

Signal for Windows

Version 3

Copyright © Cambridge Electronic Design Limited 1997-2007

Neither the whole nor any part of the information contained in, or the product described in, this guide may be adapted or reproduced in any material form except with the prior written approval of Cambridge Electronic Design Limited.

Version 3.00	Aug 2004
Version 3.01	Dec 2004
Version 3.02	Jan 2005
Version 3.03	Apr 2005
Version 3.04	Sep 2005
Version 3.06	Jan 2006
Version 3.07	Apr 2006
Version 3.08	Nov 2006
Version 3.09	Feb 2007

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.

Trademarks and Tradenames used in this guide are acknowledged to be the Trademarks and Tradenames of their respective Companies and Corporations.

Table of Contents

Signal for Windows.....	1-1
Introduction.....	1-1
New features in version 3.....	1-1
Hardware required.....	1-2
Installation.....	1-2
Updating Signal	1-2
Removing Signal.....	1-2
File icons.....	1-3
Using this manual	1-3
Direct access to the raw data.....	1-3
Licence information	1-4
 Getting started	 2-1
Introduction.....	2-1
Basic operations	2-1
Selecting channels.....	2-1
Toolbar and Status bar	2-2
A discussion about frames	2-2
Controlling the display.....	2-4
Cursor measurements	2-7
Memory views	2-9
Saving and reloading memory views	2-12
XY views	2-12
Summary.....	2-12
 Sampling data	 3-1
Introduction.....	3-1
Types of channel	3-1
Waveform channels.....	3-1
TTL compatible signals	3-2
Marker channels.....	3-2
Marker codes.....	3-3
Sampling configuration.....	3-4
General configuration	3-4
Peri-trigger configuration.....	3-7
Ports configuration.....	3-8
Clamp configuration	3-9
Outputs configuration	3-9
Pulses or sequencer	3-13
DAC outputs	3-14
Digital outputs.....	3-14
Digital inputs.....	3-14
Automation configuration	3-15
Artefact rejection	3-16
Creating a new document.....	3-16
Online analysis.....	3-17
Sampling control panel	3-17
During sampling.....	3-18
Stopping sampling.....	3-19
Finishing sampling.....	3-19
Saving new data	3-20
Saving configurations	3-20
Sequence of operations to set and save the configuration.....	3-20

Sampling with clamp support.....	4-1
Introduction	4-1
Online clamp support features	4-1
Running a clamping experiment	4-3
Pulse outputs while sampling	5-1
Introduction	5-1
Pulses dialog	5-1
Dragging and dropping	5-3
Adding a new pulse	5-3
Moving a pulse	5-4
Removing a pulse	5-4
Finding a pulse.....	5-4
Copying pulses	5-4
Editing the pulse parameters.....	5-5
Simple square pulse	5-5
Varying amplitude pulse.....	5-5
Varying duration pulse	5-6
Square pulse train	5-6
Ramp with varying amplitudes	5-6
Sine wave.....	5-6
Arbitrary waveform	5-6
Setting a waveform	5-7
Pulses with variations	5-7
Outputs frame and Fixed interval sweeps	5-8
Controlling pulse outputs during sampling	5-9
Sequencer outputs during sampling	6-1
Overview	6-1
Sequencer control panel.....	6-1
Sequencer technical information.....	6-1
The sequence editor	6-2
Loading sequencer files for sampling	6-3
Getting started.....	6-3
Instructions	6-6
Instruction format	6-7
Expressions.....	6-7
Variables	6-8
Table of values.....	6-9
Sequencer instruction reference.....	6-10
Digital I/O.....	6-10
DAC outputs	6-12
Cosine output control instructions	6-14
General control	6-19
Variable arithmetic	6-21
Table access	6-23
Access to data capture.....	6-24
Randomisation.....	6-26
Arbitrary waveform output	6-28
Sequencer compiler error messages.....	6-28
Sampling with multiple states	7-1
What can multiple states do?	7-1
Defining multiple states.....	7-1
Dynamic outputs states	7-2
Protocols	7-4

External digital states	7-5
Static outputs states	7-6
Controlling multiple states online	7-7
File menu	8-1
New	8-1
Open	8-2
Import data	8-3
Import Op/Cl	8-3
Close	8-3
Close All	8-3
Save and Save As	8-3
Export As	8-4
Revert To Saved	8-4
Send Mail	8-5
Data update mode	8-5
Load and Save Configuration	8-5
Page Setup	8-5
Print preview	8-6
Print visible, Print and Print selection	8-6
Print screen	8-6
Exit	8-6
Edit menu	9-1
Undo	9-1
Cut	9-1
Copy	9-1
Paste	9-1
Delete	9-1
Clear	9-2
Copy As Text...(Data view)	9-2
Copy As Text (XY view)	9-3
Find, Find Again and Find Selection	9-3
Select All	9-4
Replace	9-4
File comment	9-4
Frame comment	9-4
Preferences	9-4
Show event details	9-9
View menu	10-1
Toolbar and Status bar	10-1
Previous frame and Next frame	10-1
Goto frame	10-1
Show buffer	10-1
Overdraw frame list	10-1
Add frame to list	10-1
Frame display list	10-2
Enlarge view and Reduce view	10-2
X Axis range	10-2
Y Axis Range	10-3
Standard Display	10-3
Customise display	10-3
Channel Information	10-4
File Information	10-4
Options	10-4
Draw mode	10-4

XY Draw Mode	10-6
Font.....	10-6
Use Colour and Use Black And White	10-7
Change Colours	10-7
Keyboard display control.....	10-8
Analysis menu	11-1
New Memory View	11-1
Waveform average.....	11-1
Auto average.....	11-2
Amplitude histogram	11-2
Power spectrum	11-3
Leak subtraction.....	11-4
Process.....	11-5
Process command with a new file.....	11-6
Process settings.....	11-6
New XY View	11-7
Trend plot	11-7
Fit data	11-8
Open/Closed times.....	11-13
New idealised trace (SCAN).....	11-13
New idealised trace (Threshold)	11-15
Open/Closed time histogram	11-16
Open/Closed amplitude histogram.....	11-16
Burst duration histogram	11-16
Append frame	11-17
Append frame copy.....	11-17
Delete frame	11-17
Delete channel	11-17
The frame buffer	11-17
Clear buffer.....	11-17
Copy to buffer.....	11-18
Copy from buffer	11-18
Exchange buffer.....	11-18
Add to buffer	11-18
Subtract buffer	11-18
Average into buffer.....	11-18
Multiple frames.....	11-18
Modify channels	11-19
Tag frame.....	11-19
Digital filters.....	11-20
Keyboard analysis control	11-20
Cursor menu	12-1
New cursor.....	12-1
Cursor mode	12-1
Delete.....	12-3
Fetch	12-3
Move to.....	12-3
Display all.....	12-3
Label mode	12-3
Renumber.....	12-3
New horizontal cursor.....	12-3
Delete horizontal.....	12-3
Fetch horizontal	12-3
Move to level	12-4
Display all horizontal.....	12-4
Horizontal label mode.....	12-4

Renumber horizontal.....	12-4
Display y values.....	12-4
Cursor regions.....	12-5
Sample menu	13-1
Sampling configuration.....	13-1
Sample Bar List.....	13-1
Signal conditioner.....	13-1
Show Sampling controls.....	13-2
Show Pulse controls.....	13-2
Sample now.....	13-2
Show Sequencer controls.....	13-2
Script menu	14-1
Compile Script.....	14-1
Run Script.....	14-1
Evaluate.....	14-1
Turn Recording On/Off.....	14-1
Debug bar.....	14-2
Script Bar.....	14-2
Script List.....	14-2
Window menu.....	15-1
Duplicate window.....	15-1
Hide.....	15-1
Show.....	15-1
Tile Horizontally.....	15-1
Tile Vertically.....	15-1
Cascade.....	15-1
Arrange Icons.....	15-1
Close All.....	15-2
Windows.....	15-2
Help menu	16-1
Using help.....	16-1
About Signal.....	16-1
Tip of the Day.....	16-2
View Web site.....	16-2
Other sources of help.....	16-2
Digital filtering.....	17-1
Introduction.....	17-1
Apply.....	17-1
Show details.....	17-2
Filter bank.....	17-3
Filter types.....	17-3
FIR filters.....	17-5
FIRMake() filter types.....	17-7
FIR filter examples.....	17-8
Hilbert transformer.....	17-14
Programmable Signal Conditioners	18-1
What a signal conditioner does.....	18-1
Serial ports.....	18-1
Control panel.....	18-2
Setting the channel gain and offset.....	18-3
Conditioner connections.....	18-4

Amplifier Telegraph	19-1
Standard 1401 telegraphs.....	19-1
Telegraph configuration - 1401	19-1
MultiClamp 700 telegraphs	19-2
MultiClamp 700 telegraph configuration.....	19-2
 Auxiliary states devices	 20-1
MagStim support	20-1
CED 3304 support	20-8
CED 3304 Safety notice	20-8
Introduction	20-8
CED 3304 support configuration.....	20-8
Connections and cabling.....	20-10
Notes on use.....	20-10

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 cut and paste the results to other applications. Alternatively, you can obtain printer hard copy directly from the application. When you close a data file, 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.

If you or your colleagues use the DOS SIGAVG or Patch and VClamp software, you can transfer data files to Signal without effort. All these programs use CFS data files and Signal will directly read and use them. The DOS programs can also read data files generated by Signal, but it may not be able to access all the information stored in them.

New features in version 3 Version 3 of Signal is completely compatible with earlier versions; it will read data files and sampling configurations created by Version 2.xx without problems. Scripts that ran in version 2 should work without modification. The main new features in Version 3 are:

- Curve fitting has been substantially extended to include polynomial and sine fits with enhanced fit quality information including a display of residuals.
- Curve fitting is now available in XY views.
- A text sequencer has been added, allowing more complex control of the digital and analogue outputs.
- Single channel patch clamp data analysis including idealised trace formation and dwell time and amplitude histograms.
- Logarithmic axes have been added.
- An extra sampling mode has been added to allow for external convert trigger for arbitrary or varying sample rates.
- Independent colours for channel drawing allow visual grouping of related channels.
- Cubic spline drawing joins data with smooth curves for better visual presentation and estimation of data values between sampled points.
- Channel information dialog added.
- A new Windows dialog lists all views, their types and their view handles and allows rapid view selection.
- You can now copy and paste data from one XY view to another.
- Matrix mathematics support, including matrix inversion, matrix multiply and principal component extraction.
- The script language now supports multiple display monitors.
- The script language now supports text to speech (TTS) conversion for systems with TTS support installed.

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 3 can download updated versions of Signal version 3 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 64 MB of memory running Windows 95. 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 need a CED Power1401, micro1401 Mk II, micro1401, 1401*plus* or standard 1401. Pulse output during sampling is not available with a standard 1401. USB operation requires Windows 98SE, Me, 2000 or XP.

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

From CD-ROM Just put the CD-ROM in the drive and it will start the installation. You can also run the installation manually by opening the folder `Signal` on the CD-ROM, then open the `disk1` folder and run `setup.exe`.

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

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

Place the Signal installation disk 1 in a suitable drive (assumed to be `a:` below). Click **Start**, then **Run**, then type:

```
a:setup
```

During installation (CD-ROM and floppy disk) You must select a suitable drive and folder for Signal and personalise your copy with your name and organisation. You can run both version 2 and version 3 on the same system (but not simultaneously) as long as they are in different folders. If you have version 1 on your system, install version 3 to a different folder. The installation program copies the Signal program plus help and example files. It can also install 1401 support in Windows NT 4. If you use Windows 95, 98, Me, NT 2000 or XP there are instructions for loading the 1401 drivers using the Windows Device Manager.

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 3 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 3 release free of charge from our Web site: <http://www.ced.co.uk> in the updates section. You can only update a correctly installed copy of Signal version 3. There are full instructions for downloading the update on the Web site. You can also sign up for email notification of updates.

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 select an installation directory that contains a copy of Signal for Windows version 3.

Removing Signal You can remove Signal from your system: open the system Control Panel, select **Add/Remove Programs**, select **CED Signal for Windows version 3** and click **Remove**. This removes files installed with Signal; you will not lose files 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.

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.

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 3.04 of the Signal software. Later versions of the software will have further enhancements which will be talked about 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.

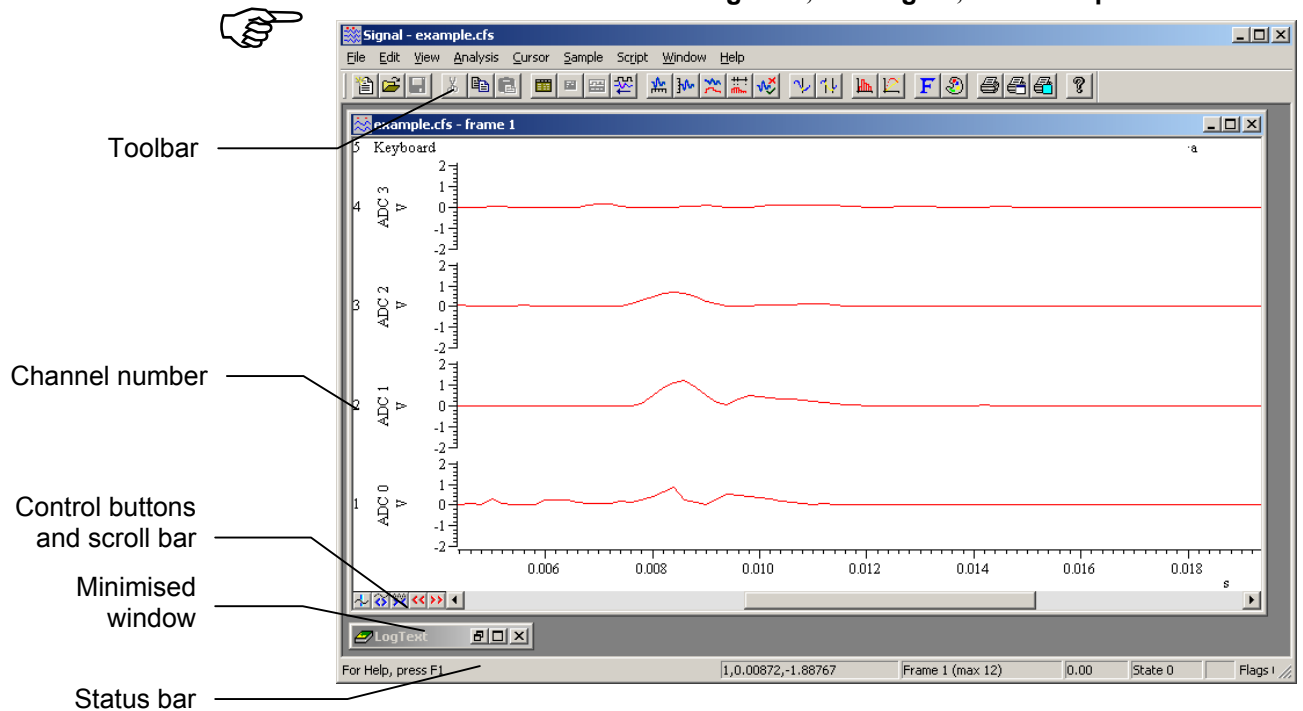
Additional software licences The original licensee of a CED software product can purchase additional licences to run multiple copies of the same software. CED supplies an additional manual set with each licence. CED does not supply additional software media. As these additional licences are at a substantially 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.

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. 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 a shortcut to commonly used menu items. Each toolbar button matches an action from the menu. The buttons appear 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** 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 view that has axes, this displays the pointer X and Y positions. The first figure is the channel number from which the Y value is taken.
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 <i>Sampling with multiple states</i> .

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. For files collected by Signal version 1.00, this will always be zero.
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 frame 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 PC. 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.

Specifying frames

You will often need to specify the frame or frames to be used for an operation. You can select the current frame in the view or you can enter a single frame number directly. To specify more than one frame, you can enter a frame list such as 1..50,60,61,70..80, or you can select options such as **All frames**, **Tagged frames**, **Untagged frames** or **Frames with state n** (with a separate field for entering the value of n), giving a wide range of possibilities. These mechanisms are also available in the Signal script language.



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

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.



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).

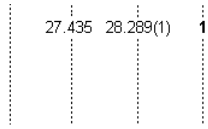


Click this button to add a cursor to the display. Up to 10 cursors can be present in a window. A cursor is a vertical dashed line used to mark positions. You can remove cursors by using the Cursor menu **Delete** command. You can add a cursor in two ways:

1. To add a new cursor in the centre of the window, click on the button.
2. Use the Cursor menu **New cursor** command, or its shortcut Ctrl+I.



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 two possible shapes to indicate the two actions you can take with a cursor:



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



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

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.

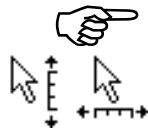


If you hold down the **Ctrl** key before you hold the mouse button down, the mouse pointer changes to the un-magnify symbol. If you drag a rectangle, the data in the view shrinks to cover an area the same size as the rectangle you have dragged, making this the inverse of the effect without the **Ctrl**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

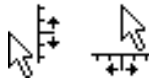


If you hold down the **Alt** key before you click and drag, Signal displays the size of the dragged rectangle next to the mouse pointer and does not zoom the display.



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 2-9), 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.

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 labeled 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.

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.

The Y Range dialog box has a title bar 'Y Range'. It contains a 'Channel' dropdown menu set to '1 ADC 0 (Waveform)'. Below it are 'Top' and 'Bottom' input fields with values '2.412791' and '-0.1451823' respectively. To the right of these fields are buttons 'Optimise', 'Show All', and 'Draw'. At the bottom, there are checkboxes for 'Large tick spacing' (unchecked), 'Tick subdivisions' (set to 4), 'Logarithmic' (unchecked), and 'Auto adjust units' (unchecked). There are also 'Cancel' and 'Close' buttons.



Open the **View** menu **Waveform Draw Mode** dialog. Experiment with different drawing modes for the channels.

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 Waveform draw mode for example dialog box has a title bar 'Waveform draw mode for example'. It contains a 'Channel' dropdown menu set to '2 ADC 1 (5 kHz)' and a 'Line' dropdown menu. To the right of these fields are buttons 'OK', 'Cancel', and 'Draw'.

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.

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!

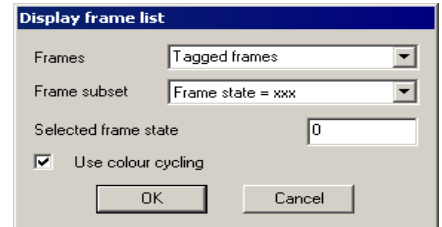
The list on the left of the dialog holds all the channels that can be displayed. You can also show or hide the axes, grid and scroll bar in the window from this dialog and control the appearance of the x and y axes. Check the boxes next to the items for display and click the **Draw** button to see the result. The **Scale only** option draws axes as scale bars.

The Display for example.cfs dialog box has a title bar 'Display for example.cfs'. It contains a list of channels on the left: '5 Keyboard - Marker', '4 ADC 3 - Waveform', '3 ADC 2 - Waveform', '2 ADC 1 - Waveform', and '1 ADC 0 - Waveform'. Above the list are checkboxes for 'Chan numbers' (checked) and 'Grid' (unchecked). To the right of the list are two columns of checkboxes: 'Y options' (Numbers, Title, Units, Small ticks, On right, Scale only) and 'X options' (Numbers, Title, Units, Small ticks, Scroll bar, Scale only). At the bottom are buttons 'All On', 'All Off', 'Draw', 'Close', and 'OK'.



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.

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 page 2-3 for details. 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.

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.

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

Below the table are checkboxes for 'X Zero' and 'Y Zero', each with four radio buttons corresponding to the four cursors.

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.

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.

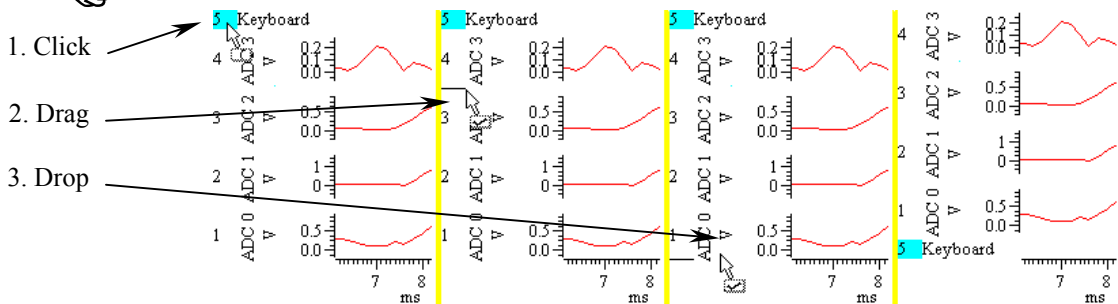
Cursor regions for example.cfs			
Cursors	1 - 2	2 - 3	3 - 4
s	0.00604196	0.0060585	0.00712329
5 Keyboard	0	165.057	0
4 ADC 3	0.0992839	0.0397135	0.0305854
3 ADC 2	0.103597	0.0412598	0.0985379
2 ADC 1	0.254639	0.0199382	0.0411648
1 ADC 0	0.140625	0.0164388	0.0157335
201 ADC 0			
<input type="checkbox"/> Zero region			
Mean			

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.



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

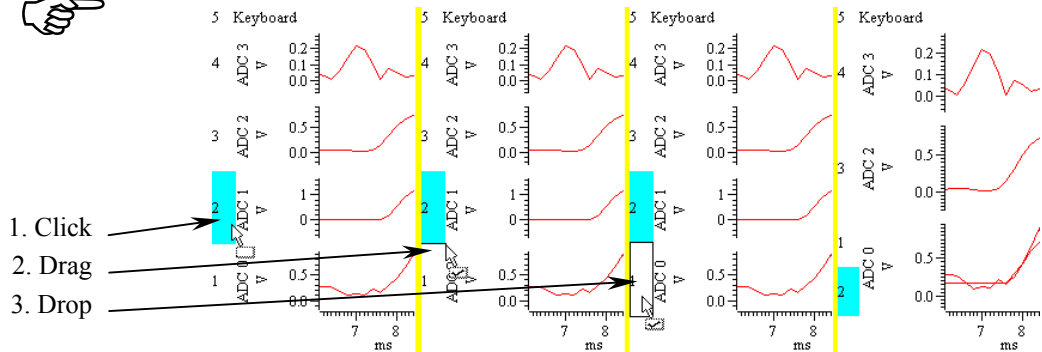


As the mouse pointer passes over each channel, a horizontal line appears above or below the channel. This horizontal line shows where the selected channel will be dropped. Drag until you have a horizontal line below channel 1 and release the mouse button. The **Keyboard** channel will now move to the 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.

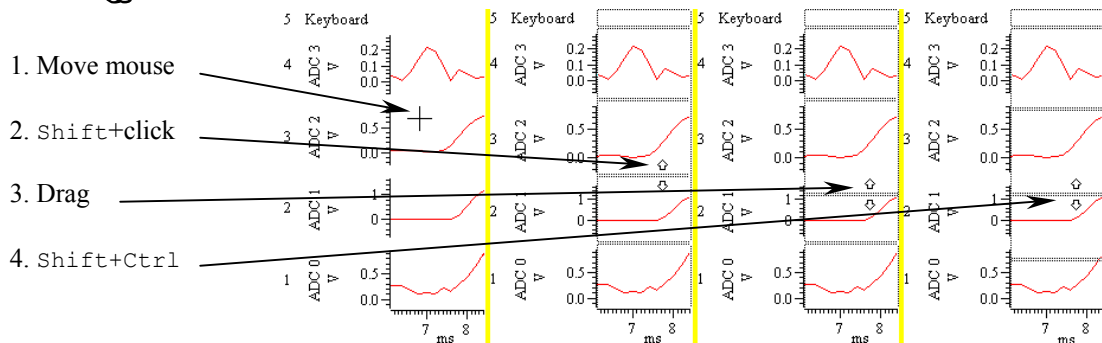


The channels now share the same space with the channel numbers stacked up next to the y axis. The visible y axis is for the top channel number in the stack. To move a stacked channel to the top, double-click the channel number. Stacked channels keep their own y axes and scaling. To remove a channel,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

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.

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

New Cancel

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.

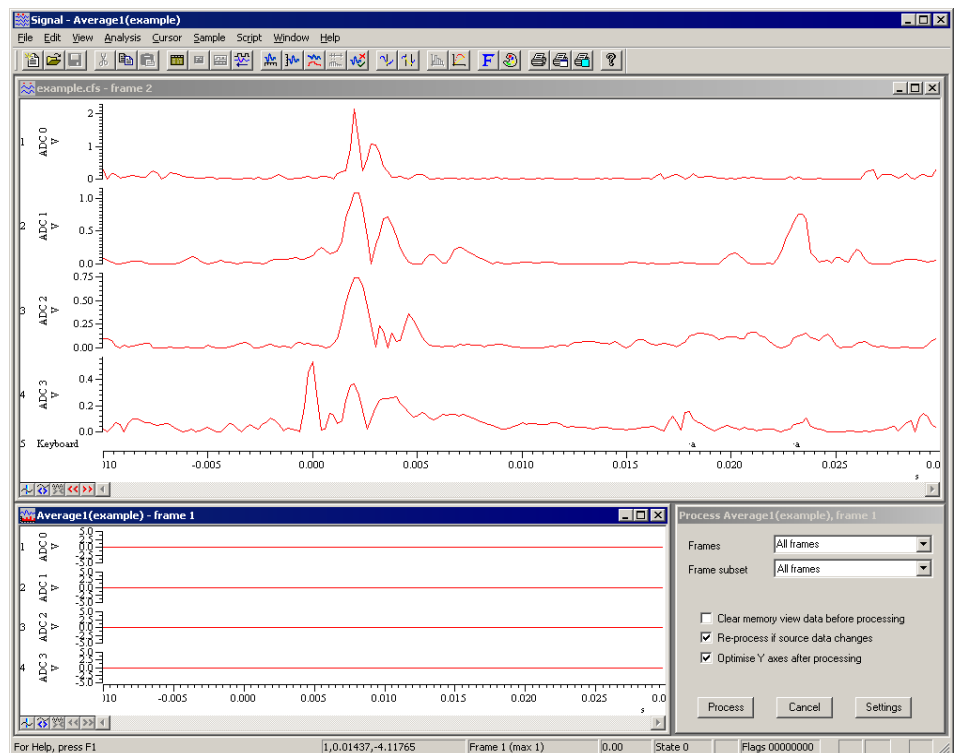
The first thing to do is to select the channels to analyse. For this example we use All waveform channels, so select this option using the pop-up menu. Set the other fields as they are in the picture above; 0.04 seconds width and a start offset of 0.



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

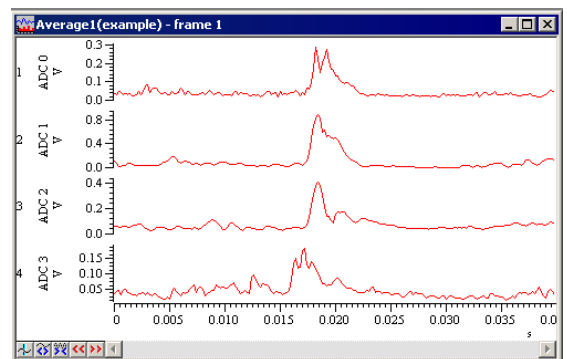
When you click the New button several things happen. Signal creates a new memory window ready to display the result of the analysis, the Settings dialog vanishes and the Process dialog appears. You must now set frames from the data document to analyse.

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.



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, utility programs 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.

Introduction

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. We have modified the **Process** dialog used during sampling to provide mechanisms for automatic processing and updates, otherwise you would have to use it constantly in order to follow the course of your experiment.

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.

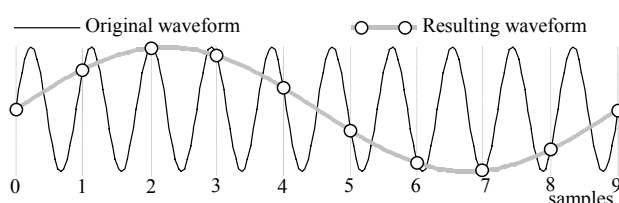
Waveform channels



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 that is the result of analysis can be stored more accurately as floating point values 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.



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

Connect your waveform signals to the 1401 ADC input ports. Ports 0-7 (0-3 for an unexpanded micro1401 or Micro1401 mk II) are the labelled BNC connectors on the front of the 1401. For the 1401 and 1401*plus* ports 8-15 are on the 15 way Cannon connector. The input voltage range is normally ± 5 volts (some 1401s are supplied set to ± 10 volts, see the *1401 Owners manual*). Connections are:

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

For the Power1401 without an ADC expansion box the extra ADC ports are on the back panel labelled Analogue Expansion. The connections are:

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

Users of the micro1401 and Micro1401 mk II will find all the ADC ports on BNC connectors. If you have a micro1401 or Micro1401 mk II ADC expansion box installed, ports 4 to 15 are BNC connectors on the expansion box. Expansion boxes are also available for the Power1401. For port numbers above 15 you will require a 32-channel expansion card (for the standard 1401 or 1401*plus*) or extra expansion boxes for the micro1401, Micro1401 mk II or Power1401. If you sample a port above the number available, the result is undefined.

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.

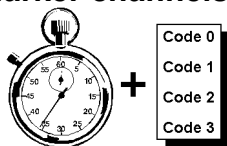
The TTL inputs and outputs on the 1401 are the digital inputs and outputs, the event inputs, the clock F external frequency inputs, the ADC external convert input, the clock output, the DAC Bri output and the micro1401, Micro1401 mk II or Power1401 trigger input. On the micro1401, Micro1401 mk II and Power1401, 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 markers and digital markers. 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. This is the number of sample intervals on each waveform channel for keyboard markers and multiples of the outputs resolution for digital markers. In addition the 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 the micro1401, Micro1401 mk II and Power1401. 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 <i>plus</i>	All others
Data Available	pin 24	pin 23

To log a digital marker, apply a low going TTL pulse at least 1 μ s wide to the **Data Available**. When the 1401 detects the falling edge of the input, it latches the input data on the Power1401, micro1401 and Micro1401 mk II. 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:

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

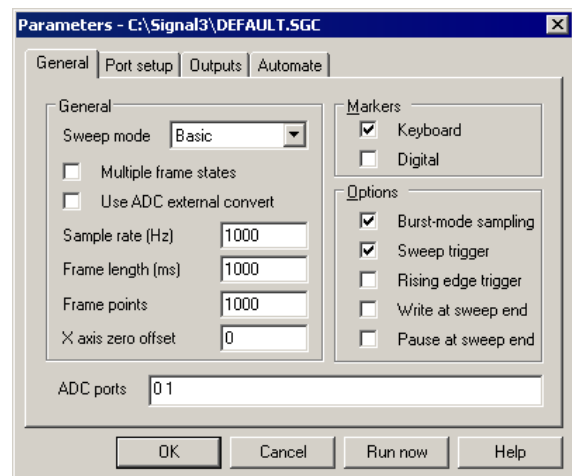
+	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
20	!	"	#	\$	%	&	'	()	*	+	,	-	.	/	
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{		}	~	

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; it contains a number of tabs for selecting separate sections of the sampling 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** and **States**, 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.



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 general 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 sweeps relate to the outputs system. It has six selections listed below:

- | | |
|-----------------------|--|
| 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. |
| Outputs frame | the pulse outputs are triggered rather than the sweep, pulses can occur before the sweep starts and after the sweep is over, and the sampling sweep is started by the outputs. |
| Fixed interval | similar to Outputs frame , but the sweeps are internally timed so that they occur at the specified interval. External sweep triggers are not used in this mode. |
| Fast triggers | like Basic mode except multiple frame states and incremental pulsing are not available. This keeps the inter-sweep interval to a minimum. |
| Fast fixed int | has the same limitations as Fast triggers but uses a fixed interval between sweeps rather than requiring an external trigger. |

More details can be found in the **Pulse outputs during sampling** chapter.

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 effects and styles of data acquisition.

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

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 sampling rate is 625 kHz. A Micro1401 mk II will sample at up to 500kHz. 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 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 available.

X axis zero offset Normally zero appears either at the start of the frame or at the trigger time for peri-triggered sampling (see below). This position can be moved by entering a non-zero value in this field. This will not alter when sampling takes place but just the way in which the time is displayed on the x-axis.

- ADC ports** This field sets the ADC ports to sample. 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 31 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 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 3-2 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 $1/(\text{Sample rate} * \text{number of ADC ports})$. In burst mode sampling all the ADC ports are sampled in a burst, as close together as possible, the interval between bursts is $1/\text{Sample rate}$. 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, micro1401 or Micro1401 mk II 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 for the Power1401.
- In Peri-triggered sweeps burst mode is always used for efficiency reasons so the state of this checkbox is ignored.
- 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, Micro1401 mk II or Power1401 Trigger input. For **Peri-triggered** sweeps the trigger can be any of a number of signals (see page 3-7).

There is a small (~10 microseconds) delay between the time of the sweep trigger and start of sampling. This delay 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 on the rising edge.

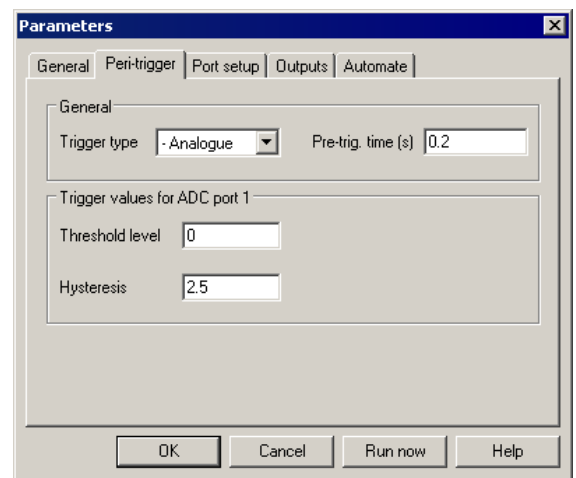
Write sweep to disk This checkbox sets the initial state of the **Write to disk at sweep end** checkbox in the sampling control panel. When this is set, sampled sweeps are automatically written to disk when the sweep finishes.

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.

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

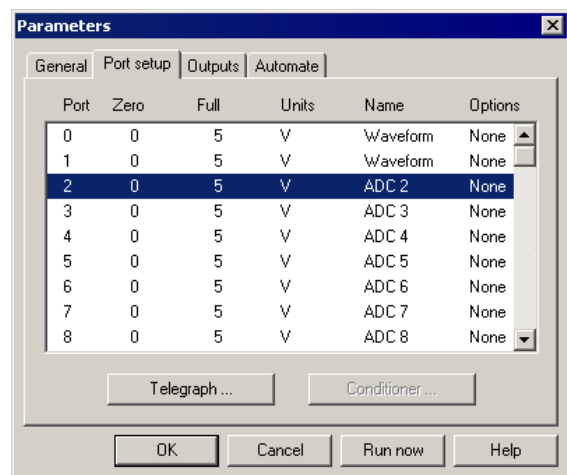
Event Trigger on a TTL pulse. There are no parameters; the trigger occurs when a TTL pulse is detected on the standard 1401 or 1401*plus* Event 0 input, or the Trigger input on a micro1401 or Power1401.

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 $-(1,000,000 * \text{sample interval})$ to the length of the frame $-(2 * \text{sample interval})$. 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.

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).

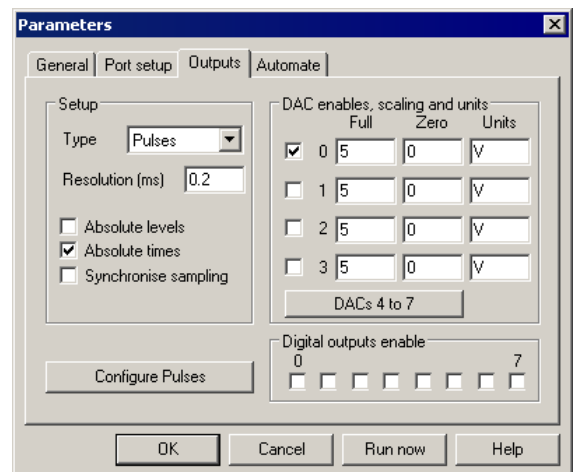
The **Telegraph...** button on the bottom left opens a dialog that allows you to configure amplifier telegraph support. Amplifier telegraphs are signals, usually analogue outputs, from an amplifier that signal current amplifier settings such as gain and offsets. By collecting and interpreting the amplifier telegraphs, Signal can automatically adjust for changes in the amplifier gain 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 chapter *Amplifier Telegraphs*. Only a few types of amplifier support telegraph signals; if you do not have such an amplifier you can ignore this chapter.

Clamp configuration

This option only appears when clamping support is enabled in the preferences and is described in the *Sampling with clamp support* chapter.

Outputs configuration

Use this section of the sampling configuration dialog to set the outputs required during sampling, to set 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 outputs. It contains a selector for the type of outputs required plus items specific to the type of outputs. The right-hand areas enable and set up the output ports.



Outputs type

This control selects the type of outputs to use, 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 in a separate chapter of this manual: *Pulse outputs during sampling*. 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 the Power1401 you can go down as far as 10 microseconds.

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.

Synchronise sampling This selects between mechanisms for synchronising external triggers, pulse outputs and the actual sampling sweep. Normally, the sweep trigger starts the sampling sweep and the pulse output mechanism, which is free running throughout sampling, is synchronised to the sweep.

This arrangement gives the maximum accuracy of the sampling sweep start time relative to the sweep trigger (about 2 to 5 microseconds), but will give a 10 to 20 microsecond delay in pulse outputs relative to the sampling. If you select synchronised sampling, the sampling sweep start is delayed relative to the sweep trigger by 10 to 20 microseconds, but the pulse outputs are more precisely synchronised with the sweep, to an accuracy of 2 to 5 microseconds.

If **Outputs frame**, **Fixed interval** or **Fast fixed int** sweep mode is used, this item is ignored, as the sweep is triggered directly by the pulses system, giving the same effect as if synchronised sampling was selected.

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 mode of operation is only available when synchronised sampling is disabled. On the micro1401, Micro1401 mk II and Power1401 in this circumstance, the trigger input is automatically routed to both E0 and E4 internally, thus guaranteeing a precise sampling start relative to the trigger.

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

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.

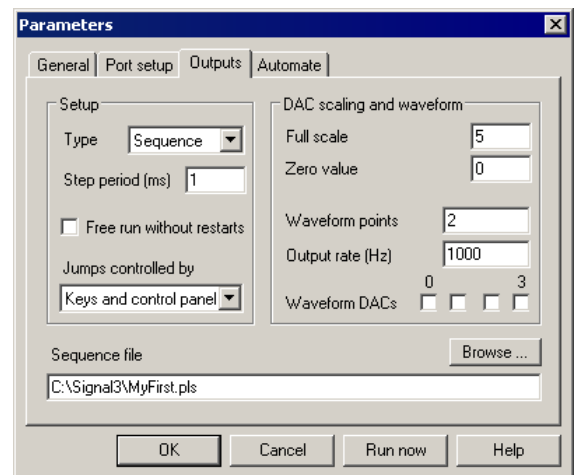
Zero	The value (in calibrated units) corresponding to a zero value output from the DAC. This value, along with Full, is used to convert the floating-point values used by Signal into the integer quantities actually used by the DAC hardware. This conversion process occurs when generating pulse outputs, 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 <i>plus</i> 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 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 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 includes 64 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 given in a separate chapter of this manual *Sequencer outputs during sampling*. 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 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.
- 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?

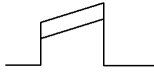
Pulses output is easier to get started with, and supports multiple states directly, it will be suitable for most users of Signal. You should consider using the sequencer if you are unable to achieve the effect you require with Pulses output. The table below summarises the main differences between these 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 the 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 generation of outputs is carried out identically. When you use Pulses outputs, the pulses information is used to build sequencer data that is loaded into the 1401 and executed in exactly the same manner as Sequencer outputs. The timing requirements and limits for these two forms of output are therefore 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.

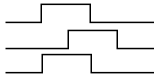
	Pulses	Sequence	Free-run sequence
Basic	Sweep and outputs triggered or free-run together and are of same length.	Sweep and sequencer triggered or free run together.	Sequencer starts immediately, runs throughout. Can trigger sweeps or react to sweep progress.
Peri-trigger	Pulse output triggered when sweep trigger recognised, same length as remaining sweep.	Sequencer triggered when sweep trigger recognised.	Sequencer starts immediately, runs throughout. Can trigger sweeps via digital and DAC outputs, can monitor sampling and react.
Outputs frame	Pulse output triggered or free-runs. Pulse output starts sampling sweep at the required point.	Sequencer triggered or free-runs, responsible for triggering the sampling sweep at the required point.	Sequencer starts immediately, responsible for triggering sweeps as and when wanted. No external trigger or free-running sampling.
Fixed interval	Outputs triggered by internal timer, starts sweep at specified time within outputs.	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 output set only, same length as sampling sweep. Sweep and outputs triggered or free run together.	Sweep and sequencer triggered together.	Sequencer starts immediately, runs throughout. Can trigger sweeps itself or react to sweep progress.
Fast fixed interval	Single output set only, same length as sampling sweep. Sweep and outputs triggered together by internal timer.	N/A, simulate using Basic mode with free running script.	N/A, simulate using Basic mode with free running script.

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.

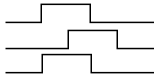
Digital outputs

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. 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 <code>DIGOUT</code> pins	1	14	2	15	3	16	4	17	13
<code>DIGLOW</code> pins	5	18	6	19	7	20	8	21	13

Digital inputs

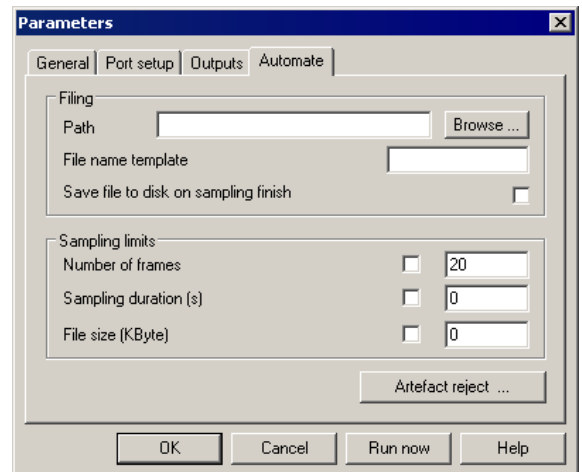
Digital input bits 8 to 15 are used in External digital states mode and in Peri-trigger sampling. Digital input bits 0 to 7 are read by the output sequencer and will be used in the future for digital markers. Signal refers to both sets of inputs as bits 0 to 7. The 1401 digital inputs are on a 25-way '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 Signal 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 Signal 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.



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. 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".

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.

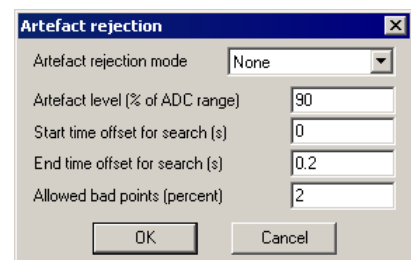
In addition to these user-defined limits, Signal has a built-in sampling limit based on the available free disk space. If the available free disk space drops below 0.5 Mbytes, sampling stops automatically. This limit cannot be disabled as it is important not to allow hard disks to get completely full, 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.



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 onwards 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 starts or when sampling of the next frame is triggered. 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**, **Fixed interval** or **Fast fixed int** 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.

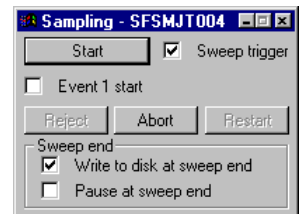
The small floating window is the *Sampling control panel*. It contains buttons and other controls to interact with the sampling. Sampling will not start until you click **Start** in this control panel (the **Sample** menu duplicates the panel controls). If the **Event 1 start** box is checked, sampling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

Online analysis

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.

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 minimised using the **-** button.

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 μ s 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.

-----Trigger above 0-----
-----Arm below -1-----

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

You can tag data frames as they are sampled by pressing **Ctrl+T**. Most standard display manipulation mechanisms work in exactly the same manner online, but frame overflow 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 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 sampling control panel).

- 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`. 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.

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** command to add memory 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. Once 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 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.

Introduction

Signal incorporates specialised features directly supporting whole-cell, and single-channel clamping experiments. In order to avoid confusing users who are not using this type of experiment, these features can be hidden or shown by using a checkbox in the Signal preferences dialog, the initial state of this option is set up by Signal if it detects that it has not already been set. The features controlled by this option are the online clamp support as documented below, leak subtraction analysis and the generation and analysis of idealised single-channel traces.

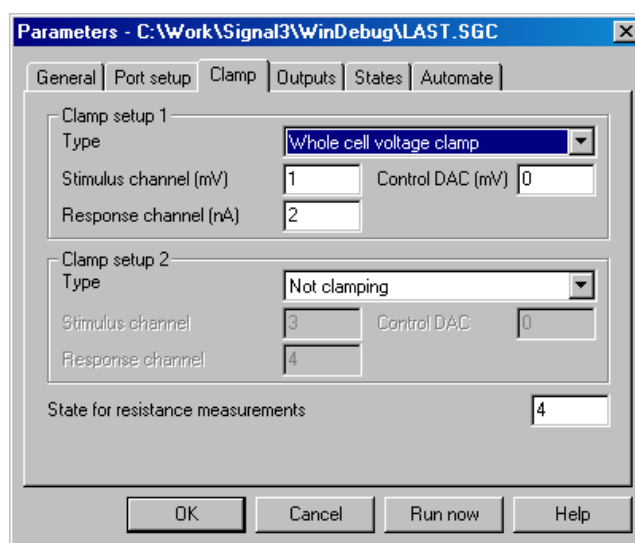
Unlike many applications used for clamping experiments, Signal is a very general-purpose program that can be used for many other types of experiment. This can make it harder for new users setting up clamping experiments to find the aspects of Signal that they need. In addition to this chapter (and general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

Online clamp support features

The online clamp support within Signal consists of a separate page within the sampling configuration where the clamp setup can be configured, 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. Because Signal is a very general-purpose program that can be configured and used in many ways, there are a number of limitations to the types of experiments that can be used with the clamping support. A significant amount of the clamping support consists of checking for incompatibilities between the clamp setup and other aspects of Signal sampling rather than simply configuring the clamping.

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. Clamp setups define the data file channels that hold the current and voltage data (so that analysis can find the correct data) and checks of the units for these channels (so that current and voltage values can be scaled correctly to amps and volts). In addition the setup defines the DAC used to control the membrane current or voltage (so that the holding potential and pulses can be manipulated).



Each clamp setup specifies the experiment type, the data file channels holding the stimulus and response signals (in voltage-clamp the stimulus is the voltage and the response is current, in current clamp experiments this is reversed) and the DAC used to control the stimulus.

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 is disabled so you can carry out non-clamping experiments.

The `Stimulus channel` and `Response channel` items set the data file channels that will hold relevant 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. The channel 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, 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 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.

The `Control DAC` item sets the DAC in the pulse outputs that is used to control the stimulus. The DAC specified must be from 0 to 7, be in use and the units for the DAC specified 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 in the picture above, whenever state 4 is used during sampling the membrane resistance will be recalculated and displayed. Signal states are quite a complex topic; for more information on them, see the chapter 'Sampling with multiple states'.

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.

Other sampling configuration considerations

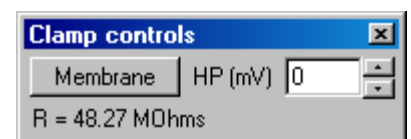
As mentioned above, because Signal is a very general-purpose program it is possible to produce a sampling configuration that does not allow the clamping features to operate correctly, though we have tried to make the software as flexible as possible. The aspects of the sampling configuration that you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

- 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 very limiting and will not work well.
- 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.
- All 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 will be incorrect.
- If multiple states are used (which is almost always necessary), these should use dynamic outputs variation.
- As described above, the state specified for resistance measurements must exist. State zero always exists, even if multiple states are not used.

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

Running a clamping experiment

When a sampling configuration with at least one enabled clamping setup is used, the clamping toolbar is created in addition to the standard Signal sampling toolbars. The clamping toolbar contains separate controls and information for each clamping setup that is enabled and can either be floating or docked to any edge of Signal. 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 current, for current clamp) controls are used to adjust the baseline level of the stimulus DAC output. 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 set up in 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. The spinner to the right of the edit field changes the value in steps of 5 millivolts for voltage clamp, and 1 nanoamp for current clamp, changes made with the spinner take effect immediately.

At the bottom is a text field where the total (electrode plus membrane) resistance is displayed. This field is automatically updated whenever the specified sampling state is used, you can select the state manually or it can be used as part of normal state cycling or protocol execution. Resistance measurements are carried out regardless of whether sampled data is being written to disk or discarded.

The Membrane button is used to provide the membrane analysis dialog:

Membrane analysis The membrane analysis 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.

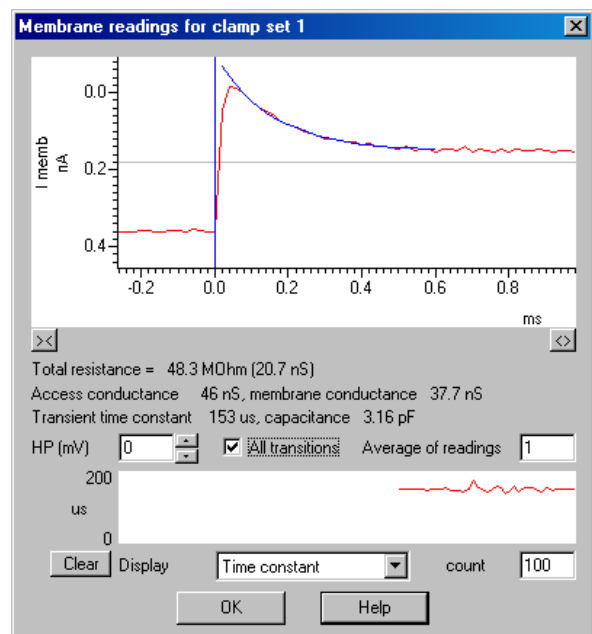
The membrane analysis measures the total resistance (and total conductance), access and membrane conductance, the time constant for the decay of the capacitance transient and the membrane capacitance. See below for details of the analysis.

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 start 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.

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 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.

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 pulse there are two transitions; one at the start of the pulse and one at the end.

To measure total resistance, the static level attained after the transition is measured over the final quarter of the stimulus pulse (for transitions at the end of a pulse the previous pulse width is assumed). The delay of $\frac{3}{4}$ of the pulse width is intended to allow the capacitive transient to settle, this settling time can be adjusted by altering the stimulus pulse width. The baseline (pre-transition) level is measured over the same period just before the transition. Measurements are taken from both the stimulus and response channels, allowing an easy calculation of resistance using Ohms law.

Once the static levels before and after the transition have been measured, voltage clamp data may be analysed further. Firstly the response data is searched to find key positions; the position and value of the peak overshoot and the points at which the overshoot drops to 90% and 5% of the peak value.

The decay phase of the overshoot is analysed by fitting a straight line to the log current values against time with all values between the 90% and 5% points being fitted. This gives us the time constant for the decay. The area underneath the overshoot is also measured to give the total amount of charge.

Having 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 $R_s = \text{time constant} / \text{capacitance}$, which allows us to separate the membrane and access components of the total resistance.

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

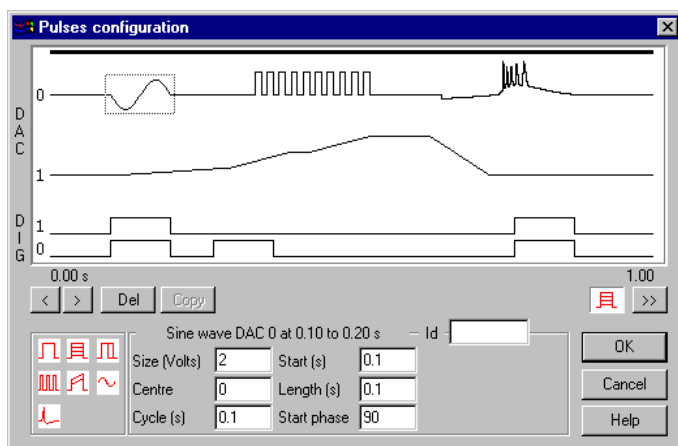
When all transitions are being analysed, the results of analysis of each transition are averaged to give a single set of measurements for the sweep.

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 output frame may be the same length as the sampling sweep, longer or shorter. In all circumstances, the pulse outputs are fixed in time relative to the sampling sweep. In **Basic** mode, the pulse output frame starts at the same time as the sweep (triggered or un-triggered) and is 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 If you press the **Configure Pulses** button in the outputs page in the sampling configuration, Signal displays the Pulses configuration dialog to allow you to view and edit the pulses.

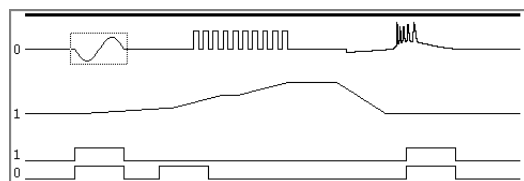
The dialog can also be used to control and adjust the pulses while Signal is sampling, in which case it is accessed using the sampling menu, the toolbar or by using the right mouse button popup menu.



The pulse configuration dialog can be used to define square pulses, pulse trains, ramps, sine waves and arbitrary waveforms that will be output during the 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 used to set and control the length of the pulse outputs frame, the start of the sampling sweep within the outputs frame and the fixed repeat interval.

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. The output traces are labelled with the DAC number or digital output bit number.

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 and displays providing information.



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:



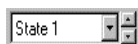
These buttons are used to toggle the selected pulse through the pulses on the current output trace in forwards or reverse time order.



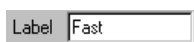
This button is used to delete the currently selected pulse.



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 that the dialog shows; 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 ..” as shown in the state selector (above) 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 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 segment at 0.1 to 0.2				Id	
Size (Volts)	2	Start (s)	0.1		
Centre (Volts)	0	Length (s)	0.1		
Cycle (s)	0.1	Start phase	90		


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 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 cannot drag the pulse to a different output). 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.

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 this is a drag 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 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 and the trigger time within the outputs are not copied, otherwise the target states are set to the length 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 the 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

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 this to 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.

Simple square pulse

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 (s)** parameter sets the start time for the pulse. If absolute times are not in use, this parameter changes to **Delay (s)** and sets the delay from the start of the previous pulse to the start of this pulse. The **Length (s)** parameter sets the length of the pulse. Both of these fields will use the current time units.

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

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. This type of pulse is not available for digital outputs. The behaviour of the built-in variation is described under *Pulses with variations*, below.

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

The **Size**, **Start** and **Length** parameters are exactly 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 by the same amount. Decreases in the pulse duration move the following pulses earlier in time. If the checkbox is clear, changes in the pulse duration do not affect the time of following pulses.

Square pulse train 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 exactly the same as for the simple pulse. The extra parameters are **Pulses**, which sets the number of pulses in the train, and **Gap(s)** which sets the interval between the end of one pulse and the start of the next.

Ramp with varying amplitudes 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 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 parameters are **Centre**, which sets the level about which the cosine oscillates, **Cycle (s)**, which sets the duration of one complete cycle and **Start phase**, which sets the initial phase in degrees. 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 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:

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

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 $\text{Initial} + (\text{Steps} * \text{Step change})$. 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 basic pulse amplitude to 30 and the step change to -10.

Outputs frame and Fixed interval sweeps

With Basic, Peri-triggered, Fast triggers and Fast fixed int sweep modes, the length of the pulse output frame is set by the data frame. In Basic, Fast triggers and Fast fixed int modes, the pulse frame length is the same as the data frame length, in Peri-triggered mode the pulse frame length is the data frame length less any pre-trigger points. With Outputs frame sweep mode in use, the pulse frame length can be set independently of the data frame and the data acquisition sweep starts at a fixed time within the pulse output frame.

These times are set by extra parameters shown with the Initial level data for all outputs. The **Frame (s)** parameter sets the length of the pulse frame, this value can not be less than the sampling sweep length. The **Trigger (s)** parameter sets the time, relative to the start of the pulse outputs frame, of the start of the sampling sweep.

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

Overview

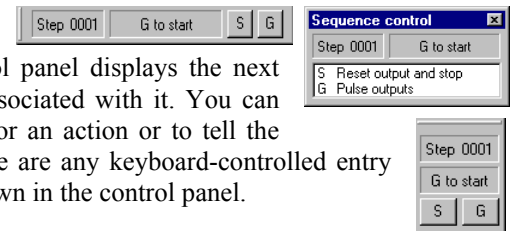
An output sequence is a list of up to 2047 instructions that are executed by the 1401 at a constant, user-defined rate to produce the outputs required. Sequences are defined by a sequence file, which is a text file with a .pls file extension. The Signal sequencer has the following features:

- 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 64 variables (V1 to V64) that can be read and set by on-line scripts.
- 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 two are in units of milliseconds and remainder are in microseconds.

	Power	Micro mk II	micro1401	1401plus
Minimum tick (ms)	.010	.025	.10	3.0
Time used per tick (us)	<1	~1	<8	<10
Cosine penalty/tick (us)	0.55	~1	~5	~10
Ramp penalty/tick (us)	0.5	0.7	~3	~10

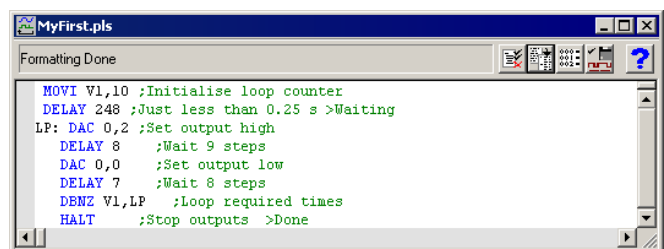
The Minimum tick is the shortest interval we recommend that you set. The Time used per tick is how long it takes to process a typical instruction. The Cosine penalty/tick is the extra time taken per cosine output. The Ramp penalty/tick is the extra time taken per ramped DAC. Time used by the sequencer 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 Power1401 and Micro1401 mk II), 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.

The sequence editor

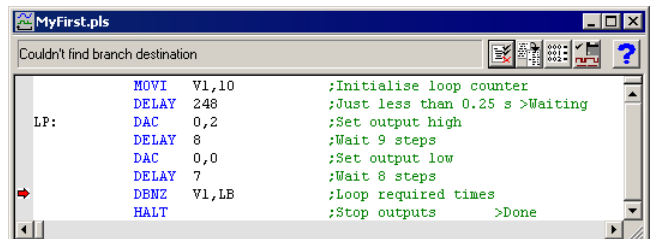
The output sequence is stored as a text file with the extension .PLS. You can open existing Signal output sequence files or create new ones with File menu New. There are five buttons at the top of the window:



Compile

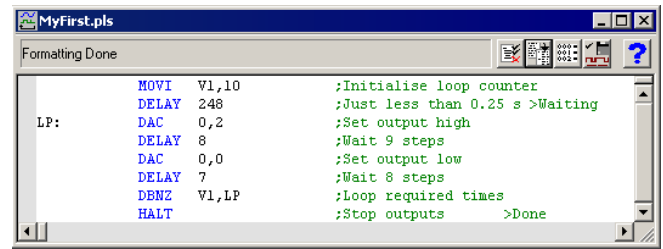


This checks the sequence to make sure that it is correct with no labels missing or duplicated and no duplicated key codes. The picture shows a sequence with a simple error (the LB in the seventh line should be LP). The line in error is marked and an explanatory message is shown at the top of the window.

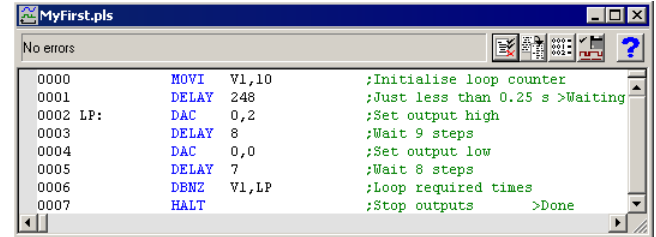


Format

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.

**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.

**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 a list of the sequencer topics for which help is available. You can copy and paste text from the help window into your sequence.

Loading 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 be set to 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.

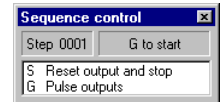
```

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

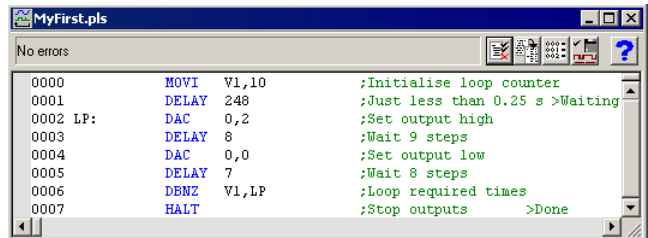
```

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.

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.



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). 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, `LP:`, so we can branch back here from elsewhere in the sequence.

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, so we get ten pulses.

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 output loops 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 in Signal version 3.
<i>Digital (TTL compatible) input and output (see page 6-10)</i>	DIGOUT	Write to digital output bits 15-8
	DIGLOW	Write to digital output bits 7-0 (does nothing in 1401 <i>plus</i>)
	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
<i>DAC (waveform/voltage) outputs (see page 6-12)</i>	DAC	Set DAC value (for DACs 0-7)
	ADDAC	Increment DAC by a value
	RAMP	Automatic DAC ramping to a target value
<i>Sinusoidal waveform output (see page 6-14)</i>	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
<i>General control (see page 6-19)</i>	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 OPERATION)
<i>Variable arithmetic (see page 6-21)</i>	ADD	Add one variable to another
	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	Move minus the value of a variable to another
	SUB	Subtract one variable from another
<i>Table support (see page 6-23)</i>	TABLD, TABST	Load a register from the table and store a register to the table
	TABINC	Increment a register and branch while within the table
<i>Access to data capture (see page 6-24)</i>	CHAN	Get the latest waveform value from a channel
	POINTS	Get the number of points sampled into the current sweep
	REPORT	Forces a digital marker to be sampled
	MARK	Records a specified digital marker
	STATE	Get the current sweep state code
	SETS	Set the state code for the current sweep
	SWEEP	Load a variable with the sweep start time
	WSWP	Wait until a given time within the sweep
	TRIG	Trigger the start of a sampling sweep
	TICKS	Load a variable with the time in Signal time units
<i>Randomisation (see page 6-26)</i>	BRAND	Random branch with a probability
	MOVRND	Load a variable with a random number
<i>Arbitrary waveform output (see page 6-28)</i>	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*) The number of sequencer steps in *expr* seconds, milliseconds and
ms(*expr*) microseconds. For example, with a step size of 200 milliseconds,
us(*expr*) *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 the current rate down by
0.1 degrees per step use `RINC Hz(-0.1)`. Use in `RATE`, `RATEW`, `RINC`
and `RINCW` instructions.

VHz(*expr*) The same as *Hz*(), but the result is scaled into the 32-bit integer units
used when a variable sets the rate. `MOVI V1,VHz(2)` followed by
`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. Use with variables for DAC and ADDAC.

`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.

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

Variables

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

Some variables have specific uses: Variables `V57` through `V64` hold the last value written by the sequencer to DACs 0 through 7 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 the variable values with 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.

Table of values The Signal sequencer supports a table of 32-bit values when used with the Micro1401 and Power1401. The 1401*plus* does not support this feature.

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 64 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 64 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

TLOOP:  MOVI   V1,0              ; use V1 as table index, set 0
        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.

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],OptLB
```

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

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

Vn With a variable the new output is: (old output `BAND Vn(7-0)`) `BXOR Vn(15-8)`. The variable equivalent of `[...001i]` is `241+256*3`, and of `[.....i]` is `255+256*1`. 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.

```
MOV      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 VDACC0-VDACC7). 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 **VDACC32()** 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

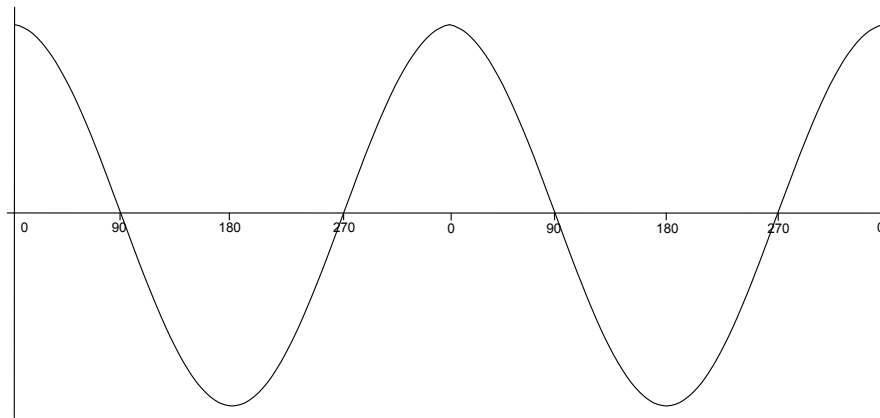
where A is an amplitude scaling factor in the range 0 to 1
 Theta an angle in the range 0° to 360° that changes each step (set by ANGLE)
 Phi is a fixed phase angle in the range -360° to 360° (set by PHASE)
 offset A voltage offset defined by the OFFSET command

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` (Decrement and Branch if Not Zero) 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:

```
WT:      MOVI     V2,1000    ;set times to loop
          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.

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

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

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

NOP The `NOP` instruction (NO OPERATION) 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

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

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

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

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

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

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

The following examples assume that v3 holds 1000:

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

ADDI This instruction adds a 32-bit integer constant to a variable. 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 μ s in the Power1401, 3 in a micro1401 mk II, 10 in a micro1401 mk I 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)
```

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.

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, and may cause sequencer 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, and may cause sequencer 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 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.

Introduction

In the *Getting started* chapter you will have encountered the frame state; a value attached to each frame in a Signal data file that can indicate a condition or classification of the frame. You will also have encountered the **Enable multiple states** checkbox in the **General** section of the sampling configuration dialog. This chapter describes the uses and control of multiple states in data acquisition.

What can multiple states do?

Simply put, Signal multiple states sampling is a way of generating data files where each frame's state value is set according to external conditions. The external condition can either be signalled by external equipment or it can be controlled by Signal generating different outputs for each state in use. There are three different forms of multiple states.

Imagine an experiment involving measuring responses from a test subject to two forms of stimulation. For example, you are recording the EEG evoked response to a tone in either the left ear (stimulus A) or the right ear (stimulus B). You have external equipment to generate the different stimuli and are using Signal to record the result. You want to generate two averaged responses, one for stimulus A and the other for stimulus B.

One way to do this would be to look at signals generated by the external equipment to indicate whether stimulus A or B was used. You could then generate two averages, one for each stimulus. This is what one form of Signal multiple states can do by reading the 1401 digital inputs and using the value read to work out a state code for each frame. This mode of operation is called *External digital states*.

perhaps you go on to make the experiment more complicated. You want four separate stimuli now (maybe two different tones for each ear) and rather than using a fixed stimulus order you want to randomise them. Rather than design hardware complex enough to do this, you can use Signal to generate outputs which control the stimulus, record the outputs used with each frame as the frame state and randomise the state order. The analysis is essentially unchanged apart from now producing four averages, but Signal is now in control of states sequencing and much more complex patterns of stimuli are possible. This mode of operation is called *Static outputs states*.

Finally, rather than just using Signal to select tones from the external equipment, you now want to generate the tones directly from within Signal using the sine wave outputs available from the pulse outputs system. Signal now has multiple sets of pulse outputs, one per state, and as it switches state it switches between the sets of pulses to generate different outputs. This mode of operation is called *Dynamic outputs states*.

These three scenarios briefly describe the three ways that Signal multiple states can be used and touch on some of the ways that states can be sequenced during the experiment. There are many other things that can be done using multiple states, but they all share a common theme; different conditions in the experiment that need to be recorded so that the analysis can distinguish data recorded under the various conditions.

Enabling multiple states

The **General** page in the sampling configuration dialog contains a checkbox labelled **Multiple frame states** which is used to enable 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 this checked, sampling will use multiple states and set the data frames to the appropriate state value.

When multiple frame states are enabled, the sampling configuration dialog gains another page labelled **States**. This page holds controls used to set up multiple states.

Defining multiple states

The **State variation** selector at the top left of the **States** page select the type of multiple states to use from **External digital**, **Static outputs** and **Dynamic outputs**. The controls shown in the dialog change dramatically as each type of multiple states is selected, so the three forms are described separately below.

What type of states do I need?

For almost all purposes **Dynamic outputs** states are the most useful as this allows you to set different output pulses for each state. **External digital** states can be used only with external hardware that generates digital codes to indicate what is happening, while **Static outputs** states are limited to unchanging digital and DAC outputs, so both of these are more limited. The main part of this chapter will cover **Dynamic outputs** states with the other forms of multiple states described afterwards.

Dynamic outputs states

Dynamic outputs states are the most useful and general form of multiple states; each state generates a separate set of output pulses. The actual digital and DAC outputs for each state (along with some other information including the state label) are set by using the **Configure 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 **Signal** will switch from state to state during sampling.

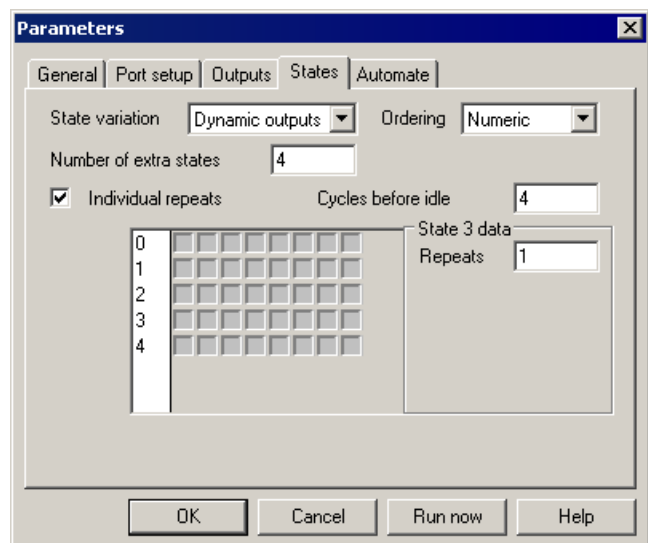
The **Number of extra states** item controls the number of extra states, from 1 to 256. Note that this item sets the number of extra states in addition to state zero. Thus in the example shown below there are 5 states possible with codes running from 0 to 4. In many circumstances **Signal** will use only states 1 to 4, state zero is treated as an background or idle state. This also controls the states available in the pulses configuration dialog; you should set the extra states you want here before you set up the pulse outputs themselves.

State sequencing

Because in **Dynamic outputs** mode **Signal** is controlling the states (as it is in **Static outputs** mode), we need ways of defining which state is used when. The simplest way to do this is to control the state manually (which can be done using a control bar), but often we want some form of automatic states sequencing.

Signal provides essentially two separate forms of states sequencing – numeric or protocol. In the numeric modes (there are three of these) simple numeric or randomised sequencing is provided while, in protocol mode a

precisely defined complex sequence of states can be generated.

**Numeric (non-protocol) sequencing modes**

In non-protocol states sequencing modes, the user is limited to specifying how many of each state are to be used and how they are to be randomised (if at all).

The **Individual repeats** checkbox selects if each state should be repeated a different number of times or if all states are repeated the same number of times. If it is clear, the **Repeats** item is used to set how many times each state is repeated. If the checkbox is set, separate controls are used to set the repeats wanted for each state (see below).

The **Cycles before idle** item sets how many times the full set of states (states * repeats) is repeated before states sequencing is automatically stopped and **Signal** switches automatically to state 0. Set this item to the number of sequencing cycles you want or to 0 if you want states sequencing to repeat forever until stopped.

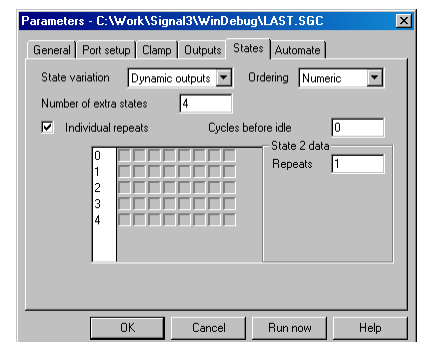
The **Ordering** selector to the top right of the states page selects the type of sequencing to be used, this can be set to:

- | | |
|--------------------|---|
| Numeric | In this mode the extra states are used in numerical order with each state being used a specified number of times (as set by the overall or individual repeat counts). First state 1 is used n times, then state 2 and so forth. Once the last state has been done, one sequencing cycle has been completed and the sequencing either stops or restarts with state 1 as set by the Cycles before idle control. |
| Random | In this mode, one cycle of the sequencing again uses each state the number of times specified by the repeats, but the order of the states is randomised. So, with two states and two repeats, each cycle lasts for four frames, two of which are certain to be state 1 and two state 2, but in a random order. When the sequencing starts another cycle, the states order is re-randomised. |
| Semi-random | This is a slight modification of Random mode, where the repeats are not all randomised together across a cycle. If there are n states then the first n frames will contain one of each state (in a random order), as will the next n and so forth. So, with 2 extra states and two repeats, the first two frames will contain one each of states 1 and 2, in random order, and the next two frames will again contain one each of state 1 and 2, again randomised. So really this achieves the same thing as setting the number of repeats to 1, but a sequencing cycle still consists of each state being used the number of times specified by the repeats. |
| Protocol | In this mode, a separate protocol dialog is used to define arbitrary sequences of states repeated an arbitrary number of times. This, of course, is not a numeric sequencing mode and is documented separately, see the section on Protocols below. When Protocol mode is selected, all controls for state repeats and cycles are hidden and replaced by a Protocols... button which provides the protocol dialog. |

When sampling using dynamic outputs Signal provides controls for the state and states sequencing, see the section *Controlling multiple states online* below.

Individual state repeat counts

When the **Individual repeats** checkbox is set, the **Repeats** item that sets the repeat count for all states is hidden. Instead the dialog shows a state selector and repeat so you can set repeats for each state. The state selector is derived from the digital output control used in other modes, to set the repeat count for a state, click on the desired state and edit the **Repeats** value shown in the state data box.



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 the specified number of times within each sequencing cycle, whereas with Randomised ordering each state is repeated the set number of times, but the order of states within each sequencing cycle is randomised. Semi-random ordering does not work well with individual repeats.

State sequencing by protocol

A protocol consists of a list of up to 10 steps, each step setting the state that will be used, a repeat count and a step to use next. A protocol can use up to ten steps, protocols can loop and be chained together to produce longer sequences of states. Up to 50 separate protocols can be defined within a single sampling configuration, giving great flexibility.

Protocols are defined using the protocols dialog which is obtained by pressing the **Protocol...** button on the states page. The dialog contains 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 protocols 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.

The checkboxes at the top of the protocol details set general options and what happens at the start of protocol execution. The **Create toolbar button for protocol** item enables a separate button for this protocol in the states control bar – see the section *Controlling multiple states online*, below. **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 protocol does not advance. **Turn on writing at protocol start** causes writing of sampled sweeps to disk (normally controlled by the **Write to disk at sweep end** checkbox in the sampling control panel) to be automatically turned on when the protocol starts. **Reset pulse steps at protocol start** causes all varying pulses defined in the pulses dialog to be reset to their initial state when the protocol starts.

	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

The checkboxes at the top of the protocol details set general options and what happens at the start of protocol execution. The **Create toolbar button for protocol** item enables a separate button for this protocol in the states control bar – see the section *Controlling multiple states online*, below. **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 protocol does not advance. **Turn on writing at protocol start** causes writing of sampled sweeps to disk (normally controlled by the **Write to disk at sweep end** checkbox in the sampling control panel) to be automatically turned on when the protocol starts. **Reset pulse steps at protocol start** causes all varying pulses defined in the pulses dialog to be reset to their initial state when the protocol starts.

The table below defines the protocol steps. There are ten steps in a protocol, each one specifies a **State**, **Repeats** 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. A step is marked as being the end of the protocol by setting it's next step to zero.

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 number 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.

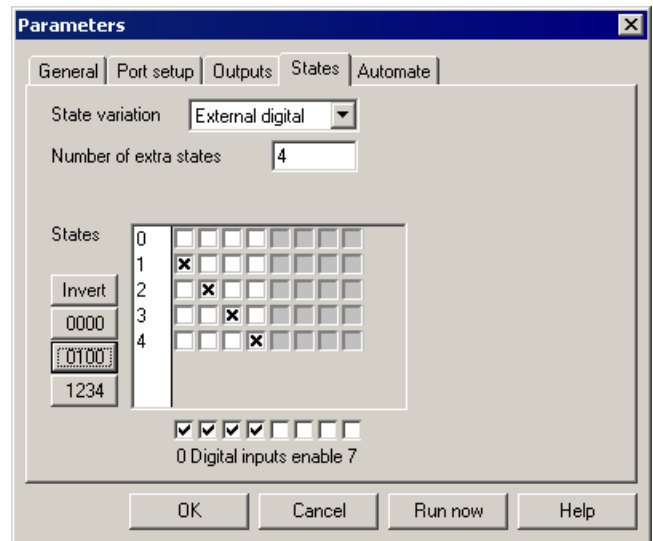
The items below the repeat count control what happens when protocol execution ends. The **At end** selector selects between **Finish** and a protocol that will be 'chained-to', chaining to a protocol allows more complex sequences than is possible with just ten steps. If the **At end** selector is set 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 that 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, so it is meaningful to allow a protocol to chain to itself..

The checkboxes at the bottom control what happens when a protocol actually finishes and 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.

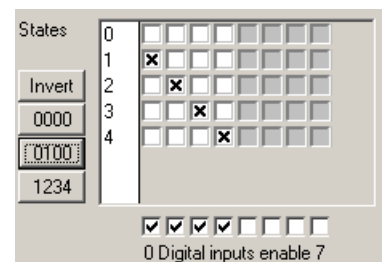
External digital states

External digital states are the simplest but most limited form of multiple states. External equipment generates a digital code of up to 8 bits corresponding to the current experiment state. Signal reads this code from the 1401 digital inputs 8-15 at the end of a sampling sweep and uses it to set the state for each sampled data frame. There is no software control over the state values, so all of the state sequencing control are not present and no difference in Signal's behaviour for the different states.



The number of extra states is set as normal, while the rest of the states page is used to defines the mapping from digital input values to the frame state. The **Digital inputs enable** checkboxes at the bottom of the dialog control which inputs are used for this; unchecked inputs are ignored.

The main area in the centre of the dialog defines, for the enabled digital input bits only, the bit patterns that correspond to the various states. The bits 0 to 7 are shown in a row, with 0 on the left. A blank location corresponds to a zero bit (low or 0 volts), a checked location gives a set bit (high or 5 volts). When Signal is sampling, it reads 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 frame state is set to the first one that matches. If no match is found then the frame state is set to zero. The bits for state zero are ignored. See the Sampling data chapter for details of the digital input connections. Note however that if an input is not connected then it will read as high. It is therefore important to disable any unused inputs.



The buttons to the left of the digital bit controls set or modify the state bit patterns to various useful ways. These 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. This is useful for clearing the control before entering new patterns.
- 0100 This sets the 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 input bits.
- 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. Again, the pattern is not adjusted to skip disabled digital input bits.

When sampling using **External digital states**, Signal behaves much as it does with multiple states disabled. The only difference is that the state value for sampled data frames varies according to the digital inputs.

Static outputs states

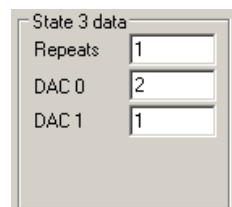
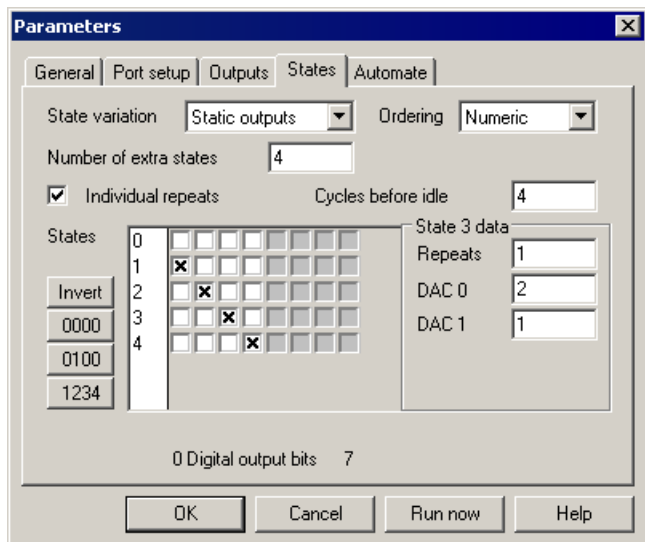
Static outputs states are more complicated than **External digital states** as Signal controls the state of each sampled data frame. At the start of a sampling sweep, Signal writes values to the 1401 DACs and digital outputs according to the current state of the experiment. These outputs could select the stimulus or have other effects as required.

In addition to defining the values to be output, **Static outputs** states requires that you also set up how Signal will sequence the states. This adds quite a number of extra controls to the dialog, for details of these see the section on *States sequencing* above.

The main box in the centre of the dialog now defines the values to be written to the digital outputs for each sweep. Because these are now digital outputs, the enables for the bits to be used are in the **Outputs** page, but otherwise this and the adjacent buttons behave in the same way as for **External digital** mode. The actual digital outputs and connector pins used are the same as for pulse outputs (see the *Pulse outputs during sampling* chapter).

The **State data** box to the right of digital output bits 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.

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 the other outputs. When sampling using Static output states, controls for the state and states sequencing are now provided, see the section *Controlling multiple states online* below.



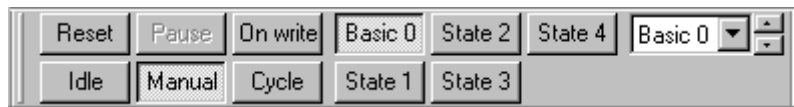
Controlling multiple states online

When sampling using dynamic outputs (or static) multiple states, Signal provides a states control bar to allow you to control the sampling state and sequencing. The control bar can be docked at the edges of the Signal application or be left floating. When sampling starts the states control bar will be shown initially and can be hidden or shown during sampling by using the sample menu or a popup menu that is displayed if you right-click on an unused part of the toolbar area.

The states control bar is set up differently depending upon whether numeric/random or protocol sequencing are being used; without protocols extra buttons for controlling states sequencing are provided whilst when using protocols extra items to select and run a protocol are provided along with optional individual protocol buttons.

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 sampling and states sequencing to idle. It switches manual control of states on, 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 always begins with manual control selected. |
| On write | Press this button for automatic states sequencing with the states changing only if the previous data was written. 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 was saved to disk. Rejected sweeps will repeat the same state until the data is accepted. This allows 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 data is written to disk or not. |
| Basic 0 ... | Press these buttons when in manual mode to use a state immediately. Buttons for unused states are hidden, as are buttons for states numbers greater than 9. If you have set labels for the states in the pulses configuration dialog, these labels will be used in the buttons instead of the standard names. During automatic states sequencing these buttons are disabled but they are pressed automatically to show the state 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. The 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 used instead of the simple state name. Selecting a state uses it immediately. During automatic states sequencing the current state is shown. |

Protocol ordering When protocols are used, some of the states bar controls are hidden and other controls are now displayed. Those controls that are retained behave in the same way as described above, while the **On write** and **Cycle** buttons are hidden because the individual protocols select between these two modes 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 and protocol run 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 switch out of manual mode and start executing the selected protocol immediately, terminating any other executing protocol. Like the protocol selector, if all protocols have individual buttons this button is not shown.
- Buttons** These are buttons created for the individual protocols which 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 is 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 executing, by the user pressing the **Idle** or **Manual** buttons in the states control bar or by another protocol being started by whatever means.

Auxiliary state hardware

In addition to the standard multiple states facilities that control 1401 outputs, it is possible to install support for auxiliary states hardware with Signal. 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 normally 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 settings for the hardware for each state to be defined. In addition, the `SampleAuxStateParam()` and `SampleAuxStateValue()` script language functions can be used to read and write the settings.

See the appendices for details of the individual auxiliary states hardware supported.

The **File** menu is used for operations that are mainly associated with files (opening, closing and creating) and with printing.

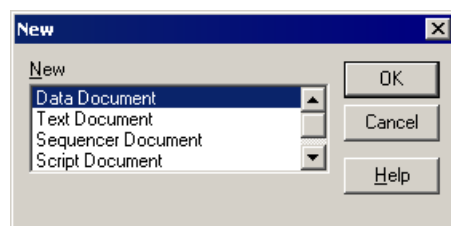
The first section of the menu is used for operations that create a new window. The second section closes windows and saves the contents of a window and also includes commands for saving data under a new name and exporting data to a new data file or a different data format. It can also be used to revert a text-based view to its last saved state. If you have Mail enabled on your system it will include a **Send Mail** command. The third section holds a control for how changed data is written back to the data file. The fourth section of the menu loads and saves the configuration files that control data sampling. The fifth section controls printing. Both the printer setup and the commands to print a window are found here. The sixth section holds a list of the most recently used files. The final section is the standard way out of any Windows application, the **Exit** command.

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, XY documents, sequencer documents, script documents and text 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). Sampling documents are not initially stored in memory, like most new files, but are kept on disk. Until they are saved after sampling they are temporary files in the directory set in the Signal preferences. 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. 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

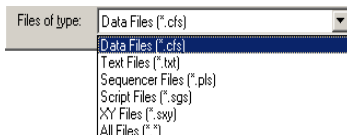
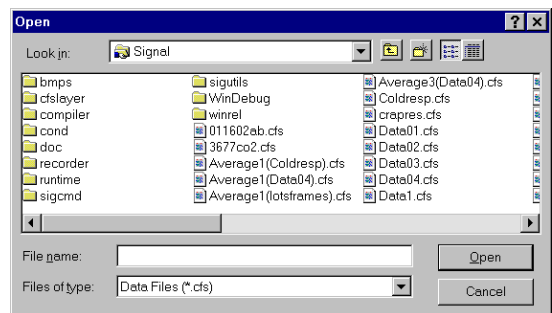
Signal stores some of its preferences in a file called `cfsview.sgp`. This file holds the position of the application window on the screen, the colour palette and 'use colour' switch. The main preferences information from the preferences dialog is now largely stored in the Windows registry. Signal initialises 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.

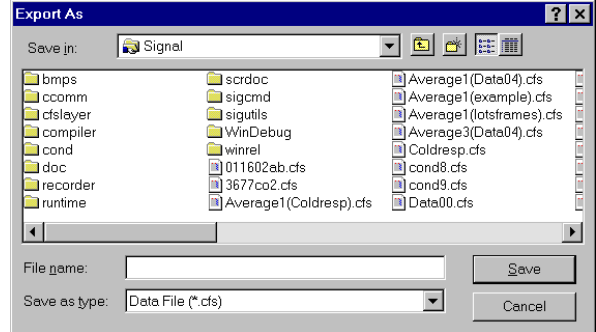
If a read-only CFS data file is opened, you will be warned that the file cannot be modified.

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 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 on the file type.
- Supported formats include the SON files used by CED programs such as Spike2; Spike2 for Macintosh files as well as data from several 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 to convert files is `FileConvert$()`.
- Import Op/Ci** Signal can import idealised traces from the CED Patch software. These are files of the type `*.res` or `*.r??` generated by the Pulsed Analysis or Continuous Analysis programs. 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 in the Analysis menu (see the *Analysis menu* 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 (in Windows 95 and NT 4.0 etc). If you use this command on a new memory 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.
- 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, this frame 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 the 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 and do not save changes until the entire document is saved.

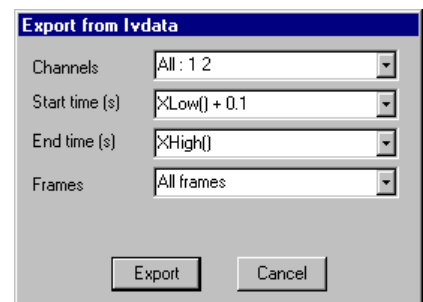
Other documents Text, script and XY documents are held in memory. Changes made to them are not permanent until the document is saved to disk.

Export As This menu item, available only when the current window is a data 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) for a scaleable image and Bitmap file (*.bmp) for a copy of the screen rectangle containing the window. Select one of the formats and set a file name, then use the **Save** button.



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, individual channels, selected channels or enter a list of channel numbers directly. You can export all frames, the current frame, all tagged or untagged frames, frames with a given state, or you can enter a list of frames.



Once you have selected the channels, frames and a time range, click the **Export** button to write the data to the new data file.

Text file This is the same as the **Edit** menu **Copy As Text** command, but with the output sent to a text file and not the clipboard. See the **Copy As Text** command for details of the dialogs.

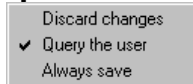
Bitmap file This copies the screen area containing the window to a file. Make sure that the window is completely on the screen and that it is not covered by any other window. You should only use this option when the image you require is an exact copy of the screen. If you need to scale the image, or want to edit it, a Windows Metafile copy is usually better.

Metafile This copies the window as a Windows Metafile. 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.

Revert To Saved You can use this command with a text file or a script file. The file changes back to the state it was in at the time of the last save to disk.

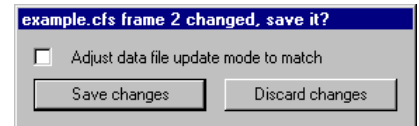
Send Mail

If your system has support for Mail installed (for example Outlook Express), you can send any document from Signal to another linked computer. This command vanishes if you do not have mail support. 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 cannot be sent as Signal keeps them open so they cannot be copied.

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. Any changes to the frame data can be written back to the disk, or the changed data 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, use this command to change the mode for a particular file.

If Query the user mode is selected, the **Save changes** dialog will appear as required. This allows changed frames to be individually discarded or saved, check the **Adjust data file update mode to match** checkbox if you don't want to see the dialog again.

**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 that 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.

For file, memory and XY views the important options that are always present include the paper orientation (portrait or landscape), the paper source (if your printer has a choice), the printer margins and the choice of printer.

In text-based views, in addition to page margins you can set header and footer text to appear on each page. If you include `&f` in the text it is replaced by the file name, `&p` is replaced by the page number and `%c` is replaced by the file or system time.

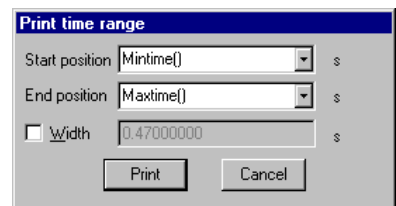
Print preview

This command allows you to view the current document as it would be printed. You can preview file and memory views and text based windows. You can zoom in and out, step through pages of multi-page documents and print using a toolbar at the top of the screen. Use the **Close** button to leave this mode.

**Print visible,
Print and
Print selection**

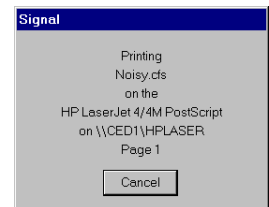

These commands print data views, XY views, cursor windows and text-based windows. The scroll bar at the bottom of the window is not printed. **Print selection** prints the selected area of a cursor or text window. **Print visible** prints the visible data in the current window. **Print** or its **Ctrl+P** 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 will be provided in later versions of Signal.

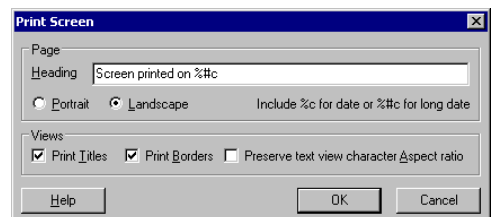


All the print commands open the standard print dialog for your printer. You can set the print quality you require (the better the quality, the longer the print takes) and you can also go to the Setup page for the printer. Once you have set the desired values, click the **Print** button to continue or 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.

**Print screen**

This command will print all the file, memory, XY and text based views on the screen to one printer page. The screen is scaled to fill the printer page and the views are scaled to occupy the same proportional positions on the printed page as they do on the screen. The command leads to a dialog in which you can set a page title. If you use %c in the page title then the date and time will be included in the title in compact form, e.g. 23/11/2004 15:47:31. Using %#c will include the date and time in long form, e.g. 23 November 2004 15:47:31. The exact format will depend on the regional settings and preferences set in the operating system. Landscape print orientation often looks better than portrait as the output aspect ratio is closer to that of the screen. You can also choose to print view titles and choose to draw a box round each view. The fonts used by the printer for text views may appear stretched in order for the view's proportional size to be preserved. If you prefer the font to have an aspect ratio closer to that used on the screen then check the box to have it do this.

**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.

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 In a text-based view this undoes edit operations. 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). Changes made to an idealise 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 than the visible data, only the visible data is modified. This is not true for XY views. 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. If you are looking at frame zero in sampled data, **Clear** erases any overdrawn traces. **Clear** removes everything, **Delete** removes the current selection.

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 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.

The screenshot shows the 'Text output configuration' dialog box. It has a title bar 'Text output configuration'. Inside, there's a section for 'Interpolation method' with a dropdown set to 'Linear'. Below that is a table-like structure with columns 'Data', 'Time', and 'Headings'. Under these columns are checkboxes for 'Waveform', 'Marker', 'Fitted', and 'Idealised'. 'Waveform' is checked under 'Data', 'Time', and 'Headings'. 'Marker' is checked under 'Time'. 'Fitted' and 'Idealised' are checked under 'Headings'. Below the table are input fields for 'Decimal places (data, time)' with values '5' and '5', 'Field width for all items' with value '0', 'String delimiter' with value '"', and 'Separator' with a dropdown set to 'Tab'. At the bottom is an unchecked checkbox 'Add header to text block' and 'OK' and 'Cancel' buttons.

In the next section you can choose whether to output **Headings**, **Data** values and **Time** values (where appropriate) for each of the **Waveform**, **Marker** and **Fitted** data by checking the boxes.

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.

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.

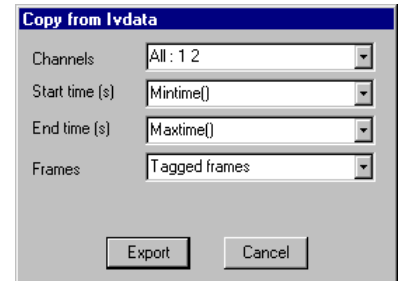
```
"s"      "Keyboard"
20.20608 "a"
21.24544 "n"
```

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. When you are satisfied with the selection, click on **Cancel** to quit or **Export** to start the copy process.

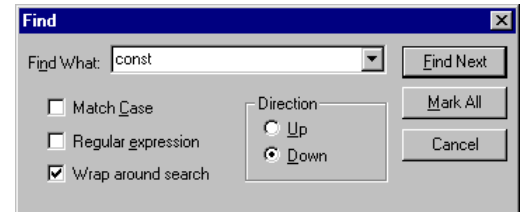


Copy As Text (XY view)

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

These commands move the text cursor to the next occurrence of a string in a text-based view. The search is normally insensitive to the case of characters; if you want to **Match Case** in a search, check the box. The search starts at the cursor position and stops at the end of the document unless the **Wrap around search** box is checked in which case the search continues at the start of the document after reaching the end. Searches are line-based (you cannot search for text that spans more than one line). If you check the **Regular expressions** box, you can include pattern matching characters in your string. The simplest pattern matching characters are:



search box is checked in which case the search continues at the start of the document after reaching the end. Searches are line-based (you cannot search for text that spans more than one line). If you check the **Regular expressions** box, you can include pattern matching characters in your string. The simplest pattern matching characters are:

- ^ Start-of-line marker. If you include this character it must be at the start of the search text. The following search text is matched only if it is found at the start of a line.
- \$ End-of-line marker. If you include this character it must come at the end of the search text. The preceding text will only be matched if it is found at the end of a line.
- .

Matches any character.

To treat these special characters as normal characters with regular expressions enabled, put a backslash before them. A search for "^\\^\\.\\." would find all lines with a "^" as the first character, anything as a second character and a period as a third character.

To search for a character list, enclose it in square brackets, for example "[aeiou]" will find any vowel. For a character range use a hyphen to link the start and end of the range. For example, to find any alphanumeric character use "[a-zA-Z0-9]". If the search is not case sensitive you can omit one of "A-Z" or "a-z". To include the "-" character in a

search, place it first or last in the list. To search for any character that is not in a list by placing a “^” as the first character. To find a non-vowel character use “[^aeiou]”.

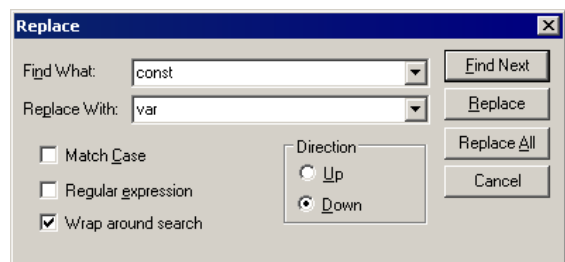
There are three more special search characters that control how many times to find a particular character. These characters follow the character to search for:

- * Match 0 or more of the previous character. So “51*2” matches 52, 512, 5112, 51112 and so on. You can also match 0 or more of a character pattern, for example: “h.*l” matches hl, hel, hail, horribl and “B[aeiou]*r” matches Br, Bear, Beer, Beeeeaaaooor and so on.
- + Match 1 or more. The same as “*”, but there must be at least one matching character.
- ? Match 0 or 1. The same as “*”, but matches one character at most.

The Find Again command repeats the last search with the same options.

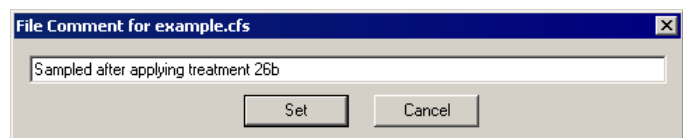
Select All This command is available in text-based views and selects all the text, usually in preparation for a copy to the Clipboard command.

Replace This command is used to replace text with different text when you are working with a text-based view. Press Find Next to move the text cursor to the next occurrence of the search string, Replace to replace the search string found with the new text and Replace All to find and replace all occurrences



of the search text. The search is normally insensitive to the case of characters, if you want to Match Case in a search, check the box. The search starts at the cursor position and stops at the end of the document unless the Wrap around search box is checked in which case the search will continue from the start of the document after reaching the end.

File comment 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.

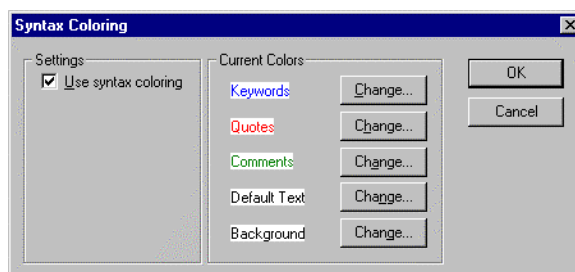
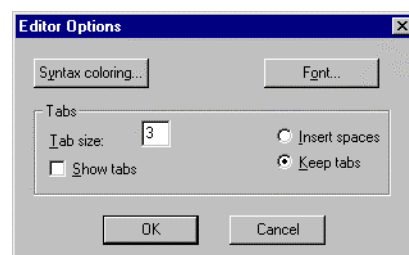
Frame comment This command is available with file and memory views, it allows you to see and edit the frame comment. The frame comment is a single line of text attached to each frame in the data file that is available for any purpose.

Preferences The Edit menu Preferences command opens a dialog in which you set items relating to the screen appearance, metafile output, hardware connections and where to store data during sampling. Most of 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.

Display The Display tab contains preferences relating to the way data is shown on the screen

Default font for data views Signal data views record their font so that they appear the same when re-opened, but they always have a default font when first created. This font is shown in the rectangular area within the default font box, and can be set using the **Set Font** button, which provides a standard font selector. Once data views are open, their fonts can be individually set using the **View menu Font** command.

Editor settings These buttons lead to dialogs for the standard settings for script, sequencer and text files and to control syntax colouring. The **Font...** button opens a font dialog that sets the font used when you open a file. You are restricted to fixed pitch fonts in the editor. The **Tab size** field sets how many spaces to move for each press of the **Tab** key. You can also choose to keep the tabs as tab characters, or have tabs converted to spaces.



The **Syntax colouring...** button opens a dialog where you can enable syntax colouring for script windows. Enabling it for other windows has no effect. You can also change the colours used for syntax colouring and preview the effect.

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.

Data The **Data** tab controls the way data is stored and exported.

Data export format The data export format items control decisions on how waveform data is written to CFS data files. 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 are less accurate as data are converted to 16-bit integer values and can overflow. 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 set by the destination file, or it is forced to scaled integer when sampling data using the 1401, but when creating a new CFS data file by other mechanisms (saving a memory view or exporting to a CFS data file), the format used is controlled by this field. Select **Real** for maximum data accuracy, **Integer** to produce smaller data files or for SIGAVG compatibility.

The **Keep calibrated zero at zero volts** checkbox 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 calibrated data becomes 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.

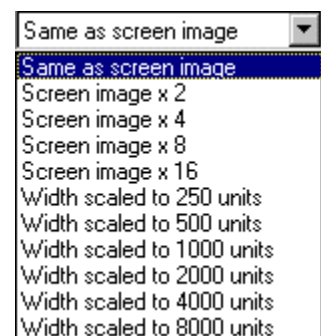
Signal saves the current preferences to the registry so they are preserved next time the program is run.

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.

Metafile scaling 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 2004).

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.

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.

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.

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 optimise 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.

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

Programmable signal conditioners (see the *Programmable signal conditioners* chapter) are controlled through communication (serial) ports. Check the *Dump errors...* box to write diagnostic messages to CEDCOND.LOG in the current folder. The lower half of the dialog is used to display the current status of the conditioner.

Script

This tab contains a couple of check boxes regarding the script editor.

Save modified scripts

Save modified scripts when run can be checked so that any changes to a script will be automatically committed to disk each time the script is run.

Enter the debugger

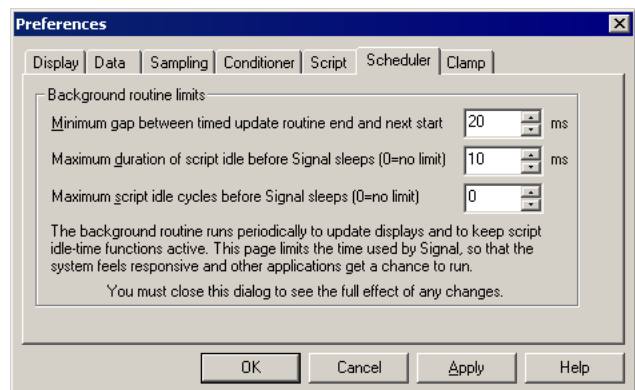
If Enter the debugger when a script has a runtime error is set then if a runtime error in a script happens then the script view will be displayed with the debugger running. This allows the user to check on the values stored in the variables as well as looking at the call stack.

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:

- 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.

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 etc then use this page to turn off clamping features to avoid confusion.

Show event details

The Edit menu **Show event details** command is available when the current view contains an idealised trace of patch clamp 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 details dialog gives information about the selected event in textual form. The **Start time**, **Duration** and **Amplitude** can all be changed. 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 start time of the following event. 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 Analysis menu chapter.

Previous and Next The **Previous** and **Next** buttons step the current selection of event through the idealised trace.

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.

Merge **Merge** will combine 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 adopt its amplitude from there.

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.

Tips for fitting Often the fitting routines may produce unexpected results and there can be a number of reasons for this.

1. There is too much level data at the start and end of the fit. Restrict the visible area to only view the transitions you are interested in fitting.
2. The initial guess is too good. When the initial guess for the fit is so close to the “correct” solution the fitting routines can be unable to detect which direction to move the parameters to improve the result. This can be overcome by worsening the initial guess manually first. Dragging the amplitude of a principle event usually works best.
3. The initial guess is too poor. Use the other edit functions to produce a more plausible guess.
4. 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.

10

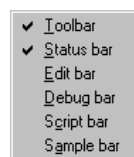
View menu

This menu, divided into six regions, controls the appearance of the data and XY views. The first region holds commands to show and hide the application toolbar, edit bar and status bar. The second holds commands that move the display from one frame to another within the data. These commands are not available for XY views, which have a single frame. The third region holds controls for frame overdraw mode and the frame list used for overdrawing. The fourth region holds commands for zooming in and out in both the x and y directions. The fifth region controls the channels and other items that are displayed on the screen and the waveform and marker drawing mode. The final region controls the fonts used and the use of colour in the data and XY windows.

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, 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 showing the last frame filed until the frame is manually changed.

Goto frame

This command (shortcut Ctrl+G) opens a dialog that gets the frame number to move to.

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 maintained by Signal. See the Analysis menu chapter for more details of the frame buffer.

Overdraw frame list

This command (shortcut Ctrl+D) switches the display between normal mode, displaying the current frame, and overdraw mode, which also displays all frames in the frame display list. When overdraw mode is enabled, the menu item is shown checked.

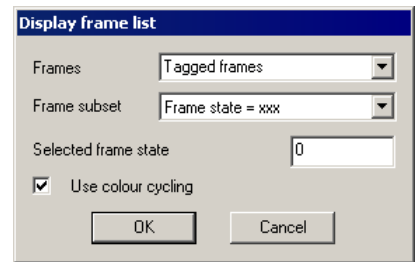
The frame display list holds frames to display in addition to the current frame. These additional frames use different colours from the current frame trace. They can either all have the same colour, set using the View menu **Change Colours** command, or each frame in the list can have a different colour.

Add frame to list

This adds the currently displayed frame to the frame display list. If overdraw mode is enabled, all frames in the frame display list are displayed along with the current frame. 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.

Frame display list...

This command opens a dialog that manipulates the frame display list and also controls the manner in which the frames are displayed. The upper part of the dialog allows you to define which frames are in the display list. You can either enter a list of frame numbers directly, or you can use the drop-down list to select a category of frames from: **All frames**, **Current frame**, **Buffer**, **Tagged frames**, **Un-tagged frames** and **Frame state = xxx**. A second field allows a subset of these frames to be defined. The frame list is dynamic in that if, for example, you request all tagged frames and subsequently tag a frame it will be added to the overdraw list.



If you choose the special category **Frame state = xxx**, an extra field appears: **Selected frame state**. You should set this field to the state number that you want to display.

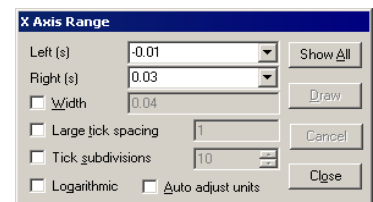
The **Use colour cycling** checkbox selects drawing of data from each frame in a different colour; Signal contains a fixed table of 18 colours that it uses for this mode of operations. If the checkbox is clear then all the display list frames are drawn using the same colour; this colour can be set from the colour setup dialog.

**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 displayed area is moved backwards. If the result of expanding would display more data than exists in the frame, all 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.



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.

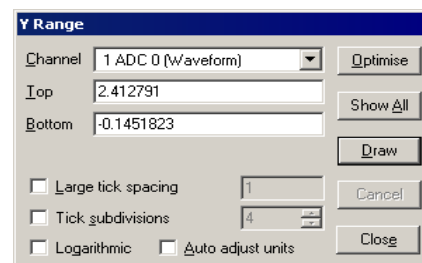
The **Auto adjust units** option will cause the units displayed on the axis to switch to multiples of powers of 10 in order to keep the figures sensible when zoomed well in or well out. This option affects only the axis; the units used by the cursors etc will still be the same. Checking the **Logarithmic** 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 labeled 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 the **Ctrl+Y** shortcut open the **Y Range** dialog. The dialog sets the y axis range and style for data or XY view channels. The **Channel** field chooses one, all or selected channels (see the *Getting started* chapter for information about selecting channels). If more than one channel is selected, the displayed settings are from the first channel.



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 and from 0 to the estimated event rate for marker channels drawn with a frequency axis. Both buttons close the dialog. To optimise without opening the dialog, right-click a channel and select the optimise option from the context menu.

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. **Cancel** undoes any changes and closes the dialog. **Close** accepts changes and closes the dialog.

The remaining options are all identical to those already described under **X Axis Range** (see above).

Standard Display

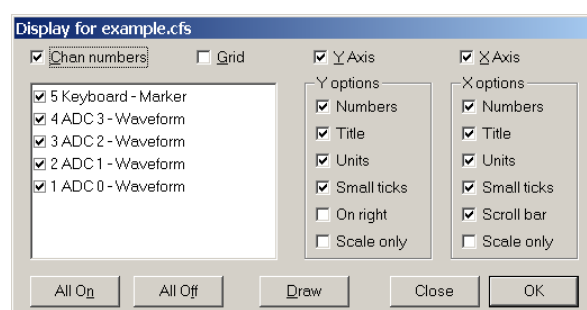
This command sets the current data, memory or XY view to a standard state. In file and memory views it turns on all channels in a standard display mode and size and ordered as set in the **Edit menu Preferences**, with x and y axes on and grids off. In an XY view, all channels are made visible, the point 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.

Customise display



This command opens a dialog that sets the channels to display in a data or XY view and the display of x and y axes, grids and the horizontal scroll bar.

Check the **Chan numbers** box for channel numbers in file and memory views. **All On** and **All Off** select all or none of the channels. **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

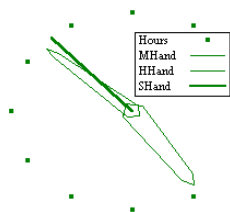
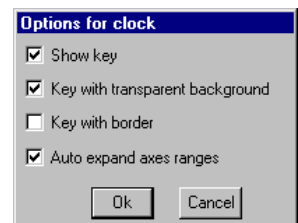
Use this dialog to view and edit data view channel information. You can open it by double-clicking a channel title, from the **View** menu or by right-clicking the channel to open the context menu. You can edit the **Title** of the channel set by the **Channel** field. The remaining fields are hidden 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 and the shortcut key **Ctrl+I** display information about the current data document. Currently, it just 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 it is the number of points included which is displayed.

Options

This command is for XY windows only and opens the XY options dialog. The main purpose of this dialog is to control the XY window “key”. The key is a small region that can be dragged around within the XY window that identifies the visible XY data channels. For each channel it display the name and draws the line and point style. This dialog also has a checkbox that controls the automatic expansion of the axes when new data is added.

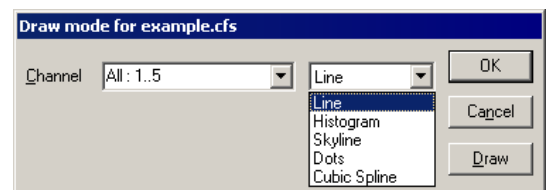


This example (made by the `clock.sgs` script in the `Scripts` folder) shows the key. You can choose to make the key background transparent or opaque and choose to draw a border around the key. If you move the mouse pointer over the key, the pointer changes to show that you can drag the key around the 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.



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

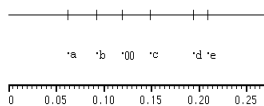
There are five drawing modes for waveform channels: Line, Histogram, SkyLine, Dots and Cubic Spline. Line joins the data points with straight lines. Skyline joins points with horizontal and vertical lines. Cubic spline joins the points with smooth curves based on the assumption that the first and second 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.

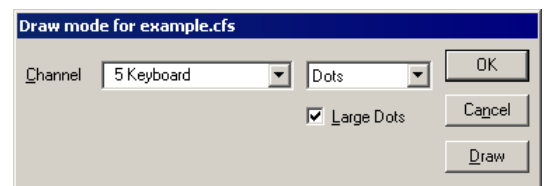
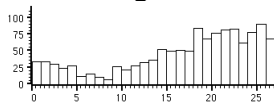
If each point of your data can be modelled as a constant "real" value to which is added normally distributed noise with zero mean, then you would expect the measured mean value to lie within 1 standard error of the mean (SEM) 68% of the time, or within 2 SEM 95% of the time. The standard deviation represents the width of the normal distribution of the underlying data at each data point.

Marker draw modes

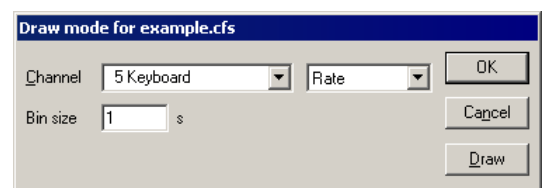
This menu item is used to set the display mode for marker data channels. You can set the mode for a single channel, all suitable channels, or any subset of the suitable channels. When this command is used the marker draw mode dialog is provided. This dialog changes depending on the display mode selected, there are three modes available; Dots, Lines and Rate.

Dots and Lines mode

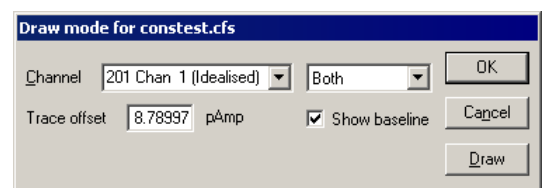
The simplest method is to draw the marker channel as dots. You can choose large or small dots (small dots can be very difficult to see on some displays). You can also select Lines in place of Dots. The picture above left shows the result of both types of display. If you select lines mode the display of marker values is suppressed.

**Rate histogram mode**

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**

There are three draw 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 $\text{erf}()$ defined as the integral of the



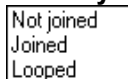
Gaussian function. This is because it is assumed that the raw data was filtered using a close approximation to a Gaussian filter. The cut-off frequency for this filter can be changed by using the channel information dialog. For an idealised trace generated using threshold crossing the cut-off frequency 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 you get bored waiting. Simply 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 other fields are:



Join style



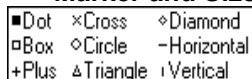
There are three channel join styles. If a channel is Not joined, data points are not connected. If the channel is Joined, each point is linked to the next by a straight line. If the channel is Looped, the points are joined and the final point is linked back to the first point. The Line type and Width fields define the lines joining the points.

Line type and Width



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



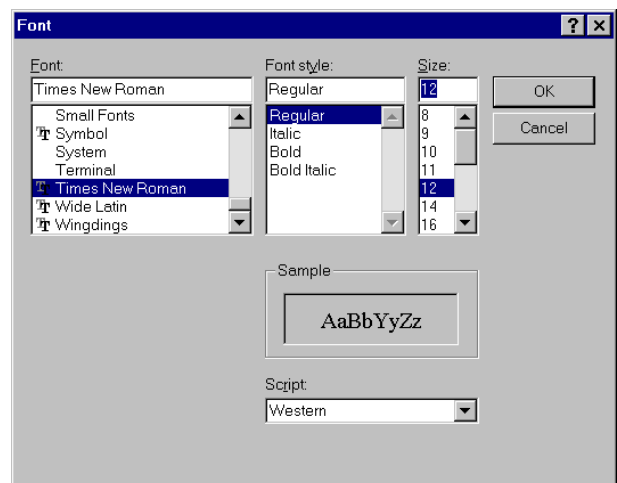
The Size field sets how far, in points, the markers extend around their screen position. A size of 0 makes the markers invisible. There is a wide range of marker styles to choose from, including boxes, circles, triangles and vertical and horizontal bars.

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.



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.

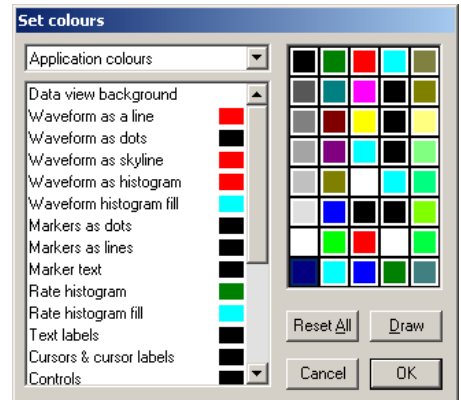
Change Colours



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:

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
Waveform as line
Waveform as dots
Waveform as skyline
Waveform as histogram
Markers as dots
Markers as lines

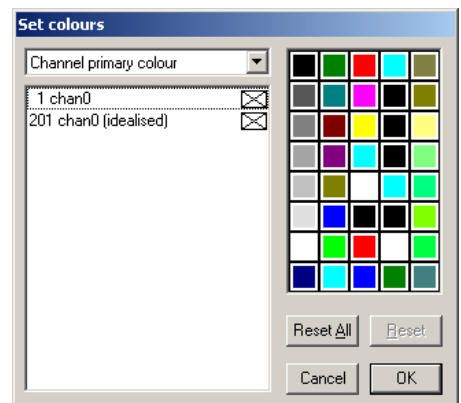
Marker text
Rate histogram
Rate histogram fill
Text labels
Cursors and cursor labels
Controls
Data display grid

Axis markings & text labels
Tagged frames background
Frame list traces
XY view background
Standard deviation/SEM
Fitted data

See the Edit menu Preferences for text view colours.

Channel primary colour Channel secondary colour Channel background colour

These two 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.



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()`.

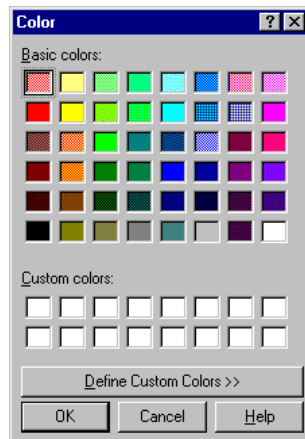
XY channel colours

This page changes the XY channel 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.

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.

The **Analysis** menu creates memory or XY views that hold analysed data from data channels from other data documents and provides access to other analysis functions. The menu is divided into seven regions.

The first region holds commands that operate on existing memory or XY views. The second is used to create new memory or XY views holding data created by built-in Signal analysis mechanisms. It also contains the curve fitting option. 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. It also includes the multiple frames dialog, which carries out operations on many frames. The fifth region holds channel data modification commands. The sixth region controls frame tagging. The last region is the digital filtering command, described 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**. 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.

Waveform Average...
Amplitude Histogram...
Auto-Average...
Power Spectrum...
Leak Subtraction...

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

This analysis averages a waveform 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 memory view x axis starts at zero. If this is clear, 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.

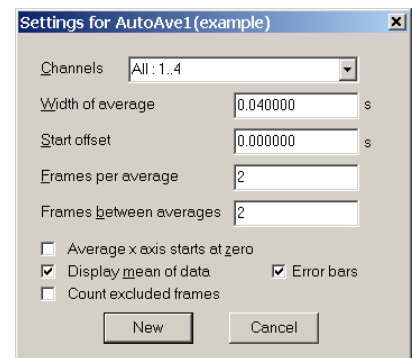
The **Display mean of data** checkbox selects between displaying the mean data value or the sum of all sweeps added into the average.

If you check the **Error bars** box, extra information is saved with the result so that you can display the standard deviation and standard error of the mean of the resulting data. The **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 analysis averages waveforms as for the standard waveform average processing, and automatically produces multiple averages. Each average frame is generated from a fixed number of frames of source data. The first frame used for each average is offset from the previous average start by a set number of frames. The **Settings** dialog holds fields for the channels, the width of the average, the data start offset and the frames and frame start offset 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** and **Display mean of data** checkboxes. **Count excluded frames** is described below.



The **Frames per average** item sets the number of source frames that are used to make up the first frame in the 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 we are using 2 frames to make up each average, setting **Frames between averages** to 2 will use frames 1 and 2 for the first average, 3 and 4 for the second average and so on. Setting **Frames between averages** to 4 means that frames 1 and 2 go to average 1, frames 5 and 6 are used for average 2 while frames 3 and 4 are unused. If you set **Frames between averages** to less than **Frames per average**, then the data for each will overlap with some frames being used for more than one average.

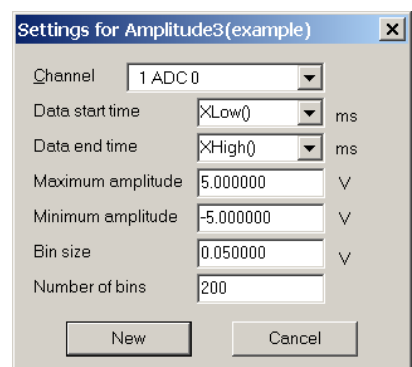
If **Count excluded frames** is not checked and you want to process only un-tagged frames (see below), then Signal will scan the data file for enough un-tagged frames to form each average. If it is checked then tagged frames are not included in the average but are included when working out which frames to add. Thus some of the averages would be formed from less frames. Averages formed from no frames are set to zero.

If you check the **Error bars** box, extra information is saved with the result so that you can display the standard deviation and standard error of the mean of the resulting data. The **Waveform Draw Mode** dialog controls the display of the error information.

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.

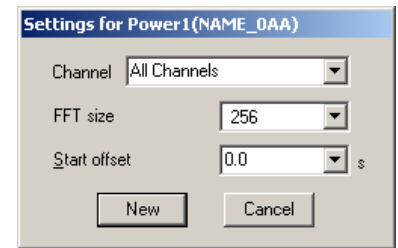
Amplitude Histogram

This analysis creates a memory view that is a histogram of the number of data points found at a particular amplitude range of a channel in the file. A time range is given in which the data will be analysed. The **Maximum** and **Minimum amplitudes** determine the start and end points of the x-axis in the memory view. These fields of the dialog are linked to the **Bin size** and **Number of bins**. If the amplitude range is changed, the **Bin size** changes to fill the new range and keep the **Number of bins** constant. If the **Bin size** is changed then the **Number of bins** changes 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

This analysis creates a memory view that holds the power spectrum of a section, or sections of data. If multiple sections are processed the result is an averaged power spectrum. The result of the analysis is scaled to RMS power, so it can be converted to energy by multiplying by the time over which the transform was done. There are three fields to set in the dialog: the waveform channels to analyse, the number of points in the Fast Fourier Transform (FFT) used to convert the waveform data into a power spectrum and the start offset within the frame for the data. The channels and offset fields behave the same as their equivalents in the waveform average 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 4096. You set the FFT block size from a pop-up menu. The way the maths works out, the resulting 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

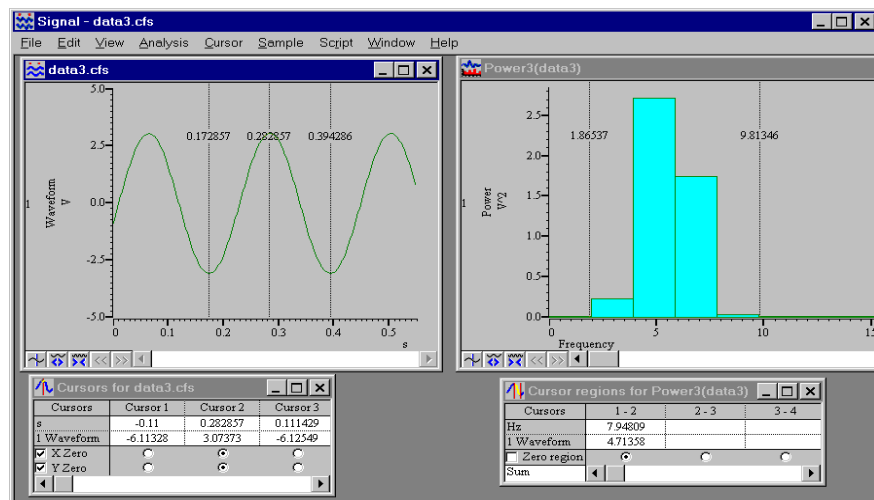
Another feature of the FFT is that the input waveform is assumed to repeat cyclically. This means that the maths treats the block of data as though it was taken from an input consisting only of that block, repeated over and over again. In most waveforms this is far from the case; if the block were spliced end to end there would be sharp discontinuities between the end of one block and the start of the next. Unless something is done to prevent it, these sharp discontinuities cause high frequency components in the result.

The standard solution to this problem is to taper each data block to zero at the start and end, so that the start and end join smoothly. This is known as *windowing* and the mathematical function used to smooth the data is called the *window function*. The use of a window function causes smearing of the data, and also loss of power in the result. A discussion of the relative merits of different window functions is beyond the scope of this manual. We use a raised cosine window and compensate for the loss of power it causes.

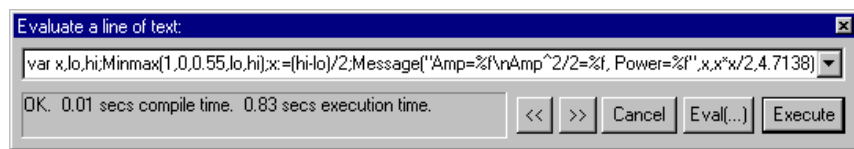
Power spectrum of a sine wave

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 volts². 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.

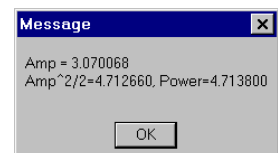
If the sampled waveform was a perfect sine wave we would predict a RMS power of 4.6818 volts² from this waveform ($3.06^2/2$). The cursor analysis of the power shows a total power of 4.7135 volts². This is about 0.7% above the predicted result for a perfect waveform.



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:



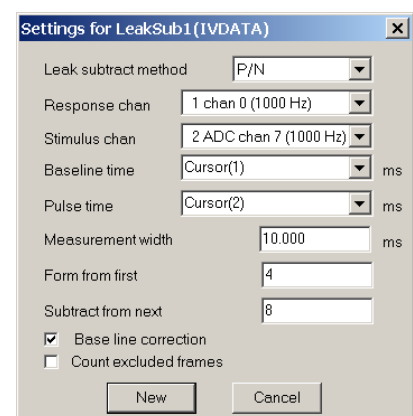
You can use the **Evaluate** command in the **Script** menu if you want to try this. It gives a slightly larger value for the amplitude and now the power calculated from the amplitude and the measured power differ by 0.025%. For an explanation of the text in the **Evaluate** window see *The Signal script language manual*.



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

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. The basic technique is to use a small stimulus, one that does not cause the cell membrane ion channels to turn on, and measure the current flow through the membrane impedance (made up from resistive and capacitive components). This 'leak' measurement is then scaled to give the expected non-ionic conductance during a larger pulse and subtracted from the recorded traces to leave only the ion-channel effects. Normally an average leak trace is assembled from a number of small pulses, to minimise the effects of noise, but this makes no substantial difference to the technique. Leak subtraction makes special use of



two channels; the stimulus channel which is used to measure the amplitude of the pulse so it can be scaled, but is not modified by the analysis, and the response channel which is the only channel modified by the leak subtraction process. All other channels are ignored and copied unchanged.

The **Settings** dialog holds fields for the leak subtraction mode and the channels for the stimulus and the response to be corrected. The **Baseline time** is a time within the sweep where there is no stimulus and the **Pulse time** is a time where there is a stimulus. These are used to measure the stimulus amplitude; the level measurements are averaged over the requested width. The last two edit fields specify the frames to use to calculate the theoretical scaled trace and which frames to subtract the leak from. These fields are interpreted in different ways depending on the leak subtraction method described 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 is unchanged.

The **Leak subtraction method** can be set to **Basic**, **P/N**, or **States**. These three modes are very similar, the only real difference is how the leak trace is assembled. In **Basic** mode, the leak data is assembled from a fixed contiguous set of frames; these frames are set in the dialog by entering a first and last frame. All the frames processed use this set of leak data, frames contributing to the leak data are automatically skipped. In **P/N** mode, the frames processed provide the leak traces, the first *n* frames processed are used to make the leak, then the next *m* frames are processed using the leak, this cycle continues until all frames are done. The values for *n* and *m* are entered in the settings dialog. In **States** mode, the leak data is assembled from all frames 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.

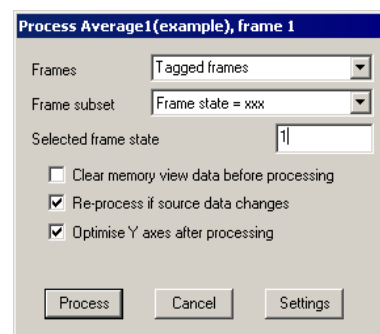
As for **Auto-Average** if, for instance, we only process un-tagged frames and there are some tagged frames in the file, the process will continue to search for un-tagged frames until the required number have been found to complete the formation or subtraction of the leak. If **Count excluded frames** is turned on, however, then even frames that are excluded from the process will count in the total number of 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 and if not set doesn't accumulate data into the existing frames but rather appends more frames to the memory 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 a similar XY view is the current view. When you use it a dialog prompts you to select the frames of the source data document to process. 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.

The simplest way to use this dialog 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 displays a field into which you can enter the state code to use.



The frame list values are evaluated when the **Process** button is used; the frames are processed and the results added into the memory view data. The dialog window will remain on screen until removed with **Cancel**. This means that you can set the frames to **Tagged** then adjust the frame tagging in the source data document and click the **Process** button to analyse the selected frames. Some sets of frames have subsets which can be used to give a more specific list of frames to use.

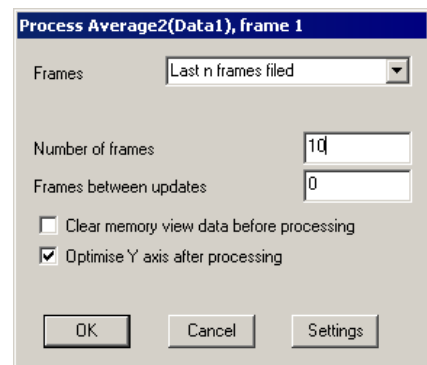
If you check the **Clear memory view before process** checkbox, the memory view is cleared before the results of the processing are added. The **Reprocess if source data changes** checkbox enables automatic re-processing. Automatic re-processing is optimised to try to prevent unnecessary work, but can 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 the **Settings** button is pressed, the process dialog is 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.

Process command with a new file

The **Process** command behaves slightly differently when the current window is a memory view derived from a sampling document. The command activates a modified version of the **Process** dialog. This 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 or Fast fixed int 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.

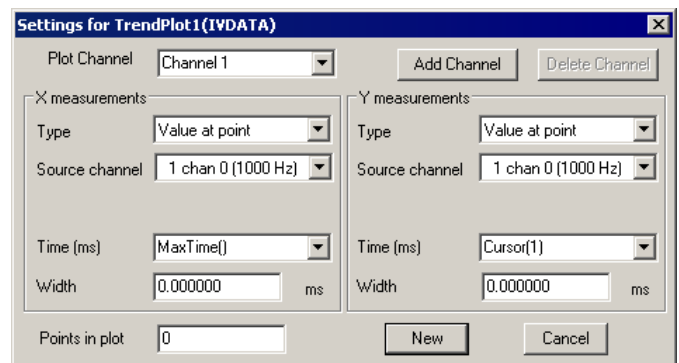
Process settings...

This menu command 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 and rebuilds and clears the memory view.

New XY View This command, analogous to **New Memory View**, is available when a file or memory view is selected. It provides a pop-up menu from which you can select an analysis type, currently **Trend plot** analysis is the only one available.

Trend plot Trend plots consist of sets of measurements taken from a source data document and plotted into an XY view. Two measurements are taken from each frame; one measurement generates the X part of the XY data, the other generates the Y data. Selecting **Trend plot** analysis leads into a **Settings** dialog where you define the analysis parameters and other information to construct the XY view. This dialog is also available later on to change the analysis and view parameters. Once you have set the required values in the **Settings** dialog the new XY view is created and the standard **Process** dialogs can be used to control the analysis.

An XY view holds lists of XY points, one list per channel. The trend plot analysis can create data points using a wide variety of types of measurement for both the X and Y values. Up to 32 channels of XY data can be created. When the command is used the **Trend plot** settings dialog is provided to allow you to enter the trend plot settings.



The dialog consists of four regions. At the top is a control to select a channel and buttons to add and delete channels, you cannot delete the last channel. In the middle are two similar regions setting the parameters for the X and Y measurements, and at the bottom are the standard buttons plus a control for the points per channel.

You can set the channel title in the **Plot Channel** box and create additional channels with the **Add Channel** button. The **Delete Channel** button deletes the current channel; you cannot delete the last channel.

The two measurements sections are the same. Both hold a selector for the channel to take measurements from, another for the type of measurement, items for the one or two time values needed for the measurements and a measurement **Width** item. A non-zero **Width** will cause a measurement to be taken by averaging the readings from $-Width/2$ to $+Width/2$ of the specified time. The types of measurement available are:

Value at point	the channel value at the time specified using the specified width.
Value difference	the difference between 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.
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.
Fit coefficients	the resulting coefficients from a fit . The coefficient index should be specified.
User entered value	a value entered by the user. A dialog opens to read the value.
Cursor regions	any measurement available in the cursor regions window can be used (see the <i>Cursor menu</i> chapter).
Value ratio	the product of the channel value at the time specified and the value at the reference time, both using the specified width.
Value product	the ratio of the channel value at the time specified and the value at the reference time, both using the specified width.
Value above baseline	the difference between the channel value at the time specified and the value at the reference time. Only the reference time uses the specified width.

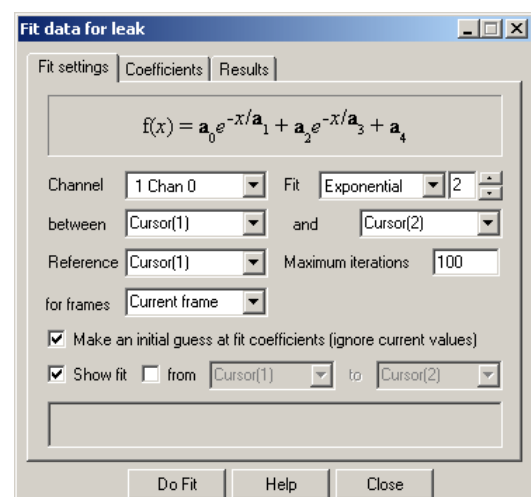
The channel selector and time entry fields are as standard for Signal, and should be easy to use. Don't forget that, in addition to entering a time value directly or selecting an item such as "Cursor(2)" or "XLow()", you can apply an offset to these selected value allowing you to enter "Cursor(2) - 0.1" or "XLow() + 1".

The Points in plot field at the bottom allows you to specify the number of points this channel can contain before old points are deleted to make way for new. Set this field to zero to allow all points to accumulate.

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 an extra checkbox for view X axis optimisation after processing.

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.

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 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.

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.

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.

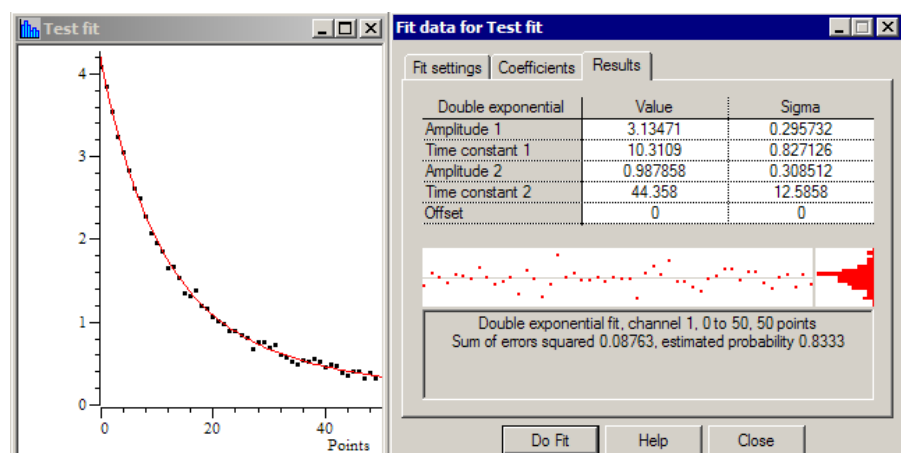
a	Value	Hold	Lower limit	Upper limit
0	1.7251128	<input type="checkbox"/>	-1000000	1000000
1	0.00020014878	<input type="checkbox"/>	0	1000000
2	0.84136252	<input type="checkbox"/>	-1000000	1000000
3	0.0063516657	<input type="checkbox"/>	0	1000000
4	-0.32806755	<input type="checkbox"/>	-1000000	1000000

Estimate values Clear fit

Exponential fit from 0.00844749 to 0.0144894, 30 points, 28 iterations

Do Fit Help Close

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.



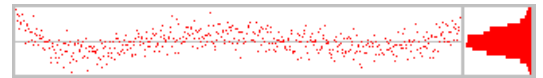
Coefficient values The **Value** column holds the fitted value that minimised the chi-squared or sum of squares error for the fit. The **Sigma** column is an estimate of how the errors between the fitted curve and the original data translates into uncertainty in the fit coefficients given that the model 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.

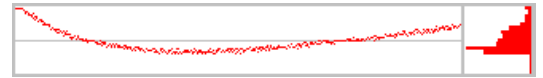
In the case that the data can be modelled by the fitting function plus normally-distributed noise, you would expect to see residuals distributed randomly around the 0 error line and the histogram on the right should resemble a normal curve.



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 of 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.

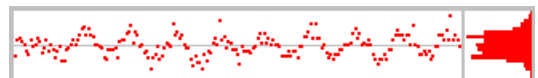
Testing the fit When you fit a model to measured data to obtain the best-fit coefficients, there are two questions you would like answered:

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.

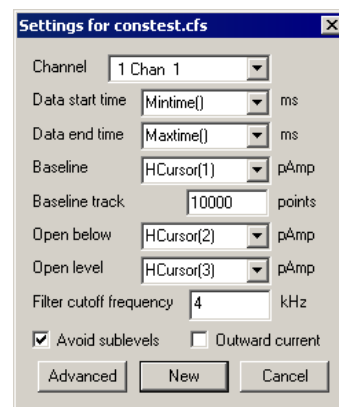
Open/Closed times

This command, analogous to **New Memory View**, is available when a file or memory view is selected. It provides a pop-up menu from which you can select an analysis type. The analysis types are intended for use by patch clampers doing single channel experiments. Currently **New idealised trace (SCAN)**, **New idealised trace (Threshold)**, **Open/Closed time histogram**, **Open/Closed amplitude histogram** and **Burst duration histogram** analyses are available. Selecting these leads to a settings dialog.

An idealised trace needs to be fitted to a data trace before histograms can be built. If you set up histograms without an idealised trace fitted they will still be created but will not contain any data. If you are working on-line, it is worth noting that the idealised trace will be generated first so that other analyses that depend on it will still work. Once an idealised trace exists, it is stored in the `.sgr` file associated with the data file and re-loaded each time the data file is opened.

New idealised trace (SCAN)

An idealised trace is one in which the raw data trace is converted into a set of events. Each event has a start time, amplitude, duration and a set of flags to identify the type of event: a closed time; first latency etc. The settings dialog holds fields to determine how this trace should be generated. The SCAN method of generating such a trace 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 with the step response function for the amplifier. The default draw mode for the resulting trace is to show the convolution over the top of the raw data with the idealised trace drawn below. 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. 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 for the results.



The **Channel** is the waveform channel to be fitted. The time range to be fitted is then defined by the **Data start time** and **Data end time**. 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.

The **Baseline** is used at the start of the analysis to tell the event detection where the closed state is expected to fall. **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.

New will take you to the standard process dialog described above but will not generate a result view. The results are drawn instead in the file view on top of the original waveform.

Advanced 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 immediately 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 opposite 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.

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	4
Minimum amplitude change of a transition	5 %
Sensitivity for multiple transitions vs sublevel	3
Percentage of filter length to jump after a transition	60 %
Free amplitudes longer than this multiple of rise time	3
<input type="button" value="Restore Defaults"/> <input type="button" value="OK"/> <input type="button" value="Cancel"/>	

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)

Threshold crossing as a method for generating an idealised trace is needed if you have more than one channel in your patch or you want a simpler technique and are not concerned with a very high time resolution of the result.

The first three fields are exactly the same as for the **SCAN** method. 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.

Transition points can be set to provide a “dead time” after a transition before another can be detected. This allows for settling of data which has been filtered.

The **Base level** is used in multiple level analysis. The thresholds for subsequent levels are calculated by first averaging the two thresholds 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.

By default an event will start at the first data point found to be across a threshold and have an amplitude which is the average amplitude of all the data point within the event period. The **Interpolation** field can be used so that each transition has a start time calculated by assuming that the transition occurred between sample points at a time calculated by extrapolating between the points. Currently only linear interpolation is available. This assumes a straight line represents the data between sample points.

Select **Multiple level** if you have more than one channel in your patch. **Outward current** means that a channel opening produces a more positive current. This is a convention. An inward current would produce a more negative current when a channel opens.

New will take you to the standard process dialog described above but will not generate a result view. The results are drawn instead in the file view on top of the original waveform.

Open/Closed time histogram

This opens a settings dialog, which defines the parameters for a histogram of event durations. The **Channel** field selects a waveform channel from the source view. This channel must have an idealised trace fitted in order for the histogram to be built. The **x-axis** of the result starts at zero, so the **Bin width** and **Maximum duration** together define the number of bins in the histogram.

The two regions labelled **Include** and **Exclude** represent the flags associated with each event. An event is included in the histogram if at least one flag of the event matches the **Include** set and none of the flags of the event match the **Exclude** set. The flags are:

Open time	A period when the channel is open.
Closed time	A period when the channel is closed.
First latency	The first event in the idealised trace.
Truncated	The last event in the idealised trace.
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.

The dialog box 'Settings for OpClHist1(example)' contains the following fields and checkboxes:

- Channel:** 201 ADC 0 (Idealised)
- Bin width:** 0.001 s
- Maximum duration:** 0.1 s
- Include section:**
 - ☒ Open time
 - ☐ Closed time
 - ☐ First latency
 - ☐ Truncated
 - ☐ Assumed amp.
 - ☐ Bad data
- Exclude section:**
 - ☐ Open time
 - ☐ Closed time
 - ☒ First latency
 - ☒ Truncated
 - ☐ Assumed amp.
 - ☒ Bad data
- Buttons:** New, Cancel

Open/Closed amplitude histogram

The open/closed amplitude histogram is almost identical to the amplitude histogram described already. The differences are that the **y-axis** will be a count of events rather than time spent at that amplitude and the ability to include and exclude events using the **Include** and **Exclude** flags as with the Open/Closed time histogram.

The **Maximum amplitude** and **Minimum amplitude** fields set the amplitude range that is divided into bins and that sets the **x axis** range in the result.

The dialog box 'Settings for OpClAmp1(example)' contains the following fields and checkboxes:

- Channel:** 201 ADC 0 (Idealised)
- Maximum amplitude:** YHigh() V
- Minimum amplitude:** YLow() V
- Bin size:** 0.022 V
- Number of bins:** 200
- Include section:**
 - ☒ Open time
 - ☐ Closed time
 - ☐ First latency
 - ☐ Truncated
 - ☐ Assumed amp.
 - ☐ Bad data
- Exclude section:**
 - ☐ Open time
 - ☒ Closed time
 - ☒ First latency
 - ☒ Truncated
 - ☐ Assumed amp.
 - ☒ Bad data
- Buttons:** New, Cancel

Burst duration histogram

This opens a settings dialog, which defines the parameters for a histogram of event burst durations. The **Channel** field selects a waveform channel from the source view. This channel must have an idealised trace fitted in order for the histogram to be built. The **x-axis** of the result starts at zero, so the **Bin width** and **Maximum duration** together define the number of bins in the histogram.

The two regions labelled **Include** and **Exclude** represent the flags associated with each event. A burst duration begins with the start time of an event included (using the rules described for the Open/Closed time histogram) and ends at the start time of an excluded event having a duration longer than the critical interval.

The dialog box 'Settings for Burst1(constest)' contains the following fields and checkboxes:

- Channel:** 201 Chan 1 (Idealised)
- Bin width:** 0.1 ms
- Maximum duration:** 10.0 ms
- Critical interval:** 10.0 ms
- Include section:**
 - ☒ Open time
 - ☐ Closed time
 - ☐ First latency
 - ☐ Truncated
 - ☐ Assumed amp.
 - ☐ Bad data
- Exclude section:**
 - ☐ Open time
 - ☒ Closed time
 - ☒ First latency
 - ☒ Truncated
 - ☐ Assumed amp.
 - ☒ Bad data
- Buttons:** New, Cancel

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.

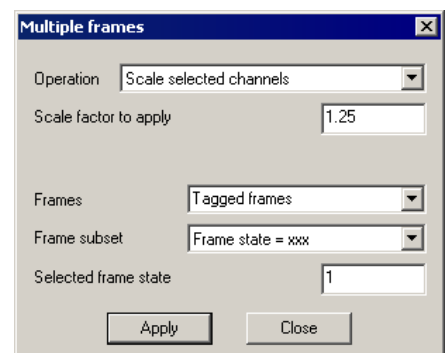
Add to buffer This command (keyboard shortcut `+`) adds the data in the current frame to the data in the frame buffer. There is an alternative form of this command, available only through the `Ctrl++` shortcut and the **Multiple frames** dialog, which adds the buffer data to the data in the current frame.

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.



The operations available from the dialog include all of the frame buffer operations, all of the channel data modification options, plus tag and untag frames. 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. 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 opens the Digital filtering dialog, which can create FIR filters and apply them to waveform channels.

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	Clear buffer data	Ctrl 0
Rectify data	Shift+R	Add frame to buffer (average)	Enter
Subtract DC level	Shift+O	Add frame to buffer	+
Differentiate data	Shift+D	Add buffer data to frame	Ctrl++
Integrate data	Shift+I	Subtract frame from buffer (average)	Ctrl+Enter
3-point smooth	Shift+3	Subtract buffer data from frame	-
5-point smooth	Shift+5	Subtract frame from buffer	Ctrl+-
Shift 1 point left	Shift+<	Copy frame data to buffer	Insert
Shift 1 point right	Shift+>	Copy buffer data to frame	Ctrl+Ins
		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 manipulation; see the *View menu* chapter.

12

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. 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.

New Cursor



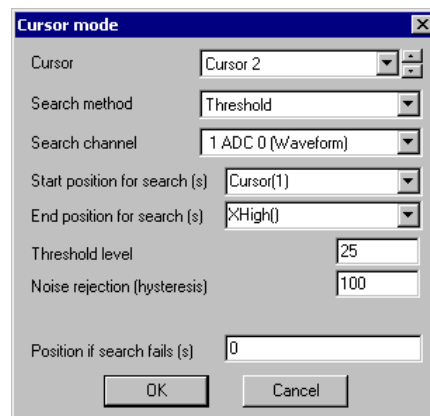
This menu command (keyboard shortcut `Ctrl+I`) duplicates the action of the new cursor button at the bottom left of data views. The command is available when a data view is the current window and there are less than ten cursors already active in the view. A new vertical cursor is added at the centre of the window. The cursor is given the lowest available cursor number and is labelled with the cursor label style for the window.

Cursor mode

This menu command opens a dialog from which you can select a cursor and its search mode. The command is available when a data view is the current window and there are cursors present in the view.

Normally, Signal cursors are static; they stay where they are put. Using the cursor mode dialog, a cursor can be made active; it will move to the position of a data feature, if it can be found. This search-and-move occurs when the view changes to a different frame or when the current frame data is changed.

The cursor mode dialog allows you to define the feature searched for, the channel to be searched, the limits to the search and to set parameters defining the feature being searched for. The **Cursor mode** item at the top selects the feature to be searched for or defines the cursor as static.



The **Active channel** field sets the channel and the start and end times set the time range to search. The start or end time for a search can be a cursor position, which itself could be active and changing. If the start time is later than the end time then Signal will search backwards for the feature. At the bottom of the dialog are up to three parameters (**Noise rejection (hysteresis)**, **Threshold level** and **Width for slope**) used to define the feature more precisely, one or more of these may be present depending upon the cursor mode selected. The possible cursor modes are:

Static

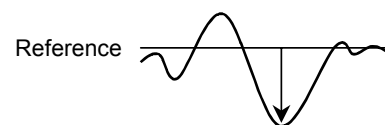
When you add a new cursor, it starts in **Static** mode. In this state, the cursor stays where you put it; it is not changed by a change in the position of a lower numbered cursor.

Maximum and Minimum

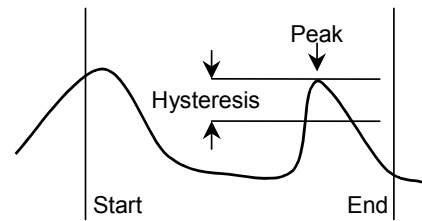
The result is the position of the maximum or minimum channel value in the search range.

Maximum excursion

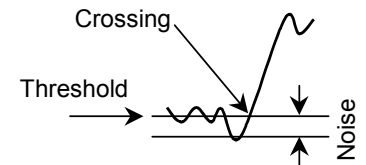
There is an extra field in this mode for the **Reference level**. The cursor is positioned at the point that is the maximum distance in the y direction away from the reference level.



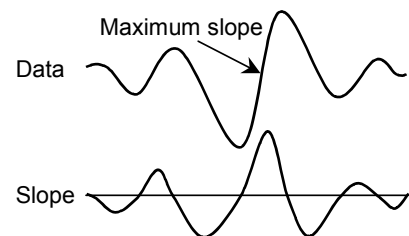
Peak, Trough The Noise rejection (hysteresis) field sets by how much the data must rise before a peak and fall after it (or fall before a trough and rise after it), to be accepted as a peak. In the diagram of a peak search, the first peak is not detected because the data did not rise by Hysteresis within the time range.



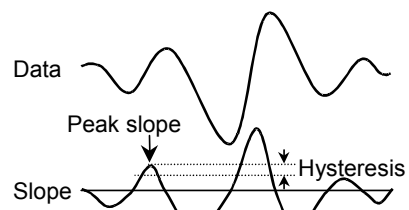
Rising threshold, Falling threshold, Threshold These modes have two new fields: Threshold and Noise rejection (hysteresis). The data must cross Threshold from a level more than Noise rejection away from it. For a Rising threshold, the data must rise through the threshold, for a Falling threshold it must fall. In Threshold mode the crossing can be in either direction. The picture shows a rising threshold.



Steepest rising, Steepest falling, Steepest slope (+/-) The Width for slope measurement field sets the length of data used to evaluate the slope at each data point. The result is the position of the maximum, minimum or maximum absolute value of the slope.

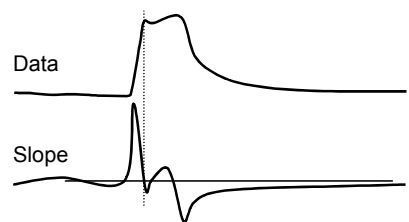


Slope threshold, +ve slope threshold, -ve slope threshold The result is the position at which the slope crosses a particular threshold. The units for the threshold are y axis units per second

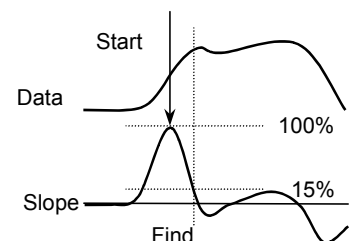


Slope peak, Slope trough These modes calculate the slope of the data in the search range, and then find the first peak or trough in the result that meets the Hysteresis specification. The Width for slope measurement field sets the length of data used to evaluate the slope at each data point. The Hysteresis 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 Hysteresis 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

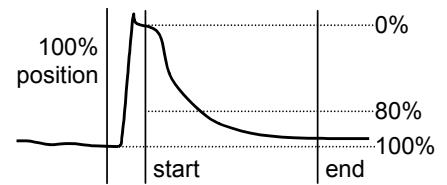


Slope% This method finds the start and end of a fast up or down stroke in a waveform. The Width for slope measurement field sets the time width used to calculate the slope. The Slope% field sets the percentage of the slope at the search start to find. To use this mode, set a cursor at the peak slope, then use it as the start point and search for the required percentage. 15% usually works well.



Repolarisation %

This mode finds the point at which a waveform returns a given percentage of the distance to a baseline inside the search range. The search start position defines the 0% repolarisation level. The 100% position and Width fields identify the 100% level (this can lie outside the search range). Repolarise % (drawn at 80% in the picture) sets a threshold relative to the 0% and 100% levels. The position is the first point in the search range to cross it.

**Expression**

The cursor position is obtained by evaluating the Position field. This field will normally hold an expression based on cursor positions, for example "Cursor (1) + 2.5".

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

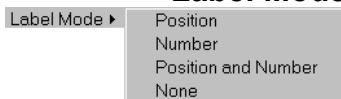
This opens a pop-up menu where you select a cursor to place in the centre of the x axis.

Move To

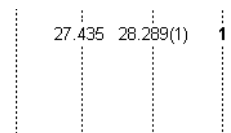
This command activates a pop-up menu from which you can select the cursor to move to. The cursors are listed with their number and position as an aid to identification. The window scrolls to show the cursor in the screen centre, or as close to it as possible.

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

The four cursor labelling styles are: **None**, **Position**, **Position and Number**, and **Number**. 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 to suit the data (see the *Getting started* chapter).

**Renumber**

This command rennumbers vertical cursors by position, with cursor 1 on the left.

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.

Display All Horizontal This command is the equivalent of using the **Fetch Horizontal** command for all cursors.

Horizontal Label Mode There are four labelling styles for horizontal cursors, as for vertical cursors, use this command to select the most appropriate mode for your purposes. 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 along the cursor to suit the data (see the *Getting started* chapter).

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 top.

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.

The values displayed depend upon the channel type and display mode.

There is an entry in the table for each channel displayed. The displayed values are as follows:

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 ☐ ☐ ☐ ☐

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 Time 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



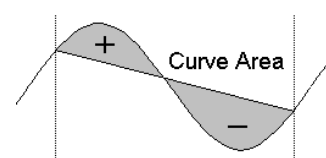
This command opens the Cursor regions dialog for the current data view. The dialog displays values for data regions between cursor pairs. One pair can be designated the Zero region by checking the box and selecting the column with a radio button. The value in this column is then subtracted from the values 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	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			
<input type="checkbox"/> Zero region <div> <input type="radio"/> Mean </div>			

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:

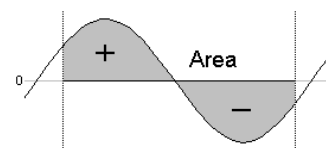
Curve area Each data point makes a contribution to the area of its amplitude above a line joining the endpoints multiplied by the x axis distance between the data points. The picture makes this clearer.



Mean The mean value of all the waveform points in the region. If there are no samples between the cursors the field is blank.

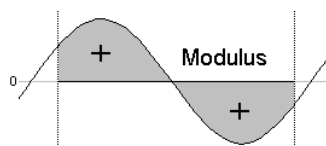
Slope The slope of the least squares best fit line to waveform points in the region.

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 zero to be treated as positive.



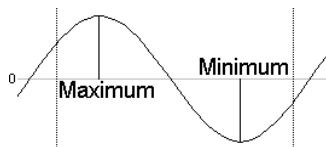
Sum The sum of all the waveform points in the region. If there are no samples between the cursors the field is blank.

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.

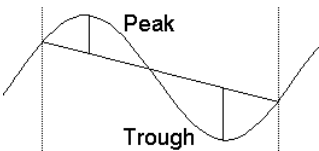


Amplitude The value shown is the difference between maximum and minimum values found between the cursors.

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.

RMS 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.

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.	

The measurements available for marker type channels are **Mean**, **Sum**, **Maximum**, **Minimum**, **Amplitude** and **Extreme**. 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.
Extreme	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:

Copy	The position of the selected cursor is copied to the clipboard.
Active mode...	Puts up the active cursor mode dialog for the selected cursor.
Delete	Deletes the cursor.

For horizontal cursors the list of commands are as follows:

Copy Position=X	Copies the position (X) of the selected cursor to the clipboard.
Copy Position-HCursor(n)=Y	Subtract the position of horizontal cursor n from the selected cursor position and copy the result (Y) to the clipboard.
Delete	Deletes the selected cursor.

13

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.

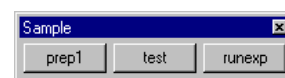
Sampling configuration



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.

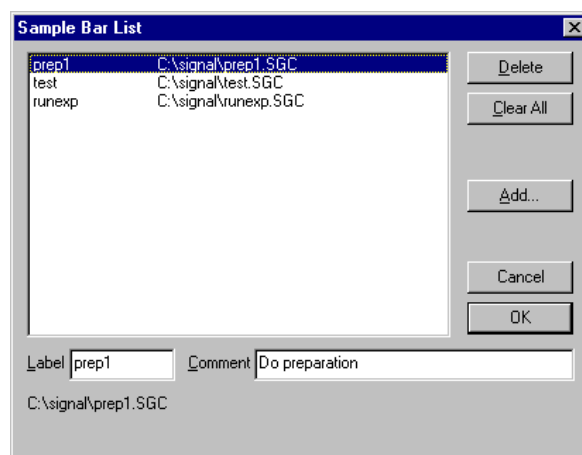


Each button is linked to a **Signal** configuration file. When you click a button, the associated configuration file is loaded and a new data file is opened, ready for sampling. You can also show and hide the **Sample Bar** by clicking the right mouse button on any **Signal** toolbar or on the **Signal** background.

Sample Bar List

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 opens the conditioner dialog (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.

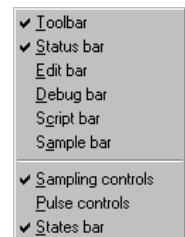
Restart sampling This command discards all data, returns sampling to the state it was in before sampling started and restarts sampling. It is the same as the sampling control panel **Restart** button.

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.

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

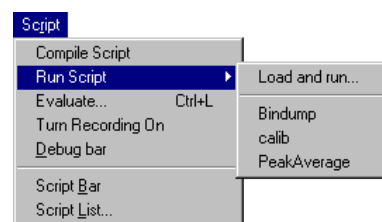


This command is enabled when the current view holds a script. It is equivalent to the **Compile** button in the script window. Signal checks the syntax of the script, and if it is correct, it generates a compiled version of the script, ready to run.

Run Script



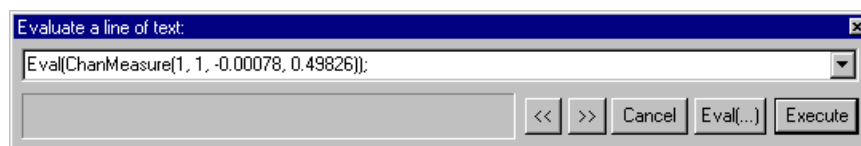
This command pops-up a list of all the scripts that have been loaded so that you can select a script to run. Signal compiles the selected script and if there are no errors, runs the script. If you run a script twice in succession, Signal only compiles it for the first run, saving the compilation time. 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 to include an **Eval()** on the last statement, so you can see the result. You can execute any script that can be typed in one line, which can include variable declarations.



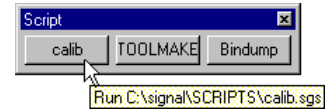
Turn Recording On/Off

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.

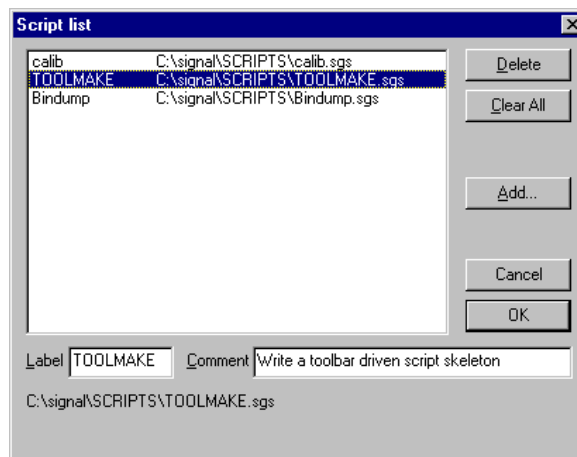
You can use this mechanism to record a sequence of actions that you wish to rerun at some later date, to find out what script commands correspond to a given menu command or user action or to record a sequence of actions that can be copied into another script or edited to produce a complete scripted 'application'.

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



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.

15

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.

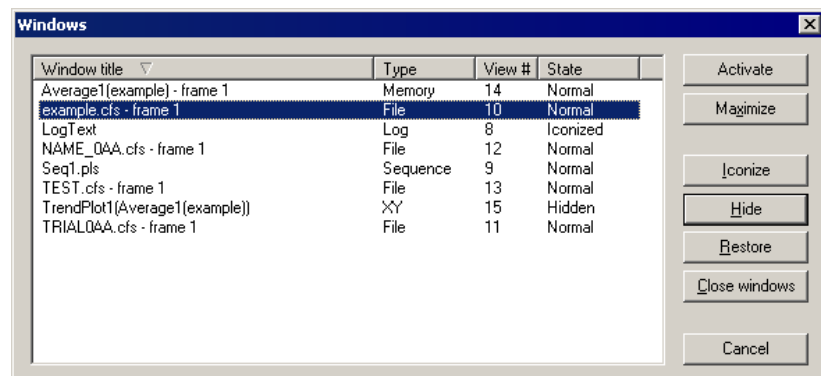
Tile Vertically You can arrange all the visible Signal windows so that they are arranged in a vertically tiled pattern by using this command. Vertical tiling arranges the windows so that they tend to be tall and thin, again the exact arrangement depends upon the number of windows.

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.



16

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.

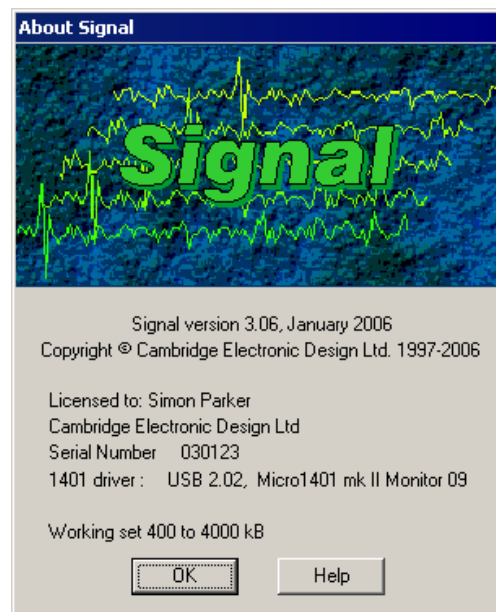
From a script view or the script evaluate dialog you can obtain help by placing the cursor on any keyword in the script and pressing F1. To get help on a script function, type the function name followed by a left hand bracket, for example `MemChan (`, then make sure that the cursor lies to the left of 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.

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 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 Spike2 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 downloadable 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:

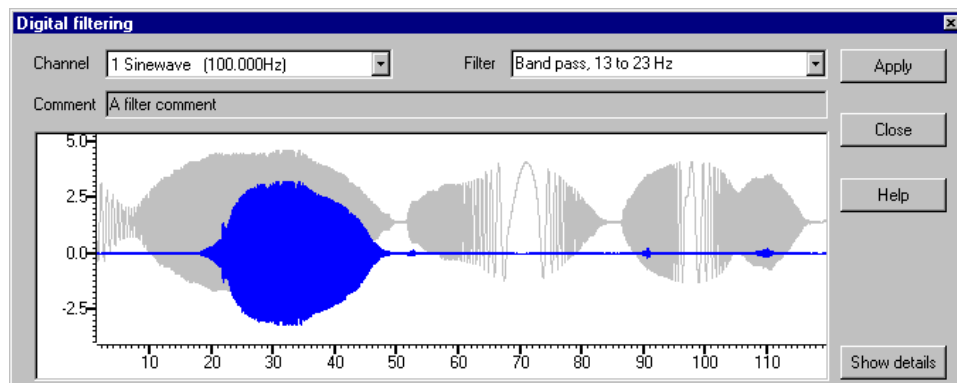
1. Read about the topic in the manual. Use the Index to search for keywords related to the topic.
2. Try the help system for more information. Use the Search facility to find related topics.
3. If none of the above helps, FAX, email or call the CED Software Help Desk (numbers and addresses are to be 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.

17

Digital filtering

Introduction

The Analysis menu **Digital filters...** command is available when you have a data file open. You can apply one of twelve stored digital filters to a set of waveform channels over a selection of frames, though you can only preview one channel in the current frame at a time. You can also create your own digital filter. The program implements FIR filters (Finite Impulse Response) optimised to minimise the filter ripple in each filter band (see page 17-5 for more technical information on the filters).

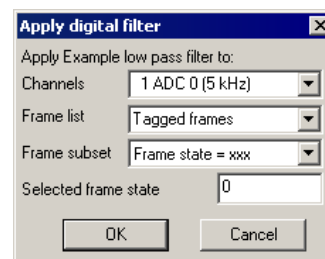


The **Filter** field of the dialog box selects the filter to apply and the **Channel** field sets the waveform channel to preview. The **Comment** field is for any purpose you wish; there is one comment per filter. The dialog shows the original waveform in grey, and a filtered version in the waveform colour.

The **Close** button shuts the dialog and will ask if you want to save any changed filter and the **Help** button opens the on-line Help at the digital filtering topic.

Apply

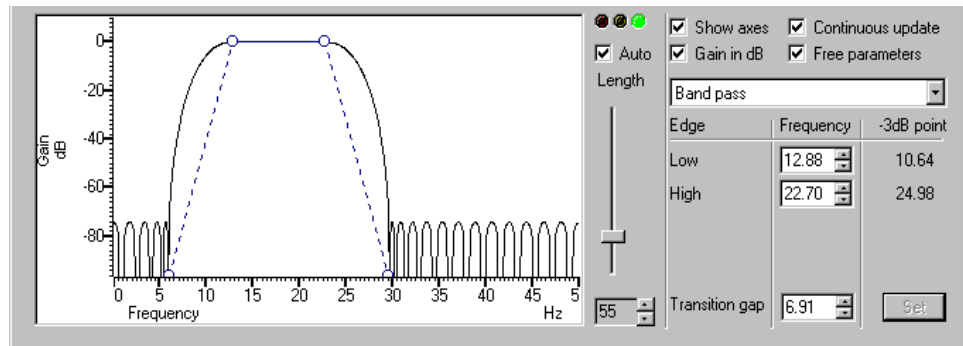
The **Apply** button opens a new dialog in which you set which channels and frames to filter with the **Channels** and **Frame list** fields. The **Frame subset** field can be used to further specify which frames are to be filtered. The **Selected frame state** field only appears if you select **Frame 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.



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.

As applying a filter can be a lengthy process, a progress dialog appears with a **Cancel** button during the filtering operation.

Show details The **Show details** button increases the dialog size to display a new area in which you can design and edit filters. Click this button again to hide the new dialog area.



The picture in the new area shows the frequency response of the filter. The **Gain in dB** check box sets the y axis scale to dB if checked, linear if not checked. The **Show axes** check box controls the axes of the raw and filtered data display. The frequency response display shows the ideal filter as solid lines for each defined band linked by dotted lines which mark each **Transition gap** between the bands. All transition gaps have the same width. The calculated frequency response is drawn as solid lines and is greyed when the filter specification has been changed and the response has not been calculated to match.

The circles can be dragged sideways to make the edges of the bands steeper or less steep or you can edit the band edges as numbers in the **Frequency** panel on the right. You can also drag the bands sideways and the band gaps. The mouse pointer changes to an appropriate symbol to indicate the feature you are dragging.

If you edit the numbers in the **Frequency** field, the **Set** button is enabled so you can force a recalculation of the filter.

The filters produced by the program are not defined in terms of -3dB corner frequencies and n dB per octave as is often the case for traditional analogue filters. The **-3dB point** column is present to help users who are more comfortable describing filter band edges in terms of the 3 dB point.

If you check the **Continuous update** box, the filter is updated while you drag the filter features around. If you have a slow computer and this feels ponderous you can clear the check box, in which case the filter is not recalculated until you stop changing features.

If you check the **Free parameters** box, dragged features are not limited by the next band and will push bands along horizontally. If you clear the box, the horizontal motion of a dragged feature is limited by the next moveable object.

To the right of the frequency response display is a slider that controls the number of filter coefficients. In general, the more coefficients, the better the filter. However, the more coefficients, the longer it takes to compute them and the longer to filter the data. If you check the **Auto** box, the program will adjust the number of coefficients for you to produce a useful filter. The “traffic light” display above the slider shows green if the filter is good, amber if the result is usable but not ideal, and red if the result is hopeless.

If you change a filter or create a new filter, you will be prompted to save the filter bank when you close the digital filter dialog.

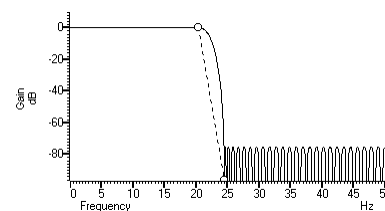
Filter bank A digital filter definition is complex and it would be tedious to specify all the properties of a filter each time you wanted to apply one to data. To avoid this, Signal contains a filter bank of 12 filter definitions. This filter bank is saved to the file `Filtbank.cfb` when you close Signal 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 an index number in the range 0 to 11.

Filter types The type of the filter is set by the drop down list to the right of the display. If you need a filter that is not in this list you can generate it from the script language. Users of the `FIRMake()` script language command should note that the bands referred to here are pass bands. In the script language there are additional stop bands between the pass bands. There are currently 12 different filter types:

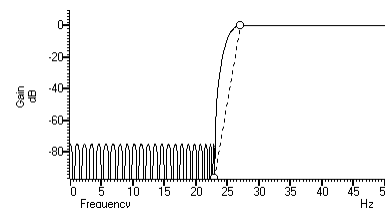
All pass This has no effect on your signal. This filter type covers the case where you apply a low pass filter designed for a higher sampling rate to a waveform with a much lower sampling rate, so that the pass band extends beyond half the sampling frequency of the new file.

All stop This removes any signal; the output is always zero. This filter type is provided to cover the case where you apply a high pass filter designed for a higher sampling rate to a waveform with a much lower sampling rate, so that the stop band extends beyond half the sampling frequency of the new file.

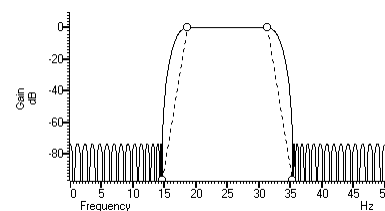
Low pass This filter attempts to remove the high frequencies from the input signal. The **Frequency** field holds one editable number, **Low pass**, the frequency of the upper edge of the pass band. The stop band starts at this frequency plus the value set by the **Transition gap** field.



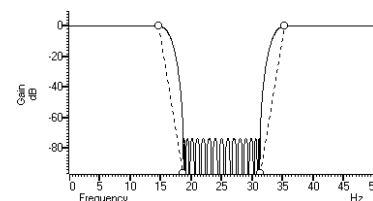
High pass A high pass filter removes low frequencies from the input signal. The **Frequency** field holds one editable number, **High pass**, the frequency of the lower edge of the pass band. The stop band starts at this frequency less the value set by the **Transition gap** field.



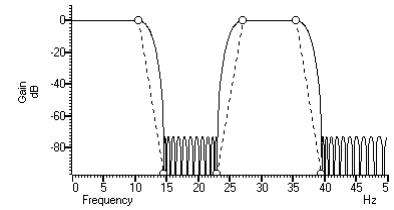
Band pass A band pass filter passes a range of frequencies and removes frequencies above and below this range. The **Frequency** field has two editable numbers, **Low** and **High**, which correspond to the two edges of the pass band. The stop band below runs up to **Low-Transition gap**, and the stop band above from **High+Transition gap** to one half the sampling rate.



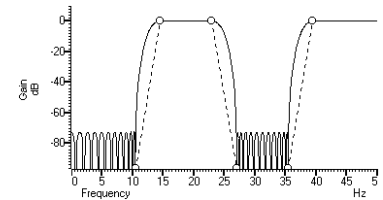
Band stop A band stop filter removes a range of frequencies. The **Frequency** field has two editable numbers, **High** (the upper edge of the first pass band) and **Low** (the lower edge of the upper pass band). The stop band below runs from **High+Transition gap** up to **Low-Transition gap**.



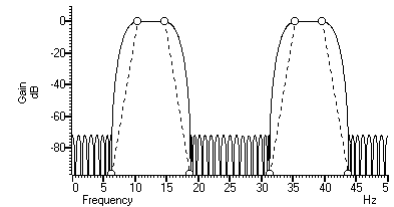
One and a half low pass This filter has two pass bands, the first running from zero Hz and the second in the frequency space between the upper edge of the first pass band and one half the sampling rate. The Frequency field has three editable numbers: Band 1 high, Band 2 low and Band 2 high. These numbers correspond to the edges of the pass bands.



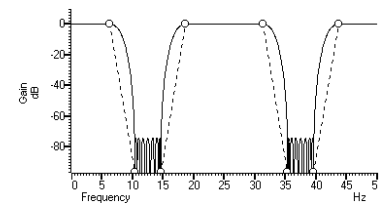
One and a half high pass This filter has two pass bands. The second runs up to one half the sampling rate. The first band lies in the frequency space between 0 Hz and the lower edge of the second band. The Frequency field has three editable numbers: Band 1 low, Band 1 high and Band 2 low. These numbers correspond to the edges of the pass bands.



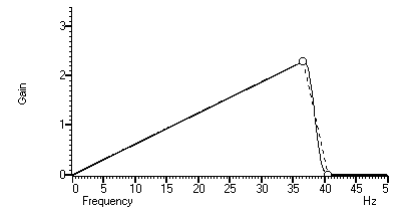
Two band pass This filter passes two frequency ranges and rejects the remainder. Both 0 Hz and one half the sampling frequency are rejected. The Frequency field has 4 numeric fields: Band 1 low, Band 1 high, Band 2 low and Band 2 high. These fields correspond to the four edges of the two bands.



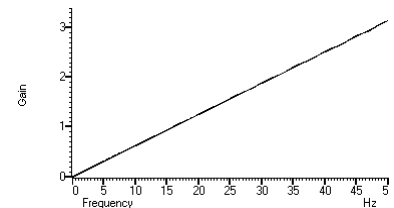
Two band stop This filter passes three frequency ranges and rejects the remainder. Both 0 Hz and one half the sampling rate are passed. The Frequency field has 4 numeric fields: Band 1 high, Band 2 low, Band 2 high and Band 3 low. These fields correspond to the four edges of the three bands.



Low pass differentiator This filter is a combination of a differentiator (that is the output is proportional to the rate of change of the input) and a low pass filter. The y axis scale is linear, rather than in dB (although you can display it in dB if you wish). There is one editable number in the Frequency field, Low pass, the end of the differential section of the filter.



Differentiator The output of the filter is proportional to the rate of change of the input. The y axis scale is linear, rather than in dB (although you can display it in dB if you wish). The Frequency field is empty as there is only one band and it extends from 0 Hz to half the sampling rate.

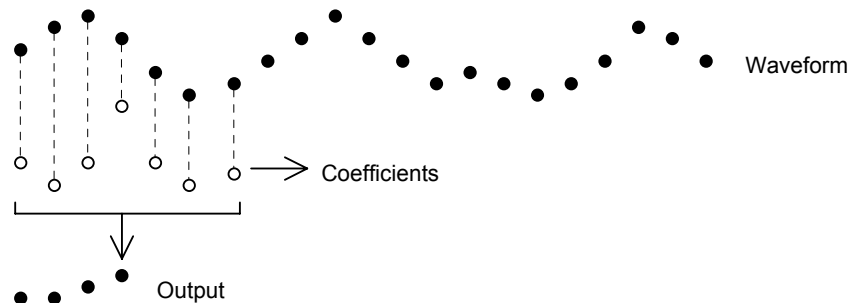


FIR filters

The `FIRMake()`, `FIRQuick()` and `FiltCalc()` script commands and the Analysis menu **Digital filters...** dialog generate FIR (Finite Impulse Response) filter coefficients suitable for a variety of filtering applications. The generated filters are optimal in the sense that they have the minimum ripple in each defined band. These filter coefficients are used to modify a sampled waveform, usually to remove unwanted frequency components. The algorithmic heart of the filter coefficient generation is based on the well-known FORTRAN program written by Jim McClellan of Rice University in 1973 that implements the *Remez exchange algorithm* to optimise the filter.

The theory of FIR filters is beyond the scope of this document. Readers who are interested in learning more about the subject should consult a suitable text book, for example *Theory and Application of Digital Signal Processing* by Rabiner and Gold published by Prentice-Hall, ISBN 0-13-914101.

FIR filtering



This diagram shows the general principle of the FIR filter. The hollow circles represent the filter coefficients, and the solid circles are the input and output waveforms. Each output point is generated by multiplying the waveform by the coefficients and summing the result. The coefficients are then moved one step to the right and the process repeats.

From this description, you can see that the filter coefficients (from right to left) are the *impulse response* of the filter. The impulse response is the output of a filter when the input signal is all zero except for one sample of unit amplitude. In the example above with 7 coefficients, there is no time shift caused by the filter. With an even number of coefficients, there is a time shift in the output of half a sample period.

Frequencies

The Analysis menu **Digital filters...** command deals with frequencies in Hz as this is comfortable for us to work with. However, if you calculate a FIR filter for one sampling rate, and apply the same coefficients to a waveform sampled at another rate, all the frequency properties of the filter are scaled by the relative sampling rates. That is, the frequency properties of an FIR filter are invariant when expressed as fractions of the sampling rate, not when expressed in Hz.

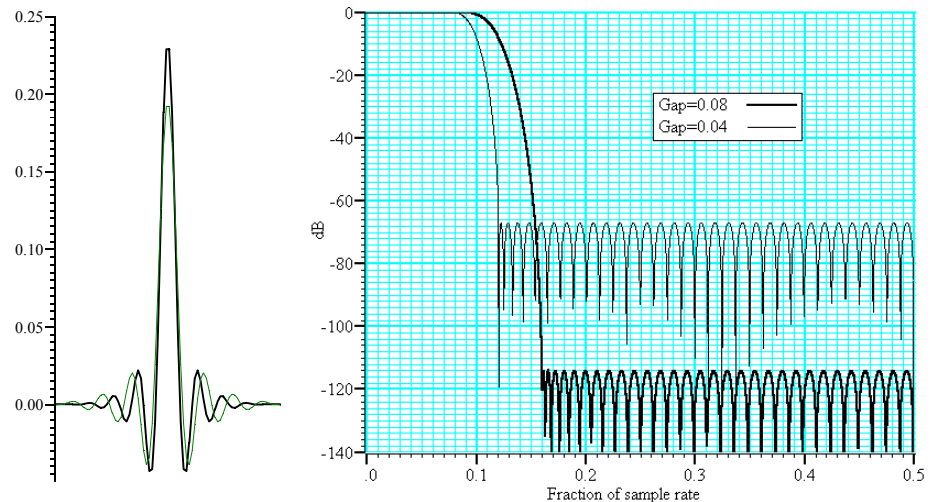
It is usually more convenient when dealing with real signals to describe filters in terms of Hz, but this means that each time a filter is applied to a waveform the sampling rate must be checked. If the rate is different from the rate for which the filter was last used, the coefficients must be recalculated. Unless you use the `FIRMake()` script command, 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



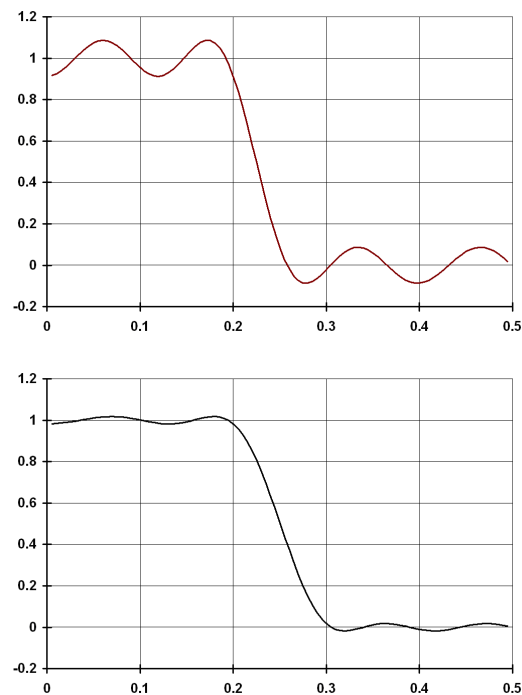
The frequency response is shown in dB which is a logarithmic scale. A ratio r is represented by $20 \log_{10}(r)$ dB. A change of 20 dB is a factor of 10 in amplitude, 6 dB is approximately a factor of 2 in amplitude. The graph shows that a frequency in the stop band is attenuated by over 110 dB (a factor 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 $-\pi/2$ in a band, often used to separate a signal from a carrier. The theory and use of this form of filter is way beyond the scope of this document. Unless you know that you need this filter type you can ignore it.

Multiband with 3dB/octave cut This is a variation on the multiband filter that can be used to filter white noise to produce band limited pink noise. The filter is identical to the band pass filter except that the attenuation increases by 3 dB per octave in the band (each doubling of frequency reduces the amplitude of the signal by a factor of the square root of 2). It is used in exactly the same way as the multiband filter.

Low pass filter example A waveform is sampled at 1 kHz and we are interested only in frequencies below 100 Hz. We would like all frequencies above 150 Hz attenuated by at least 70 dB.

A low pass filter has two bands. The first band starts at 0 and ends at 100 Hz, the second band starts at 150 Hz and ends at half the sampling rate. Translated into fractions of the sampling rate, the two bands are 0-0.1 and 0.15 to 0.5. The first band has a gain of 1, the second band has a gain of 0. We will follow our own advice and give the stop band a weight of 10 and the pass band a weight of 1. We will try 40 coefficients to start with, so a possible script is:

```
var prm[5][2];          'Array for parameters
var coef[40];           'Array for the coefficients
'   band start      band end      function      weight
prm[0][0]:=0.00; prm[1][0]:=0.1;  prm[2][0]:=1.0;  prm[3][0]:= 1.0;
prm[0][1]:=0.15; prm[1][1]:=0.5;  prm[2][1]:=0.0;  prm[3][1]:=10.0;
FIRMake(1, prm[], coef[]);
PrintLog("Pass Band ripple=%.1fdB Stop band attenuation=%.1f\n",
        prm[4][0], prm[4][1]);
```

If you run this, the log view output is:

```
Pass Band ripple=-28.8dB Stop band attenuation=-48.8
```

The attenuation in the stop band is only 48 dB, which is not enough. The ripple in the pass band is around 3% of the signal amplitude. We can increase the stop band attenuation in three ways: by increasing the number of coefficients, by giving the stop band more weight, or by making the gap larger between the bands.

We don't want to give the stop band more weight as 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.

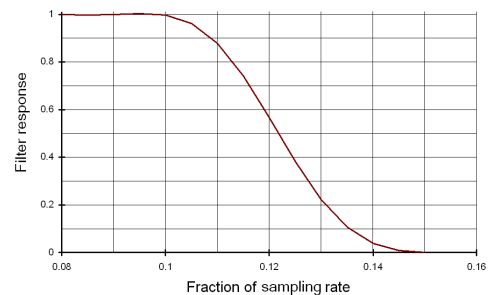
```
' dB      is the mean ripple/attenuation in dB of the bands
' deltaF is the width of the transition region between the bands
' return An estimate of the number of coefficients
Func NCoefMultiBand(dB, deltaF)
return (dB-23.9*deltaF-5.585)/(14.41*deltaF+0.0723);
end;
```

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 ($\text{dB} = 60$, $\text{deltaF} = 0.05$), 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.

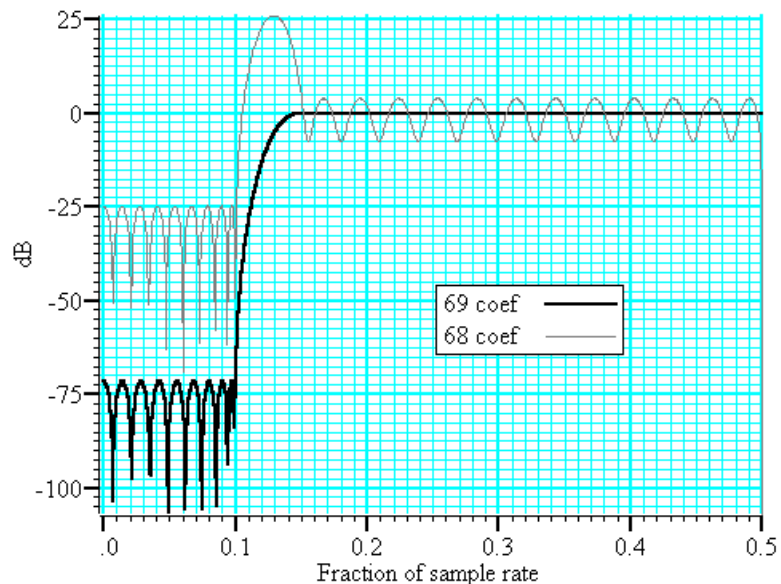


High pass filter

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[0][1], coef[0]);
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 $a, -a, a, -a, \dots$ 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

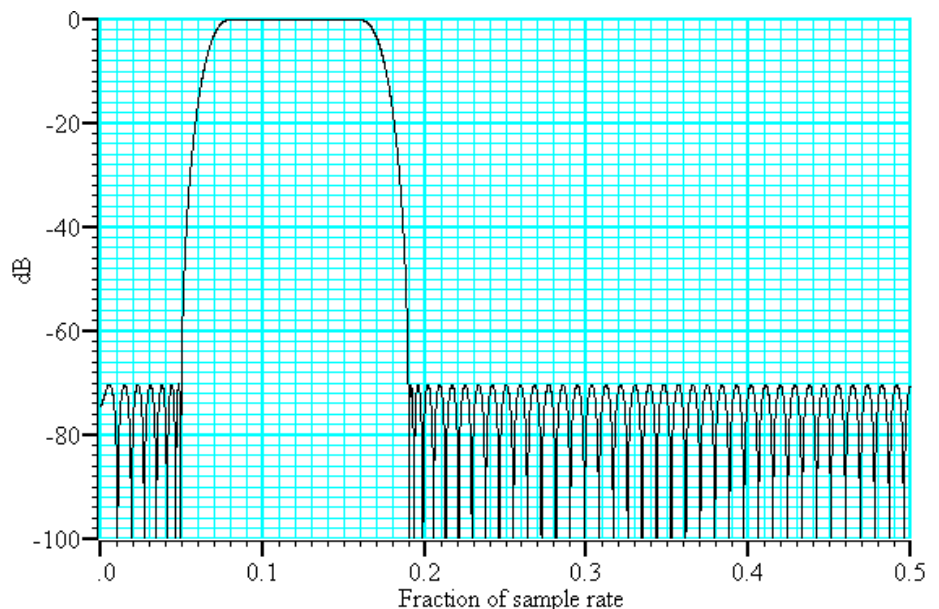
You can define up to 10 bands. However, it is unusual to need more than three. The most common cases with three bands are called band pass and band stop filters. In a band pass filter, you set a range of frequencies in which you want the signal passed unchanged, and set the frequency region below and 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[1][], coef[]);
```

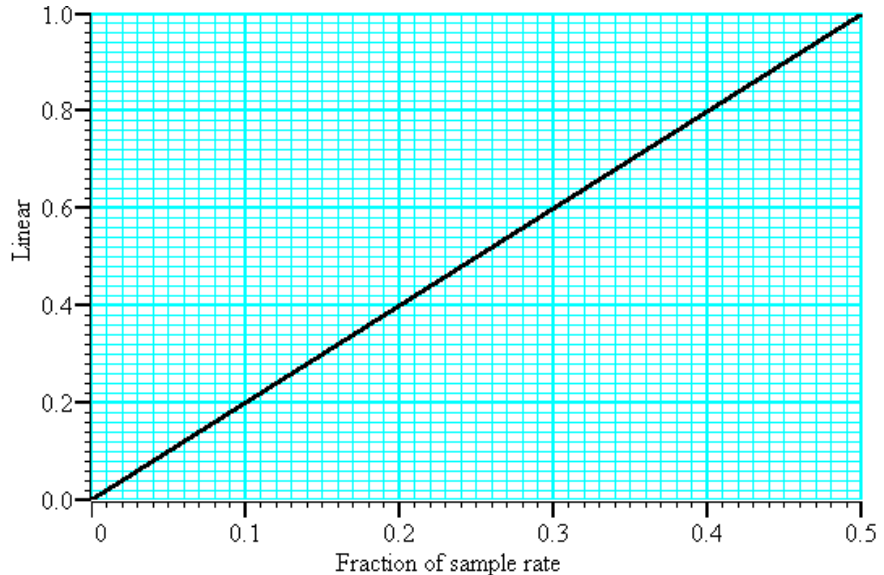
Band pass filter with 111 coefficients



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° ($\pi/2$) 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:

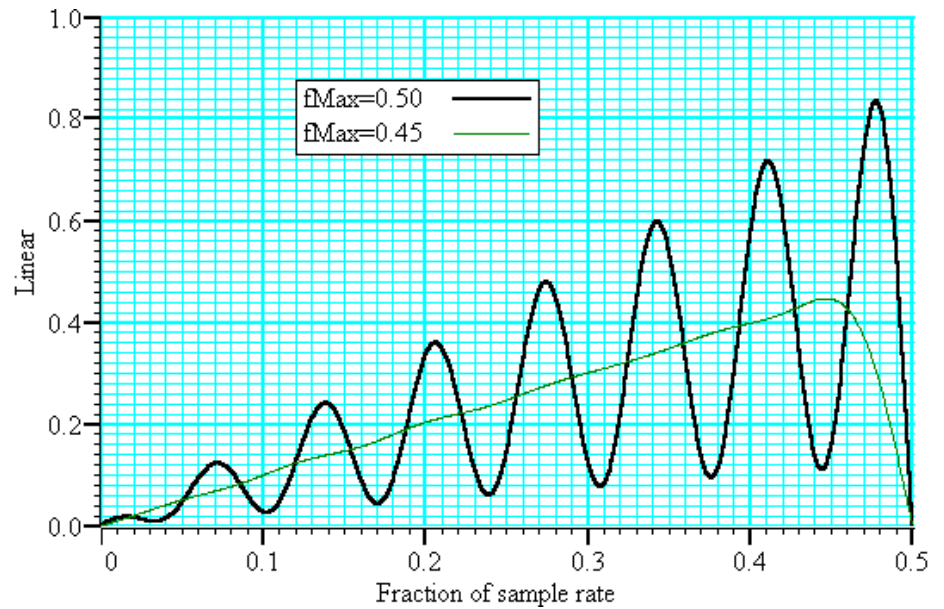
```
' dB    the proportional ripple expressed in dB
' f      the highest frequency set in the band
' even% Non-zero if you want an even number of coefficients
func NCoefDiff(dB, f, even%)
if (f<0) or (f>0.5) then return 0 endif;
f := 0.5-f;
var n%;
if (even%) then
    n% := (dB+43.837*f-35.547)/(0.22495+29.312*f);
    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.

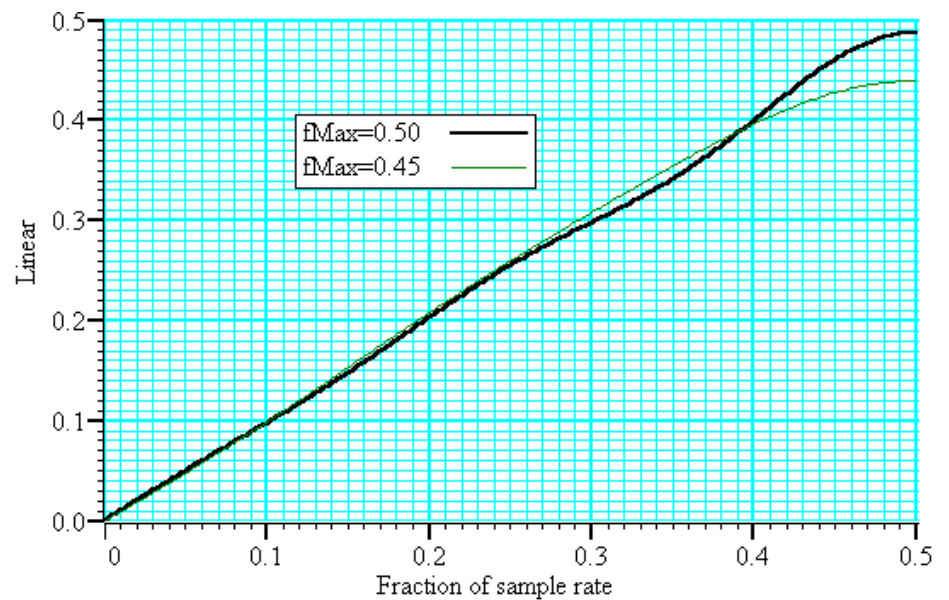
The graph below shows the effect of setting an odd number of coefficients when generating a differentiator that spans the full frequency range. The second curve shows the improvement when the maximum frequency is reduced to 0.45.

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[1][], coef[1]);
```

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 $0.5 - F_{\text{low}}$. 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 $F_{\text{high}} = 0.5 - F_{\text{low}}$. In this case, every other coefficient is 0. This is no help to the MSF and MSF programs, but users who write their own software can use this fact to minimise the number of operations required to make a filter.

It is extremely unlikely that a Hilbert transformer will be of any practical use in the context of 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.

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

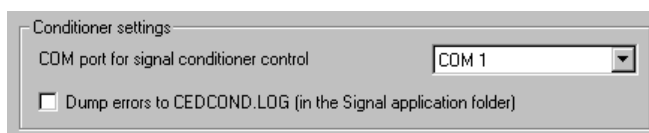
A signal conditioner takes an input signal and amplifies, shifts and filters it so that the data acquisition unit can sample it effectively. Many input signals from experimental equipment are too small, or are masked by high and or low frequency noise, or are not voltages and cannot be connected directly 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 capabilities.

Serial ports

Most signal conditioners use a serial line connection to allow software control and communication. The



serial port used to communicate with any installed signal conditioner 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 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 at least twice the highest frequency component present in the data. If you do not, the result may appear to contain signals at unexpected frequencies due to an effect called aliasing. As the highest frequency present will be above the corner frequency you should sample a channel at several times the filter corner frequency (probably between 3 and 10 times depending on the signal and the application).

You can choose a range of filter corner frequencies, or you can choose to have the data unfiltered (for use when the signal is already filtered due to the source).

High-pass filter A high-pass filter reduces low-frequency components of the input signal. 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.

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 1 The 1902 provides two conditioned trigger inputs, and one output. This control selects which of the inputs is connected to the output.

Setting the channel gain and offset If you change the gain or offset in the control panel, 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 μV .
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, signal conditioners normally start at channel zero and go up to the last channel. The conditioners are searched for assuming this arrangement; the search starts at channel 0 and continues up until a missing channel is encountered. This search process can be adjusted by two more lines that can optionally be put into `CEDCOND.INI`:

```
First=4
Last=12
```

The search for conditioners will start with the channel specified as `First`, or zero if this isn't set and continue until the channel specified as `Last` has been checked and a missing channel has been encountered. If `Last` is not specified a value of zero is used so the search stops at the first missing channel.

The arrangement is somewhat different for CyberAmps, as they have multiple channels. The channel numbers are specified by the rotary address switch at the back of the amplifier, note that the first amplifier must have the address set to zero, not 1 as normally shipped. The channels are numbered off in the order of the address switches (assuming 380s and 320s are not mixed), the `Last` value can be used to extend the search.

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, which can make the signal calibrations incorrect. Signal can avoid this problem by using amplifier telegraphs to determine the current amplifier settings and 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 the amplifier telegraphs, Signal can automatically adjust for changes in the amplifier gain settings to maintain signal calibration.

Signal currently supports two types of telegraph signal; a standard telegraph method using one or more analogue outputs 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 with Signal; this supports the Axon Instruments MultiClamp 700 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 telegraph support uses 1401 ADC inputs to sample telegraph signals, in this case analogue voltages, generated by the amplifier. A lookup table is then used to convert these voltages to the amplifier gain setting, which is used to adjust the signal calibration. Up to four separate telegraph systems can be defined, each controlling a separate ADC input (sampled signal). Setting up standard 1401 telegraphs consists of defining the ADC ports involved and setting up the table used to convert telegraph voltages to gains.

1401 telegraph configuration

The standard telegraph configuration dialog can configure up to four ports whose scaling depends upon telegraph outputs. For each of these ports, you set a port from which to read the telegraph signal and a list of signal values and the corresponding amplifier gains. The four telegraph settings are selected with the Telegraph set field.

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 field sets the ADC port to read the telegraph signal from.

Telegraph data	
-2 V signals gain of	1
-1 V signals gain of	2
0 V signals gain of	5
1 V signals gain of	10
2 V signals gain of	20
3 V signals gain of	50

The Volts and Gain controls are used to enter a list of telegraph signal voltages and corresponding amplifier gains. 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 at the bottom of the dialog is used to set the 1401 ADC range; it applies to all sets of telegraph information. Only set this checkbox if your 1401 is set up to sample voltages from -10 to +10 volts, otherwise leave it clear.

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 set in the dialog. 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 will then be applied to this 'basic' calibration. It is very important that you enter the value for unity amplifier gain, not whatever gain you 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.

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 software controlling the amplifier can generate telegraph information telling interested applications about changes to the amplifier gain settings and other relevant information such as filter settings. Using the MultiClamp 700 telegraph configuration dialog you can configure Signal software so that it can receive and make use of this telegraph information and control how you want the information to be used.

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, we 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 of these values are retrieved for both channels, and the current values are placed in the first ten user frame variables in the sampled data file. These values can be retrieved using the script language; later versions of Signal may make use of them directly.

MultiClamp 700 telegraph configuration

The MultiClamp 700 amplifiers supplied by Axon Instruments can generate telegraph information telling interested applications about changes to the amplifier gain settings and other relevant information. 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 control how you want the link to the amplifier to be used.

At the top of the dialog there is a checkbox used to select between a MultiClamp 700A or 700B. You should set this according to the type of amplifier you will be 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 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 there is a section defining the ADC ports connected to the amplifier; so that Signal can tell which channels will be affected by the amplifier set up and gain changes. 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). You should set which ADC ports on the 1401 interface are connected to the various amplifier outputs, leave any amplifier outputs that are not being used set to 'None'.

The checkbox labelled 'Get signal names, units and calibration from amplifier setup' selects how the data channels generated from ports connected to the amplifier are set up at the start of sampling. If this checkbox is set then, at the start of sampling, the channel name and units are set using information supplied by the MultiClamp 700, depending upon what signal you have set up to be routed to the relevant

output. In addition, the channel calibration is overwritten with values supplied from the amplifier. This is the easiest way to use the MultiClamp 700 telegraph data; it is generally recommended, as it will automatically force the settings to be correct.

If this checkbox is left clear, then the channel 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 MultiClamp amplifier **gain set to unity**. It is very important that you enter the value for unity MultiClamp 700 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.

The checkbox for a 10 volt system should be set if you have a 1401 with ± 10 volt inputs, otherwise it should be left clear.

The button marked "Test" at the bottom of the dialog can be used to test that Signal can communicate successfully with the Multiclamp Commander controlling 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.

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 to access auxiliary device settings are also provided.

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 200² 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 BiStim²) and to cause the MagStim to ignore the coil interlock switch and to enable single pulse mode (for the Rapid²). This documentation only describes the controls available from within Signal; for a complete understanding of the effect these have upon the behaviour of the stimulator you should consult the MagStim user manuals.

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 Rapid² 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.

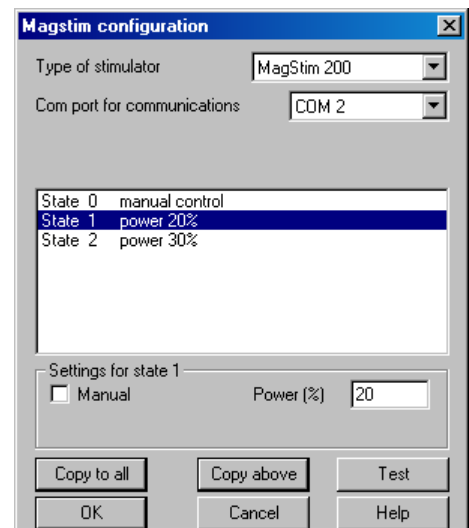
The MagStim support monitors the stimulator health; both waiting until the MagStim is ready for use before allowing a sampling sweep to proceed and terminating sampling if a hardware problem develops. To aid data analysis, Signal saves the MagStim stimulation parameters used in the data section variables of the sampled file. The parameters saved are the power level, the pulse interval (BiStim only), the secondary power level (BiStim² and dual 200²), the pulse frequency (Rapid² only) and the pulse count (Rapid² 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. Open 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, 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.

At the top of the configuration dialog there is a selector 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 this will not always be detected, though errors during sampling will almost certainly result. Below the device selection 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.



Below the basic stimulator configuration there 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.

200² 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, clear it to give Signal control.

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.

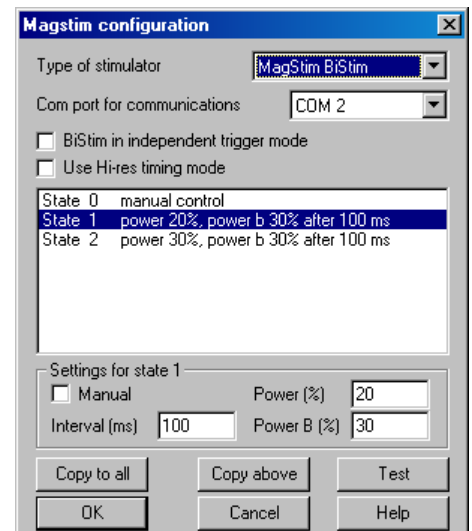
Copy... The buttons labelled **Copy to all** and **Copy above** can be used to set up many states quickly; **Copy to all** copies the currently selected state to all other states, while **Copy above** copies the current state's settings to all higher state numbers. These buttons are available for all 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.

BiStim² device The BiStim² is more complex than the 200², in essence it is two 200² units with one slaved to the other. When a BiStim² device is selected two controls governing overall behaviour are shown:

BiStim in independent trigger mode

This sets how Signal will expect the BiStim² to be configured, rather than how it will be set up by Signal. Normally, a BiStim² 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 BiStim² 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 trigger firing the main pulse and the other firing the second (Power B) pulse. Because a BiStim² behaves rather differently when in IBT mode it is important that you set this checkbox correctly to match the BiStim setup you will be using otherwise errors will be generated when you try to sample.



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 BiStim² 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 BiStim² 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, clear it to give Signal control.

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.

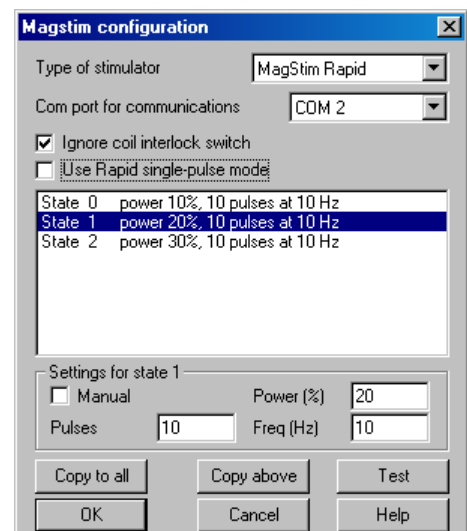
Interval This sets the interval between the two pulses, you can use any value from 0 to 999 milliseconds (or 0 to 99.9 milliseconds for Hi-res timing). Setting a value of zero for a state will switch the BiStim² into simultaneous pulse mode, setting a value greater than zero switches it out of simultaneous mode. While switching back and forth between modes during an experiment is possible, it may increase the inter-sweep delay. This control is disabled if independent 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.

Rapid² device The Rapid² is rather different from the other two 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 Rapid² 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:

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 Rapid² 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, clear it to give Signal control. 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.

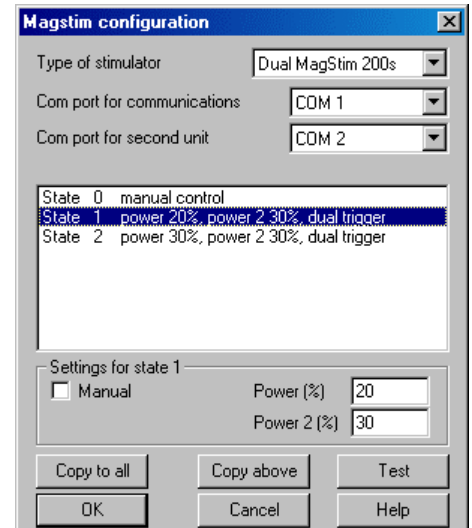
Freq This item is only available single-pulse mode is not in use, it sets the pulse frequency in Hertz, you can set any value from 0 to 100. This control is disabled for manually controlled states or if single-pulse mode is selected.

Dual 200² devices Two MagStim 200²s can be controlled by Signal to achieve much the same effect as using a BiStim², 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 200² option is selected a separate selector for the second serial-line port is shown but as for the 200² there are no controls governing 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, clear it to give Signal control.

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.

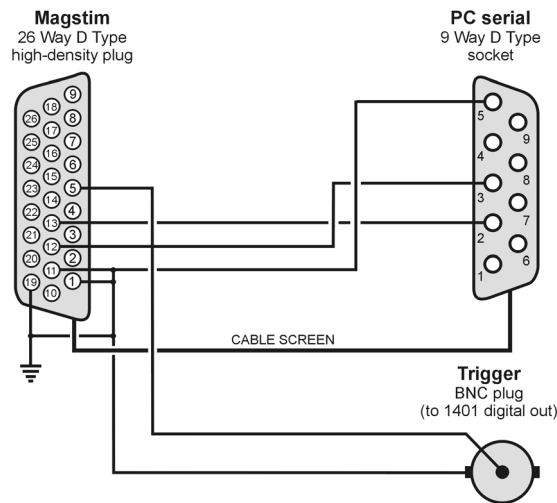


MagStim Connections and cabling

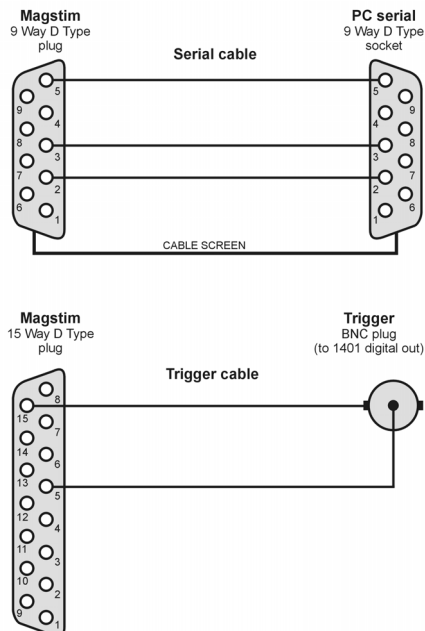
There are two connector arrangements provided by MagStim for serial line control and external triggering of the unit. Older 200 and BiStim units have an arrangement with a 9-way serial line and separate 15-way trigger connector while newer 200 and BiStim units and all Rapids have a high-density 26-way 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 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.

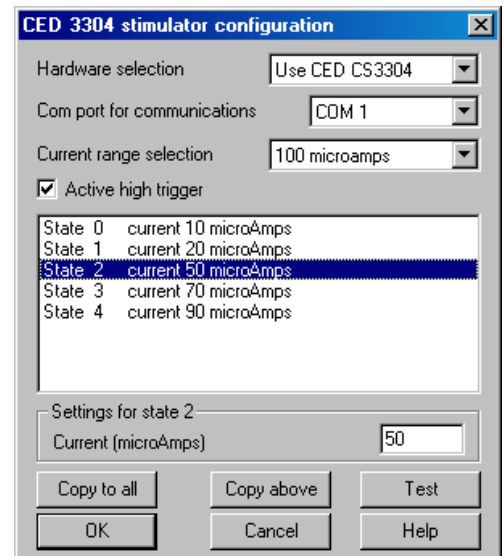
Signal monitors the stimulator health; waiting until it is ready for use before allowing a sampling sweep to proceed and terminating sampling if a hardware problem develops. To aid data analysis, Signal saves the stimulation current in the data section variables of the sampled file. This value, in uA, 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.



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.

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.

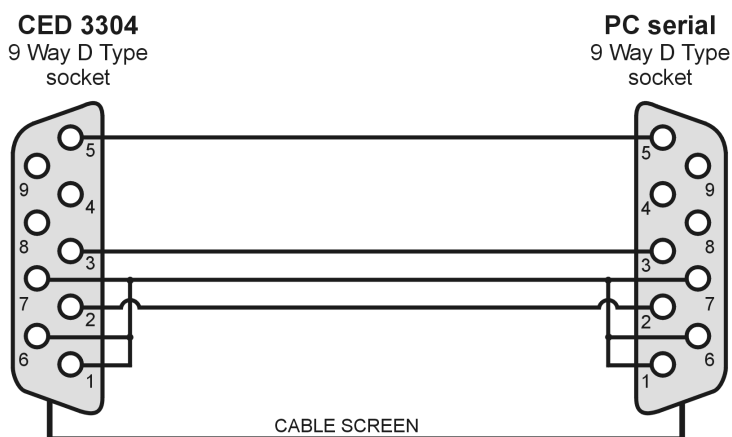
Simple use If you want to use the CED 3304 from Signal very simply with a single current setting, you must still enable multiple states. Set one extra state (so states are enabled), and set the desired current for state zero, and then sample using state zero only. This gives you access to the configuration 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 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 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.

Index

- 1—
- 1401
 Device driver, 16-1
 Monitor revision, 16-1
- 1902
 AC coupling, 18-3
 Interactive support, 18-1
 Trigger inputs, 18-3
- 3—
- 3304 auxiliary states device, 20-8
3-point smooth data, 11-19
- 5—
- 5-point smooth data, 11-19
- A—
- Abort sampling, 3-18
Active cursors, 12-1
ADD output instruction, 6-22
ADDAC output instruction, 6-12
ADDI output instruction, 6-22
Adding a pulse, 5-3
All pass filter, 17-3
All stop filter, 17-3
Amplifier telegraphs, 19-1
Amplitude histogram, 11-2
Analysing data, 2-10
Analysis menu
 Fit data, 11-8
Analysis menu, 11-1
 Add to buffer, 11-18
 Append frame, 11-17
 Append frame copy, 11-17
 Average into buffer, 11-18
 Burst duration histogram, 11-16
 Clear buffer, 11-17
 Copy from buffer, 11-18
 Copy to buffer, 11-18
 Delete channel, 11-17
 Delete frame, 11-17
 Exchange buffer, 11-18
 Fit data, 11-8
 Keyboard alternatives, 11-20
 Modify channels, 11-19
 Multiple frames, 11-18
 New idealised trace, 11-13
 New Memory View, 11-1
 New XY View, 11-7
 Open/Closed amplitude histogram, 11-16
 Open/Closed time histogram, 11-16
 Open/Closed times, 11-13
 Process settings, 11-6
 Subtract buffer, 11-18
 Tag frame, 11-19
- ANGLE output instruction, 6-16
Append frame, 11-17
Append frame copy, 11-17
Area
 between cursors, 12-5
 under curve between cursors, 12-5
Area under curve, 2-8, 12-5
Arrange icons, 15-1
Artefact rejection, 3-16
Artefact rejection dialog, 3-16
ASCII values table, 3-3
Assume Power1401 hardware, 9-7
Auto-Average of waveform, 11-2
Automate
 Directory for file, 3-15
 File naming, 3-15
 File saving, 3-15
 File size limit, 3-15
 Frames limit, 3-15
 Time limit, 3-15
Auxiliary states devices, 20-1
Average. *see* Mean
Average of waveform, 11-1
 Number of sweeps, 10-4
Axis controls, 2-5, 10-2
 Show and Hide, 10-3
- B—
- Band pass filter, 17-3
Band stop filter, 17-3
Bitmap output, 8-4, 9-1
Black and white displays, 10-7
BRAND output instruction, 6-26
BRANDV output instruction, 6-26
Breaking out of processing, 11-6
Buffer
 Add frame data into buffer, 11-18
 Add to current frame, 11-18
 Average into, 11-18
 Clear buffer, 11-17
 Copy frame data into buffer, 11-18
 Copy to frame data, 11-18
 Description, 11-17
 Exchange with frame data, 11-18
 Multiple frame operations, 11-18
 Remove frame from average, 11-18
 Show buffer, 10-1
 Subtract current frame, 11-18
 Subtract from current frame, 11-18
Bxx output instructions, 6-21
- C—
- CALL and CALLV output instructions, 6-19
CANGLE output instruction, 6-16
Cascade windows, 15-1
Case sensitivity
 In searches, 9-3, 9-4
 Output sequencer, 6-7
CED Software help desk, 16-2
CEDCOND.INI, 18-4
CEDCOND.LOG file, 9-8
cfb file extension, 8-2
CFS
 Library, 1-3
 Standard file extension, 8-1
cfs file extension, 8-1
CHAN output instruction, 6-24
Change colours, 10-7
Change process settings, 11-6
Channel display
 Colour override, 10-7
 Draw mode colour, 10-7
 Order, 9-4
Channel Information, 10-4
Channel selection, 2-1
Channels
 3-point smooth, 11-19
 5-point smooth, 11-19
 Differentiate, 11-19
 Integrate, 11-19
 Marker channels, 3-2
 Modify data, 11-19
 Negate, 11-19
 Offset data, 11-19
 Rectify, 11-19
 Scale data, 11-19
 Set DC measurement area, 11-19
 Shift data, 11-19
 Show and hide, 10-3
 specifying, 2-6
 Subtract DC level, 11-19
 Types of channel, 3-1
 Waveform channels, 3-1
 Zero, 11-19
Clamping configuration, 4-1
Clamping experiment, 4-3
Clamping support, 4-1
Clear data, 11-19
Clear text or memory view, 9-2
Clipboard
 Copy cursor values, 12-4, 12-6
 Copy data view as picture, 9-1
 Copy data view as text, 9-2
 Copy to, 9-1
 Copy XY view as text, 9-3
 Cut text to, 9-1
 Paste text, 9-1
Close all associated windows, 8-3, 15-1
Close document, 8-3
CLRC output instruction, 6-18
Coefficients of filters, 17-5
COFF output instruction, 6-17
Colour
 Enabling, 10-7
-

- Colour dialog, 10-7
 - Channel colours, 10-7
 - Comment
 - Output sequencer, 6-7
 - Comment file at sampling end, 9-7
 - Compile output sequence, 6-2
 - Conditional averaging state, 11-5
 - Conditioner
 - CEDCOND.INI settings file, 18-4
 - CEDCOND.LOG file, 9-8
 - Sample menu, 13-1
 - Serial port, 9-8
 - Configuration files, 8-2
 - Contents, 3-20
 - Load and run from Sample Bar, 13-1
 - Load and save, 8-5
 - Connections
 - Power1401 DACs 2 and 3, 6-12
 - Signal conditioner, serial, 9-8, 18-4
 - Waveform output, 6-12
 - Continue sampling, 3-18, 3-19
 - Convert
 - foreign file format, 8-3
 - Copy
 - Cursor values, 12-4, 12-6
 - Data view as text, 9-2
 - File and Memory views as pictures, 9-1
 - Text, 9-1
 - XY view as text, 9-3
 - Copy data
 - As binary numbers, 9-1
 - As bitmap file, 9-1
 - As picture, 9-1
 - As text file, 9-1
 - Cosine wave output, 6-14
 - Count of markers, 2-8
 - CPHASE output instruction, 6-16
 - CRATE output instruction, 6-15
 - CRATEW output instruction, 6-15
 - Creating a new document, 3-16
 - CRINC output instruction, 6-18
 - CRINCW output instruction, 6-18
 - CSZ output instruction, 6-14
 - CSZINC output instruction, 6-15
 - Cubic Spline draw mode for waveforms, 2-6
 - Cursor labelling styles, 2-4, 12-3
 - Cursor menu, 12-1
 - Cursor mode, 12-1
 - Cursor regions, 12-5
 - Delete, 12-3
 - Delete horizontal, 12-3
 - Display all, 12-3
 - Display all horizontal, 12-4
 - Display Y Values, 12-4
 - Move To, 12-3
 - Move To Level, 12-4
 - New cursor, 12-1
 - New horizontal cursor, 12-3
 - Renumber cursors, 12-3
 - Renumber horizontal cursors, 12-4
 - Cursors
 - Active, 12-1
 - Add horizontal, 12-3
 - Adding, 2-3, 12-1
 - Button for new, 2-3
 - Delete cursor, 12-3
 - Delete horizontal, 12-3
 - Display all, 12-3
 - Display all horizontal, 12-4
 - Horizontal label style, 12-4
 - Label style, 12-3
 - Mode, 12-1
 - Move window to centre a cursor, 12-3
 - Offset channel to centre horizontal cursor, 12-4
 - Regions, 12-5
 - Renumber, 12-3
 - Renumber horizontal, 12-4
 - Value at, 12-4
 - Values between, 2-8, 12-5
 - Curve fitting, 11-8
 - Testing the fit, 11-12
 - Curve Fitting, 11-8
 - Cut text, 9-1
 - CWAIT output instruction, 6-17
 - CWCLR output instruction, 6-18
 - CyberAmp
 - Interactive support, 18-1
- D—**
- DAC connections for Power1401, 6-12
 - DAC output during sampling, 6-1
 - DAC output instruction, 6-12
 - DAC output range, 3-14
 - DAC outputs, 3-14
 - DAC outputs for state, 7-6
 - DAC scaling and units, 3-10
 - DANGLE output instruction, 6-16
 - Data
 - Exporting, 8-4
 - Incoming sampled data, 3-16
 - Saving, 8-3
 - Data channels, 2-1
 - Data export
 - CFS data format, 9-6
 - Data update mode, 8-5
 - Data view
 - Copy as Text, 9-2
 - dB scale, 17-6
 - DBNZ output instruction, 6-19
 - DC measurement area for subtract, 11-19
 - Decibel scale, 17-6
 - DEFAULT.S2C configuration, 10-8
 - DEFAULT.SGC default configuration file, 8-5
 - DELAY output instruction, 6-19
 - Delete channel, 11-17
 - Delete frame, 11-17
 - Delete selected text, 9-1
 - DIBEQ output instruction, 6-11
 - DIBNE output instruction, 6-11
 - Differentiate data, 11-19
 - Differentiator, 17-8
 - Differentiator filter, 17-4
 - Digital filter, 17-1
 - Digital filter types, 17-7
 - Digital input
 - test bits, 6-11
 - test saved bits, 6-11
 - Digital markers, 3-2
 - Digital output, 6-10
 - micro1401 and Power1401, 6-10
 - Digital output voltages, 3-14
 - Digital outputs, 3-14
 - Bit numbers, 3-14
 - Connections, 3-14
 - Socket pins, 3-14
 - Digital outputs enable, 3-11
 - Digital outputs for state, 7-6
 - Digital state outputs
 - Bit numbers, 3-14
 - Socket pins, 3-14
 - DIGLOW output instruction, 6-10
 - DIGOUT output instruction, 6-10
 - Directory for file name generation, 3-15
 - Directory for file saving, 3-15
 - Directory for new data files, 9-4
 - DISBEQ output instruction, 6-11
 - DISBNE output instruction, 6-11
 - Display
 - Customise, 2-6
 - DIV output instruction, 6-22
 - DOFF output instruction, 6-17
 - Dots draw mode for markers, 10-5
 - Dots draw mode for waveforms, 2-6
 - DPHASE output instruction, 6-16
 - DRATE output instruction, 6-15
 - DRATEW output instruction, 6-15
 - Draw mode, 10-4
 - Drawing modes
 - Data view, 10-4, 10-5
 - DRINC output instruction, 6-18
 - DRINCW output instruction, 6-18
 - DSZ output instruction, 6-14
 - DSZINC output instruction, 6-15
 - Duplicate data view, 15-1

DWAIT output instruction, 6-17
 DWCLR output instruction, 6-18
 Dynamic output states, 7-2

—E—

Edit bar, 10-1
 Edit menu, 9-1
 Clear, 9-2
 Copy, 9-1
 Copy as text, 9-3
 Cut text, 9-1
 Delete selection, 9-1
 File comment, 9-4
 Find text, 9-3
 Frame comment, 9-4
 Paste, 9-1
 Preferences, 9-4
 Select All, 9-4
 Show event details, 9-9
 Undo, 9-1
 Edit text
 Editor settings, 9-4
 Email support, 8-5
 Enlarge view, 10-2
 Error bars, 11-1
 Errors
 Output sequencer compiler, 6-28
 Evaluate, 14-1
 Event 1 sampling trigger, 3-2, 3-18
 Exchange data with another computer, 8-5
 Exit, 8-6
 Export data
 As bitmap file, 8-4
 As CFS file, 8-4
 As text file, 8-4
 As Windows Metafile, 8-4
 To clipboard, 9-3
 External convert sampling, 3-5
 External digital states, 7-5

—F—

FFT (Fast Fourier Transform), 11-3
 File comment, 9-4
 File comment at sampling end, 9-7
 File format converters, 8-3
 File icons, 1-3
 File menu, 8-1
 Close, 8-3
 Close All, 8-3
 Data update mode, 8-5
 Exit, 8-6
 Export As, 8-4
 Import, 8-3
 Load configuration, 8-5
 New File, 8-1
 Open, 8-2

Page Setup, 8-5
 Print, 8-6
 Print preview, 8-6
 Print screen, 8-6
 Print selection, 8-6
 Print visible, 8-6
 Revert To Saved, 8-4
 Save and Save As, 8-3
 Save configuration, 8-5
 Send Mail, 8-5
 File name extensions, 8-1
 File name generation, 3-15
 File size limit, 3-15
 File view, 2-1, 2-9
 Filtbank.cfb filter bank file, 17-3
 Filter bank, 17-3
 Finding a pulse, 5-4
 Finish sampling, 3-18, 3-19
 FIR filter, 17-1
 Attenuation and ripple, 17-7
 Coefficients, 17-5
 Differentiator example, 17-12
 Frequencies, 17-5
 Frequency bands, 17-6
 Maximum useful attenuation, 17-6
 Multiband example, 17-11
 Number of coefficients, 17-9
 Nyquist frequency, 17-10
 Pink noise from white noise, 17-8
 Ripple in bands, 17-7
 Slope for differentiator, 17-8
 Transition region, 17-6
 Weighting, 17-6
 FIR filters, 17-5
 FIRMake()
 filter types, 17-7
 FIRMake()
 Discussion, 17-5
 First frame, 10-1
 Fit data, 11-8
 Fitting, 11-8
 Fixed interval period, 5-8
 Fixed interval sweeps, 3-4
 Fixed interval variation, 5-8
 Font
 for new data documents, 9-5
 for text views, 9-5
 Font selection, 10-6
 Footer, 8-5
 Format of Signal data files, 1-3
 Frame buffer. *See* Buffer
 Frame comment, 9-4
 Frame zero, 3-16
 Frames
 Channels, 2-2
 Comment, 2-2
 Description, 2-2
 Display list, 10-1

Extra data, 2-2
 Flags, 2-3
 Goto frame, 10-1
 Next, 10-1
 Overdrawing, 10-1
 Overdrawing online, 3-19
 Previous, 10-1
 Specifying, 2-3
 State, 2-2
 Status bar information, 2-2
 Tag, 2-3
 Tagging and untagging, 11-19
 User variables, 2-3
 Variation, 2-2
 Frames limit, 3-15

—G—

Gradient of line, 2-8, 12-5
 Grid
 Set colour, 10-7
 Grid show and hide, 10-3
 Group channels, 10-3

—H—

HALT output instruction, 6-20
 Hardware required for Signal, 1-2
 Header, 8-5
 Help, 2-1, 16-1
 Help desk, 16-2
 Hexadecimal marker codes, 3-3
 Hide window, 15-1
 High pass filter, 17-3
 High pass filter example, 17-10
 Hilbert transformer, 17-8, 17-14
 Histogram draw mode for waveforms, 2-6
 Horizontal cursor labelling styles, 12-4
 Hz () sequencer expression function, 6-7

—I—

I/V Curves, 11-7
 Icons for files, 1-3
 Icons, arrange, 15-1
 Idealised trace
 Drawing mode, 10-5
 Idealised trace editing, 9-9
 Idealised traces, 11-13
 import folder, 8-3
 Import foreign data file, 8-3
 Impulse response, 17-5
 Inflection cursor mode, 12-2
 Installing Signal, 1-2
 Instructions for output sequencer, 6-6
 Integrate data, 11-19

—J—

JUMP output instruction, 6-20

—K—

Key for XY view, 10-4
Keyboard control of display, 10-8
Keyboard control of sequencer, 6-1, 6-7
Keyboard driven analysis, 11-20
Keyboard markers, 3-2
 Enable, 3-6
 Entering, 3-19

—L—

Label for cursor, 2-4, 12-3
Label for horizontal cursor, 12-4
Last frame, 10-1
LAST.SGC last sampling configuration, 8-5
Leak subtraction, 11-4
Least squares fitting, 12-5
Licence information, 1-4
Line draw mode for markers, 10-5
Line draw mode for waveforms, 2-6
Line style in XY views, 10-6
Line thickness for printing, 9-4
Line width for displays, 9-5
Lock y axes, 10-3
Log view, 2-1
Logarithmic scale, 17-6
Low pass differentiator filter, 17-4
Low pass filter, 17-3
Low pass filter example, 17-8

—M—

Magnify pointer, 2-4
MagStim auxiliary states device, 20-1
MARK output instruction, 6-25
Marker count, 2-8
Marker display
 Dots, 10-5
 Lines, 10-5
 Rate, 10-5
Markers
 Drawing mode, 10-5
Maximum
 Sampled frame length, 3-5
 Total sampling rate, 3-5
Maximum and Minimum cursor modes, 12-1
Maximum between cursors, 12-5
Maximum waveform output rate, 5-7
Mean value between cursors, 2-8, 12-5
Membrane analysis, 4-4
Memory
 Minimum required in Windows, 1-2
Memory view, 2-9

Behaviour, 2-12
Clear, 9-2
Creating a new view, 11-1
Loading as file view, 2-12
Number of sweeps, 10-4
Saving to disk, 2-12

Menus

Analysis, 11-1
Cursor, 12-1
Edit, 9-1
File, 8-1
Help, 16-1
Sample, 13-1
View, 10-1
Windows, 15-1
Metafile image export, 8-4
Metafile output scaling, 9-4
Microseconds time, 9-5
Milliseconds time, 9-5
Minimum between cursors, 12-5
Modify channel data, 11-19
Modulus of data, 12-5
MOV output instruction, 6-21
MOVI output instruction, 6-21
Moving a pulse, 5-4
MOVRND output instruction, 6-28
ms () sequencer expression function, 6-7
MUL and MULI output instructions, 6-23
Multiband filter, 17-7
Multiband with 3 dB/octave cut, 17-8
MultiClamp 700 telegraph configuration, 19-2
MultiClamp 700 telegraphs, 19-2
Multiple frame operations, 11-18
Multiple frame states, 3-5, 7-1
 3304, 20-8
 Auxiliary devices, 20-1
 DAC outputs for state, 7-6
 Digital output bits, 7-6
 MagStim, 20-1
 Number of states, 7-2
 Online control, 7-7
 Randomised, 7-3
 Sequencing, 7-3
 Set number, 7-2
 Static outputs, 7-6
 Uses, 7-1
Multiple states enable, 3-5, 7-1

—N—

NEG output instruction, 6-21
Negate data, 11-19
New document, 3-16, 8-1
 Temporary directory for data files, 9-4
New file from existing file, 8-4
New memory view, 11-1

New XY view, 11-7
Next frame, 10-1
 Button, 2-3
NOP output instruction, 6-20

—O—

Offset data, 11-19
OFFSET output instruction, 6-17
One and a half high pass filter, 17-4
One and a half low pass filter, 17-4
Online clamping support, 4-1
Online data processing, 11-6
Online pulses control, 5-9
Online update of memory view, 11-6
Open/Closed times, 11-13
Opening
 Configuration file, 8-5
 New document, 8-1
 Old document, 8-2
Optimise Y axis, 10-3
Options for XY view, 10-4
Order of channels, 9-4
Output sequencer, 6-1
 Access to data capture, 6-24
 Add constant to variable, 6-22
 Arbitrary waveform output control, 6-28
 Calculate variable values, 6-8, 6-10, 6-11, 6-18
 Compare variable, 6-21
 Compile sequence, 6-2
 Compiler errors, 6-28
 Control panel, 6-1
 Copy variable, 6-21
 DAC outputs, 6-12
 Disable interactive jumps, 3-12
 Divide variable, 6-22
 Example, 6-3
 Expressions, 6-7
 Format text, 6-3
 Format with step numbers, 6-3
 Get current sample points, 6-24
 Get current sample time, 6-26
 Get current sweep state, 6-25
 Get sweep start time, 6-25
 Instruction format, 6-7
 Instructions, 6-6
 Multiply variables, 6-23
 Negate variable, 6-21
 Randomisation, 6-26
 Reciprocal of variable, 6-22
 Set file to use, 6-3
 Set state of current sweep, 6-25
 Set variable value, 6-21
 TABDAT directive, 6-9
 Table of values, 6-9
 TABSZ directive, 6-9

- Trigger sweep, 6-26
 - Variable sum and difference, 6-22
 - Variables, 6-8
 - Wait till time in sweep, 6-26
 - Outputs
 - Absolute levels, 3-10
 - DAC, 3-14
 - DAC enable, 3-10
 - DAC scaling, 3-10
 - DAC units, 3-10
 - Delay relative to sampling, 3-10
 - Digital, 3-14
 - Digital outputs enable, 3-11
 - Pulse output frame, 5-1
 - Relative levels, 3-10
 - Start of pulses, 5-1
 - Synchronisation with sampling, 3-10
 - Time resolution, 3-10
 - Type, 3-9
 - Outputs dialog
 - Absolute times, 3-10
 - Outputs frame length, 5-8
 - Outputs frame sweeps, 3-4
 - Overdraw mode, 10-1
- P—**
- Pass band*, 17-6, 17-7
 - Paste data into view, 9-1
 - Paste text, 9-1
 - Patch clamping, 4-1
 - Pausing sampling, 3-18
 - Peak between cursors, 12-6
 - Peak search
 - cursor mode, 12-2
 - Peri-trigger
 - Configuration, 3-7
 - Level adjust online, 3-19
 - Peri-trigger sweeps, 3-4
 - PHASE output instruction, 6-16
 - pls file extension, 8-1
 - POINT style in XY views, 10-6
 - POINTS output instruction, 6-24
 - Port
 - Configuration, 3-9
 - Full value, 3-8
 - Options, 3-9
 - Zero value, 3-8
 - Power spectrum, 11-3
 - Number of sweeps, 10-4
 - Power1401 configuration adaption, 9-7
 - Preferences, 9-4
 - Assume Power1401 hardware, 9-7
 - File comment at sampling end, 9-7
 - Font for text views, 9-5
 - Line widths, 9-5
 - Time units, 9-5
 - Preferences file, 8-2
 - Preferences folder, 3-20
 - Previous frame, 10-1
 - Button, 2-3
 - Print
 - Line thickness, 9-4
 - Line widths, 9-5
 - Preview printed output, 8-6
 - Print screen, 8-6
 - Printing data, 8-6
 - Selected cursor values, 12-4, 12-6
 - Print control
 - Line thickness, 9-4
 - Printing, 8-5
 - Process dialog, 2-11, 11-5
 - for new file, 3-17, 11-6
 - Process frames dialog, 11-5
 - Process settings, 11-6
 - Prompt to save result and XY views, 9-4
 - Pulse output frame, 5-1
 - Pulse outputs start, 5-1
 - Pulses
 - Adding a pulse, 5-3
 - Configuration, 5-1
 - Controlling online, 5-9
 - Current pulse, 5-2
 - Dialog, 5-1
 - Digital bits, 5-5
 - Display, 5-1
 - Drag and drop, 5-3
 - Finding a pulse, 5-4
 - Fixed interval period, 5-8
 - Fixed interval variation, 5-8
 - Initial level, 5-5
 - Moving a pulse, 5-4
 - Paste waveform, 5-7
 - Pulse selection, 5-2
 - Pulse train, 5-6
 - Ramp, 5-6
 - Removing a pulse, 5-4
 - Sine wave, 5-6
 - Square pulse, 5-5
 - Square with varying amplitude, 5-5
 - Square with varying duration, 5-6
 - Step change, 5-8
 - Total variation, 5-8
 - Trigger sampling, 5-8
 - Values, 5-3
 - Varying, 5-8
 - Pulses or sequencer, 3-13
- R—**
- RAMP output instruction, 6-13
 - Randomisation in output sequencer, 6-26
 - Rate marker display mode, 10-5
 - RATE output instruction, 6-15
 - RATEW output instruction, 6-15
 - RECIP output instruction, 6-22
 - Rectify data, 11-19
 - Reduce view, 10-2
 - Regions dialog, 2-8
 - Regular expression search, 9-3
 - Relative measurements, 12-4
 - Remove Signal, 1-2
 - Removing a pulse, 5-4
 - Renumber cursors, 12-3
 - Renumber horizontal cursors, 12-4
 - Replace text, 9-4
 - Repolarisation cursor mode, 12-3
 - REPORT output instruction, 6-25
 - Resource files, 8-2
 - Resources, free, 16-2
 - Restart sampling, 3-18
 - Result view
 - Prompt to save unsaved view, 9-4
 - RETURN output instruction, 6-19
 - Revert text document to last saved, 8-4
 - RINC output instruction, 6-18
 - RINCW output instruction, 6-18
 - Rotate data, 11-19
 - Run script
 - from Script Bar, 14-2
- S—**
- s () sequencer expression function, 6-7
 - Sample Bar, 13-1
 - Sample Bar List, 13-1
 - Sample interval, 3-1
 - Sample menu
 - Output controls, 13-2
 - Sample Bar, 13-1
 - Sample Bar List, 13-1
 - Sampling configuration, 13-1
 - Signal conditioner setup, 13-1
 - Sample rate
 - For waveform data, 3-1
 - Minimum, 3-1
 - Sampling
 - Aborting, 3-18
 - Accept sweep, 3-18
 - ADC ports, 3-6
 - Artefact reject configuration, 3-16
 - Automation configuration, 3-15
 - Configuration contents, 3-20
 - Configuration dialog, 3-4
 - Continue with next sweep, 3-18, 3-19
 - Control of states, 7-7
 - Controls during, 13-2
 - Controls during, 3-17
 - Digital outputs, 6-1
 - File size limit, 3-15
 - Finish button, 3-18, 3-19
 - Fixed interval sweeps, 5-8

- Fixed interval variation, 5-8
 - General configuration, 3-4
 - Keyboard marker entry, 3-19
 - Maximum frame length, 3-5
 - Maximum rate, 3-5
 - Menu, 13-1
 - Output during, 6-1
 - Outputs configuration, 3-9
 - Outputs frame sweeps, 5-8
 - Overdraw frames, 3-19
 - Pausing at sweep end, 3-18
 - Peri-trigger configuration, 3-7
 - Peri-trigger level adjust, 3-19
 - Reject sweep, 3-18
 - Restarting, 3-18
 - Sample rate, 3-5
 - Saving configuration, 3-20
 - Saving new data, 3-20
 - Setting configuration, 3-20
 - Stopping sampling, 3-19
 - Time limit, 3-15
 - Triggered start, 3-18
 - Write sweep automatically, 3-17
 - Sampling configuration, 13-1
 - Loading and saving, 8-5
 - Save changed data, 8-5
 - Preferences, 9-6
 - Save file at sampling end, 3-15
 - Saving configurations, 3-20
 - Scale data, 11-19
 - SCAN method, 11-13
 - Screen dump, 8-6
 - Script Bar, 14-2
 - Script menu
 - Evaluate, 14-1
 - Script Bar, 14-2
 - Scroll bar show and hide, 10-3
 - Search for text, 9-3
 - Selecting a channel, 2-1
 - Selecting channels, 2-1
 - Send Mail, 8-5
 - Sequencer
 - Compile sequence, 6-2
 - Example, 6-3
 - Expressions, 6-7
 - Format text, 6-3
 - Format with step numbers, 6-3
 - Instruction format, 6-7
 - Instructions, 6-6
 - Set file to use, 6-3
 - Variables, 6-8
 - Sequencer control panel, 6-1
 - Sequencer or pulses, 3-13
 - Sequencer outputs, 6-1
 - Sequencer technical information, 6-1
 - Serial number, 16-1
 - SETS output instruction, 6-25
 - SGC Configuration file extension, 8-2
 - SGP Preferences file extension, 8-2
 - SGR Resource file extension, 8-2
 - sgs file extension, 8-1
 - SGS standard file extension, 8-1
 - Shift data, 11-19
 - Show hidden window, 15-1
 - Show\Hide
 - Axes, 10-3
 - Channels, 10-3
 - Edit bar, 10-1
 - Grid, 10-3
 - Scroll bar, 10-3
 - Status bar, 10-1
 - Toolbar, 10-1
 - Signal conditioner, 18-1
 - Connections, 18-4
 - Sample menu, 13-1
 - Sine wave output, 6-14
 - Skyline draw mode for waveforms, 2-6
 - Slope of line, 2-8, 12-5
 - Slope peak and trough cursor modes, 12-2
 - Slope search cursor modes, 12-2
 - Slope% cursor mode, 12-2
 - Smooth data, 11-19
 - Software help desk, 16-2
 - Standard 1401 telegraphs, 19-1
 - Standard deviation
 - curve fitting, 11-12
 - Standard display settings, 10-3
 - STATE output instruction, 6-25
 - States sequencing, 7-3
 - Static output states, 7-6
 - Static outputs states, 7-6
 - Status bar, 2-2
 - Stop band, 17-6, 17-7
 - Stop Process command, 11-6
 - Stop sampling, 3-18, 3-19
 - SUB output instruction, 6-22
 - Subtract DC level from data, 11-19
 - Sum of data, 12-5
 - Sweep
 - Mode selection, 3-4
 - Sweep mode
 - Basic, 3-4
 - Fixed interval, 3-4
 - Outputs frame, 3-4
 - Peri-trigger, 3-4
 - SWEET output instruction, 6-25
 - sxy file extension, 8-1
 - Synchronisation of outputs, 3-10
 - System resources, 16-2
 - System software required, 1-2
 - SZ output instruction, 6-14
 - SZINC output instruction, 6-15
- T—
- TABDAT directive, 6-9
 - TABLD output instruction, 6-23
 - TABST output instruction, 6-23
 - TABSZ directive, 6-9
 - Tag frame, 11-19
 - Telegraph configuration dialog, 19-1
 - Telegraph output, 19-1
 - Text output, 9-1
 - Data specification, 9-3
 - Format specification, 9-2
 - From data view, 9-2
 - From XY view, 9-3
 - Threshold crossing cursor mode, 12-2
 - TICKS output instruction, 6-26
 - Tile windows
 - Horizontally, 15-1
 - Vertically, 15-1
 - Time limit, 3-15
 - Time shift, 17-5
 - Time units
 - in X axis dialog, 2-5
 - Setting, 9-5
 - Time zero, 2-7, 12-4
 - Times
 - Specifying, 2-5
 - Toolbar, 10-1
 - Toolbar, 2-2
 - Trend plots, 11-7
 - TRIG output instruction, 6-26
 - Trigger
 - Analogue, 3-7
 - Digital bit, 3-8
 - Peri-trigger modes, 3-7
 - Pre-trigger points, 3-8
 - Sweep trigger, 3-6
 - Waveform, 3-7
 - Triggered start of sampling, 3-18
 - Trough between cursors, 12-6
 - Trough cursor mode, 12-2
 - TTL compatible signals, 3-2
 - Two band pass filter, 17-4
 - Two band stop filter, 17-4
 - TXT standard file extension, 8-1
- U—
- Undo command, 9-1
 - Un-magnify mouse pointer, 2-4
 - Update mode for changed data, 8-5
 - Updating Signal, 1-2
 - us () sequencer expression function, 6-7
- V—
- Value at cursor, 12-4
 - Value between cursors, 12-5

- VAngle () sequencer expression
 - function, 6-7
- VAR directive in output sequencer, 6-8
- Variables for output sequencer, 6-8
- VarValue script, 6-8, 6-10, 6-11, 6-18
- VDAC0-7 sequencer variables, 6-8
- VDAC16 () sequencer expression
 - function, 6-8
- VDAC32 () sequencer expression
 - function, 6-8
- VDigIn sequencer variable, 6-8
- VHz () sequencer expression function, 6-7
- View handle
 - Interactive access to, 15-2
- View menu, 10-1
 - Add frame to list, 10-1
 - Channel Information, 10-4
 - Colour commands, 10-7
 - Draw mode, 10-4
 - Edit bar, 10-1
 - Enlarge and reduce, 10-2
 - Font, 10-6
 - Frame display list, 10-2
 - Goto frame, 10-1
 - Info, 10-4
 - Keyboard alternatives, 10-8
 - Next frame, 10-1
 - Overdraw frame list, 10-1
 - Previous frame, 10-1
 - Show buffer, 10-1
 - Show/Hide channel, 10-3
 - Standard display, 10-3
 - Toolbar, 10-1
 - X Axis Range, 10-2
 - Y Axis Range, 10-3
- Voltage clamp, 4-1
- Voltage limits
 - TTL inputs, 3-2
- W—
 - WAITC output instruction, 6-17
 - WAVE output instruction, 6-28
 - WAVEBR output instruction, 6-28
 - Waveform average, 2-10
 - Waveform data
 - Amplitude histogram, 11-2
 - Area, 12-5
 - Area over zero, 12-5
 - Average, 11-1
 - Connections, 3-1, 3-2
 - Copy and Export format, 9-2
 - Leak subtraction, 11-4
 - Mean value, 12-5
 - Modulus, 12-5
 - Multiple averages, 11-2
 - Power spectrum, 11-3
 - Slope of best fit line, 12-5
 - Sum, 12-5
 - Underlying CFS data, 3-1
 - Value at cursor, 12-4
 - Waveform display
 - Cubic Spline, 2-6
 - Dots, 2-6, 10-5
 - Histogram, 2-6, 10-5
 - Line, 2-6, 10-5
 - Skyline, 2-6, 10-5
 - Waveform output, 5-6
 - During sampling, 6-1
 - Maximum rates, 5-7
 - test from output sequencer, 6-28
 - Waveforms
 - Drawing mode, 10-5
 - Web site, 1-2
 - whole cell, 4-1
 - Width of a memory view, 2-10
 - Window for FFT, 11-3
 - Window menu, 15-1
 - Arrange icons, 15-1
 - Cascade, 15-1
 - Duplicate, 15-1
 - Hide, 15-1
 - Show, 15-1
 - Tile Horizontally, 15-1
 - Tile Vertically, 15-1
 - Windows, 15-2
 - Write to disk, 3-17
 - WSWP output instruction, 6-26
- X—
 - X axis control, 2-5
 - X Range dialog, 2-5, 10-2
 - XY Draw Mode, 10-6
 - XY view, 2-12
 - Auto-expand axes, 10-4
 - Copy as text, 9-3
 - Creating a new view, 11-7
 - Delete channel, 11-17
 - Draw mode, 10-6
 - Example, 2-12
 - Key, 10-4
 - Line style, 10-6
 - Options, 10-4
 - Point style, 10-6
 - Prompt to save unsaved view, 9-4
- Y—
 - Y axis control, 2-6
 - Y Range dialog, 10-3
 - Y zero, 2-7, 12-4
- Z—
 - Zero data, 11-19
 - Zero region, 2-8, 12-5
 - Zoom in button, 2-3
 - Zoom out button, 2-3