

USER GUIDE

FLOWSTONE



DSP Robotics

Version 1.0

CHAPTER 1 **1 Introduction** **6**

ABOUT THIS GUIDE..... 7
 WHAT IS FLOWSTONE?.....8
 HOW IT WORKS.....8
 DIGITAL SIGNAL PROCESSING IN A NUTSHELL.....10

CHAPTER 2 **2 User Interface** **12**

TOOLBOX..... 14
 COMPONENT BROWSER.....14
 FILTER BAR15
 SEARCH BAR.....15
NAVIGATOR..... 16
 JUMPING.....16
 PANNING.....17
 BOOKMARKS.....17
SCHEMATIC WINDOW..... 18
 ZOOMING.....18
 PANNING.....19

CHAPTER 3 **3 Components & Links** **20**

COMPONENTS..... 25
 ADDING A COMPONENT.....25
 SELECTIONS AND THE ACTION PANEL.....28
 MOVING COMPONENTS.....28
 NAMING COMPONENTS.....29
 DELETING COMPONENTS.....30
 RESIZING.....30
 MULTIPLE SELECTIONS.....31
 CONNECTOR LABELS.....32
 CUT, COPY AND PASTE.....32
LINKS..... 33
 CREATING A LINK.....33
 ALLOWED LINKS.....34
 MOVING A LINK.....34
 DELETING A LINK.....35
 LINK ORDER.....36
 BENDING LINKS.....37
 AUTO LINKING.....38
 SMART LINKING.....39
 REMOVING MULTIPLE LINKS.....40
 WIRELESS LINKS.....40

KEY DIFFERENCES.....	45
APPEARANCE.....	45
FRONT PANEL.....	46
PROPERTIES.....	46
TOOLBOX.....	47
BASIC OPERATIONS.....	48
MOVING INTO A MODULE.....	48
INPUTS AND OUTPUTS.....	49
TEMPLATE CONNECTORS.....	50
INPUT AND OUTPUT NAMES.....	50
MAKE MODULE.....	51
PROPERTIES.....	51
WIRELESS MODULES.....	52
FRONT PANEL.....	53
ENABLING THE FRONT PANEL.....	53
EDITING THE FRONT PANEL.....	54
SELECTING.....	55
MOVING.....	56
RESIZING.....	56
JUMPING.....	57
DRAW ORDER.....	58
OUT OF VIEW ITEMS.....	58
GROUPED ITEMS.....	59
CLIENT AREA.....	59
HIDING THE FRONT PANEL.....	60
VISIBILITY IN PARENT MODULE PANELS.....	60
PROPERTIES.....	62
ENABLING THE PROPERTIES PANEL.....	62
ADDING PROPERTY ITEMS.....	63
CONTROL TYPES.....	64
EDITING THE PROPERTIES PANEL.....	66
RESIZING.....	66
CUSTOMIZING.....	66
SYNCHRONISING.....	68
PASTE SYNCHRONISE.....	68
SYNCHRONISE ALL.....	69
SYNCHRONISE PAINTER.....	69
REMOVING SYNCHRONISATION.....	70

STREAM DATA.....	73
MONO.....	73
POLY.....	73
WHEN TO USE POLY OR MONO.....	74
STREAM CONNECTORS.....	74

C O N T E N T S

	POLY AND MONO SECTIONS IN AUDIO APPLICATIONS.....	75
	BOOLEAN CONNECTORS.....	76
	POLY INT.....	77
	SSE.....	77
	MONO 4.....	77
	PERFORMANCE.....	77
	TRIGGERED DATA.....	78
	HOW IT WORKS.....	78
	WHAT IT'S USED FOR.....	79
	TRIGGERED DATA TYPES.....	79
	CONVERTING BETWEEN DATA TYPES.....	81
	STRING SHORTCUTS.....	84
CHAPTER 6	6 Exporting	87
	CREATING STANDALONE APPLICATIONS.....	88
	LIBRARY DEPENDENCIES.....	89
CHAPTER 7	7 Advanced GUI Editing	91
	MODULE GUI.....	92
	MODULE GUI COMPONENT.....	92
	MGUI CONNECTORS.....	93
	GUI CONNECTOR TYPES.....	94
	COORDINATE SYSTEM.....	95
	DRAWING	96
	DRAWING ON A PANEL.....	96
	DRAWING ORDER.....	97
	CHAINING GUI COMPONENTS.....	98
	MOUSE HANDLING.....	99
	MOUSE AREA.....	99
	MOUSE CLICKS.....	99
	MOUSE DRAGGING.....	100
	MOUSE MOVES.....	101
	DRAG ACCUMULATE.....	101
	REDRAWING.....	103
	REDRAW CONTROL	103
	PRECISION REDRAWS.....	103
CHAPTER 8	8 Code Component	105
	DSP CODING.....	106
	THE CODE COMPONENT.....	106
	INPUTS AND OUTPUTS.....	106
	SYNTAX COLOURING.....	107

EDITOR.....	108
LOCAL VARIABLES.....	108
ASSIGNMENTS.....	109
EXPRESSIONS.....	111
CONDITIONAL STATEMENTS.....	112
COMMENTS.....	112
ADVANCED FEATURES.....	113
ARRAYS.....	113
HOP.....	114
LOOP.....	114
STAGES.....	115
DEBUGGING.....	117

CHAPTER 9

9 Options 119

THE OPTIONS DIALOG.....	120
APPLICATION.....	121
NAVIGATOR.....	123
SCHEMATIC.....	124
MODULES.....	125
EXPORT.....	126
ADVANCED.....	128

1

Introduction

ABOUT THIS GUIDE AND SOFTWARE OVERVIEW

About This Guide

This manual provides a detailed description of the FlowStone software and its functions. Its purpose is to show you how the software works and what its capabilities are.

If you are looking for tutorials then see the Tutorials section of the DSP Robotics web site:

<http://www.dsrobotics.com/tutorials.html>

For information about individual components and modules that ship with the software, see the Component Reference guide. These documents can be found on the Manuals section of our web site at:

<http://www.dsrobotics.com/manualsarea.php>

Additional information and articles about the software can be found at:

<http://www.dsrobotics.com/support.html>

If you have any comments about this guide please email them to info@dsrobotics.com.

What is FlowStone?

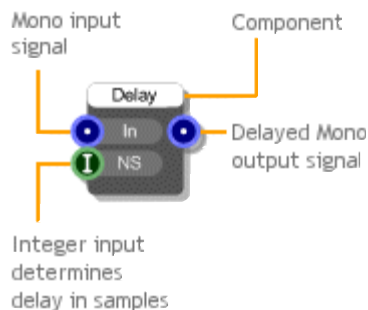
FlowStone is a graphical computer programming language for creating real time Digital Signal Processing (DSP) and Robotics applications. Using the software you have complete flexibility to create exactly the kind of application you want. You can add customised controls to modify parameters in real time and group these controls together to make powerful user interfaces.

Your completed creations can then be exported as completely independent executable applications. These applications can then be used on any PC running Microsoft Windows or used at the centre of your own embedded system.

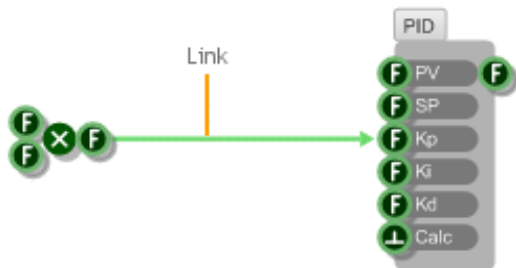
How It Works

Conceptually, FlowStone is very simple. The software provides a set of building blocks called components. Each component performs a different function. A component can take data in, process it, and pass it out. A component can therefore have inputs or outputs or both.

The inputs and outputs are called Connectors. There are different types of connector for different types of data. Each connector has it's own symbol so that you can easily identify the data type.



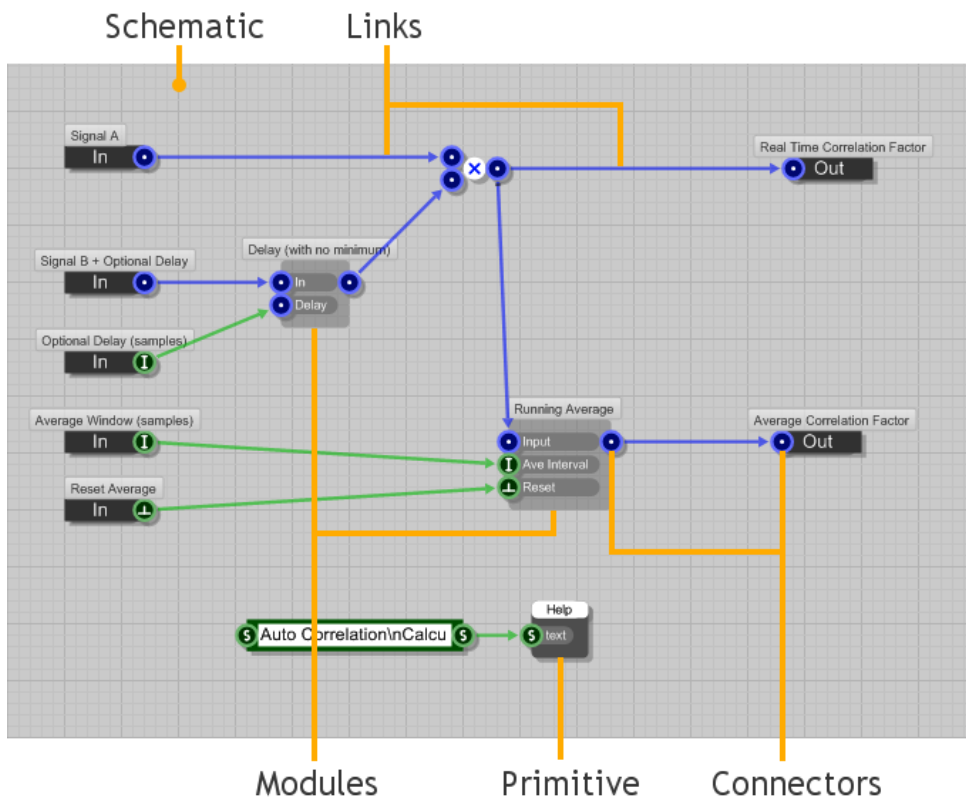
Data is passed between components by connecting them together with links. A link starts at a component output connector and ends at an input connector. In general, data passes through a link from start to end (left to right). However, in some cases, data is also passed from end to start (more on this later).



Components and links are laid out on a 256x256 square grid which we call a schematic.

In order to allow more sophisticated schematics we have a special type of component called a module. Modules are special types of component in that they are defined by their own schematic, containing other components and modules. Modules can also have an interactive front panel with it's own controls and custom graphics.

Any component that is not a module we call a Primitive.



By combining different primitives and modules you can create a vast range of different behaviours and ultimately modules that are individual applications in their own right. These can then be exported as a standalone executable application for use outside of the software.

Digital Signal Processing in a Nutshell

In order to get the most out of the software it's worth knowing a little bit about digital signal processing (DSP) is. Don't worry, there's no need to get into any heavy mathematics. Essentially you just need to know what DSP means.

So lets start with a **signal** which we will define as the continuous value of a some parameter over time . This could be anything from the temperature of a room to audio taken from some listening device.

If the signal is passed into your computer from an external source like a microphone then it needs to be converted from an analogue signal to a digital one. This is done by measuring the magnitude of the signal repeatedly over discrete time intervals. Each measurement is called a sample and the result is a stream of numbers which is the **digital** signal.

The process is called sampling and the rate at which you measure is called the sampling rate. The most common rate used for audio signals is 44100 samples per second, often represented as a frequency 44 Khz but higher rates are also used to get higher quality signals.

The **processing** part of DSP involves taking the stream of numbers that represent the digital signal and converting them into another stream of numbers by applying some combination of mathematical transformations. The signal can then be converted back to analogue and listened to as sound through headphones or speakers.

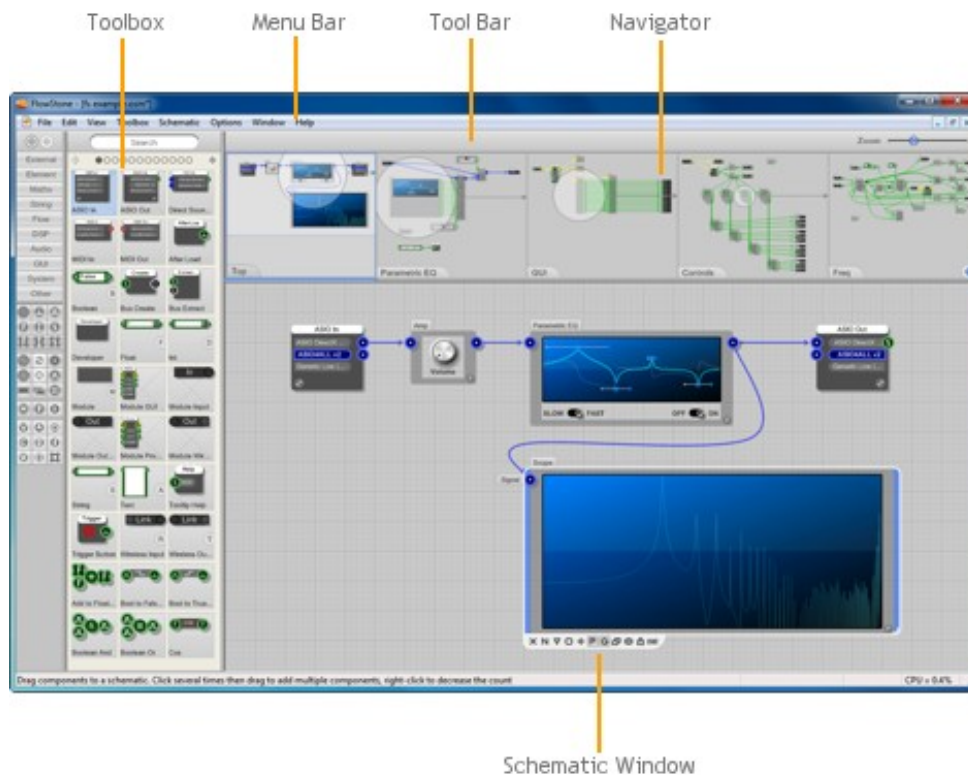
A key feature of the software is that it can process these samples individually at sampling rate. Many applications are not capable of this and have to process a collection of samples together in one frame in order to keep cpu usage at acceptable levels. This restriction limits the processing possibilities as there is often a requirement to process a given sample based on the sample that preceded it. This situation is called feedback and single sample feedback is a capability of FlowStone that sets it apart from other audio software.

2 User Interface

A FIRST LOOK AROUND

Before you can begin using the software you need to know a little bit about the user interface. Let's start by taking a look at the main application window and how it's laid out.

Across the top of the window is the menu bar. All the applications functions can be accessed from here and it's the first place you should look if you're new to the software and you want to get an idea of what you can do with it. As your mouse passes over menu items help text is displayed on the status bar at the bottom of the application window.



You won't stay with the menu bar for very long though. In FlowStone there are usually several ways to execute the same action so in time you'll find yourself using direct interaction, context menus or shortcut keys instead.

Running down the left-hand side of the application is the **Toolbox**, which provides all the components for building your schematics.

Across the top of the window is the **Navigator**. As it's name suggests, this is used for navigating through your schematic. You'll find this invaluable once your schematics start to become more complex.

The majority of the application workspace is taken up by the **Schematic Window**. This is where your schematics are created.

Toolbox

The toolbox provides the building blocks for a schematic – the components. There are already over 300 components to choose from and they are growing in number all the time.

Of course if there are hundreds of components then you'll want to be able to put your hands on the component you're after quickly and with little effort. Thankfully there are several mechanisms in place to make this exceptionally easy.

Component Browser

Exploring

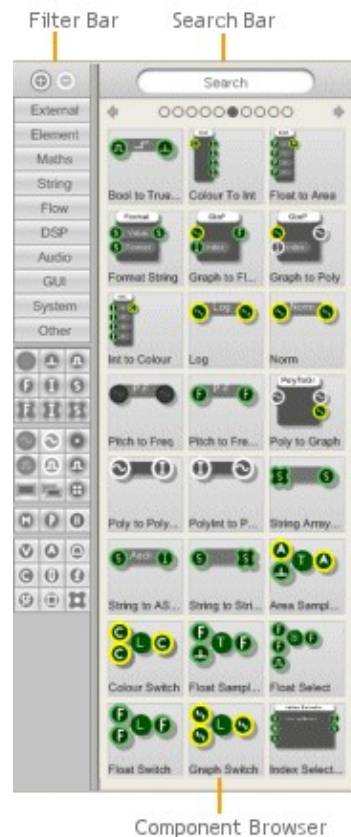
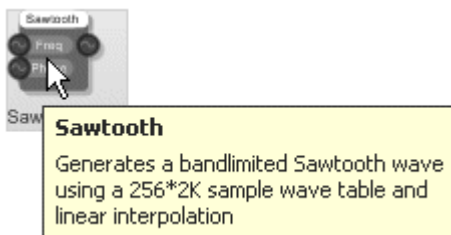
The main part of the toolbox is the component browser. This shows every component and its name. You can scroll through the components by clicking on the page buttons.



You can also scroll by clicking on the arrows to the left and right of the page buttons or by pressing the PGUP and PGDN keys.

Instant Help

You can get help for a component by hovering your mouse over it. A pop-up window will appear giving a short outline of the purpose of the component.





Filter Bar

Groups

The filter bar provides two simple but effective ways to find the components you want. At the top of the filter bar are the filter groups. Each component can appear in at most one group.

Click on a group and the toolbox will immediately filter out all the components that are not in that group.

You can create your own groups by clicking on the  button at the top of the filter bar. User defined groups are listed in their own section below all the other groups. You can delete groups you have created by pressing the  on. Modules inside a deleted group are NOT DELETED but moved to Other instead.

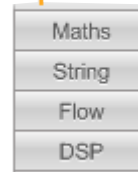
Types

Below the filter groups are the filter types. You'll see a button for each connector type here. Click on one of these and the toolbox will change to show only components which have a connector of the selected type.

You can apply more than one type filter at the same time. To do this, hold CTRL while clicking. This will show components that use at least one of each of the selected types. So if you select Int and Float only components that have Int AND Float connectors will be shown.

Alternatively, if you hold down SHIFT while clicking, you'll get components that have one type or the other or both. This time if you select Int and Float components that have Int or Float connectors will be shown.

Filter Groups



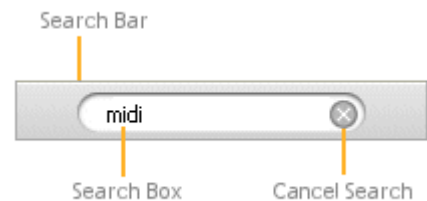
Filter Types

Search Bar

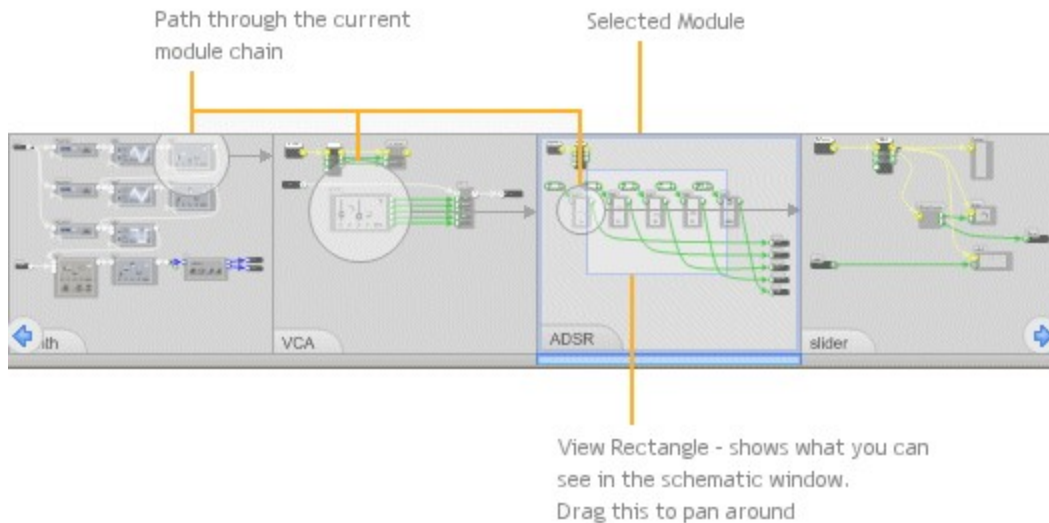
The Search Bar appears at the top of the toolbox and provides a quick way to go straight to the component you want. Just type some text in the box and as you type the components will reduce to show only those that match.

You can type the name of a component, a filter group or a connector type - all these are recognised. For example, if you type "midi" you'll get any component with "midi" in its name and also any components with MIDI connectors.

You can quickly jump to the search box at any time by pressing CTRL+F



Navigator



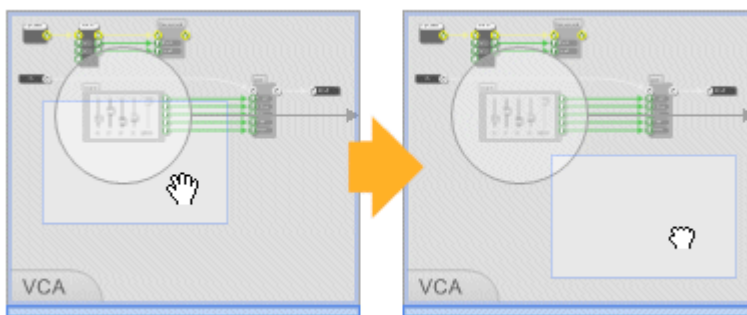
The Navigator allows you to see where you are in your schematic. This is extremely useful when you have several layers of modules because the Schematic Window only shows the module that you're editing.

Jumping

The Navigator is not just visual, it's interactive too. You can jump to any module in the current chain by clicking on it. You can also use the number keys to do this (providing you are not using the PC keyboard for MIDI input). Number 1 will always take you to the Top level. The PGUP and PGDN keys will move you up and down the current hierarchy.

Panning

The current module in the chain is highlighted. Inside you'll see a rectangle representing the portion of the module that you can see in the schematic window. This is called the View Rectangle. You can click and drag this to pan around the schematic.

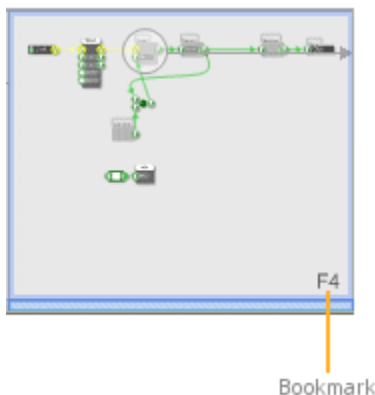


Bookmarks

You can bookmark particular parts of your schematic so that you can easily return to them. To do this simply go to the part of your schematic you want to bookmark, hold SHIFT and press one of the function keys (F1...F15). The Navigator will show the associated function key.

To return to the bookmark at a later time just press the function key. You can also then flick back to the point that you came from by pressing the function key one more time.

To remove a bookmark all you need to do is jump to it then hold SHIFT and press the same function key again.



Schematic Window

The schematic window is where everything comes together. Components can be dragged here from the toolbox. You can connect components by dragging links between them.

The schematic window has all the features that you'd expect in an editor: undo, copy and paste, multiple selection, zooming, and context sensitive help are all fully supported.

Zooming

The Schematic Window is fully zoomable. There are several ways you can zoom the schematic window:

Mouse Wheel

The easiest way to zoom is to use the mouse wheel. Roll the mouse forward to zoom in and back towards you to zoom out.

When mouse wheel zooming, the software will zoom towards or away from the point where you position your mouse. This allows you to zoom and pan in one movement.

If your mouse supports it, you can also return the schematic to the default zoom level by pressing the mouse wheel button. The schematic pan position is also returned to its default.

Zoom Slider

The Zoom slider is located on the right-hand side of the tool bar. Moving the slider to the left will zoom out and to the right will zoom in.



The default zoom position is shown as a small notch above the slider. The slider will snap to this position as you move past to make it easy to return to the default. The default zoom level can also be achieved by double-clicking on the slider.

Context Menu

Right-click on an empty part of the schematic and select Zoom In or Zoom Out

Keyboard

Simply press the + (zoom in) or - (zoom out) keys.

Panning

In addition to using the Navigator, you can quickly pan around the schematic window by grabbing the background and moving it using your mouse. To do this:

1. Left-click on an empty part of the schematic and hold the mouse button down.
2. Now right-click and hold that button down too. The cursor will change to show a hand grabbing the schematic.
3. Now move the mouse and the schematic will move with it. Release both mouse buttons to finish dragging.

You can also pan by holding the space bar down. The cursor will change to the hand. Keep the space bar down and drag the schematic around.

3 Components & Links

THE BASIC ELEMENTS OF A SCHEMATIC

Components and links are the bricks and mortar of a FlowStone schematic. Understanding how to edit and manipulate these basic elements is essential for working with the software as most of the interaction you'll have with FlowStone will be through the schematic.

We introduced the concept of components and links right at the start of this guide. We'll go over this again now but in a little more detail this time.

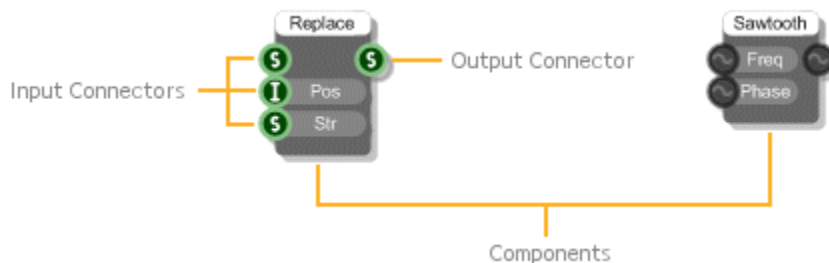
Components

Components provide the functionality in a schematic. Each one performs some well defined task. They are represented by rectangular blocks with circular adornments called Connectors which may appear on the left-hand or right-hand sides (or both).



The connectors on the left-hand side are called Input Connectors. These provide the component with information that it uses when performing it's defined task.

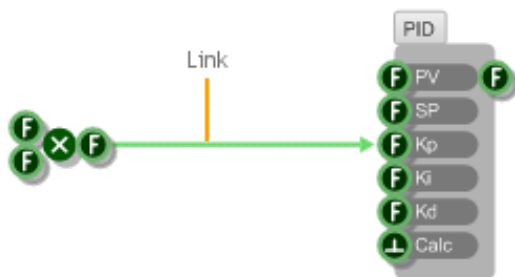
The connectors on the right-hand side are called Output Connectors. These provide information about the outcome of the of the task performed by the component.



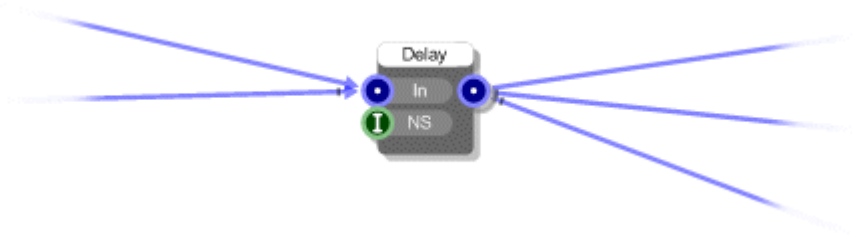
In the example above the String Replace component takes 3 inputs: the original text string, the position at which the replacement is to be made and the text string to be inserted. The output is the modified string.

Links

Links define how information flows between components in a schematic. A link passes from the output connector of one component to the input connector of another component. The direction of information flow is generally left to right or from output to input but in some cases information can pass from input to output as well.



You can have multiple links from the same output connector. You can also have multiple links passing to the same input connector. However, you can only have one link going from the same output connector to the same input connector.



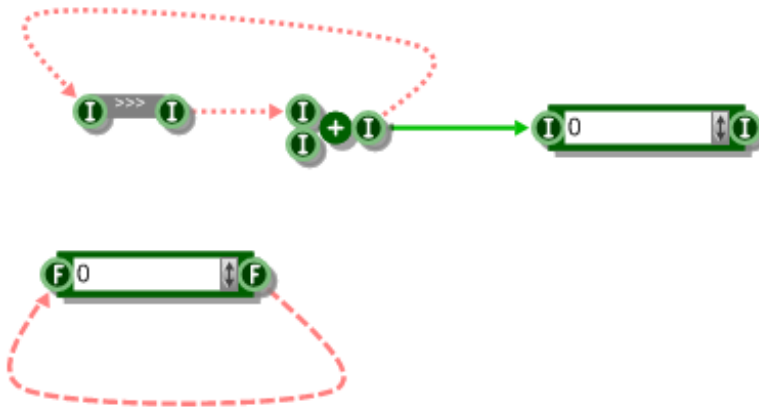
When multiple links arrive at the same input connector then, for some connector types, the data from each link is added together. In other cases the data is just merged.

Infinite Feedback

Links can be connected from a component into itself to create feedback paths. However, in some circumstances this may create an infinite feedback loop. If this happens the software will 'freeze' the affected link(s). A frozen link is shown as a red dashed or dotted line.

Frozen links continue to function but data flow through them is reduced (but not restricted altogether) to prevent the software locking up.

[Note that links between stream, poly or mono connectors never become frozen as feedback is always allowed in these cases.]



A dashed line indicates that the infinite loop is in the forward direction ie. left to right as changes are triggered. A dotted line indicates that the direction is backwards ie. right to left as data is being gathered.

Frozen links are relatively rare. However, when they do occur they can be easily located because every module that contains a frozen link somewhere below it is highlighted in red with a dashed red border. Here are two such examples:



If you follow the red modules down the hierarchy you'll eventually get to the frozen link(s).

Handling Frozen Links

So how do you deal with frozen links once you've found them?

You may be able to get the behaviour you were wanting to achieve by inserting a Trigger Blocker or by using a Sample and Hold component that triggers from somewhere else.

It could be that the feedback may only be required in one direction at a time so you could use select components to make sure that data is only sent in one direction or the other.

Some components have built in handling to prevent infinite feedback from occurring. The Switch components for example (Float Switch, String Switch etc.) have this kind of behaviour. Inserting one in between frozen links can often solve the problem.

Sometimes a frozen link can simply be removed and the same functionality is retained. The link may have been added as overkill or in error.

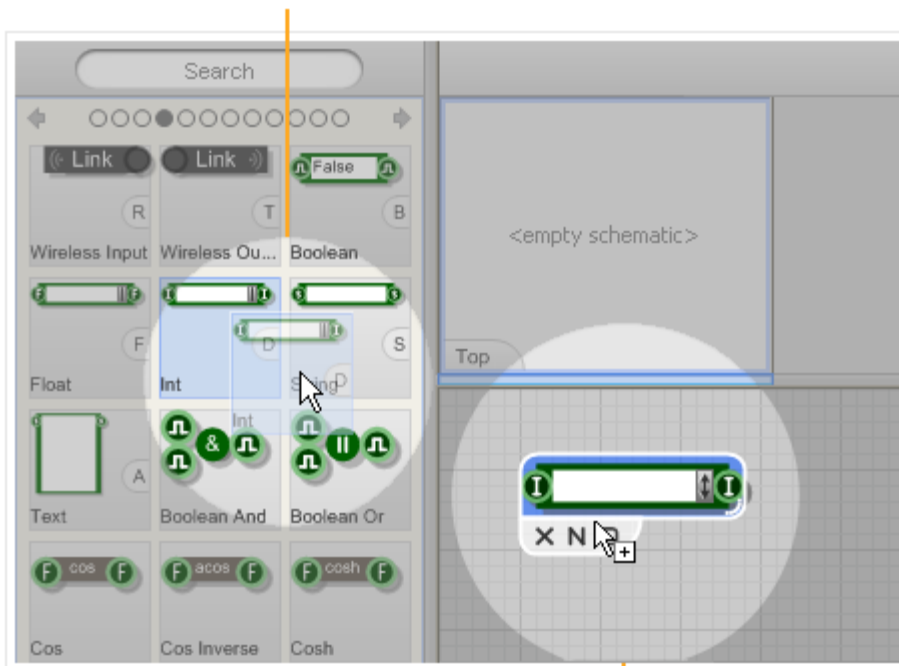
Components

Adding a Component

Dragging From the Toolbox

To add a component to a schematic, simply go to the toolbox, pick the component you want and drag it into your schematic.

1. Click on the component you want then hold and drag across to the schematic window



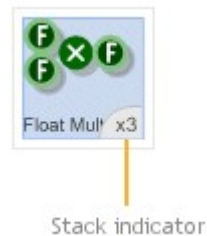
2. Release the mouse button to drop the component into the schematic

Stacking Components

Sometimes you may want to add more than one component at a time. You can do this by stacking up components before dragging them all to the schematic.

To increase the number in the stack just click on the component the same number of times as the number you require. You'll see a counter appear in the bottom-left corner of the component in the toolbox.

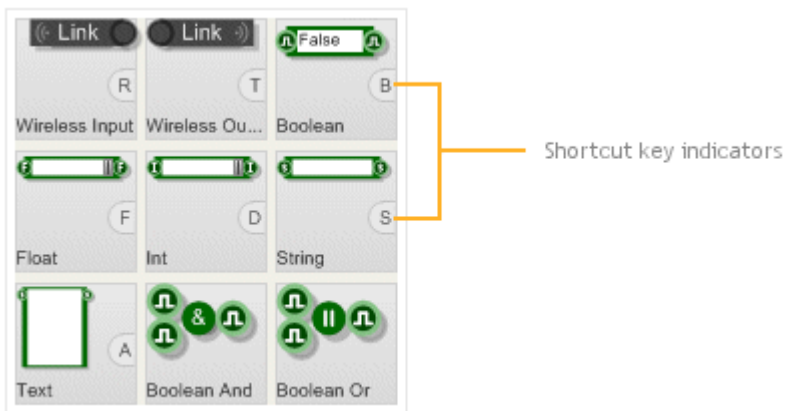
To decrease the number in the stack right-click on the component. When you have the number you need just drag them across as before.



Keyboard Shortcuts

As you become familiar with the software you'll find that there is a certain small group of components that you use much more than others. In order to save you time constantly going back and forth to the toolbox you can make use of keyboard shortcuts.

Where a component has a shortcut key assigned this is displayed on the component in the toolbox.



Disabled Components

On some occasions it is not possible for you to add a particular component to your schematic. This is because some components are only allowed once within a particular module or the whole schematic. This can also occur if you're running the Free or Enterprise editions and you reach one of the limits for components of a particular type.

If this happens the component will have a grey cross over the top of it.



Cross indicates that the component cannot currently be added to the schematic

Hitting Return

One final way to add components from the toolbox is to use the return Key on the keyboard. This is handy if you've just done a search say. Hitting return when in the toolbox (or the search bar) will always add the currently selected component to the schematic.

Selections and the Action Panel



To select a component simply click on it. When you do this you'll see the action panel. This contains a number of buttons. the 'X' button will delete the component. The 'N' button will allow you to name the component.

Other buttons may appear depending on the type of component selected - more on these later.



The commands in the action panel can also be invoked from the menu bar and the context menu. To get the context menu, right-click on the component.

Note that as well as the selected component(s), any links that start and end in a component that's selected will also be considered part of the selection

Moving Components

To move a component:

1. Move your mouse over it. The cursor will change to Move/Select.
2. Click and hold the mouse button. the cursor will change to Move.
3. Drag to the desired position and release the mouse button.



1. Move mouse over

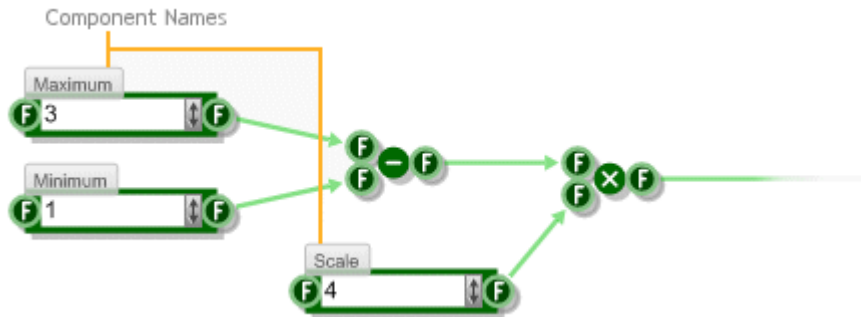


2. Click, hold and drag

For fine movements you can use the nudge feature. With a component selected use the cursor keys to move it up, down, left or right by one grid square.

Naming Components

You can give a component a name. This is just a label that can be used to remind you the role of a component in your schematic.

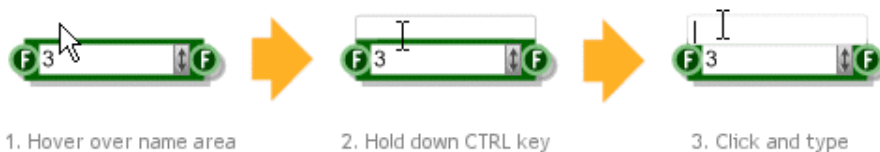


To give a component a name, click the 'N' button on the action panel or right-click on the component and select Rename. Alternatively you can select the component and press CTRL+R.

An edit box will appear above the component. Type the name here then press ENTER, TAB or just click on another part of your schematic. You can press ESC at any time to cancel.

Direct Interaction

For a more interactive way to add names, hover the mouse over the area just above the component and hold CTRL. The frame of the name box will be highlighted. Click in the box and proceed as before.



Editing Names

To edit a name you can use the action panel and right-click options as before or you can just click on it as shown below.



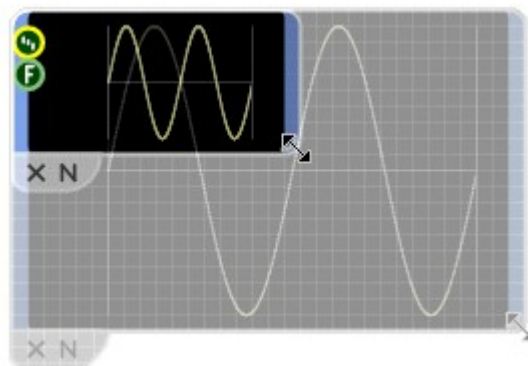
Deleting Components

Deleting a component is extremely easy. The quickest way to do this is to select it then press the DEL key. You can also press the 'X' button on the action panel or right-click on the component and select Delete.

Resizing

Some components can be resized to make them bigger or smaller. Some components only resize horizontally, others resize vertically as well.

If a component can be resized, the resize control will appear in the bottom-right corner of the selection highlight. The control is a white arc that traces the corner of the selection.



To
resize,
move

your mouse over the resize control. The mouse pointer will change to the resize cursor. Click and hold then drag to resize.

Multiple Selections

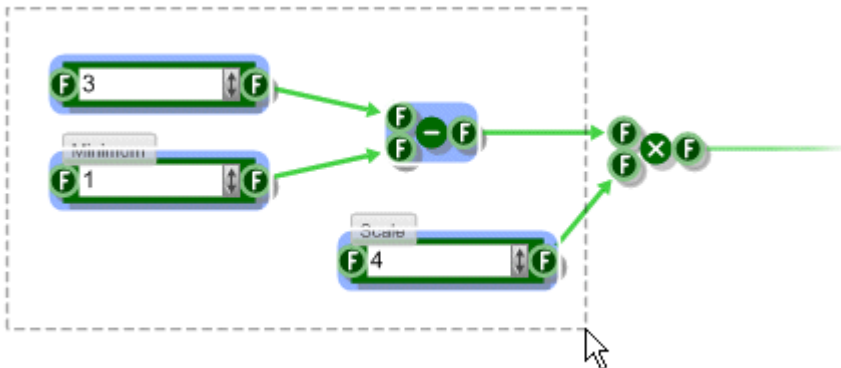
You can select multiple components at the same time. This is useful if you want to move a group of components but maintain their relative spacing or if you want to delete a whole load of things in one go.

There are two ways to create a multiple selection. The first way is to hold down SHIFT and then click on each of the components in turn. If a component is already selected, clicking on it will remove it from the selection.

A quicker way to make a multiple selection is to drag select. This involves dragging out a rectangle to enclose all the components that you want to select.

To drag select:

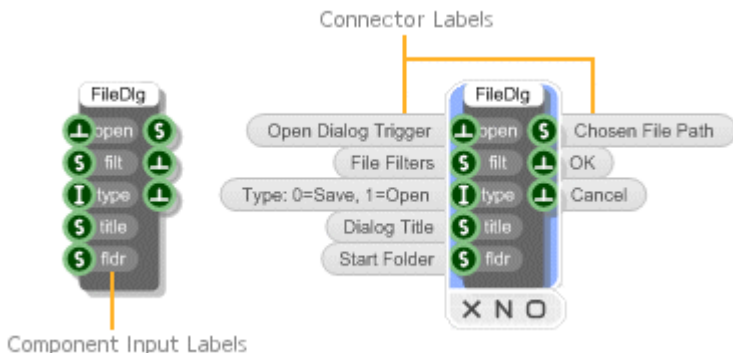
1. Click on a blank part of the schematic to the top-left of the components you want to select.
2. Holding the mouse down, move the mouse down and to the right. You'll see a dotted rectangle appear as you move.
3. The components will show as selected as you move the mouse so when the selection is what you want release the button.



Connector Labels

Most components have a short label for each of their input connectors. There isn't enough room on the component body for both inputs and output labels. The outputs are usually less ambiguous and fewer in number and so input labels are usually preferred.

Component labels are usually very short and provide a quick reminder of what each connector is for. If you need more information you can select the component (by clicking on it). Where available, additional connector labels will appear to the left and right of the component.



Cut, Copy and Paste

FlowStone supports the standard Cut, Copy and Paste operations for moving or duplicating parts of a schematic. Both single and multiple selections can be cut, copied and pasted.

You can access these operations from the Edit menu or by right-clicking on a selection in the case of Cut or Copy or anywhere on your schematic in the case of Paste.

Copying a selection will place a duplicate of the components and links on the Clipboard. The clipboard is an invisible buffer that retains what was last copied to it. Cutting is like a combined copy and delete as the original selection is removed from the schematic.

Having Cut or Copied a selection you can then go to any other part of your schematic or to any other schematic you have open and paste in a copy of the clipboard contents. Copies are pasted at the current mouse location. You can hold ALT while pasting to have pasted elements stack up at the location of the source elements.

Links

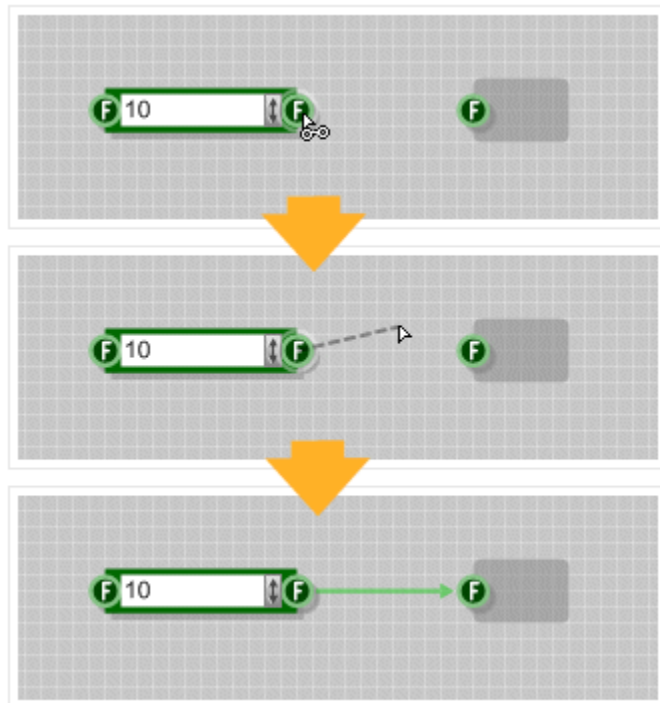
In this section you'll learn how to move a link to another connector or delete it completely. You'll also see how to change link order and how to bend a link.

Creating a Link

Links must start at an input connector and end at an output connector. You can link another component or to itself, to create a feedback path say.

To create a link:

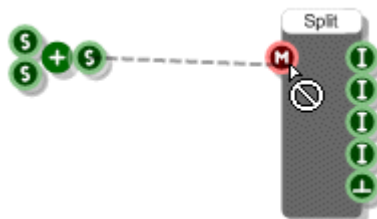
1. Move your mouse over an output connector. The connector highlight will show as you pass over it.
2. Click and hold, then drag towards the input connector. You'll see a dotted line representing the potential link.
3. Keep dragging until your mouse passes over the input connector. The potential link will snap to the connector to indicate that the link can be formed.
4. Release the mouse button to create the link.



Allowed Links

Sometimes it is not possible to make a link between a pair of connectors. When this happens the mouse cursor will change to show this.

In general links can be made only between connectors of the same type. However, there are several exceptions to this. For example, Floats and Ints can be connected so can Floats and Stream connectors. A complete description of all these exceptions can be found in Chapter 5 Converting Between Data Types on page 81.



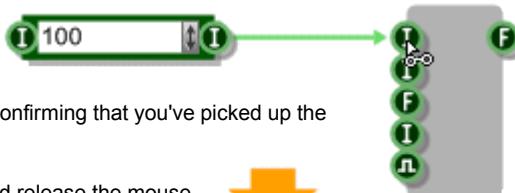
Moving a Link

You can move the end of a link from the input connector to another input connector on the same component. You can also move it to an input connector on a completely different component.

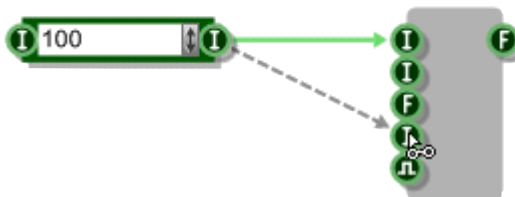
To move a link:

1. Move the mouse pointer over the input connector where the link ends. The cursor will change to the linking pointer.
2. Click and hold. You'll see a dotted outline confirming that you've picked up the link that you wanted.
3. Drag the link to another input connector and release the mouse button.

1. Move over connector and click

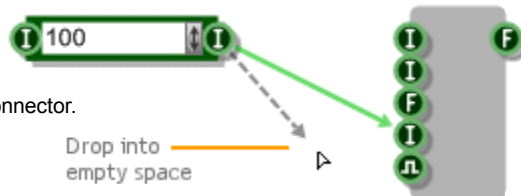


2. Drag and drop

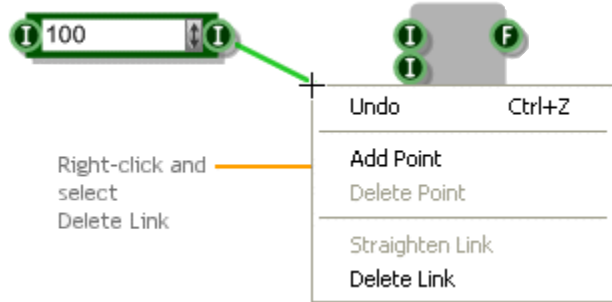


Deleting a Link

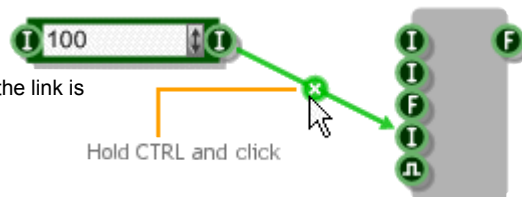
There are several ways to delete a link. You can pick it up and drop it into empty space. Just move it from the input connector (as described above) but don't link it to another connector.



You can also right-click on the link and select Delete Link from the context menu.



For really quick deleting, hold down CTRL. When your mouse passes over a link, the delete button will appear. Click the button and the link is gone.



Link Order

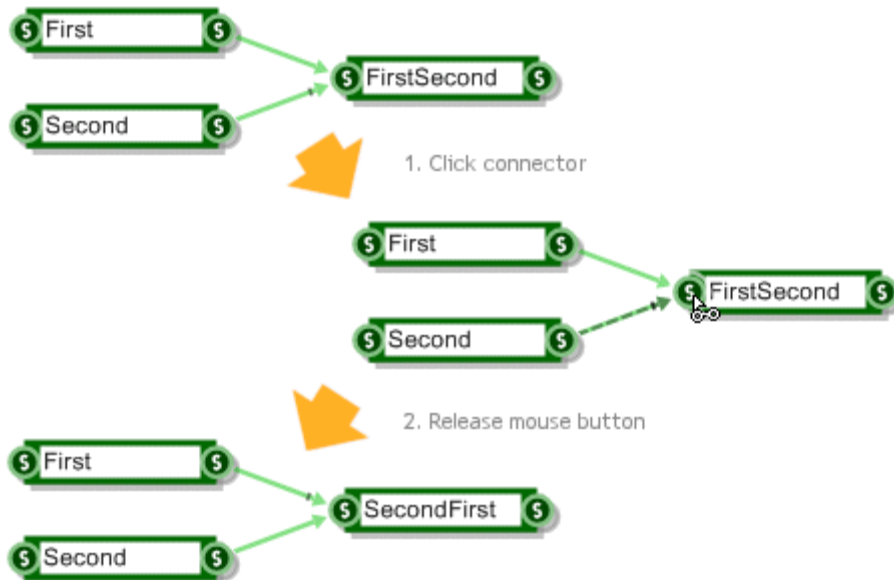
When there are many links ending at the same input connector, the order of the links can affect the behaviour of your schematic. For example, if you have two links that are passing String data to a string connector, the first string to arrive will have the second one appended to it.

The order of a link is indicated by an order marker. This is a series of notches perpendicular to the link at it's end point. The first link in the order has no marker, the second has a marker with one notch, the third has two notches etc.



Changing the Order

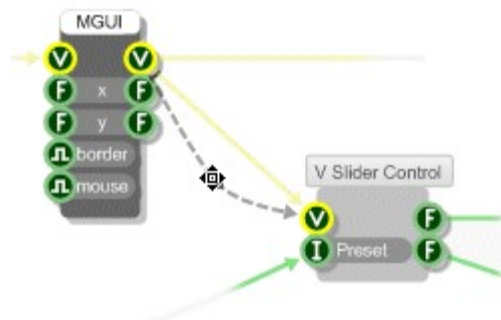
It's useful to be able to change the order of links at an input connector. To do this simply click on the connector and the current top link (the last in the order) will become the bottom link (the first in the order).



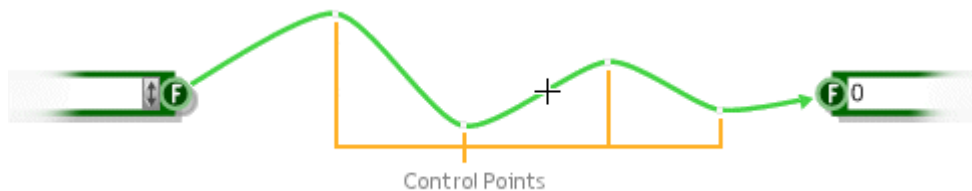
Bending Links

In a complex schematic with many components, sometimes a straight line just won't do. Often you'll struggle to avoid creating links that run over components or each other.

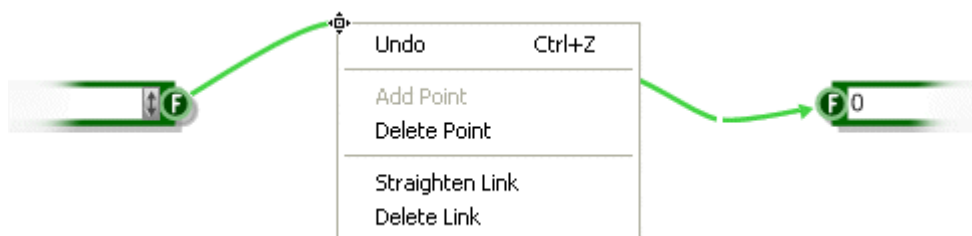
To get round this problem, you can bend any link to make your schematic clearer and easier to read. All you have to do is click on the link and pull it into place.



Each bend you put in a link creates a control point. The control points define the path of the link. If you move your mouse over a link it becomes highlighted and the control points are shown as small squares. The control points will either be black or white depending on the colour of the link.



You can drag the control points around to re-shape the link. You can also delete a control point by right-clicking on a point and selecting Delete Point from the context menu.



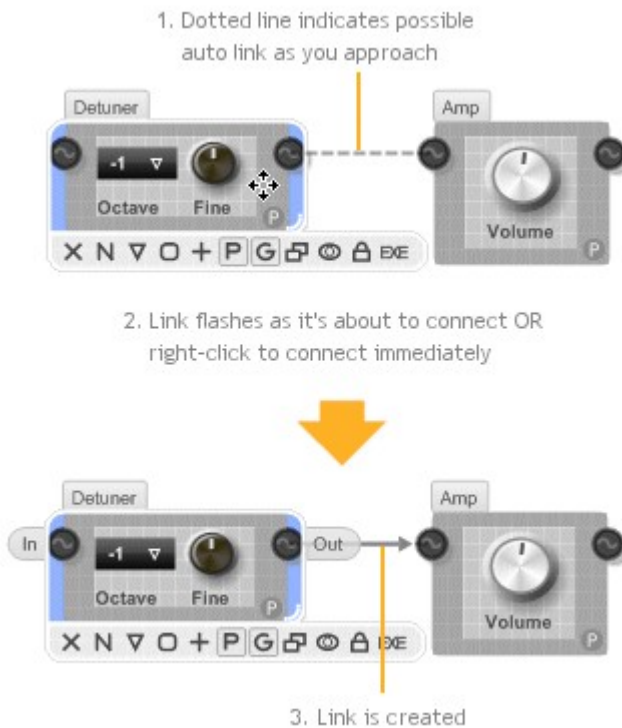
To remove all the control points and make the link a straight line again, select Straighten Link from the context menu.

Auto Linking

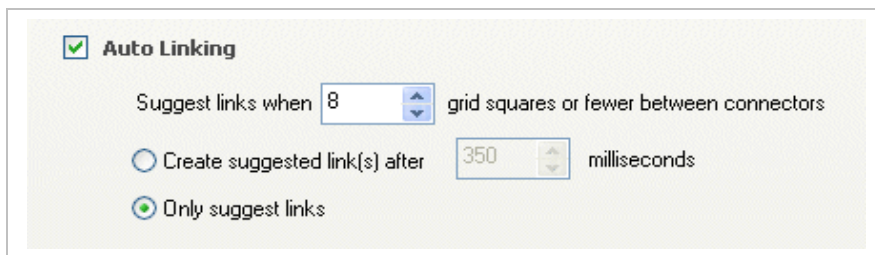
To help you build your schematic more quickly we have the Automatic Linking feature. When this feature is switched on you can create a link between two connectors by moving a component so that the required output connector is on the same horizontal and sufficiently close to the required input connector.

When these conditions are satisfied a dotted line will appear. This suggests where the link would be created. If you then pause for a fraction of a second the link will flash and become permanent.

If you don't want to have to wait for the link to be created you can instantly accept the suggested link by clicking the right mouse button.



There are several options for Auto Linking. These can be adjusted by selecting Schematic from the Options menu.



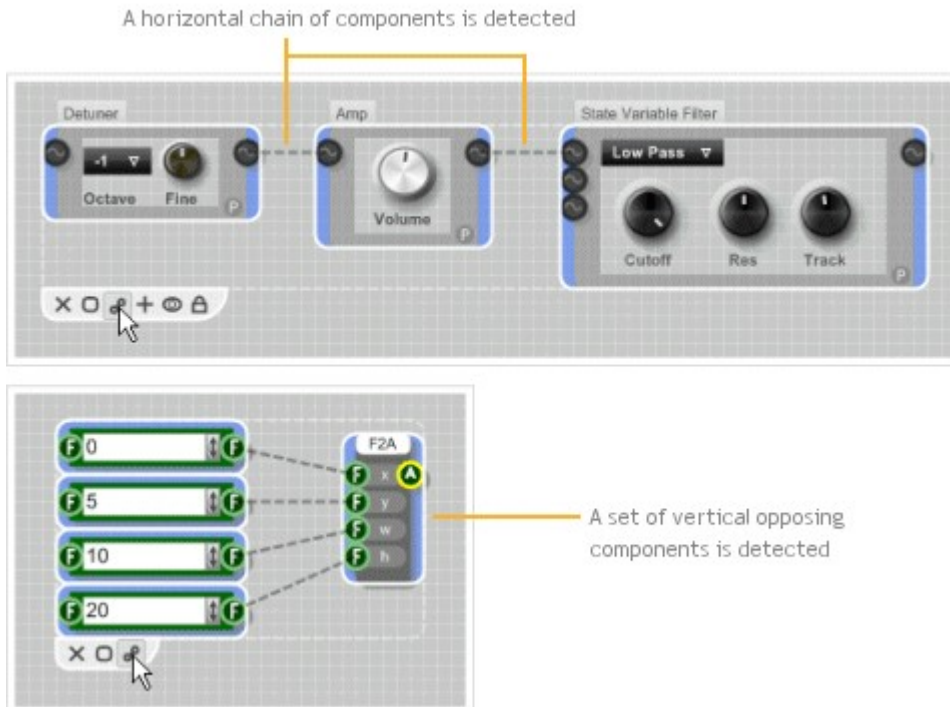
You can change the time between suggestion and link creation you can also set how close connectors have to be before a link is suggested. If you only want to use the right-click method to make a link then you can set the auto linking to only suggest links and of course you can turn the feature off altogether.

Smart Linking

If you need to link several components together quickly then you can use the smart linking feature. Simply position your components in a logical way, select them then either click the Link Components button on the action panel or select Link Components from the Schematic Menu.

The software will make a best guess at how you want the components to be linked together. If you use the action panel button to do this then when you hover the mouse pointer over the button the software will indicate the links that would be created.

Smart linking is useful if you have several components that you want to link to one other component or if you want to chain components together.



Removing Multiple Links

You can remove all the links in a selection in one go. Simply right-click on the selection and choose Remove Links from the pop-up menu.

This can be very useful when you have a large section of schematic that you need to disconnect

Undo	Ctrl+Z
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Paste Synchronise	Shift+Ctrl+V
Rename	Ctrl+R
Delete	Del
Remove Links	Shift+Ctrl+L
Make Module	Ctrl+M
Property	Shift+Ctrl+P
Synchronise Painter	

Wireless Links

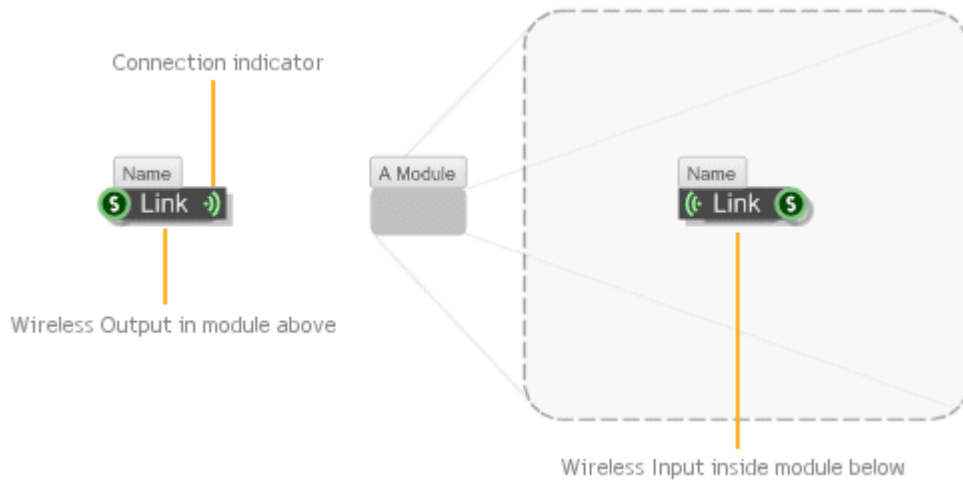
Wireless links provide a mechanism for passing data through the module hierarchy without having to create any physical link. They are just like wireless networks in the real world. You have a transmitter called a Wireless Output at one end and a receiver, a Wireless Input at the other end.



A connection is established between a Wireless Output and a Wireless Input only if the following three conditions are met:

1. The Wireless Input must appear in a module **below** the Wireless Output in the hierarchy
2. The Wireless Input and Output must have the **same label**
3. The Wireless Input and Output must have the **same connector type**

When a link is established the connection indicators on the Wireless Input and Output will light up.



Wireless links only work down the module hierarchy, you can't link back upwards. Also, the range of a wireless output only extends as far as the next wireless output below it which has the same label and connector type.

The same wireless output can connect to multiple wireless inputs and vice-versa so long as they conform to the 3 criteria described above.

There is another wireless component called a Module Wireless Output. This is used to make a wireless module. See the Modules section for more information about this.



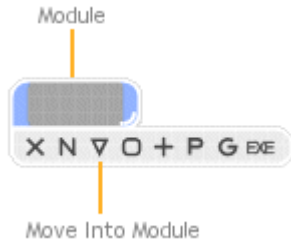
4 Modules

MANAGING COMPLEXITY

You may recall from the introduction that components are broken into two types: primitives and modules. A primitive has a predefined behaviour. It's a black box whose behaviour can't be changed.

A module on the other hand has its behaviour defined by a schematic. You can add inputs and outputs and the internal behaviour can be modified to do virtually anything you want.

The module component can be found in the toolbox under the Module filter group. Drag one into a schematic and you'll see that an empty module is just a grey box.



The action panel for a module has addition buttons to reflect the additional operations that you can perform on it. Of these the most important is the Move Into Module button (represented as an inverted triangle). By pressing this button you can go inside a module to view its schematic and from there define the module's behaviour.

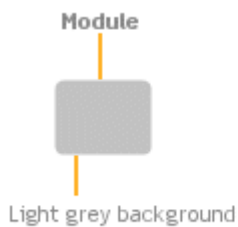
Modules are very simple in concept but they are extremely powerful. They can be used to partition off functionality into pieces that can be reused again and again. They are essential for managing complexity in anything but the very simplest of schematics.

Key Differences

Apart from the fact that a module has its own schematic there are a few other key differences between Modules and Primitives. This section outlines those differences. More details are given in subsequent sections.

Appearance

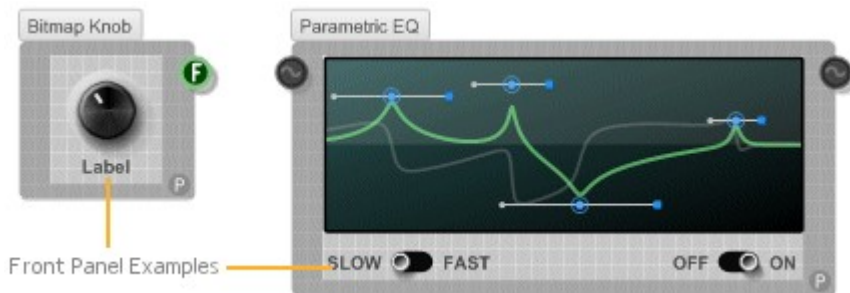
First off modules can be differentiated from Primitives by the way they look. A module has a light grey background whereas a primitive has a dark grey border with a drop shadow and title bar.



A module can also have additional adornments indicating syncing, properties or wireless capabilities. More on these later.

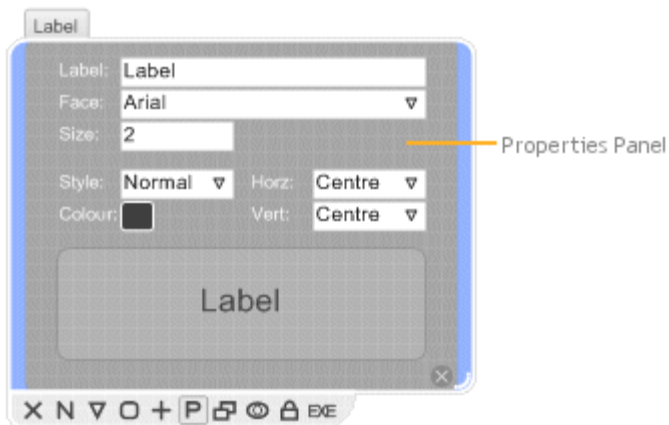
Front Panel

Any module can have a front panel. This is shown on the module itself and provides an interactive surface for adding controls or displaying graphics. The front panel graphics and interaction is all defined in the module's schematic.



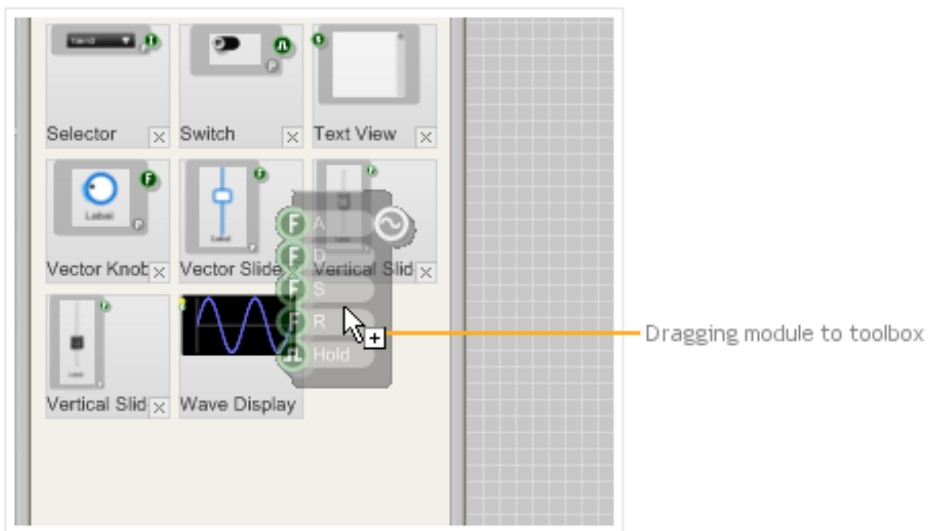
Properties

If you want to control the behaviour of a module in a more interactive manner then instead of using input connectors you can define a properties panel. Again the properties panel is defined by the module's schematic.



Toolbox

The final key difference between a module and a primitive is that a module can be dragged to the toolbox for use at a later date. You can drag a module on it's own or you can drag all selected modules in one go.



When you do this the module will become associated with whatever Filter Group you have selected at the time. If no group is selected then the module will appear in the group called Other.

Basic Operations

You can do everything to a module that you can do to a primitive. You can delete it, name it, move it around etc. The key difference between a module and a component is that you can go into a module and view or edit its schematic.

Moving into a Module

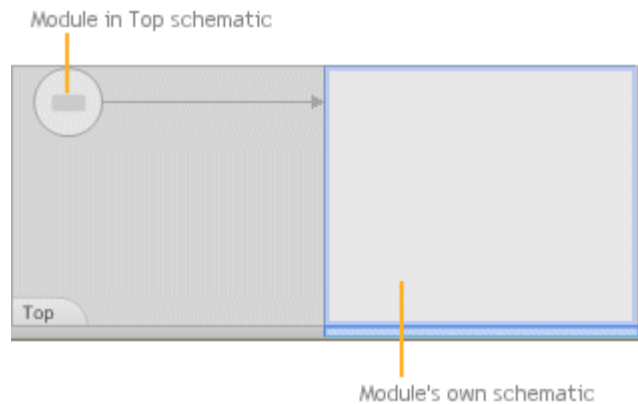
With most operations in FlowStone there are many ways to do the same thing. Moving into a module is no exception.

To move into a module, either:

1. Click the Move into Module button on the action panel
2. Double-click on the module
3. Right-click on the module and select Move into Module from the context menu
4. With the module selected, press the PGUP key

The Schematic Window will change to show the module's own schematic, the Navigator will also change to reflect this.

To move back out of the module again, either right-click and select Move to Parent or press the PGDN key. You can also double-click on an empty part of the schematic or of course use the Navigator.

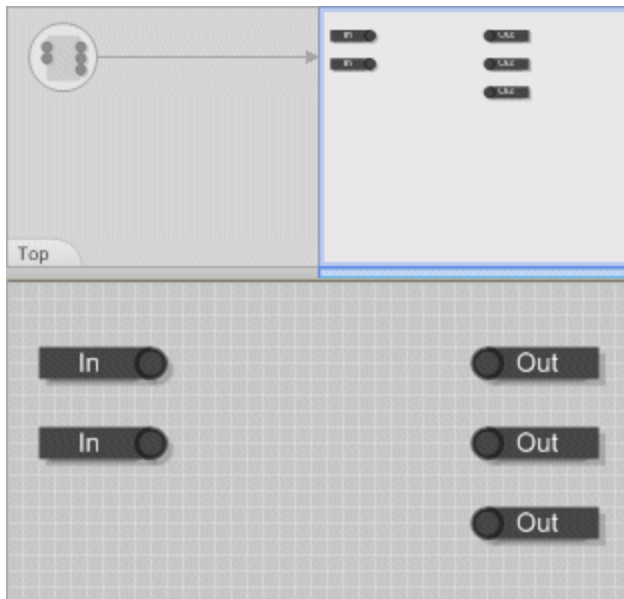


Inputs and Outputs

To get information into and out of a module you need to add inputs and outputs. To do this, just drag them in from toolbox in the usual way (you'll find them under the Module group). The example opposite shows a module with two inputs and three outputs.

For a quick way to add Module Inputs and Module Output components you can use keyboard shortcuts. A select few components are used much more frequently than the others. To save you going to the toolbox each time you can just press a particular key and a new component is dropped at your mouse position.

To add an input press 'I', to add an output press 'O'. Note that these shortcuts will not work if you are using your PC keyboard for MIDI input.



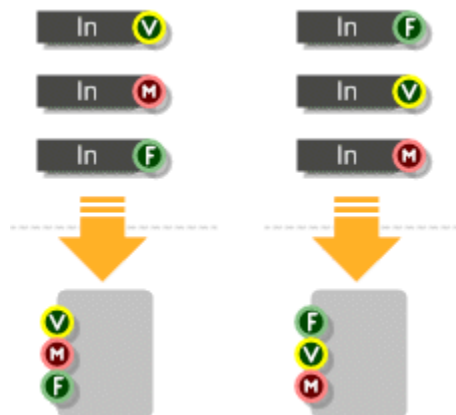
Input/Output Ordering

It's usual to keep the inputs on the left and the outputs on the right but it's up to you.

What does matter is the position of inputs relative to one another as this determines the order of the inputs on the module itself. By swapping the vertical positions of the input and output components you can easily change the order of inputs for a module.

If components have the same vertical position the horizontal position is taken into account, with the left-most input being higher in the order.

The same rules apply to the outputs.



Template Connectors

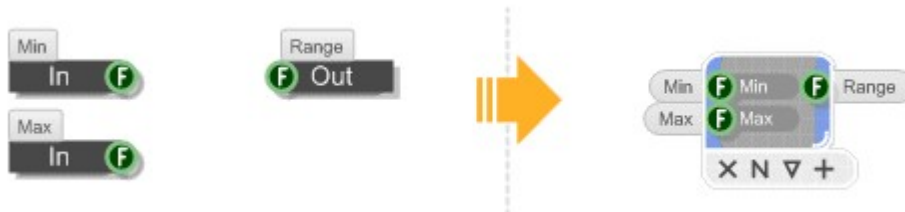
The connectors for the Module Input and Module Output components are what we call Template Connectors. These have no defined type. There are two ways you can assign a type to a template connector:

1. Automatically pick up the type from another connector by creating a link to that connector.
2. Set the type explicitly by right-clicking on the connector and choosing the type from the context menu.



Input and Output Names

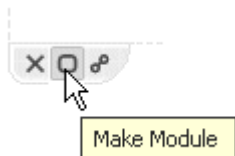
If you give your inputs and outputs names then these will be displayed when you select your module. In addition, input names will be automatically displayed on the module body itself.



If you want to be even more complete you can provide both short and long descriptions. To do this you need to name the module input or output component in the form `<long name>\n<short name>`. So for example, the label "Maximum height of rectangle\nmaxh" would give an input or output a short name of "maxh" and a long name of "Maximum height of rectangle".

Make Module

You'll find that as you build a schematic you'll want and need to section parts off into separate modules in order to manage the increasing complexity. You can do this easily using the Make Module feature. Simply select a number of components then click the Make Module button on the selection action panel (or right-click on the selection and select Make Module).



The software will create a new module and place the selected components and links inside (the layout of all components is preserved). A module input or output will be created for each link that connects to the selection. The old links are then connected to the inputs and outputs on the new module instead of the selection.

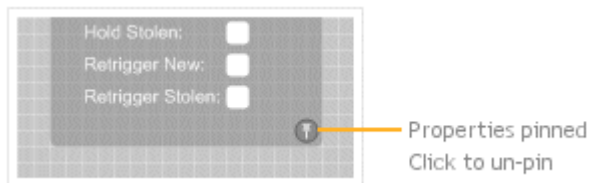
Properties

We mentioned earlier that modules can have properties. These affect how the module looks or behaves. When a module has properties the properties button will be shown in the bottom-right corner of the module. This is in the form of a small circle with a letter P in the centre.



When you click the properties button the module will expand to show the properties panel. With the panel open you are free to make any changes you want. The panel will then remain open until you click on some other part of your schematic.

If you want to make the properties panel display permanently you can pin it open by holding CTRL as you click the P button. The P will change to a pin icon to show that the properties will stay open.



Wireless Modules

Most modules will have fixed output connectors that you physically link up to other connectors. However, it is sometimes useful to make a module output wireless. Instead of using a Module Output component you use a Module Wireless Output component.



By adding wireless outputs to your module the module becomes a wireless module. The module will behave in the same way as a Wireless Output component establishing wireless links with matching Wireless Input components lower down in the module hierarchy.

As with Wireless Outputs a match is determined by the type of connector and the component label.

Wireless modules can be identified by the wireless symbol which appears on the module body. This will appear grey when no links have been established. However, if one or more Module Wireless Outputs within the module have established connections with matching Wireless Inputs the wireless symbol will light up.



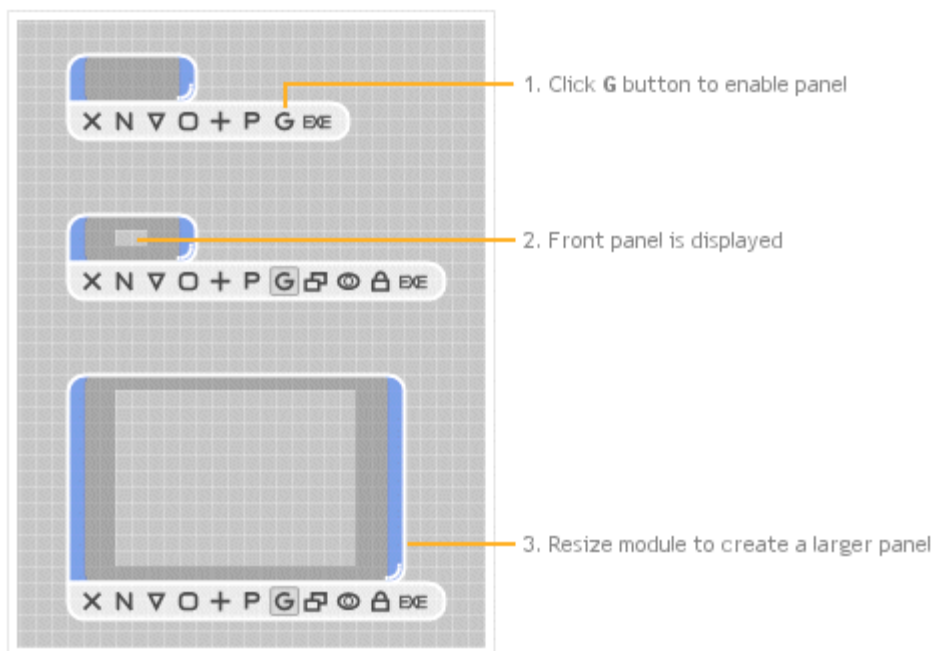
Front Panel

Every module has the option of having a front panel. Front panels allow you to add interactive elements to your schematic and they are the mechanism by which you provide a graphical user interface (GUI) for your creations.

Enabling the Front Panel

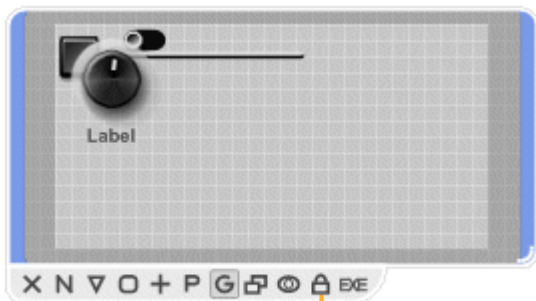
The front panel is shown on the module itself. By default the front panel is disabled. To enable it, select the module then click the **G** (GUI) button (you can also right-click on the module and select Enable Front Panel).

If you resize the module the front panel will resize accordingly.



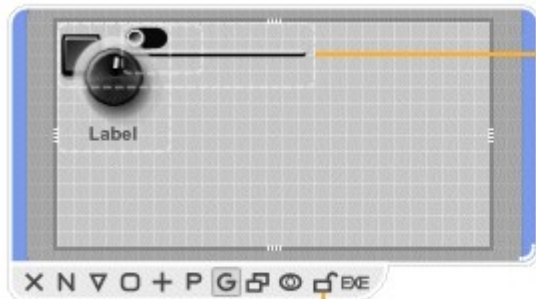
Editing the Front Panel

When you place knobs, sliders and other controls inside a module that has a front panel then these items will instantly appear on the front panel. However, everything will be stacked up in the top-left corner.



Click to unlock the front panel

In order to arrange the items you need to unlock the front panel. First select the module then click the padlock button on the action panel (you can also right-click on the module and select Edit Front Panel or hold CTRL and press E).



Panel items that you can interact with have dotted outlines

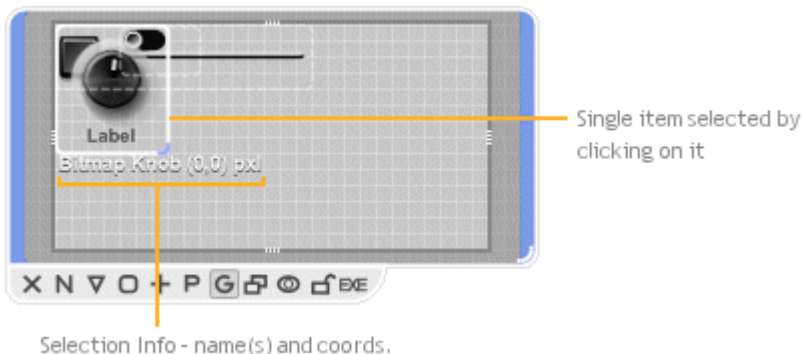
Panel is now unlocked for editing

The front panel will stay unlocked until you lock it again. However, you'll only be able to edit the panel while the module is selected. When the module becomes deselected it will operate as if the panel was locked.

Selecting

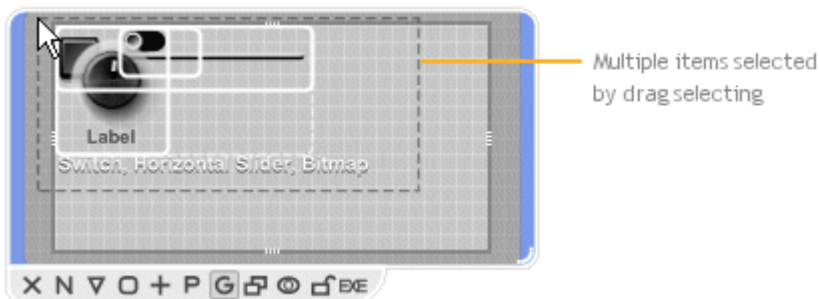
You can select items in the front panel in the usual way by clicking on them. If you hold SHIFT you can click to add or remove items from the selection. You can also click on an empty part of the front panel and drag to select all items in a rectangular area. To select all the items you can use CTRL+A.

If one item is underneath another you can hold ALT and click to alternate between selecting the top item and the one below.



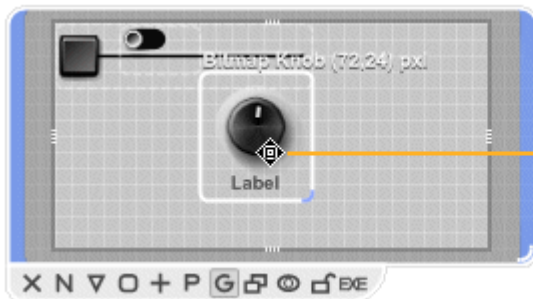
Selection Info

Either above or below the selected items you'll see some text showing information about the selection. The first part this is the module names for the items in the selection. The second part is shows you the coordinates of the top-left corner of the selection. The coordinates are in the form "(x,y)" followed by "sq" or "px". If sq is showing then the coordinates are in grid squares. If px is showing then the coordinates are in pixels at the default zoom level. You can switch between grid squares and pixels by clicking on the selection info text.



Moving

Having selected items you can then move them by dragging them around. By default the position snaps to the grid but you can prevent this by holding CTRL as you drag.



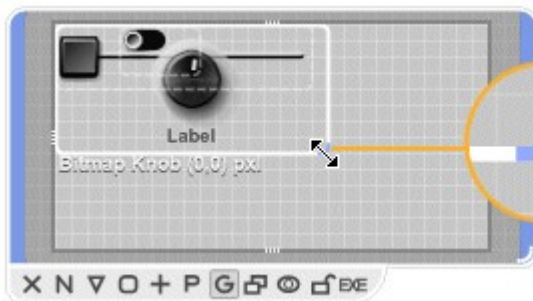
Click and drag the item to move it
Hold CTRL to prevent snap to grid

You can also use the cursor keys to nudge a selection by one grid square at a time in any direction. If you hold CTRL while nudging you will move a distance equivalent to 1 pixel at the default zoom level.

If you're zoomed in or out then you'll move by a distance that is equivalent to 1 pixel at the current zoom level. This allows you to have ultra fine control over the placement of items on a front panel.

Resizing

You can resize front panel items. This works in the same way as for components in a schematic. First select the item you want to resize. The bottom-right corner of the selection will show a blue resize control. Click and drag this to resize.

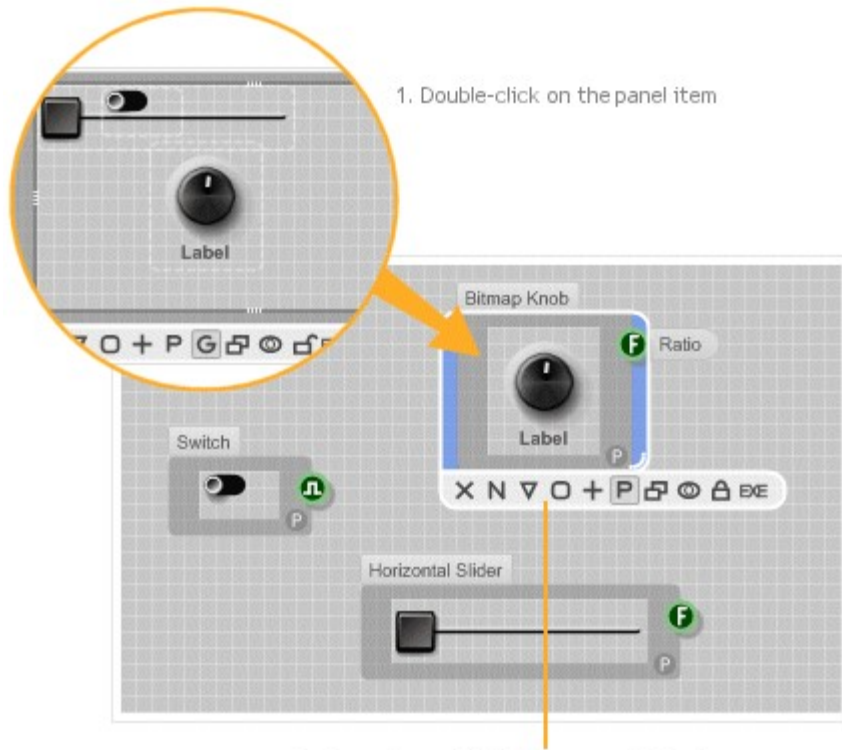


Click and drag the blue resize handle

Note that because each item in the front panel is actually the front panel of a module somewhere lower down in the hierarchy, resizing an item will resize the corresponding module in the schematic.

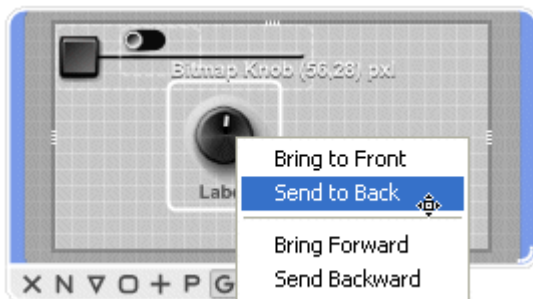
Jumping

If you want to jump straight to the module that is represented by a front panel item then all you need to do is double-click on it.



Draw Order

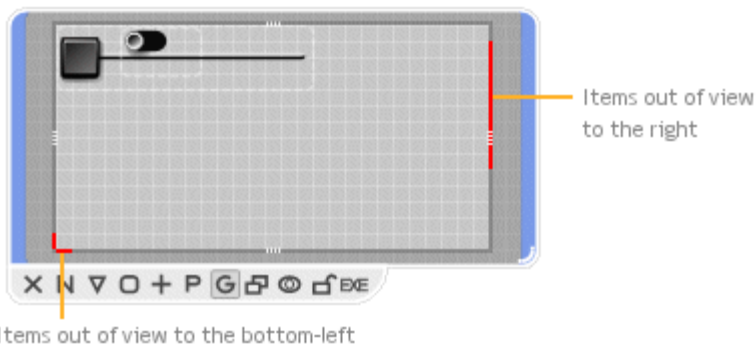
If you have several items that overlap with each other then you can determine which ones appear on top of the others by using the order menu. Right-click on a selected item (or items) and a pop-up menu will appear.



You can then choose whether to bring a selection forward in the draw order or send it backwards. You can send the item straight to the back or bring it to the front.

Out of View Items

It is possible for panel items to move out of view. This can happen when resizing the panel or when copying and pasting modules. When this occurs you'll see red markers on the edge of the panel's exterior. These also indicate the location of the items.

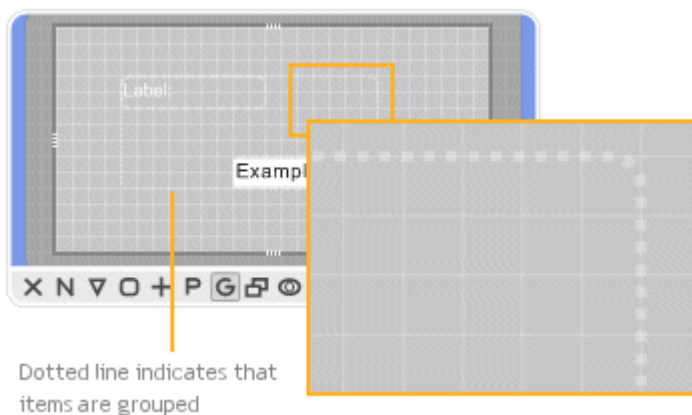


To get the items back into view simply right-click on the module and select Bring Panel Items Into View. This option is also on the Schematic menu on the menu bar.

Grouped Items

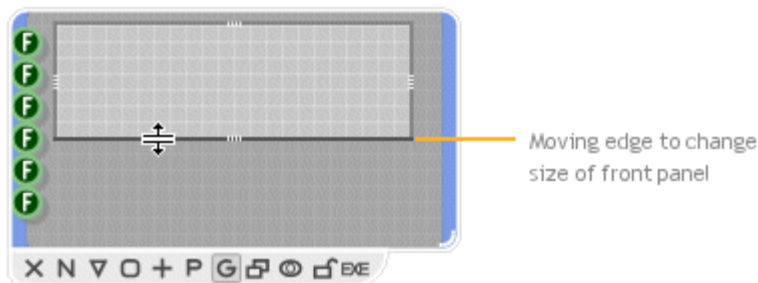
Sometimes front panel items are grouped together. This is determined by the way that the modules are constructed. We'll talk about how this comes about later on so for now we'll just talk about what happens when it occurs.

Grouped items are indicated by a thin dotted line around the items. The only way that grouped items behave differently from those that are not grouped is that they are constrained to appear in the same draw order relative to each other. For example, if you send one item in the group backwards, all the other items in the group will move backwards too.



Client Area

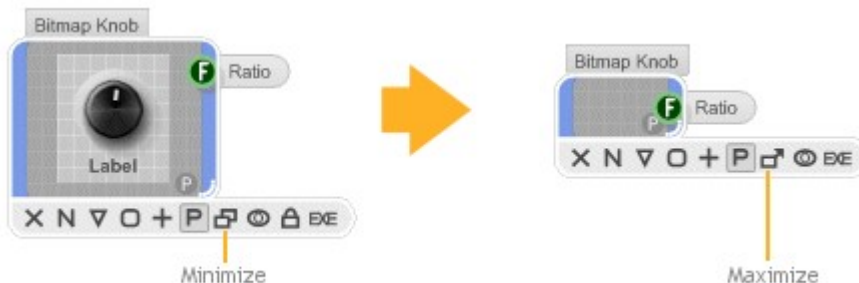
On occasions you may need to change the size of a front panel without changing the size of the module. This can happen when the number of module inputs or outputs imposes a minimum size on the module but you want the height of front panel to be smaller.



When the front panel is unlocked you'll see a thick dark grey border around the panel edge. This defines the client area. You can drag the borders around to change the size of the client area. When an edge is moved from its standard position it changes to a darker grey colour. Note that when this happens the edge becomes detached from the module boundary and will not resize when the module resizes unless it is forced to do so. You can reattach an edge to the module by dragging it back to its original position

Hiding the Front Panel

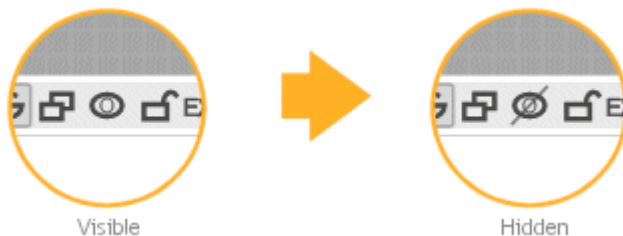
Sometimes you want a module to have a front panel but you don't want to display it. This may be because it is too big and you want your schematic to be compact. You can hide the front panel by minimizing the module. First select the module then click on the Toggle Minimize button on the front panel (or right-click on the module and select Minimize Front Panel).



The module will appear as it would be if no front panel were present. You can maximize the module again to show the front panel if you wish by performing the same action.

Visibility in Parent Module Panels

There are some circumstances under which you don't want a module's front panel to appear in parent module front panels. For example, you may be using a knob to control a variable for experimentation purposes or you may want to hide a background image to stop it from getting in the way while you arrange other panel items.



You can hide a module by selecting the module and clicking on the eye button in the action panel (or you can right-click on the module and select Show In Parent Panel). The eye button will show with a strikethrough to indicate that the module will not be shown in parent module front panels.

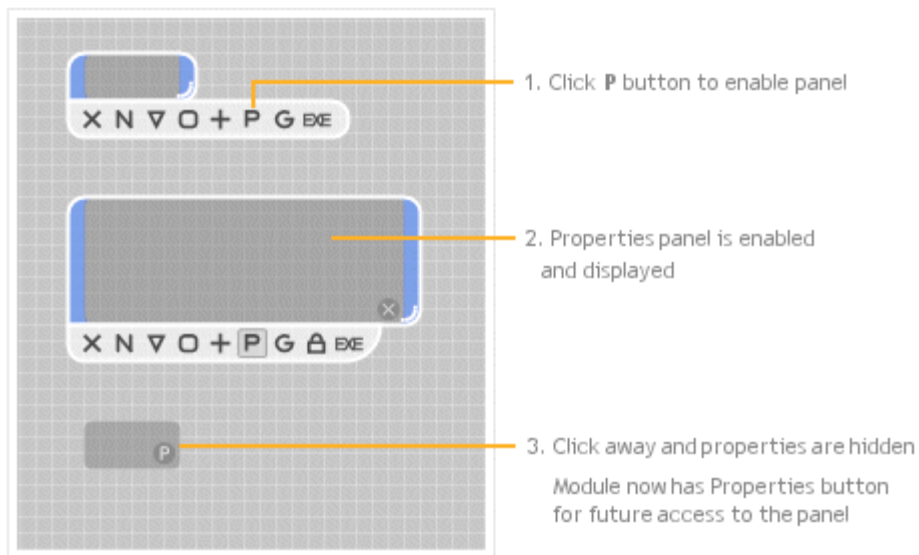
The Show In Parent Panel operation will work on multiple selections too.

Properties

We've already seen how to access and change the properties of a module but how do they get there in the first place? In this section you'll learn how to add properties to your own modules.

Enabling the Properties Panel

To enable the properties panel all you need to do is select the module and click the **P** button on the action panel. You can also right-click on the module and select Enable Properties.

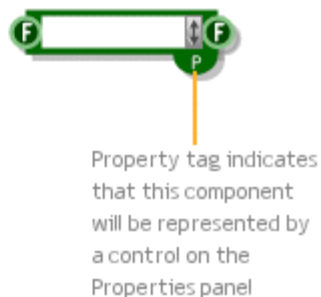
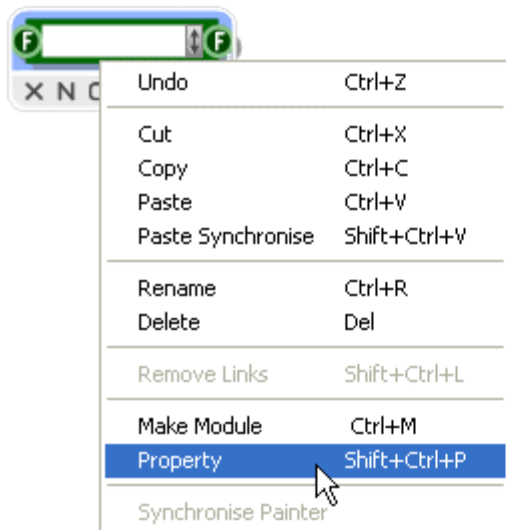


Adding Property Items

The items on the property panel each map onto a particular component somewhere within the module or it's sub-modules. There are six different types of components that can be enabled for display on the properties panel. They are: Int, Float, String, Boolean, Index Selector and Trigger Button.



To enable one of these components for display on the properties, right-click on the component and select Property. You'll see a small tag with a letter P in the middle appear in the bottom-right corner of the component. This indicates that this component is a Property and will be represented by a control on the Properties panel.



If you add a label to the component this will be used on the front panel to identify the component. If you now go back to the properties panel you'll see the property control for the component.



Control Types

Each of the five property enabled component types has a control that represents it on the Properties panel. There are just three different types of control.

Float, Int and String components are represented by Edit controls. These have a label which takes its text from the label of the component.



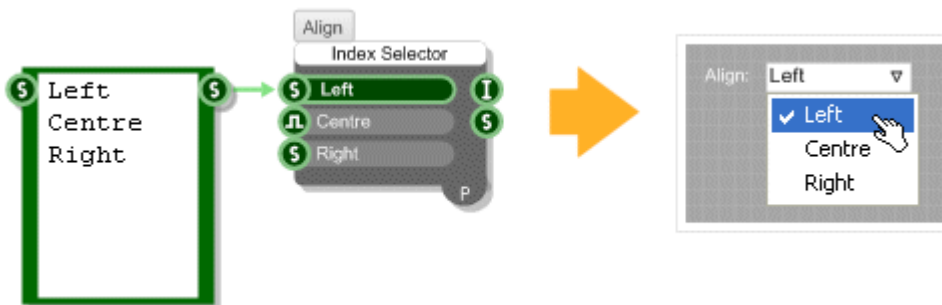
Boolean components are shown as a check box. Again the label for the check box uses the text from the label of the Boolean component



The Trigger Button component is represented by a button. The button text is taken from the label assigned to the Trigger Button component.

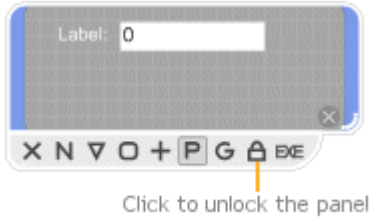


The Index Selector component is used for selecting from a list of options so this is represented as a drop-down list. The component label once again supplies the text for the control label.



Editing the Properties Panel

To edit the positions of items on the panel just unlock the panel, exactly the same as you would for editing a front panel.



Editing is then exactly the same as for the front panel. You can select items, drag and nudge items around or resize them. See the section on the module front panel for more details.

Resizing

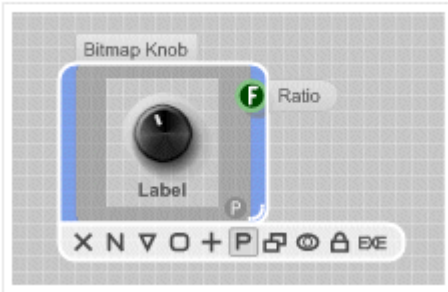
The size of the properties panel can be adjusted independently of the size of the module. When the panel is open any resizing of the module will only apply to the properties panel. When you click away or close the properties panel the module will return to its original size.

Customizing

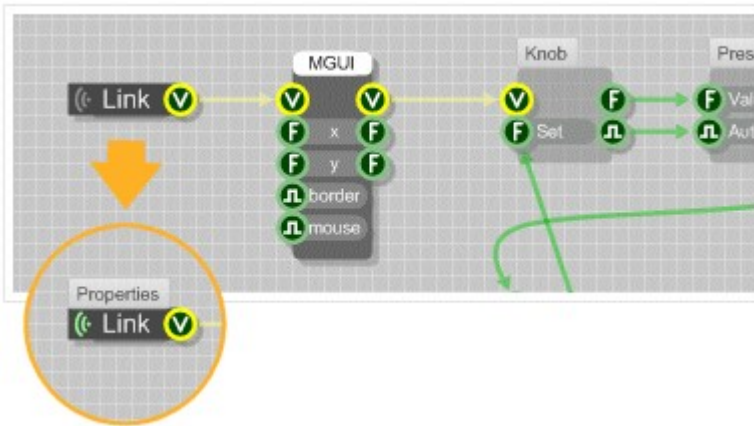
You are not just restricted to the three control types described earlier. Because the panel works just like an extra front panel you can in fact add any kind of controls or graphics you like. All you need to do is connect your custom GUI to a Wireless Input with the label 'Properties'.

The example below shows how to show a knob on the properties panel.

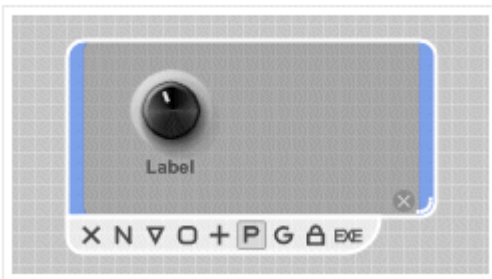
1. Drop knob inside your module



2. Go inside knob module and rename Wireless Input



3. Go back up and knob is now on properties panel



Synchronising

You'll often find that you have many copies of the same module scattered around your schematic. This is one of the great benefits of modules – you create one and then you can use it in many other places. However, what happens when you want to change the behaviour of the module?

Well, you certainly don't want to have to change all your copies one by one. You could change just one of them and then copy and paste it again but that's not great either. It would be much better if you could just transmit the changes to all your modules as you make them. That's exactly what Module Synchronising or Module Syncing is designed to do.

When one or more modules are synchronised any changes you make to one module are instantly made to all the others, it's that simple.

Paste Synchronise

You can put copies of modules in sync as you make them. Instead of pasting a copy of the modules, choose Paste Synchronise from the Edit menu or use SHIFT+CTRL+V.



Syncing symbol and sync count

When you Paste Synchronise for the first time you'll notice that both the original module and the copy now have the module syncing symbol. The pasted copy will be selected. When a synced module is selected you will also see the sync count. This indicates how many other modules are synced with this one. The count is of the form xN where the x symbol represents multiplication and N is a number.


Synchronise All

If you didn't think in advance that you wanted copies of a module to be in sync then don't worry. There's an easy way to put them all in sync after they have been created. Just right-click on one of the modules and select Synchronise All. All the modules that match the module you clicked will be put into sync, regardless of where they reside in your schematic.

Note that if any of the original copies have been altered in any way then they cannot be brought into sync.

Synchronise Painter

For more selective syncing we have the Synchronise Painter feature. This is a mode that you put the software into which allows you to click on the modules that you want to put in sync.

Right-click on the module that you want to synchronise with and select Synchronise Painter. The cursor will change to the synchronise painter  cursor. To make another module sync with this one all you need to do is click on it.

If the module is identical and so can be synced the module border will flash blue to indicate that the operation succeeded.



You can navigate through your schematic while the Synchronise Painter is switched on. You can use the Navigator or the module action panels. You can also hold CTRL to temporarily disable painting while you double-click on a module to go inside it.

To exit Synchronise Painter mode you can right-click on your schematic and uncheck the Synchronise Painter option. Alternatively, you can hit the ESC key. If you try and perform any other editing operation whilst the Synchronise Painter is on it will immediately switch off and you'll return to standard editing mode.

Removing Synchronisation

You will on occasions need to be able to stop a set of modules from syncing with each other. It's actually a good idea to remove syncing when you know that you don't need to make any more changes.

There are three ways to do this. You can use the Synchronise Painter as described above. For a really quick way to remove syncing you can right-click on a synced module and select Un-synchronise All. This will remove syncing from all of the modules that are synced with the one you clicked.

You can also just remove syncing for a particular module on it's own by right-clicking on it and selecting Un-synchronise.

5 Data Types & Signal Flow

CONNECTOR TYPES EXPLAINED

FlowStone supports over 30 different types of data. Each type has its own connector with a unique symbol for easy recognition. Although there are many different types of data they all fall into one of two different categories: Stream or Triggered.

Stream data covers all digital audio and control signals. These signals are fluctuating at sampling rate and so any components which process them will also perform calculations at sampling rate.

Triggered data works in a completely different way. Where stream data is continuously flowing, triggered data only flows in response to some event. Usually the event is a user interaction or data arriving from some external source like an interface card or a timer.

Note that in the vast majority of cases data flows from left to right, by this we mean from output connector to input connector. This is true for all Stream data but there are a few exceptions with some of the triggered data types.

Stream Data

There are two main stream data types: **Poly** and **Mono**. Poly is only used for audio applications where sound signals are generated from MIDI notes. If you're not generating audio in this way then you can ignore Poly completely.

Mono

Mono carries a single signal at sampling rate. A set of mono components connected together (to form what we call a Mono section) always has data flowing through it. Once connected on the right to a Direct Sound Out or ASIO Out component a Mono section will run constantly even if there is a zero level signal passing through.



Mono connector - a single channel of fast moving stream of floating point data that fluctuates at sampling rate. Provided it is connected to a sound output component like Direct Sound Out it is always active.

Poly

Poly connectors can carry many digital signals at one time. A Poly section only uses processing when there are signals passing through it. Poly is only used for audio applications and is generated in response to MIDI notes.

The number of signals is determined by the number of notes being played. If there are no signals then there is no cpu hit. However when there are one or more notes playing you'll get proportional usage of cpu for each signal generated (although because of the way FlowStone uses SSE, you only get an increase on every 4th note).

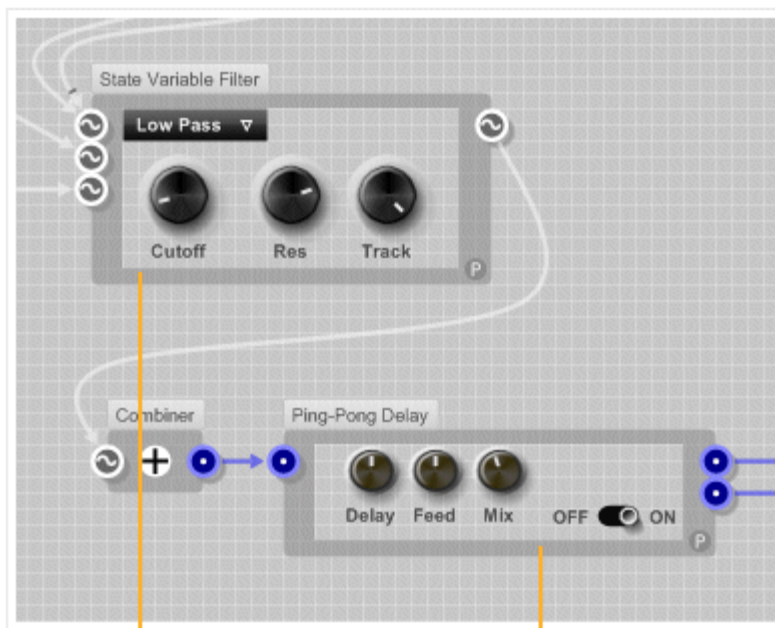


Poly connector - represents multi-channel audio signals. Each channel is an independent fast moving stream of floating point data that fluctuates at sampling rate. The data is multi-channel because there is one channel for each note that you play.

When to Use Poly or Mono

You would use a combination of Poly components (a Poly section) to model the polyphonic part of a synth. With a Poly section you can generate separate audio signals for each voice or note that you play. You can then merge the Poly signals to Mono and have a Mono section where you would apply effects to the combined signal as a whole.

Poly and Mono connectors allow you to easily see what kind of data is flowing through different parts of your schematic. This is important because when adding to your schematic you need to know whether the processing you introduce will be applied cumulatively for each voice (Poly) or constantly for one signal (Mono).



Filter is in Poly section so applies to individual notes but only when they are playing

Ping-Pong Delay is in Mono section and applies to all notes at once, processing constantly

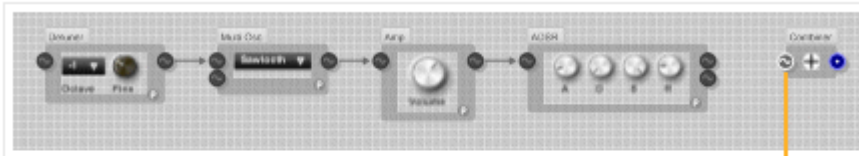
Stream Connectors

You can create modules that can be used in a Poly or a Mono section by using Stream connectors. These look like dark grey versions of the poly connector.

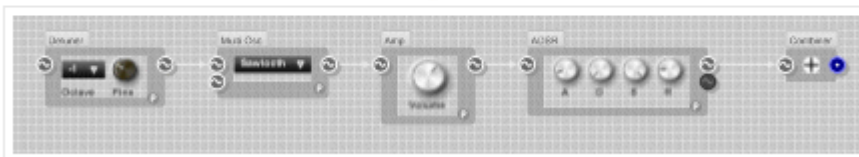
 Stream connector

When a Stream output connector is linked to a Poly or Mono connector it instantly picks up that type. The type change then flows from left to right back through any other stream connectors changing them to the new type accordingly.

1. Stream section (indicated by dark grey links and connectors)



2. Link to Poly connector

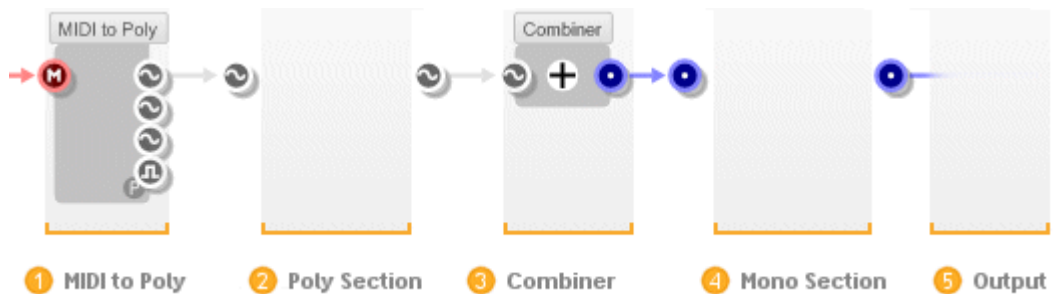


3. Turns to Poly section

Poly and Mono Sections In Audio Applications

To get any data to flow through a poly or mono section you have to connect it up correctly. In most cases you'll want to use MIDI input from a keyboard or other source to generate notes. You then want each note to make a sound that you can hear.

The most common setup would be as shown below:



1. The MIDI to Poly module converts incoming MIDI note data into Poly signals
 2. The Poly section processes the signals
 3. The Combiner module then combines the independent Poly signals into one Mono signal
 4. The Mono section applies processing to the result
 5. The final signal is sent to your Sound Card via a Direct Sound Out or ASIO Out component within FlowStone or via the host audio setup if used within a plugin
- NOTE:** If you don't have Direct Sound Out or ASIO Out component no audio processing will be performed.

A complete, usable poly section always starts with a MIDI to Poly module and ends with a Combiner module. The only exception is if you're using a Signal Analyser to look at the output from a Poly section.

You can have a Mono section without a Poly Section. This is what you'd have if you were creating an effect that performs some DSP on an incoming signal. However, if you want to hear anything you have to connect a Mono section to an Output device via a Direct Sound Out or ASIO Out component.

Boolean Connectors

There are Boolean equivalents of the Poly, Mono and Stream data types. They are only seen on a few components, most notably the Poly and Mono comparison components.

The MIDI to Poly module has a Poly Boolean output which shifts from false to true when a note is played and can therefore be used as a gate signal.



Poly Boolean - multi-channel mask, one mask for each note playing



Mono Boolean - single channel mask, constantly running



Stream Boolean

Poly Int

The Poly Int data type is the integer equivalent of the Poly type. It is used only in the Graph to Poly to allow sample rate indexing of Float Arrays.



PolyInt - multi channel integer, one channel for each note playing

SSE

If a cpu supports SSE then it has a set of instructions built in that allow mathematical operations to be performed on multiple sets of data at the same time. FlowStone makes full use of this when processing stream data. The result is that you can effectively process up to 4 channels at the same time for the same cpu cost as just one.

Mono 4

If you really want to get the best performance out of a Mono section then you need to consider using the Mono4 data type. It's not always possible to use this but when you can it makes a huge difference to performance.

Mono sections only have one sound process. Because FlowStone uses SSE this process could be doing four times the work for the same cpu cost. To take advantage of this we have the Pack and Unpack components. These allow you to literally pack four mono channels into one Mono4 stream.



Mono4 - four Mono signals processed as one

Performance

To help you gauge the effect of changes to the stream parts of your schematic we have a cpu meter. This can be found on the right side of the status bar at the bottom of the application window. The cpu meter is very basic. It only measures cpu performance of the stream sections, GUI performance is not measured.

PC Keyboard CPU = 2.1%

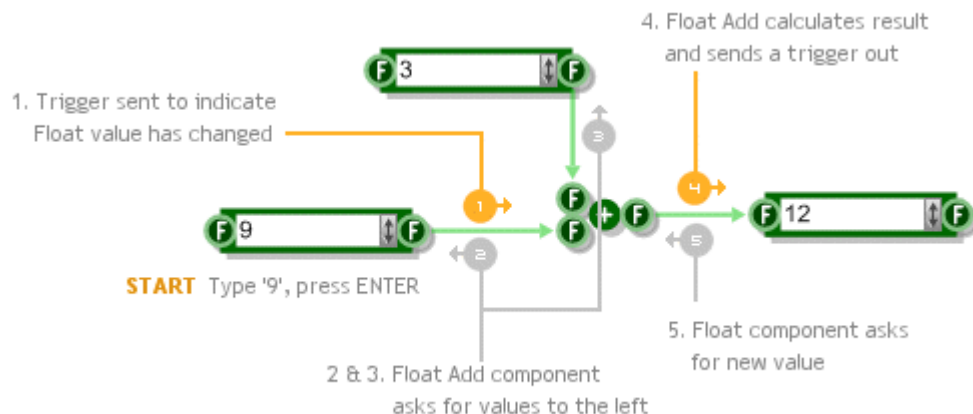
Triggered Data

How it Works

Triggered data only flows through a schematic when an event occurs that 'triggers' a change. Until this happens sections of triggered data remain in a steady state.

When an event does occur a message called a Trigger is sent out through output connectors on one or more components. The trigger flows through the links until it reaches another component. That component then assesses whether its state would be affected. Any necessary recalculations are then performed and a trigger is sent out through any output connectors on that component. The process continues creating a waterfall effect through the schematic.

The example below shows a very simple example of two numbers being added together. When you type in a Float component this results in one of the events that we've been talking about and a trigger flurry begins.



Note that during the recalculation part a component may ask the components that provide values to its inputs for their latest values. This in itself can cause a flurry of messages to the left.

What it's Used For

Triggered data has two main functions. First it provides a way of performing calculations that don't need to be done at sampling rate. Second it allows you to build extremely detailed and complex user interface elements which only need to respond when you interact with them.






As a general rule the fast DSP is handled by the Stream data and the slower DSP and user interface is handled by the Triggered data.

Triggered Data Types

The various triggered data types can be grouped into several categories based upon their purpose.




Primary Types

These are the most commonly used types. They represent simple data like numbers and text.

-  Float - a 32 bit floating point number
-  Int - a 32 bit signed integer in the range -2147483648 to 2147483648
-  String - an alphanumeric string of characters of unlimited length
-  Boolean - one of two values: true or false
-  Trigger - not really a data type (there is no data) but used to pass trigger messages












Array Types

These types represent resizable arrays of some of the primary types. An array is just an ordered list of items of the same type. Array types have a 'four leaf' outline pattern instead of a circle.

-  Float Array - array of 32 bit floating point numbers
-  Int Array - array of 32 bit signed integer in the range -2147483648 to 2147483648
-  String Array - array of characters strings



GUI Data Types

These are used only for low-level GUI editing so you won't need to know about them until you come to use the GUI components.

-  View - transports all drawing and mouse information
-  Area - an area defined by coordinates of top-left corner, a width and a height
-  Graph - a list of floating point integers
-  Mouse - mouse events (left button up/down, mouse move etc.)
-  Colour - in argb format ('a' is the transparency level)
-  Pen - for drawing lines - defined by colour, thickness and style
-  Font - font information comprising typeface, size and style
-  String Format - alignment information for drawing text
-  Bitmap - 32bit image
-  Point Array - an array of points (floating point pairs)
-  Bitmap Array - an array of bitmaps




Memory Types



There are two types that represent buffers of data stored in memory. They are most frequently used for storing wave file data for samples.

-  Mem - a contiguous section of data in memory
-  Mem Array - an array of memory sections of unlimited length

Special Types

There are a handful of data types that stand on their own.

-  MIDI - all standard MIDI messages
-  Voice - currently only used within the MIDI to Poly module but may be extended in future
-  Bus - a user defined set of data types that travel together through the same connector

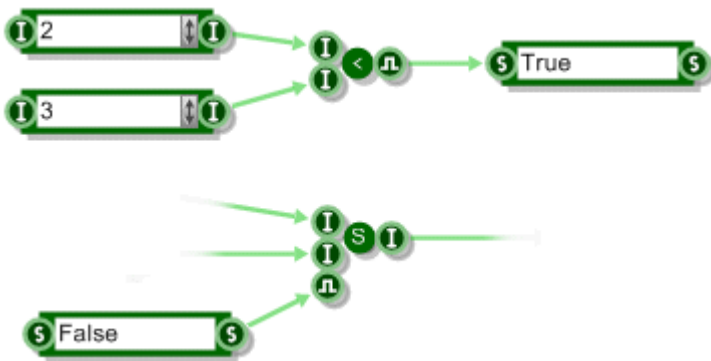
-  Preset - carries information about preset parameter and program changes for plugins
-  Template - not a data type, instead this takes the type of whatever you connect it to

Converting Between Data Types

In most cases you will create links between connectors of the same type. However, FlowStone supports several automatic and intuitive conversions between data types which you'll find extremely handy.

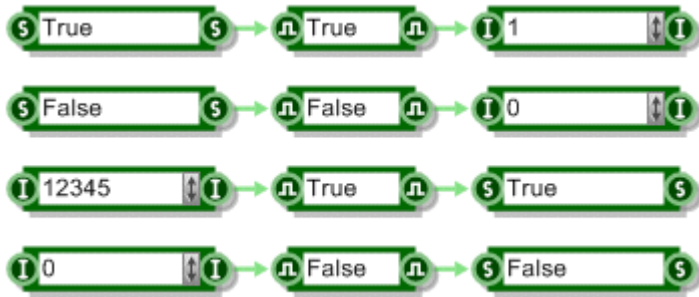
String <> Boolean

You can get a boolean value from a string and also convert from a boolean to a string:



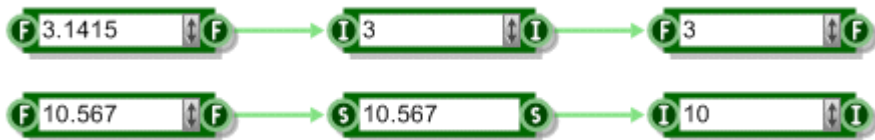
Int <> Boolean

Boolean and Int are also interchangeable. Zero represents a false value, any other value is considered to be true.



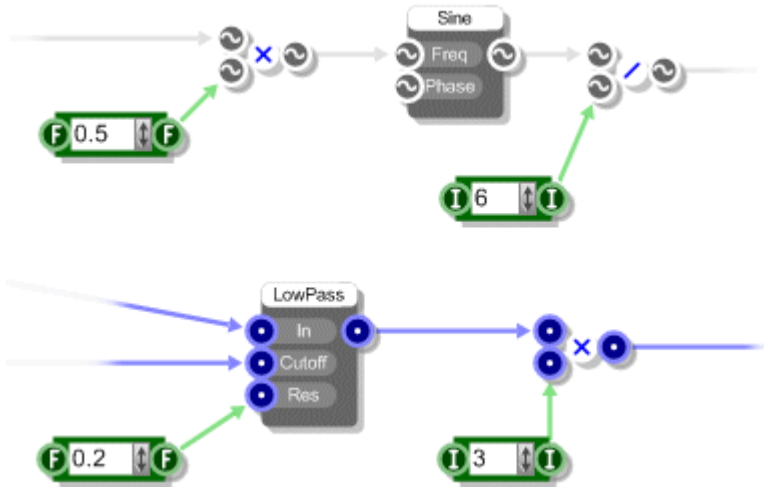
String <> Float <> Int

You can convert back and forward between strings, floats and ints. Rounding will obviously occur when converting from Float to Int (the decimal part is just ignored).



Int/Float > Poly/Mono

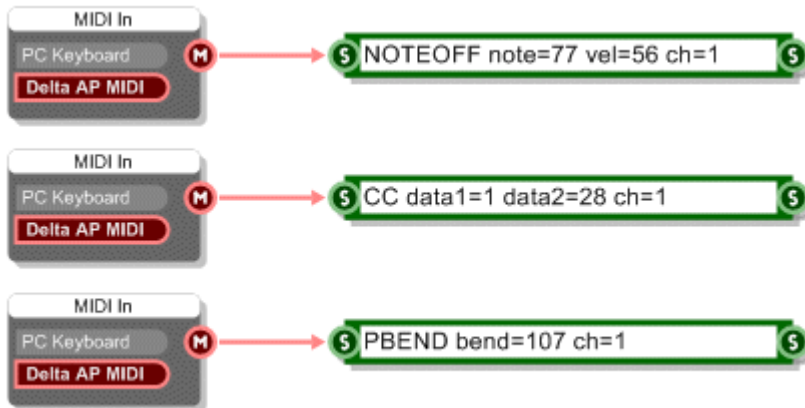
A Float or Int can be connected to Poly or Mono inputs. These act like signals that maintain a constant level. You can't however connect a Poly or Mono to a Float or Int. In fact Poly and Mono connectors can only be connected to themselves.



MIDI <> String

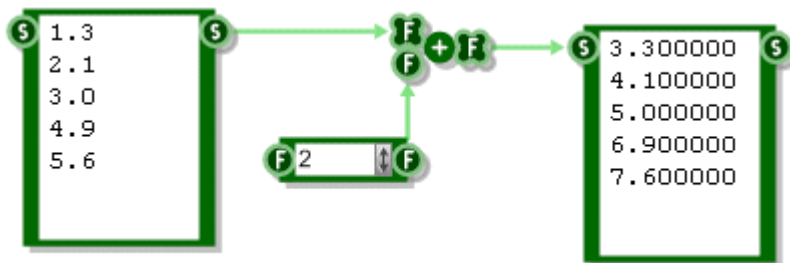
This is not so much a conversion as a handy way to check what MIDI data is coming from a MIDI output. Attach a String data component and note on/off, control change and pitch bend messages etc. are shown together with the various parameters that define them.

As this isn't a conversion as such you need to 'force' MIDI to String links by holding SHIFT + CTRL when linking.



Float Array <> String

You can easily convert between a string and a float array.



String Shortcuts

The String data component can be used as a shortcut for defining various GUI data types. Colours, Areas, Pens and more can all be defined in a single text string.

Area

To create an area use the format "x,y,w,h" where x and y give the top-left corner of the area and w and h give the width and height of the area. All dimensions are in grid squares of course.



Colour

There are two ways to specify a colour using a data String. You can use one of the 14 predefined colours which are as follows:

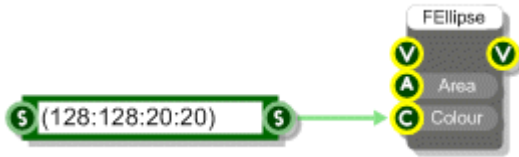
Black, White, Red, Green, Blue, Yellow, Grey

tBlack, tWhite, tRed, tGreen, tBlue, tYellow, tGrey

(the 't' colours are partially transparent)

Use the colour by name in the String component. Note that the colour names are not sensitive to case.

The second way to specify a colour is by ARGB. Use the format "(a:r:g:b)" where **a** is the transparency, **r** is the red component, **g** is the green component and **b** is the blue component. All values are integers in the range zero to 255.



Pen

A pen has three attributes: colour, thickness and style. Using a string component you can specify a pen by providing these attributes in the format "colour,thick,style".

The colour part is exactly the same as for specifying a colour with a string (see above). The thickness is a floating point number in grid squares. The style can be any one of the following strings:

`solid, dash, dot, dashdot, dashdotdot`

You can leave the style parameter out and a solid style will be assumed.



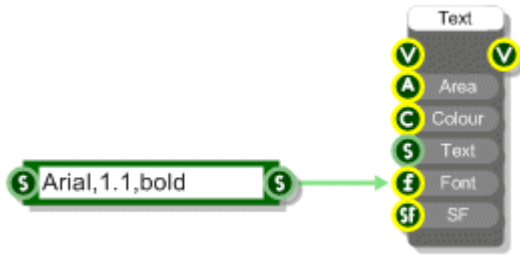
Font

Fonts can be generated by Strings using the format "typeface,size,style". Typeface is the name of the font face eg. Arial or Tahoma. Size is the height of the text in grid squares (it is not a point size).

Style can be any combination of the following strings (in any order):

`normal, bold, italic, underline, strike`

Some examples: bolditalic, underlineboldstrike, italicunderline. You can leave the style parameter out and a regular style will be assumed.



StringFormat

StringFormats are specified using the format "style,horizalign,vertalign". The style can be any combination of the following strings (in any order):

`normal, righttoleft, nowrap, vertical`

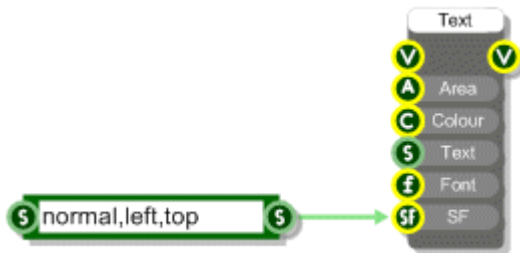
Some examples: nowrapvertical, nowraprighttoleft. You can also use 0 to indicate no style options apply.

For horizontal alignment you can use:

`left, center or right`

For vertical alignment you can use

`top, center or bottom`



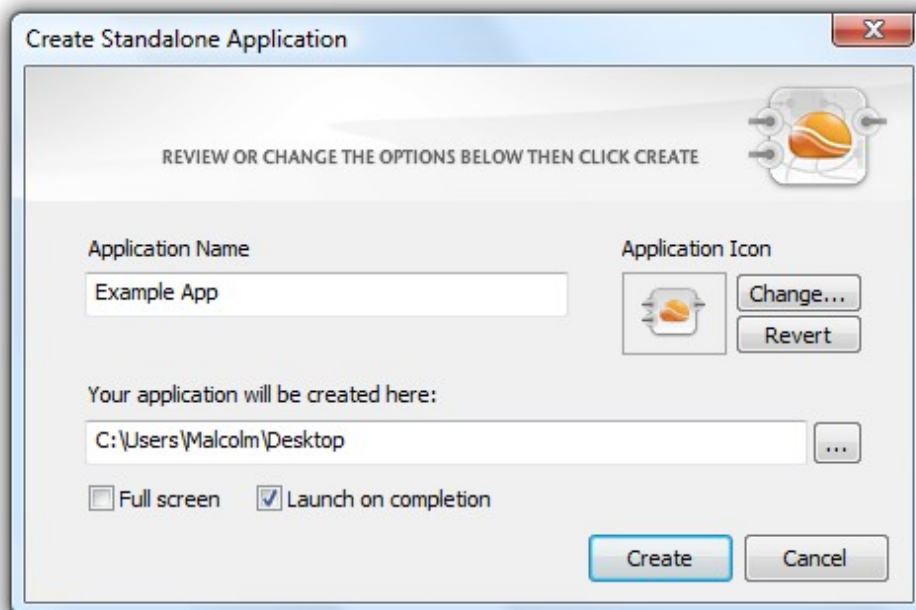
6 Exporting

CREATING STANDALONE APPLICATIONS

Creating Standalone Applications

FlowStone can be used to create complete applications that will run on their own. First click the EXE button on the action panel of the module you want to export (or right-click on the module and select Create Standalone from the pop-up menu).

The Create Standalone Application dialog will appear.



Application Name

The application name is the name of the .exe file that will be generated. By default the software will use the module label. If there is no label the dialog will show the last name that you used.

Application Icon

You can choose your own icon for your application. Click Change and locate the icon you want using the resulting dialog box. Icons need to be stored in .ico files before they can be used by FlowStone. If you decide you just want to use the default icon click the Use Default button. A preview of the icon that the software will use is shown on the dialog box.

Your application will be created here

Applications are created in a particular folder and you have the option to change this.

Full Screen

Check this box if you want your application to launch and fill the whole screen. This is the ideal option for applications intended for embedded systems.

Launch on Completion

You can choose whether to launch the newly generated application on completion by checking this box.

Click Create and within a few seconds your application .exe will be created in the target location that you specified. If you checked the Launch On Completion box the application window will appear.

Library Dependencies

On most occasions your exported exe can be distributed on its own. However, some components in FlowStone are reliant on external libraries. If your project uses these components and you then export to a standalone you will need to distribute the supporting libraries together with your executable.

You will be informed of any dependency when you export.

We provide pre-packaged installers for the libraries on our web site. You are free to distribute these in the form they are provided. For more information see our web site page:

<http://www.dsrobotics.com/libraries.html>

7 Advanced GUI Editing

THE GUI COMPONENTS AND HOW TO USE THEM

Module GUI

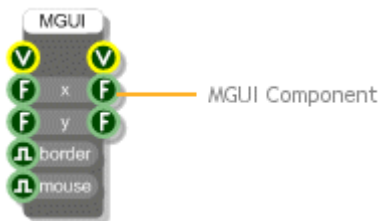
In the Modules chapter we learned how to enable a module's front panel, add items to it and move them around. We used sliders, knobs and switches from the toolbox. These are themselves modules but their front panels are constructed using GUI components.

In this chapter you'll see how to create your own front panels using the GUI components. Working at this level you have complete control over every detail of how your front panel looks and behaves.

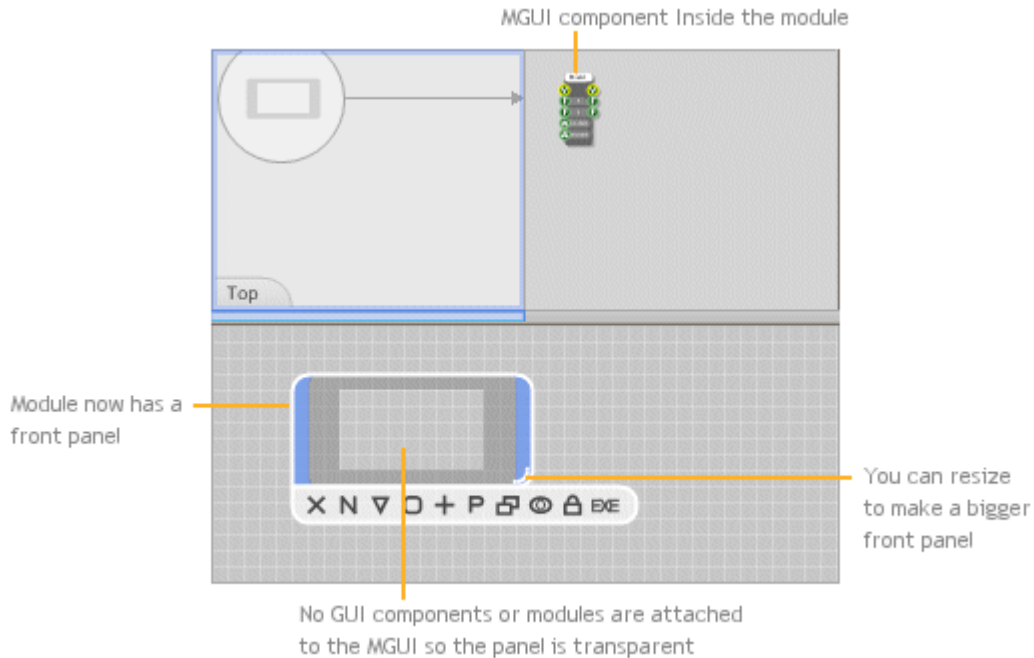
Module GUI Component

We've already seen that the starting point for any GUI in FlowStone is a module. Modules provide the base on which a GUI can be constructed via their front panel. You can enable the front panel by clicking the G button on the action panel.

However, if you want to use the GUI components you need to have access to the panel within your schematic and this is done through the Module GUI component (or MGUI for short).



Create a new module then drop an MGUI component inside. You can find the MGUI component under the Module filter group in the toolbox.



If you move back up a level you'll see that the module now looks a bit different. It now has a front panel, albeit with nothing in it. You can resize the module to make the panel bigger or smaller. You may also notice that the G button has been removed from the action panel.

MGUI Connectors

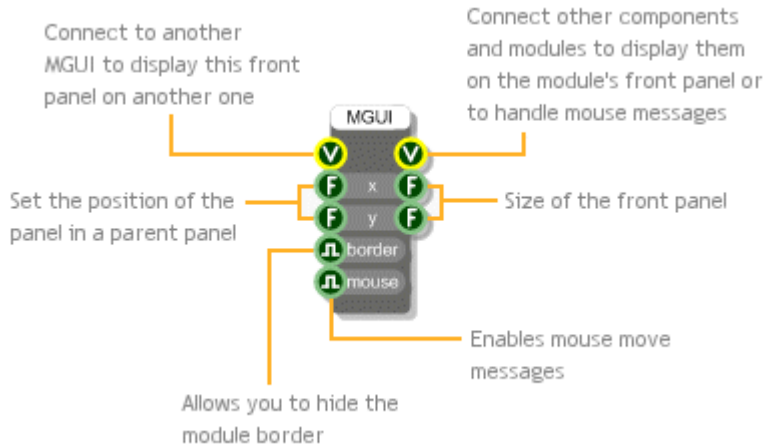
All GUI information is sent through View connectors. These are yellow circles with a V in the middle. The MGUI has one View output. Anything connected to this will either draw onto the front panel or handle mouse messages from it.



View - handles all drawing and mouse messages

The two Float outputs can be used to get the size of the front panel if this is needed.











For the moment you only need to know about the output connectors. The input connectors come into play when things start getting more advanced. We'll cover this in a later section.



GUI Connector Types

The GUI components introduce a new set of data types. Each type has its own connector and each connector has its own symbol. However, the symbols all have the yellow circle in common to show that they are GUI related.

We saw these first in the chapter on Data Types:

-  View - transports all drawing and mouse information
-  Area - an area defined by coordinates of top-left corner, a width and a height
-  Mouse - mouse events (left button up/down, mouse move etc.)
-  Colour - in argb format ('a' is the transparency level)
-  Pen - for drawing lines - defined by colour, thickness and style
-  Font - font information comprising typeface, size and style
-  String Format - alignment information for drawing text
-  Bitmap - 32bit image
-  Point Array - an array of points (floating point pairs)
-  Bitmap Array - an array of bitmaps

Coordinate System

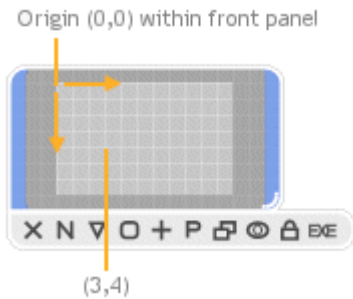
FlowStone uses a grid system to position components. Exactly the same grid system is used by the vast majority of GUI components for drawing and for mouse events.

The grid system is floating point based so you can have fractions of a grid square. This allows for more precise positioning of drawing elements.

Of course there's no getting away from the fact that the screen uses pixels and that these form a discrete grid of points. But by using special rendering techniques, FlowStone can still display graphics as if they were on a continuous surface.

Working in Pixels

Sometimes you need to work in pixels instead of grid squares. To do this you can make use of the Grid Square to Pixel and Pixel to Grid Square components to move between the two systems.



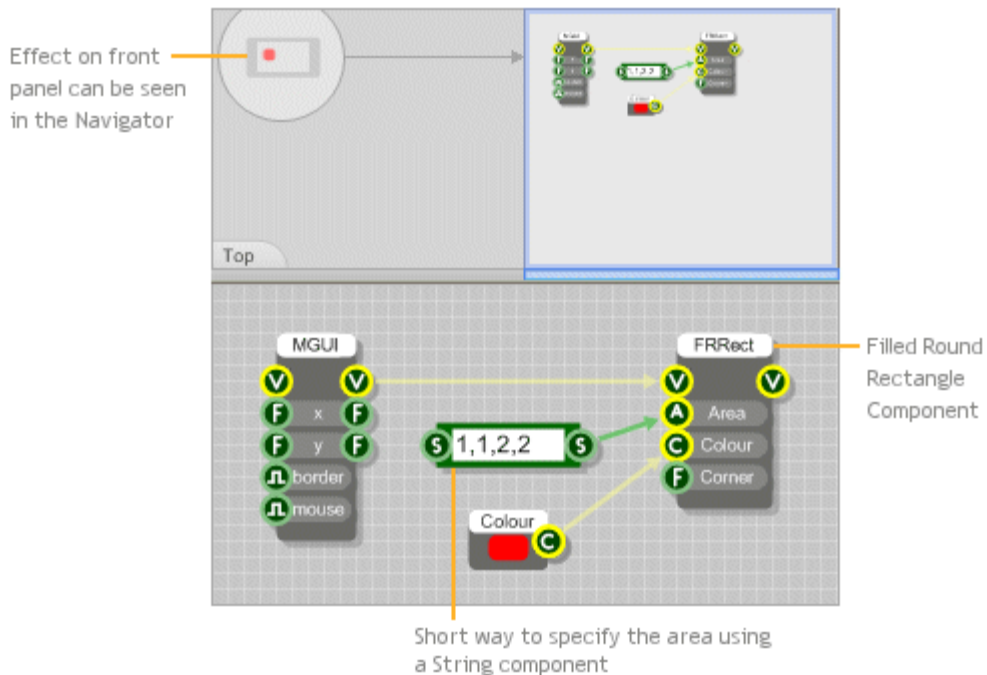
Drawing

Drawing on a Panel

Drawing on a front panel is a simple matter of picking a drawing primitive and connecting the View output of the MGUI to the View input of the primitive.

All the drawing primitives can be found by selecting the GUI filter group. There are primitives for drawing lines, rectangles, text, bitmaps and more.

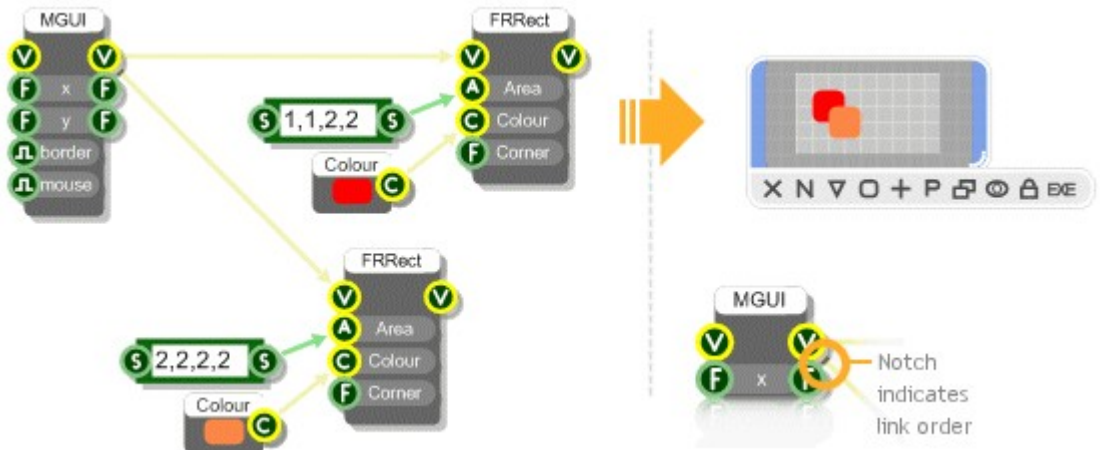
The example below shows how to draw a simple filled rectangle. We've used a little shortcut to specify the Area for the rectangle. This makes use of a string component to specify the x,y,width and height that define the Area. The colour is defined using the colour component.



Drawing Order

Often you'll be drawing more than one element on a front panel. If two elements overlap then the one that is last in the link order will be displayed last and therefore over the top of the other element.

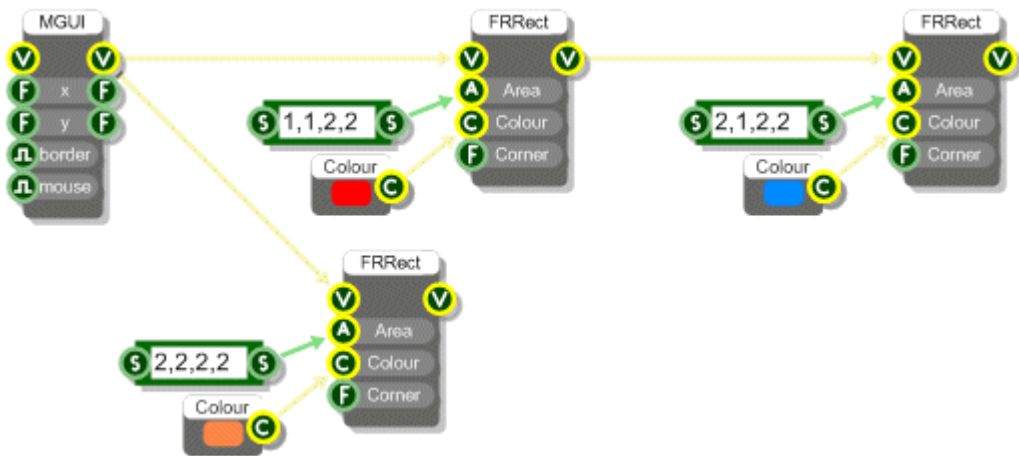
The example below shows this. The orange rectangle is on the second link from the MGUI (see Links for more on link order). The orange rectangle will therefore be drawn on top of the red one.



Chaining GUI Components

Most GUI components have a view output connector. This allows other GUI components to be linked to them so chains of graphical elements can be created.

The example below shows how this is done. The effect on the link order is as follows: the link to the red rectangle is taken first followed by any links from its output connector. Then it's back to the next link from the MGUI etc. In this example the red rectangle would be drawn under the blue one which would then be under the orange one.

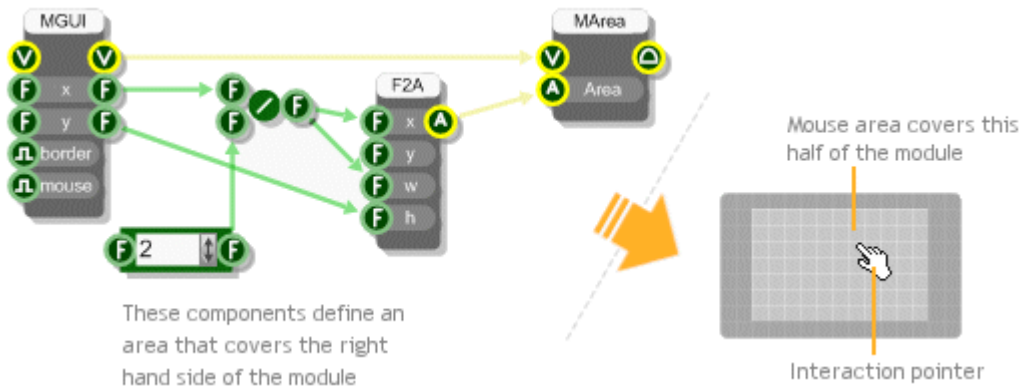


Mouse Handling

Mouse Area

In order to receive mouse messages in a part of your front panel you must first define a mouse area. This is done using the Mouse Area component.

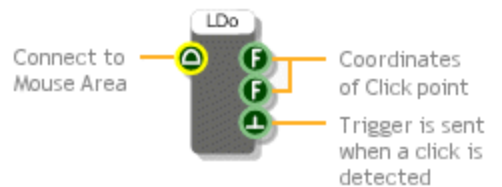
The example below shows how the right half of a module can be made to receive mouse messages. This is indicated by a change in cursor as the mouse pointer passes over the mouse area.



Mouse Clicks

To trap mouse clicks on a mouse area you'll need a Mouse L-Button Down component. Link the Mouse output connector on the Mouse Area component to the input on the Mouse L-Button Down component.

Whenever you click in the mouse area a trigger will be sent to the Trigger output on the Mouse L-Button Down. The coordinates of the click point (in grid squares) will be available from the two Float outputs.

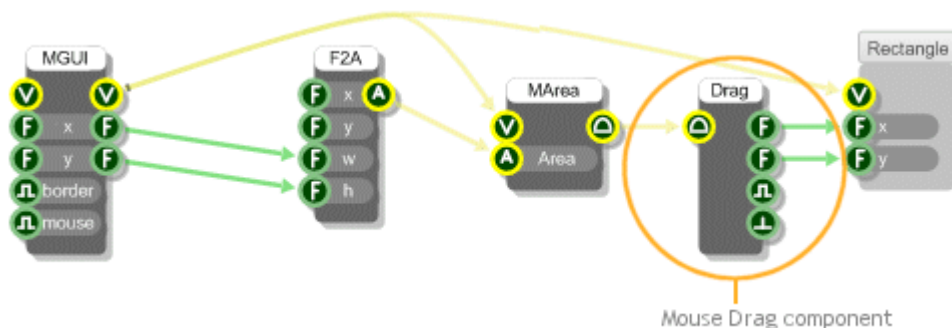


Mouse Dragging

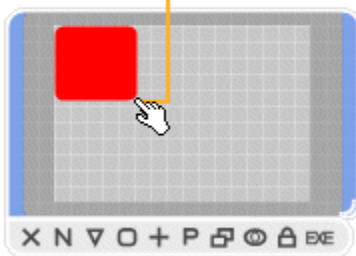
You can track the mouse position while the left mouse button is held down. This allows you to implement drag operations.

The component you need to do this is the Mouse Drag component. This takes Mouse messages at it's input and sends the coordinates to it's two Float outputs as you drag.

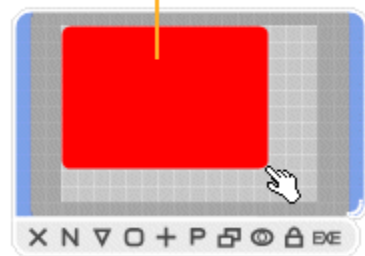
The example below shows how the Mouse Drag component can be used to change the size of a rectangle. We've created our own rectangle module to handle the drawing. This makes the schematic a bit neater.



Click and hold then start dragging

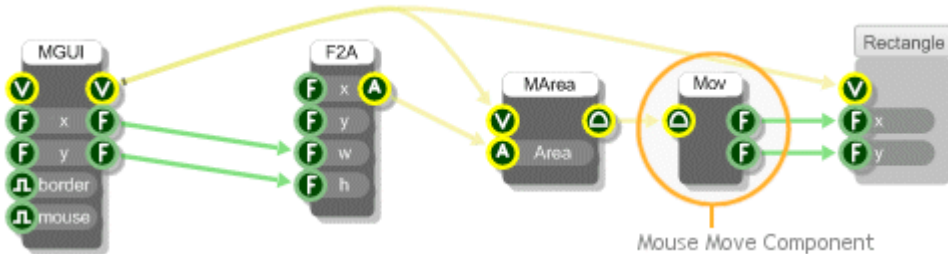


Rectangle changes to reflect mouse



Mouse Moves

You can also track the mouse as it moves over a mouse area. This is done using the Mouse Move component. Before you can use this you need to enable mouse move messages on the appropriate MGUI component. Mouse move messages are disabled by default to reduce unnecessary performance overheads.



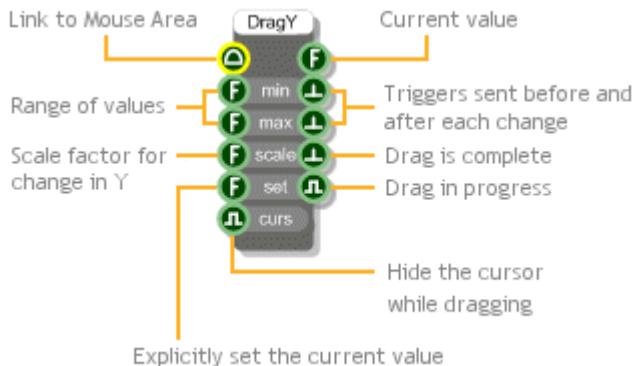
Try replacing the Mouse Drag component in the previous example with a Mouse Move component and you'll see how this works.

Drag Accumulate

The Mouse Drag component is a bit too low-level for some tasks. If you want to create a slider or anything with moving parts that can be dragged around then it's much easier to use a Drag Accumulate component.

Drag Accumulate components manage much of the legwork of drag operations for you. There are three varieties: X, Y and XY.

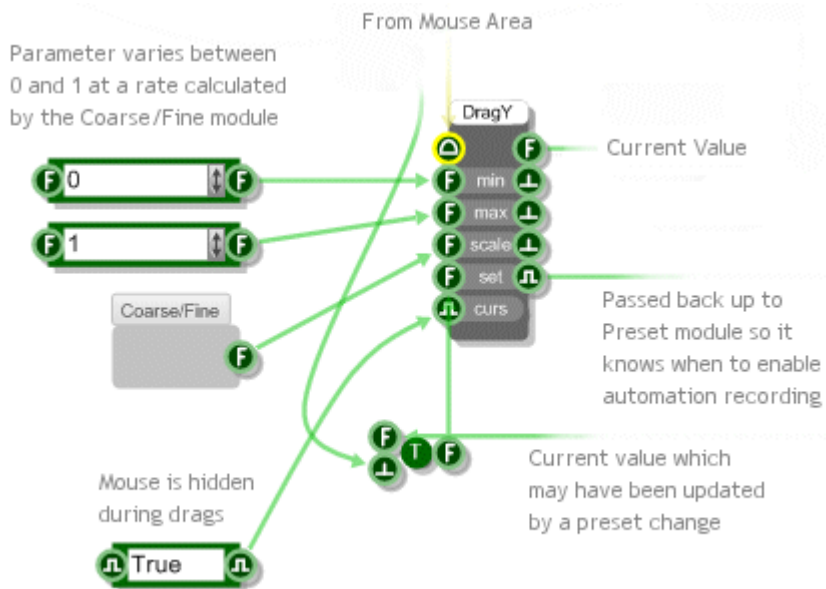
The Y Drag Accumulate manages a parameter that varies over a particular range. When you drag the mouse, the offset from the point where you clicked is maintained by the component. As you continue dragging the parameter is updated according to the Y position of the mouse and a scale value that you can specify.



If you have a look deep inside the Bitmap Knob module in the toolbox you'll see how the Y Drag Accumulate component is used. Move through the following modules:

Bitmap Knob\Knob\Interaction\Knob\Control\Moving Part\Knob Control

Inside the Knob Control module and you'll see the Y Drag Accumulate in the middle. The picture below explains what's going on.



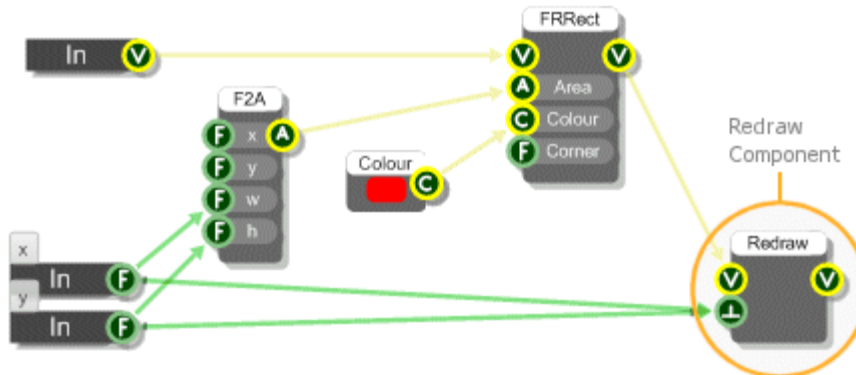
Redrawing

Redraw Control

If you change a property of a drawing element during a mouse operation like dragging, you'll often want the changes to be reflected immediately in the module front panel.

To allow for maximum flexibility FlowStone allows you to control when parts of the front panel are redrawn. In the rectangle dragging example we used the simplest kind of redraw - we just forced the whole panel to refresh. This was done using the Redraw component.

If you look inside the Rectangle module you'll see the Redraw component. When the component receives a trigger it sends a message back up through the View links to the first MGUI it finds. When the MGUI receives the message it redraws everything on it's front panel.

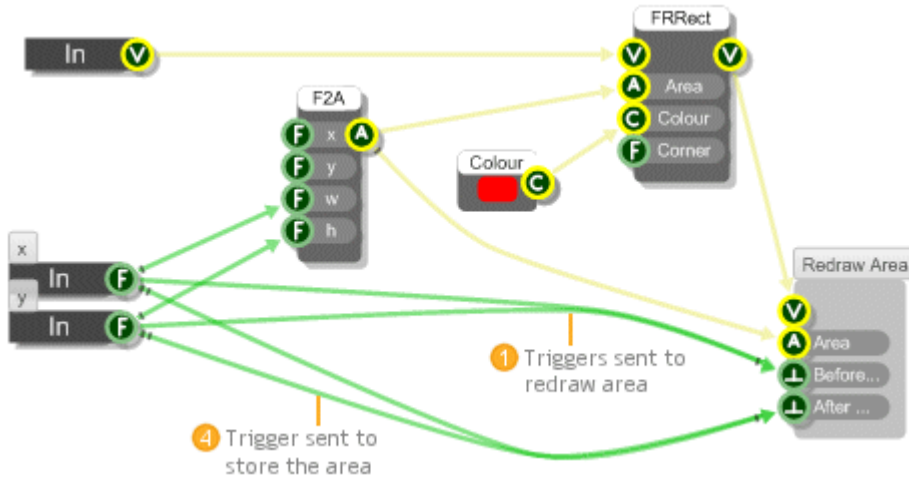


Precision Redraws

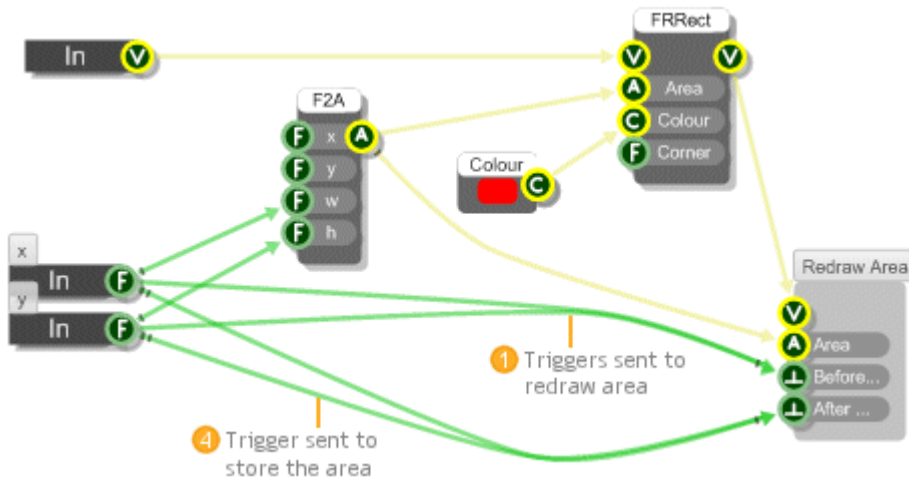
Redrawing the whole panel each time can be slow when the area is large (try resizing the module in the example above so that it's very big - you'll notice that dragging becomes sluggish).

Often only a small area of the panel is changing at any one time so it's much more efficient to redraw only the bit that has changed. For this purpose we have the Redraw Area component. This works in exactly the same way as the Redraw component except that it only redraws the area that you supply to it's Area input.

We've modified the drag rectangle example so it uses the Redraw Area component. Unfortunately it's not just a simple case of linking the rectangle area to the Redraw Area component. This is because the area before the last mouse move may need to be redrawn too. What we need is the combined area of the old and the new rectangles.

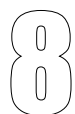


For this example we've created a module that handles the storage of the old area and its combination with the new area. This uses an Area Sample and Hold to keep the old Area for when it's needed.



Here's a link to the modified schematic:

<http://www.FlowStone.com/docs/pics/overview/drag%20rectangle%20example%20redraw.osm>



Code Component

THE ULTIMATE LOW-LEVEL DSP TOOL

DSP Coding

For most tasks, FlowStone's graphical programming approach meets your needs nicely. However, when it comes to programming DSP a more algorithmic approach is required. For this purpose we have the Code Component.

Using the code component you can do any kind of processing you like. The component uses a very small set of commands to allow you to translate DSP algorithms into very simple code. The code is then dynamically compiled into binary that runs extremely fast for maximum performance.

The Code Component

You'll find the code component in the Code group of the toolbox. The component provides an area into which you can type your code. Click on the component to go into edit mode. You'll stay in edit mode until you click or tab away.

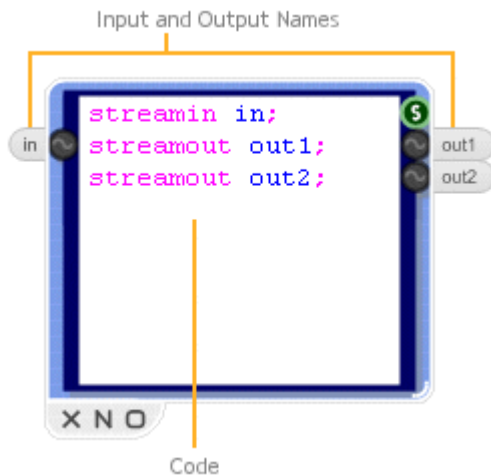
Inputs and Outputs

In order to link the code into a schematic you'll need some inputs and outputs. The code component is used for audio signal processing so it only allows you to create Poly, Mono, Stream or Stream Boolean inputs and outputs. There are eight commands for adding inputs and outputs:

```
polyin, polyout, monoin, monout, streamboolin,  
streamboolout, streamin, streamout
```

You need to make sure that you give each input and output a name and that each line of code is terminated with a semicolon. Once the correct syntax has been applied, the input or output will appear automatically when you click or tab away.

Stream boolean inputs/outputs allow you to process or receive boolean mask data. See the sections on Expressions and Conditional Statements for more information on masks.



The input/output names become variables that you can use in subsequent code. The variables will take whatever values come in or go out when you link the component to other parts of a schematic. If nothing is linked to an input, the variable will assume a value of zero.

One final point worth noting is that it's a good idea to use streamin and streamout instead of the poly and mono alternatives as your code can then be used in both poly and mono sections.

Syntax Colouring

The editor colours the code to indicate correct syntax. Input/Output commands, operators and brackets are purple, variables are blue and data types and functions are green.

If at some point the syntax colouring stops and all text after that point turns black, this is an indication that there is an error in the syntax at that point.

Editor

The code editor supports copy and paste through the standard shortcut keys (CTRL+C, CTRL+V). A local undo is also implemented to allow you to undo and redo typing, deletions, pastes etc. The local undo applies to any changes you make during a particular session (Between clicking in the component and clicking away). After that you can still undo via the application's undo system, but this will only go back through changes made between edits.

You can scroll using the mouse wheel or use the cursor keys. You can page using the PGUP and PGDN keys. CTRL+HOME will go to the top and CTRL+END will go to the bottom.

Local Variables

In addition to the data you draw from your schematic via inputs you can also create local variables. These are defined in exactly the same way as the inputs and outputs.

There is only one type of variable at the moment and that's a float. This represents a floating point number. As with the inputs and outputs you need to give the variable a name and terminate the command with a semicolon.

Variables can be initialised at the same time as you declare them. You can write:

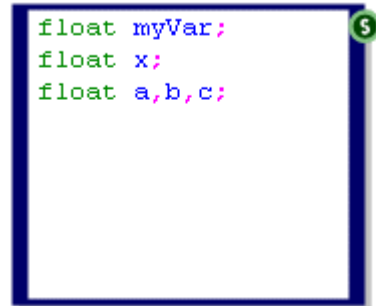
```
float x=3;
```

And x will be assigned the value 3. Variables that are not initialised explicitly will be set to zero.

You can declare multiple variables in a single line of code. For example:

```
float x,y,z;
```

Will create three variables x,y and z.



```
float myVar;
float x;
float a,b,c;
```

Assignments

Having set up your inputs, outputs and local variables you can then move on to implementing your DSP algorithm.

To assign a value to a variable or to send it to an output, use the equals operator =. You can assign fixed values or you can use a regular expression to calculate the value.

```
streamin in;
streamin cutoff;
streamin res;
streamout out;
float out1,out2,out3,out4;
float in1,in2,in3,in4;
float f,fb,f2,input;
input=in;
f=cutoff*1.16;
f2=f*f;
fb=res*(1.0-0.15*f2);
input=input-out4*fb;
input=input*0.35013*f2*f2;
out1 = input+0.3*in1+(1-f)*out1;
in1=input;
out2=out1+0.3*in2+(1-f)*out2;
in2=out1;
out3=out2+0.3*in3+(1-f)*out3;
in3=out2;
out4=out3+0.3*in4+(1-f)*out4;
in4=out3;
out=out4;
```

CHAPTER 8

```
streamin in;  
streamout out;  
float gain;  
  
gain = 2.0; — Fixed Value  
  
out = in * gain;  
      └───┬───┘  
      Expression
```

The example opposite shows how you can combine all the elements discussed so far to create a simple moog filter.

Expressions

Expressions can use a combination of mathematical operations and built in functions. The following represents a complete summary of the operators and functions that are currently supported.

<code>sin1(a)</code>	Mathematical sine, cosine and tangent
<code>cos1(a)</code>	The 1 indicates that the input value <code>a</code> is converted to a value in the range 0-1 and not 0 and 2π . So 0.5 would be equivalent to π and 1.25 would be equivalent to $\pi / 2$
<code>tan1(a)</code>	
<code>log10(a)</code>	Logarithm (base 10) of <code>a</code>
<code>max(a,b)</code>	The numerically largest (max) or smallest (min) of <code>a</code> and <code>b</code>
<code>min(a,b)</code>	
<code>rndint(a)</code>	Rounds <code>a</code> to the nearest integer value above or below (so 1.4 becomes 1 but 1.6 becomes 2)
<code>*</code> <code>+</code> <code>-</code> <code>/</code>	Standard mathematical operations
<code>%</code>	Calculates the remainder after dividing the left-hand side by the right-hand side
<code>></code> <code>>=</code> <code><</code> <code><=</code>	Comparison operators These generate a mask based on the result of the comparison
<code>&</code>	Bitwise AND operator. Use this to apply a mask

One limitation of the code component is that you can only have 8 successive operations in an expression. For example, the expression:

```
a = 0+1+2+3+4+5+6+7+9+10;
```

is not allowed because there are 9 addition operations in a row. You can easily get round this by splitting the expression up into smaller sections using brackets '(' and ')'. For example:

```
a = (0+1+2+3+4) + (5+6+7+9+10);
```

Conditional Statements

Because FlowStone uses SSE to process four channels at a time, conditional statements can't be implemented in the traditional way. However, you can achieve the effect of a conditional statement by using a mask.

A mask can be applied to all four channels at the same time. Each channel is affected differently. FlowStone includes conditional operators that generate exactly these kind of masks.

For example, the following FlowStone code:

```
x = x - (x >= 1) & 1.0;
```

is equivalent to the following C/C++ code:

```
if( x >= 1 ) x = x - 1.0;
```

The >= operator creates a mask. The mask will effectively be true for each channel that meets the condition and false for each one that doesn't. The bitwise & will return a value of 1.0 or 0.0 for each SSE channel based on the mask.

Comments

You can leave comments in your code. These are just snippets of text that remind you of what a particular part is doing. Comments take up no processing time. They can be put at the end of a line or they can have a whole line to themselves

For example:

```
index = index + 1; //Increment index
//Increment index
index = index + 1;
```


Advanced Features

Arrays

You can declare arrays in the code component. These act as buffers that you can use for storing past values for example or for using as a lookup table.

An array is declared as follows:

```
float buffer[100];
```

This defines an array called buffer which is 100 floats (4x100=400 bytes) in size.

Initialising

By default all the entries in the array are set to zero. You have two other options for initialising an array:

1. Set all entries to a particular value

```
float buffer[100] = 0.62;
```

2. Randomise the contents of the array

```
float buffer[100] = rand(-0.5, 0.5);
```

Accessing

To access a particular entry in an array use the array specifiers (square brackets) '[' and ']'.
out = buffer[25] ;

```
buffer[6] = 3.14159;
```

Note that the array index is zero based so in the example above buffer[0] would be the first entry and buffer[99] would be the last entry in the array.

You must make sure that you use indexes that are within range for the array, failure to do so will produce unpredictable results and could even crash the software.

Hop

There are some calculations that don't need to be calculated for every sample, it's sufficient to recalculate for every 4th sample say. For this purpose we have the hop command.

The syntax for the hop command is as follows:

```
hop (8)
{
    < code in here is executed only for every 8th sample >
}
```

Hops must be powers of 2 and the maximum hop length is 4096 so values of 2,4,8,16,32,64,128,256,512,1024,2048 and 4096 are the only acceptable values.

Loop

Sometimes you need to execute the same code several times, changing one or two parameters on each pass. You can do this in the code component using the loop command.

Here's the syntax:

```
loop (10)
{
    < code in here is executed 10 times in succession >
}
```

You could use a loop to initialise an array for example:

```
float index;
float buffer[10];
loop (10)
{
    buffer[10] = sin1(index);
    index = index+0.1;
}
```

You should take care not to use too many loops or loops with many iterations as they can cause significant increases in cpu. Also for some loops (like the array initialisation example) you only want them to be executed once, this is where the stages come in.

Stages

The code is split into four parts called stages. These are numbered 0,1,2 and 3.

Stage(0)

Stage(0) runs only for the first sample and is intended for processing that you only want to happen at the very start, usually to set up other parameters that you will then use on every sample.

For the example we've just been looking at we could put the array initialisation into Stage(0) and it will run only once.

Defining the stage is very easy:

```
stage (0)
{
    < code in here is executed only for the first sample >
}
```

Stage(2)

We'll jump to Stage(2) now because this is the default stage. Any code written outside of a Stage definition is assumed to be Stage(2).

Stage(2) code is executed for every sample (including the first) and happens after Stage(0).

Stage(1) and Stage(3)

These two stages are only used for implementing delays. They are needed to make sure that data flows correctly when one or more delays are chained together in a feedback loop.

The software executes the code in stage order so after Stage(0) Stage(1) goes next.

Stage(1) should be used to move data from internal variables (most likely a buffer) and the outputs. This ensures that all delays have a consistent output value before doing any calculations.

Next the standard Stage(2) is executed. Any code that needs the outputs of any delays will have the values they need now.

Finally Stage(3) is executed. This should be used to move data from the inputs to internal variables (again most likely a buffer) and perform any calculations.

Note that each stage is executed for the whole of the schematic before proceeding to the next stage.

An example of how to use Stage(1) and Stage(3) for a delay is shown below.

```

streamin in;
streamout out;
streamin delay;
float mem[44100];
float index;
stage(1)
{
    out = mem[index];
}
stage(3)
{
    mem[index] = in;
    index = index + 1;
    index = (index < delay) & index;
}

```

Note that because the transfer to outputs in Stage(1) occurs before the buffer update in Stage(3) this will always produce a minimum of 1 sample delay. If you need a delay that will work correctly when the input delay length is zero samples then you can implement this by modifying the above code as follows:

```

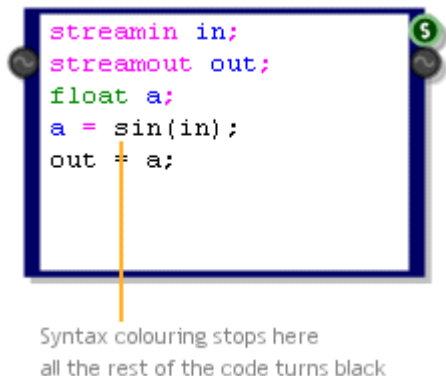
stage(1)
{
    out = mem[index] & (delay > 0) + in & (delay == 0);
}

```

Debugging

Using Syntax Colouring

We've already seen that the first indication you'll have that your code is incorrect will be a discontinuity in the syntax colouring. The following example shows this.



You can see that the colouring stops with the sin function. This indicates that there is something wrong with the expression. Sure enough we forgot the '1', the line should read:

```
a = sin1(in);
```

If we make this correction everything becomes coloured correctly.

Checking the Assembler

The code component converts the high-level language that it uses into x86 assembler prior to compiling. You can look at this assembler code by connecting a Text component to the String output of the code component.

If you understand assembler (and even if you don't) it can be useful to examine this code to make sure that nothing out of place is happening.

When there is a syntax error the assembler code will only be generated up to the point at which the syntax broke down.

If the syntax is correct then the other possible problem could be the limitation on expressions discussed earlier. This is much more difficult to spot in the code because the colouring will indicate correct syntax. However, a quick look at the assembler code can identify this type of problem.

The example below shows exactly this. The way to spot the problem is to look for xmm999 in the code. If this happens then you know to look at your expressions to make sure that the 8 consecutive operator limit has not been exceeded.



The presence of an xmm999 indicates that you have an expression that combines more than 8 consecutive operations

9

Options

CUSTOMISING THE APPLICATION

The Options Dialog

There are a number of application and schematic level options/features that you can change in order to make the software work the way you want it to. You can get at these by going to the Options menu and selecting one of the six categories under which the various parameters are organised.

Application

These options apply to features and settings that apply at application level.

Application

Startup Options

Begin using FlowStone with:

Example schematic

Most recent schematic

Empty schematic

Most Recent File List

Display at most files

Auto Recovery

Save data every secs

Automatic File Compression

Your schematic files can be compressed automatically to reduce their size. With very large schematics this can begin to affect save times so if you are storing large quantities of wave or image data you can disable this feature

Check for updates on startup

Startup Options

By default the software will load up the example file that ships with the software. This is fine when you first start looking at the software but once you begin making your own creations this behaviour becomes undesirable.

Thankfully you can switch this off. You can choose between the software launching with a blank schematic (the fastest way to load up) or the schematic that you were working on last time you closed down.

Most Recent File List

The most recently used schematics are shown at the bottom of the File menu for rapid access. The default size of this list is 4 items but you can choose to have up to 16 schematics displayed. You can also choose to clear the recently used file list by clicking the Clear List button.

Auto Recovery

Auto Recovery is a handy little feature that saves a copy of your schematic periodically so that if your session should suddenly be closed down due to a crash or loss of power you won't lose your work. Instead, when you next launch FlowStone the file(s) you were editing will be recovered based on the state when they were last saved.

You can switch auto recovery on and off at any time. By default the save period is every 5 seconds. For larger schematics you may want to choose a longer save interval to allow for the increased save time.

Recovery files are saved to the App Data folder on your system. If you need to access this folder you can do so by clicking the Open Recovery Folder button.

Automatic File Compression

Schematic files are automatically compressed to keep file sizes down. With very big files that contain large amounts of image or sample data you may find that your save times begin to increase. In these cases you can reduce the save time by switching file compression off.

Check For Updates On Startup

FlowStone is updated regularly so the software can check whether you have the latest version and notify you that there is an update waiting. If you prefer not to do this you can switch the behaviour off. You can also manually check for updates by clicking the Check Now button.

Please be assured that your privacy is maintained at all times. No data is transmitted during the update check and only the latest version information is download to your PC from our server.

Navigator

Navigator

Performance

Navigator updating:

Frequent (slowest) Selective Minimal (fastest)

Navigation Keys

PGUP moves out of module, PGDN moves into module

PGUP moves into module, PGDN moves out of module

Performance

When you change a front panel item like a knob or a slider this often results in values changing in other parts of your schematic. Any modules that are shown in the navigator will be updated to reflect these changes. On some PCs this can be very slow.

You can therefore choose how often the Navigator updates. The default is for minimal updates and often this is all you need. If you want to see the results of your interactions reflected in just the current module in the navigator then choose Selective. To make the Navigator update on every change select Frequent.

Navigation Keys

Here you can choose whether the PGUP and PGDN keys move you up and down through the module hierarchy or down and up.

Schematic

Schematic

Mouse Behaviour

<p>Drag Selection:</p> <p><input checked="" type="radio"/> Enclose components</p> <p><input type="radio"/> Touch components</p>	<p>Right-click:</p> <p><input type="radio"/> Moves to parent module</p> <p><input checked="" type="radio"/> Shows context menu</p>
---	--

Zoom Level

Default zoom level pixels per grid square

Auto Linking

Suggest links when grid squares or fewer between connectors

Create suggested link(s) after milliseconds

Only suggest links

Links on Top

Determines whether links are displayed on top of or below the components

Mouse Behaviour

When drag selecting you can choose whether components will become part of the selection once they are fully enclosed in the drag rectangle or when they touch some part of the area covered by the drag rectangle.

You can decide whether a right-click brings up the standard context menu or if moves you up to the parent when inside a module.

Zoom Level

The default zoom level is defined as a number of pixels. This represents the size of a single grid square so a higher number will result in a default zoom that appears magnified. If you click Use Current you'll get the zoom level for the current module.

Auto Linking

You can decide how close components have to be before the software will suggest an automatic link. You can also choose whether links are then created after a short time or if the software waits for you to right-click to accept a suggested link.

Links On Top

Check this box if you want links to appear on top of components in your schematic (applies to the schematic as a whole).

Modules

Modules

Animation

Enable module animation
Allows you to animate module resizing as you change between front panel, minimized state and properties

Animation Speed:

Slower Faster

Properties

Auto Close
Select this option if you want module properties to close automatically when the module is deselected

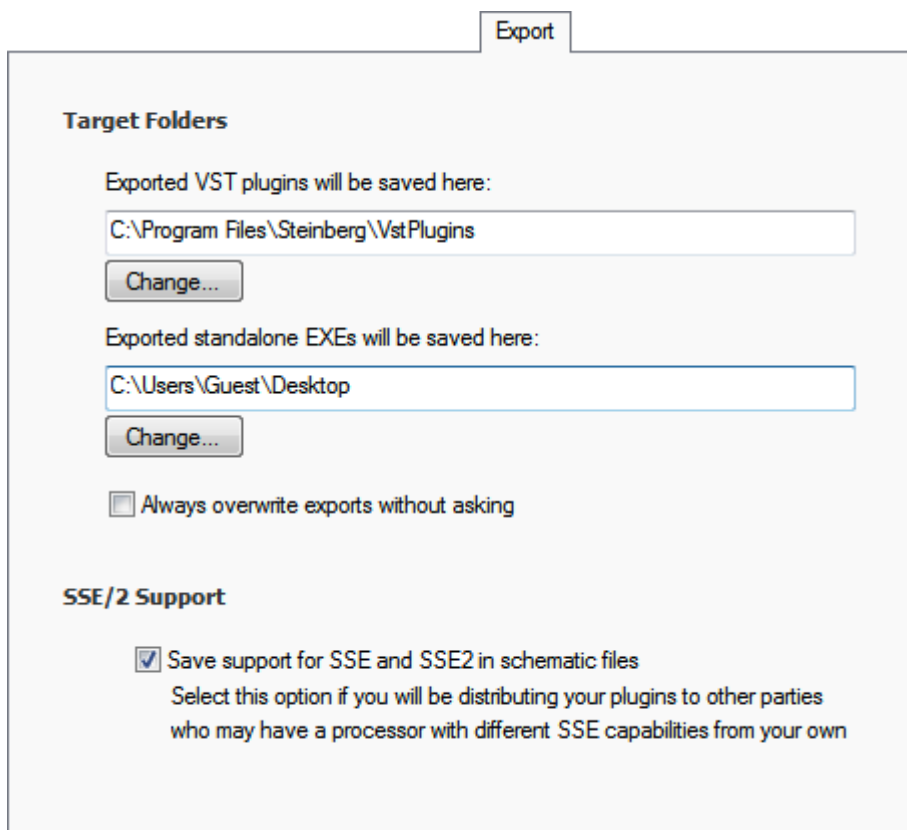
Animation

You can choose whether to enable or disable module animation for when you minimize or maximize a module or open/close the properties panel.

Properties

By default a module's properties panel will close as soon as you click away (provided it isn't pinned open). You can switch this off so that the properties remain open until you close them.

Export



Target Folders

When you export standalone exes they are saved to a particular folder. You can set this here.

SSE/2 Support

The PC on which you generate your exports may support SSE2 or it may just support SSE. When you export the software saves code that is optimised for the SSE capabilities of your own PC. If you then give the export to someone whose PC has different SSE capabilities from you the software needs to make adjustments on loading so that the export will work.

These adjustments can slow down the loading process. To avoid this you can choose to include support for all SSE setups at export time. This way there is no reduction in loading speed when exports are used on PCs with different SSE capabilities.

Advanced

Advanced

R&D Components

Show any research and development (R&D) components in the toolbox
 If you try these components please let us know what you think of them.
 Be aware that such components are still under development and may change or be dropped at any time

Link Curvature

Straight

 Curved

Code Compilation

Compile as you type in the code component
 Only enable this if you need to see or hear the effect of your code changes as you are making them

Maximum code size: bytes

SSE/2 Support

Save support for SSE and SSE2 in schematic files (results in slower saves)
 Select this option if you will be distributing your schematics to other parties who may have a processor with different SSE capabilities from your own

Floating Point Numbers

Display floats as 32 bit binary representations

R&D Components

We will periodically offer some components on a trial basis to gather feedback or for testing. You can decide whether to show these components by checking the box.

Link Curvature

When you bend links they take a curved form. You can decide how curved you want them to be from highly curved to perfectly straight lines. Note that this affects the way that links look in any schematic you open.

Code Compilation

In order to increase performance code compilation is switched off while you are typing in a code component. However, sometimes you want to be able to see the effects of your code as you type.

SSE/2 Support

The PC on which you create your schematics may support SSE2 or it may just support SSE. When you save a schematic the software saves code that is optimised for the SSE capabilities of your own PC. If you then give the schematic to someone whose PC has different SSE capabilities from you the software needs to make adjustments on loading so that the schematic will work.

These adjustments can slow down the loading process. To avoid this you can choose to include support for all SSE setups at save time. This way there is no reduction in loading speed when schematics are used on PCs with different SSE capabilities.

You only need to enable this option for schematics if you intend to distribute them. One side affect of enabling this option is that your save times are increased.

Floating Point Numbers

In computing floating point numbers cannot represent every possible number using 32bits. The system therefore has to approximate some numbers.

For example, 0.01 is represented as 9.9999998×10^{-3}

These small inaccuracies can sometimes lead to unexpected results in calculations because FlowStone will display 0.01 as 0.01 when in fact the number being used is of course the floating point approximation.

These situations are quite rare but if you are doing precise calculations where these inaccuracies may be significant in your results then you can choose to display the underlying representations instead so that all your calculations are completely transparent.