter of this documentation for complete details of these. If cursor zero is already set up as an active cursor when a Measurements analysis is created the current active cursor parameters are copied into the settings dialog.

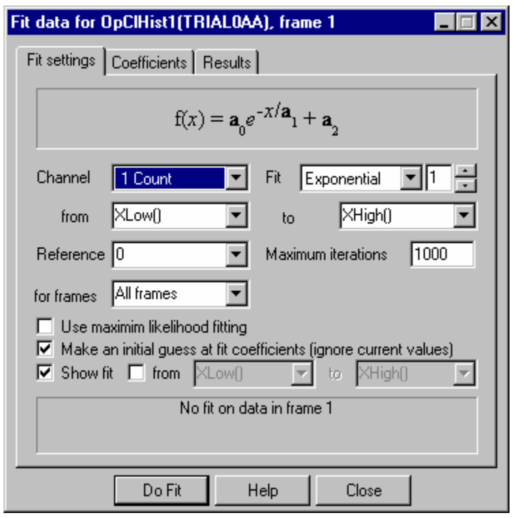
The Skip if parameter can contain a string such as “Cursor(1) < Cursor(2)” that is evaluated after each cursor zero iteration and other cursor placement. If the result of evaluation is non-zero then no measurement will be taken for this cursor zero position. The User check position checkbox, if set, allows the user to check each cursor positioning and accept or reject them individually.

The X and Y measurements are almost identical to the trend plot measurements but they do contain an extra measurement type, Iteration count which, when used with averaged measurements, counts how many features the cursor 0 stepping found, this mechanism could be used to count action potentials, for example.

The Average measurements within frame checkbox at the bottom of the dialog applies to all measurement channels. If set then only a single XY data point is generated per frame, this being the average of the data values measured. So, for example, you could produce a plot of mean response amplitude against frame number.

Fit data

This command opens a tabbed dialog from which you can fit mathematical functions to channels in a file, memory or XY view. If you fit data to a channel in a file or memory view and error bars are displayed, the fit minimises the chi-squared value, otherwise the fit minimises the sum of squares of the errors between the data and the fitted curve.


Fit data for OpClHist1(TRIAL0AA), frame 1

Fit settings | Coefficients | Results

$f(x) = \mathbf{a}_0 e^{-x/\mathbf{a}_1} + \mathbf{a}_2$

Channel: 1 Count   Fit: Exponential 1
from: XLow()   to: XHigh()
Reference: 0   Maximum iterations: 1000
for frames: All frames

☐ Use maximim likelihood fitting
☑ Make an initial guess at fit coefficients (ignore current values)
☑ Show fit ☐ from XLow() to XHigh()

No fit on data in frame 1

In addition to best-fit coefficients and an estimate of how much confidence to place in them, you also get an estimate of how likely it is that the model you have fitted to your data can explain the size of the chi-squared value or sum of errors squared. If your data does not have error bars, these estimates are based on the assumption that all data points have the same, normally distributed error statistics.

The dialog has three tabs:

Fit settings	Set the fit type and range of data to fit and range to display.
Coefficients	Set the starting point for your fit and optionally fix coefficients.
Results	Display the fitting results and show residual errors.

The three buttons at the bottom of the dialog are common to all pages. The Help and Close buttons do what they say. Do Fit attempts to fit with the current fit settings.

Fit settings This page of the Fit Data dialog controls the type of fit, the data to fit and what to display. The area at the bottom of the window gives a synopsis of the current fit state. Fields are:

Channel You can select a single channel from the current view. If this is a file or memory view, the channel must have a y axis. If you change the display mode of a marker-based channel, any fit associated with the channel will most likely become invalid.

Fit The fit to use is defined by its name and the order of the fit (a number). For example, an exponential fit allows single exponents or double exponents. The window at the top of the dialog displays the mathematical formula for the fitting function. The following fits are currently supported (N is the maximum order allowed):

Name	N	Comments
Exponential	2	This fit includes an offset. You can force a zero offset in the coefficients page. Set a local reference point for the fit, otherwise the even-numbered coefficients may become too large to be useful.
Polynomial	5	These fits do not require starting values for the coefficients.
Gaussian	2	If you attempt to fit two overlapping peaks you may need to manually adjust the guesses for the peak centres to get convergence.
Sine	1	You can fit a single sinusoid with an offset. If the frequency guess (in radians) is not reasonably close, the fit may not converge.
Sigmoid	1	A single sigmoid may be fitted.

Range You fit data over a defined x axis range, set by the between and and fields. You can choose values from the drop down list or type in simple expressions, for example `Cursor(1)+1`. There must be at least as many data points to fit as there are coefficients. For example, to fit a double exponential, which has 5 coefficients, you need at least 5 points. Most fits will use many more points than coefficients.

Reference This is the x axis position to use as the zero value of *x* in the fitting function. The most common value for this would be the start point of the fit. However, in some cases you may want this to be elsewhere. For example, in exponential fitting, you may want to calculate the likely amplitude of a trace at some position. Making this position the reference point makes it easy to calculate the amplitude (it is the sum of the even-numbered coefficients).

Maximum iterations All fits except the polynomial are done by an iterative process. Each iteration attempts to improve the coefficient values. The iterating stops when improvements in the fit become insignificant, the iteration count is exceeded, the mathematics of the fitting process suggests that the fit is not going to improve or there is a mathematical problem. This field sets the maximum number of iterations to try before giving up.

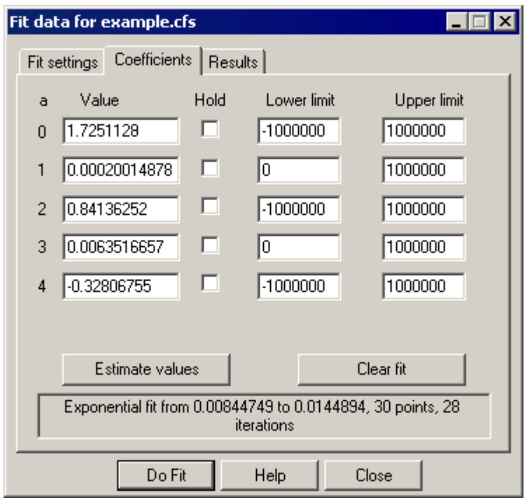
for frames It is possible to have the fit run automatically across several frames at once. Choose the frames whose data you wish to fit here. If you select Frame state = xxx, an extra field is provided for entry of the state code value.

Use maximum likelihood fitting This option is only available when fitting exponential curves to data derived from a single-channel idealised trace by generating open/closed time or burst duration histograms and when the processing link between the original trace data and the binned histogram data being fitted is still present. When these conditions apply Signal can use maximum likelihood fitting techniques (which require access to the original, unbinned, trace data). The details of maximum likelihood fitting are a topic beyond the remit of this manual; suffice it to say that by using the original idealised trace data you avoid mathematical complexities and errors caused by the histogram binning process.

Make an initial guess The iterative fits need a starting point. There are built-in guessing functions that usually generate a starting point near enough to the solution that the fitting process can converge. If you check this box, these guessing functions are used each time you click the Do Fit button. Otherwise, each fit starts with the current values.

Show fit Check this box to display the current fit for the current channel. If the *From* box is checked, you can also choose the range over which to display the fitted data. If this box is not checked, the fit is displayed over the range that the data was fitted.

Coefficients This page of the Fit Data dialog lets you set the starting values for iterative fits. You can also use this page to hold some of the coefficients to fixed values and you can set the allowed range of values for fitting.



Fit data for example.cfs

a	Value	Hold	Lower limit	Upper limit
0	1.7251128	☐	-1000000	1000000
1	0.00020014878	☐	0	1000000
2	0.84136252	☐	-1000000	1000000
3	0.0063516657	☐	0	1000000
4	-0.32806755	☐	-1000000	1000000

Exponential fit from 0.00844749 to 0.0144894, 30 points, 28 iterations

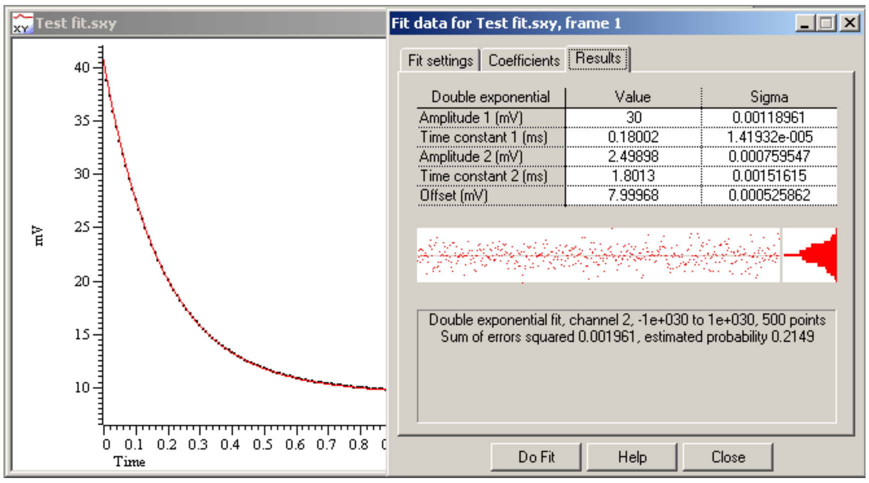
If you know the value of one or more of the coefficients, type the value in and check the Hold box next to it. For example, in an exponential fit you may know that the final coefficient (the offset) is zero.

The limit values are applied after each iteration. The fit may have to follow a convoluted path before it converges on a solution, so do not set the fit limits too close to an expected solution as this may prevent convergence.

The Estimate values button can be used to guess initial values for fitting based on the raw data. The Clear fit button removes the fit from the channel.

Results The results page of the Fit Data dialog holds information about the last successful fit done with the dialog The page has three regions: coefficient values at the top, a message area at the bottom, and a plot of the residuals (differences between the fit and the data) in the middle. The residuals are displayed immediately after a fit but will not be displayed if you close the dialog and reopen it.

Coefficient values



Fit data for Test fit.sxy, frame 1

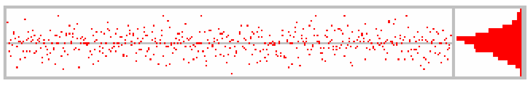
Double exponential	Value	Sigma
Amplitude 1 (mV)	30	0.00118961
Time constant 1 (ms)	0.18002	1.41932e-005
Amplitude 2 (mV)	2.49898	0.000759547
Time constant 2 (ms)	1.8013	0.00151615
Offset (mV)	7.99968	0.000525862

Double exponential fit, channel 2, -1e+030 to 1e+030, 500 points
Sum of errors squared 0.001961, estimated probability 0.2149

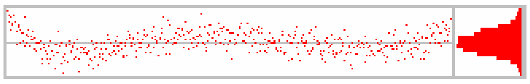
The Value column holds the fitted value that minimised the chi-squared or sum of squares error for the fit. The Sigma column is an estimate of how the errors between the fitted curve and the original data translates into uncertainty in the fit coefficients given that the model fits the data and that the errors in the original data are normally distributed. If a coefficient is held, the Sigma value will be 0. The *Testing the fit* section gives more information on the derivation of these values and how to interpret them. You can select rows, columns or individual cells in this area, the use `Ctrl+C` to copy them to the clipboard. This also copies a bitmap image of the page to the clipboard.

Residuals

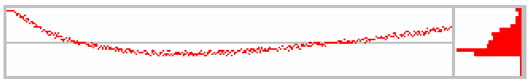
This section of the page displays the differences between the data points and the fitted curve in the large rectangle and a histogram of the error distribution on the right. The error plot is self-scaling based on the distribution of errors; the plot extends from +3 at the top to –3 at the bottom times the RMS (root mean square) error. The grey line across the middle of the plot indicates an error of zero.



In the case that the data can be modelled by the fitting function plus normally-distributed noise, you would expect to see residuals distributed randomly around the 0 error line and the histogram on the right should resemble a normal curve.



If the data cannot be modelled in this way, you would expect to see evidence of this in the residuals. In this example (generated by fitting a cubic to data that was actually a double exponential), you can see that there are clear trends in the errors.



In extreme cases, the error due to the wrong model being used becomes much larger than the errors due to uncertainty in the data values, and you get a residual plot like this one.

Message area

This area displays a summary of the fit information that you can select with the mouse and copy to the clipboard. The first line holds the type of the fit, the channel number, the ordinate range and the number of points in this range. For example: "Double exponential fit, channel 1, 0 to 50, 50 points".

The contents of the second line depend on the source of the data. If you are fitting a result view channel that has error information displayed, the second line displays the chi-squared error value for the fit and the probability that you would get a chi-squared value of at least that size if the function fits the data and the errors are normally distributed. For example: "Chi-square value 58.6, probability 0.5867".

In all other cases, the second line displays the sum of the squares of the errors between the data and the fitted function and an estimate of the probability that you would get a sum of squares of errors of at least this size based on the assumptions that the errors in the original data had a normal distribution that was the same for all points. For example: "Sum or errors squared 1.22, estimated probability 0.8553".

If the probability value is very low or very high, there are extra lines of information warning that the fitted function plus normally-distributed noise is unlikely to model the data, or that the errors in the original data have probably been over-estimated.

Context menu

If you right click on this page you are offered a context menu that contains Copy, Log and Log Titles commands. The Copy command copies selected sections of the results, or all the results if there is no selection to the clipboard as text. It also copies the page as a bitmap. The Log command prints a one-line synopsis of the current fit to the log window. The Log Titles command copies a suitable set of titles for the logged data.

Testing the fit

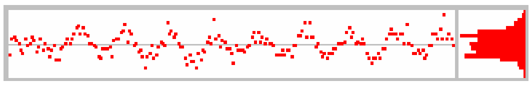
When you fit a model to measured data to obtain the best-fit coefficients, there are two questions you would like answered:

1. How well does this model fit the data? Put another way, how likely is it that this model plus some degree of random variation can explain my data set?
2. Given that the model does fit the data, how much confidence can I place in each of the fitted coefficient values?

When we talk about fitting curves to data, we are making the implicit assumption that you took measurements from some process that follows a model, and that this model can be expressed as a mathematical function with adjustable parameters, which are our fitting coefficients. Further, we assume that the measurements you make are not perfect; they have random variations with a known probability distribution about the correct value. To allow us to calculate likelihoods, we assume that this probability distribution is a normal (Gaussian) distribution. In the real world, or course, only some of this may apply. You may have no a priori knowledge of the distribution of errors in your original data, and this distribution may be anything but normal.

Chi-square fits

In the ideal case, where you know the standard deviations of each data point, the fitting minimises the chi-squared value, which is the sum of the squares of the differences between the model and the data points divided by the standard deviation of data point values. Given a chi-squared value and the number of points it was measured from, we can calculate that probability of getting a chi-squared value at least this large, due to random variations in the data. This is the value given in the Results tab. Ideally, you would like to see a value around 0.5, meaning that you were equally likely to get a larger value as a smaller one. Values very close to 1 mean that, given the errors in each point, the data is too close to the model. Either the error estimates are too large, or the data has been "improved". Although you can hope for probabilities in the range 0.1 to 0.9, values down to 0.01 may occur for acceptable fits, and even smaller values can occur if your error distribution is not as normal as you thought.



Very low fit probabilities will occur if your data contains variations that are significant compared to the errors in the input values and that are not included in the model. For example, if you are fitting exponents to a sampled waveform that includes perceptible mains interference, you can get a good fit (by eye) to the exponential data, but with a probability of 0.0000 as far as the mathematics is concerned because the model does not include the mains hum and cannot explain why the chi-squared value is so high.

If we assume that the model fits the data, we can make an estimate of the standard deviation of the fitted coefficients. This means, that if we re-ran the experiment many times and fitted the data to each set of results, what would be the likely variation in the fitted coefficients. This is presented as the Sigma value in the results tab.

Least-square fits

If there is no error information for each point, we assume that all the points have the same, normal error distribution and the fit minimises the sum of squares of errors between the model and the data. Because there is no independent estimate of the likely spread of the errors in the original data, strictly speaking, there is no way to give a probability of getting an error of at least this size.

However, we can say (though statisticians may shudder), *"Given that the model does fit the data, and that the errors all have the same, normal distribution, then the differences between consecutive errors should also be normally distributed with twice the variance of the errors"*. We use this to estimate the standard deviation of the data and then we apply the probability test. We label this as *estimated probability*. The same comments about likely values apply as for the Chi-square fits, except that very small values may just mean that our estimation process fails for your data.

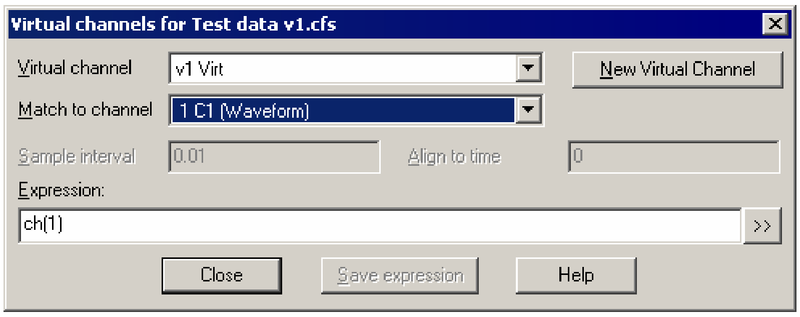
The coefficient Sigma values are calculated on the assumption that the model fits the data, that all the original points have the same standard deviation, and that the standard deviation of the original data can be deduced from the residual sum of squares errors.

Virtual channels

Virtual channels hold waveform data derived by a user-supplied expression from existing waveform channels and built-in function generators. No data is stored on disk; the virtual channel data is recalculated as required. You can match the sample interval and data alignment to an existing channel, or type in your own settings. Channel sample intervals and alignments are matched by cubic splining of the source waveforms. The script language equivalent of this command is `VirtualChan()`.

You can use virtual channels to carry out a wide variety of channel arithmetic (for example sums and differences of channels), to linearise non-linear transducers, to process channel data in various ways (for example DC offset removal), to generate waveform data from scratch and to construct waveforms proportional to a marker channel rate.

There are Analysis menu commands to create a new virtual channel or edit the settings of existing virtual channels. Both commands open the Virtual channel dialog:


Virtual channels for Test data v1.cfs

Virtual channel: v1 Virt

Match to channel: 1 C1 (Waveform)

Sample interval: 0.01

Align to time: 0

Expression: ch(1)

Virtual channel

Use this field to select a virtual channel to work with when you have more than one virtual channel. The New Virtual Channel button creates a new channel.

Match to channel

You can select an existing waveform-based channel (but not a virtual channel) from which to copy the sample interval and data alignment, the virtual channel first point time, sample interval and point count will all be set to match the designated channel. Alternatively, you can select Use manual settings and type in the interval and alignment yourself.

Sample Interval

This field sets the sample interval between data points in the virtual channel in seconds. You can edit the sample interval if you select Manual Settings in the Match to channel field. This field accepts expressions; for example, to set 27 Hz you can type `1/27`. The number of data points in the virtual channel will be set to the frame width divided by the interval.

Align to time

This field sets the time of a data point in the virtual channel. The time of any point and the sample interval completely defines all the sample times for the channel - the first point for the channel will be at the time `Align mod Interval`. You can edit the alignment if you select Manual Settings in the Match to channel field.

Expression

This field holds an expression that defines the virtual channel. Expressions are composed of scalars, vectors and operators. A scalar is a number, such as 4.6 or `Sqrt(2)`. A vector is a list of data values (a script programmer might think of them as arrays) that are derived from a channel or generated by a waveform function. The result of the expression should be a vector - this is the channel data that will be displayed.

Arithmetic operators

You can use the four standard arithmetic operators plus (`+`), minus (`-`), multiply (`*`) and divide (`/`) and comparison operators together with numbers, round brackets and some mathematic and channel functions. The result of combining a vector and scalar with an operator is a vector, for example, the expression `Ch(1)+1` is a vector, being the data points of channel 1 with 1.0 added to each of them.

Dividing by a scalar value of zero is an error. Dividing by a vector holding zeros is not an error and generates special floating-point numbers for positive and negative infinities. These can cause problems in subsequent calculations (and they are difficult to display), but they are more useful than an error!

Comparison operators

You can also use comparison operators less than (<), less than or equal (<=), equal (=), not equal (<>), greater than or equal (>=) and greater than (>); the result of these is 1.0 if the comparison is true and 0.0 if it is false. In the expression `a op b`, where `a` and `b` are vectors and/or scalars and `op` is a comparison operator, if either `a` or `b` is a vector, the result is a vector with value 1.0 at points where the comparison is true and 0.0 where it is false. For example `Ch(1)>1` has the value `1.0` where channel 1 is greater than 1.0 and 0.0 elsewhere. `Ch(1)<>Ch(2)` is 1 wherever channels 1 and 2 are not the same. Be cautious using <> (not equals) and = (equals) as we are dealing with floating point numbers and exact equality may be compromised by arithmetic rounding.

Operator precedence

Multiply and divide have a higher precedence than all the other operators which all have the same precedence. You can use round brackets to force other evaluation orders. Apart from that, evaluation is from left to right, so `a>b*c+d` is interpreted as `(a>(b*c))+d`.

You can insert spaces between operators and numbers and between round brackets and the items within them. You may not insert spaces between a function name and the opening bracket that follows it.

Channel functions

The following functions take a channel of data (but not a virtual channel) and create a vector holding data at the sample interval and alignment set for the virtual channel.

`Ch(n)`	`n` is a waveform channel. Copy channel `n` data.
`If(n,g)`	`n` is a a marker channel to convert to a waveform by linear interpolation of the instantaneous frequency. `g` is the maximum gap to interpolate across, in seconds. Omit `g` or set it to 0 for no limit to the gap.
`Ifc(n,g)`	The same as `If()` except that cubic spline interpolation is used.

Marker kernel functions

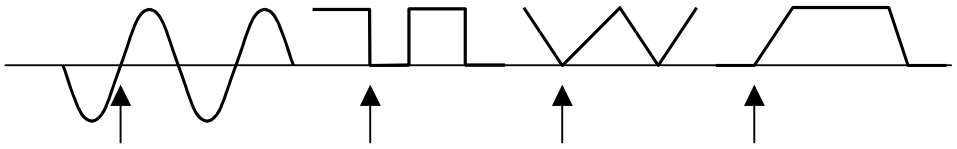
The following functions convert marker channel `n` to a waveform by replacing each marker by a kernel of unit area with a width set by `w` to the left and `r` to the right. You may omit `r`, in which case the kernels are symmetrical (`r` is set to `w`). You can set one (but not both) of `r` or `w` to 0 for a single sided kernel.

`Ec(n,w{,r})`	This vector expression converts the marker channel into a waveform by counting the number of markers from `–w` to `+r` seconds around each waveform point.
`Et(n,w{,r})`	This converts a marker channel into a waveform by weighting the markers from `–w` to `+r` seconds around each waveform point with a triangle function.
`Es(n,w{,r})`	This converts a a marker channel into a waveform by weighting the markers from `–w` to `+r` seconds around each waveform point with a sinusoid.
`Eg(n,w{,r})`	This converts a maker channel into a waveform by weighting the markers from `–w` to `+r` seconds around each waveform point with a Gaussian function with a sigma (standard deviation) of `w/4`.

`Ee(n,w{,r})`	This sets an exponential kernel. The time constant of the exponential is `w` to the left of each marker and `r` to the right. The kernel extends to 8 times the time constant in each direction.

Waveform generation

These functions generate waveforms without the need for an input channel, they all produce output from the start to to the end of the frame. All the arguments are in units of seconds, except the sine wave frequency, which is in Hz. All these waveforms have unit amplitude. You can use the standard maths operators `*/+` and `-` to scale and shift the output to different values.



Generated waveforms and their alignment points

If you attempt to create a cyclic waveform with a frequency above half the channel sample rate, no output will be generated. You can generate the following outputs:

`WLev(v)`	A constant level of value `v` running from the start of the frame up to the end.
`WSin(f, a)`	Sine wave of frequency `f` Hz running from the start to the end of the frame and aligned so that phase 0 (the point where the rising sinusoid crosses 0) is at time `a` seconds. The amplitude of the output runs from -1.0 to +1.0. The sinusoid is most accurate at the start of a frame; the accuracy falls off as the number of cycles increases (this is highly unlikely to be noticeable).
`WSqu(l,h,a)`	Square wave with low period `l` seconds and high period `h` seconds aligned so that a low period starts at time `a` seconds. Both `l` and `h` must be greater than 0 seconds. The output level of the low section is 0, the output level of the high section is 1.
`WTri(r,f,a)`	Triangle wave with a rise time of `r` seconds and a fall time of `f` seconds aligned so that a rise starts at time `a` seconds. Either of `r` or `f` may be zero, but not both. The triangle output level is from 0 to 1.
`WEnv(r,h,f,a)`	Envelope with a rise time of `r` seconds, a hold time of `h` seconds, a fall time of `f` seconds with the rise starting at time `a` seconds. The output waveform is 0 before time `a` and after time `a+r+h+f` and is 1 during the hold time. At least one of `r`, `h` or `f` must be non-zero.
`WPoly(f,t,r,L)`	Polynomial in time from `f` to `t` seconds and 0 before `f` and from `t`. The value at time t is a function of (t-`r`) where `r` is a reference time. `L` is a list of 1 to 6 coefficients. `WPoly(f,t,r,a,b,c,d,e,f)` is: y(t) = `a` + `b`*(t-`r`) + `c`*(t-`r`)^ 2 + `d`*(t-`r`)^ 3 + `e`*(t-`r`)^ 4 + `f`*(t-`r`)^ 5
`WT(s,e)`	Ramp of time starting at time `s` and running up to up to time `e`. The ramp value at time t is (t-`s`) with value 0 before `s` and from `e` onwards. Omit `e` to run to the end of the frame, omit both `e` and `s` to start from the beginning and run to the end.

Mathematical functions

The mathematical functions all have the form `Func(x)`, where `x` can be either a scalar or a vector and `Func()` is the operation. If x is a vector, the result is a vector with the function applied to each value otherwise the result is a scalar.

`Max(x,y)`	The maximum of x and y. This can be used to set a lower limit, for example `Max(Ch(1),1)` is a vector with minimum value 1.
`Min(x,y)`	The minimum of x and y. This can be used to set an upper limit, for example `Min(Ch(1),Ch(2))` can be thought of as the value of

	channel 1 with the maximum value limited by the value of channel 2 (or vice versa).
`Sqr(x)`	This calculates the square of `x`. `Sqr(x)` is the same as `x*x`, but is faster, particularly when x is a vector.
`Sqrt(x)`	This calculates the square root of `x`. When `x` is a vector, negative values are set to 0. If `x` is a scalar, negative values cause an error.
`Cub(x)`	This calculates the cube of `x`. `Cub(x)` is the same as `x*x*x` but is much faster, particularly when `x` is a vector.
`Poly(x,L)`	Replace `x` with a polynomial in `x` of order 1 to 5; `L` is a list of 1 to 6 coefficients. Use this to apply a non-linear calibration to a signal. For example: `Poly(x,a,b,c,d) = a + b*x + c*x*x + d*x*x*x`
`Sin(x)`	Calculates the sine of `x` (`x` is in radians).
`Cos(x)`	Calculates the cosine of x (`x` is in radians).
`Tan(x)`	Calculates the tangent of x (`x` is in radians). The result can be infinite.
`ATan(x)`	The arc tangent of `x`. The result is in radians in the range `-pi`/2 to `pi`/2.
`ATan(s,c)`	Both `s` and `c` must be vectors or both must be scalars. The result is the arc tangent of `s/c` with the quadrant set by the signs of `s` and `c`. The result is in the range `-pi` to `pi`.
`Ln(x)`	Natural logarithm of `x`. Negative or zero `x` values generate -100 when `x` is a vector and an error when `x` is a number.
`Exp(x)`	Exponential of `x`. If the result overflows, this is an error when `x` is a number and sets the largest allowed result when `x` is an array.

Channel process functions The channel process functions all have the form `Func(x)` where `x` can generally only be a vector and `Func()` is the operation. The result is a vector.

`Abs(x)`	The absolute value of `x` (negative values are replaced by positive values of the same size). This can also be used to generate the absolute value of a scalar.
`Hwr(x)`	Half wave rectify `x` (negative values are replaced by zeros). This function can also be used on a scalar quantity.
`Int(x)`	The integral of `x`, where the value at position `p` is replaced by the sum of all values from the start of the data frame up to and including p, multiplied by the sample interval. This gives the total area from the start of the data.
`Dif(x)`	The slope of x calculated by differences, where the value at position `p` is replaced by the difference between the current value and the preceding value divided by the sample interval.
`Smth3(x)`	A 3-point smoothing of `x`, where the value at position `p` is replaced by the average of the 3 points around that position.
`Smth5(x)`	A 5-point smoothing of `x`, where the value at position `p` is replaced by the average of the 5 points around that position.
`Smooth(x, tc)`	Smoothing of `x` with time constant `tc`. The value at position `p` is replaced by the mean of the data over the range `p-tc` to `p+tc`.
`DCRem(x, tc)`	DC offset removal of `x` with time constant `tc`. The value at position `p` is replaced by the value at `p` minus the mean of the data over the range `p-tc` to `p+tc`.
`Slope(x, tc)`	Slope measurement of `x` with time constant `tc`. The value at position `p` is replaced by the slope measured using least-squares fitting over the range `p-tc` to `p+tc`.

`RMSAmp(x, tc)`	RMS amplitude of `x` with time constant `tc`. The value at position `p` is replaced by the RMS value of the data over the range `p-tc` to `p+tc`.
`Median(x, tc)`	Median filter of `x` with time constant `tc`. The value at position `p` is replaced by the median value (the middle point after the data has been sorted into order) of the data over the range `p-tc` to `p+tc`.

Frame, tag and state functions

These functions all generate a scalar quantity based upon an aspect of the current frame, which makes is possible for the virtual channel behaviour to vary with the frame.

`Frm()`	The result of this function is the current frame number.
`Tag()`	The result of this function is 1 if the current frame is tagged, 0 if it is not.
`State()`	The result of this function is the state code number of the current frame.

Example expressions

If channels 1 to 3 hold waveform data, then `Ch(2) – 2*Ch(1)` displays the difference between channel 2 and twice channel 1.

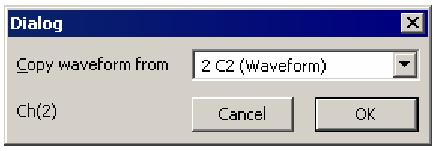
`Sqrt(Sqr(Ch(1))+Sqr(Ch(2)+Sqr(ch(3)))` displays the square root of the sum of squares of three channels. You could use this to display the magnitude of the resultant of three perpendicular forces or movements. `Sqr(Ch(1))` is the same as `Ch(1)*Ch(1)`, but `Sqr()` is faster. To generate a polynomial function of the input it is much quicker to use `Poly()` than to use `Ch()`, `Squ()`, `Cub()` and so on to generate a power series.

Build expression

There is no need to remember all the expression functions; just click the >> button and choose from a list of possible items to add. You can choose from:

Waveform from channel

Select one of these items to generate the commands that create a waveform from an existing channel. You build the commands using a dialog. All dialogs display the equivalent command in their lower left corner. The result replaces the selection in the Expression field.

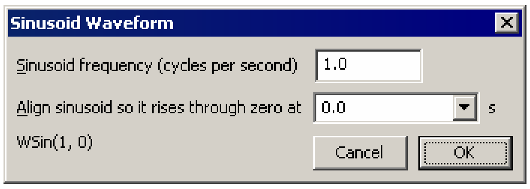

Dialog

Copy waveform from: 2 C2 (Waveform)

Ch(2)

Generate waveform

Select one of these items to generate the commands that create a channel based on a level, sine, square or triangle wave or on a waveform envelope or based on linear time or on a polynomial of time. You build the commands using a dialog, the dialogs vary according to the waveform that is being generated. All dialogs display the equivalent command in their lower left corner. The result replaces the selection in the Expression field.


Sinusoid Waveform

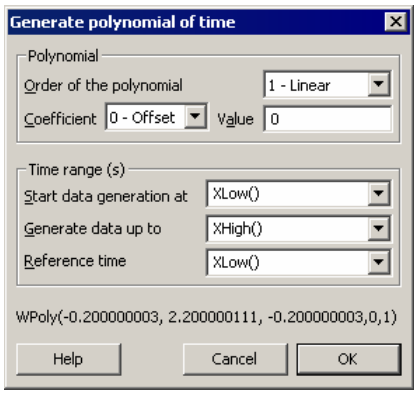
Sinusoid frequency (cycles per second): 1.0

Align sinusoid so it rises through zero at: 0.0 s

WSin(1, 0)

The `WPoly()` command is more complex, and can be used to generate polynomials in time relative to a reference time. Such curves can be used to create complex envelopes, or to subtract out a curve generated by the interactive curve fitting routines.

The `WT()` command generates a ramp from a start time up to, but not including an end time. The data is zero outside this time range. Within the time range, the ramp value is the current time minus the start time.


Generate polynomial of time

Polynomial

Order of the polynomial: 1 - Linear

Coefficient: 0 - Offset   Value: 0

Time range (s)

Start data generation at: XLow()

Generate data up to: XHigh()

Reference time: XLow()

WPoly(-0.200000003, 2.200000111, -0.200000003,0,1)

Channel process functions

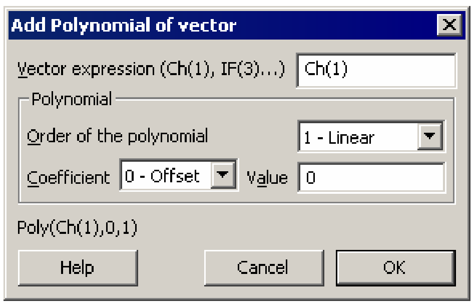
Channel process functions are generally used to modify existing channel data in some useful fashion. For example, if the Expression field holds `Ch(1) + Ch(2)` and you want to rectify the channel 1 data before adding it, select `Ch(1)`, click the >> button, select Channel process functions, then select Rectify.

Mathematical functions

The commands in this section apply a nominated mathematical function to the selection in the Expression field, which will usually be vector expressions (but do not have to be). For example, if the Expression field holds `Ch(1) + Ch(2)` and you want to rectify the channel 1 data before adding it, select `Ch(1)`, click the >> button, select Channel process functions, then select Rectify.

Mathematical functions: Poly()

Select this to generate a polynomial. The *Vector expression* field is set to whatever was selected when this dialog was opened, or is set to `Ch(1)` if nothing is selected. This field is not tested for validity. Each element `x` of the vector is replaced by a polynomial in `x`. You can set the order of the polynomial (order means the highest power of x used) in the range 1 to 5. The command is:



Vector expression (Ch(1), IF(3)...) Ch(1)
Order of the polynomial 1 - Linear
Coefficient 0 - Offset Value 0
Poly(Ch(1),0,1)

```
Poly(x, a0, a1, a2, a3, a4, a5)
```

where *x* is the vector expression (you can also use a scalar, but this is not very useful) and the `a0` to `a5` are the coefficients of the polynomial. This expression generates the quintic:

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5$$

For lower order polynomials, omit the coefficients from the right. For example, for a quartic (fourth order polynomial), omit `a5`, for a cubic omit `a5` and `a4`. To set coefficient values in the dialog, use the Coefficient field to select the required coefficient and then set its value.

Frame, tag and state functions

The commands in this section insert a function that returns a frame-dependent scalar value, for example the frame number or frame state code.

Mathematical operators

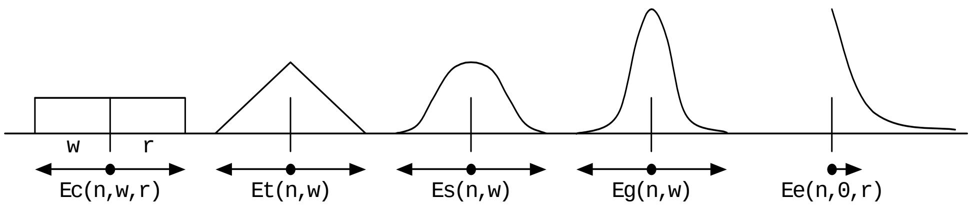
These commands replace the selection with `+`,`-`, `*` and `/` to remind you that you can use these operators to add, subtract, multiply and divide vectors and scalars.

Previous virtual channel expressions

This option lists expressions that you have used previously. Valid expressions are added to the list when you click the Save expression button, when you change to a different virtual channel and when you close the dialog with the Close button. The expressions are stored in the system registry. The most recently used expression is at the top of the list.

The Marker to waveform functions

The `Ec(n,w,r)`, `Et(n,w,r)`, `Es(n,w,r)`, `Eg(n,w,r)` and `Ee(n,w,r)` functions convert marker channel `n` into a virtual waveform and can be used anywhere in an expression that you can use the `Ch()` function. They replace each marker by a kernel (shape), centered on the marker time. For a marker at time `t`, the kernel extends from `t-w` to `t+r` seconds except for `Ee()`, which extends from `t-8w` to `t+8w` seconds. Normally you will omit `r`, in which case the shapes are symmetrical with r set equal to `w`. The resulting waveform is the sum of the kernels for all the markers. The area of each kernel is unity, so the area under the waveform between any two times is the number of markers within that time interval.



The `Ec()` function simply counts the markers in the time range. This is the fastest analysis method, but produces the most jagged output. The `Et()` function weights the markers with a triangle function. This is slower than `Ec()`, but much faster than `Es()` and `Eg()`. The `Es()` function weights the markers with a raised cosine. The `Eg()` function weights the markers with a Gaussian curve extending to 4 sigmas.

The `Ee()` function is rather different from the others as it is not suitable for use as a smoothing function and is more likely to be used in a single-sided form. It weights the markers with an exponent `exp(-t/r)` to the right and `exp(-t/w)` to the left (t stands for the time difference between the point and the time). The exponent extends to 8`w` to the left and to 8`r` to the right.

Append frame

This command appends a new, blank, frame to the end of a file or memory view. This command can be used on a file view to generate an extra frame that will be used to hold processed data; for example a frame containing leak subtraction data that will be subtracted from other frames in the file. The extra frame can be used as part of a script, or it can be manipulated via the frame buffer.

If the command is used to append a frame to a memory view created by processing, the new frame will have its own processing parameters. When the frame is appended, the process dialog is provided to define the new frame’s processing parameters. This allows for result views with different frames holding the results of processing different sets of source frames. For example, if you were sampling with multiple states, you might want to produce a multiple frame average with each frame holding the results of averaging source frames with a different state. Multiple-frame memory views, with attached processing parameters, can be saved as part of a Signal sampling configuration.

Append frame copy

This command appends a new frame, containing a copy of the data in the current frame, to the end of a file or memory view. This command is not available for memory views created by processing.

Delete frame

This command removes the current frame from the file or memory view. It is only available if the current frame has been appended and not yet written to disk. The last frame in a memory view and file view frames stored on disk may not be deleted.

Delete channel

This command may be used to remove a channel permanently from an XY data document or an idealised trace from a file view. Once the channel has been deleted, it cannot be retrieved.

The frame buffer

The frame buffer is an extra frame of data that is automatically provided by Signal. Every open data file and memory document has a separate frame buffer that is used in conjunction with the document data, this buffer is shared by all of the views of that document. The frame buffer can be used to carry out arithmetic on frames (for example, subtract the average of frame 1 to 4 from all frames), either interactively via the commands described below or by using the multiple frames dialog, also described below.

The way to think of the frame buffer is as an extra frame of data that is behind the current frame in the view. When you change to a different current frame, the buffer moves too so that it is always associated with the current frame. Understanding this association is important because all of the frame buffer arithmetic commands described below work with the current frame. If you are displaying the frame buffer the buffer moves so that it is in front of the current frame, but it is still closely associated with the current frame. When the buffer is shown the view title changes to show that the buffer is visible, the current frame number is still shown in brackets because the user still needs to be aware of which frame is current in order to use the buffer.

Clear buffer

This command (keyboard shortcut `Ctrl+0`) clears the data in all channels of the frame buffer to zero. This is the initial state of the buffer after a CFS data file has been loaded.

Copy to buffer

This command (keyboard shortcut `Ins`) copies the data in the current frame into the frame buffer.

Copy from buffer

This command (keyboard shortcut `Ctrl+Ins`) copies the data in the frame buffer, and any count of sweeps averaged, to the current data frame.

Exchange buffer

This command (keyboard shortcut `Shift+Ins`) exchanges the data in the frame buffer, and any count of sweeps averaged, with the data in the current frame.

Add to buffer

This command (keyboard shortcut `+`) adds the data in the current frame to the data in the frame buffer. There is an alternative form of this command, available only through the `Ctrl++` shortcut and the Multiple frames dialog, which adds the buffer data to the data in the current frame.

Subtract buffer

This command (keyboard shortcut `-`) subtracts the data in the frame buffer from the current frame. There is an alternative form of this command, available only through the `Ctrl+-` shortcut and the Multiple frames dialog, which subtracts the current frame data from the buffer.

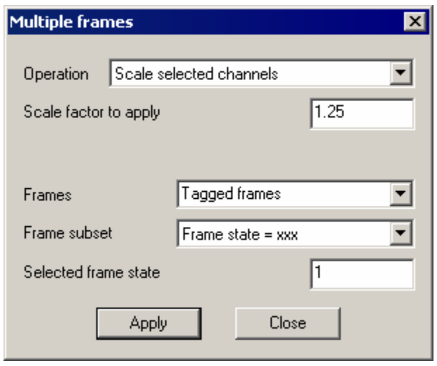
Average into buffer

This command (keyboard shortcut `Enter`) adds the data in the current frame into an average accumulating in the frame buffer. The addition is carried out in such a way that the buffer holds the average of frames accumulated. If the buffer contains data before averaging starts, this will be included as the first frame of the average. There is an alternative form of this command, available only through the `Ctrl+Enter` shortcut and the Multiple frames dialog, which removes the current frame data from an average in the buffer - this will not work correctly if the frame was not accumulated into the buffer average in the first place.

If you mix the buffer averaging commands with the normal addition and subtraction operations, you will find that normal addition and subtraction on the buffer ‘resets’ the average by setting the count of sweeps so far to one. The actual addition and subtraction act as you would expect.

Multiple frames

This command (keyboard shortcut `Ctrl+M`) provides a dialog that can be used to carry out numerous operations on multiple frames in the document. The dialog contains a selector for the operation to be carried out, a field to enter any operation data required (this is hidden if the operation does not need it) plus a standard set of controls to specify frames in the data document to be used. The dialog can be used repeatedly by pressing the Apply button, it doesn’t disappear until Close is clicked.


Multiple frames

Operation	Scale selected channels
Scale factor to apply	1.25
Frames	Tagged frames
Frame subset	Frame state = xxx
Selected frame state	1

The operations available from the dialog include all of the frame buffer operations, all of the channel data modification options, plus tag and un-tagging frames and clearing any fits.

For operations that modify the frame buffer, such as accumulating an average or summing frames, the effect is straightforward. For operations that modify file view data, such as rectifying channels or subtracting buffer data from frames, the changed data must be saved to disk if the action is to have an effect (otherwise the changed file data will be discarded). This is because Signal only holds one frame from a data file in memory at a time; frames are loaded from disk as required and discarded when another frame is wanted. Use the File menu Data update mode option to ensure that changed frame data is saved, either unconditionally or by querying the user. For memory views, all of the document data is held in memory and changes to frame data are always saved.

Modify channels

This command provides a pop-up menu specifying the data modifications that are available. All the modifications operate on the selected waveform channels or on all visible waveform channels if none are selected. If the frame buffer is being shown then they operate on frame buffer data. In either case, all data points in the channels are modified. Most of the modifications are also available via the keyboard shortcuts shown in the menu. As for the frame buffer operations, changes to the frame data will be saved, or not, according to the file data update mode.

The behaviour of the modifications themselves are mostly straightforward. Subtract DC measures the mean value of the channel data, then subtracts this DC offset value from all data points. Normally, the DC level is measured over the visible frame area, the X range for the DC measurement can be set using the Area of DC item. Differentiation replaces each data point with the difference between that point and the previous point and divides the result by the sample interval; the first data point is set to zero. Integration replaces each data point with the sum of all data points up to and including that point multiplied by the sample interval. Neither integration nor differentiation are available for log-binned data. 3-point and 5-point smoothing replace each point with the average of the 3 or 5 points centred on that point. The scale and offset data options provide a dialog in which a numeric value can be entered. Scaling the data multiplies each data point by the number entered, offsetting adds the number entered to each data point.

The shift data option rotates data points by shifting data from one time to another within the frame. Note that this operation rotates; data points that fall off one side of the frame are shifted back in on the other side, so the operation can be reversed without loss of data. There are special shortcuts `Shift+<` and `Shift+>` to shift left and right by one point.

The last four options are for inter-channel arithmetic. A channel will be prompted for and this channel will be applied with the appropriate operand to the other channels on a point by point basis.

Tag frame

This command (keyboard shortcut `Ctrl+T`) is used to tag or untag the current frame in the current view. When the current frame is tagged, this menu item is shown checked. All data frames in files handled by Signal can be tagged or untagged, the tagged status of a frame is displayed as part of the application status bar and can be interrogated by scripts. Frame tagging can be used for any purpose you require; all commands requiring a frame selection are able to operate on all tagged frames or all untagged frames. The command toggles the tag state of the current frame, changing tagged frames to untagged and vice-versa.

Digital filters

This option provides the FIR and IIR digital filtering dialogs, which can create filters and apply them to waveform channels. See the *Digital filtering* chapter for details of these. This option will be greyed out for log-binned data

Keyboard analysis control

Windows software is usually orientated towards control by means of the mouse and menus, but it is often convenient to use the keyboard instead. For interactive analysis of the data, using the keyboard can often be much faster. With this in mind, Signal includes keyboard shortcuts designed to handle most common data manipulation requirements:

Channel arithmetic	Key	Frame buffer operations	Key
Zero channels	`Shift+Z`	Toggle display of frame buffer	`Ctrl+B`
Negate data	`Shift+N`	Add frame to buffer (average)	`Enter`
Rectify data	`Shift+R`	Add frame to buffer	`+`
Subtract DC level	`Shift+O`	Add buffer data to frame	`Ctrl++`
Differentiate data	`Shift+D`	Subtract frame from buffer (average)	`Ctrl+Enter`
Integrate data	`Shift+I`	Subtract buffer data from frame	`-`
3-point smooth	`Shift+3`	Subtract frame from buffer	`Ctrl+-`
5-point smooth	`Shift+5`	Copy frame data to buffer	`Insert`
Shift 1 point left	`Shift+<`	Copy buffer data to frame	`Ctrl+Ins`
Shift 1 point right	`Shift+>`	Exchange buffer and frame data	`Shift+Ins`
		Multiple frames dialog	`Ctrl+M`

All of these shortcuts are documented with the appropriate menu commands. All analysis shortcuts are listed here for convenience. There are more keyboard shortcuts for display and text view manipulation; see the *General information* chapter.

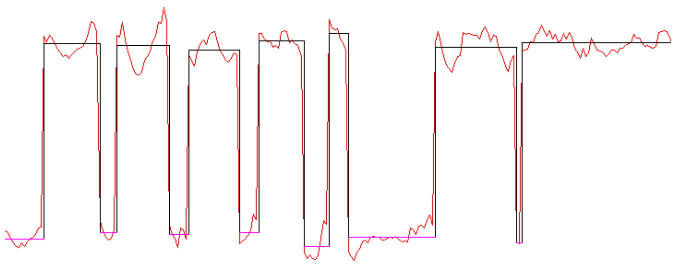
13 Single-channel analysis

Open/Closed times

The Analysis menu for file and memory views contains a separate section, Open/Closed times, that is dedicated to facilities used by patch clamp researchers to analyse single channel data. To avoid confusion among non-clampers, these features will be hidden unless clamping support is enabled in the Edit menu Preferences dialog. With clamping support enabled, the Open/Closed times item is visible in the analysis menu and the available commands for analysis and interaction will be displayed in an attached sub-menu.

The currently available single-channel analysis commands are New idealised trace (SCAN), New idealised trace (Threshold), which generate idealised trace data from a waveform channel and Open/Closed time histogram, Open/Closed amplitude histogram and Burst duration histogram, which process previously generated idealised trace data. A final command; View and modify event details, provides the Event details dialog used for interactive generation and editing of idealised traces.

About idealised traces



An idealised trace represents the opening and closing (assumed to be instantaneous) of an ion channel in a membrane. It shows the noise-free underlying current through the membrane that, processed though imperfect transducers and amplifiers, would generate the distinctly non-ideal signal that is actually sampled. Idealised trace data is generated by fitting the trace to the sampled data in such a manner as to most accurately represent the behaviour of the actual ion channel, taking into account such things as the sampling rate, the average level of sections of the data and the frequency response of the signal amplifiers.

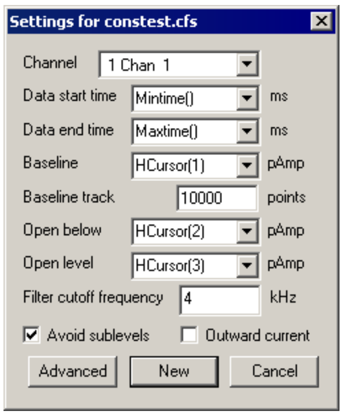
Idealised trace data is very different, in a number of ways, from other data handled by Signal. Whereas normal waveform data consists of a series of measurements equally spaced through time, an idealised trace is a sequence of levels, called events, of arbitrary duration, each running from the end of the previous level up until the start of the next one. Each event has a start time, an amplitude, a duration and a set of flags to identify the type of event: a closed time; first latency etc. Because this is rather different from a sampled waveform many analyses such as waveform averaging and the cursor regions measurements are not available for idealised trace data. On the other hand other analyses such as open\closed time histograms are possible with idealised traces.

An idealised trace must be generated from a waveform before histograms based upon the trace data can be built. If you set up histograms based on an idealised trace channel where the trace has not yet been generated the histograms will still be created, but will not contain any data. If you are working on-line, it is worth noting that idealised trace data will be generated first for newly sampled data so that other analyses that depend on it will work correctly. Once an idealised trace exists, it is stored in the `.sgr` file associated with the data file and will be re-loaded each time the data file is opened.

]

New idealised trace (SCAN)

This analysis generates idealised trace data from a waveform using the SCAN technique, the settings dialog provided when you use this command holds fields to determine how the trace should be generated.


Settings for constest.cfs

Channel	1 Chan 1	
Data start time	Mintime()	ms
Data end time	Maxtime()	ms
Baseline	HCursor(1)	pAmp
Baseline track	10000	points
Open below	HCursor(2)	pAmp
Open level	HCursor(3)	pAmp
Filter cutoff frequency	4	kHz

[✓] Avoid sublevels [ ] Outward current

Note that before you can use the SCAN method you must first perform the Baseline measurements option available in the same sub-menu.

SCAN analysis makes use of an assumption that a Gaussian filter was used to remove noise from the signal to produce a high time resolution guess of what the original unfiltered, noise free waveform was. This is a form of reverse convolution of the idealised trace using the step response function for the amplifier and other electronics. It is worth noting that the usual multiple poles Bessel filter used by most patch clamp amplifiers is a good approximation to a Gaussian filter so produces quite satisfactory results with this technique. The default draw mode for the analysis is to show the convolution over the top of the raw data with the idealised trace drawn below. Once this process has been used to produce a rough idealised trace, the trace will need to be fitted to the data using the Event details dialog to achieve maximum accuracy.

The Channel field sets the waveform channel to be fitted. The time range to be fitted is then defined by the Data start time and Data end time fields. If you are analysing voltage activated channels then you should set these times to be the start and end times of the stimulus. The first event generated by the analysis will be flagged as a first latency so if you are interested in first latency times you will need to make sure that the first event starts when the stimulus does. You should note however, that the transition times calculated will be skewed by the effects of the filter. This does not affect the durations of most of the events as they are all skewed by the same amount but first latencies will be slightly extended. If you are analysing a spontaneously active channel you should set the end time to be XHigh() as you will need to process small chunks of data at a time to keep a check on progress through the file. More details of how to do this are given at the end of this chapter.

The Baseline field is used at the start of the analysis to tell the event detection where the closed state is expected to fall. Using a horizontal cursor here will allow rhe cursor to be moved by the analysis routines to update their positions as the baseline is tracked. Baseline track will keep a running average of points in the closed state in order to correct for baseline drift during the recording.

The next field will be either Open below for an inward current where an opening is downwards on the display or Open above for an Outward current where an opening is upwards on the display. This field should be set to just outside the noise level. It is used to determine when an event is too short to be distinguished from noise.

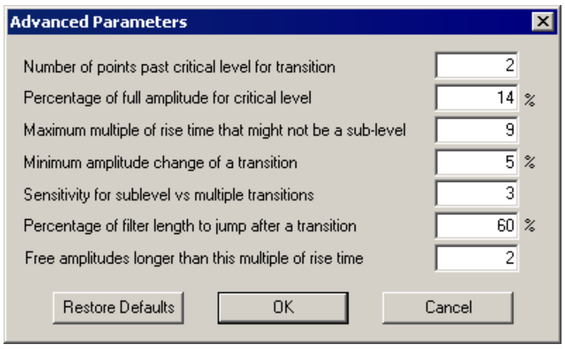
Open Level defines the full open level. It is used in conjunction with the Advanced parameters to determine what amplitude change is significant enough to constitute a transition.

The Filter cut-off frequency is the –3 dB frequency of the Gaussian filter. If an analogue Bessel filter is used it is worth noting that the –3 dB frequency is often about half the cut-off frequency set on the front panel of the filter which uses a different definition for the cut-off.

Sometimes a transition takes longer than expected for a given filter cut-off. When this happens Signal can either insert an event at a sub-level or insert two full height transitions in opposite directions in order to make the resulting convolution approximate to the raw data. With the Avoid sublevels checkbox checked then the latter option will be used whenever appropriate though it does not guarantee that no sub-levels will be fitted.

The New button generates the channel to hold the idealised trace and provides the standard process dialog where you can control the processing. The idealised trace data channel is originally positioned in the file view on top of the original waveform channel, the drawing is set up so that the convoluted trace is shown on top of the waveform, with the idealised trace itself offset below.

Advanced


Advanced Parameters

Number of points past critical level for transition	2
Percentage of full amplitude for critical level	14 %
Maximum multiple of rise time that might not be a sub-level	9
Minimum amplitude change of a transition	5 %
Sensitivity for sublevel vs multiple transitions	3
Percentage of filter length to jump after a transition	60 %
Free amplitudes longer than this multiple of rise time	2

The algorithm used by the SCAN technique is highly complex. The Advanced button allows various parameters used in the trace formation to be changed. The basic principle of the trace formation is that an average is kept of the point amplitudes of the data. When a number of consecutive points fall outside a critical level then a transition is deemed to have taken place. The critical level is defined as a percentage of the difference between the baseline and the full open level. To begin scanning for the next transition imediately would probably result in another transition being detected straight away when in fact it was just part of the same transition already found. For this reason the scanning jumps ahead by an amount to get past the transition just found. This amount is specified as a percentage of the filter length; the filter length being defined as the time taken for the step response to get from 1% to 99% of the step amplitude. For a Gaussian filter this turns out to be $0.6165062/f_c$ where f_c is the –3 dB filter cut-off frequency. Rise time is defined as $0.3321412/f_c$.

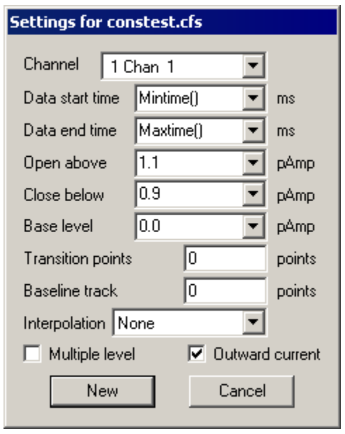
An average of the data point values found while looking for a transition is kept and used as the default amplitude for an event. If this amplitude is less than the critical level from the baseline then the event is flagged as closed. The data skipped over by the transition detection is checked for turning points that might indicate that a transition in the oposite direction to the previously detected one had been missed. If a turning point is found, a transition is inserted at a time calculated from the amplitude of the turning point and an assumption that the original data reached full amplitude or was closed.

Consecutive transitions in the same directions can either mean a sub-conductance level or a pair of transitions have been missed. If the Avoid sublevels box has been checked then by looking for turning points in the first derivative of the raw data the times of transitions can be deduced or their presence eliminated. Any missed events are assumed to be of full amplitude or closed and are flagged as assumed amplitude.

Transitions having less than a certain amplitude are stripped out in a final pass. At the fitting stage in the Event details dialog amplitudes having the assumed amplitude flag set are held fixed then a second pass of fit is made freeing up assumed amplitudes above a certain duration. The Advanced parameters dialog can be accessed from the Event details dialog by clicking the Parameters button.

New idealised trace (Threshold)

This analysis generates idealised trace data from a waveform using a threshold crossing technique. Threshold crossing as a method for generating an idealised trace is needed if you have more than one channel in your patch or if you want to make use of a simpler technique and are not concerned with a very high time resolution result.



Settings for constest.cfs

Channel	1 Chan 1	
Data start time	Mintime()	ms
Data end time	Maxtime()	ms
Open above	1.1	pAmp
Close below	0.9	pAmp
Base level	0.0	pAmp
Transition points	0	points
Baseline track	0	points
Interpolation	None	

☐ Multiple level ☑ Outward current

The first three fields in the setting dialog set the channel to be analysed and the time range to analyse in exactly the same way as for the SCAN method settings.

The next two fields set the thresholds to use for detecting openings. These are labelled Open above and Close below for outward currents or Open below and Close above for inward currents. Having two thresholds in this way provides some protection against false events being detected, which are actually noise. It is assumed that these thresholds will be positions approximately around the half-way mark between the baseline level and the open level, so if the open level was approximately 1.8 pA then the thresholds would be placed around 0.9 pA; at 0.8 and 1.0 pA for example.

The Base level field is mostly used in multiple level analysis, where the thresholds for subsequent levels are calculated by averaging the two thresholds and then doubling the difference between this average and the base level. The result is assumed to be the current per channel opening for subsequent levels and so subsequent thresholds are calculated by adding multiples of this value to the original thresholds. This value is also used to provide the reference level for event amplitude measurements when baseline tracking is not in use.

Transition points can be set to a number greater than zero to provide a 'dead time' after a transition before another transition can be detected. This allows for settling of data which has been filtered.

The Baseline track field sets the number of points over which the baseline level is averaged to produce a current baseline level. The current baseline level is used to set the level for closed events, plus all thresholds are shifted to match shifts in the current baseline level. Set this field to zero to disable baseline tracking, in which case the level of a closed event is set by the average value of the waveform making up this event.

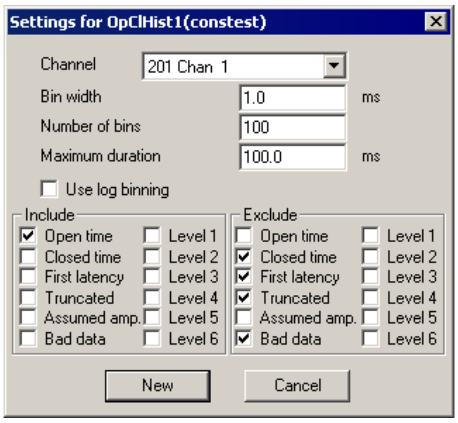
By default an event will start at the first data point found to be across a threshold and will have an amplitude which is the average amplitude of all the data point within the event period. The Interpolation field allows the start time of the event to be calculated by extrapolating the data around the transition to find the exact time that the threshold was crossed. Currently only linear interpolation is available, this assumes that a straight line can represent what happens between sample points.

Select Multiple level if you have more than one channel in your patch or if other circumstances give your preparation more than one open level. Outward current means that a channel opening produces a more positive current; a commonly used convention. An inward current would produce a more negative current when a channel opens.

The New button generates the channel to hold the idealised trace and provides the standard process dialog where you can control the processing. The idealised trace data are originally positioned in the file view on top of the original waveform.

Open/Closed time histogram

This analysis generates a histogram showing the relative frequency of events of various durations, the flags associated with trace events are used to select which events are included in the analysis. The command opens a settings dialog which defines the analysis and histogram parameters.



Settings for OpClHist1(constest)

Channel: 201 Chan 1
Bin width: 1.0 ms
Number of bins: 100
Maximum duration: 100.0 ms
[ ] Use log binning

Include: [✓ Open time] [ ] Closed time [ ] First latency [ ] Truncated [ ] Assumed amp. [ ] Bad data [ ] Level 1 [ ] Level 2 [ ] Level 3 [ ] Level 4 [ ] Level 5 [ ] Level 6

Exclude: [ ] Open time [✓ Closed time] [✓ First latency] [✓ Truncated] [ ] Assumed amp. [✓ Bad data] [ ] Level 1 [ ] Level 2 [ ] Level 3 [ ] Level 4 [ ] Level 5 [ ] Level 6

The Channel field selects the idealised trace channel that will be analysed. The Bin width, Number of bins and Maximum duration fields define the histogram that will be created; the x-axis of the result starts at zero (unless log binning is used) and the Bin width and Number of bins fields are linked so as to remain consistent with Maximum duration.

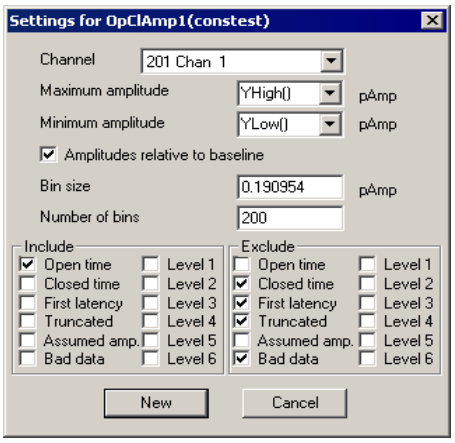
If the check box marked Use log binning is checked then the Bin width field changes to Minimum duration. Log binning means that the width of each bin increases geometrically so that when the resulting histogram is drawn with a logarithmic x-axis, the bins occupy a constant number of pixels across the histogram. The contents of each bin are divided by the bin width to maintain the overall shape of the histogram.

The two sections labelled Include and Exclude represent the flags associated with each event and are used to control which events are included in the histogram and which are not. An event will be included in the histogram if at least one flag set for the event matches the Include set specified and none of the event flags match the Exclude set. The flags are:

Open time	An event where the channel is open.
Closed time	An event where the channel is closed.
First latency	The first idealised trace event in the frame.
Truncated	The last idealised trace event in the frame.
Assumed amp.	An event where the amplitude is not an average of the raw data points.
Bad data	Event flagged as not suitable for analysis.
Level n	Six flags, one for each level of multiple level data. The Level 1 flag is set for all closed times as well as the first open level.

Open/Closed amplitude histogram

This analysis generates a histogram showing the relative frequency of occurrence of events with specific levels, the flags associated with trace events select which events are included in the analysis. This analysis is almost identical in concept to the amplitude histogram for waveform data described previously. The differences are that the y-axis is a count of trace events at a level rather than the time spent at that level and that the analysis can select events for analysis using the Include and Exclude flags for the analysis in the same manner as the Open/Closed time histogram.



Settings for OpClAmp1(constest)

Channel	201 Chan 1	
Maximum amplitude	YHigh()	pAmp
Minimum amplitude	YLow()	pAmp
[✓ Amplitudes relative to baseline]		
Bin size	0.190954	pAmp
Number of bins	200	

Include		Exclude	
[✓ Open time]	Level 1	Open time	Level 1
Closed time	Level 2	[✓ Closed time]	Level 2
First latency	Level 3	[✓ First latency]	Level 3
Truncated	Level 4	[✓ Truncated]	Level 4
Assumed amp.	Level 5	Assumed amp.	Level 5
Bad data	Level 6	[✓ Bad data]	Level 6

The Channel field in the settings dialog selects the idealised trace that will be analysed.

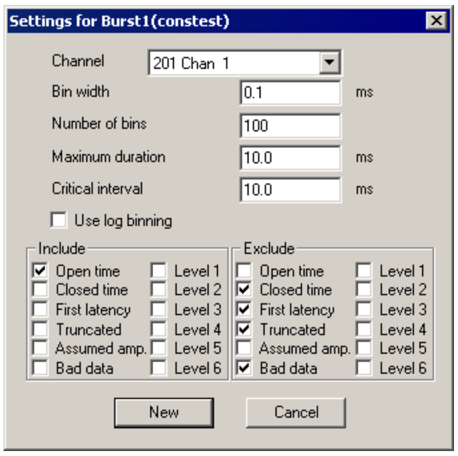
The Maximum amplitude and Minimum amplitude fields set the amplitude range that will be divided into bins and thereby set the x axis range in the memory view holding the histogram. The Bin size and Number of bins fields both set the number of bins that cover the amplitude range, whenever you change one of these items the other one will change to match.

By default, the amplitudes added to the histogram will be absolute amplitudes as measured by Signal. The amplitudes can be measured relative to the baseline by checking Amplitudes relative to baseline.

The Include and Exclude sections control which events are included in the analysis in exactly the same manner as for the Open/closed time histogram analysis.

Burst duration histogram

This analysis generates a histogram showing the relative frequency of occurrence of event bursts of different durations, the flags associated with trace events are used to select which events are part of a burst and which events can terminate a burst. The command opens a settings dialog which defines the analysis and histogram parameters.



Settings for Burst1(constest)

Channel	201 Chan 1	
Bin width	0.1	ms
Number of bins	100	
Maximum duration	10.0	ms
Critical interval	10.0	ms
Use log binning		

Include		Exclude	
[✓ Open time]	Level 1	Open time	Level 1
Closed time	Level 2	[✓ Closed time]	Level 2
First latency	Level 3	[✓ First latency]	Level 3
Truncated	Level 4	[✓ Truncated]	Level 4
Assumed amp.	Level 5	Assumed amp.	Level 5
Bad data	Level 6	[✓ Bad data]	Level 6

The Channel field selects the idealised trace channel to be analysed. The Bin width, Number of bins and Maximum duration fields define the histogram that will be created; the x-axis of the result starts at zero (unless log binning is used) and the Bin width and Number of bins fields are linked so as to remain consistent with Maximum duration.

If the check box marked Use log binning is checked then Bin width changes the Minimum duration. Log binning means that the width of each bin increases geometrically so that when the resulting histogram is drawn with a log x-axis, the bins appear to occupy a constant number of pixels across the histogram. The contents of each bin are divided by the bin width to maintain the overall shape of the histogram as if it were binned normally.

The Critical interval field defines how an event burst can be terminated; a burst is terminated by a non-included event whose duration is longer than the critical interval.

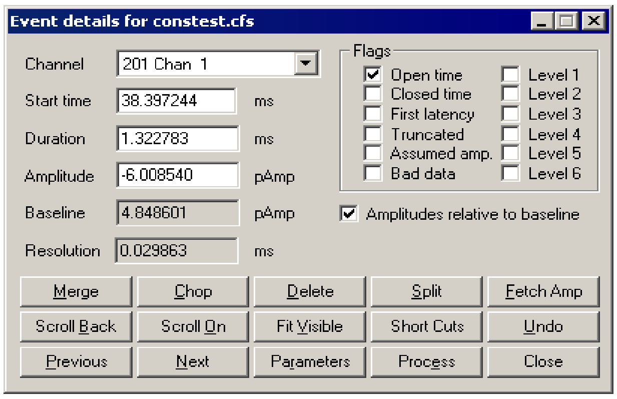
The Include and Exclude sections control which events can start or terminate a burst. An event can start a burst if at least one flag set for the event matches the Include set specified and none of the event flags match the Exclude set. All events that cannot start a burst are capable of terminating it. A burst begins with any suitable event and is terminated by any suitable event whose duration is greater than the critical interval. The burst duration runs from the start of the first event in the burst to the start of the event that terminated it.

Baseline measurements

This option must be used before any SCAN analysis can be carried out. A dialog is presented asking for a time range over which the baseline is to be measured and the channel on which to measure it. Clicking OK produces a message window with the mean data point value in the time range; the standard deviation of the measured values and the standard deviation of the first derivative of the measured values. This information is also be written to the log window and saved internally for use by the SCAN analysis.

View and modify event details

The View and edit event details command is available when the current view contains idealised trace data. It opens a dialog that allows information about individual events to be viewed and changed. Click on a horizontal portion of the idealised trace to select an event or on a vertical section to select the following event. You can also click and drag the idealised trace to change transition times and levels. The event whose details are displayed in the dialog will have a small box drawn at both ends to indicate that it is the current event.


Event details for constest.cfs

Channel	201 Chan 1	
Start time	38.397244	ms
Duration	1.322783	ms
Amplitude	-6.008540	pAmp
Baseline	4.848601	pAmp
Resolution	0.029863	ms

Flags: [✓ Open time] Closed time, First latency, Truncated, Assumed amp., Bad data, Level 1, Level 2, Level 3, Level 4, Level 5, Level 6

[✓ Amplitudes relative to baseline]

The event details dialog shows information about the current event. The Start time, Duration and Amplitude items and all of the Flags can all be changed manually. Any start time you enter must be between the start times of the neighbours. Changing the start time adjusts the event duration so that the end time remains unchanged. The duration must be a positive value and cannot extend the event beyond the end time of the following event. The baseline level, calculated by averaging points in the closed state, is also displayed. The Resolution item will be shown if the idealised trace has a SCAN process attached to it. It shows the minimum duration of full amplitude opening needed for the data trace to reach the trigger level. Each event also has a number of flags associated with it and these may be set or unset using this dialog. These flags and their uses are explained in the documentation of Open/Closed time histogram generation in this chapter. The buttons below the event details provide many useful functions:

Merge Merge combines the current event with the one to the right to produce a single event with the combined duration of the two events but the attributes of the first.

Chop Chop will break the current event in two. If the current event has an amplitude between those before and after then the first and second new events created by the break will be given amplitudes and attributes from the following and preceding events respectively.

Delete The Delete button will delete both the current and following event and extend the preceding event to cover the time gap created. This event will have an amplitude adjusted to the weighted average of all the events previously covering the time period.

Split Split will divide the current event into three separate events all having the same amplitude and duration.

Fetch Amp Fetch Amp will scan backwards through the trace to find an event of the same type and copy the amplitude of this previous event to the current event.

Scroll On and Scroll Back Scroll On and Scroll Back will do the same thing as Previous and Next but will do so using a smoothly scrolling display. Repeatedly hitting the scroll buttons will double the scroll speed each time. To stop the scrolling either click on the Next, Previous or the other scroll button or simply click on the data window.

Fit Visible The Fit Visible button will adjust the time and amplitude of the transitions using the filter cut-off frequency defined during the SCAN process to build a step response function to fit. Idealised traces created using threshold crossings will be fitted using the Nyquist frequency as the cut-off frequency unless this frequency is changed in the Channel information dialog. Amplitudes having the assumed amplitude flag set are held fixed then a second pass of fit is made freeing up assumed amplitudes above a certain duration. This duration is defined in the Advanced Parameters dialog, which can be accessed by clicking the Parameters button. If you do Fit Visible when the end of the display is beyond the least event of the idealised trace then a process will be done to extend the idealised trace before a fit is done. After fitting either the first event to fail to be fitted will be made the current event or the last displayed event will be.

Undo Press this button to undo the last operation. Up to 100 operations can be undone.

Previous and Next The Previous and Next buttons step the current selection of event through the idealised trace forwards and backwards respectively. Stepping past the end of the idealised trace will cause the display to jump on to the next possible opening ready for processing. If this is done there is then a short "dead" period during which the Next or Scroll On keys do nothing after that pressing either Next or Scroll On will extend any closed time terminating the idealised trace to include the displayed data.

Parameters The parameters button displays the advanced parameters dialog for SCAN processing. Though the button and dialog are still available for threshold analysis, the settings do not affect the processing.

Process This button causes the process for generating the idealised trace to be called. If the process settings are such as to process `MinTime()` to `XHigh()` then the trace will be extended from the current end time to the end of the displayed data.

View event list

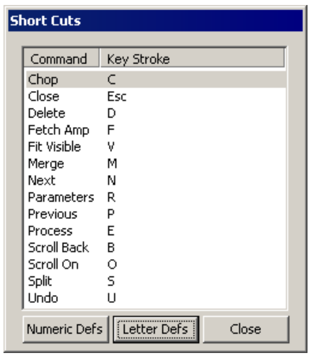
The View event list command opens a new window showing a list of idealised trace event durations and amplitudes. Up to 10 events are shown (drag the right edge of the window to change the number of events visible) with the currently selected event on the trace being shown highlighted in the list. Information for events in the closed state is shown using the colour set for drawing such events. You can click on an event in the list to select it and this will be reflected in the data trace as well as the Event details dialog.

Export to HJCFit

This command exports idealised trace data to a `.scn` data file suitable for use by HJCFit and EKDIST. HJCFit is an analysis program used for modelling channel states and calculating the rate constants for each possible state change. EKDIST is a separate program used to build and plot histograms. These applications take an idealised trace as input, this needs to be in a `.scn` format file. HJCFit and EKDIST are not CED software, they are written and maintained by Prof. David Colquhoun of UCL. For more information on these programs see http://www.ucl.ac.uk/Pharmacology/dcpr95.html.

Short cuts

Because of the intensely interactive nature of idealised trace generation using the event details dialog, you can set keyboard shortcuts so that a single keypress is the equivalent of clicking on a button. These shortcuts can be freely adjusted to match your preferences. Two standard configurations of shortcuts can also be set using the Numeric Defs and the Letter Defs buttons. Letter Defs will set the standard shortcuts as indicated by the underlines on the event details dialog. Numeric Defs will select a set of shortcuts that uses the numeric keypad. This is to provide the keys close together by the right hand to speed up interaction as much as possible. If you use numeric shortcuts you should remember to have Num Lock on.



Command	Key Stroke
Chop	C
Close	Esc
Delete	D
Fetch Amp	F
Fit Visible	V
Merge	M
Next	N
Parameters	R
Previous	P
Process	E
Scroll Back	B
Scroll On	O
Split	S
Undo	U

Tips for fitting

Occasionally the fitting routines may produce unexpected results and there can be a number of reasons for this.

1. The initial guess is too poor. Use the other edit functions to produce a more plausible guess.

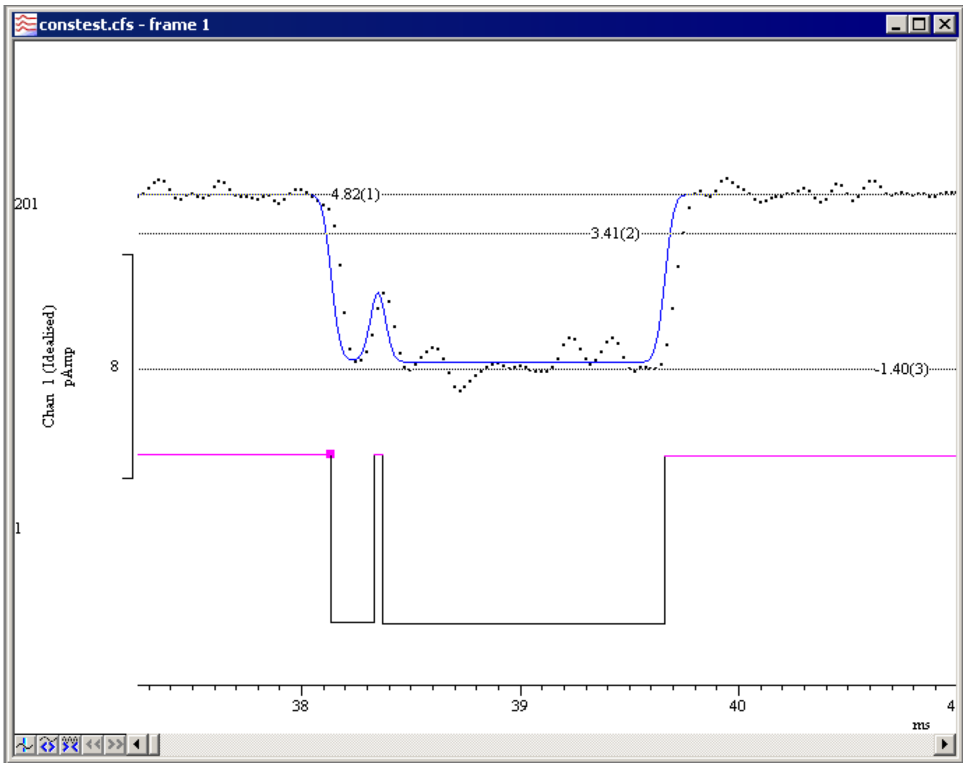
2. An amplitude needs fixing. If you flag an event as having an assumed amplitude it will be held fixed for the fit. If it is a very short event the assumed amplitude flag will remain set otherwise the fit routine will clear the flag.

Strategy for long recordings

Although it is possible to analyse a long single channel recording by simply zooming all the way out and processing the entire recording in one go then hitting Fit Visible to give your result, this is not to be recommended. Subtle changes in conditions throughout the file combined with mistakes in the processing caused by noise conspire to make this an error prone strategy. An incremental approach is needed and here is a suggested approach for doing this.

The SCAN method is by far the superior method for event detection. It copes well with sub-conduction bands and can detect events far shorter and with far higher time resolution than would be possible with threshold crossing. To begin with, select an area of data near the start of the recording with a full opening and an area of baseline showing. Place a horizontal cursor on the baseline, another on the full open level and a third between the two towards the baseline but out of reach of the noise. Once these have been set up you can now evoke the SCAN settings dialog via the analysis menu as described above. Use the drop downs in the dialog to select the relevant horizontal cursors for the different levels. Select a Data Start Time of Mintime() and a Data End Time of XHigh(). Long recordings will normally only be a single frame of data, however, if there are multiple frames you will want to process just the first one to start with. Once the initial process has been done an idealised trace will appear. This is just the initial guess and will now need refining. Zoom in on the first opening or burst so you can see the transitions in more detail.

The convolution of the idealised trace with the step response function for the amplifier will be drawn on top of the data. By default this will be in blue; indicating that the trace has not yet been fitted. At this stage you may decide that the initial guess is poor and edit the trace to something more plausible. Once you are happy with the guess, click on Fit Visible and the convolution should jump to fit the data and turn black; indicating the fit was successful. If it does not then the initial guess may need to be refined further (See Tips for Fitting, above).



Once you are happy with the idealised trace on the screen, step forward using either the Next or Scroll On buttons. These will take you either to the next part of the idealised trace to work on or to the next possible opening past the end of the idealised trace. If it is the latter you can then decide if this possible opening is real or just an artefact. If it is an artefact you can step past it using either the Next or Scroll On buttons, optionally setting the Bad data flag. If a possible opening extends past the end of the display you may like to adjust the display range to include the full open time. The incremental analysis will work better if breaks in the middle of openings are avoided though if this is not possible you can still edit any discontinuity that appears as a result and re-run the fit. A fit done on a section of data with no idealised trace fitted will generate a guess first. Closing the data file at any time will save the idealised trace and process settings to disk allowing the analysis to be spread over several sessions.

14 Cursor menu

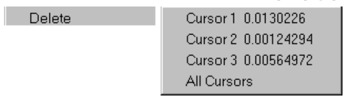
A cursor is a vertical or horizontal dashed line drawn in a data view to mark or obtain a position. The Cursor menu creates and destroys cursors, defines the behaviour of vertical active cursors, changes their labelling mode and obtains the values of channels where they cross the cursors and between the cursors. Up to 10 cursors of each type, numbered 1 to 10, can be active in each data view, there is also an extra vertical cursor, number 0, which is used for special purposes. Cursors can be dragged over and past each other and horizontal cursors can be dragged from channel to channel. Cursors in separate windows are independent of each other. When a window is duplicated, the cursors are also duplicated.

In addition to using the Cursor menu commands, if you right-click on a vertical or horizontal cursor there are additional cursor commands available from the context menu. For example you can copy the position of cursor, or the difference of two horizontal cursor positions to the clipboard.

New Cursor

This menu command (keyboard shortcut `Ctrl+|`) duplicates the action of the new cursor button at the bottom left of data views. The command is available when a data view is the current window and there are less than ten cursors already active in the view. A new vertical cursor is added at the centre of the window. The cursor is given the lowest available cursor number and is labelled with the cursor label style for the window.

Delete



This command opens a pop-up menu in which you select a cursor to remove, or you can delete all cursors. The cursors are listed with their number and position as an aid to identification. Deleting a cursor removes it from the view; other cursors are not affected.

Fetch

This opens a pop-up menu where you select a cursor to place in the centre of the x axis.

Move To

This command activates a pop-up menu from which you can select the cursor to move to. The cursors are listed with their number and position as an aid to identification. The window scrolls to show the cursor in the screen centre, or as close to it as possible.

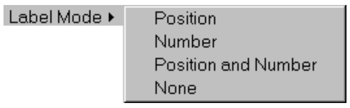
Position Cursor

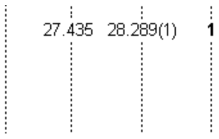
This command opens a pop-up menu to select a cursor and then opens a dialog in which you can type or select a cursor position. You can also activate this dialog by right clicking on a vertical cursor and selecting Set Position from the cursor context menu.

Display All

This command has no effect if there are no 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



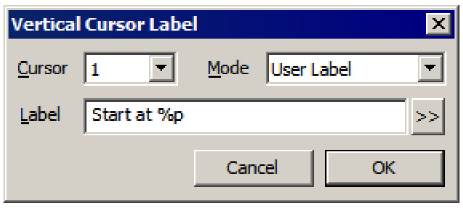


Each cursor has an optional label used to identify it. You can drag the cursor labels up and down the cursor with the mouse to suit the data. There are five cursor label modes: None, Position, Position and Number, Number and User-defined. Select the most appropriate mode for your purposes from the pop-up 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 cursors in the window. You can drag the cursor labels up and down the cursor with the mouse to suit the data.

Each cursor stores its own mode and label, and the view has a mode that is applied to new cursors. The first four items in the menu set the view mode and the mode of all cursors. You can also set a user-defined label for cursors (but not for the view) with the Set Label command.

Set Label

You can open the cursor label dialog from the Label Mode and Horizontal Label Mode cursor menu commands or by right-clicking on a cursor and using the Set Label command from the cursor pop-up menu.


Vertical Cursor Label

Cursor: 1 — Mode: User Label

Label: Start at %p

From this dialog you can set the cursor label mode for one or all cursors. The Cursor field can be set to the number of any cursor in the view, or All. If you choose All, any change applies to all cursors and sets the view mode except in User-defined mode, where the view mode does not change.

The Label Mode field lets you choose one of None, Position, Number, Position and Number, and User-defined as the cursor mode. If you select User-defined, the Label field appears together with the >> button and you can set a label of your choosing.

User-defined labels

User-defined labels display the text you type, except that the text sequences `%p`, `%n` and `%v(`*n*`)` are replaced with the cursor position, cursor number, or the value of channel *n* where it is crossed by the cursor. You can also stipulate the width (*w*) and the number of decimal places (*d*) used for the position and value by using `%`*w*`.`*d*`p` and `%`*w*`.`*d*`v(n)`. For example: `%n at %6.4p, %v(2)` might display: `1 at 2.2346, 87.128756` if cursor 1 was at 2.2346 x axis units and channel 2 had the value 87.128756 at this point. The `%v(`*n*`)` option is not allowed for horizontal cursors or for vertical cursors in an XY view. The value returned by `%v(`*n*`)` is the same value as displayed in the cursor values dialog for that cursor and channel.

The >> button pops up the list of replacements, and if you choose one, it replaces any selection in the Label field. If you choose the `%v(`*n*`)` option, you are prompted to select a channel to measure.

We allow you to type in quite long labels. However, when you close a data file, only the first 19 characters of a user-defined label are saved and long labels look messy, so it is usually a good idea to keep labels short.

Renumber

This command renumbers vertical cursors 1 to 10 by position, with cursor 1 on the left.

Active cursors

Normally, Signal cursors are static; they stay where they are put. Using the cursor mode dialog, cursors in time and memory views can be made active; they will move to the position of a data feature, if it can be found. This search is repeated whenever the view data changes, cursor 0 is iterated, or the view switches to a different frame. This repositioning is carried out in order of cursor number so cursor 2 can reliably make use of the current position of cursor 1 but not vice-versa. Active cursors can be used as a simple way of quickly finding features within your data; they are also very powerful tools for extending the capabilities of analyses that generate measurements from data files in XY views.

Cursor 0

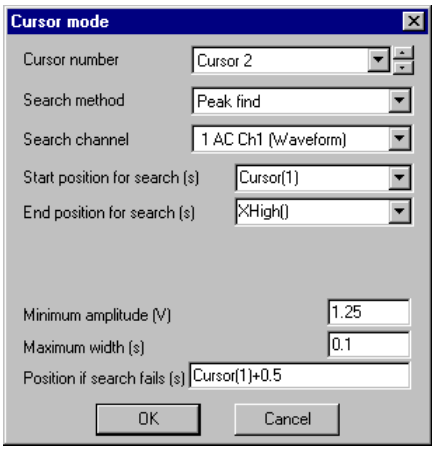
Vertical cursor 0 is special. It always exists and cannot be deleted, but it can be hidden (and will normally be initially hidden by Signal). Unlike the other cursors, when cursor 0 is active it is designed to iterate through the data to repeatedly find features in the waveform. Whenever cursor 0 is moved, all the other active cursors will recalculate their positions in order of cursor number. To hide cursor 0, right-click on it and select Hide in the cursor 0 context menu.

Valid and invalid cursors

Active cursor positions are either *valid* or *invalid*; invalid cursors have an exclamation mark at the end of the label. The cursor position is invalid if the active search 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 and Measurements analyses reject points that come from invalid measurements. Cursor positions are made valid by any operation that moves them to a specific place such as dragging.

Active mode

This command opens a dialog from which you can select a cursor and set up its active search mode. The contents of the dialog depends upon the Search method field and the selected cursor. An active cursor has an associated Search channel and start and end positions for the search that define the data within a frame that is searched.


Cursor mode

Cursor number	Cursor 2
Search method	Peak find
Search channel	1 AC Ch1 (Waveform)
Start position for search (s)	Cursor(1)
End position for search (s)	XHigh()
Minimum amplitude (V)	1.25
Maximum width (s)	0.1
Position if search fails (s)	Cursor(1)+0.5

Cursor 0 does not have a Position if search fails parameter. However, it does have a Minimum step field; iterations of cursor 0 forwards or backwards will reject features that are closer to the previous feature than this limit. Cursor 0 also has a restricted range of search methods: Peak find, Trough find, Rising threshold, Falling threshold, Outside dual thresholds, Within dual thresholds, Slope peak, Slope trough, +ve slope threshold, -ve slope threshold, Turning point and Expression.

The start and end search positions can be a fixed time, but they will often be expressions that involve the positions of other active cursors, particularly when cursor 0 is being used to iterate through features. For cursor n, an expression that refers to an active cursor less than n refers to the new cursor position. An expression that refers to a cursor greater than or equal to n refers to the old position – this is to be avoided as it will cause odd effects when switching between data frames.

If the End position for search is less than Start position for search, searches go backwards through the data. If a search is backwards, read *previous* for *next* and *last* for *first* in the descriptions of the search methods. As far as is possible, we have designed searches that run backwards to work in a similar manner, and to find the same positions, as searches that run forwards. However you will find that threshold crossing searches that use the Delay after crossing parameter will behave differently when running backwards.

Several modes use the slope of a waveform. These methods all have the field Width for slope measurement, which sets the time range over which slopes are calculated. Signal uses the data points from Width/2 before the current point to Width/2 after to calculate the slope unless there are more than 2000 points, in which case 1000 points before and 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 uses data around it, the slope within Width/2 of the ends of the data frame cannot be relied upon.

Static

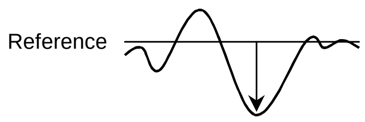
When you add a new cursor, it starts off in Static mode. In this state, the cursor stays where you put it; it is not changed by a change in the data or the position of a lower numbered cursor.

Maximum and Minimum

This finds the position of the maximum or minimum value found within the search range.

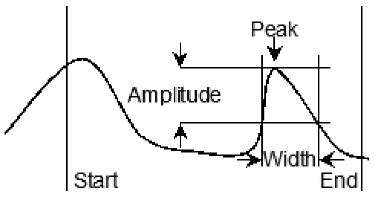
Maximum excursion

This finds the position of the data point that is the maximum distance in the y direction away from a reference level. There is an extra field in this mode for the Reference level.



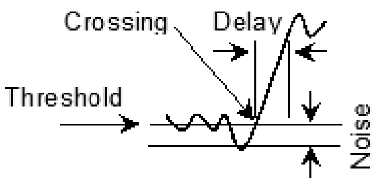
Peak, Trough

The Minimum amplitude field sets the minimum acceptable size of the peak or trough; by how much the data must rise before a peak and fall after it (or fall before a trough and rise after it), to be accepted. In the diagram of a peak search, the first peak is not detected because the data did not rise by Minimum amplitude within the time range of the search. The Maximum width for peak field rejects peaks that are too broad (set it to 0 for no width restriction). The peak position is located by fitting a parabola through the highest point and the points on either side.



Rising threshold, Falling threshold, Threshold

The data must cross Threshold from a level that is at least Noise rejection/hysteresis away from it and stay crossed for a time of at least Delay after crossing. For a Rising threshold mode, the data must increase through the threshold, for a 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. The crossing point is found by linear interpolation of the data points on each side of the crossing.

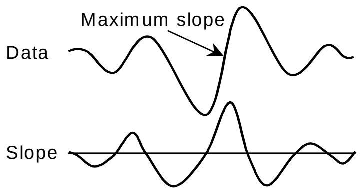


Outside dual thresholds, Within dual thresholds

These two modes are similar to the rising and falling threshold modes except that they search for the data being outside or within two threshold levels. In addition to the First threshold, Delay after crossing and Noise rejection/hysteresis fields there is a Second threshold which sets the other threshold level. The data must cross a threshold from a level that is at least Noise rejection away from it.

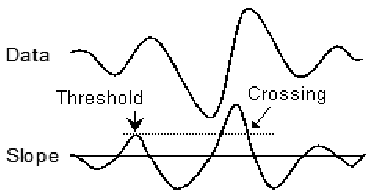
Steepest rising, Steepest falling, Steepest slope (+/-)

This finds the position of the maximum, minimum or maximum absolute value of the waveform slope. The Width for slope measurement field sets the length of data used to calculate the slope.



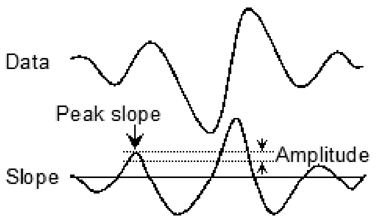
Slope threshold, +ve slope threshold, -ve slope threshold

This finds the position at which the slope crosses a particular threshold level. The Width for slope measurement field sets the length of data used to calculate the slope. The Threshold units are y axis units per second



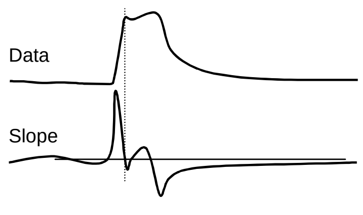
Slope peak, Slope trough

These modes calculate the slope of the data in the search range, and then find the first peak or trough in the result that meets the Amplitude limit. 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), to be accepted. The Amplitude units are y axis units per second.



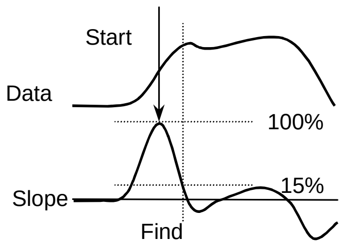
Turning point

This mode finds the first point in the search range where the slope changes sign. Put another way, it finds a localised peak or trough. 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



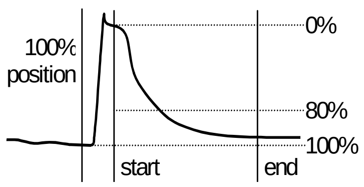
Slope%

This method finds 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 measured at the start of the search that we want to find. To use this mode, set a cursor at the peak slope, then use it as the start point and search forwards or backwards for the required percentage. 15% usually works well.



Repolarisation %

This mode finds the point at which the waveform returns to a set percentage of the distance to a baseline. The search start position defines the 0% repolarisation level. The 100% position and 100% measurement width fields set the position of the 100% level (this can lie outside the search range) and the width over which it is measured. Repolarisation percentage (drawn at 80% in the picture) sets a threshold relative to the measured 0% and 100% levels. The position is the first point in the search range to cross it.



Data point

Moves on by a specified number of data points. This is most useful for stepping through digital markers.

Expression

The cursor position is obtained by evaluating the Expression field. This field will normally hold an expression based on cursor positions, for example "`Cursor(1)+2.5`".

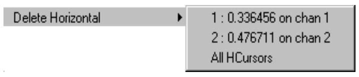
Search Right
Search left

If cursor 0 is active, these two commands cause the cursor to search for the next or previous position that satisfies the active mode. If the cursor active mode is Expression, the cursor goes the same way for both commands.

New Horizontal Cursor

This menu command is available when a data view is the current window and there are less than four horizontal cursors in the view. A new horizontal cursor is added at the centre of the data for the lowest numbered visible channel. The cursor is given the lowest available number and is labelled using the horizontal 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 of them. The available cursors are listed with their number, position and channel number as an aid to identification. Deleting a cursor removes it from the window; other cursors are not affected.

Fetch Horizontal

This opens a pop-up menu where you select a horizontal cursor that is placed in the centre of the visible y axis for the relevant channel.

Move To Level

This command activates a pop-up menu from which you can select the horizontal cursor to move to. The cursors are listed with their number, positions and channel as an aid to identification. The Y axis of the relevant channel will be scrolled to display the nominated cursor in the centre of the axis, or as close to the centre as possible. This command does not change the y axis scaling.

Position Horizontal

This command opens a pop-up menu in which you can choose a horizontal cursor and then opens a dialog in which you can set the position and channel for the cursor. You can also open the dialog by right clicking on a horizontal cursor and selecting Set Position from the cursor context menu.

Display All Horizontal

This command is the equivalent of using the Fetch Horizontal command for all cursors.

Horizontal Label Mode

Each cursor has an optional label used to identify it. You can drag the cursor labels to the left and right with the mouse to suit the data. There are five cursor label modes: None, Position, Number, Position and Number, and User-defined. You select the most appropriate for your application using the pop-up menu or by right clicking on a cursor and choosing to set the cursor label from the context 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.

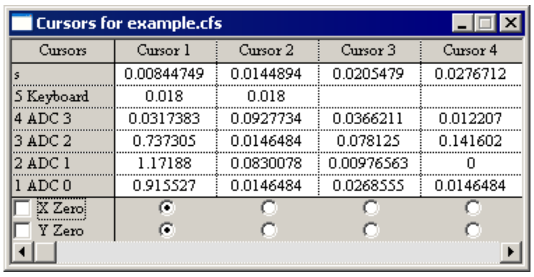
Each cursor stores its own mode and label, and the view has a mode that is applied to new cursors. The first four items in the menu set the view mode and the mode of all cursors. You can also set a user-defined label for cursors (but not for the view) with the Set Label command.

Renumber Horizontal

When created, cursors take the lowest available cursor number rather than being ordered by position. You can also drag cursors over each other, confusing the ordering further. This command renumbers the cursors by position, with cursor 1 at the bottom.

Display Y values

This command opens a new window containing the values at the position of any cursors in the current data view. Columns for cursors that are absent, or for which there is no data, are blank.



Cursors for example.cfs

Cursors	Cursor 1	Cursor 2	Cursor 3	Cursor 4
s	0.00844749	0.0144894	0.0205479	0.0276712
5 Keyboard	0.018	0.018		
4 ADC 3	0.0317383	0.0927734	0.0366211	0.012207
3 ADC 2	0.737305	0.0146484	0.078125	0.141602
2 ADC 1	1.17188	0.0830078	0.00976563	0
1 ADC 0	0.915527	0.0146484	0.0268555	0.0146484
☐ X Zero	●	○	○	○
☐ Y Zero	●	○	○	○

The values displayed depend upon the channel type and display mode. There is an entry in the table showing the time for each cursor, plus entries for each channel displayed. The displayed values are as follows:

Waveform	The y axis value of the nearest data point that is within one sample period of the cursor, or nothing if there is no data point close enough. Waveform measurements are not affected by the drawing mode.
Marker as Rate	The height of the rate bin that the cursor crosses. If the cursor lies on a bin boundary, the cursor is considered to lie in the bin to the right.
Marker	The time of the next marker at or to the right of the cursor.

The X zero check box enables relative cursor time measurements. 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 an absolute time, not 0.

The Y zero check box enables relative cursor value measurements. 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.

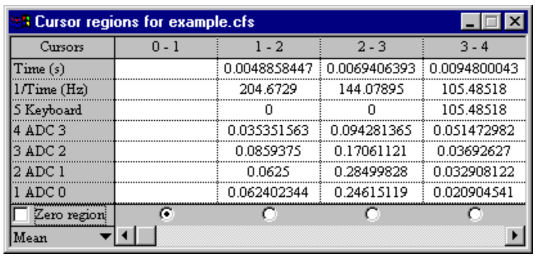
Selecting and copying data

You can select areas of this window by clicking on them. 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 `Ctrl` key to select non-contiguous rows and columns.

To copy selected rows and columns to the clipboard, by right-click the values window and use the Copy command in the popup menu. If you use the Log command the selected text is copied and pasted directly into the log window in one operation. You can also print the selected portions of the window by right-clicking and using the Print command in the popup menu, or use the Font command to change the window font.

Cursor Regions

This command opens the Cursor regions dialog for the current data view. The dialog displays values for data regions between cursor pairs. At the top of the dialog is a row of cells showing the separation in time of each cursor pair, and a second row showing the inverse of the separation as Hz. One pair 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. The pop-up menu in the bottom-left corner indicates and controls how the values are calculated.



Cursor regions for example.cfs

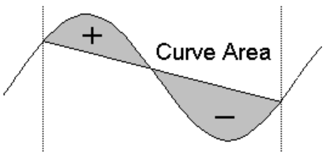
Cursors	0 - 1	1 - 2	2 - 3	3 - 4
Time (s)		0.0048858447	0.0069406393	0.0094800043
1/Time (Hz)		204.6729	144.07895	105.48518
5 Keyboard		0	0	105.48518
4 ADC 3		0.035351563	0.094281365	0.051472982
3 ADC 2		0.0859375	0.17061121	0.03692627
2 ADC 1		0.0625	0.28499828	0.032908122
1 ADC 0		0.062402344	0.24615119	0.020904541
☐ Zero region	◉	○	○	○

Mean

Cursor region measurements

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. For a waveform channel (including one drawn as histogram etc), the measurements are:

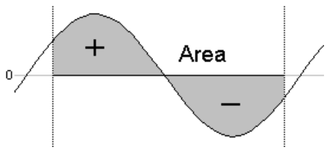
Curve area Each data point makes a contribution to the area of its amplitude above a line joining the endpoints multiplied by the x axis distance between the data points. The picture makes this clearer. This measurement cannot be made on log-binned data.



Mean The mean value of all the waveform points in the region. If there are no samples between the cursors the field is blank.

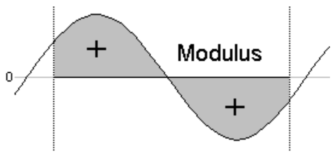
Slope The slope of the least squares best fit line to waveform points in the region.

Area The area between the data points and the y axis zero. Area is positive for sections above zero and negative for sections below zero. Use Modulus if you want areas below the y axis to be treated as positive.



Sum The sum of all the waveform points in the region. If there are no samples between the cursors the field is blank.

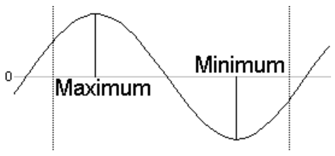
Modulus Each waveform point makes a contribution to the area of its absolute amplitude value multiplied by the time between samples on the channel. This is equivalent to rectifying the data, then measuring the area over zero.



If a zero region is specified, the amount subtracted from the other regions is scaled by the relative width of the regions.

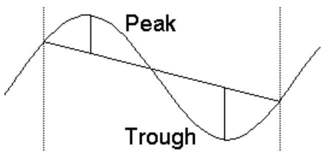
Maximum The value shown is the maximum value found between the cursors.

Minimum The value shown is the minimum value found between the cursors.



Amplitude The value shown is the difference between maximum and minimum values found between the cursors.

RMS amp.	The value shown is the RMS level of the values found between the cursors. If there are no values between the cursors the field is blank.
SD	The value shown is the standard deviation from the mean of the values between the cursors. If there is no data, the field is blank.
Abs Max	The value shown is the maximum absolute value between the cursors. If the maximum was +1, and the minimum was -1.5, this mode would display 1.5.
Peak	The value shown is the maximum found between the cursors measured relative to the baseline formed by joining the two points where the cursors cross the data.



Trough	The value shown is the minimum value found between the cursors measured relative to the baseline formed by joining the two points where the cursors cross the data.
Point Count	The number of waveform or marker channel data points, or idealised trace channel transitions.

The measurements available for marker channels are Mean, Sum, Maximum, Minimum, Amplitude and Abs max. If you select other measurements the result is a blank field. The values calculated for the measurements are:

Mean	The count of markers between the cursors divided by the time difference between the cursors. This could be thought of as the mean marker rate.
Sum	The total number of markers between the cursors.
Maximum	The maximum inter-marker interval, or the maximum histogram value for Rate display mode.
Minimum	The minimum inter-marker interval, or the minimum histogram value for Rate display mode.
Amplitude	The difference between the Maximum and Minimum values.
Abs max	The largest absolute value of Maximum and Minimum, this will always be the same as Maximum for marker channels.

Selecting and copying data

You can select areas of this window by clicking them. 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 `Ctrl` key to select non-contiguous rows and columns.

To copy selected rows and columns to the clipboard, right-click in the values window and use the Copy command in the popup menu. If you use the Log command the selected text is copied and pasted directly into the log window in one operation. You can also print the selected portions of the window by right-clicking and using the Print command in the popup menu, or use the Font command to change the window font.

The above popup menu commands are also available via the following keyboard shortcuts:

`Ctrl+C`	Copy
`Ctrl+P`	Print
`Ctrl+F`	Font
`Ctrl+L`	Log

Context menu commands

In addition to the main menu commands it is also possible to access some commands by right-clicking on a cursor. This produces a popup menu which has a sub-menu specific to the cursor which has been clicked on. For vertical cursors the sub-menu has the following items:

Active mode…	Puts up the active cursor mode dialog for the cursor.
Set position …	Puts up the position cursor dialog for the cursor.
Set Label …	Puts up the cursor label mode dialog for the cursor.
Copy Position=xxx	The position of the selected cursor is copied to the clipboard.
Delete	Deletes the cursor (this is Hide for cursor 0).

For horizontal cursors the list of commands are as follows:

Set position …	Puts up the position cursor dialog for the cursor.
Set Label …	Puts up the cursor label mode dialog for the cursor.
Copy Position=xxx	Copies the position (xxx) of the cursor to the clipboard.
Copy Position-HCursor(n)=Y	Subtract the position of horizontal cursor n from the cursor position and copy the result (Y) to the clipboard. Only available if there are multiple horizontal cursors on this channel.
Delete	Deletes the selected cursor.

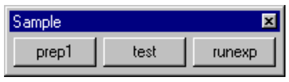
15 Sample menu

The sampling menu divides into three regions. The first configures the channels to sample and provides support for users with a serial line controlled signal conditioner, for example the CED 1902. The second region shows or hides the sampling and output control panels during sampling. The third region matches the sampling control panel and holds commands to start, continue and end sampling, enable and disable sweep triggers and to enable and disable data storage to disk.

Sampling configuration

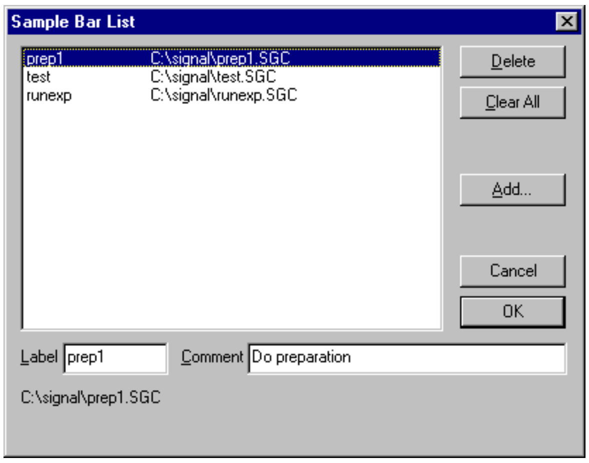
This command opens the Sampling Configuration dialog, which sets the data capture parameters used when you select the File menu New command (see the *File menu* chapter for details). You can load and save the sampling configuration with the File menu Save Configuration and Load Configuration commands. You can also access this command from the Signal toolbar.

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 Signal 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 Signal toolbar or on the Signal background.

Sample Bar List


Sample Bar List

prep1	C:\signal\prep1.SGC
test	C:\signal\test.SGC
runexp	C:\signal\runexp.SGC

Label prep1 Comment Do preparation

C:\signal\prep1.SGC

The Sample menu Sample Bar List… command opens the Sample Bar List dialog in which you control the Sample Bar contents.

Add opens a file dialog in which you can choose Signal configuration files (`*.SGC`) 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 configuration file contents. 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.

The Sample Bar state is saved in the registry when Signal closes and is loaded when Signal opens. Each Windows logon account has a different registry configuration. If your system has three user accounts, each has its own Sample Bar settings.

Signal conditioner

Signal 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 command is available during sampling to open the conditioner dialog so that you can view and change the amplifier settings online (see the *Programmable signal conditioners* chapter for a full description).

Show Sampling controls

This command, or its toolbar equivalent, hides and shows the sampling control panel; the menu item is checked when the control panel is visible. The main controls within the control panel are duplicated in this menu as the Start sampling, Continue sampling, Triggered sweeps, Write to disk at sweep end, Pause at sweep end, Abort sampling and Restart sampling commands, (see the *Sampling data* chapter for full details of the control panel commands). In summary, the commands are:

Start sampling This command starts sampling. It is the same as the sampling control panel Start button.

Stop sampling When sampling has started, the Start sampling command changes to Stop sampling. This is equivalent to the sampling control panel Finish button. There is no warning before this command takes effect.

Continue sampling This command enables sampling of the next sweep when sampling is paused after collecting a sweep. It is equivalent to the sampling control panel Continue button.

Triggered sweeps This command toggles the state of the Sweep trigger checkbox in the sampling control panel. The menu item displays a checkbox when this option is selected.

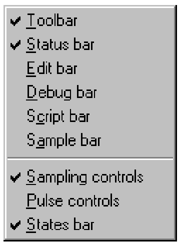
Write to disk at sweep end This command toggles the state of the Write to disk at sweep end checkbox in the sampling control panel. The menu item displays a checkbox when this option is selected.

Pause at sweep end This command toggles the state of the Pause at sweep end checkbox in the sampling control panel. The menu item displays a checkbox when this option is selected.

Abort sampling This command aborts sampling and discards any sampled data. It is equivalent to the Abort button in the sampling control panel.

Restart sampling This command discards all data, returns sampling to the state it was in before sampling started and restarts sampling. It is the same as the sampling control panel Restart button.

Show Pulse controls



This command, and its toolbar equivalent, hides and shows the pulse definition dialog that is available during sampling if pulse output is in use. This dialog can be used to change the output pulses while data acquisition is in progress, changes made will be saved in the current sampling configuration. The menu item is checked when the control panel is visible.

The sampling control panel, pulse controls and states control bar can all be shown and hidden by using the popup menu generated by clicking the right mouse button on an unused part of the Signal window (the blank parts of the toolbar area are suitable and always visible) during sampling.

Sample now

This command is only available on the toolbar. It is equivalent to selecting New in the File menu then choosing Data Document. That is to say: it prepares Signal to start sampling with the current sampling configuration.

Show Sequencer controls

This command hides or shows the sequencer control panel that is available during sampling if the output sequencer is in use.

16 Script menu

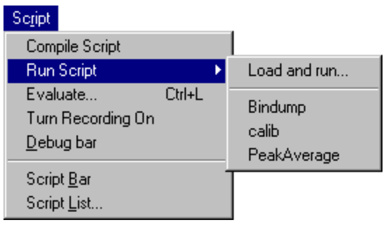
The Script menu gives you access to the scripting system. From it you can compile a script, run a loaded script, evaluate a script command for immediate execution and record your actions as a script. You can find details of the script language and a description of the script window in the separate manual *The Signal script language* and in the on-line help. The script menu commands are:

Compile Script

This command is enabled when the current view holds a script. It is equivalent to the Compile button in the script window. Signal checks the syntax of the script, and if it is correct, it generates a compiled version of the script, ready to run.

Run Script

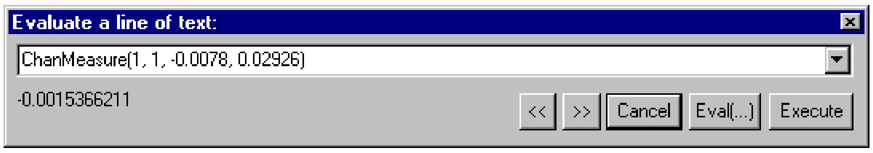
This command pops-up a list of all the scripts that have been loaded so that you can select a script to run. Signal compiles the selected script and if there are no errors, runs the script. If you run a script twice in succession, Signal only compiles it for the first run, saving the compilation time. 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 and the Ctrl+L keyboard shortcut open the Evaluate dialog where you can type a line of script commands for immediate execution. The window remembers the last ten lines of script entered, which are shown in the drop-down list. You can cycle round the saved lines using the << and >> buttons. The Execute button executes the line entered, Eval(...) adjusts the line internally to include an `Eval()` on the last statement so that you can see the result. You can execute any script that can be typed in one line, which can include variable declarations.



```
ChanMeasure(1, 1, -0.0078, 0.02926)
-0.0015366211
```

Turn Recording On/Off

You can record your actions into a script that will produce equivalent actions. Use this command to turn recording on and off. When you turn recording on, Signal begins to save script commands corresponding to your actions. While script recording is in progress, the rightmost indicator in the Signal status bar will display the text REC as a reminder. When you turn recording off, a new script window opens that holds the saved script commands. If you then compile and run this script, the actions that you performed while recording was on will be repeated.

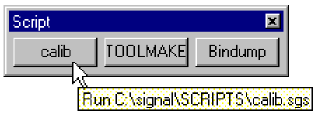
You can use this mechanism to record a sequence of actions that you wish to rerun at some later date, to find out what script commands correspond to a given menu command or user action or to record a sequence of actions that can be copied into another script or edited to produce a complete scripted 'application'.

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 clicking the right mouse button on any Signal toolbar or on the Signal background.

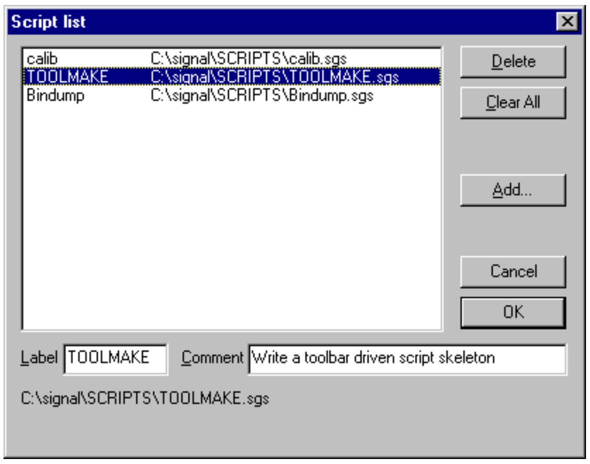
Script Bar

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 Signal toolbar or on the Signal background.



The Script Bar is a dockable toolbar with up to 20 user-defined buttons. Each button is linked to a Signal script file. When you click a button, the associated script is loaded and run. There is also a user-defined comment associated with each button which appears as a tool-tip when the mouse pointer lingers over a button.

Script List



```
calib       C:\signal\SCRIPTS\calib.sgs
TOOLMAKE    C:\signal\SCRIPTS\TOOLMAKE.sgs
Bindump     C:\signal\SCRIPTS\Bindump.sgs
```

Label: TOOLMAKE Comment: Write a toolbar driven script skeleton

C:\signal\SCRIPTS\TOOLMAKE.sgs

This 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 Signal script files (`*.SGS`) 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 Signal closes and is loaded when Signal 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.

17 Window menu

The Window menu has seven permanently available commands in two sections. The first section holds three commands, one to duplicate a data document window and two to hide and show windows.

The second section holds five commands, four to arrange windows and the final one to close all windows.

The remaining space in the menu holds a list of all the windows that belong to the Signal application. If you select one of the windows in the list, the window is brought to the front and made the current window. The list shows the current window checked. The last item in the list activates a dialog, which lists the windows together with their view handles (used by scripts to access them) and the window state (maximised etc).

Duplicate window

This command creates a duplicate window with all the attributes (list of displayed channels, display modes, colours, cursors and size) of the original window. Once you have created the new window, it is independent of the original. Duplicating a window allows you to have different views of the same data with different scales and different channels visible.

You can close all windows associated with a data document using the File menu Close All command (see 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 memory views with analysis results are required.

Show

This command lists all hidden windows. Select a hidden window to make it visible.

Tile Horizontally

You can arrange all the visible Signal windows so that they are arranged in a horizontally tiled pattern by using this command. Horizontal tiling arranges the windows so that they tend to be short and wide, the exact arrangement depends upon the number of windows.

Tile Vertically

You can arrange all the visible Signal windows so that they are arranged in a vertically tiled pattern by using this command. Vertical tiling arranges the windows so that they tend to be tall and thin, again the exact arrangement depends upon the number of windows.

Cascade

All windows are set to a standard size and are overlaid with their title bars visible.

Arrange Icons

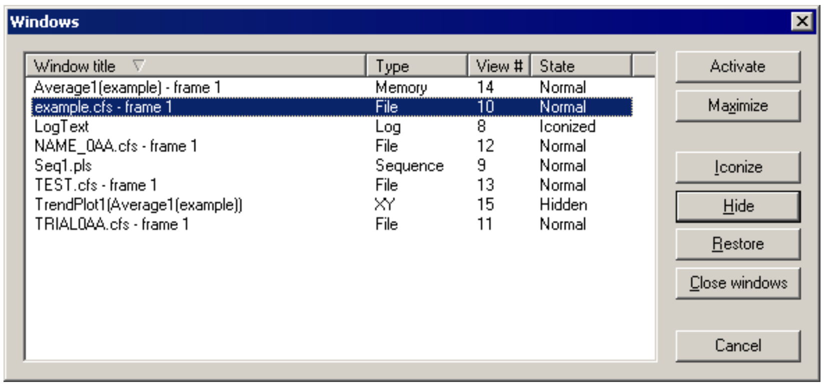
You can use this command to tidy up the windows that you have iconised in Signal.

Close All

This command closes all windows in the Signal application. You are asked if you want to save the contents of any text windows that have changed. The positions of data document windows are all saved.

Windows

This dialog lists all the document-related windows that are open and lets you apply common window operations to one or more of the windows. You can sort the list based on the window title, type, view number (as seen by the script language) and window state by clicking the title bar at the top of the list.



Window title	Type	View #	State
Average1(example) - frame 1	Memory	14	Normal
example.cfs - frame 1	File	10	Normal
LogText	Log	8	Iconized
NAME_0AA.cfs - frame 1	File	12	Normal
Seq1.pls	Sequence	9	Normal
TEST.cfs - frame 1	File	13	Normal
TrendPlot1(Average1(example))	XY	15	Hidden
TRIAL0AA.cfs - frame 1	File	11	Normal

18 Help menu

Using help

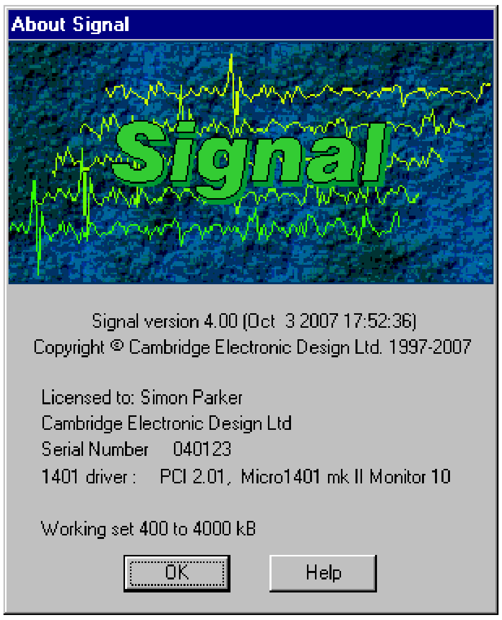
Signal supports context sensitive help and also duplicates the contents of this manual in the help file. You can activate context sensitive help with the `F1` key, or by pressing the Help button, from most dialogs to get a description of the dialog and its fields. You can use the Help menu Index command to get a dialog holding the help contents, an index to help keywords and a word search system to find topics that are not covered by the contents and index.

From a script view or the script evaluate dialog 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 `FileOpen(`, then make sure that the cursor lies to the left of the bracket and in the function name and press `F1`. Pressing the help button (the button with a question-mark) at the top right of the script window provides overall script language help.

The help is implemented using the standard Windows help system, with contents, indexes, hypertext links, keyword searches, help history, bookmarks and annotations. If you are unsure about using Windows help, use the Help menu Using help command to get detailed instructions.

About Signal

This command is found in the Help menu. It opens an information dialog that contains the serial number of your licensed copy of Signal, plus your name and organisation. Please quote the serial number if you call us for software assistance.



Signal version 4.00 (Oct 3 2007 17:52:36)
Copyright © Cambridge Electronic Design Ltd. 1997-2007

Licensed to: Simon Parker
Cambridge Electronic Design Ltd
Serial Number 040123
1401 driver : PCI 2.01, Micro1401 mk II Monitor 10

Working set 400 to 4000 kB

1401 device driver

If there is a 1401 device driver installed, the driver revision is displayed. If the driver is older than Signal expects, you will be warned. Signal 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 Signal was released, an asterisk follows the version. If it is so old that it compromises data sampling, two asterisks follow the version.

The Power1401 and Micro1401 mk II, and more recent devices, have firmware in flash memory. Flash updates and instructions for applying them are available as downloads from the CED web site; you can update the flash firmware without opening the 1401 case.

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.

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 Signal to use. If you use Windows NT and suffer from error -544 when you sample data, these numbers are important.

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 with 256 MB of memory I can generate about 90 result windows before I run out of resources, as long as Signal is the only open application. If system resources is a problem, consider changing to Windows NT 2000 or XP where this is not an issue.

Tip of the Day

This command provides a dialog with a small piece of information in a "Did you know?" form. Further details can then be requested. This dialog can also be set to appear when Signal is first run.

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 Signal and information about CED products.

Other sources of help

If you are having trouble using Signal, please do the following before contacting the CED Software Help Desk:

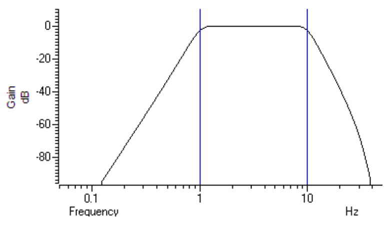
1. Read about the topic in the manual. Use the Index to search for keywords related to the topic.
2. Try the help system for more information. Use the Search facility to find related topics.
3. If none of the above helps, FAX, email or call the CED Software Help Desk (numbers and addresses are to be found at the front of this manual, and in the Contacting CED help page to be found near the start of the help contents). Please include a description of the problem, the Signal serial number and program version number 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 Windows you are running.

19 Digital filtering

FIR and IIR filters

Filtering is used to remove unwanted frequency components from waveforms and can also be used to differentiate a signal. In Signal we provide you with two basic types of filter: Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). Both types of filter have their advantages and disadvantages.

IIR filters



These are similar to analogue filters, and we design them by mapping standard Butterworth, Bessel, Chebyshev filters and resonators into their digital forms. IIR filters have these advantages:

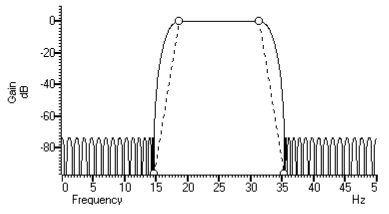
- They can generate much steeper edges and narrower notches than FIR filters for the same computational effort.
- IIR filters are causal; they do not use future data to calculate the output, so there is no pre-ringing due to transients.

They also have disadvantages:

- IIR filters are prone to stability problems particularly as the filter order increases or when a filter feature becomes very narrow compared to the sample rate.
- IIR filters impose a group delay on the data that varies with frequency. This means that they do not preserve the shape of a waveform, in particular, the positions of peaks and troughs will change.

The output of an IIR filter may take a long time to settle down from the discontinuity at the start (transition from no data to the supplied data).

FIR filters



We describe FIR filters in terms of frequency bands: *pass bands*, *stop bands* and *transition gaps*. You define a filter by the arrangement of bands and the corner frequencies of each band. FIR filters have these advantages:

- They are unconditionally stable as they do not feedback the filter output to the input.
- There is no phase delay through the filter, so peaks and troughs do not move when data is filtered (this is called *linear phase* in the literature).

They also have disadvantages:

- They are poor at generating very narrow notches or narrow band pass filters.
- The narrowest frequency band or band gap is limited by the number of coefficients (we allow up to 511 coefficients).
- FIR filters are not causal; they use future as well as past data to generate each output point. A transient in the input causes effects in the output before the transient.

If your FIR filter has n coefficients, the first and last n/2 output points are estimates due to the discontinuity at the start and end of the data.

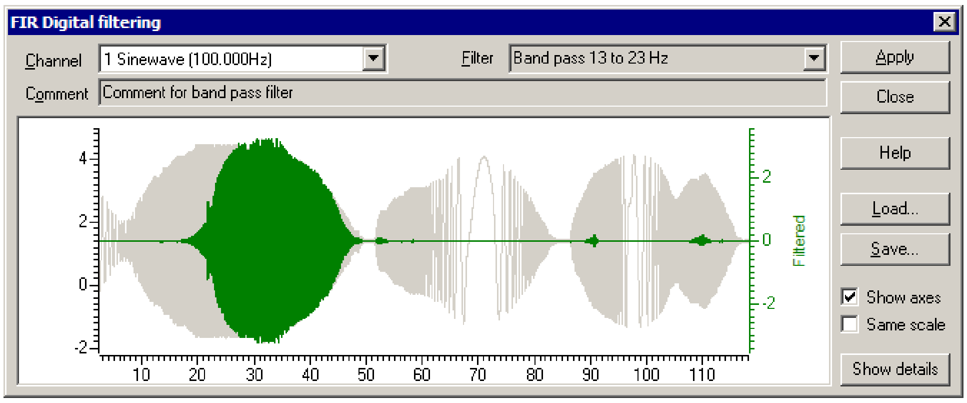
So which to choose?

If you want a differentiating filter, you have no choice as we have not implemented differentiators for IIR filters. Unless one of the disadvantages of the FIR filter is a problem, you will likely have fewer unexpected effects with an FIR filter.

However, there are circumstances in which only an IIR filter will do. If you need a high Q notch filter or resonator, then use an IIR filter. If you are interested in small changes just before a large discontinuity, only the IIR filter will help you. However, make sure that you understand the disadvantages of IIR filters before you depend on their output.

Digital filter dialog

The Analysis menu Digital filters command is available when you have a data or memory view open. You can choose to apply Finite Impulse Response (FIR) filters or Infinite Impulse Response (IIR) filters. Apart from the dialog title, the initial dialog display is the same for IIR and FIR filters. You can apply one of twelve stored digital filters to a waveform channel, or you can create your own digital filter. You can also load and save additional sets of filters from the dialog.



The dialog shows the original waveform in grey, and a filtered version in the colour you have set for waveform data. Whenever you change the filter, the display updates to show the effect of the change.

Show axes

You can choose to Show axes for the original data and for the filtered version. The axis for the original data is always on the left. If a separate axis is required for the filtered data, it is drawn in the filtered data colour on the right.

Same scale

Initially, the filtered data is drawn at the same scale as the original data. However, sometimes this is inconvenient, for example when high-pass filtering a signal with a significant DC offset or when the result of filtering is very small compared to the original. If you clear the Same scale checkbox, the filtered data is scaled to fit in the window independently of the scaling of the original waveform.

Channel

The Channel field allows you to select a waveform channel to filter.

Filter

The Filter field of the dialog box selects the filter to apply. There are normally 12 filters to choose from. When you first open the dialog, this field is grey, indicating that you cannot edit the filter name. If you display the filter details you can modify the filter name.

Comment

The Comment field is for any purpose you wish; there is one comment per filter. When you first open the dialog, this field is grey, indicating that you cannot edit the comment. Click the Show details button to edit the comment.

The Filter field of the dialog box selects the filter to apply and the Channel field sets the waveform channel to filter.

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.

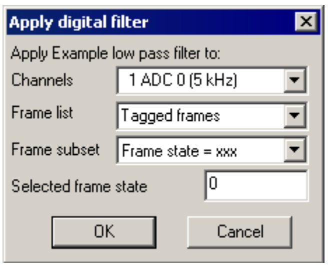
Close

Click the Close button to shut the filtering dialog. If you have made a change to any of the filters, or loaded a new filter set, you are asked if you want to save the current set of filters as your standard filter set.

Load and Save You can choose to save the current set of filters to a `.cfb` file, or to load a new set of filters from a `.cfb` file. If you are working with FIR filters, this only saves or loads FIR filters. If you are working with IIR filters, this only saves or loads IIR filters. Loading a new filter set does not change your standard filter set, however, you will be asked if you want to change your standard set when you close the dialog.

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. When you display the filter details, the Filter and Comment fields become editable. If you change a filter or create a new filter or load a new set of filters from a file, you will be prompted to save the filter bank when you close the digital filter dialog. The details are different for FIR and IIR filters.

Apply The Apply button opens a new dialog in which you set which channels and frames to filter with the Channels and Frame list fields. The Frame subset field can be used to further specify which frames are to be filtered. The Selected frame state field only appears if you select Frame state = xxx in either the Frame list or the Frame subset field. The channels should all have the same sampling rate as the one you were previewing in the main dialog. Channels of a differing sampling rate will be ignored – this situation should only occur with data imported into Signal.


Apply digital filter

Apply Example low pass filter to:

Channels	1 ADC 0 (5 kHz)
Frame list	Tagged frames
Frame subset	Frame state = xxx
Selected frame state	0

As applying a filter can be a lengthy process, a progress dialog appears with a Cancel button during the filtering operation.

FIR filter details 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 either end of the input data should not be used for any critical purpose.

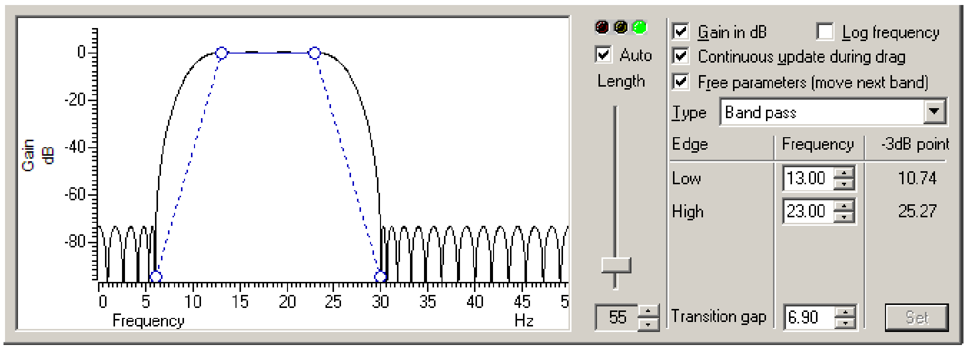
IIR filter details IIR filters are applied to all the data in the selected area. IIR filters feed back the results of previous filter steps into the current step. This means that the result of the filter on a data point depend on all the data points up to and including that point. For a stable filter, the effect of a distant point is much less than that of an adjacent one. After a discontinuity (the start of a frame), it will take some time for the filter to settle down. Generally, the sharper the filter, the longer it takes for the result to settle. If a filter is unstable or close to unstable, the result may never settle.

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, Signal contains a filter bank of 12 FIR and 12 IIR filter definitions. This filter bank is saved to the file `Filtbank.cfb` when you close Signal 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 its type (FIR or IIR) and an index number in the range 0 to 11. Script users also have access to two additional temporary filters with index number -1 (one for FIR and the other for IIR filters) that they can set and use for channel filtering operations without changing the standard filter set.

FIR filter details



☑ Auto
Length
55

☑ Gain in dB ☐ Log frequency
☑ Continuous update during drag
☑ Free parameters (move next band)
Type: Band pass

Edge	Frequency	-3dB point
Low	13.00	10.74
High	23.00	25.27

Transition gap: 6.90

The graph in the details area displays the ideal and actual frequency response of the filter. The ideal response appears as blue solid lines for each pass band linked by dotted lines that mark transition gaps between the bands. All transition gaps have the same width. The calculated frequency response is drawn as solid black lines and is greyed when the filter specification has changed and the response has not yet been calculated.

The mouse pointer changes to indicate the feature it is over: for a pass band or stop band, for a transition gap and for a band corner. The small circles can be dragged sideways to change the slope of the band edges or you can edit the band edges as numbers in the Frequency panel on the right. You can also drag the bands and the transition gaps sideways.

Set If you edit the numbers in the Frequency panel, the Set button is enabled so you can force a recalculation of the filter.

-3dB point 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.

Gain in dB The Gain in dB check box sets the y axis scale to dB if checked, linear if not checked. Using dB is usually the more convenient, except when working on a differentiator.

Log frequency You can choose to display the frequency axis as the logarithm of the frequency, which gives you more resolution at low frequencies. However, this can make working with FIR filters more awkward as it removes the visual symmetry of the transition bands. In log mode, the frequency axis extends down to 0.001 of the data sample rate.

Continuous update 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.

Free parameters 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.

Length, Auto and traffic lights 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.

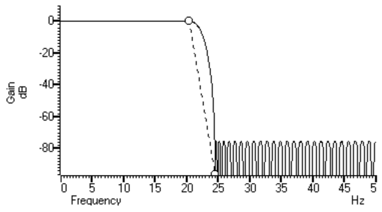
An FIR filter of length n uses the n/2 points before and after each input point to produce each output point. When there is no input data available before or after an input point, the filter uses a duplicate of the nearest point as an estimate of the data value. The n/2 output points next to any break in the input data should not be used for any critical purpose.

FIR Filter types The Type field sets the arrangement of filter bands. If you need a filter that is not in this list you can generate it from the script language with the `FIRMake()` command. There are currently 12 different filter types:

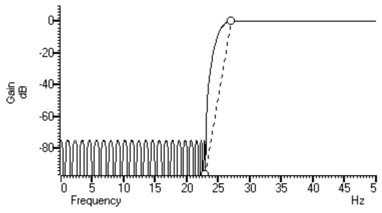
All pass This has no effect on your signal. This filter type covers the case where you apply a low pass filter designed for a higher sampling rate to a waveform with a much lower sampling rate, so that the pass band extends beyond half the sampling frequency of the new file.

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.

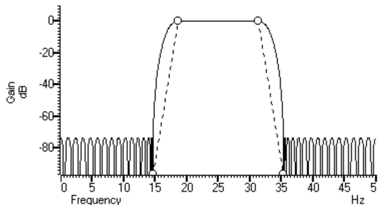
Low pass This filter attempts to remove the high frequencies from the input signal. The Frequency field holds one editable number, Low pass, the frequency of the upper edge of the pass band. The stop band starts at this frequency plus the value set by the Transition gap field.



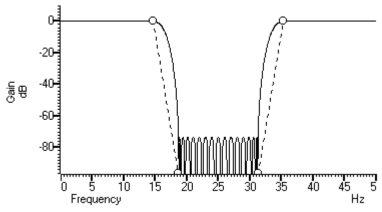
High pass A high pass filter removes low frequencies from the input signal. The Frequency field holds one editable number, High pass, the frequency of the lower edge of the pass band. The stop band starts at this frequency less the value set by the Transition gap field.



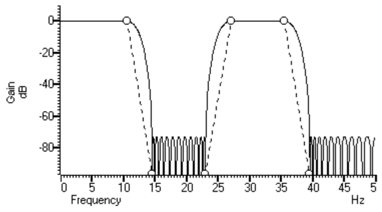
Band pass A band pass filter passes a range of frequencies and removes frequencies above and below this range. The Frequency field has two editable numbers, Low and High, which correspond to the two edges of the pass band. The stop band below runs up to Low-Transition gap, and the stop band above from 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.

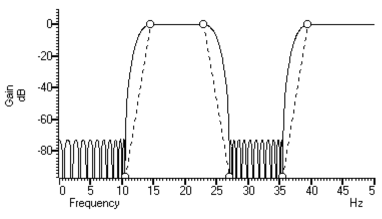


One and a half low pass This filter has two pass bands, the first running from zero Hz and the second in the frequency space between the upper edge of the first pass band and one half the sampling rate. The Frequency field has three editable numbers: Band 1 high, Band 2 low and Band 2 high. These numbers correspond to the edges of the pass bands.



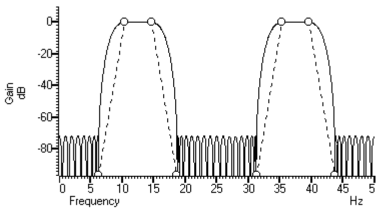
One and a half high pass

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.



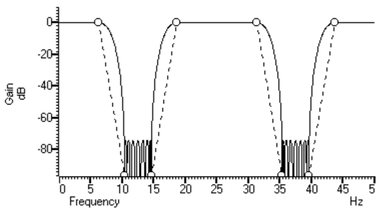
Two band pass

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.



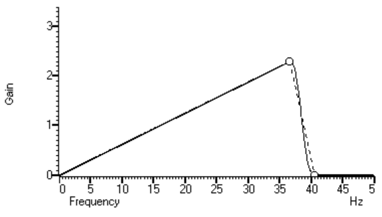
Two band stop

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.



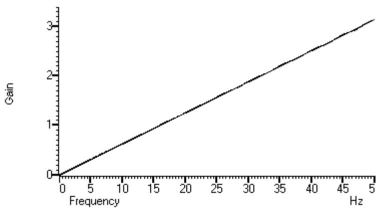
Low pass differentiator

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.



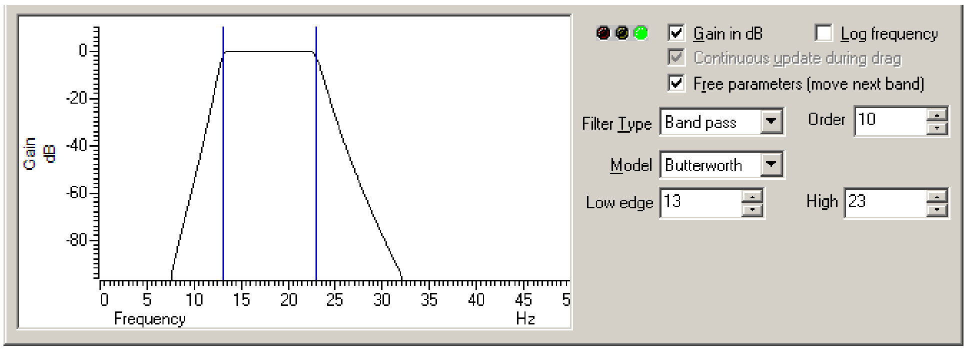
Differentiator

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.



IIR filter details

We describe IIR filters in terms of a filter type (low pass, high pass, band pass or band stop), the analogue filter model that they are based on (Butterworth, for example), the corner frequencies and the filter order (which determines the steepness of the cut-off outside the desired pass bands). Filters based on Chebyshev designs also require a ripple specification and resonators require a Q factor.



☑ Gain in dB ☐ Log frequency
☑ Continuous update during drag
☑ Free parameters (move next band)
Filter Type: Band pass Order: 10
Model: Butterworth
Low edge: 13 High: 23

The graph in the details area displays the corner frequencies and the frequency response of the filter. You can adjust the filter by clicking and dragging in this area or by editing the filter parameters as text. The mouse pointer changes to indicate the feature it is over: ↤↦ for a corner frequency and ↕ for an adjustable parameter (ripple or Q factor).

Gain in dB

The Gain in dB check box sets the y axis scale to dB if checked, linear if not checked. Using dB is usually the more convenient.

Log frequency

You can choose to display the frequency axis as the logarithm of the frequency, which gives you more resolution at low frequencies. The display extends to 0.0001 of the sample rate. If you need a corner frequency below this you should consider sampling the signal more slowly in the first place. Alternatively, low pass filter the signal and then use a channel process to down sample it before filtering.

Continuous update

This is always checked for IIR filters, and is greyed out.

Free parameters

If you check the Free parameters box, corner frequencies are not limited by the next corner, and will push them along. If you clear the box, corner frequencies cannot be moved past each other.

Traffic lights and messages

The traffic lights show green if the filter appears to be OK, amber if the filter may be unstable and red if the filter calculation failed, the filter is unstable or if one of the input parameters is illegal. There will usually be an explanatory message in the lower right hand corner of the dialog explaining the problem. In the amber state, the filter may still be usable; you can look at the frequency response and the filtered data to see if the result is acceptable.

Filter Order

In an analogue filter, the filter order determines the sharpness of the filter, for example Butterworth and Bessel filters tend towards 6n dB per octave, where n is the filter order. In a digital filter, the order determines the number of filter coefficients. The higher the order, the sharper the filter cut-off and also, the more likely the filter is to be unstable due to problems in numerical precision. You can set filter orders of 1 to 10. You should always use the lowest order that meets your filtering criteria. Phase non-linearity gets worse as the filter order increases.

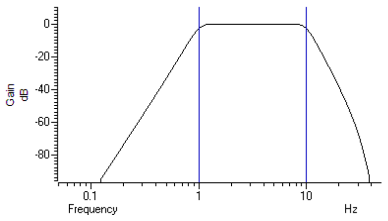
Filter type

There are four filter types: Low pass, High pass, Band pass and Band stop. However, if you set the filter model to Resonator, there are only two types, being Band pass (this creates a resonator filter) and Band stop (this creates a notch filter). For all filter models except Chebyshev type 2 and Resonator, the frequencies given are the points at which the filter achieves a cut of 3 dB. For Chebyshev type 2 filters, the frequencies are the point at which the filter cut reaches the value set by the Ripple parameter. For Resonators, the frequency is the centre frequency of the resonator.

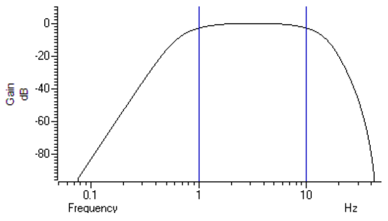
Low pass	Use this to remove high frequencies and pass low frequencies. This has a single corner frequency.
High pass	Use this to remove low frequencies and pass high frequencies. This has a single corner frequency.
Band pass	This passes a range of frequencies between the Low edge and the high edge. If the filter model is Resonator, then this has a single Centre frequency.
Band stop	This removes a range of frequencies between the Low edge and the high edge. If the filter model is Resonator, then this has a single Centre frequency.

Filter Model The IIR filters we provide are digital implementation of standard analogue filter models. The following descriptions use fifth order 1 to 10 Hz band pass filters on 100 Hz data as examples (except for the Resonator filters). The five filter models are:

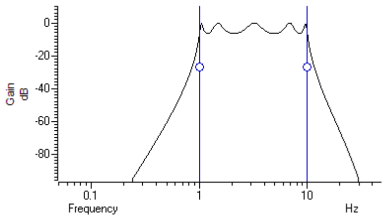
Butterworth This has a maximally flat pass band, but pays for it by not having the steepest possible transition between the pass band and the stop band. This is a good choice for a general-purpose IIR filter, but beware that the group delay can get quite bad near the corners, especially for high-order filters.



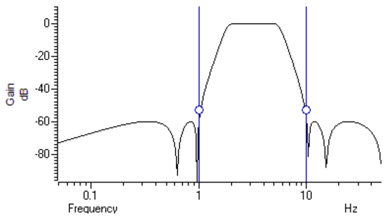
Bessel An analogue Bessel filter has the property that the group delay is maximally flat, which means that it tends to preserve the shape of a signal passed through it. This leads to filters with a gentle cut-off. When digitised, the constant group delay property is compromised; the higher the filter order, the worse the group delay.



Chebyshev type 1 Filters of this type are based on Chebyshev polynomials and have the fastest transition between the pass band and the stop band for a given ripple in the pass band and no ripples in the stop band. You can adjust the ripple by dragging the small circles vertically or with the Ripple field to the right of the displayed frequency response.

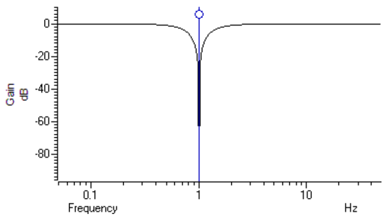


Chebyshev type 2 Filters of this type are defined by the start of the stop band and the stop band ripple (the minimum cut in the stop band). They have the fastest transition between the pass and stop bands given the stop band ripple and no ripple in the pass band. Drag the small circles to adjust the ripple, or use the Ripple field to the right of the frequency response.

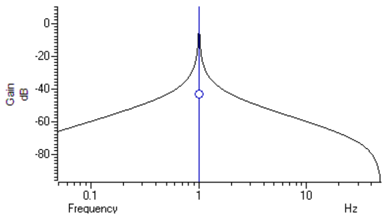


Resonator Resonators are defined by a centre frequency and a Q factor. The Q is the width of the resonator at the -3 dB point divided by the centre frequency. You can have a band stop or a band pass resonator. The Q is adjusted by dragging the small circle or with the Q field to the right of the displayed frequency response.

Band stop resonators are also called Notch filters. The higher the Q, the narrower the notch. Notch filters are often used to remove mains hum, but if you do this you will likely need to set notches at the first few odd harmonics of the mains frequency. The example has a very low Q (1.24) to make the filter response visible.



A band pass resonator is the inverse of a notch. Band pass resonators are sometimes used as alternatives to a narrow bandpass filter. The example has a Q of 100. The higher the Q set for a resonator, the longer it will take for the output to stabilise at the start of the filter output.

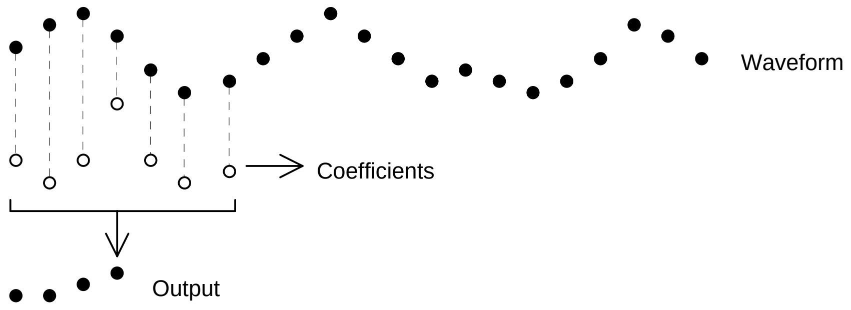


FIR filters technical information

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, Signal 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 Signal.

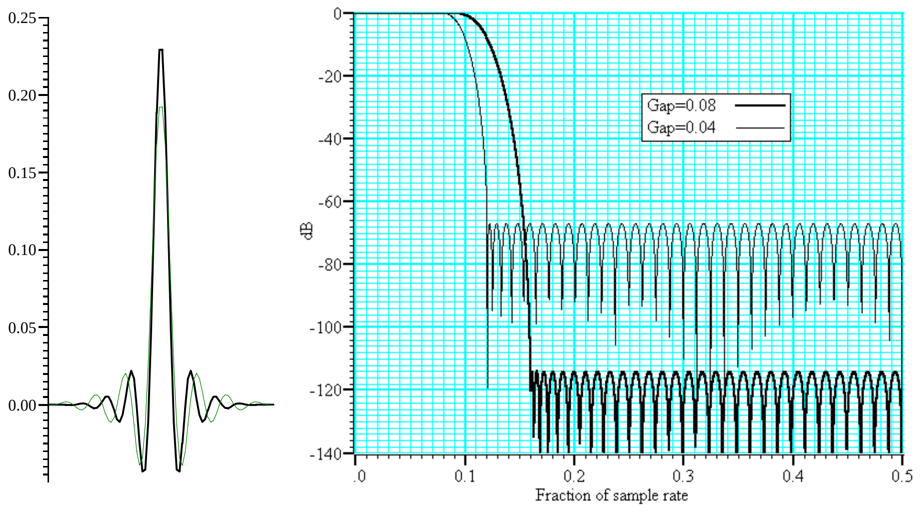
Users of the `FIRMake()` script command must specify frequencies in terms of fractions of the sample rate from 0 to 0.5. For example, if you were sampling at 10 kHz and you wanted to refer to a frequency of 500 Hz, you would call this 500/10000 or 0.05.

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



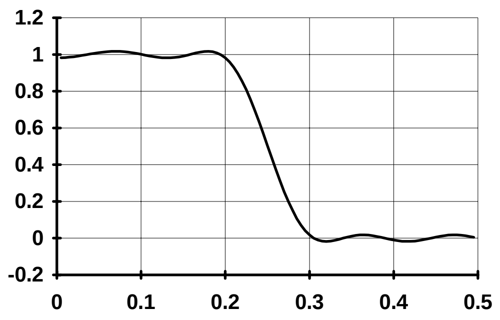
The frequency response is shown in dB, which is a logarithmic scale. A ratio *r* is represented by 20 $\log_{10}(r)$ dB. A change of 20 dB is a factor of 10 in amplitude, 6 dB is approximately a factor of 2 in amplitude. The graph shows that a frequency in the stop band is attenuated by over 110 dB (a factor of 300,000 in amplitude with respect to the signal before it was filtered).

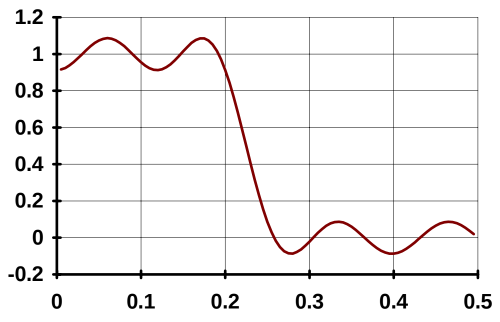
Because we didn't specify what happened between a frequency of 0.08 and 0.16 of the sampling rate, the optimisation pays no attention to this region. You might ask what happens if we make this transition gap smaller. The lighter line in the graph shows the result of halving the width of the gap by making 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 -π/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
```

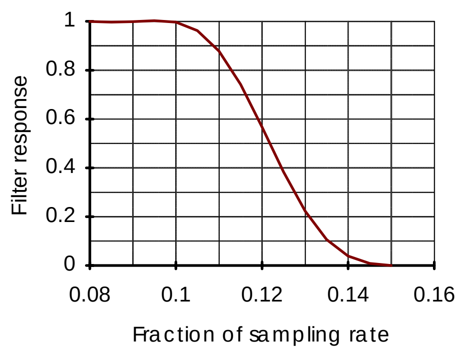
This is much closer to the filter we wanted. You might wonder if there is a formula that can predict the number of coefficients based on the filter specification. There is no exact relationship, but the following formula, worked out empirically by curve fitting, predicts the number of coefficients required 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).



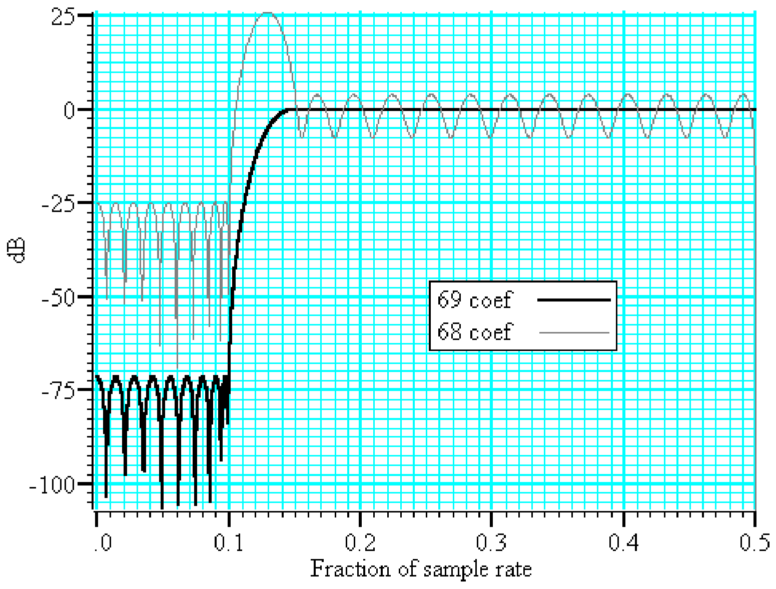
If we use the analogue filter definition of corner frequency, we see that we have produced a filter that passes from 0 to 0.115 of the sampling rate, and we wanted from 0 to 0.1, so we can move the corner frequency back. This will increase the attenuation in the stop band, and reduce the filter ripple, 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

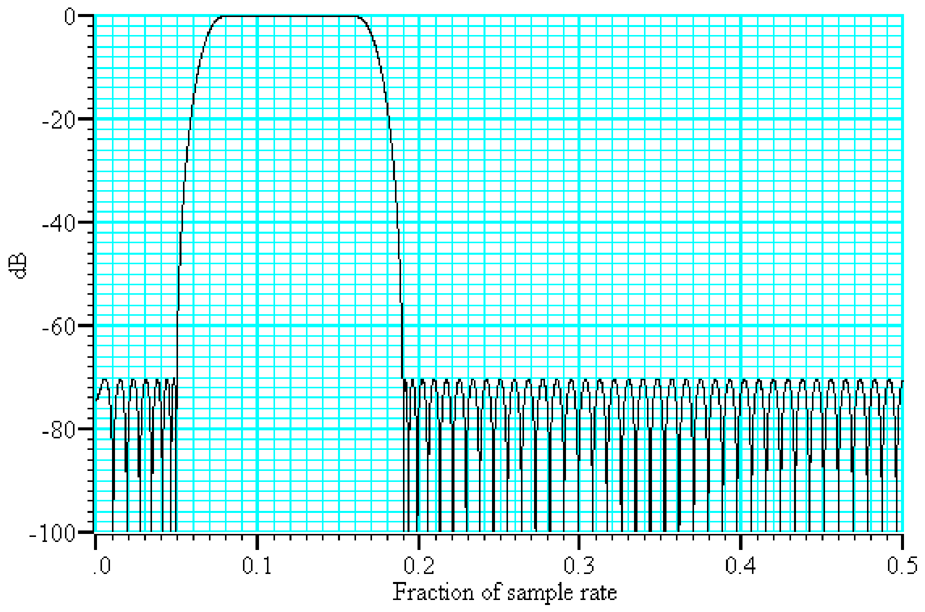
You can define up to 10 bands. However, it is unusual to need more than three. The most common cases with three bands are called band pass and band stop filters. In a band pass filter, you set a range of frequencies in which you want the signal passed unchanged, and set the frequency region below and 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[]);
```

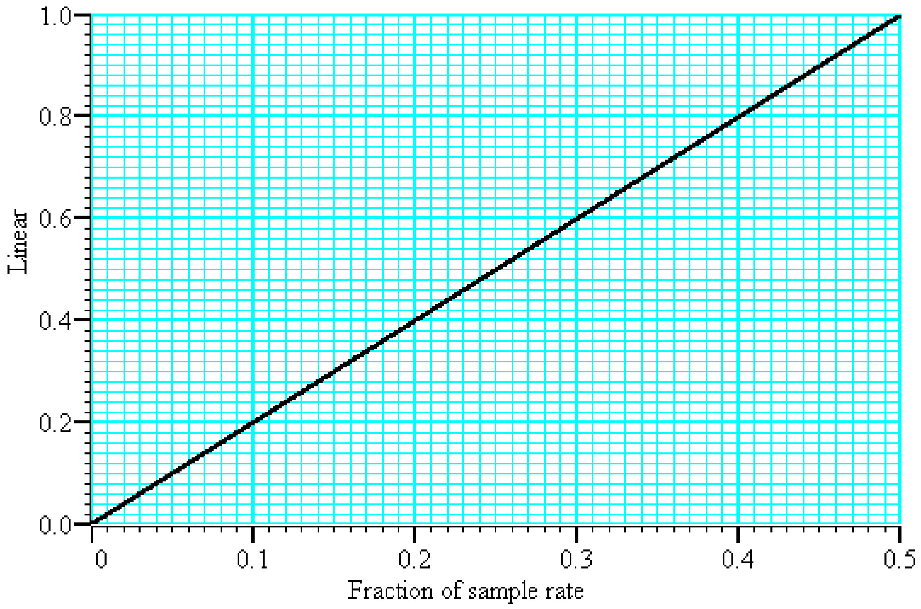
Band pass filter with 111 coefficients



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:

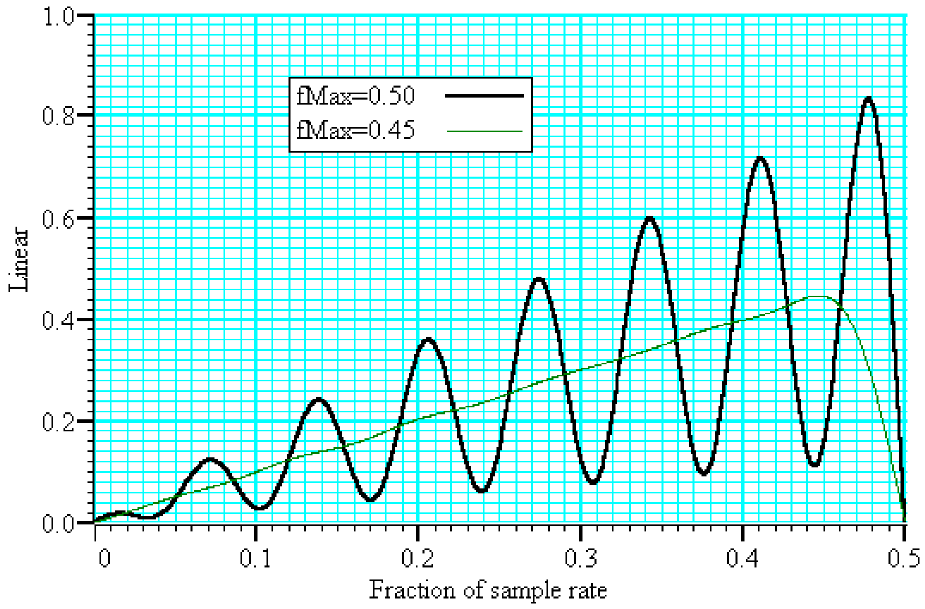
```
' dB    the proportional ripple expressed in dB
' f     the highest frequency set in the band
' even% Non-zero if you want an even number of coefficients
func NCoefDiff(dB, f, even%)
if (f<0) or (f>0.5) then return 0 endif;
f := 0.5-f;
var n%;
if (even%) then
   n% := (dB+43.837*f-35.547)/(0.22495+29.312*f);
   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.

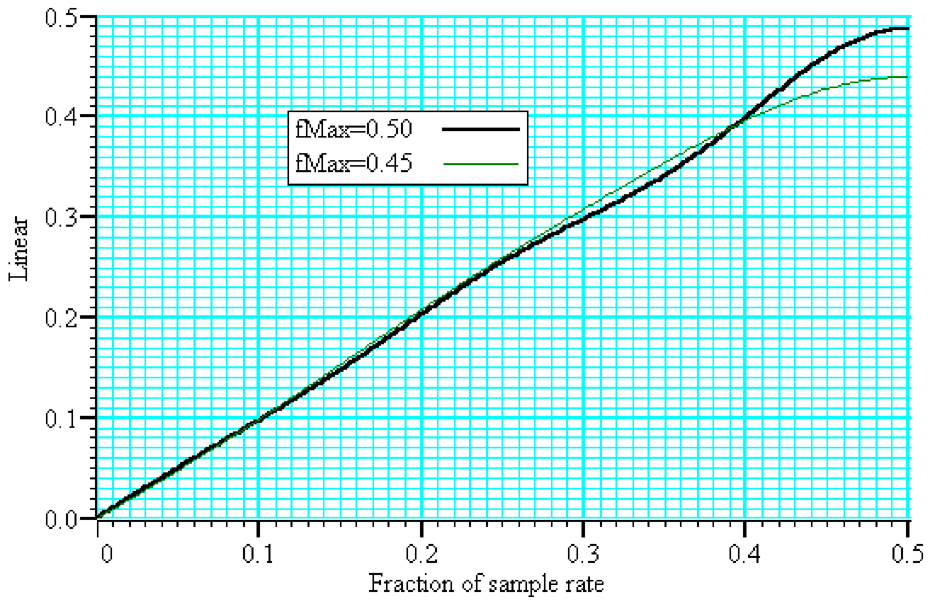
The graph below shows the effect of setting an odd number of coefficients when generating a differentiator that spans the full frequency range. The second curve shows the improvement when the maximum frequency is reduced to 0.45.

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 $-\pi/2$. 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 Signal and 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 Signal, 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.

20 Programmable Signal Conditioners

Signal can control programmable signal conditioners using the computer serial line ports and can use the signal conditioners to alter input signal gains, offsets or filtering before or during sampling. Three types of signal conditioner are supported: the CED 1902 Mk III, the Power1401 with programmable gains option and the Axon Instruments CyberAmp.. When you install the Signal software you can chose to install support for one of these types of signal conditioner or you can chose not to install any conditioner support at all. You can not choose to install support for more than one type of conditioner and only one type of conditioner can be used at any time. You can open the conditioner control panel from either the sampling configuration dialog port setup page or from the Sample menu.

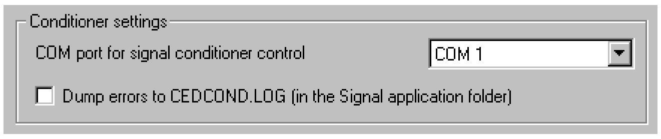
What a signal conditioner does

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 and filtering. With the Power1401 with programmable gains option only the gain may be set from within Signal.

You should consult the documentation supplied with your signal conditioner to determine the full range of its capabilities.

Serial ports

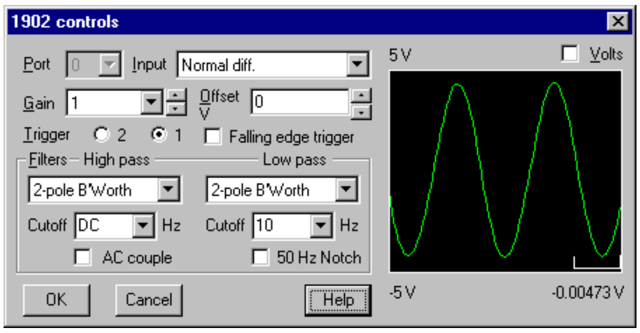

Conditioner settings
COM port for signal conditioner control: COM 1
☐ Dump errors to CEDCOND.LOG (in the Signal application folder)

Most signal conditioners use a serial line connection for software control. The serial port, if any, used to communicate with signal conditioners is set in the Edit menu Preferences dialog, in the Conditioner page. In addition to using serial line ports built into your computer you can also make use of serial lines provided by plugging an adaptor into a USB socket. USB serial lines generally work fine but you do need one which always appears as the same COM port, or where you can set the COM port as which it appears.

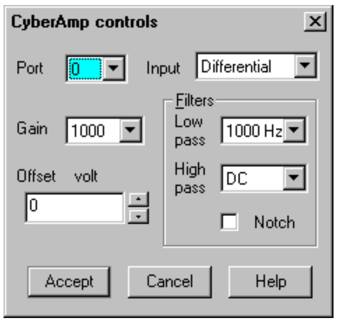
Check the Dump errors to CEDCOND.LOG box if you are having problems using your conditioners; this will cause a log file detailing communications attempts to be created which will allow the cause of problems to be determined. Leave this checkbox clear if you are not having any problems connecting to signal conditioners – it's intended for troubleshooting only.

Control panel

The signal conditioner control panel is in two halves. The left-hand side holds the controls that change the conditioner settings, the right-hand side displays data from the conditioner. Signal omits the right-hand side while sampling data or if the 1401 interface is not available for any reason.



If the right-hand side 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 1401 ADC port number whose conditioner settings are being adjusted. Only ports for which a conditioner was found are shown.

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 dialog field sets the gain to apply to the signal selected by the Input field. Signal tracks changes of gain (and offset) and will change the channel scaling factors in the ports configuration to preserve the y axis scale. To make the best use of the accuracy of the 1401 family, you should adjust the gain of your signal so that the maximum input signal does not exceed the limits of the data displayed on the right of the control panel.

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.

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).

If you are using a modern 1902 that uses digital filtering, you can set the low-pass filter type (from 2-pole and 3-pole Butterworth and 2-pole and 3-pole Bessel) as well as set the filter corner frequency.

High-pass filter A high-pass filter reduces low-frequency components of the input signal. The high-pass filters area is 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 that 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.

Again, if you are using a modern 1902 that uses digital filtering, you can set the high-pass filter type (from 2-pole and 3-pole Butterworth and 2-pole and 3-pole Bessel) as well as set the filter corner frequency.

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).

Reset calibration The Reset Calib. button resets the calibration information to show raw volts taking into account the current gain and offset. The units for the calibration will be set to V and the port Full and Zero values adjusted as appropriate. On the CyberAmp, this option will use the 'native' calibration information and units specified by a SmartProbe, if present.

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

The 1902 provides two conditioned trigger inputs, and one output. These controls select which of the trigger inputs is used to generate the output and, in a 1902 mk IV, allow selection of the trigger input polarity.

Filter type

A 1902 mk IV with digital filters has extra fields that allow you to set the low-pass and high-pass filter types. You can choose Butterworth or Bessel characteristics and 2 pole or 3 pole filters.

Setting the channel gain and offset

If you change the gain or offset in the control panel, Signal will adjust the port Full and Zero values in the sampling configuration to compensate so as to keep the y axis showing the correct values. This means that if you change the gain, the signals will still be correctly calibrated 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 ADC port in the Ports setup page.
3. Press the Conditioner button to open the conditioner control panel.
4. Adjust the gain to give a reasonable signal. Make a note of the gain *G* you have set.
5. Close the signal conditioner control panel.
6. Set the Units field of the Channel parameters to uV.
7. Set the Full field to 5000000/*G*.

You only need do steps 6 and 7 once. Any subsequent change to the conditioner gain will adjust the channel Full value to leave the units in microvolts.

For the more general case imagine you have a transducer that measures some physical quantity (Newtons, for example) and it has an output of 152.5 Newtons per V. If you wanted the y axis scaled in Newtons, you would replace steps 6 and 7 above with:

6. Set the Units field of the Channel parameters dialog to N.
7. Set the Full field to (5 * 152.5)/*G*.

To work this out you must express the transducer calibration in terms of Units per Volt (in this case Newtons per Volt), multiply this by 5 to get the Full value at five volts, then divide it by the gain of the conditioner.

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

Signal normally expects the first channel of signal conditioning to be connected to 1401 ADC port 0, the second to port 1 and so on. Connect the conditioner output BNC (labelled Amp Out on the 1902, and OUT on the CyberAmp) to the relevant 1401 input. This arrangement can be adjusted to start with another port, but in all cases the conditioner channel number must match the ADC port to which it is connected.

Most 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 folder 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, Signal writes a log file called `CEDCOND.LOG` to the current directory when it initialises the conditioner.

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	-	-

21 Amplifier Telegraphs

If you are using an external amplifier to scale your signals before they are sampled by the 1401, you can encounter problems if you change the signal gain while sampling – the amplitude of the sampled data changes and makes the signal calibrations incorrect. Signal can avoid this problem by using amplifier telegraphs to determine the current amplifier settings and dynamically adjusting the signal calibration to match. Amplifier telegraphs are signals, usually analogue voltages, from an amplifier that signal the current amplifier settings, such as gain and offsets. By collecting and interpreting amplifier telegraphs, Signal can automatically adjust for changes in the amplifier settings to maintain signal calibration.

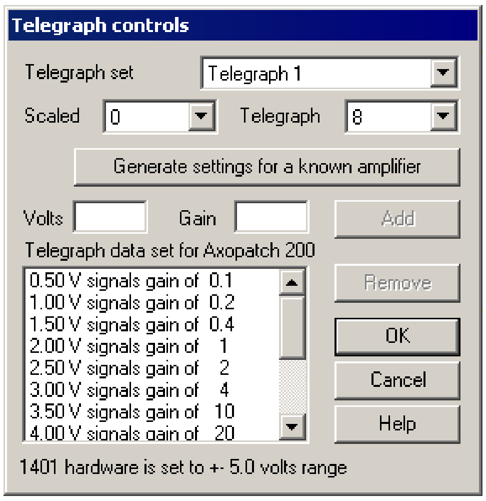
Signal currently supports two types of telegraph signal; a standard telegraph using one or more analogue signals that are sampled using the 1401 interface, and an auxiliary mechanism provided by a customisable DLL that can be installed with Signal. Currently, only one type of auxiliary telegraph DLL is available; this supports the Axon Instruments MultiClamp 700 range of amplifiers. If you do not install support for an alternative amplifier telegraph then standard telegraphs using 1401 inputs will be available.

Standard 1401 telegraphs

The standard 1401 telegraph support uses 1401 ADC ports to sample telegraph signals (analogue voltages) generated by the amplifier. A lookup table is used to convert the telegraph signal voltage to an amplifier gain which is used to adjust the relevant signal calibration. Up to four separate telegraph signals can be defined, each controlling a separate sampled channel. Setting up standard 1401 telegraphs consists of defining the ADC ports involved and setting up the lookup table used to convert telegraph voltages to gain settings.

1401 telegraph configuration

The 1401 telegraph configuration dialog can define up to four ADC ports whose scaling depends upon amplifier telegraph outputs. For each of these ports, you set a port from which the telegraph signal is read and a list of telegraph voltages and the corresponding amplifier gains. Select between the four separate telegraph settings using the Telegraph set field.



Telegraph data set for Axopatch 200

```
0.50 V signals gain of  0.1
1.00 V signals gain of  0.2
1.50 V signals gain of  0.4
2.00 V signals gain of   1
2.50 V signals gain of   2
3.00 V signals gain of   4
3.50 V signals gain of  10
4.00 V signals gain of  20
```

1401 hardware is set to +- 5.0 volts range

The Scaled item sets the ADC port whose scaling is controlled by a telegraph signal, set this to None to disable use of this telegraph set. The Telegraph item sets the ADC port from which to read the corresponding telegraph signal.

The button labelled 'Generate settings for a known amplifier', if pressed, provides a list of common amplifiers which provide telegraph signals. Selecting an amplifier from the list will generate telegraph data matching that amplifier. If your amplifier is not in the list please contact CED and we will try to add it – our thanks to John Dempster for providing much initial amplifier information.

If your amplifier is not in the list of known types you can use the Volts and Gain controls to directly enter a list of telegraph signal voltages and corresponding amplifier gains. For each amplifier gain setting enter a voltage value and a corresponding gain, then press Add to add this pair of values to the list. To remove a pair of values from the list simply select them by clicking with the mouse and then press Remove.

The checkbox for a 10 volt system present in earlier versions of this dialog has been replaced by an indicator of the voltage range in use. The 1401 voltage range is now set by an item in the sampling preferences.

The telegraph voltage values are not affected by the Full and Zero calibration values set for the telegraph ADC port, they are raw voltages as read from the 1401 inputs and converted to actual volts using the ADC voltage range in use. The scaled port should be calibrated (in the main ports configuration page) as though the external amplifier **gain is set to unity**; the telegraphed gains with then be applied to this 'basic' calibration. It is very important that you enter the value for unity amplifier gain, not whatever gain you happen to have in use at the moment. During sampling, the telegraph signals are read at the start and end of each sampling sweep and used to adjust the signal scaling to match the amplifier.

MultiClamp 700 telegraphs

The MultiClamp 700 amplifiers supplied by Axon Instruments have two sections (referred to as channels) within them, each of which has two outputs, giving four separate signals that can be sampled. The MC commander software controlling the amplifier generates telegraph information telling interested applications about changes to amplifier gain settings and other settings such as the voltage or current clamp mode and external command sensitivity. Using the MultiClamp 700 telegraph configuration dialog you can configure Signal so that it can receive and make use of this telegraph information.

In addition to signalling changes to the amplifier gain, the MC700 control software provides information about other aspects of the amplifier setup. For each channel in the amplifier, Signal can retrieve:

The primary output's low-pass filter settings
The secondary output's low pass filter settings
The membrane capacitance
The series resistance
The external command sensitivity

All these (for both channels) are retrieved for every frame and the values obtained placed in the first ten user frame variables in the sampled data file. These values can be retrieved using the Signal script language.

Integration with Signal channel scaling

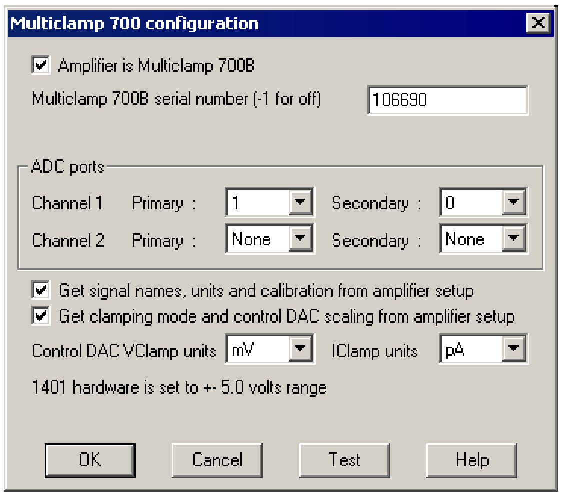
The basic MC700 support uses the information provided by the MC700 control software to adjust the port scaling factors to match the channel gains set on the amplifier in the same way as for 1401-based telegraphs. The MC700 support is also able to set the initial port scale factors, name and units to match the MC700 settings, ensuring that all information for ports sampled from MC700 outputs match the amplifier settings.

Integration with Signal clamping support

Extra MC700 support features integrate the Signal support for clamping experiments with the MC700 amplifier setup. This additional mechanism selects between voltage-clamp and current-clamp according to the amplifier controls and in addition sets up the scaling and units of the DAC used for external amplifier control. This ensures that all relevant controls and calibrations are correct at the start of a clamping experiment.

MultiClamp 700 telegraph configuration

The MultiClamp 700 telegraph configuration dialog is used to configure the Signal software so that it can communicate with the MultiClamp Commander software controlling the amplifier and to define how you want information from the amplifier to be used.



Multiclamp 700 configuration

☑ Amplifier is Multiclamp 700B

Multiclamp 700B serial number (-1 for off): 106690

ADC ports

Channel 1	Primary :	1	Secondary :	0
Channel 2	Primary :	None	Secondary :	None

☑ Get signal names, units and calibration from amplifier setup

☑ Get clamping mode and control DAC scaling from amplifier setup

Control DAC VClamp units: mV   IClamp units: pA

1401 hardware is set to +- 5.0 volts range

At the top of the dialog is a checkbox to select the type of amplifier and information used to identify the amplifier. Set this according to the type of amplifier you are using. For the 700A you will also have to specify the COM port used to control the amplifier and the Axon bus ID for the amplifier, for the 700B you have to specify the serial number of the amplifier. If you do not want to make use of the MC700 telegraph system in this sampling configuration, set the bus ID to Do not use (for the 700A) or set the serial number to –1 (for the 700B).

Below the amplifier identification section is an area defining the 1401 ADC ports that are connected to the amplifier. Each MC700 amplifier has two channels, each of which has two outputs (called Primary and Secondary for the 700B, Scaled and Raw for the 700A). Set the ADC ports on the 1401 that are connected to the various amplifier outputs, leaving any amplifier outputs that are not being used set to None.

Automatic signal calibration

The checkbox labelled 'Get signal names, units and calibration from amplifier setup' is used to allow automatic signal definition & calibration from the amplifier up. When set, ports connected to MC700 outputs are automatically set up using information retrieved from the amplifier. The information retrieved is the port name and units (depending upon what signal you have routed to the relevant MC700 output) plus both the port calibration values. This is the easiest way to use MC700 telegraph data - it is generally recommended as it will force channels sampled from MC700 outputs to be correct.

If this checkbox is left clear, then the name, units and calibration entered in the Signal port setup dialog (from the Port Setup page in the sampling configuration) are used. You can set the channel name and units to anything you wish, the port Zero value should be set to 0 and the Full scale value should be set to the value of a full-scale signal with the MC700 amplifier **gain set to unity**. It is very important that you enter the value for unity amplifier gain, not whatever gain you have in use at the moment. The full scale value is the value corresponding to a +5 volt (or +10 if appropriate) signal on the 1401 input.

Automatic clamp setup

The checkbox labelled 'Get clamping mode and control DAC scaling from amplifier setup' is used to allow automatic setup of the Signal clamp system from the amplifier settings. When set (the checkbox for automatic signal calibration must also be set for this option to be available) clamp setups within the Signal software that match a MC700 channel will be set to match the amplifier channel settings at the start of sampling.

In order for a clamping setup to match a MC700 channel and thus be set up using amplifier information, the port connected to the Primary channel output (Scaled output for a 700A) must be used to sample the Response channel defined in the clamping setup. In addition, the port connected to the Secondary channel output (Raw output for a 700A) must be used to sample the Stimulus channel. If these criteria are met then the clamping mode is set to voltage clamp or current clamp as appropriate and the control DAC in use in that clamp setup is calibrated. The DAC scaling is set using the external command sensitivity information from the amplifier, the DAC units are set as defined using the Control DAC VClamp units and IClamp units fields in the configuration dialog.

With this option enabled you can set up a single sampling configuration containing control DAC pulse sequences suitable for both voltage clamp and current clamp (knowing which units that the DAC will use in either mode) and can quickly switch between voltage and current clamp experiments by stopping sampling, switching the MC700 amplifier mode, and beginning sampling again. You do have to keep track of which sets of DAC outputs are suitable for which mode! When editing the DAC pulses in a sampling configuration intended to be used in both clamping modes in this manner you should make sure that the full scale value currently set in the DAC calibration is the larger of the two values you expect to use, otherwise the pulses dialog will limit the pulse sizes that you can enter.

The Get MultiClamp settings button in the clamping configuration page of the sampling configuration can be used to retrieve the current MultiClamp settings and use them to set the ADC calibrations, clamping mode and DAC scalings in the sampling configuration. This allows you to see what the settings are without starting sampling and should make it easier to set up an integrated Signal/MultiClamp system.

The checkbox for a 10 volt system present in earlier versions of this dialog has been replaced by an indicator of the voltage range in use. The 1401 voltage range is now set by an item in the sampling preferences.

The button marked 'Test' at the bottom of the configuration dialog is used to test that Signal can communicate successfully with the MultiClamp Commander software and will provide some diagnostic information if the test fails. Of course, you need to be running MultiClamp Commander on the same machine for Signal to be able to communicate with it.

22 Auxiliary states devices

When sampling using multiple frame states, Signal can control external devices such as stimulators in addition to switching the 1401 outputs. This is achieved by using auxiliary states devices; external equipment that is set up in different ways according to the sampling state.

Signal auxiliary states device support provides a mechanism for controlling arbitrary equipment and setting it up in different ways according to the sampling state in use. In addition to controlling equipment settings by state, Signal is also able to provide device-specific configuration dialogs, to check the equipment health and readiness while sampling, to save the stimulation parameters used in the new data file and to save and load auxiliary states setup information to and from sampling configurations. Script language functions (SampleAuxStateParam and SampleAuxStateValue) are also provided to give access to the auxiliary device settings.

Signal currently supports two types of auxiliary states device; the Magstim range of transcranial magnetic stimulators and the CED 3304 programmable constant current stimulator. You can select which of these devices you wish to use (or none) during the Signal installation process, if you wish to change the type of auxiliary states device in use the simplest way to do this is to re-install Signal. If you need auxiliary states device support for other equipment please contact CED.

Magstim auxiliary device support

The Magstim range of transcranial magnetic stimulators are widely used to provide non-invasive neuronal and muscular stimulation. Signal provides support all three types of Magstim (the 200, BiStim and Rapid), as the software uses a serial line for control of the stimulator only the modern Magstims that provide serial line control (these have a ^{2} suffix to the model name) can be used. In addition Signal can also control a pair of Magstim 200^2 units using two serial lines.

Magstim Safety notice

Transcranial magnetic stimulators are dangerous devices capable of causing serious harm and should only be used by qualified medical practitioners. Before using a Magstim device you should read the user's manual produced by Magstim paying particular attention to the warnings and precautions section. It is your responsibility to ensure that Signal's control of a Magstim is set up in an appropriate and safe fashion for the intended use and to verify that it is operating correctly. CED is not responsible in any way for problems caused by use of this software.

introduction

For Magstim devices, the auxiliary states system sets the power output and (where appropriate) inter-pulse timing, remote control of the hardware can be disabled for specific states to allow the user to control the stimulation manually. Options are provided to cause Signal to assume independent triggering mode and to enable high resolution interval timing (for the BiStim^2) and to cause the Magstim to ignore the coil interlock switch and to enable single pulse mode (for the Rapid^2). This documentation only describes the controls available from within Signal; consult the Magstim user manuals for complete information.

In addition to using a serial-line to set the stimulation parameters, standard operation with Signal requires that the stimulation be triggered using one or more TTL pulses generated by the 1401 digital outputs. This allows the pulse timing to be precisely controlled relative to the Signal sampling. A Rapid^2 can be configured to either generate a train of pulses in response to a single trigger or to generate one pulse per trigger, allowing complete control of the patterns of stimulation. Manual triggering of the stimulator is also possible, as is triggering from other external hardware, in which case you would use the Magstim trigger to trigger the sampling sweep as well.

The Magstim support monitors the stimulator health; both waiting until the Magstim is ready for use before allowing a sampling sweep to proceed and terminating sampling if a hardware problem develops. To aid data analysis, Signal saves the Magstim stimulation parameters used in the data section variables of the sampled file. The parameters saved are the power level, the pulse interval (BiStim only), the secondary power level (BiStim^2 and dual 200^2), the pulse frequency (Rapid^2 only) and the pulse count (Rapid^2 only). These values are placed in the first user frame variables in the sampled data file. The saved values can be retrieved using the script language; later versions of Signal may make use of them directly.

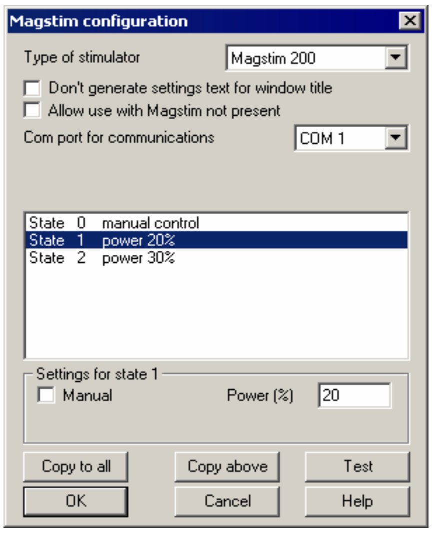
Magstim configuration

To configure a Magstim in Signal, open the sampling configuration dialog and make sure that Multiple states is enabled in the General tab. Select the States tab and make sure that you are in *Dynamic outputs* or *Static outputs* mode. If Magstim support is installed, there will be a button labelled Magstim. If this button is not present, reinstall Signal and make sure that you select Magstim auxiliary states device support. If the button is present, but disabled, you are in *External Digital* states mode which does not operate with auxiliary states devices, change the mode to enable the button. Click the button to open the Magstim configuration dialog.

It is meaningful to use the Magstim support without using multiple states, but this is not directly supported because the states system is so closely linked to the Magstim support. If you want to make use of Magstim support without using multiple states (so that the pulse parameters are recorded and Magstim health checks are made, but with complete manual control) you should turn on multiple states, set only one extra state, set state zero to use manual control, and only make use of state zero while sampling. This will allow you access to the Magstim configuration dialog to set the basic Magstim parameters.

Type of stimulator

This selector is used to select the Magstim type. You should set this to the type of stimulator you will be using or to Do not use if you do not want to make use of the Magstim support in this sampling configuration. If you set the device type incorrectly errors during sampling will almost certainly result.


Magstim configuration

Type of stimulator: Magstim 200
☐ Don't generate settings text for window title
☐ Allow use with Magstim not present
Com port for communications: COM 1

State 0 manual control
State 1 power 20%
State 2 power 30%

Settings for state 1
☐ Manual   Power (%) 20

Don't generate settings text for window title

This checkbox prevents the Magstim system from generating text showing the power level and other settings that Signal can display in the title bar of a sampling document. It is often useful to see what the Magstim settings are, but some users find it distracting. A separate checkbox in the Signal sampling preferences must also be set to allow use of this text.

Allow use with Magstim not present

This checkbox is provided to allow sampling with a configuration that uses a Magstim without Signal having a connection to a Magstim device, for demonstrations or training.

Com port for communications

This is a selector for the COM port used to control the stimulator, you can select any port from COM1 to COM9. As many PCs only have one COM port you may have to use a USB serial port expander to provide a spare COM port – we have found that most of these USB serial ports work satisfactorily. When a dual-200^2 device is selected, two COM port selectors are shown.

Below the basic stimulator configuration are controls that vary according to the type of stimulator selected. These are in three sections; first there are controls which govern the overall behaviour for all states, next is a display of settings for all states which is used to select a state for editing (by clicking on the information for that state) and finally an area where the settings for the selected state may be edited.

200^2 device

This is the simplest type of Magstim and does not require any controls for overall behaviour. The settings available for the individual states are:

Manual Set this checkbox for this state to be controlled by the manual controls and not by Signal.

Power This sets the stimulator power output as a percentage of the maximum available, you can set any value from 0 to 100. This control is disabled for manually controlled states.

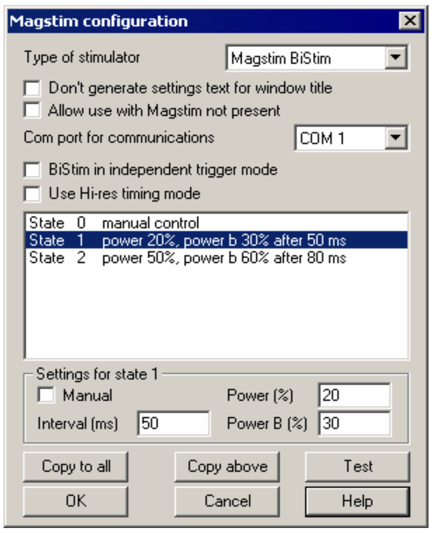
Copy... The buttons labelled Copy to all and Copy above can be used to set up many states quickly; Copy to all copies the currently selected state to all other states, while Copy above copies the current state's settings to all higher state numbers. These buttons are available for all types of Magstim.

Test The button labelled Test tests if communications with the Magstim can be established. It checks that you have the correct COM port, that the serial line is connected and that the Magstim is communicating correctly with Signal. These buttons are available for all types of Magstim.

BiStim^2 device

The BiStim^2 is more complex than the 200^2; in essence it is two 200^2 units with one slaved to the other. When a BiStim^2 device is selected two controls governing overall behaviour are shown:

BiStim in independent trigger mode This sets how Signal will expect the BiStim^2 to be configured, rather than how it will be set up by Signal. Normally, a BiStim^2 generates two output pulses, one at the time of the trigger and the second at a preset interval after the trigger. Using the controls on the front of the BiStim^2 master unit, you can set the device into independent trigger mode (referred to as IBT mode in the Magstim documentation). When this mode is in use two separate triggers are used (presumably signals from two 1401 digital outputs), one firing the main pulse and the other firing the second (Power B) pulse. Because a BiStim^2 behaves rather differently when in IBT mode it is important that you set this checkbox to match the BiStim setup you will be using.


Magstim configuration

Type of stimulator: Magstim BiStim

☐ Don't generate settings text for window title

☐ Allow use with Magstim not present

Com port for communications: COM 1

☐ BiStim in independent trigger mode

☐ Use Hi-res timing mode

```
State  0   manual control
State  1   power 20%, power b 30% after 50 ms
State  2   power 50%, power b 60% after 80 ms
```

Settings for state 1

☐ Manual

Power (%): 20

Interval (ms): 50

Power B (%): 30

Use Hi-res timing mode When this checkbox is clear, the interval between the two pulses can be set to any value between 0 and 999 milliseconds, the actual timing in the BiStim^2 will be set with a resolution of 1 millisecond. With this checkbox set, the maximum interval is reduced to 99.9 milliseconds but the interval timing resolution is improved to 0.1 milliseconds. This control is disabled if independent trigger mode is selected.

The BiStim^2 settings available for the individual states are:

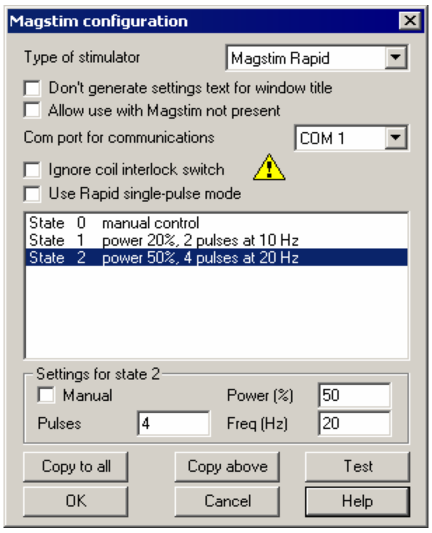
Manual Set this checkbox for this state to be controlled by the manual controls and not by Signal.

Power This sets the main (master) stimulator power output as a percentage of the maximum available, you can set any value from 0 to 100. This control is disabled for manually controlled states.

Interval This sets the interval between the two pulses, you can use any value from 0 to 999 milliseconds (or 0 to 99.9 milliseconds for Hi-res timing). Note that the BiStim cannot tolerate timing intervals between 0 and 1 millisecond in Hi-res mode, if you enter one of these values the interval will be rounded to the nearest of 0 and 1 millisecond. Setting a zero interval will switch the BiStim into simultaneous pulse mode, setting a value greater than zero switches it out of simultaneous mode. In simultaneous pulse mode both stimulators must use the same power level, so both levels are set by the main power field. This field is disabled if independent BiStim triggers are in use or for manually controlled states.

Power B This sets the second (slave) stimulator power output as a percentage of the maximum available, you can set any value from 0 to 100. This control is disabled for manually controlled states and also if a zero stimulus interval has been set.

Rapid^2 device The Rapid^2 is rather different from the other devices as it is capable of producing a train of pulses at high rates. Rather than the simple case-mounted manual controls of the other two units, the Rapid^2 has a separate control system that has to be disconnected to gain access to the serial line control port, limiting the usefulness of manual control. There are two controls governing overall behaviour:


Magstim configuration

Type of stimulator: Magstim Rapid

☐ Don't generate settings text for window title

☐ Allow use with Magstim not present

Com port for communications: COM 1

☐ Ignore coil interlock switch

☐ Use Rapid single-pulse mode

```
State  0   manual control
State  1   power 20%, 2 pulses at 10 Hz
State  2   power 50%, 4 pulses at 20 Hz
```

Settings for state 2

☐ Manual   Power (%) 50

Pulses 4   Freq (Hz) 20

Ignore coil interlock switch This disables checks on the coil interlock switch on the handle of the Magstim coil, so that the unit will fire without a button being held down. Magstim do not recommend disabling these checks as they are a safety feature.

Use Rapid single-pulse mode This turns on single-pulse mode, which allows higher power levels - up to 110%. In single-pulse mode only a single pulse is produced per trigger and the pulse train parameters are ignored. The pulse rate can be up to 100 Hz normally (depending upon the power level used – see the user manual) but if you select power levels above 100%, the maximum allowed pulse rate is forced to 0.5 Hz. With single pulse mode turned off a train of up to 1000 pulses can be generated at the specified frequency from one trigger and power levels above 100% are not available. Again, the pulse rate can be up to 100 Hz depending upon the power level used.

The Rapid^2 settings available for the individual states are:

Manual Set this checkbox for this state to be controlled by the manual controls and not by Signal. As you have to disconnect the manual controls in order to connect the serial line used by Signal, I fear that this option will not be particularly useful.

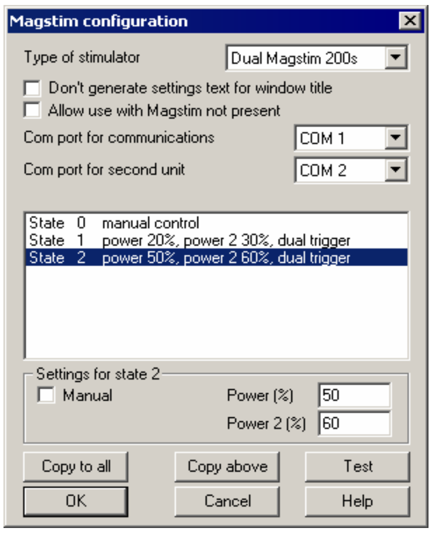
Power This sets the stimulator power output as a percentage of the maximum available, you can set any value from 0 to 100. This control is disabled for manually controlled states.

Pulses This item is only available if single-pulse mode is not in use, it sets the number of pulses, you can set any value from 1 to 1000. This control is disabled for manually controlled states or if single-pulse mode is selected.

Freq This item is only available single-pulse mode is not in use, it sets the pulse frequency in Hertz, you can set any value from 0 to 100. This control is disabled for manually controlled states or if single-pulse mode is selected.

Dual 200^2 devices

Two Magstim 200^2s can be controlled by Signal to achieve much the same effect as using a $BiStim^2$, when using this configuration two separate serial lines are used to control the Magstims and two separate trigger pulses are required (as for a BiStim in independent trigger mode). When the dual 200^2 option is selected a separate selector for the second serial-line port is shown, but as for the 200^2 there are no controls governing overall behaviour. The settings available for the individual states are:


Magstim configuration
Type of stimulator: Dual Magstim 200s
☐ Don't generate settings text for window title
☐ Allow use with Magstim not present
Com port for communications: COM 1
Com port for second unit: COM 2

```
State  0   manual control
State  1   power 20%, power 2 30%, dual trigger
State  2   power 50%, power 2 60%, dual trigger
```

Settings for state 2
☐ Manual
Power (%): 50
Power 2 (%): 60

Manual Set this checkbox for this state to be controlled by the manual controls and not by Signal.

Power This control sets the main (master) stimulator power output as a percentage of the maximum available, you can set any value from 0 to 100. This control is disabled for manually controlled states.

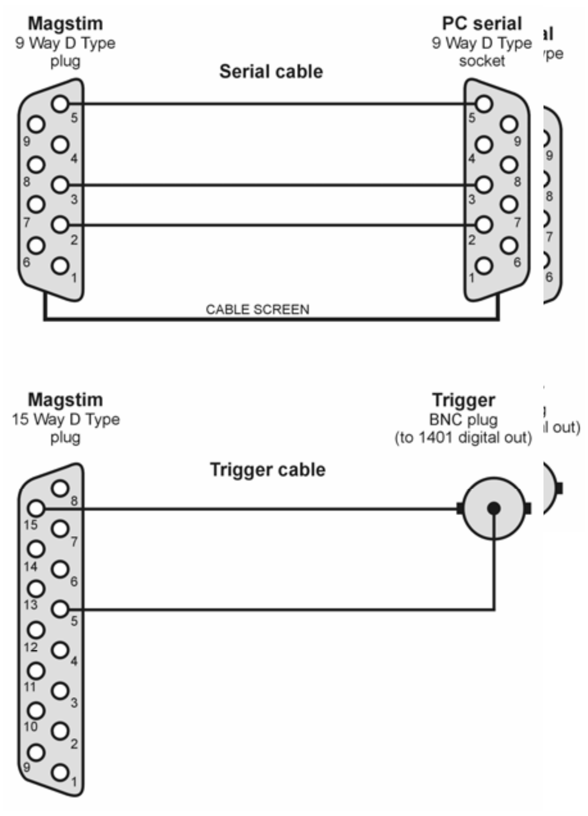
Power 2 This control sets the second stimulator power output as a percentage of the maximum available, you can set any value from 0 to 100. This control is disabled for manually controlled states.

Magstim Connections and cabling

There are two connector arrangements provided by Magstim for serial line control and external triggering of the unit. Older 200 and BiStim units have an arrangement with a 9-way serial line and separate 15-way trigger connector while newer 200 and BiStim units and all Rapids have a high-density 26-way combined serial line and trigger connector.

In both cases two separate connections need to be made; serial line control from a PC and a TTL trigger signal from the 1401. The standard high level trigger input suggested by Magstim works well with Signal and the 1401 and is used in the diagrams shown below, consult the Magstim documentation if you require a different arrangement. In the case of a Rapid stimulator, the 26-way connector is also used by the integrated control system and this will need to be disconnected before the control cable can be connected.

Cabling for 26-way combined serial and trigger connector: Cables for separate 9 pin serial and 15-way trigger connectors:



If you are not using the external trigger input to trigger the Magstim from the 1401 you will need a different cabling arrangement, for example to take the Magstim low-going trigger out signal (on pin 8 of the 26-way connector) and use that as an input to the 1401 to trigger the 1401 sweep. Consult your Magstim documentation for more information.

Notes on Magstim use

A straightforward arrangement to use a Magstim with Signal would be as follows:

1. Connect the serial port of your computer to the Magstim serial line input using the appropriate serial line cable. Connect the trigger BNC plug to the 1401 digital output port 0 BNC socket found on the front of all modern types of 1401. If you are using a 1401plus the digital output pulse is available from the 25-way digital output socket on the front of the 1401.

2. In the outputs page of the sampling configuration, make sure that digital output bit zero is enabled for use. Using the pulses configuration dialog set the initial level of digital output bit zero to 0 and place a pulse (which will be high-going) in the outputs at the time when you want the Magstim to fire. This output pulse should be at least 10 microseconds long – 1 millisecond works well.

3. The Magstim support uses Signal multiple states sampling, which should be set up in dynamic outputs mode. You can use any number of extra states, each extra state providing separate Magstim settings. Each set of pulse outputs should include digital output pulses to trigger the Magstim along with any other outputs required. The Magstim support will work correctly with manual control of the states or with any style of automatic states sequencing including protocols.

When Signal begins sampling with the Magstim support enabled, it checks for a correctly functioning Magstim device as part of the process of initialising for sampling. If a Magstim is found then Signal will carry out the initial configuration of the Magstim and arm the device. While sampling is in progress, Signal will set up the Magstim using the current state data before each sweep, it will then delay the start of each sweep until the Magstim reports that it is armed and ready. At the end of each sweep the Magstim health is checked to make sure it is OK. Note that the checks on Magstim readiness can impose a significant extra inter-sweep delay though steps have been taken to minimise this. When sampling finishes normally, the Magstim is disarmed and remote control disabled.

If the Magstim coil temperature rises too high, Signal will stop sampling. Once the coil temperature has dropped sufficiently you can press ‘More’ on the sampling control panel to resume sampling again.

While sampling is in progress, Signal continuously maintains communications to prevent the Magstim from disabling remote control and disarming itself. If Signal ceases to communicate with the Magstim because it has encountered a significant problem (“crashes”), the Magstim will disarm itself automatically within 1 second, but this safety feature only applies if manual control has not been selected by Signal beforehand.

If manual control is selected, the Magstim will disarm itself spontaneously only after 60 seconds have passed without a stimulus trigger, so it is your responsibility to make sure the Magstim is disarmed if manual control is used and Signal encounters a significant problem. The Rapid stimulator does not appear to disarm itself after 60 seconds in this manner and must be disarmed manually if Signal fails during sampling.

Because Signal needs to communicate frequently with a Magstim to stop it disarming, scripts that operate while Signal is sampling need to be correctly designed. If a script carries out a lengthy operation without yielding or using a toolbar or dialog to allow control to pass back to the operating system, this may interfere with Magstim communications and cause the unit to disarm.

CED 3304 auxiliary device support

The CED 3304 current stimulator can be used to generate controlled constant-current stimulations of up to 10 milliamps and is fully controlled by Signal, allowing you to set a different level of current for each state. Signal controls the CED 3304 programmable constant current stimulator using a serial line or USB port to interact with the hardware.

CED 3304 Safety notice

The CED 3304 current stimulator can generate outputs of up to 90 volts, which lies within the potentially lethal range. Before using the device you must read the *CED 3304 Owners Handbook* paying particular attention to the warnings and precautions section. In particular, you are reminded that the CED 3304 is not qualified for human use. It is your responsibility to ensure that Signal’s control of a CED 3304 is set up in an appropriate and safe fashion for the intended use.

Introduction

This documentation only describes the CED 3304 controls available from within Signal; for a complete understanding of the effect these have upon the behaviour of the stimulator you should consult the *CED 3304 Owners Handbook*.

The CED 3304 produces current pulses in one of four current ranges: 10 and 100 microamps and 1 and 10 milliamps. The current range is selected by a manual switch on the CED 3304. You specify a current range when you define the stimulus settings in Signal; Signal will not let you sample if the wrong range has been set on the CED 3304.

The timing of the 3304 current pulses is controlled by a TTL signal connected to the front panel Trigger input. Signal can control the timing of the current pulses with one of the 1401 digital outputs in *Dynamic outputs* mode, or external equipment can control the pulsing in *Static outputs* mode. For safety reasons, the CED 3304 will switch off a pulse after a preset limit is reached – see the 3304 documentation for details of how to control this limit.

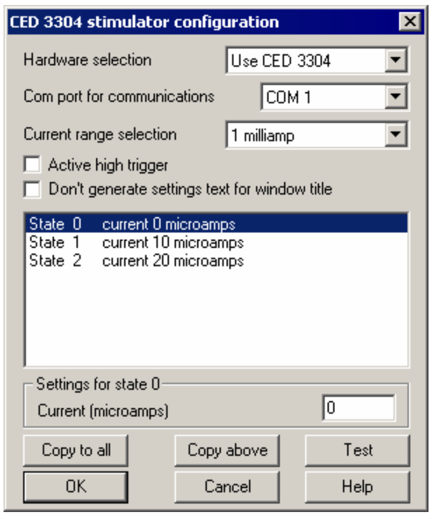
Signal monitors the stimulator health; waiting until it is ready for use before allowing a sampling sweep to proceed and terminating sampling if a hardware problem develops. To aid data analysis, Signal saves the stimulation current in the data section variables of the sampled file. This value, in uA, is placed in the first user frame variable in the sampled data file. The saved values can be retrieved using the script language; later versions of Signal may make use of them directly.

CED 3304 support configuration

To configure the CED 3304 in Signal, open the sampling configuration dialog and make sure that Multiple states is enabled in the General tab. Open the States tab and make sure that you are in *Dynamic outputs* or *Static outputs* mode. If CED 3304 support is present, there will be a button labelled CED 3304. If this button is not present, reinstall Signal and make sure that you select CED 3304 stimulator support. If the button is present, but disabled, you are in *External Digital* states mode. Change the mode to enable the button. Click the button to open the CED 3304 configuration dialog.

From the configuration dialog you can enable use of the CED 3304, set the serial line port used to communicate with the hardware and other overall parameters, define the current settings for each state in use, set the active level for the Trigger input and test for successful communications with the CED 3304.

At the top of the dialog are settings that apply to all states. In the centre is a list of the currents set for each state. Below this is a field to edit the current for the selected state. At the bottom are buttons to copy settings, get help, test the CED 3304 and accept the dialog settings.


CED 3304 stimulator configuration

Hardware selection: Use CED 3304
Com port for communications: COM 1
Current range selection: 1 milliamp
Active high trigger
Don't generate settings text for window title

State 0 current 0 microamps
State 1 current 10 microamps
State 2 current 20 microamps

Settings for state 0
Current (microamps): 0

Hardware selection Set this to Use CED 3304 to use the stimulator or to Do not use if you do not want to make use of the 3304 support software in this sampling configuration.

Com port for communications Signal connects to the CED 3304 either via a serial communication port, or through a USB port, which is setup as a virtual COM port. This field sets the COM port that the CED 3304 is connected to. If you use the USB connection, follow the instructions at the start of the Operation chapter of the *CED 3304 Owners Handbook* to obtain a suitable device driver for your operating system.

Current range selection This field sets the maximum current that you can enter for each state. You can select from 10 and 100 microamps and 1 and 10 milliamps. This control matches the current range rotary switch on the front of the CED 3304 but does not override its setting – the actual range switch setting must match the setting in this dialog or an error will be generated when sampling is begun.

Active high trigger Check the box for an active high trigger (current is generated while the CED 3304 Trigger input is at a high TTL level). Clear the box for active low operation (current generated when the Trigger input is at a low TTL level). When you are driving the trigger input from the 1401 digital outputs, it is usual to set an active high trigger.

Don't generate settings text for window title This checkbox prevents the 3304 system from generating text showing the current level that Signal can display in the title bar of a sampling document. It is often useful to see what the 3304 settings are, but some users find it distracting. A separate checkbox in the Signal sampling preferences must also be set to allow use of this text.

Settings for state... This field displays and lets you edit the desired current for the state selected in the list of states. You can enter any value (in microamps) from zero to the maximum available, for example in the 10 uA range you could enter: 0 or 8.234 or 10.

Copy... The buttons labelled Copy to All and Copy above can be used to set up many states quickly; Copy to All copies the currently selected state to all other states, while Copy Above copies the current state's settings to all higher state numbers.

Test This button tests if communications with the CED 3304 can be established. It checks that you have the correct COM port, that the serial line is connected and that the CED 3304 is operating correctly.

The Help, Cancel and OK buttons have their usual meanings.

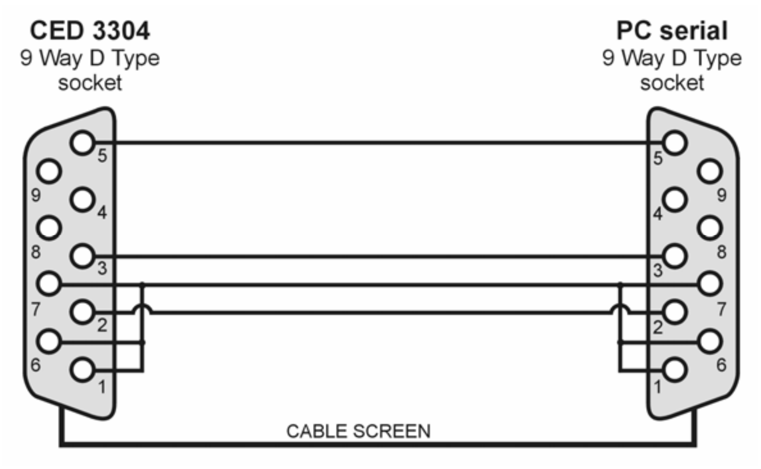
Simple use If you want to use the CED 3304 from Signal very simply with a single current setting, you must still enable multiple states. Set one extra state (so states are enabled), and set the desired current for state zero, and then sample using state zero only. This gives you access to the configuration dialog to set the basic CED 3304 parameters and enables checking of CED 3304 behaviour during sampling and the recording of the stimulation settings.

CED 3304 Connections and cabling

The CED 3304 uses a USB A-B cable or a 9-pin serial-line cable (both are supplied with the unit) for control. Use one cable or the other. If you plug in both, the USB connection will take precedence. See the Operation chapter of the *CED 3304 Owners Handbook* for more information about the control cables.

The front panel Trigger input should be connected to the 1401 digital outputs or other TTL pulse source using a standard BNC cable. The front panel Data and Clock inputs are not used with Signal and should be left unconnected.

3304 serial cable



Notes on 3304 use

A straightforward arrangement to use a CED 3304 with Signal, assuming we use digital output bit 0 to control the current pulse timing, would be as follows:

1. Connect your computer to the CED 3304 using the either a serial or use a USB connection. Connect the CED 3304 front panel Trigger BNC plug to the 1401 digital output 0 BNC socket found on the front of all modern types of 1401. If you are using a 1401*plus* the digital output pulse is available from the 25-way digital output socket on the front of the 1401.
2. In the Outputs page of the sampling configuration, make sure that digital output bit zero is enabled for use. Using the pulses configuration dialog, set the initial level of digital output bit zero to 0 and place a pulse (which will be high-going) in the outputs at the time when you want the CED 3304 to fire. This output pulse should be as long as the required stimulation.
3. The CED 3304 support is designed to work with Signal multiple states sampling, which should be set up in *Dynamic outputs* mode. Set one extra state for each output current setting you require and each state should generate digital output pulses to trigger the CED 3304 plus any other outputs required. The CED 3304 support will work with manual control of the states or with any style of automatic states sequencing including protocols.

When Signal begins sampling with the CED 3304 support enabled it checks for a correctly functioning device. If a CED 3304 is found, Signal will check the range switch setting, initialize the device and enable the trigger. While sampling is in progress, Signal sets the output current using the state data before each sweep. At the start and end of each sweep the CED 3304 health is checked to make sure it is OK. When sampling finishes, the CED 3304 trigger is disabled.

Because of the extra overhead in setting up current levels for each sweep and checking that the device is operating correctly, you may find that the maximum sweep rate with the CED 3304 is reduced.

23 MATLAB file export

Using Signal versions 4.02 onwards you can export data and XY view data to MATLAB by selecting MATLAB data (*.mat) as the Save as type from the File menu Export As dialog. To enable this functionality, you must select the MATLAB file export option when installing Signal. You do not need a copy of MATLAB to be installed on your computer.

Once you have selected MATLAB data, set a file name and click on Save in the Export As dialog. Signal will then proceed to dialogs where you choose the data to export and the format to export it in. The details of the following dialogs depends on the source view type.

If you have a problem getting the export to work, check that the file `MatExp.sxl` is present in the `Signal4\Export` folder. A whole bunch of other files are also needed in this folder, but if this one is there, it is likely that the rest have made it too. If the file is not present, you need to reinstall Signal and make sure that you select the option to install MATLAB file export support.

Mat-files represent a collection of workspace variables inside MATLAB. Signal creates mat-files containing one variable holding aggregated data for all waveform channels and separate variables for each other data channel that is exported. The variables are structures containing multiple fields that hold the channel data.

Workspace variable naming

The workspace variables in a mat-file must have names that are both unique and legal. If a variable name is not unique it will overwrite the existing workspace data with the same name. MATLAB cannot read variables with illegal names. Signal mat-file export creates variables with names based upon the source view file name and the channel name. You can independently select if either of these is to be used. This produces variable names that are both useful and unique. The default setting is to use the source view file name but not the channel name. The variable name settings are used to build the variable name as follows:

If the source view name is to be used, it is placed at the start of the variable name followed by an underscore character '_'. Following this a channel identifier is added. The identifier is always '`wave_data`' for the aggregated waveform channels, for other channels it is either the channel title (if using the source channel name) or Signal builds a name using '`Ch`' followed by the channel number. For example, when exporting a view called `Expt1` containing 2 waveform channels and a keyboard marker channel called '`Keyboard`' we would get the following MATLAB variable names:

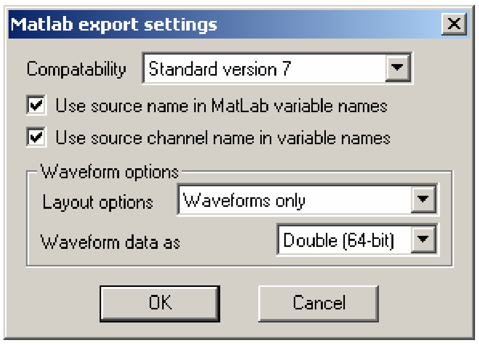
Using source view and channel name:	`Expt1_wave_data, Expt_Keyboard`
Using source view name only:	`Expt1_wave_data, Expt1_Ch3`
Using channel name only:	`wave_data, Keyboard`
Using neither name:	`wave_data, Ch3`

Having generated a name according to these rules, Signal then checks and, if necessary, modifies the name to guarantee that it is legal. The rules for a legal name are that it must not be more than 63 characters, must begin with an alphabetic character and must only contain alphanumeric characters and the underscore character '_'. Signal modifies the name by appending a '`V`' to the start of the name if it does not begin with an alphabetic character, converting all illegal characters to underscores and finally truncating if more than 63 characters long.

If despite all this you end up with variable names that are the same, the variables will overwrite each other when you read the file into MATLAB and you will only see the last variable.

Data view data

When exporting data from a file or memory view you will first get a dialog that allows you to select the channels, frames and time range that you want to export. There is also a checkbox that selects offsetting the start of the time range being exported to zero. This dialog is the same as the dialog used to select data to export to a CFS file.



Matlab export settings

Compatability: Standard version 7

[✓] Use source name in MatLab variable names

[✓] Use source channel name in variable names

Waveform options

Layout options: Waveforms only

Waveform data as: Double (64-bit)

When you have selected your channels, frames and time range you are then presented with a second dialog that controls mat-file export options. These options allow you to define how the MATLAB workspace variables will be named, what data is generated for different channel types and the format used for various types of Signal data.

Compatibility If you are using an older version of MATLAB (before version 7), you must use the compatibility field to select a suitable output format.

Use ... in variable names These two options in the dialog control how the mat-file variable names are constructed as described previously.

Waveform options These two controls specify how waveform data is handled. The Layout options item can be set to *Waveform only* or *Waveform and errors*, if the second option is selected and there are error values available these will be exported to a separate array. The Waveform data as item sets the type of data created for waveforms, it can be set to *Float (32-bit)* or *Double (64-bit)*. Floats use less memory while doubles are the most generally useful in MATLAB, it is probably best to leave this set to doubles unless you are suffering from memory problems with large amounts of data.

XY view data

For an XY view you do not get a dialog to control what data is exported; all the visible channels are exported and only data points that lie within the displayed X and Y axis ranges are used.

A dialog is provided to control how the data is exported. This is a simplified version of the data view export options dialog shown above; it only contains the compatibility selector and the two checkboxes to select the use of the source view name and channel names to create the MATLAB variable names.

Mat-file data format

A mat-file represents a collection of MATLAB variables rather than simple data values which allows for a complex representation of Signal data within MATLAB. Each Signal channel exported into a mat-file is represented by one variable, except for waveform channels which are grouped together into a single variable. The variables are structures containing fields that hold information about the channel and other fields holding the data. For example, a structure holding data from waveform channels has the fields `xlabel`, `interval` and `start` each of which is a simple (scalar) datum holding the x axis title, the sample interval in seconds and the time of the first data point. In addition there is a field called `wave_data` that is a 3D matrix holding the waveform data values and an optional field called `SD` which is a 3D matrix holding the standard deviation for the corresponding waveform points. The structure varies according to the channel type, though some fields are common to all channels.

Waveform data

Waveform data is exported as a 3-dimensional array `<points` *x* `chans` *x* `frames>` of waveform values with an optional associated array of error values. All waveform channels are exported together in the same variable. There are separate arrays of structures holding information about the channels and frames that are exported. The fields in the waveform data structure are:

`xlabel`	a string holding the X axis label, normally blank.
`xunits`	a string holding the X axis units, this is normally '`s`' even if you have opted to display time as milliseconds or microseconds.
`start`	a double holding the start time, in X axis units, for the waveform data.
`interval`	a double holding the waveform data sample interval, in X axis units.
`points`	an integer holding the number of waveform data points exported for each channel (and frame).
`chans`	an integer holding the number of waveform channels exported.
`frames`	an integer holding the number of frames exported.
`chaninfo`	an array of structures, length `chans`, holding information on the channels exported. Each structure contains the following items: `number` an integer holding the actual source file channel number. `title` a string holding the channel title. `units` a string holding the channel units.
`frameinfo`	an array of structures, length `frames`, holding information on the frames exported. Each structure contains the following items: `number` an integer holding the actual source file frame number. `start` a double holding the start time, in seconds, for the frame. `state` an integer holding the frame state code. `tag` an integer holding the frame tag (0 or 1). `sweeps` an integer holding the frame sweep count. This will be zero for frames not produced by averaging.
`values`	an array with dimensions `<points` *x* `chans` *x* `frames>` holding the waveform values. Depending upon the output options selected, this array could be double or single-precision real values.
`SD`	an optional array with dimensions `<points` *x* `chans` *x* `frames>` holding the waveform error values as standard deviations. Depending upon the output options selected, this array could be double or single-precision real values.

Marker channels

Marker channel data is represented as a 2-dimensional array of marker times and a corresponding 2-dimensional array of marker code values. Each marker channel is exported to a separate variable in the MATLAB workspace. The fields in the channel structure are:

`title`	a string holding the channel title.
`units`	a string holding the channel units, normally blank.
`resolution`	a double holding the underlying timing resolution in seconds.
`points`	an integer holding the maximum number of markers in a frame.
`frames`	an integer holding the number of frames exported.
`times`	a `<points` *x* `frames>` 2-dimensional array of doubles holding the marker times in seconds. Times for markers not present in a frame are set to zero.
`codes`	a <`points` *x* frames> 2-dimensional array of integer values holding the marker codes.

Idealised trace channels

Idealised trace channel data is represented as a 2 dimensional array of structures, with each structure holding information on one trace segment. Each marker channel is exported to a separate variable in the MATLAB workspace. The fields in the channel structure are:

`title`	a string holding the channel title.
`units`	a string holding the channel units.
`points`	an integer holding the maximum number of trace segments in a frame.
`frames`	an integer holding the number of frames exported.
`values`	a <`points` *x* `frames`> 2-dimensional array of structures holding the trace segment information. Each structure contains the following items:

`start`	a double holding the start time of a segment.
`period`	a double holding the length of a segment. This will be zero for segments not in this frame.
`amplitude`	a double holding the level of a segment.
`flags`	an integer holding the flags set for a segment.
`baseline`	a double holding the baseline level for a segment.
`level`	an integer holding the level of a segment. This is zero for a closed segment, 1 to n for open levels.

XY channels

XY view channel data is represented as two 1-dimensional arrays holding X and Y data plus ancillary information. Each marker channel is exported to a separate variable in the MATLAB workspace. The fields in the channel structure are:

`title`	a string holding the channel title.
`yunits`	a string holding the Y value units.
`xunits`	a string holding the X value units.
`points`	an integer holding the number of XY points.
`xvalues`	a <`points` *x* 1> array of doubles holding the x values of the data.
`yvalues`	a <`points` *x* 1> array of doubles holding the y values of the data.

Script export to mat-files

Export to mat-files is also available from the script language by using the `FileExportAs()` function with the file type set to 100. With this type in use, the file name selection dialog (if provided) will use a '*.mat' filename filter, an extra `exp$` argument becomes available to allow MATLAB-export specific options to be set and the time range, frames and channels to be exported are set by `ExportTimeRange()`, `ExportFrameList()` and `ExportChanList()`. The extra `exp$` argument that sets options is a string of the form:

```
name=value|name=value|…|name=value
```

where `name` specifies some export option and `value` sets it's value. You can include as many options as you want, options that you omit are set to the default value. Option names are not case-sensitive. It is not an error to use an unknown option.

Data view export options

`UseSName`	selects use of the source name in the channel variable name. Set to 1 if you want to use the source name, 0 if not. The default is 1.
`UseCName`	selects use of the channel name in the channel variable name. Set to 1 if you want to use the channel name, 0 if not. The default is 0.
`WaveOpts`	selects generation of an array of waveform error values in the workspace variable, if errors are available. Set to 1 if you want errors, 0 if not. The default is 0.
`WaveData`	selects the sort of data generated for waveform channels. Set to 1 for single-precision reals and 2 for double-precision reals. The default is 2.
`Compat`	sets the compatibility option. Set to 0 for version 7 or later, 1 for version 4 or earlier and 2 for version 6. The default is 0.

XY view export options

`UseSName`	selects use of the source name in the channel variable name. Set to 1 if you want to use the source name, 0 if not. The default is 1.
`UseCName`	selects use of the channel name in the channel variable name. Set to 1 if you want to use the channel name, 0 if not. The default is 0.
`Compat`	sets the compatibility option. Set to 0 for version 7 or later, 1 for version 4 or earlier and 2 for version 6. The default is 0.

Index