THE NEW GRAPHICAL PROGRAMMING LANGUAGE

FLOW STONE

DSP Robotics

Ruby Inside

COMPONENT REFERENCE

VERSION 3.0.2

1 Introduction

About this guide

About This Guide

In this guide you'll find individual descriptions for all the components supplied with FlowStone. It is intended as reference material to accompany the main user guide.

The components are listed in ascending alphabetical order. If you want to look up a particular component quickly use the table of contents at the beginning of this guide.

Other Information

We have a separate guide which describes how to use the software. This can be found in the Manuals section of our web site at:

http://www.dsprobotics.com/manualsarea.php

If you are looking for tutorials then see the Tutorials section of the DSP Robotics web site:
http://www.dsprobotics.com/tutorials.html

Additional information and articles about the software can be found at:
http://www.dsprobotics.com/support

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

2 Components

A-Z Listing of All Primitives & Modules

Add

Description

This component adds two values together.

Connectors

Inputs	Type	Outputs	Type
Input 1	Template	Sum of inputs	Template
Input 2	Template		

Add to Float Array

Description

This primitive adds a single float value to every entry in the input array.

Connectors

Inputs	Type	Outputs	Type
Array to modify	Float Array	Array of added values	Float Array
Float value to add to each entry in the array	Float		

After Duplicate

Description

The After Duplicate component sends a trigger after a schematic has been duplicate. You can use this to do any initialisation that may be needed after dragging a module from the toolbox or pasting it or any other operation that involves duplication.

Connectors

Inputs	Type	Outputs	Type
N/A		Trigger when the containing module has been duplicated or reproduced	Trigger

After Load

Description

The After Load component sends a trigger after a schematic has been loaded. You can use this to do any post loading initialisation that may be needed.

Connectors

Inputs	Type	Outputs	Type
N/A		Trigger when schematic has just completed loading	Trigger

Append Array

Description

The Append Float Array, Append String Array and Append Int Array component will append the contents of two arrays together resulting in one single array which contains the contents of both arrays.

Connectors

Inputs	Type	Outputs	Type
First array	Float Array	Array containing values from both arrays	Float Array
Second array	Float Array		

Arc

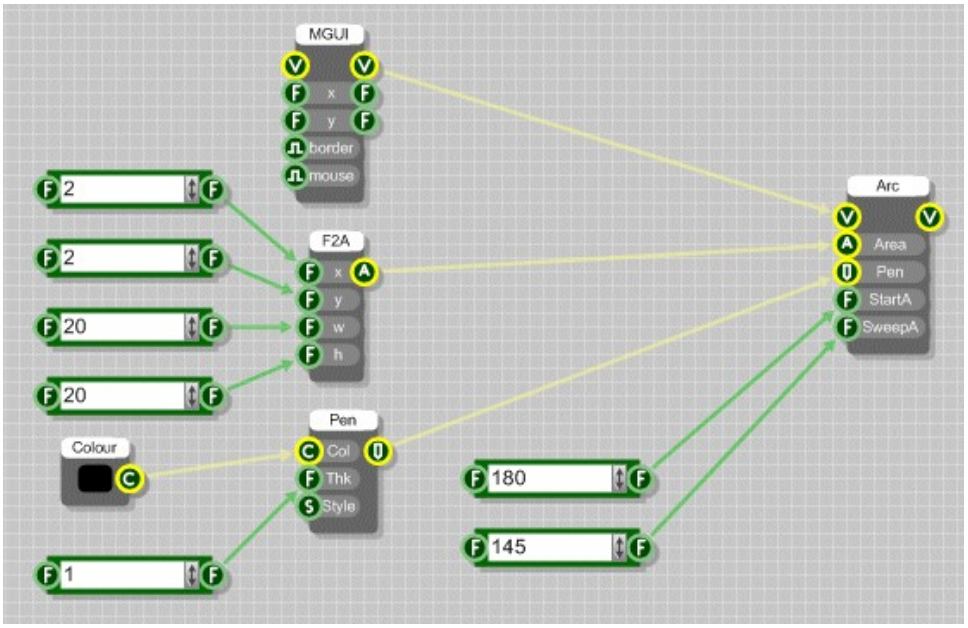
Description

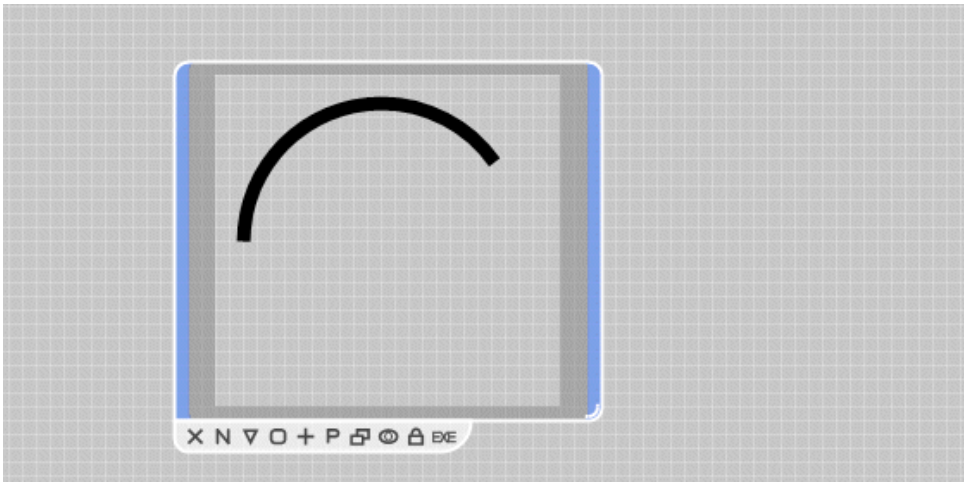
The Arc primitive draws a portion of the edge of a circle or ellipse depending on whether the bounding area is square or rectangular (Technically an arc is a portion of the circumference of a circle but here it is extended to ellipses as well). The arc is defined by the bounding area in which it is to be drawn, a starting angle and an angle to indicate the amount of sweep (both in degrees), as well as by the pen object to be used to draw the segment.

Connectors

Inputs	Type	Outputs	Type
View to draw onto	View	The same View as the input, but anything connected here is drawn on top	View
The bounding area of the arc. (Note: the pen line will extend beyond this area by one half its width)	Area		
The pen used to draw the segment	Pen		
The Start Angle (in degrees). This starts from the 9 ‘o’ clock position and runs clockwise	Float		
The Sweep Angle (in degrees) runs clockwise and defines the length of the segment	Float		

Example


MGUI
x
y
border
mouse
2
2
20
20
F2A
x
y
w
h
Colour
Pen
Col
Thk
Style
1
180
145
Arc
Area
Pen
StartA
SweepA



Area to Float

Description

The Area to Float component splits an Area into X, Y, Width and Height.

Connectors

Inputs	Type	Outputs	Type
Area to split	Area	X component	Float
		Y component	Float
		Width component	Float
		Height component	Float

Area Union

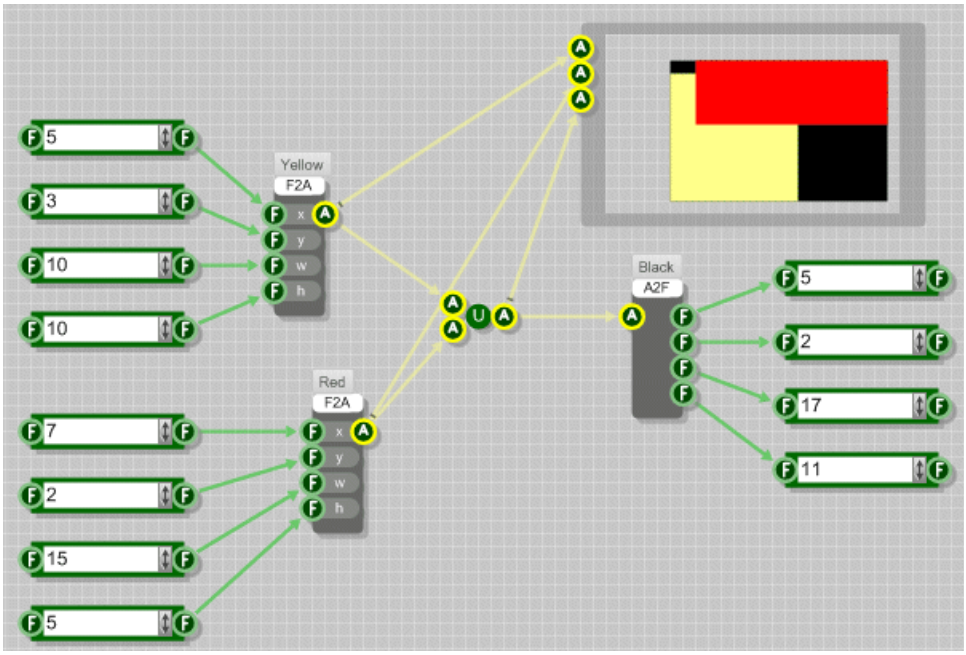
Description

The Area Union primitive takes two area inputs and finds the smallest bounding rectangular area in which both will fit.

Connectors

Inputs	Type	Outputs	Type
First area	Area	Union of the two areas	Area
Second area	Area		

Example



Array Builder

Description

The Array Builder components provide you with a more visual way of creating an array of Strings, Floats or Ints. This component is useful for small to medium sized arrays. It can also make it easier to manage arrays where the elements are frequently changing value.

The inputs define the values at each index in the array. So connecting a value to the first input will set the first entry in the array. As this is a template connector the first link you make will also determine the data type (String, Float or Int).

To add subsequent entries simply connect a value to the 'spare' template connector. This will become a connector of the appropriate type and a new 'spare' will appear below it. By connecting more inputs in this way you can quickly build up an array.

Connectors

Inputs	Type	Outputs	Type
The first entry in the array	String / Float / Int	The array itself	String Array / Float Array / Int Array
Any number of other inputs	String / Float / Int or template if 'spare'		

ASCII to String

Description

The ASCII to String primitive converts an integer into its corresponding ASCII character (or more accurately the ISO Latin 1 character as ASCII is only defined through 127). For values above 255 the output 'wraps' back to zero (so the character returned will equal the modulus remainder dividing by 256).

Connectors

Inputs	Type	Outputs	Type
ASCII character code (0-255)	Int	Corresponding ASCII character	String

ASIO In

Description

ASIO (Audio Stream Input Output) is a digital audio protocol specified by Steinberg which provides an interface between an application and the sound card. The ASIO In primitive provides a mono connection for each mono input supported by the hardware sound device on your computer.

The ASIO In components provide the only way to receive audio signals from an external source (via inputs on your sound card). Note that only one ASIO In component is allowed in your schematic.

Connectors

Inputs	Type	Outputs	Type
N/A		One output for each channel supported by the selected audio device	Mono

Other Features

The body of the component displays which input device is currently being used. All devices supporting the ASIO protocol are listed so to select a different one just click on it.

To deselect a device (and therefore switch ASIO input off) simply click on it again.

A small spanner (wrench) icon allows quick access to the device driver's user interface.

ASIO Out

Description

ASIO (Audio Stream Input Output) is a digital audio protocol specified by Steinberg which provides an interface between an application and the sound card. The ASIO Out primitive provides a mono connection for each mono output supported by the hardware sound device on your computer.

The ASIO Out and Direct Sound Out components provide the only way to send audio signals to your sound card. You therefore must have at least one of these connected up to your schematic if you want to hear any sound. Note that only one ASIO Out component is allowed in your schematic.

Connectors

Inputs	Type	Outputs	Type
One input for each channel supported by the selected audio device	Mono	The assembler code generated by the component	String

Other Features

The body of the component displays which output device is currently being used. All devices supporting the ASIO protocol are listed so to select a different one just click on it.

To deselect a device (and therefore ASIO output off) simply click on it again.

A small spanner (wrench) icon allows quick access to the device driver's user interface.

Assembler

Description

The Assembler component allows you to write low-level x86 assembler code and use it in your schematic. There are special commands for creating inputs and outputs so that you can connect the Assembler component to other components. See the Code Component section in the main user guide for more information.

NOTE: This is a very advanced feature and should be used with care. With such low-level control it is possible to crash the software.

Connectors

Inputs	Type	Outputs	Type
N/A		Compiled code – attach to a Text component to view	Code

Supported Instructions

The assembler component only supports the following subset of the x86 instruction set:

```
add reg,reg;                  add reg,integer;              add reg,var;
addps xmmReg,sseVar;          addps xmmReg,xmmReg;          andps xmmReg,xmmReg;
andnps xmmReg,xmmReg;         andps xmmReg,sseVar;          and reg,integer;
call reg;                     cmp reg,integer;              cmp reg,reg;
cmpps xmmReg,sseVar,type;     cmpps xmmReg,xmmReg,type;     cvtps2dq xmmReg,sseVar;
cvtdq2ps xmmReg,sseVar;       divps xmmReg,sseVar;          divps xmmReg,xmmReg;
fistp sseVar[channel];        fist sseVar[channel];         fild sseVar[channel];
fld [reg];                    fstp [reg];                   fld sseVar[channel];
fld sseVar[channel];          fstp [reg];                   fld sseVar[channel];
fld sseVar[eax];              fsin;                         fsub;
fsincos;                      fptan;                        fstp sseVar[channel];
fstp sseVar[eax];             fst sseVar[channel];          fxch;
fmul;                         fadd;                         fprem;
frndint;                      fldlg2;                       fyl2x;
inc [reg];                    jnz integer;                  jnz label;
jz label;                     maxps xmmReg,sseVar;          minps xmmReg,sseVar;
mov reg,sseVar[channel];      mov sseVar[channel],reg;      mov reg,reg;
mov reg,integer;              mov [reg],integer;            mov reg,[reg];
mov eax,[ebp+integer];        mov [reg],reg;                minps xmmReg,[eax];
movaps [eax],xmmReg;          movaps xmmReg,[eax];          movaps xmmReg,xmmReg;
movaps xmmReg,sseVar;         movaps xmmReg,sseVar[eax];    movaps sseVar,xmmReg;
movaps sseVar[eax],xmmReg;    mulps xmmReg,sseVar;          mulps xmmReg,xmmReg;
pop reg;                      push reg;                     rcpps xmmReg,xmmReg;
rdtsc;                        shl reg,integer;              shr reg,integer;
subps xmmReg,xmmReg;          subps xmmReg,sseVar;          sub reg,integer;
fscale                        f2xm1                         fld1
fld st(N)                     fstp st(n);                   sqrtps xmmReg,xmmReg;
```

Audio Devices

Description

You can use the Audio Devices component to find out how many audio devices there are on the system you're running on. The first output tells you how many devices there are and the second output gives you a list of device names.

This component is used inside the Audio Selector module to provide the list from which you the choose an audio device.

Connectors

Inputs	Type	Outputs	Type
N/A		Number of devices available	Int
		Array of device names	String Array

Audio Select

Description

The Audio Select primitive allows you to select an audio device. It overrides whatever you have selected on the ASIO In/Out or DirectSound In/Out components.

This component is used inside the Audio Selector module and is used in conjunction with the Audio Devices primitive.

Connectors

Inputs	Type	Outputs	Type
Index of device you want to select	Int	Index of the currently selected device (or -1 if no devices are selected)	Int

Audio Stream

Description

This component allows you to stream audio from media file locally or across a network.

You must provide a valid URL or file path.

To begin streaming, trigger the Start input. The audio arrives as a stereo mono stream.

To toggle streaming trigger the Pause/Play input. Triggering the Stop input will end streaming. Having stopped, for an audio file, you can only resume by re-starting from the beginning.

Connectors

Inputs	Type	Outputs	Type
URL or file path for the audio stream	String	Left channel of streamed audio	Mono
Start streaming	Trigger	Right channel of streamed audio	Mono
Pause or resume play	Trigger	The play state (0=stopped, 1=playing, 2=paused)	Boolean
Stop streaming	Trigger	The play position (for files) as a percentage (0-100%)	Float
The input buffer size in seconds (default = 1)	Float		

Bar Start Pos

Description

When your VST plugin is used within a host this component will output the current songs bar start position (in 1 pulse (unit) per quarter). For more details see the VST SDK documentation.

Connectors

Inputs	Type	Outputs	Type
N/A		Bar start position	Stream

Binary to Hex

Description

Converts a string of binary to a string of hex. Each 8 bits of binary is converted to Ascii and then the hex representation of that byte is used in the hex string.

For example, the binary string “0010111110111011” is converted to the hex string “2FBB”.

Connectors

Inputs	Type	Outputs	Type
String of binary	String	String of hex	String

Biquad Filter

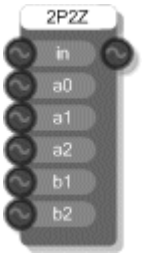
Description

A low pass biquad IIR filter using 2 poles, 2 zeros and 12dB per octave.

Connectors

Inputs	Type	Outputs	Type
Input signal	Stream	Output signal	Stream
Normalised cutoff frequency (0-1) where 1 is half sampling rate	Stream		
Resonance	Stream		

Biquad Filter Coeff



Description

A coefficient controlled biquad filter using 2 poles, 2 zeros and 12dB per octave specified by the filter coefficients.

Connectors

Inputs	Type	Outputs	Type
Input signal	Stream	Output signal	Stream
a0	Stream		
a1	Stream		
a2	Stream		
b1	Stream		
b2	Stream		

Bitmap

Description

The Bitmap primitive component loads a bitmap image from a file. Once the bitmap has been loaded the component retains the data, no reference to the original file path is required.

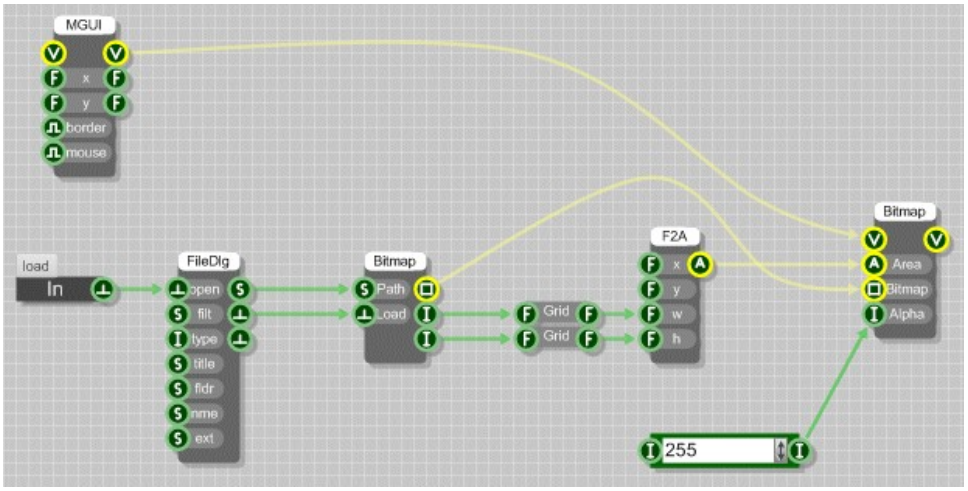
FlowStone supports bmp, jpeg, tiff, gif and png image types.

Connectors

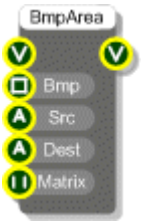
Inputs	Type	Outputs	Type
Path to image file on disk	String	The bitmap	Bitmap
Trigger to load the file	Trigger	Width of the image in pixels	Int
		Height of the image in pixels	Int

Example

The following example shows how to use the Bitmap component to load an image into a schematic.



Bitmap Area



Description

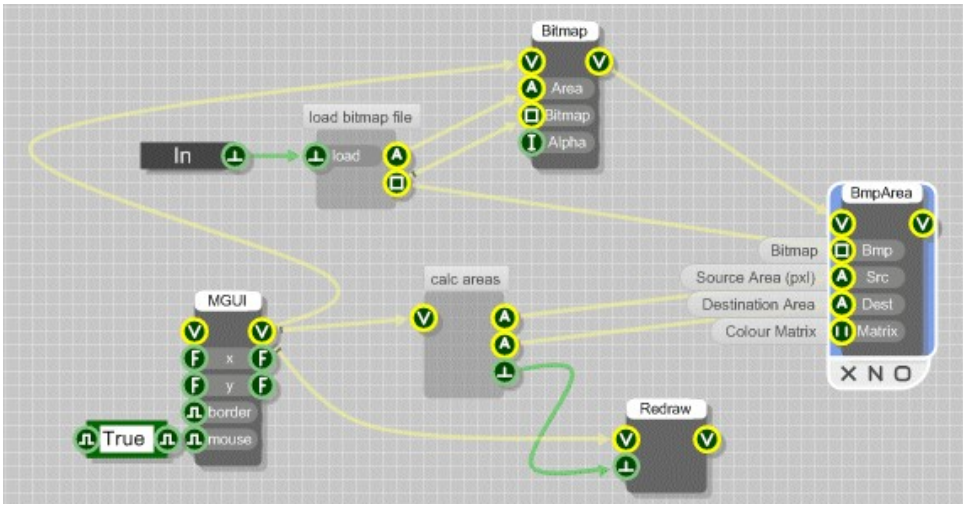
The Bitmap Area primitive draws a portion of a bitmap into a view. The bitmap is drawn to the destination area provided. If the destination area is a different size from the source area the bitmap will be resized to fit. The can be used, for example, to magnify some portion of a bitmap. There is also an optional colour matrix input that can be used to alter the resulting colours of the bitmap in the destination area.

Connectors

Inputs	Type	Outputs	Type
View to draw onto	View	The same View as the input, but anything connected here is drawn on top	View
Source bitmap to use	Bitmap		
Source area in the bitmap (in pixels, with the origin in the top-left corner).	Area		
Destination area on the View (in grid squares)	Area		
An optional colour matrix for performing colour transformations	Colour Matrix		

Example

The schematic file below is bitmap viewer with a small magnifying box that works by moving the cursor over the bitmap image. The mouse movements are tracked and a small segment directly over the mouse movement becomes the source bitmap area and is displayed in a destination are in the same location of the view but at a higher magnification.



Some calculations are required in order to achieve any interesting results as the source and destination areas must calculated and use different dimensions (pixels vs. grid squares). In the above schematic these calculations are hidden within the module labelled ‘calc areas’.

Bitmap Array from Bitmap

Description

The Bitmap Array from Bitmap primitive allows you to build an array of bitmaps from images that you already have loaded in your schematic. You can choose how the bitmaps are stored when they are saved with the schematic by selecting from one of five different image encoder types: bmp, jpeg, gif, tiff and png. You can also choose not to store the bitmaps with the schematic using ‘none’ as the encoder type.

Connectors

Inputs	Type	Outputs	Type
Bitmap to add to the array	Bitmap	Array of bitmaps	Bitmap Array
Index of the bitmap in the array to be copied in	Int		
Trigger to load the specified bitmap	Trigger		
Encoder type for storing the bitmaps when the schematic is saved (none, bmp, jpeg, gif, tiff or png). If ‘none’ is chosen then the bitmaps will not be saved with the schematic.	String		

Trigger to clear the array Trigger

Bitmap Array from File

Description

The Bitmap Array from File primitive allows you to build an array of bitmaps by loading them individually from files. You can choose how the bitmaps are stored when they are saved with the schematic by selecting from one of five different image encoder types: bmp, jpeg, gif, tiff and png. You can also choose not to store the bitmaps with the schematic using ‘none’ as the encoder type.

Connectors

Inputs	Type	Outputs	Type
Path to a bitmap file on disk	String	Array of bitmaps	Bitmap Array
Index of the bitmap in the array to be loaded	Int		
Trigger to load the specified bitmap	Trigger		
Encoder type for storing the bitmaps when the schematic is saved (none, bmp, jpeg, gif, tiff or png). If ‘none’ is chosen then the bitmaps will not be saved with the schematic.	String		
Trigger to clear the array	Trigger		

Bitmap Create

Description

The Bitmap Create primitive generates a bitmap and draws onto it whatever is attached to its View output. The bitmap can then be used as an input by other bitmap components.

Connectors

Inputs	Type	Outputs	Type
Width of the bitmap in pixels	Int	The generated bitmap	Bitmap
Height of the bitmap in pixels	Int	Connect GUI components to draw to the bitmap here	View
Grid Step to use when drawing to the bitmap	Float	Trigger when bitmap has been created	Trigger
Trigger to create the bitmap	Trigger		
Trigger to redraw the bitmap	Trigger		

Bitmap Draw

Description

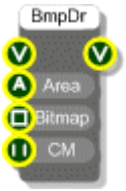
The Bitmap Draw primitive draws a bitmap onto a view. An Area must be supplied to define the part of the view that the bitmap is drawn into. The bitmap will be stretched if necessary to fit exactly into the area supplied.

You can also define a transparency for the rendered bitmap. You should take care when using this option for bitmaps that are likely to be redrawn many times per second as the calculation can be quite cpu intensive, particularly with larger bitmaps.

Connectors

Inputs	Type	Outputs	Type
View to draw onto	View	The same View as the input, but anything connected here is drawn on top	View
Destination area on the View (in grid squares)	Area		
The bitmap to draw	Bitmap		
Transparency level (0-255) where 0 is invisible and 255 is opaque	Int		

Bitmap Draw Transform



Description

The Bitmap Draw Transform primitive draws a bitmap onto a view. An Area must be supplied to define the part of the view that the bitmap is drawn into. The bitmap will be stretched if necessary to fit exactly into the area supplied.

You can also apply a colour transformation using a colour matrix. See the Colour Matrix component for more information. You should take care when using this option for bitmaps that are likely to be redrawn many times per second as the calculation can be quite cpu intensive, particularly with larger bitmaps.

Connectors

Inputs	Type	Outputs	Type
View to draw onto	View	The same View as the input, but anything connected here is drawn on top	View
Destination area on the View (in grid squares)	Area		
The bitmap to draw	Bitmap		
The colour transformation (if any) to be applied	ColourMatrix		

Bitmap Get At

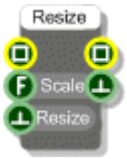
Description

This component extracts the bitmap at a particular index in an array of bitmaps.

Connectors

Inputs	Type	Outputs	Type
Array of bitmaps	BitmapArray	Extracted bitmap	Bitmap
Index of the bitmap to be extracted	Int	Trigger when the extraction is complete	Trigger
Trigger to do the extraction	Trigger		

Bitmap Resize



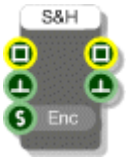
Description

This component resizes a bitmap by applying a specified scale factor. Depending on the scale factor the bitmap can be increased or decreased in size.

Connectors

Inputs	Type	Outputs	Type
Source bitmap	Bitmap	Resized bitmap	Bitmap
Scale factor to apply. Below 1 will reduce the size and above 1 will increase the size.	Float	Trigger when the bitmap has been resized	Trigger
Trigger to do the resize	Trigger		

Bitmap Sample and Hold



Description

The Bitmap Sample and Hold component will store a source bitmap when triggered. Any requests for the bitmap from the Bitmap output will return the last one that was stored.

You can choose how the bitmaps are stored when they are saved with the schematic by selecting from one of five different image encoder types: bmp, jpeg, gif, tiff and png. You can also choose not to store the bitmaps with the schematic using ‘none’ as the encoder type.

Connectors

Inputs	Type	Outputs	Type
Source bitmap	Bitmap	Stored bitmap	Bitmap
Trigger to store the current source bitmap	Trigger	Trigger when the bitmap has been stored	Trigger
Encoder type for storing the bitmaps when the schematic is saved (none, bmp, jpeg, gif, tiff or png). If ‘none’ is chosen then the bitmaps will not be saved with the schematic.	String		

Bitmap Save

Description

The Bitmap Save component saves a bitmap to a specified file path. You can choose from one of five different image encoder types for saving bitmaps: bmp, jpeg, gif, tiff and png.

Connectors

Inputs	Type	Outputs	Type
Source bitmap	Bitmap	Trigger when saving is complete	Trigger
Full path to the bitmap file you want to save to	String		
Trigger to do the save	Trigger		
Encoder type for storing the bitmap (bmp, jpeg, gif, tiff or png).	String		

Bitmap Size

Description

This component gives you the size of a Bitmap in pixels.

Connectors

Inputs	Type	Outputs	Type
Source bitmap	Bitmap	Width of bitmap in pixels	Int
		Height of bitmap in pixels	Int

Bool to False

Description

The Bool to False component sends a trigger whenever the value at the input changes from True to False.

Connectors

Inputs	Type	Outputs	Type
Boolean value to test	Boolean	Trigger on transition from True to False	Trigger

Bool to True

Description

The Bool to True component sends a trigger whenever the value at the input changes from False to True.

Connectors

Inputs	Type	Outputs	Type
Boolean value to test	Boolean	Trigger on transition from False to True	Trigger

Boolean

Description

The Boolean primitive stores a two-state value: either True or False. To change the value stored click on the main body of the component.

The component can be resized horizontally.

You can also change the type by right-clicking on the input or output. A pop-up menu will appear as shown below.

Simply click on the type you want to change to.

Connectors

Inputs	Type	Outputs	Type
Set the value	Boolean	The current stored value	Boolean

Boolean And

Description

Calculates the logical AND of two boolean values.

Connectors

Inputs	Type	Outputs	Type
First value	Boolean	AND value	Boolean
Second value	Boolean		

Boolean Or

Description

Calculates the logical OR of two boolean values.

Connectors

Inputs	Type	Outputs	Type
First value	Boolean	OR value	Boolean
Second value	Boolean		

Boolean XOr

Description

Calculates the logical XOR of two boolean values.

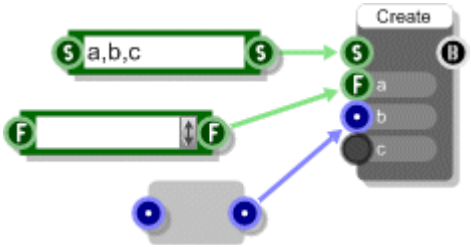
Connectors

Inputs	Type	Outputs	Type
First value	Boolean	XOR value	Boolean
Second value	Boolean		

Bus Create

Description

Allows you to combine several channels of data into just one. This can greatly simplify a schematic as data can be passed in just a single bus link. The channels are defined by a comma separated string which provides a name for each channel. You can extract data from a bus using the Bus Extract component.



When you connect the channel names you'll get a new template input for each channel. You then connect these up to whatever you want to pass through the bus. You can resize the Bus Create component so that longer connector labels can be read more easily.

Only the following types can be combined into a bus:

Trigger, Boolean, Float, Int, String, Float Array, Int Array, String Array, Stream, Stream Boolean, Poly, Poly Boolean, Mono, Mono Boolean, Mono4, Area, Pen, Colour, Colour Matrix.

Connectors

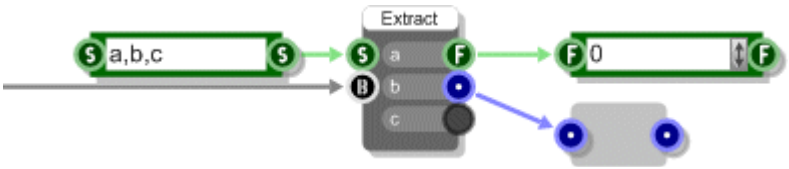
Inputs	Type	Outputs	Type
Comma separated list of channel names	String	Bus containing all the channels	Bus
connectors for each channel			

Bus Extract

Description

The Bus Extract component extracts one or more channels of data from a bus. Buses can be used to greatly simplify a schematic as data can be passed in just a single bus link. The channels are defined by a comma separated string which provides a name for each channel.

A bus is created by a Bus Create component.



When you connect the channel names you'll get a new template output for each channel. You then connect these up to whatever you want to pass through the bus. You can resize the Bus Extract component so that longer connector labels can be read more easily.

Only the following types can be combined into a bus:

Trigger, Boolean, Float, Int, String, Float Array, Int Array, String Array, Stream, Stream Boolean, Poly, Poly Boolean, Mono, Mono Boolean, Mono4, Area, Pen, Colour, Colour Matrix.

Connectors

Inputs	Type	Outputs	
Comma separated list of channel names	String	*connectors for each channel*	
Source bus	Bus		

CamShift Tracker

Description

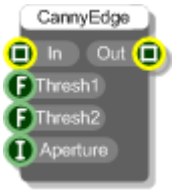
The CamShift Tracker is a video processing component that allows you to track an area of an image as it changes over time.

The tracker works by computing the histogram of the initial area and using this to compare against the image as it changes. The image is converted to HSV in the process and the smin and vmin values allow you to tweak how the S and V components are used in the algorithm.

Connectors

Inputs	Type	Outputs	Type
The image source	Bitmap	The current location of the area you are tracking	Area
The initial area of the image you want to track in pixel coordinates	Area	The rotation angle of the area you are tracking in degrees	Float
Starts tracking	Trigger	Tracking is in progress	Boolean
Stops Tracking	Trigger		
Minimum HSV saturation	Int		
Minimum HSV value	Int		

Canny Edge Detection



Description

This component finds the edges in an image using the Canny edge detection algorithm.

The Canny algorithm uses hysteresis and the Thresh1 and Thresh2 inputs define the high and low boundaries for this. They are in the range 0-255.

The Aperture is another input into the calculation. There are three options specified by an Int index, 0=3, 1=5 and 2=7. The default is 0 i.e. and aperture of 3.

Connectors

Inputs	Type	Outputs	Type
The source image you want to process	Bitmap	Processed grayscale image showing detected edges	Bitmap
First threshold for the hysteresis (0-255)	Float		
Second threshold for the hysteresis (0-255)	Float		
The aperture size. 0=3, 1=5, 2=7	Int		

Changed

Description

This simple primitive will only send data to the output when the input value changes to a different value. You can use this to reduce the flow of triggers through a schematic. For example, if you may be constantly calculating a result for display purposes but may only want to redraw the GUI when the value differs from what went before.

The component has template connectors which means it can be used with multiple data types including Floats, Ints, Strings, Booleans, Float/Int/String arrays and Areas. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
Input Value	Template	The input value if it has changed	Template

Clear Audio

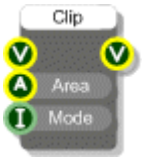
Description

The Clear Audio component simply resets all audio streams. You can use this as a panic option if your schematic uses feedback that might spiral out of control. It is also needed in some occasions to clear the audio buffers when changing settings or presets.

Connectors

Inputs	Type	Outputs	Type
Trigger to clear the audio	Trigger	N/A	N/A

Clip



Description

When you draw onto a View you draw inside a region called the Clipping Area. By default the clipping area of a View is defined by the bounding area of the module panel to which it applies.

The clipping area can be changed by applying one or more Clip components in sequence. Each Clip modifies the current clipping area by applying another area according to a particular clipping mode. The clipping modes are as follows:

0	Intersect	The clipping area becomes the intersection of the current clipping area and the new area
1	Union	The clipping area becomes the areas covered by the current area and the new area
2	Complement	The clipping area becomes the part of the new area that does not intersect with the current clipping area
3	Exclude	The clipping area becomes the area covered by the current area but not the new area
4	Xor	The clipping area becomes the areas covered by the current area or the new area, but not both

Connectors

Inputs	Type	Outputs	Type
Source View	View	View with modified clipping	View
Area to apply to clipping	Area		
Clipping mode	Int		

Colour

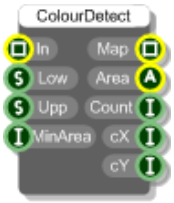
Description

This component allows you to define a colour. Click on the central colour swatch to change the current colour. The standard Windows colour dialog will appear. You can use this to select a colour.

Connectors

Inputs	Type	Outputs	Type
N/A		Currently selected colour	Colour

Colour Detect



Description

The Colour Detect component finds the areas in an image that match a particular colour.

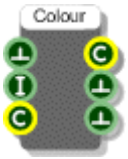
A colour range in HSV (Hue/Saturation/Value) format is used to define the regions of interest. You provide a lower and upper HSV. The algorithm will check the H,S and V components of pixels in the image against the upper and lower ranges for each component. H is in the range 0-360, S and V are in the range 0-255. An example input would be “360,255,255”.

The MinArea input allows you to specify a threshold for discarding detected areas based on their size. This input is an area in pixels^2.

Connectors

Inputs	Type	Outputs	Type
The source image you want to process	Bitmap	Processed image showing the detected areas	Bitmap
Lower HSV limit. Three numbers, comma separated	String	Bounding box of largest detected area	Area
Upper HSV limit. Three numbers, comma separated	String	Number of detected areas	Int
Threshold for discarding areas in square pixels	Int	X coordinate of centroid of largest detected area	Int
		Y coordinate of centroid of largest detected area	Int

Colour Dialog



Description

This component allows you to launch the standard Windows colour dialog and use it to get colour selections from users. You can supply an alpha blend value so that the returned colour is modified to have that value. You can also specify the default colour that the dialog shows when launched.

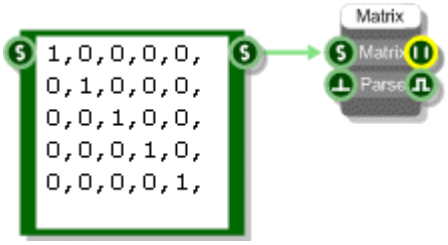
Connectors

Inputs	Type	Outputs	Type
Trigger to open the dialog	Trigger	Selected colour	Colour
Alpha blend transparency for the colour that is returned (0-255) where 255 is opaque and 0 is transparent	Int	Trigger if OK was pressed	Trigger
Colour to show when the dialog opens	Colour	Trigger if Cancel was pressed	Trigger

Colour Matrix

Description

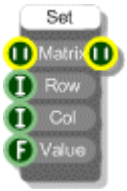
This component creates a Colour Matrix which you can use for performing colour transformations on bitmaps. The matrix itself is a 5x5 floating point matrix and is supplied by a comma separated string of values. The best way to define a colour matrix is to use a Text component as shown below.



Connectors

Inputs	Type	Outputs	Type
Comma separated string of matrix entries	String	The colour matrix	Colour Matrix
Trigger to parse the supplied data and create the matrix	Trigger	Whether the matrix was created	Boolean

Colour Matrix Set



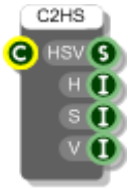
Description

The Colour Matrix Set component allows you to set the value for a particular entry in the matrix.

Connectors

Inputs	Type	Outputs	Type
Colour matrix to change	Colour Matrix	The modified colour matrix	Colour Matrix
Row index (0-4) of entry to be changed	Int		
Column index (0-4) of entry to be changed	Int		
The value to change to	Float		

Colour to HSV



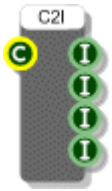
Description

The Colour to HSV component converts a colour to Hue, Saturation and Value components.

Connectors

Inputs	Type	Outputs	Type
Colour to convert	Colour	HSV as comma separated string	String
		Hue component (0-360)	Int
		Saturation component (0-255)	Int
		Value component (0-255)	Int

Colour to Int



Description

The Colour to Int component splits a colour into Alpha Transparency, Red, Green and Blue components.

Connectors

Inputs	Type	Outputs	Type
Colour to split	Colour	Alpha transparency (0-255) where 255 is opaque and 0 is transparent	Int
		Red component (0-255)	Int
		Green component (0-255)	Int
		Blue component (0-255)	Int

COM Port

Description

The COM Port component allows you to send and receive messages through a specified COM port.

Note: In the Free edition you are limited to using just one COM port at a time. In the Enterprise edition you can use up to 4 ports. In the Professional edition you can use as many as you like.

Having specified the characteristics of the Port you need to send a trigger to the Open input. The Open output will respond with True if this was a success. Any errors are reported through the Log output so long as you have the Log input set to true.

Data is sent as strings. Data is received as complete strings only if you specify a terminator. The terminator needs to be an ASCII code. So for example, specify zero for null terminated strings or 16 to use carriage return.

If no terminator is specified then the component will deliver received data one byte at a time. If the Hex input is set to True then this will be a two character hex code, otherwise it will be a single character.

You may find it useful to store this data in a String Queue for post processing.

Connectors

Inputs	Type	Outputs	Type
Index of the COM port you want to use. So 1 = COM1 and so on.	Int	Whether the COM port was opened successfully.	Boolean
Baud rate. If not specified then 9600 is used	Int	Data received in from the COM port	String
Stop bits (0 for 1, 1 for 1.5 and 2 for 2). Defaults to 1 stop bit.	Int	Trigger when data has been successfully sent through the COMport	Trigger
Parity (0=none, 1=odd, 2=even, 3=mark, 4=space). Default is no parity scheme.	Int	Log showing activity on the port (if logging switched on).	String
Byte size, the number of bits in each byte sent. If not specified then 8bits are used.	Int	Array of port numbers that are available on the host PC	Int Array
Open the COM port	Trigger		
Close the COM port	Trigger		
Data you want to send out through the port	String		
ASCII code of the terminator you want to look for when receiving data. If not supplied then data is output in single bytes.	Int		
Send the specified data through the port	Trigger		
Whether the data being sent and received is hex	Boolean		
Whether to log activity on the port. Useful for debugging.	Boolean		
Clear the log	Trigger		

Cos

Description

Standard trigonometric Cosine function with radians as the input units.

Connectors

Inputs	Type	Outputs	Type
Float value in radians	Float	Result of calculation	Float

Cos Inverse

Description

Standard trigonometric Inverse Cosine function with radians as the output units.

Connectors

Inputs	Type	Outputs	Type
Float value	Float	Result of calculation in radians	Float

Cosh

Description

Standard hyperbolic cosine function with radians as the input units.

Connectors

Inputs	Type	Outputs	Type
Float value in radians	Float	Result of calculation	Float

Counter

Description

The Counter component is a simple counter that increments in unit steps from zero every time the Inc input is triggered. You can reset the counter to zero by triggering the Reset input.

Connectors

Inputs	Type	Outputs	
Trigger to increment the counter	Trigger	The current value of the counter	Int
Trigger to reset the counter to zero	Trigger		

Counter Advanced

Description

This is a more advanced version of the simple Counter component. You set a Minimum and Maximum limit for the counter and also a place to start. The type input determines how to count:

0	Up	The counter starts from the minimum value, counts up to the maximum and then stops
1	Down	The counter starts from the maximum value, counts down to the minimum and then stops
2	Up/Down	The counter starts from the minimum value, counts up to the maximum and then counts back down again. The counting continues cycling between the two limits in this way.

De-zipper

Description

The De-zipper component is used for making smooth transitions between float values when feeding them into a stream section.

For example, say you have a knob connected to a stream multiplier for attenuating an audio signal. As you turn the knob the floating point value changes in steps over time. This stepping can be heard as background noise called zipper noise.

The De-zipper component removes zipper noise by applying a simple low-pass filter to input float values.

Connectors

Inputs	Type	Outputs	Type
Float values from a Float section of schematic	Stream	Float values as a smoothed signal	Stream
Duration of the transition	Int		

Decibel

Description

Converts an array of floats to decibels (10*Log10).

Connectors

Inputs	Type	Outputs	Type
Source array	Float Array	Array of dB values	Float Array

Delay

Description

The Delay primitive buffers incoming data and delays the output by the number of samples given in the integer input.

The delay length is limited to 262144 samples (just less than 6 seconds at 44.1 kHz). Longer delays can be implemented using the Code component.

Note that there is a minimum delay of 1 sample so even if the Delay input is zero there will be a single sample delay.

Connectors

Inputs	Type	Outputs	Type
Mono stream	Mono	Delayed Mono stream	Mono
Number of samples delay	Int		

Delay by One Sample

Description

The Delay by One Sample primitive buffers the incoming data for one sample duration before sending it to the output.

Connectors

Inputs	Type	Outputs	Type
Audio stream	Stream	Delayed audio stream	Stream

Delay Compensation

Description

This primitive allows you to send delay compensation to a host in an exported plugin. In line with the VST SDK, the delay time is expressed as a number of samples. You should only use this component once per plugin.

Connectors

Inputs	Type	Outputs	Type
Delay compensation in samples	Int	N/A	

Direct Sound In

Description

Microsoft's DirectSound is a digital audio protocol specified which provides the interface between applications and the sound card. The DirectSound In primitive provides a mono connection for the Left and Right input channels of the sound devices on your computer.

The Direct Sound In and ASIO In components provide the only way to receive audio signals from an external source (via inputs on your sound card). Note that only one Direct Sound In component is allowed in your schematic.

Connectors

Inputs	Type	Outputs	Type
N/A		Left audio channel	Mono
		Right audio channel	Mono

Other Features

The body of the component displays which input device is currently being used. All devices supporting the DirectSound protocol are listed so to select a different one just click on it.

To deselect a device (and therefore switch Direct Sound input off) simply click on it again.

Direct Sound In Devices

Description

You can use the Direct Sound In Devices component to find out how many Direct Sound input devices there are on the system you're running on. The first output tells you how many devices there are and the second output gives you a list of device names.

This component can be used used to provide information for GUI based selection controls that select the input device.

Connectors

Inputs	Type	Outputs	Type
N/A		Number of devices available	Int
		Array of device names	String Array

Direct Sound In Select

Description

The Direct Sound In Select primitive allows you to select an Direct Sound input device. It overrides whatever you have selected on the DirectSound In component.

This component can be used to build GUI components that control which Direct Sound input device is selected.

Connectors

Inputs	Type	Outputs	Type
Index of device you want to select	Int	Index of the currently selected device (or -1 if no devices are selected)	Int

Direct Sound Out

Description

Microsoft's DirectSound is a digital audio protocol specified which provides the interface between applications and the sound card. The DirectSound Out primitive provides a mono connection for the Left and Right channels of the sound devices on your computer.

The Direct Sound Out and ASIO Out components provide the only way to send audio signals to your sound card. You therefore must have at least one of these connected up to your schematic if you want to hear any sound. Note that only one Direct Sound Out component is allowed in your schematic.

Connectors

Inputs	Type	Outputs	Type
Left audio channel	Mono	N/A	
Right audio channel	Mono		

Other Features

The body of the component displays which output device is currently being used. All devices supporting the DirectSound protocol are listed so to select a different one just click on it.

To deselect a device (and therefore switch Direct Sound output off) simply click on it again.

Divide

Description

This component divides the first (top) value by the second one.

The component has template connectors which means it can be used with multiple data types including Floats, Ints and Float/Int arrays. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
Input 1	Template	Division of inputs	Template
Input 2	Template		

Double Round Nearest

Description

This component converts Double Stream values to the nearest integer value.

Double Stream is similar to standard Float Stream except that it works at double the precision. It is particularly useful for counters.

Connectors

Inputs	Type	Outputs	Type
Input signal	Double Stream	Rounded output	Double Stream

Double Stream Add

Description

This component adds two Double Stream values together.

Double Stream is similar to standard Float Stream except that it works at double the precision. It is particularly useful for counters.

Connectors

Inputs	Type	Outputs	Type
Input signal 1	Double Stream	Sum of inputs	Double Stream
Input signal 2	Double Stream		

Double Stream Multiply

Description

This component multiplies two Double Stream values together.

Double Stream is similar to standard Float Stream except that it works at double the precision. It is particularly useful for counters.

Connectors

Inputs	Type	Outputs	Type
Input signal 1	Double Stream	Product of inputs	Double Stream
Input signal 2	Double Stream		

Double Stream Subtract

Description

This component subtracts two Double Stream values from one another.

Double Stream is similar to standard Float Stream except that it works at double the precision. It is particularly useful for counters.

Connectors

Inputs	Type	Outputs	Type
Input signal 1	Double Stream	Input 1 minus Input 2	Double Stream
Input signal 2	Double Stream		

Double to Stream

Description

This component converts a Double Stream to a standard Stream.

Double Stream is similar to standard Float Stream except that it works at double the precision. It is particularly useful for counters.

Connectors

Inputs	Type	Outputs	Type
Double Stream	Double Stream	Double Stream converted to Float Stream	Stream

Draw Loop

Description

The Draw Loop component runs a loop between two float values causing a draw at the View output at each stage. You can use this to draw several similar items in one go. For example, you can draw a list of strings using the current loop value as an input to change position. The Step LFO module in the toolbox uses this to draw the bars for display.

Connectors

Inputs	Type	Outputs	Type
View to draw on	View	The same View as the input, but anything connected here is drawn for every iteration of the loop	View
Start value for the loop	Float	Current loop value	Float
Step increment value for the loop	Float		
End value for the loop	Float		

Draw to Bitmap

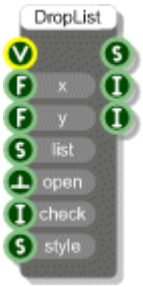
Description

This component allows you to draw onto a bitmap. By connecting GUI components to the View output you can draw onto a copy of a source bitmap and get the result at the Bitmap output.

Connectors

Inputs	Type	Outputs	Type
Source bitmap	Bitmap	The modified bitmap	Bitmap
The grid step to use when drawing onto the bitmap	Float	View that represents the bitmap canvass. Connect GUI components here	View
Trigger to initiate the drawing	Trigger	Trigger when drawing is complete	Trigger

Drop List Control



Description

The Drop List Control defines a drop list of selectable values. The drop-list is displayed when you send a trigger to the Open input. The drop-list closes when you select an item from the list.

You can decide whether to check an item in the list, usually to show the current selection. You can also choose the style of the list when there are too many items to show vertically on the screen. The default (if you leave the style blank) is “AutoCol” this will split the list into vertical columns. You can also specify the exact number of items per column manually by using a number for the style.

The other option for style is “Scroll” - this will maintain a single column but with scroll buttons at either end to scroll through the list.

For a good example of how the Drop List control works have a look inside the Selector control module in the toolbox.

Connectors

Inputs	Type	Outputs	Type
View on which to display the list when it opens	View	The last selected list item	String
The desired x-coordinate of the top-left corner of the list when it is opened	Float	The index of the selected list item	Int

The desired y-coordinate of the top-left corner of the list when it is opened	Float	The number of entries in the list	Int
Comma separated list of entries	String		
Trigger to open the drop-list	Trigger		
Index of item to show as checked	Int		
The drop-list style, either “AutoCol”,“Scroll” or a number representing the maximum number of items per column. If no value is supplied then “AutoCol” is assumed	String		

DSP Code

Description

The DSP Code component allows you to write algorithms using a small set of instructions and use it in your schematic. There are special commands for creating inputs and outputs so that you can connect the Assembler component to other components.

There is a whole chapter dedicated to the DSP Code component in the main user guide.

Connectors

Inputs	Type	Outputs	Type
N/A		Translated x86 assembler code – attach to a Text component to view	String

Edit Control

Description

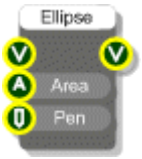
Creates an edit control on a view. You can define the size, position, font, text and background colours. You can also specify whether the edit control is single line (for input fields) or multiline (for entering text with line breaks).

Connectors

Inputs	Type	Outputs	Type
View to display the edit control on	View	The same View as the input, but anything connected here is drawn on top	View
Area defining the position and size of the edit control	Area	The string defined by the control	String
Font to use for the text	Font	Flag showing True when the control is in edit mode and False otherwise	Boolean
Text colour	Colour		
Background colour	Colour		
Sets the text in the control	String		

Only send a trigger after editing is complete	Boolean
Forces the Edit control to start editing without the need for a mouse click	Trigger
Allows you to make the Edit multi line like a text editor	Boolean

Ellipse



Description

Draws an ellipse on a View.

Connectors

Inputs	Type	Outputs	Type
View to display the ellipse on	View	The same View as the input, but anything connected here is drawn on top	View
Area defining the position and size of the ellipse	Area		
Pen defining the outline colour, thickness and style	Pen		

Envelope Control

Description

The envelope control component is used for creating your own custom envelopes. The key to this is the stage output which tells you what the voice is doing. The stages are:

0	-	Off	Voice not active or envelope stopped
1	-	On	Note on, voice triggered or retriggered (stage 1 only occurs for one sample)
3	-	Execute	Note held on, voice active, envelope executing
4	-	Release	Note off, voice still active, envelope releasing

If you have the Hold input set to true the voice will hold after note off until you send a TRUE value to the End input at which point the voice will be deleted.

So just running through it: transition to stage 1 tells you to (re)start the envelope and you continue executing until the stage moves to 4 (if Hold is true). At this point you execute the release part of your envelope. When complete send a TRUE value to the End input.

Another benefit of this component is that it allows you to hard sync automatically without having to connect to the retrigger from the MIDI to Poly module. This is exactly what happens in the Wave Player module to make the wave restart from the beginning on retrigger.

Connectors

Inputs	Type	Outputs	Type
End the envelope (when in release stage)	Stream Boolean	Current stage (Off=0,On=1,Execute=3, Release=4)	Stream
Whether to Hold for release stage after note off	Boolean		

Equals

Description

This component determines whether the two input values are equivalent and sends a True or False result to the output.

The component has template connectors which means it can be used with multiple data types including Floats, Ints, Strings and Float/Int/String arrays. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
First value to compare	Template	Whether the two input values are exactly equal	Boolean
Second value to compare	Template		

EXE Background Colour

Description

Use this in an exported exe to set the default background colour. This colour is also used for areas outside of the main GUI when running in full screen mode

Connectors

Inputs	Type	Outputs	Type
The colour you want to use	Colour		

EXE Full Screen

Description

Use this in an exported exe to toggle between full screen and windowed modes . This could be linked to a button on the GUI or in response to some outside event or timer.

Connectors

Inputs	Type	Outputs	Type
Trigger to toggle full screen	Trigger		

EXE Quit

Description

Use this in an exported exe to close the application. This could be a button on the GUI or in response to some outside event or timer.

Connectors

Inputs	Type	Outputs	Type
Trigger to close the exe	Trigger		

EXE Zoom

Description

Use this in an exported exe to set the zoom level. You may want to have a fixed level or change in response to some user input.

You can only pick from discrete zoom levels. The options are as follows:

0	25% Zoom	4	150% Zoom
1	50% Zoom	5	200% Zoom
2	75% Zoom	6	300% Zoom
3	100% Zoom	7	Fit to Screen (when in full screen mode)

Connectors

Inputs	Type	Outputs	Type
Zoom level option	Int	N/A	
Set the zoom level	Trigger		

FFT

Description

This component performs a fast fourier transform using arrays of Real and Imaginary numbers.

Connectors

Inputs	Type	Outputs	Type
Array of Real parts	Float Array	Array of transformed Real parts	Float Array
Array of Imaginary parts	Float Array	Array of transformed Imaginary parts	Float Array

File Dialog

Description

This component allows you to display the standard Windows file dialog box. You can choose whether the dialog is for saving or loading. You also have access to file filters and extensions.

The File Filters input is used to determine which file types to display in the dialog. Each filter is made from two strings. The first describes the filter. For example “Text Files (*.txt)”, you can use whatever description you like. The second string is the file extension and this must be of the format “*.extension”. The two strings are separated by a vertical bar | . You can have multiple extensions but these must be separated by a semicolon.

You can also have multiple filters and these must be separated by a vertical bar |. The completed filter specification must be terminated with a double vertical line ||.

Examples

```
Text Files (*.txt)|*.txt||
```

```
Image Files (*.png;*.bmp;*.jpg;*.tiff)| *.png;*.bmp;*.jpg;*.tiff||
```

```
PNG Files (*.png)|*.png|Jpeg Files (*.jpg)|*.jpg||
```

Connectors

Inputs	Type	Outputs	Type
Trigger to open dialog	Trigger	Path to the selected file (if any)	String
File filters	String	Trigger if OK was pressed	Trigger
Type of dialog, 0=Save and 1=Load	Int	Trigger if Cancel was pressed	Trigger
Text to show on the title bar of the dialog	String		
Path to folder to start in	String		
Default filename to use (including the file extension)	String		
Default file extension to append to filenames that are specified without an extension (include the dot)	String		

Filled Ellipse

Description

Draws a filled ellipse on a View.

Connectors

Inputs	Type	Outputs	Type
View to display the ellipse on	View	The same View as the input, but anything connected here is drawn on top	View
Area defining the position and size of the ellipse	Area		
Colour to fill the ellipse with	Colour		

Filled Rectangle

Description

Draws a filled rectangle on a View.

Connectors

Inputs	Type	Outputs	Type
View to display the rectangle on	View	The same View as the input, but anything connected here is drawn on top	View
Area defining the position and size of the rectangle	Area		
Colour to fill the rectangle with	Colour		

Filled Round Rectangle

Description

Draws a filled round rectangle on a View, that is a rectangle with rounded corners.

Connectors

Inputs	Type	Outputs	Type
View to display the rectangle on	View	The same View as the input, but anything connected here is drawn on top	View
Area defining the position and size of the rectangle	Area		
Colour to fill the rectangle with	Colour		
Corner size in grid squares. A value of 1 will give you a corner that has an effective radius of 1 grid square	Float		

Filter Coefficients

Description

Calculates the coefficients used for a low-pass biquad IIR filter using 2 poles, 2 zeros and 12dB per octave from the cutoff frequency and resonance. Use this in the Biquad Filter Coeff component.

Connectors

Inputs	Type	Outputs	Type
Normalised cutoff frequency (0-1) where 1 is half sampling rate	Stream	a0	Stream
Resonance	Stream	a1	Stream
		a2	Stream
		b1	Stream
		b2	Stream

Find Files

Description

The Find Files component will look for files matching a certain specification inside a folder on your hard disk. The Filter input specifies the folder followed by a filename. The filename can contain the * character to specify a wildcard in which case multiple matching files may be found.

Example Filters

```
C:\Program Files\Outsim\FlowStone\ *.dll
C:\ data*.txt
C:\Files\Downloads\picture.png
```

The component will return an array of filenames that match the filter. The number of files found is also given together with a trigger when the operation is complete.

Connectors

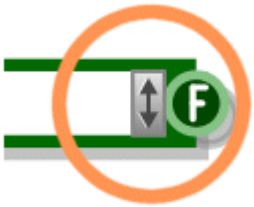
Inputs	Type	Outputs	
Filter specifying what file(s) to look for	String	Array of filenames that match the filter	String Array
Trigger to start finding matching files	Trigger	Number of matching files found	Int
		Trigger when the operation is complete	Trigger

Float

Description

The Float component allows you to enter and view floating point data. To enter a value just click on the main body of the component and type it in. Press Return, Tab or just click away to finish editing.

You can also change the value using the scroll strip to the right of the component. To use the scroll strip click on it and hold the mouse down. Keeping the mouse down, move up to increase the value and move down to decrease the value.



The increment is proportional to the amount you move your mouse to the left or right of the scroll strip. Moving to the left decreases the increment, moving to the right increases the increment. To maintain the current increment independent of the horizontal position of the mouse hold SHIFT as you move. To move in round number intervals hold CTRL (eg 1000, 100, 10, 1, 0.1, 0.01 etc. depending on the current increments size).

You can copy and paste data using the standard accelerator key combinations (CTRL+C,X and V).

The component can be resized horizontally for viewing larger numbers.

You can also change the type by right-clicking on the input or output. A pop-up menu will appear as shown below.

Simply click on the type you want to change to.

Connectors

Inputs	Type	Outputs	Type
Set the value	Float	The current stored value	Float

Float Abs 

Description

This component calculates the absolute value of the input or in other words the magnitude ignoring the sign.

Connectors

Inputs	Type	Outputs	Type
Float value	Float	Absolute value	Float

Float Array

Description

The Float Array component creates an array of floating point numbers by setting, inserting and deleting individual elements in the array. You define the index of the element you want to refer to and if needed the value you want to set or insert at that index. The array will resize automatically to accommodate elements set or inserted at indexes higher than the current size of the array.

Connectors

Inputs	Type	Outputs	
Value to use	Float	Array of floats	Float Array
The reference index used for set, insert and delete	Int	Trigger sent when the array has been changed	Trigger
Set the value at the index	Trigger	The number of array entries	Int
Clear the array	Trigger		
Array to make this equal to	Float Array		
Insert the value at the index	Trigger		
Delete the entry at the index	Trigger		

Float Array Abs

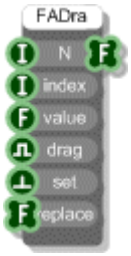
Description

The Float Array Abs component will replace each value in a float array with its absolute value .

Connectors

Inputs	Type	Outputs	Type
The source array	Float Array	Array of absolute values	Float Array

Float Array Draw



Description

The Float Array Draw component is used when you have a mouse drag operation that you need to map onto a float array. The advantage it has over using just a standard float array is that it interpolates between values when you are dragging. Without such interpolation you can get ‘missed’ points during fast drags as the mouse drag resolution decreases.

Connectors

Inputs	Type	Outputs	Type
Number of points in the float array	Int	The float array	Float Array
Index of the point to be updated	Int		
Value to change to	Float		
True if a drag operation is in progress, False otherwise	Boolean		
Trigger to set the value at the current index	Trigger		
Float array to replace the whole array with	Float Array		

Float Array Get At

Description

The Float Array Get At component extracts a particular entry from a Float Array.

Connectors

Inputs	Type	Outputs	
The source array of floating point numbers	Float Array	The number at the given index	Float
The index to get the value for	Int	Trigger sent when the value has been extracted	Trigger
Trigger to get the value	Trigger		

Float Array Resample

Description

This component allows you to up or down sample a float array to an alternative size.

Connectors

Inputs	Type	Outputs	Type
Source Float Array	Float Array	Resampled Float Array	Float Array
Required size of the resampled array	Int		
Interpolation option (0=nearest value, 1=linear, 2=cubic)	Int		
Whether to smooth the end points so that they loop round nicely from end to start (useful for looping samples)	Boolean		

Float Array Sample and Hold

Description

When a trigger is received this component will take a sample of the float array and hold it at the output until the next trigger is received.

Connectors

Inputs	Type	Outputs	Type
Source Float Array	Float Array	Last held sample of the input Float Array	Float Array
Trigger to take a sample	Trigger		

Float Array Section

Description

This component splits off a section of a Float Array. The section is defined by a start point and a section size.

Connectors

Inputs	Type	Outputs	Type
Source Float Array	Float Array	Section of Float Array	Float Array
Number of points in the section	Int		
Index of the first point in the array at which the section should start (starting at zero)	Int		

Float Array to Mem

Description

The Float Array to Mem component converts an array of floating point numbers to a memory buffer. This can then be read at sampling rate by the Wave Read component.

Connectors

Inputs	Type	Outputs	
Array of floating point numbers	Float Array	Memory buffer	Mem

Float Array to Poly

Description

Creates a Poly signal which consists of values extracted from a float array. The values are extracted using a PolyInt signal as an index into the array.

Connectors

Inputs	Type	Outputs	Type
Array of floats to use for generating the Poly signal	Float Array	Generated poly signal	Poly
PolyInt signal that defines which array value to use for each sample in the generated signal	PolyInt		

Float Inverse

Description

This component calculates the inverse sign of the input value, effectively it multiplies the input by -1.

Connectors

Inputs	Type	Outputs	Type
Input value	Float	Inverse of the input	Float

Float Invert

Description

This component inverts the input value, effectively dividing the input into 1.

Connectors

Inputs	Type	Outputs	Type
Input value	Float	Inverted input value	Float

Float Power

Description

The Float Power primitive calculates the result of the first input raised to the power of the second input.

Connectors

Inputs	Type	Outputs	Type
Base value	Float	Base value to the power of exponent value	Float
Exponent value	Float		

Float Queue

Description

The Float Queue component stores float values in a queue. Values are pushed in and popped out on a first in, first out basis (FIFO).

You can get the queue in Float Array form from the third output. The array contains items in the order they would be popped out – so the first item is at the front of the queue, the second item is next and so on.

Connectors

Inputs	Type	Outputs	
Next number to be pushed onto the queue	Float	The number at the front of the queue	Float
Trigger to push the next number onto the queue	Trigger	Number of entries in the queue	Int
Trigger to pop the next number off the queue	Trigger	The queue as a float array	Float Array
Trigger to clear all entries from the queue	Trigger		

Float Stack

Description

The Float Stack component stores float values in a stack. Values are pushed in and popped out on a last in, first out basis (LIFO).

You can get the stack in Float Array form from the third output. The array contains items in the order they would be popped out – so the first item is at the top of the stack, the second item is next and so on.

Connectors

Inputs	Type	Outputs	
Next number to be pushed onto the stack	Float	The number at the top of the stack	Float
Trigger to push the next number onto the stack	Trigger	Number of entries in the stack	Int
Trigger to pop the next number off the stack	Trigger	Stack represented as an array of floats	Float Array
Trigger to clear all entries from the stack	Trigger		

Float to Area

Description

The Float to Area component constructs an Area from X, Y, Width and Height components.

Connectors

Inputs	Type	Outputs	Type
X component	Float	Corresponding Area	Area
Y component	Float		
Width component	Float		
Height component	Float		

FlowBoard

Description

This primitive allows you to send and receive data to and from the DSP Robotics FlowBoard DAQ.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

The board has 16 digital outputs, 16 digital inputs and 8 analog inputs. These are mirrored on the component. The 16 boolean inputs will send data out through the 16 digital outputs. The 8 integer outputs receive data from the 8 analog inputs. The 16 boolean outputs receive data from the 16 digital inputs.

The 8 Int outputs are in the range 0-1023.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
16 x inputs to change the state of the 16 digital outputs on the board	Boolean	16 x outputs receiving the state of the 16 digital inputs on the board	Boolean
		8 x outputs receiving analog data from the 8 analog inputs on the board (range 0-1023)	Int
		Connector for attaching components that allow you to control some of the satellite boards, the GSM board being one such example.	Int

FlowBoard GSM

Description

This primitive allows you to control a FlowBoard GSM Board. The GSM Board hardware must be connected to the Modem connector on your FlowBoard. In Flowstone you then need a FlowBoard component and a FlowBoard GSM component in your schematic. You then need to link these components together by connecting from the FlowBoard output on the FlowBoard component (the very last output) to the FlowBoard input on the FlowBoard GSM component (the very first one).

Once the components are connected you then need to start the FlowBoard component by sending a trigger to the first input. The 'On' output on the FlowBoard component will return True if a connection has been established (False otherwise). The Board output on the FlowBoard GSM component will return 1 if the FlowBoard is detected and 2 if both the FlowBoard and the GSM Board are detected.

Signal strength is shown at the Signal output. This is in the range 0-30. A value of -1 is output if the board is searching for the network or if the board is not connected.

To send a message you need to provide message text and a phone number at the Message and Number inputs. Trigger the Send input to send the message.

The Status output indicates the message send state. A value of 0 indicates that the last message has been sent successfully. A value of 1 indicates sending in progress. A value of 2 is output if the last message failed to send.

The Count output shows how many messages are waiting to be sent (including any message currently being sent).

Received messages will automatically appear at the Messages and Numbers outputs. These two string arrays are aligned such that the first number corresponds to the first message and so on.

You can use the Pop input to pop the oldest message off the top of the arrays. The Clear input will clear all received messages.

Connectors

Inputs	Type	Outputs	Type
Connection to FlowBoard component	Int	The state of the board. 1=flowboard detected, 2=flowboard and GSM detected. 0=no board detected	Int
The text for the message you want to send (160 characters max)	String	The network signal strength. This is in the range 0-30. A value of -1 indicates searching for network or no connection	Int
The telephone number for the recipient of the message	String	The current send status. 0=last message succeeded, 1=sending in progress. 2=last message failed to send.	Int
Trigger to send the message	Trigger	The number of messages waiting to be sent	Int
Pop the oldest received message off the received messages list	Trigger	Array of messages received	StringArray
Clear all received messages from the list	Trigger	Array of numbers corresponding to messages received	StringArray

Font

Description

Creates a Font from a typeface, font size and style. The typeface is the name of the font face e.g. 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
```

Examples

```
bolditalic, underlineboldstrike, italicunderline
```

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

Connectors

Inputs	Type	Outputs	Type
Font typeface name (default is Arial)	String	The font	Font
Font size in grid squares	Float		
Style (see Description above)	String		

Format String

Description

The Format String primitive applies standard C string formatting to a numeric string input. This is particularly useful for making numeric data conform to user interface requirements such as displaying a fixed number of decimal places.

The value input has String type but should be connected to either an Int or Float for the formatting to work correctly.

The format specifications are defined here:

http://msdn.microsoft.com/en-us/library/kwwtf9ch(VS.71).aspx

Connectors

Inputs	Type	Outputs	Type
The Float or Int that you want to format	Float or Int	The formatted string	String
Format specification string	String		

Frame Sync

Description

This Frame Sync component sends an integer value precisely at the time each buffer of audio is requested. The integer value is the number of samples that has been requested for the buffer.

For full details about how to use this see the section on Frames in the Ruby Component chapter of the main user guide.

Connectors

Inputs	Type	Outputs	Type
		Size of the audio buffer	Ruby Value

Frame to Mono

Description

This Frame to Mono component converts a Frame of samples back to Mono. It should be used in conjunction with the Mono to Frame component in order to process Mono data via the Ruby component.

It can also be used with the Frame Sync component. The Frame Sync will tell your Ruby component when to send a frame and how many samples to send so that you stay exactly in sync with the Mono stream.

For more information about this see the section on Frames in the Ruby Component chapter of the main user guide.

Connectors

Inputs	Type	Outputs	Type
Ruby Frame object with a buffer of samples	Ruby Value	Mono signal	Mono

Full Screen

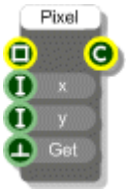
Description

This primitive is only used for exported exes and allows you to control full screen mode from the front panel by connecting a GUI component like a toggle button to the trigger input.

Connectors

Inputs	Type	Outputs	Type
Toggles between full screen mode in exported exe's	Trigger	N/A	

Get Pixel



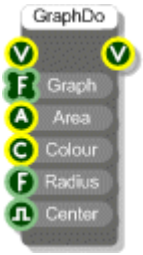
Description

Gets the colour of a pixel at a particular point in a bitmap.

Connectors

Inputs	Type	Outputs	Type
Source bitmap	Bitmap	Colour of the pixel at (x,y)	Colour
The x-coordinate of the required pixel	Int		
The y-coordinate of the required pixel	Int		
Trigger to get the colour	Trigger		

Graph Dots



Description

Draws a set of points using a dot for each point. The Float Array input supplies the y-coordinates. These are assumed to be in the range 0 to 1 (–1 to 1 if the graph is centred on zero). This means that you may need to use the Norm component if you want to make sure that your values are all visible. The x-coordinates are automatically generated and are equally spaced across the horizontal axis.

Connectors

Inputs	Type	Outputs	Type
View to draw onto	View	The same View as the input, but anything connected here is drawn on top	View
The array of y-coordinates	Float Array		
The area of the view that the graph will be drawn into	Area		
The dot colour	Colour		
Radius of the dots (grid sq.)	Float		
True if the graph is centred on zero, False otherwise	Boolean		

Graph FFT

Description

The Graph FFT primitive performs a Fast Fourier Transform (FFT) on an array of float data. The outputs are the magnitude and phase outputs from the FFT calculation.

Connectors

Inputs	Type	Outputs	Type
Source data (usually from a wave file or a snapshot of a signal)	Float Array	Magnitude component of the calculation	Float Array
		Phase component of the calculation	Float Array

Graph Lines

Description

Draws a line through a set of points. The Float Array input supplies the y-coordinates. These are assumed to be in the range 0 to 1 (–1 to 1 if the graph is centred on zero). This means that you may need to use the Norm component if you want to make sure that your values are all visible. The x-coordinates are automatically generated and are equally spaced across the horizontal axis unless you set the Log input to True in which case the x-coordinates will be logged.

By default the line drawn through the points is made from straight line segments. You can also choose to draw a best fit curve through the points by setting the Curve input to True.

Connectors

Inputs	Type	Outputs	Type
View to draw onto	View	The same View as the input, but anything connected here is drawn on top	View
The array of y-coordinates	Float Array		
The area of the view that the graph will be drawn into	Area		
The pen defining the colour, thickness and style of the lines	Pen		
Radius of the dots (grid sq.)	Float		
True if the graph is centred on zero, False otherwise	Boolean		
True if you want to log the x-axis, False otherwise	Boolean		
True if you want a curved line, False otherwise	Boolean		

Graph to Point Array

Description

The Graph to Point Array component creates an array of points from two float arrays. Currently the only component that uses a Point Array is the Point Array Lines component and so the only use of this component is in creating custom graphs.

Connectors

Inputs	Type	Outputs	Type
Array of x values	Float Array	Array of points constructed from the x and y values	Point Array
Array of y values	Float Array		

Greater Than

Description

This component compares the two inputs and returns a Boolean based on whether the first input is greater than the second input.

The component has template connectors which means it can be used with multiple data types including Floats, Ints, Strings and Float/Int/String arrays. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
Input 1	Template	Whether input 1 is greater than input 2	Boolean
Input 2	Template		

Greater Than or Equal To

Description

This component compares the two inputs and returns a Boolean based on whether the first input is greater than or equal to the second input.

The component has template connectors which means it can be used with multiple data types including Floats, Ints, Strings and Float/Int/String arrays. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
Input 1	Template	Whether input 1 is greater than or equal to input 2	Boolean
Input 2	Template		

Grid to Pixel

Description

The Grid to Pixel components convert values in Grid Squares to values in Pixels. There are two versions, one for Floats and one for Areas.

Connectors

Inputs	Type	Outputs	Type
Float or Area in grid square units	Float/Area	Float or Area in pixel units	Float/Area

Haar Face Detect

Description

The Haar Face Detect component uses the Haar algorithm to find a face in an image.

The component requires a Haar cascade which is an XML file that defines the classification criteria used in the detection process. You can download such files from the DSP Robotics support area.

Optional input parameters include a threshold for feature size, whether to apply Canny edge detection pruning, whether to look for a single object and an option to perform only a rough calculation. All of these serve to speed up the detection process.

Connectors

Inputs	Type	Outputs	Type
The source image you want to process	Bitmap	The bounding box of the largest detected face	Area
Path to the XML classification file	String	The number of detections	Int
The minimum feature size in pixels	Int		
Apply Canny pruning	Boolean		
Look for a single object	Boolean		
Rough calculation only	Boolean		

Hard Disk Serial

Description

Gets the serial number of the first hard disk attached to the host PC.

Connectors

Inputs	Type	Outputs	Type
Trigger to get the serial number	Trigger	Serial number of hard disk	String

Hex to Binary

Description

Converts a string of hex to a string of binary. Each byte of hex is converted to 8 bit binary.

For example, the hex string “2FBB” is converted to “0010111110111011”.

Connectors

Inputs	Type	Outputs	Type
String of hex	String	String of binary	String

Hex to Int

Description

Calculates the decimal equivalent of a hexadecimal number.

Connectors

Inputs	Type	Outputs	Type
Hex	String	Integer equivalent	Int

Hex to String

Description

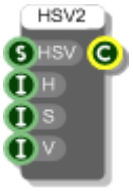
Converts a string of hex to a string of characters. Each byte of hex is converted to an Ascii character.

For example, the hex string “68656C6C6F” is converted to the character string “hello”.

Connectors

Inputs	Type	Outputs	Type
String of hex	String	String of characters	String

HSV to Colour



Description

The HSV to Colour component creates a colour from Hue, Saturation and Value components. You can choose whether to specify the HSV as a comma separated string or as separate integer components.

Connectors

Inputs	Type	Outputs	Type
HSV as a comma separated string.	String	Corresponding Colour	Colour
Hue component (0-360)	Int		
Saturation component (0-255)	Int		
Value component (0-255)	Int		

HTTP Post

Description

The HTTP Post component allows you to send data to and receive data from a web server using the HTTP post request method.

You specify the server URL at the URL input – for example, http://www.dsprobotics.com

You then specify the page on the server e.g. scripts\myphp.php

The Names and Values array inputs allow you to specify the key-value pair data that you want to send to the page. The first entry in the Names array is paired with the first value in the Values array. So for example, “forename=Fred&age=30” would have forename and age in the Names array and Fred and 30 in the Values array.

Trigger the Submit input to send the post request. The Ok output will show true if it succeeded. Any data returned can be accessed from the Data output

Connectors

Inputs	Type	Outputs	Type
Address of the web server	String	Result of the last	Colour
Name of the page on the server	String	Any data returned from the server	String
Array of data item names	String Array		
Array of data item values	String Array		
Trigger to execute the post			

iFFT

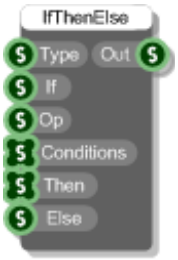
Description

This component performs an inverse fast fourier transform on arrays of Real and Imaginary numbers.

Connectors

Inputs	Type	Outputs	Type
Array of Real parts	Float Array	Array of transformed Real parts	Float Array
Array of Imaginary parts	Float Array	Array of transformed Imaginary parts	Float Array

If Then Else



Description

This component implements an if/then/else conditional statement. In a language like C this would be written as follows:

```
if( control > condition1 )
{
   // Then output for condition1
}
else if( control > condition2 )
{
   // Then output for condition2
}
…
else
{
   // Else output
}
```

For this component the If input defines the control variable this could be an int, float or string. You specify which of these using the Type input (either “int”,”float” or “string”).

The 'Op' input defines the comparison operation, one of “=”,“<=”,”<”,”>' or ”>=”. If left blank then “=” will be assumed.

The conditions and corresponding 'then outputs' are provided as two string arrays. However, these can contain ints, floats or strings. Their treatment in the comparison process is determined by the Type input.

Connectors

Inputs	Type	Outputs	Type
The data type to be used, “int”,”float” or “string”	String	The result of the comparison	String
The control variable	String		
The comparison operation, “=”,“<=”,”<”,”>' or ”>=”	String		
The condition values to use in the comparison	String Array		
The output for each condition should it be met	String Array		
The output should no conditions be met	String		

Image Download

Description

This component will download an image from a web server using HTTP GET.

This is useful for accessing web based cameras and such like.

You do is provide a valid URL for the image and then trigger the Download input and the image will be downloaded and output as a bitmap.

Connectors

Inputs	Type	Outputs	Type
URL of the image file you want to download	String	The downloaded image	Bitmap
Trigger to download the image	Trigger		

Impulse

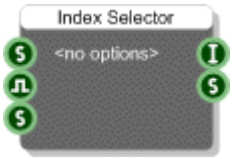
Description

Generates an impulse signal. This has a value of one as the first sample and zero for all others. Use this to test the frequency response of a filter.

Connectors

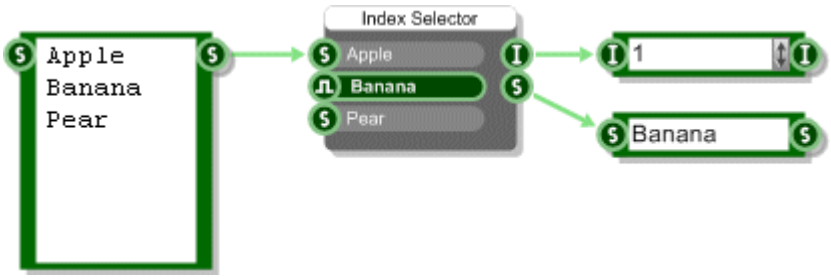
Inputs	Type	Outputs	Type
N/A		Impulse signal	Stream

Index Selector



Description

The Index Selector component is used to select between a list of string values. You supply these as a comma separated string or using a Text component. The body of the component changes to show each string as an option. You can click on these options to select one. The selected string and index will be sent to the outputs.



The main use of the Index Selector is to provide a mechanism for creating a drop list of options on a property panel (see the Properties section in the main user guide for more information).

Connectors

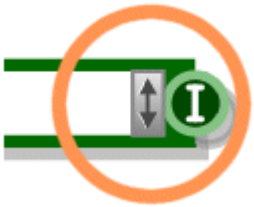
Inputs	Type	Outputs	Type
Set of option strings	String	Index of current selection (zero based)	Int
Whether to order the options in ascending alphabetical order	Boolean	The currently selected option string	String
Set the current selection	String		

Int

Description

The Int component allows you to enter and view integer data. To enter an integer just click on the main body of the component and type in a number. Press Return, Tab or just click away to finish editing.

You can also change the value using the scroll strip to the right of the component. To use the scroll strip click on it and hold the mouse down. Keeping the mouse down, move up to increase the value and move down to decrease the value.



The increment is proportional to the amount you move your mouse to the right of the scroll strip. To maintain the current increment independent of the horizontal position of the mouse hold SHIFT as you move. To move in round number intervals hold CTRL (eg 1, 10, 100, 1000 etc. depending on the current increments size).

You can copy and paste data using the standard accelerator key combinations (CTRL+C,X and V).

The component can be resized horizontally for viewing larger numbers.

You can also change the type by right-clicking on the input or output. A pop-up menu will appear as shown below.

Simply click on the type you want to change to.

Connectors

Inputs	Type	Outputs	Type
Set the value	Int	The current stored value	Int

Int Abs

Description

This component calculates the absolute value of the input or in other words the magnitude ignoring the sign.

Connectors

Inputs	Type	Outputs	Type
Float value	Int	Absolute value	Int

Int And

Description

Calculates the bitwise AND of two int values.

Connectors

Inputs	Type	Outputs	Type
First value	Int	AND value	Int
Second value	Int		

Int Array

Description

The Int Array component creates an array of integers by setting, inserting and deleting individual elements in the array. You define the index of the element you want to refer to and if needed the value you want to set or insert at that index. The array will resize automatically to accommodate elements set or inserted at indexes higher than the current size of the array.

Connectors

Inputs	Type	Outputs	
Value to use	Int	Array of ints	Int Array
The reference index used for set, insert and delete	Int	Trigger sent when the array has been changed	Trigger
Set the value at the index	Trigger	The number of array entries	Int
Clear the array	Trigger		
Array to make this equal to	Int Array		
Insert the value at the index	Trigger		
Delete the entry at the index	Trigger		

Int Array Get At

Description

The Int Array Get At component extracts a particular entry from a Int Array.

Connectors

Inputs	Type	Outputs	Type
The source array of integers	Int Array	The number at the given index	Int
The index to get the value for	Int	Trigger sent when the value has been extracted	Trigger
Trigger to get the value	Trigger		

Int Array Sample and Hold

Description

When a trigger is received this component will take a sample of the int array and hold it at the output until the next trigger is received.

Connectors

Inputs	Type	Outputs	Type
Source Int Array	Int Array	Last held sample of the input Int Array	Int Array
Trigger to take a sample	Trigger		

Int Array to Mem

Description

The Int Array to Mem component converts an array of integers to a memory buffer. This can then be read at sampling rate by the Wave Read component.

Connectors

Inputs	Type	Outputs	Type
Array of integers	Int Array	Memory buffer	Mem

Int Inverse

Description

This component calculates the inverse sign of the input value, effectively it multiplies the input by -1.

Connectors

Inputs	Type	Outputs	Type
Input value	Int	Inverse of the input	Int

Int Loop

Description

The Int Loop component cycles a fixed number of times sending a trigger on each iteration. You specify how many times to iterate (N). By default the loop counter starts at zero and increments in unit steps but you can start at any integer value you like.

To prevent hanging the loop is automatically limited to a maximum of 1000 iterations. You can bypass this safety mechanism by setting the No Limit input to True.

Connectors

Inputs	Type	Outputs	Type
Number of iterations	Int	The current iteration counter value	Int
The start value for the iteration counter	Int	Trigger sent on each iteration	Trigger
Trigger to start the loop	Trigger	Trigger sent when all the iterations are complete	Trigger
False if the iterations are to be limited to 1000	Boolean		

Int Modulus

Description

The Int Modulus primitive calculates the remainder when the first input is divided by the second input.

Connectors

Inputs	Type	Outputs	Type
Integer value	Int	Modulus	Int
Integer divisor	Int		

Int Not

Description

This component returns the bitwise NOT equivalent of the input value.

Connectors

Inputs	Type	Outputs	Type
Int input	Int	Bitwise NOT value	Int

Int Or

Description

Calculates the bitwise OR of two int values.

Connectors

Inputs	Type	Outputs	Type
First value	Int	OR value	Int
Second value	Int		

Int Queue

Description

The Int Queue component stores int values in a queue. Values are pushed in and popped out on a first in, first out basis (FIFO).

You can get the queue in Int Array form from the third output. The array contains items in the order they would be popped out – so the first item is at the front of the queue, the second item is next and so on.

Connectors

Inputs	Type	Outputs	
Next number to be pushed onto the queue	Float	The number at the front of the queue	Int
Trigger to push the next number onto the queue	Trigger	Number of entries in the queue	Int
Trigger to pop the next number off the queue	Trigger	The queue as an int array	Int Array
Trigger to clear all entries from the queue	Trigger		

Int Shift Left

Description

Shifts the bits in the first input to the left by the number in the second input. This is equivalent to multiplying the first input by 2 raised to the power of the second input.

Connectors

Inputs	Type	Outputs	Type
Int value to shift left	Int	Shifted value	Int
Number of bits to shift by	Int		

Int Shift Right

Description

Shifts the bits in the first input to the right by the number in the second input. This is equivalent to dividing the first input by 2 raised to the power of the second input.

Connectors

Inputs	Type	Outputs	Type
Int value to shift right	Int	Shifted value	Int
Number of bits to shift by	Int		

Int Stack

Description

The Int Stack component stores int values in a stack. Values are pushed in and popped out on a last in, first out basis (LIFO).

You can get the stack in Int Array form from the third output. The array contains items in the order they would be popped out – so the first item is at the top of the stack, the second item is next and so on.

Connectors

Inputs	Type	Outputs	
Next number to be pushed onto the stack	Float	The number at the top of the stack	Int
Trigger to push the next number onto the stack	Trigger	Number of entries in the stack	Int
Trigger to pop the next number off the stack	Trigger	Stack represented as an array of ints	Int Array
Trigger to clear all entries from the stack	Trigger		

Int to Colour

Description

The Int to Colour component creates a colour from an Alpha Transparency, Red, Green and Blue components.

Connectors

Inputs	Type	Outputs	Type
Alpha transparency (0-255) where 255 is opaque and 0 is transparent	Int	Corresponding Colour	Colour
Red component (0-255)	Int		
Green component (0-255)	Int		
Blue component (0-255)	Int		

Int to Hex

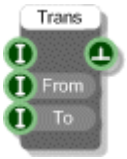
Description

Calculates the hexadecimal equivalent of a decimal number.

Connectors

Inputs	Type	Outputs	Type
Decimal integer	Int	Hex equivalent	String

Int Transition



Description

The Int Transition component monitors an input value and sends a trigger whenever the value moves from one ‘From’ value to another ‘To’ value. The monitored input value must be at the ‘From’ value then change to the ‘To’ when it next changes in order for the transition to be counted.

Connectors

Inputs	Type	Outputs	
Value to be monitored for the transition	Int	Trigger sent when the transition takes place	Trigger
The start value for the transition	Int		
The target value for the transition	Int		

Int XOr

Description

Calculates the bitwise XOR of two int values.

Connectors

Inputs	Type	Outputs	Type
First value	Int	XOR value	Int
Second value	Int		

Is Key Pressed

Description

The Is key Pressed component is used to determine whether a particular key is currently pressed. The key can be specified explicitly by it's letter or you can use the Virtual Key Code.

For a complete list of virtual key codes see the following link (all codes are in hex and must be converted to decimal before use):

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/VirtualKeyCodes.asp

You can also use "CTRL", "ALT" or "SHIFT" as shortcuts for those keys.

Note that this component does not send a trigger when the state of the key changes, it can only be used for investigating the current state of a key – this happens whenever you trigger the Check input.

Connectors

Inputs	Type	Outputs	Type
Letter or shortcut or virtual key code	String	Flag which will read True if the key is currently pressed and False otherwise	Boolean
Trigger to check the state of the key			

Is Playing

Description

When your VST is used within a host these components will tell you whether the host is playing or not. There are two versions. The first version will output 1.0 when the host is playing and 0.0 otherwise and should be used in Poly or Mono sections of your schematic.

The second version will output True when the host is playing and False otherwise. This is useful if you want to respond to changes in the playing state by performing a one-off calculation or displaying visual feedback. There are two output connectors, one Boolean for use in triggered sections of schematic and another Ruby Value connector for use in Ruby components. The Ruby value is sent with precise timing so if timing is an issue use this one.

Connectors

Inputs	Type	Outputs	Type
N/A		Whether the host is playing or not	Stream or Boolean + Ruby Value

LabJackU3-HV

Description

This primitive allows you to control and receive data from a LabJack U3-HV data acquisition device.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the address of the board you want to connect to (the address is described in the U3 user guide).

The board has 16 flexible I/O pins and 4 fixed analog inputs. These are mirrored on the component. Aside from the fixed analog inputs the I/O can be configured in various different ways to provide

analog or digital inputs. If a pin is configured as an input then it will appear as an output connector on the component so that you can retrieve its value and if it is an output then it will appear as an input connector on the component so that you can set its value.

Flexible I/O

To set the I/O you need to supply a comma separated string or a newline separated list of strings in a Text component to the I/O input. Each entry defines the format for a particular I/O pin. The entry starts with two characters to determine the type and direction: "DI", "DO" or "AI" for digital in, out and analog in respectively.

In the case of analog it defaults to single ended but you can set the negative channel with a minus symbol followed by "SE","SP","VR" (single ended, special (0-3.6v or -10/+20v), internal voltage ref) or the number of the pin you want to have as the negative channel.

To set the I/O you need to trigger the Set input below the I/O String input.

For example:

“DO4, DO5, AI6, AI7-SP, AI8-6” would make pins 4 and 5 digital outputs. Pins 6,7 and 8 would all be analog inputs with 6 being single ended, 7 using the special 0-3.6v range and 8 using input 6 as a negative channel.

Timers and Counters

The U3 has two timers and two counters that you can use. The timers and counters take over the flexible I/O. You specify a start pin and then depending on the number of timers and counters you use the pins from this start pin onwards will be used as timers or counters.

You can set the number of counters via the Counters input.

The timers are set using a configuration string. This is similar to the I/O specification in that it's a comma separated list of parameters (or a newline separated list if you use a Text component).

The timer configuration string is in the form

{number of timers} {clock{/divisor}} {mode timer0} {value timer0} {{mode timer1} {value timer1}}

Clock values can be 4, 12 or 48 MHz and can include a divisor as well. There is also a 1 MHz clock which must include a divisor. This gives 7 possible clock value strings:

1MHz/{divisor} , 4MHz/{divisor} , 12MHz/{divisor} , 48MHz/{divisor} , 4MHz , 12MHz or 48MHz.

Where {divisor} is an integer in the range 0-255.

Modes are strings and map onto the U3 modes exactly. They can be one of the following:

PWM16, PWM8, RISINGEDGES32, FALLINGEDGES32, DUTYCYCLE, FIRMCOUNTER, FIRMCOUNTERDEBOUNCE, FREQOUT, QUAD, TIMERSTOP, SYSTIMERLOW, SYSTIMERHIGH, SYSTIMERHIGH, RISINGEDGES16 or FALLINGEDGES16.

Some examples:

“1,48MHz,FREQOUT,128" – a single timer, clock 48MHz using the Frequency Output method

“2,4MHz/200,PWM16,32768,PWM8,16384" – two timers, clock 48MHz with divisor 200

To set the timer and counter configuration trigger the Set unput below the Counters input on the component.

Because the timers and counters take over some of the flexible I/O pins you should see the outputs on the component corresponding to those pins change to Integer connectors labeled TIM0, TIM1, CTR0 or CTR1.

Note that when using a clock with a divisor the LabJack U3 only allows you to use 1 counter.

The U3 has two DACs you can set the value of these using the DAC0 and DAC1 inputs on the component.

You can reset a timer or counter by sending the appropriate index value to the Reset input. Send 1 or 2 to reset the 1st or 2nd timer and 3 or 4 to reset the 1st or 2nd counter.

For more information on the workings of the U3 see the U3 user guide.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the U3 is connected and on	Boolean
Address of the board you want to connect to (optional)	String	The ambient temperature of the board	Float
Configuration string to set the I/O	String	4 x outputs used to deliver the values of the FIO0-FIO3 analog inputs on the U3	Float
Trigger to set the I/O	Trigger	12 x outputs used for the FIO and EIO pins that are setup as inputs on the U3	Boolean / Float
The start pin for any timers or counters	Int	4 x outputs used for the CIO pins that are setup as inputs on the U3	Boolean
Configuration string for setting up any timers	String		
Number of counters required	Int		
Trigger to set up the timers and counters	Trigger		
Index of timer or counter you want to reset (1=Timer1, 2=Timer2, 3=Counter1, 4=Counter2)			
Set the value of DAC0	Float		
Set the value of DAC1	Float		
12 x inputs used for the FIO and EIO pins that are setup as outputs on the U3	Boolean		
4 x inputs used for the CIO pins that are setup as outputs on the U3	Boolean		

LabJackU3-LV

Description

This primitive allows you to control and receive data from a LabJack U3-LV data acquisition device.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the address of the board you want to connect to (the address is described in the U3 user guide).

The board has 20 flexible I/O pins. These are mirrored on the component. The I/O can be configured in various different ways to provide analog or digital inputs. If a pin is configured as an input then it will appear as an output connector on the component so that you can retrieve its value and if it is an output then it will appear as an input connector on the component so that you can set its value.

Flexible I/O

To set the I/O you need to supply a comma separated string or a newline separated list of strings in a Text component to the I/O input. Each entry defines the format for a particular I/O pin. The entry starts with two characters to determine the type and direction: "DI", "DO" or "AI" for digital in, out and analog in respectively.

In the case of analog it defaults to single ended but you can set the negative channel with a minus symbol followed by "SE","SP","VR" (single ended, special (0-3.6v or -10/+20v), internal voltage ref) or the number of the pin you want to have as the negative channel.

To set the I/O you need to trigger the Set input below the I/O String input.

For example:

“DO4, DO5, AI6, AI7-SP, AI8-6” would make pins 4 and 5 digital outputs. Pins 6,7 and 8 would all be analog inputs with 6 being single ended, 7 using the special 0-3.6v range and 8 using input 6 as a negative channel.

Timers and Counters

The U3 has two timers and two counters that you can use. The timers and counters take over the flexible I/O. You specify a start pin and then depending on the number of timers and counters you use the pins from this start pin onwards will be used as timers or counters.

You can set the number of counters via the Counters input.

The timers are set using a configuration string. This is similar to the I/O specification in that it's a comma separated list of parameters (or a newline separated list if you use a Text component).

The timer configuration string is in the form

{number of timers} {clock{/divisor}} {mode timer0} {value timer0} {{mode timer1} {value timer1}}

Clock values can be 4, 12 or 48 MHz and can include a divisor as well. There is also a 1 MHz clock which must include a divisor. This gives 7 possible clock value strings:

1MHz/{divisor} , 4MHz/{divisor} , 12MHz/{divisor} , 48MHz/{divisor} , 4MHz , 12MHz or 48MHz.

Where {divisor} is an integer in the range 0-255.

Modes are strings and map onto the U3 modes exactly. They can be one of the following:

PWM16, PWM8, RISINGEDGES32, FALLINGEDGES32, DUTYCYCLE, FIRMCOUNTER, FIRMCOUNTERDEBOUNCE, FREQOUT, QUAD, TIMERSTOP, SYSTIMERLOW, SYSTIMERHIGH, SYSTIMERHIGH, RISINGEDGES16 or FALLINGEDGES16.

Some examples:

“1,48MHz,FREQOUT,128" – a single timer, clock 48MHz using the Frequency Output method

“2,4MHz/200,PWM16,32768,PWM8,16384" – two timers, clock 48MHz with divisor 200

To set the timer and counter configuration trigger the Set unput below the Counters input on the component.

Because the timers and counters take over some of the flexible I/O pins you should see the outputs on the component corresponding to those pins change to Integer connectors labeled TIM0, TIM1, CTR0 or CTR1.

Note that when using a clock with a divisor the LabJack U3 only allows you to use 1 counter.

The U3 has two DACs you can set the value of these using the DAC0 and DAC1 inputs on the component.

You can reset a timer or counter by sending the appropriate index value to the Reset input. Send 1 or 2 to reset the 1st or 2nd timer and 3 or 4 to reset the 1st or 2nd counter.

For more information on the workings of the U3 see the U3 user guide.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the U3 is connected and on	Boolean
Address of the board you want to connect to (optional)	String	The ambient temperature of the board	Float
Configuration string to set the I/O	String	16 x outputs used for the FIO and EIO pins that are setup as inputs on the U3	Boolean / Float
Trigger to set the I/O	Trigger	4 x outputs used for the CIO pins that are setup as inputs on the U3	Boolean
The start pin for any timers or counters	Int		
Configuration string for setting up any timers	String		
Number of counters required	Int		
Trigger to set up the timers and counters	Trigger		
Index of timer or counter you want to reset (1=Timer1, 2=Timer2, 3=Counter1, 4=Counter2)			
Set the value of DAC0	Float		
Set the value of DAC1	Float		
16 x inputs used for the FIO and EIO pins that are setup as outputs on the U3	Boolean		
4 x inputs used for the CIO pins that are setup as outputs on the U3	Boolean		

Last Switch

Description

The Float Switch component is used to select between two inputs depending on which one changed last. The most recently changed input will be used to supply the float value to the output.

The component has template connectors which means it can be used with multiple data types including Floats, Ints, Strings, Booleans, Float/Int/String arrays and Colours. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
First float value	Template	Most recently changed value	Template
Second float value	Template		

Less Than

Description

This component compares the two inputs and returns a Boolean based on whether the first input is less than the second input.

The component has template connectors which means it can be used with multiple data types including Floats, Ints, Strings and Float/Int/String arrays. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
Input 1	Template	Whether input 1 is less than input 2	Boolean
Input 2	Template		

Less Than or Equal to

Description

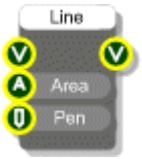
This component compares the two inputs and returns a Boolean based on whether the first input is less than or equal to the second input.

The component has template connectors which means it can be used with multiple data types including Floats, Ints, Strings and Float/Int/String arrays. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
Input 1	Template	Whether input 1 is less than or equal to input 2	Boolean
Input 2	Template		

Line



Description

Draws a straight line on a View. The line is defined by an Area. The x and y coordinates of the area define the start of the line and the width and height define a vector offset to the end point. This means that for some lines you will need to use negative widths and heights.

Connectors

Inputs	Type	Outputs	Type
View to display the line on	View	The same View as the input, but anything connected here is drawn on top	View
Area defining the start and direction of the line	Area		
Pen defining the outline colour, thickness and style	Pen		

Linear Gradient

Description

The Linear Gradient component draws either an ellipse or a rectangle with a gradient fill effect. This produces a seamless linear transition between two colours.

Connectors

Inputs	Type	Outputs	Type
View to draw on	View	The same View as the input, but anything connected here is drawn on top	View
Area defining the bounding area of the ellipse or rectangle	Area		
Either “Rectangle” or “Ellipse”	String		
First colour in the gradient	Colour		
Second colour in the gradient	Colour		
The angle of the gradient in degrees running clockwise from the horizontal	String		

Log10

Description

Calculates the logarithm (base 10) of a float.

Connectors

Inputs	Type	Outputs	Type
Source number	Float	Log10 of source	Float

MAC Address

Description

Gets the MAC addresses of all network devices attached to the host PC. The addresses are returned as a comma separated list.

Note that on some systems this component will not return any MAC addresses.

Connectors

Inputs	Type	Outputs	Type
Trigger to get the MAC addresses	Trigger	Comma separated list of MAC addresses	String

Magnitude/Phase to Real/Img

Description

This component converts arrays of Magnitude and Phase to Real and Imaginary parts. This is used mainly in FFT calculations.

Connectors

Inputs	Type	Outputs	Type
Array of Magnitudes	Float Array	Array of Real parts	Float Array
Array of Phases (0-2pi)	Float Array	Array of Imaginary parts	Float Array

Max

Description

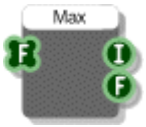
This component compares the two inputs and returns the greater of the two.

The component has template connectors which means it can be used with multiple data types including Floats, Ints and Float/Int arrays. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
Input 1	Template	The greater of the two inputs	Template
Input 2	Template		

Max Float Array



Description

The Max Float Array component will give you the maximum value in an array of floats together with the index in the array at which that maximum occurred.

Connectors

Inputs	Type	Outputs	Type
Float array to examine	Float Array	The index at which the maximum occurs	Int
		The maximum value in the array	Float

MCC-1208FS

Description

This primitive allows you to send and receive data to and from the Measurement Computing Corp. 1208FS data acquisition board.

If you have multiple boards connected to your PC then you should provide the number of the board you want to connect to at the 'Board' input. This number is assigned by Measurement Computing's InstaCal application so refer to that to get the board number if you need it.

The board has 16 digital I/O pins and 8 analog inputs. The digital I/O is split into two banks, A and B. Each bank can be configured as a set of digital inputs or digital outputs.

The analog inputs can be configured as 8 single ended inputs or 4 differential inputs.

To set the I/O configuration you need to supply a string to the I/O input of the component. This should be a comma separated list of any combination of the following:

SE - single ended analog inputs

D - differential analog inputs

AI - set bank A of digital I/O to be inputs

BI - set bank B of digital I/O to be inputs

AO - set bank A of digital I/O to be outputs

BO - set bank B of digital I/O to be outputs

For example:

“SE,AI,BO” would use single ended analog, bank A as digital inputs and bank B as digital outputs.

“D,BI” would use differential analog and bank B as digital inputs.

Note that the I/O string only alters what you specify in the list. If you don't specify what bank A does it will remain as it is – it will not be reset to default settings.

Also note that you need to trigger the Set input in order for changes to take place.

When using differential analog inputs you can set the ranges. To do this use a string formatted as follows:

R[n]=[type]

Where [n] is a single digit in the range 0-3 representing the analog I/O pin you want to configure

and [type] is one of the following strings:

BIP20VOLTS BIP10VOLTS BIP5VOLTS BIP4VOLTS BIP2PT5VOLTS

BIP2VOLTS BIP1PT25VOLTS BIP1VOLTS

So for example, "R2= BIP5VOLTS" would set analog input pin number 3 (so at index 2) to use the range +/-5 volts.

These range settings are combined with the other I/O settings as comma separated values as before.

For example, "AI,BO,R2= BIP5VOLTS".

Note that cbw32.dll must be installed on the host system for this component to work. If you have installed the Measurement Computing drivers then this file should be on your system.

MCC-1608FS

Description

This primitive allows you to send and receive data to and from the Measurement Computing Corp. 1608FS data acquisition board.

If you have multiple boards connected to your PC then you should provide the number of the board you want to connect to at the 'Board' input. This number is assigned by Measurement Computing's InstaCal application so refer to that to get the board number if you need it.

The board has 8 digital I/O pins and 8 analog inputs. The digital I/O can each be configured individually as either a digital input or a digital output. The analog inputs are all configured as single ended.

To set the I/O configuration you need to supply a string to the I/O input of the component. This should be a comma separated list of any combination of the following:

DIn - set digital I/O pin n to be an input where n is the index of the pin (0-7)

DOn - set digital I/O pin n to be an output where n is the index of the pin (0-7)

For example, “DI2,DO4,DI5” would use pins DIO2 and DIO5 as inputs and pin DIO4 as an output.

Note that the I/O string only alters what you specify in the list. If you don't specify what DIO3 does for example it will remain as it is – it will not be reset to default settings. Also note that you need to trigger the Set input in order for changes to take place.

For the analog inputs you can set the ranges. To do this use a string formatted as follows:

Rn=[type]

Where n is a single digit in the range 0-7 representing the analog I/O pin you want to configure and [type] is one of the following strings:

BIP1VOLTS BIP2VOLTS BIP5VOLTS BIP10VOLTS

So for example, "R2= BIP5VOLTS" would set analog input pin CH2 IN to use the range +/-5 volts.

These range settings are combined with the other I/O settings as comma separated values as before.

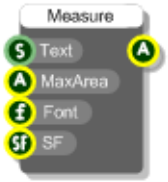
For example, "DI2,DO4,DI5,R2=BIP5VOLTS".

Note that cbw32.dll must be installed on the host system for this component to work. If you have installed the Measurement Computing drivers then this file should already be on your system.

Connectors

Inputs	Type	Outputs	Type
Board number you want to connect to (as assigned by InstaCal) or blank to connect to any board	Int	Status. OK, NOBOARD or the Measurement Computing error number.	String
I/O settings (see above)	String	8 x outputs receiving analog data from the 8 analog inputs on the board in V	Float
Trigger to set the I/O settings	Trigger	8 x outputs receiving the state of the 8digital inputs on the board	Boolean
8 x inputs to change the state of the 8 digital outputs on the board	Boolean	Value at the boards counter input	Int
Trigger to reset the boards counter	Trigger		

Measure Text



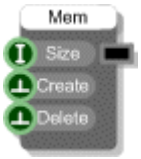
Description

This component measures the area that text will occupy when drawn. You supply the text, font and formatting information together with a maximum bounding rectangle that the text must fit inside. The component then tells you the area within that boundary which the text will occupy.

Connectors

Inputs	Type	Outputs	Type
View to display the text on	View	The same View as the input, but anything connected here is drawn on top	View
Area into which the text is to be drawn	Area		
The text colour	Colour		
The text to be displayed	String		
The font for the text	Font		
Text formatting options	String Format		

Mem Create



Description

This component creates a memory buffer of a particular size. The size is specified in bytes. This means that if you want to store N floats you'll need 4*N bytes.

Connectors

Inputs	Type	Outputs	
Size of the buffer in bytes	Int	Memory buffer	Mem
Trigger to create the buffer. If the buffer has already been created this will reset the buffer	Trigger		
Deletes the buffer	Trigger		

Mem to Float Array

Description

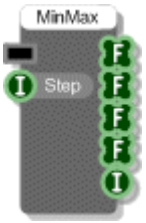
This component converts a memory buffer to an arrays of floats.

This component is useful for extracting raw float data from a wave file either for display or for manipulation.

Connectors

Inputs	Type	Outputs	
Memory buffer	Mem	Float array equivalent	Float Array

Mem to Float Array Min/Max



Description

This component converts a memory buffer to arrays of floats by finding the minimum and maximum values over groups of samples. The size of each group of samples is defined by the Step input. If the Mem contains mono data then the left and right arrays will return the same data.

This component is useful if you want to draw a wave but don't want to draw every single sample in the wave.

Connectors

Inputs	Type	Outputs	
Memory buffer	Mem	Minimum sample per grouping on the left channel	Float Array
Size of each grouping	Int	Maximum sample per grouping on the left channel	Float Array
		Minimum sample per grouping on the right channel	Float Array
		Maximum sample per grouping on the right channel	Float Array
		Number of samples in each array	Int

Message Box

Description

This component displays a Windows message box. You can set the message text, the window title, the icon and the dialog type. The output is a boolean value indicating whether the OK or Yes button was pressed.

Connectors

Inputs	Type	Outputs	Type
Trigger to open the message box	Trigger	Whether the OK/Yes button was pressed	Boolean
The message text	String		
The title for the dialog box	String		
The icon to use (see description above)	Int		
The icon to use (see description above)	Int		
The type of message box (see description above)	Int		

MIDI Aftertouch

Description

This component allows you to extract aftertouch information from MIDI data typically from the pressure applied to an aftertouch sensitive MIDI controller. Aftertouch is specified as an integer in the range 0-127 with 127 representing the highest level of aftertouch and zero meaning no aftertouch at all.

Note: this component is actually responding to channel pressure in that it is not responsive to individual differences in note pressure between different keys and it is responding to MIDI events with the status byte xC0 to xCF and not those between xA0 and xAF.

Connectors

Inputs	Type	Outputs	Type
MIDI data	MIDI	Aftertouch information in the range 0-127	Int

MIDI Control Change

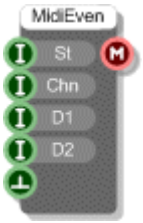
Description

The MIDI control change primitive extracts control change events for the specified control parameter. The parameter is specified as an integer value. To find out what control parameters are available please look them up in the MIDI standard documentation.

Connectors

Inputs	Type	Outputs	Type
MIDI data	MIDI	The current value of the control change parameter in the range 0-127	Int
Control change parameter you want to extract	Int	Trigger when the parameter changes	Trigger

MIDI Event



Description

MIDI data messages are composed of four distinct parts. Each part is an integer value. The parts are as follows:

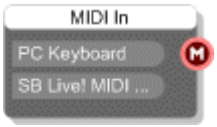
Status	-	(0-127) the type of MIDI event
Channel	-	(1-16) the MIDI channel on which the event was received
Data 1	-	(0-127) the first byte of data received (for note events this is the note pitch)
Data 2	-	(0-127) the second byte of data received (for note events this is the velocity)

The MIDI Event component takes this information and creates a MIDI event message.

Connectors

Inputs	Type	Outputs	Type
Status part (0-127)	Int	MIDI event message	MIDI
MIDI Channel part (1-16)	Int		
Data 1 part (0-127)	Int		
Data 2 part (0-127)	Int		
Trigger to determine when to send the MIDI event	Trigger		

MIDI In



Description

If you want to use MIDI input from an external source then this is the component you need. Each MIDI device installed on your PC is displayed as a button on the body of the component. To select or deselect one of these just click on it.

You can choose to receive input from as many devices as you like. You can also have as many MIDI In components in your schematic as you like. However, you will not be able to select the same MIDI device on more than one of these components.

Connectors

Inputs	Type	Outputs	Type
N/A		MIDI data from the selected device(s)	MIDI

Other Features

The first device on the MIDI In component is PC Keyboard. This isn't a device as such but it allows you to use your PC keyboard as a MIDI input device. This is very handy if you're on a laptop or if you don't have an external MIDI controller.

The PC Keyboard covers 2 consecutive octaves spanning middle C. On a U.S. or U.K. keyboard the lower octave begins at 'Z' with middle C at 'Q' and the upper octave ends at 'I'. All other note associations are shown in the diagram below.

On other keyboard layouts the key positions are the same although the key names may vary.

MIDI In Devices

Description

This component retrieves the number of MIDI In devices on the host PC together with their names and their availabilities.

Connectors

Inputs	Type	Outputs	Type
N/A	N/A	The number of devices	Int
		Array of device names	String Array
		Availability of each device (0=in use, 1=available)	Int Array

MIDI In Select

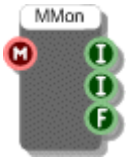
Description

The MIDI In Select component allows you to select a MIDI In device from within a schematic.

Connectors

Inputs	Type	Outputs	Type
The index of the device you want to select. The MIDI In Devices component returns an array of device names which you can refer to to find the appropriate index.	Int	MIDI data from the selected device	MIDI
		The index of the currectly selected device	Int
		Trigger when the chosen device is already in use	Trigger

MIDI Mono



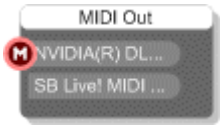
Description

The MIDI Mono primitive generates data that can be used to control a simple mono synth. A more flexible voice management scheme is available through the Midi To Poly voice management module and the MIDI to Voices and Voices to Poly primitives. However, the MIDI Mono primitive is still a viable component to read the number of open note events on a MIDI stream, the frequency of the last played note or the MIDI note number of the most recent note.

Connectors

Inputs	Type	Outputs	Type
MIDI data	MIDI	The pitch of the last note played (60 = Middle C).	Int
		The number of open note events (keys pressed on a MIDI keyboard)	Int
		The normalised frequency of the note played with 1 being half the sample rate	Float

MIDI Out



Description

The MIDI Out primitive allows you to send MIDI data to an external MIDI device. Each MIDI device installed on your PC is displayed as a button on the body of the component. To select or deselect one of these just click on it.

You can choose to send MIDI data to as many devices as you like. You can also have as many MIDI Out components in your schematic as you like. However, you will not be able to select the same MIDI device on more than one of these components.

Connectors

Inputs	Type	Outputs	Type
MIDI data	MIDI	N/A	

MIDI Out Devices

Description

This component retrieves the number of MIDI Out devices on the host PC together with their names and their availabilities.

Connectors

Inputs	Type	Outputs	Type
N/A	N/A	The number of devices	Int
		Array of device names	String Array
		Availability of each device (0=in use, 1=available)	Int Array

MIDI Out Select

Description

The MIDI Out Select component allows you to select a MIDI Out device from within a schematic.

Connectors

Inputs	Type	Outputs	Type
MIDI to send to the selected device	MIDI	The index of the currectly selected device	Int
The index of the device you want to select. The MIDI Out Devices component returns an array of device names which you can refer to to find the appropriate index.	Int	Trigger when the chosen device is already in use	Trigger

MIDI Pitch Bend

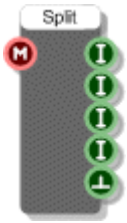
Description

MIDI Pitch Bend gives the amount of pitch bend as an integer from MIDI data typically from the pitch bend wheel of a MIDI controller.

Connectors

Inputs	Type	Outputs	Type
MIDI data	MIDI	Pitch bend in the range 0-16384 with 8192 representing the centre value.	Int

MIDI Splitter



Description

MIDI data messages are composed of four distinct parts. Each part is an integer value. The parts are as follows:

Status	-	(0-127) the type of MIDI event
Channel	-	(1-16) the MIDI channel on which the event was received
Data 1	-	(0-127) the first byte of data received (for note events this is the note pitch)
Data 2	-	(0-127) the second byte of data received (for note events this is the velocity)

The MIDI Splitter component extracts this information from MIDI data and sends the results to its four integer outputs. Note that no triggers are sent from the outputs when they change, instead a single trigger is sent through the fifth output once all the other outputs have been updated.

Connectors

Inputs	Type	Outputs	Type
MIDI data	MIDI	Status part (0-127)	Int
		MIDI Channel part (1-16)	Int
		Data 1 part (0-127)	Int
		Data 2 part (0-127)	Int
		Trigger when a new MIDI event is received	Trigger

MIDI to Multi Voice

Description

The MIDI to Multi Voice primitive takes all the functionality of the MIDI to Voices component but extends it in two ways:

1. It allows multiple voices to be generated in response to a single note
2. It allows you change the characteristics of a voice or voices depending on the note played

As with the MIDI to Voices this component works alongside the Voices to Poly component. You would always pair these components together when using them in a synth.

How it Works

The MIDI to Multi Voice has 4 int array inputs. These come in two pairs one for note key range and one for note velocity. For each pair, one array provides the lower value in the range and one provides the upper value.

The entries in the arrays are indexed from 0 upwards. These indexes define a parameter called a Voice Tag. When you play a MIDI note the MIDI to Multi Voice compares the note to the key and velocity range at each index. If the note falls within the defined boundaries then a voice is generated for that note and that voice is assigned a Voice Tag equal to the value of the index in question.

This is how (1) & (2) above are achieved. If a note matches the range(s) for more than one index then more than one voice will be generated for that note achieving (1) above.

The Voice Tag is passed on to the Voices to Poly where it is accessible as a Poly output. You can use this in your audio processing to make changes to the sound depending on the tag and this achieves (2) above.

To show the effect on the voice tag based on these arrays is much easier using some examples.

EXAMPLE 1

Let's say you have the arrays as follows:

LoKey	HiKey
0	59
60	127

With this configuration notes up to middle C (0-59) will carry a voice tag of zero and notes at middle C and beyond (60-127) will carry a voice tag of 1.

EXAMPLE 2

Now let's look at the velocity ranges:

LoVel	HiVel
0	31
62	63
63	95
96	127

With this configuration you will get voice tags in the range 0-3 and you can use this to increase the volume or to vary filter parameters for example.

EXAMPLE 3

What happens if you use both velocity and key range?

LoKey	HiKey	LoVel	HiVel
0	59	0	63
60	127	0	63
0	59	64	127
60	127	64	127

This time tag 0 will represent low notes with low velocity, tag 1 will represent high notes with low velocity and tags 2 and 3 will be represent low and high notes with high velocity.

EXAMPLE 4

In all the above cases a single note could only fall within the range(s) for any one index. However, you don't have to define your key and velocity ranges so that they are mutually exclusive.

LoKey	HiKey	LoVel	HiVel
0	127	0	127
0	127	0	127

On this occasion any note you play will match the ranges at index 0 and index 1. In this case two voices will be generated for a single note. This allows you to make some nice effects. For example, you can detune the voice with tag 1 slightly or apply a different LFO rate to create a thicker layered sound.

EXAMPLE 5

One ideal application of this new component is sample playback. You can now easily trigger different samples for each key or play a different sample based on velocity.

LoKey	HiKey
60	60
61	61
62	62
63	63

Here notes 60, 61, 62 and 63 will map onto voice tags 0,1,2 and 3 and you can use these as indexes to specify which sample to use in a Wave Array.

Some Points to Note

As you have probably seen from the examples, you don't have to provide both key and velocity information. If you leave one set blank then no restrictions will be applied.

If your key and velocity ranges are set up in such a way that a particular note doesn't fit into any of them then no voices will be generated for that note.

Finally, if you don't specify any key or velocity ranges then every voice will get a tag of zero making the MIDI to Multi Voice essentially behave the same as the MIDI to Voices.

Voice Mangement

This works in the same way as for the MIDI to Voices. You can define a maximum number of voices (this is unlimited). When the maximum number of voices are in use the next note played will ‘steal’ one of the existing voices. Stealing is done on a first in, first out basis.

The Hold Stolen input defines whether notes whose voices are stolen will be held so that, as long as the stolen notes remains on (i.e. no note off has been received) , they can be reinstated when a free voice becomes available.

The R-new input determines whether envelopes are re-triggered when a new note steals a voice.

The R-old input determines whether envelopes are re-triggered when a held note is reinstated.

The NoRpt input allows you to choose to reuse the same voice whenever the same note is repeatedly played instead of repeating the note in a new voice each time (the default behaviour).

NoSus is the same as NoRpt but applies only when MIDI sustain is on.

Connectors

Inputs	Type	Outputs	Type
MIDI data	MIDI	Voice data (currently only the Voices to Poly primitive takes this data type as an input)	Voice
Maximum number of voices that can be playing at any one time.	Int		
Key range lower limits	Int Array		
Key range upper limits	Int Array		
Note velocity lower limits	Int Array		
Note velocity upper limits	Int Array		
Hold Stolen defines whether notes whose voices are stolen will be held	Boolean		
Retrigger New determines whether envelopes are re-triggered when a new note steals a voice	Boolean		
Retrigger Old input determines whether envelopes are re-triggered when a held note is reinstated	Boolean		
NoRpt reuses the same voice(s) when the same note is repeatedly played	Boolean		
NoSus is the same as NoRpt but applies only during MIDI Sustain	Boolean		

MIDI to Voices

Description

The MIDI to Voices primitive component pairs with the Voices to Poly primitive to provide voice managed data.

Note: Currently there is no reason to use this component on its own, instead you should use the MIDI to Poly module which combines the MIDI to Voices and Voices to Poly components.

The inputs to the MIDI to Voices primitive define how the voice management works - by this we mean how the module responds to note messages from the attached MIDI source. You can define a maximum number of voices (this is unlimited). When the maximum number of voices are in use the next note played will ‘steal’ one of the existing voices. Stealing is done on a first in, first out basis.

The Hold Stolen input defines whether notes whose voices are stolen will be held so that, as long as the stolen notes remains on (i.e. no note off has been received) , they can be reinstated when a free voice becomes available.

The R-new input determines whether envelopes are re-triggered when a new note steals a voice.

The R-old input determines whether envelopes are re-triggered when a held note is reinstated.

The NoRpt input allows you to choose to reuse the same voice whenever the same note is repeatedly played instead of repeating the note in a new voice each time (the default behaviour).

NoSus is the same as NoRpt but applies only when MIDI sustain is on.

Connectors

Inputs	Type	Outputs	Type
MIDI data	MIDI	Voice data (currently only the Voices to Poly primitive takes this data type as an input)	Voice
Maximum number of voices that can be playing at any one time.	Int		
Hold Stolen defines whether notes whose voices are stolen will be held	Boolean		
Retrigger New determines whether envelopes are re-triggered when a new note steals a voice	Boolean		
Retrigger Old input determines whether envelopes are re-triggered when a held note is reinstated	Boolean		
NoRpt reuses the same voice when the same note is repeatedly played	Boolean		
NoSus is the same as NoRpt but applies only during MIDI Sustain	Boolean		

Min

Description

This component compares the two inputs and returns the lower of the two.

The component has template connectors which means it can be used with multiple data types including Floats, Ints and Float/Int arrays. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
Input 1	Template	The lower of the two inputs	Template
Input 2	Template		

Min Float Array

Description

This component works out where the minimum value appears in the input array. It also returns the index of the element in the array that corresponds to that minimum value.

Connectors

Inputs	Type	Outputs	
The source array	Float Array	The index of the entry that contains the minimum value	Int
		The minimum value found	Float

Module

Description

Modules are special types of component in that they are defined by their own schematic containing other components and modules. There are no inputs and outputs by default, you add these yourself by placing Module Input and Module Output components inside.

For a complete description of how modules work see Chapter 4 of the main user guide.

Module GUI

Description

The Module GUI component (MGUI for short) gives a module a front panel and allows you to use the low-level GUI components to define the graphics and interaction capabilities.

Note that if you don't want or need to use the GUI components then you can just use the G button on the module action panel to add a front panel (see the Front Panel section in Chapter 4 of the main user guide).

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.

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

The two float inputs allow you to set the position of the module when it appears in a parent front panel. Direct placement using front panel editing is usually preferred to explicit placement but this can sometimes be useful.

If you want to hide the grey module border then you can do this by setting the Border input to False.

Mouse move messages are suppressed by default. By mouse moves we mean movement of the mouse with no buttons held down. When a button is held we call it a mouse drag and these are not suppressed.

The reason for suppressing mouse moves is that because of their frequency they use measurable processing as messages are passed up and down the module hierarchy. Mouse moves are rarely used and so there is no point in having this extra processing when it isn't needed. If you do need to have mouse move handling then set the Mouse input to True.

Connectors

Inputs	Type	Outputs	Type
Connector for displaying this panel on another front panel. Often this is connected to a wireless input.	View	Connect this to other components in order to display them on the front panel or to handle mouse messages	View
Set the x position of the panel in a parent panel	Float	Width of the modules front panel	Float
Set the y position of the panel in a parent panel	Float	Height of the modules front panel	Float
Allows you to hide the grey module border. Useful for creating more compact utility modules	Boolean		
Enables mouse move messages to pass through the View connectors	Boolean		

Module Input

Description

Place a Module Input primitive inside a module to give the module an input connector. The component has a single Temple output connector. To change the type either right-click on the connector and choose a type from the pop-up menu or create a link from the connector to another component and the template will pick up the type.

For more information on module inputs see the section on Inputs and Outputs in the Modules chapter (Chapter 4) of the main user guide.

Connectors

Inputs	Type	Outputs	Type
N/A		Template connector which must be set to the type you require for the module input	Template

Module Output

Description

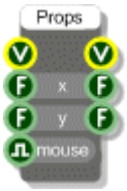
Place a Module Output primitive inside a module to give the module an output connector. The component has a single Temple input connector. To change the type either right-click on the connector and choose a type from the pop-up menu or create a link from the connector to another component and the template will pick up the type.

For more information on module outputs see the section on Inputs and Outputs in the Modules chapter (Chapter 4) of the main user guide.

Connectors

Inputs	Type	Outputs	Type
Template connector which must be set to the type you require for the module output	Template	N/A	

Module Properties GUI



Description

The Properties GUI component (PGUI for short) gives a module a property panel and allows you to use the low-level GUI components to define the graphics and interaction capabilities.

Note that if you don't want or need to use the GUI components then you can just use the P button on the module action panel to add a front panel (see the Properties section in Chapter 4 of the main user guide).

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 property panel or handle mouse messages from it.

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

The two float inputs allow you to set the position of the property panel when it appears in a parent property panel. Direct placement using front panel editing is usually preferred to explicit placement but this can sometimes be useful.

Mouse move messages are suppressed by default. By mouse moves we mean movement of the mouse with no buttons held down. When a button is held we call it a mouse drag and these are not suppressed.

The reason for suppressing mouse moves is that because of their frequency they use measurable processing as messages are passed up and down the module hierarchy. Mouse moves are rarely used and so there is no point in having this extra processing when it isn't needed. If you do need to have mouse move handling then set the Mouse input to True.

Connectors

Inputs	Type	Outputs	Type
Connector for displaying this panel on another property panel. Often this is connected to a wireless input labelled Properties.	View	Connect this to other components in order to display them on the property panel or to handle mouse messages	View
Set the x position of the panel in a parent property panel	Float	Width of the modules property panel	Float
Set the y position of the panel in a parent property panel	Float	Height of the modules property panel	Float
Enables mouse move messages to pass through the View connectors	Boolean		

Module Wireless Output

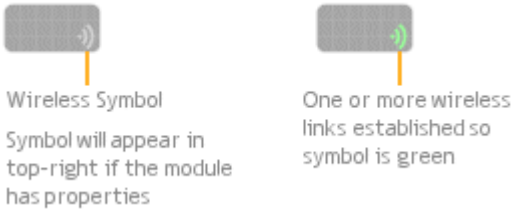
Description

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.

The component has a single Temple input connector. To change the type either right-click on the connector and choose a type from the pop-up menu or create a link from the connector to another component and the template will pick up the type.

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.

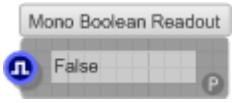


For more information on wireless links see the corresponding section in the Components and Links chapter (Chapter 3) of the main user guide.

Connectors

Inputs	Type	Outputs	Type
Template connector which must be set to the type you want for the wireless output	Template	N/A	

Mono Boolean Readout



Description

The Mono Boolean Readout allows you to inspect the value of a signal from an output connector that's linked into a running mono section. This can be very handy for debugging.

Connectors

Inputs	Type	Outputs	Type
Mono Boolean stream data	Mono Boolean	N/A	

Mono to Float

Description

The Mono to Float component allows you to take Float samples from a Mono signal. The trigger input defines when the sample is taken. This is very useful for visually examining the data passing through a Mono stream.

Connectors

Inputs	Type	Outputs	Type
Mono signal to take samples from	Mono	Current sample	Float
Trigger to say when to take the sample	Trigger		

Mono to Frame

Description

This version of the Mono to Frame component captures buffers of samples at the rate they are processed by the audio engine. Unlike the Mono to Float and Mono to Graph components the Mono to Frame allows you to capture and process every sample that passes through a Mono stream.

The output is a Ruby Frame object. You can examine or process this using a Ruby component. For more information about this see the section on Frames in the Ruby Component chapter of the main user guide.

Connectors

Inputs	Type	Outputs	Type
Mono signal	Mono	Ruby Frame object with a buffer of samples	Ruby Value

Mono to Frame

Description

This version of the Mono to Frame component captures buffers of samples at a rate determined by the Samples input. The Samples input should be an integer value (a Ruby Fixnum) that specifies the number of samples to grab.

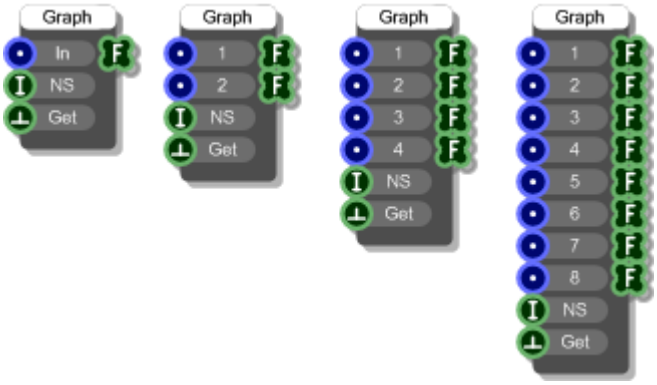
The output will be the last N samples that passed through the Mono input (where N is the number of requested samples).

The output value is a Ruby Frame object. You can examine or process this using a Ruby component. For more information about Frames see the section on Frames in the Ruby Component chapter of the main user guide.

Connectors

Inputs	Type	Outputs	Type
Mono signal	Mono	Ruby Frame object with a buffer of samples	Ruby Value
Ruby value containing the number of samples you want to grab	Ruby Value		

Mono to Graph



Description

The Mono to Graph component allows you to take an array of samples from up to 8 Mono streams. The trigger input defines when the samples are taken. The number of samples taken is defined at the NS input. This component is very useful for graphing a mono signal.

There are four versions of the Mono to Graph. If you want to compare one or more signals you need to use a single Mono to Graph component with multiple inputs. If you use two separate Mono to Graph components then because the Mono section is calculated constantly, even if you trigger the Get input from the same source it is very unlikely that you'll get exactly the same snapshot of samples from both components.

Connectors

Inputs	Type	Outputs	Type
Mono signal to take samples from (up to 8)	Mono	Array of samples taken (up to 8)	Float Array
Number of samples to take	Int		
Trigger to say when to take the samples	Trigger		

Motion Detect

Description

Motion Detect is a video processing component that locates areas of movement in an image.

Each new image is compared with the frame before and if any pixel differs by more than the Delta input value then this is registered as movement and stored in a binary image.

These binary images are stored over a period of time defined by the Duration input (Dur). The images are then faded out with the oldest image being almost transparent and the most recent being fully opaque. These images are overlayed to produce a movement gradient called the Motion History Image (MHI).

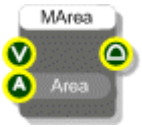
The algorithm looks at the MHI and for each pixel it examines the neighbouring pixels to determine the difference in intensity gradient. If the gradient falls between the MinT and MaxT values then it registers that movement has occurred.

MinT is the minimum MHI duration and MaxT is the maximum MHI duration. Because the gradient exactly corresponds to the duration inside the MHI these values are specified as times in seconds. So differences in intensity between neighbouring pixels are equivalent to differences in time between neighbouring pixels.

Connectors

Inputs	Type	Outputs	Type
The source image you want to process	Bitmap	Motion history image	Bitmap
Threshold for detecting image transitions. Default is 30	Int	Whether any movement was detected	Boolean
Duration of motion image history (seconds). Default is 1sec.	Float	X coordinates of centre detected movements (pixels)	Float Array
Minimum MHI duration (seconds). Default is 0.05	Float	Y coordinates of centre of detected movements (pixels)	Float Array
Maximum MHI duration (seconds). Default is 0.5	Float	Widths of detected areas of movement (pixels)	Float Array
		Heights of detected areas of movement (pixels)	Float Array
		Directions of detected areas of movement (degrees)	Float Array

Mouse Area



Description

Creates an area on a View that receives mouse messages.

Connectors

Inputs	Type	Outputs	Type
View to create the area on	View	The same View as the input, but anything connected here is drawn on top	View
The area to use	Area		

Mouse Drag

Description

The Mouse Drag component allows you to handle drag operations for a mouse area. If you connect this component to a Mouse Area then when a drag operation is in progress the component will send the x and y coordinates of the mouse as the drag progresses.

Connectors

Inputs	Type
Mouse messages from a Mouse Area component	Mouse
Whether to hide the cursor during dragging (0=show, 1=hide and hold, 2=hide)	Int

Outputs	Type
The x-coordinate of the mouse whilst dragging	Float
The y-coordinate of the mouse whilst dragging	Float
Flag indicating if a drag operation is in progress	Boolean
Trigger sent for each mouse movement	Trigger

Mouse LDbl-click

Description

This component allows you to handle left mouse button double-click events for a mouse area. If you connect this component to a Mouse Area then when you double-click on the area the component will output the x and y coordinates of the clicked point.

Connectors

Inputs	Type	Outputs	Type
Mouse messages from a Mouse Area component	Mouse	The x-coordinate of the double-clicked point	Float
		The y-coordinate of the double-clicked point	Float
		Trigger when a double-click occurs	Trigger

Mouse LDown

Description

This component allows you to handle left mouse button down events for a mouse area. If you connect this component to a Mouse Area then when you click on the area the component will output the x and y coordinates of the clicked point.

Connectors

Inputs	Type	Outputs	Type
Mouse messages from a Mouse Area component	Mouse	The x-coordinate of the clicked point	Float
		The y-coordinate of the clicked point	Float
		Trigger when a click occurs	Trigger

Mouse LUp

Description

This component allows you to handle left mouse button up events for a mouse area. If you connect this component to a Mouse Area then when you click on the area and subsequently release the button, the component will output the x and y coordinates of the clicked point.

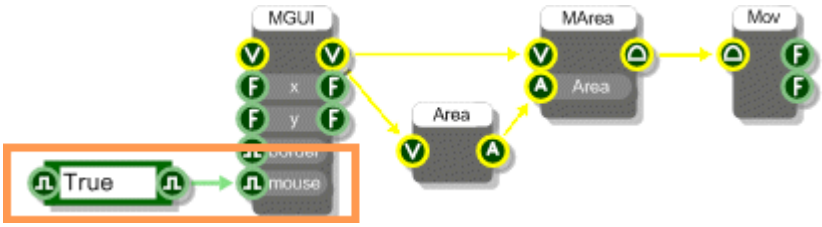
Connectors

Inputs	Type	Outputs	Type
Mouse messages from a Mouse Area component	Mouse	The x-coordinate of the mouse when the button is released	Float
		The y-coordinate of the mouse when the button is released	Float
		Trigger when a mouse up event occurs	Trigger

Mouse Move

Description

The Mouse Move component allows you to track the mouse position as it passes over an area. Note that mouse move message flow is turned off by default for performance reasons so in order to use a Mouse Move component you must ensure that mouse moves are switched on in the MGUI for the module where the mouse area is defined (see picture below).



When the Mouse Move component is configured correctly, the two Float outputs give the position of the mouse as it moves over the defined area.

Connectors

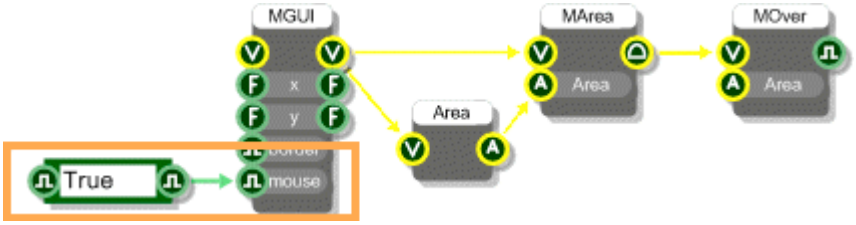
Inputs	Type	Outputs	Type
Mouse messages from a Mouse Area component	Mouse	The x-coordinate of the mouse as it passes over the mouse area	Float
		The y-coordinate of the mouse as it passes over the mouse area	Float

Mouse Over

Description

This component determines whether the mouse is over a particular area in a view. The boolean output will show True when the mouse is over and False otherwise. This output is triggered whenever the state changes.

Note that mouse move message flow is turned off by default for performance reasons so in order to use a Mouse Over component you must ensure that mouse moves are switched on in the MGUI for the module where the mouse area is defined (see picture below).



Connectors

Inputs	Type	Outputs	Type
View on which to monitor the mouse status	View	Whether the mouse is over the area	Boolean
Area of the view to be monitored	Area		

Mouse RDbl-click

Description

This component allows you to handle right mouse button double-click events for a mouse area. If you connect this component to a Mouse Area then when you right double-click on the area the component will output the x and y coordinates of the clicked point.

Connectors

Inputs	Type	Outputs	Type
Mouse messages from a Mouse Area component	Mouse	The x-coordinate of the right double-clicked point	Float
		The y-coordinate of the right double-clicked point	Float
		Trigger when a right double-click occurs	Trigger

Mouse RDown

Description

This component allows you to handle right mouse button down events for a mouse area. If you connect this component to a Mouse Area then when you right click on the area the component will output the x and y coordinates of the right clicked point.

Connectors

Inputs	Type	Outputs	Type
Mouse messages from a Mouse Area component	Mouse	The x-coordinate of the right clicked point	Float
		The y-coordinate of the right clicked point	Float
		Trigger when a right click occurs	Trigger

Mouse RUp

Description

This component allows you to handle right mouse button up events for a mouse area. If you connect this component to a Mouse Area then when you right click on the area and subsequently release the button, the component will output the x and y coordinates of the clicked point.

Connectors

Inputs	Type	Outputs	Type
Mouse messages from a Mouse Area component	Mouse	The x-coordinate of the mouse when the right button is released	Float
		The y-coordinate of the mouse when the right button is released	Float
		Trigger when a right mouse up event occurs	Trigger

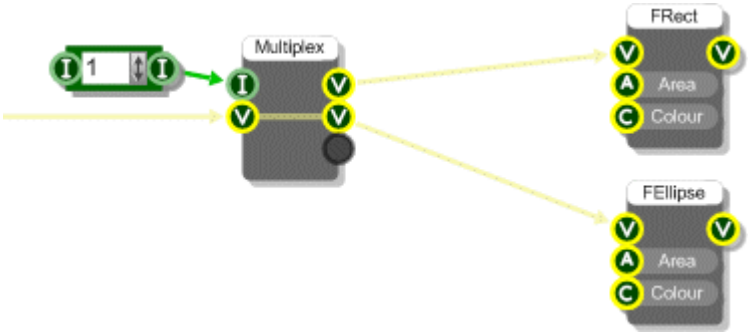
Multiplexer

Description

The Multiplexer component routes a single input to just one of a number of outputs. The component has template connectors which means that you can use it with any connector type. The type is defined when you connect your first link to the component.

When you connect an output a new, unassigned output will appear below it. By continuously connecting unassigned outputs in this way you can build up to the number of outputs you require.

Multiplexers are particularly useful with View connectors for switching between different displays.



Connectors

Inputs	Type	Outputs	Type
Index of selected output (zero based)	Int	Any number	Any Type
Input data	Any Type		

Multiply

Description

This component multiplies the two inputs.

The component has template connectors which means it can be used with multiple data types including Floats, Ints and Float/Int arrays. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
Input 1	Template	The product of the two inputs	Template
Input 2	Template		

Multiply Float Array

Description

This primitive multiplies each entry in the input array by a single float value.

Connectors

Inputs	Type	Outputs	Type
Array to modify	Float Array	Array of multiplied values	Float Array
Float value to multiply by each entry in the array	Float		

Multiply Float Array Pair

Description

This primitive multiplies each entry in the input array by the entry at the same index in a second float array. If the arrays are different sizes then the larger of the two is truncated to the length of the smaller one.

Connectors

Inputs	Type	Outputs	Type
First Array	Float Array	Array of multiplied values	Float Array
Second Array	Float Array		

Netvox Alarm Security

Description

This component allows you to connect to a Netvox alarm security device that is paired with a Netvox USB adapter attached to your PC. Netvox alarm security devices include window and passive infra red sensors.

IMPORTANT: you must first use a Netvox USB component and successfully connect to the USB adapter before attempting to connect to a device.

Before starting the component you need to send the IEEE address of the device to the 'Id' input. The IEEE address is usually written on the device itself.

Trigger the Start input to connect. The Status output will show as "Searching”. If the device can't be found the status will show as “Not Found”. If the status shows “No Network” then the Netvox USB adapter has been disconnected.

On successful startup the device details will be updated. Battery level, device type, tamper , alarm and fault status are all updated.

You have the option to set the Heartbeat via the 'Heart' input. This is the rate at which the device will send signals back to the USB adapter to tell it that it is still alive. It also reports full device details during these signals. The Heartbeat period is in seconds.

To trigger an update manually you can send a trigger to the 'Query' input.

If the device alarm is triggered the 'Alarm' output will send its status immediately (or as soon as the network allows).

If the connection is lost at any time the status will change to “Lost Connection”. Reconnection will start automatically at which point the status will change to “Reconnecting”.

Connectors

Inputs	Type	Outputs	Type
IEEE address of the device you want to connect to	String	The current connection status	String
Trigger to start the component	Trigger	The percentage battery level (-1 means if it could not be read)	Int
Trigger to query the component	Trigger	The type of security device (if applicable)	String
Set the heartbeat (in secs)	Int	The current heartbeat (secs)	Int
		Whether the alarm has been triggered	Boolean
		Whether the device has been tampered with	Boolean
		Whether the device has encountered a fault	Boolean

Netvox Light Sensor

Description

This component allows you to connect to a Netvox light sensor device that is paired with a Netvox USB adapter attached to your PC.

IMPORTANT: you must first use a Netvox USB component and successfully connect to the USB adapter before attempting to connect to a device.

Before starting the component you need to send the IEEE address of the device to the 'Id' input. The IEEE address is usually written on the device itself.

Trigger the Start input to connect. The Status output will show as "Searching". If the device can't be found the status will show as “Not Found”. If the status shows “No Network” then the Netvox USB adapter has been disconnected.

On successful startup the device details will be updated. Battery level and detected light level will be sent to the relevant outputs.

You can set the minimum and maximum time reporting intervals in seconds and the reportable change. These 3 work together as follows:

> *When the detected light intensity is greater than the change in light intensity value specified in reportable change, the device will report the intensity information at the minimum reporting interval specified. With the maximum time interval it's the other way around ie. when the detected light intensity is less than the change in light intensity value specified in reportable change, the intensity information will be reported at the maximum reporting interval specified.*

Updates are triggered automatically based on the above settings. However, you can request an update manually by sending a trigger to the 'Query' input.

If the connection is lost at any time the status will change to “Lost Connection”. Reconnection will start automatically at which point the status will change to “Reconnecting”.

Connectors

Inputs	Type	Outputs	Type
IEEE address of the device you want to connect to	String	The current connection status	String
Trigger to start the component	Trigger	The percentage battery level (-1 means if it could not be read)	Int
Trigger to query the component	Trigger	The brightness level in LUX	Int
The minimum reporting interval (in secs)	Int		
The maximum reporting interval (in secs)	Int		
The reportable change	Int		

Netvox Mains Power Outlet

Description

This component allows you to connect to a Netvox mains power outlet device that is paired with a Netvox USB adapter attached to your PC.

IMPORTANT: you must first use a Netvox USB component and successfully connect to the USB adapter before attempting to connect to a device.

Before starting the component you need to send the IEEE address of the device to the 'Id' input. The IEEE address is usually written on the device itself.

Trigger the Start input to connect. The Status output will show as "Searching”. If the device can't be found the status will show as “Not Found”. If the status shows “No Network” then the Netvox USB adapter has been disconnected.

On successful startup the device details will be updated. Battery level, On/Off status, Current, Voltage, Power and Energy usage will all be sent to the relevant outputs.

You can request an update manually by sending a trigger to the 'Query' input.

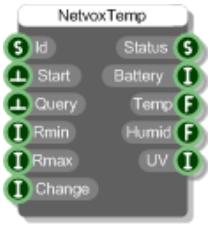
To switch the mains outlet on or off you can send a boolean value to the 'On' input. Trigger the 'Toggle' input to flip the state of the outlet.

If the connection is lost at any time the status will change to “Lost Connection”. Reconnection will start automatically at which point the status will change to “Reconnecting”.

Connectors

Inputs	Type	Outputs	Type
IEEE address of the device you want to connect to	String	The current connection status	String
Trigger to start the component	Trigger	The percentage battery level (-1 means if it could not be read)	Int
Trigger to query the component	Trigger	Whether the outlet is currently on	Boolean
Switch the outlet on or off	Boolean	Current usage in Amps	Int
Toggle the outlet from its current state	Boolean	Voltage level in Volts	Int
		Power consumption in Watts	Int
		Energy usage in kWh	Float

Netvox Temperature Sensor



Description

This component allows you to connect to a Netvox temperature sensor device that is paired with a Netvox USB adapter attached to your PC.

IMPORTANT: you must first use a Netvox USB component and successfully connect to the USB adapter before attempting to connect to a device.

Before starting the component you need to send the IEEE address of the device to the 'Id' input. The IEEE address is usually written on the device itself.

Trigger the Start input to connect. The Status output will show as "Searching". If the device can't be found the status will show as “Not Found”. If the status shows “No Network” then the Netvox USB adapter has been disconnected.

On successful startup the device details will be updated. Battery level, Temperature will be sent to the relevant outputs. Humidity and UV levels will also be reported if supported by the device.

You can set the minimum and maximum time reporting intervals in seconds and the reportable change. These 3 work together as follows:

> *Minimum reporting time interval is the minimum reporting time, in seconds. If this value is set to 0, then there is no minimum limit. Maximum reporting time interval is the fixed reporting time, in seconds. If this value is set to 0, then the device will not report the temperature. Reportable change is the minimum temperature change between the previous and the last detected. If the change in temperature is greater than Reportable Change value, the device issues the report at the minimum reporting time. If the change is less than the reportable change, it will not report until the time reaches the maximum reporting time.*

Updates are triggered automatically based on the above settings. However, you can request an update manually by sending a trigger to the 'Query' input.

If the connection is lost at any time the status will change to “Lost Connection”. Reconnection will start automatically at which point the status will change to “Reconnecting”.

Connectors

Inputs	Type	Outputs	Type
IEEE address of the device you want to connect to	String	The current connection status	String
Trigger to start the component	Trigger	The percentage battery level (-1 means if it could not be read)	Int
Trigger to query the component	Trigger	The temperature level in degrees celcius	Float
The minimum reporting interval (in secs)	Int	The percentage humidity (if supported by device)	Float
The maximum reporting interval (in secs)	Int	The UV level (if supported by device)	Int
The reportable change	Int		

Netvox USB

Description

This component allows you to connect to a Netvox home automation USB adapter.

The adapter installs as a virtual COM port so you need to find out which port it's on before you can use this component.

Send the com port number to the 'Port' input (just the number, not the 'COM' part). You can set the Baud Rate or leave this unconnected to use the default.

Trigger the Start input to connect. The 'On' output will send True if successful or False otherwise. If connected the Status output will show as "Connected". If no USB adapter can be found the status will show as “Device Not Found”.

Once you are connected to an adapter you can use the other Netvox components to connect to devices that are paired with the adapter.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether a connection was successful	Boolean
The COM port number that the adapter is connected to	Int	The current connection status	String
The baud rate in bps (optional). Leave unconnected to use the default baud rate of 115200	Int		

Network Client

Description

This primitive allows you to send and receive data to and from a server across a network.

You can choose whether to create a TCP or UDP connection. You can also specify whether you are communicating via character strings (straight text) or hex strings.

Connectors

Inputs	Type	Outputs	Type
The IP address of the server you want to connect to	String	Whether the connection is open or not	String
The port you want to connect to	Int	Data received in from the server	String
The type of connection, either 'TCP' or 'UDP'	String	Trigger when data has been sent out to the server	Trigger
Trigger to open the connection	Trigger	If the Log input is set to true then the activity log can be accessed from here	String
Trigger to close the connection	Trigger		
The data you want to send	String		
Whether the data being transmitted and received is hex (as opposed to character strings)	Boolean		
Trigger to send the data	Trigger		
Whether you want to log activity across the connection. This can be helpful to make sure everything is working as you expect. The log is accessible through the Log output	Boolean		
Trigger to clear the log	Trigger		

Network Server

Description

This primitive allows you to receive and and send data from and to clients across a network.

You can choose whether to create a TCP or UDP connection. You can also specify whether you are communicating via character strings (straight text) or hex strings.

When sending the data is sent to all clients that are connected to the server.

Connectors

Inputs	Type	Outputs	Type
The port you want the server to use through	Int	Whether the server has started and a connection is open	String
The type of connection, either 'TCP' or 'UDP'	String	The IP address of the server (once the connection has been set up)	String
Trigger to start the server	Trigger	Data received in from any clients	String
Trigger to stop the server	Trigger	Trigger when data has been sent out	Trigger
The data you want to send	String	If the Log input is set to true then the activity log can be accessed from here	String
Whether the data being received and transmitted is hex (as opposed to character strings)	Boolean		
Trigger to send the data	Trigger		
Whether you want to log activity across the server connection. This can be helpful to make sure everything is working as you expect. The log is accessible through the Log output	Boolean		
Trigger to clear the log	Trigger		

New Line

Description

This component outputs the new line character.

Connectors

Inputs	Type	Outputs	Type
N/A		The new line character	String

Norm

Description

Normalises an array of float values. The values are scaled so that they fall between –1 and 1.

Connectors

Inputs	Type	Outputs	Type
Source array	Float Array	Array of normalised values	Float Array

Not

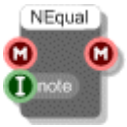
Description

This component returns the opposite value from the input turning True to False and False to True.

Connectors

Inputs	Type	Outputs	Type
Boolean input	Boolean	Inverse of the input value	Boolean

Note Equal



Description

The Note Equal primitive will filter out all MIDI events except those associated with the value fed to the 'note' input (each semitone has a unique MIDI integer value from 0 to 127 with Middle C = 60). MIDI messages that do not relate to notes other than the specified one do not pass through.

Connectors

Inputs	Type	Outputs	Type
MIDI data	MIDI	MIDI data	MIDI
MIDI Note number	Int		

Note Event

Description

The Note Event primitive generates a MIDI note on/off event pair in order to play a note.

Connectors

Inputs	Type	Outputs	Type
MIDI Channel (1-16)	Int	MIDI event message	MIDI
Note number indicating the pitch of the note to be played (60 = middle C)	Int		
Velocity of the note (0-127)	Int		
Duration of the note (milliseconds). Technically it is the period of time between sending a note on event and a corresponding note off event for the same MIDI channel and note values	Int		
Trigger to determine when to send the MIDI note event	Trigger		

Note to Int

Description

Converts a note name to a MIDI note number. For example, C3 is MIDI note 60.

Connectors

Inputs	Type	Outputs	
Note name	String	Equivalent MIDI note number	Int

Note to Int

Description

Converts a note name to a MIDI note number. For example, C3 is MIDI note 60.

Connectors

Inputs	Type	Outputs	
Note name	String	Equivalent MIDI note number	Int

Offline Mode

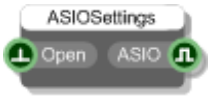
Description

When used inside an exported VST this component will determine whether the host is in offline mode or not. This is useful if you want to apply more cpu intensive processing like over sampling only when the host is using the plugin for rendering or some other non-live purpose.

Connectors

Inputs	Type	Outputs	Type
N/A		Whether the host is in offline mode	Boolean

Open ASIO Settings



Description

Use this in an exported exe to provide access to the ASIO settings dialog. You can use a button or other GUI element to trigger this. The boolean output allows you to determine whether an ASIO driver is currently selected. This is useful because the ASIO settings only apply to the selected ASIO driver and if no ASIO driver is selected the input trigger will do nothing.

Connectors

Inputs	Type	Outputs	Type
Trigger to open the ASIO settings	Trigger	Whether an ASIO driver is currently selected	Boolean

OWL Energy Monitor

Description

This primitive allows you to interface to an OWL Energy Monitor. It will work either with a direct connection via USB cable or using an OWL USB adapter for connecting wirelessly.

Once you have the board connected to your PC you need to start the component by sending a trigger to the 'Start' input. The 'On' output will return True if a connection has been established (False otherwise). If successful the 'Status' output should change from “Not Started” to “Started”.

To read data send a trigger to the 'Read' input. The status should change to “Connected - Waiting for OWL data”. It may take up to a minute or two to find any devices and gather their readings so keep triggering the Read input until your devices are found. If you want to get continuous readings use a Ticker25 or a timer to repeatedly trigger the 'Read' input.

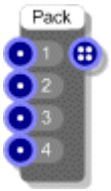
When devices are located the status will change to “Connected - Data read successfully”. The device ids are sent as an array via the 'Devices' output. The corresponding current readings (in Amps) are sent as an array via the 'Readings' output.

If you are connecting wirelessly and pick up more than one device you can focus on a particular one by passing its id into the 'Selected Device' input. The reading from the device will be sent out through the 'Selected Device' output. If no device id is supplied then the 'Selected Device' output will show the reading for the first device.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether a connection was successful	Boolean
Trigger to attempt to read data from connected devices	Trigger	The current connection status	String
Id of the device you want to focus on	Int	Returns an array of found device ids	Int Array
		Returns an array of readings , one for each device	Float Array
		Returns the reading for the device you have selected to focus on or the first device if you haven't made a choice	Float

Pack



Description

The Pack component literally ‘packs’ 4 mono streams into one Mono4 stream. You can then take full advantage of SSE as any stream components you connect up to this will effectively be processing the original 4 mono channels at the same time.

To get the 4 mono streams back again use the Unpack component. Using Pack (and Unpack) can radically increase the efficiency of a Mono section.

Connectors

Inputs	Type	Outputs	Type
First mono stream	Mono	All 4 mono streams ‘packed’ into one stream	Mono4
Second mono stream	Mono		
Third mono stream	Mono		
Fourth mono stream	Mono		

Pen

Description

This component creates a Pen that you can use for drawing. A pen is defined by colour, line thickness and style parameters. The thickness is a float and is in grid squares.

Style can be any one of the following strings:

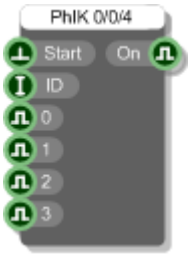
```
solid, dash, dot, dashdot, dashdotdot
```

You can leave the style input disconnected and a solid style will be assumed.

Connectors

Inputs	Type	Outputs	Type
The colour for the pen	Colour	The pen	Pen
Thickness of the pen in grid squares	Float		
Pen line style (see description above)	String		

Phidgets 0/0/4



Description

This primitive allows you to control a Phidgets 0/0/4 Interface Kit board.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

The board has 4 digital outputs. These are mirrored on the component. The 4 boolean inputs will control the 4 digital outputs on the board.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String		
4 x inputs to change the state of the 4 digital outputs on the board	Boolean		

Phidgets 0/0/8

Description

This primitive allows you to control a Phidgets 0/0/8 Interface Kit board.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

The board has 8 digital outputs. These are mirrored on the component. The 4 boolean inputs will control the 8 digital outputs on the board.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String		
8 x inputs to change the state of the 8 digital outputs on the board	Boolean		

Phidgets 0/0/16

Description

This primitive allows you to control a Phidgets 0/0/16 Interface Kit board.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

The board has 16 digital outputs and 16 digital inputs. These are mirrored on the component. The 16 boolean inputs will send data out through the 16 digital outputs on the board. The 16 boolean outputs receive data from the 16 digital inputs on the board.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String	16 x outputs receiving the state of the 16 digital inputs on the board	Boolean
16 x inputs to change the state of the 16 digital outputs on the board	Boolean		

Phidgets 2/2/2

Description

This primitive allows you to control a Phidgets 2/2/2 Interface Kit board.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

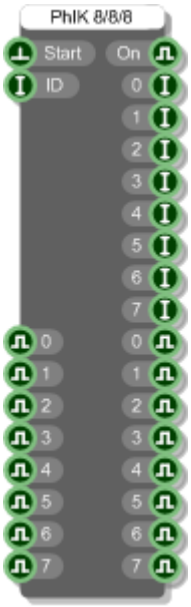
The board has 2 digital outputs, 2 digital inputs and 2 analog inputs. These are mirrored on the component. The 2 boolean inputs will send data out through the 2 digital outputs. The 2 integer outputs receive data from the 2 analog inputs. The 2 boolean outputs receive data from the 2 digital inputs.

The 2 Int outputs are in the range 0-999.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String		
Connection string (optional)	String	8 x outputs receiving analog data from the 8 analog inputs on the board	Int
8 x inputs to change the state of the 8 digital outputs on the board	Boolean	8 x outputs receiving the state of the 8 digital inputs on the board	Boolean

Phidgets 8/8/8



Description

This primitive allows you to send and receive data to and from the Phidgets 8/8/8 Interface Kit board.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

The board has 8 digital outputs, 8 digital inputs and 8 analog inputs. These are mirrored on the component. The 8 boolean inputs will send data out through the 8 digital outputs. The 8 integer outputs receive data from the 8 analog inputs. The 8 boolean outputs receive data from the 8 digital inputs.

The 8 Int outputs are in the range 0-999.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String	8 x outputs receiving analog data from the 8 analog inputs on the board	Int
8 x inputs to change the state of the 8 digital outputs on the board	Boolean	8 x outputs receiving the state of the 8 digital inputs on the board	Boolean

Phidgets Accelerometer

Description

This primitive allows you to interface to the Phidgets Accelerometer board.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

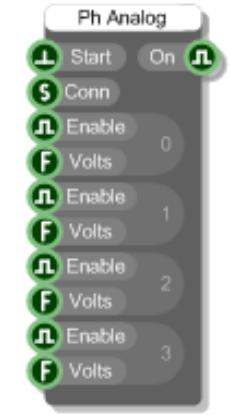
You can set the sensitivity for the three axes. This is a value from 0 to 1 and represents the amount that the acceleration has to change in order for you to be sent an update.

The X, Y and Z outputs then give you the acceleration value for each axis in the range -1 to 1.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String	X-axis acceleration	Float
The amount the X-axis acceleration needs to change by before it's updated	Float	Y-axis acceleration	Float
The amount the Y-axis acceleration needs to change by before it's updated	Float	Z-axis acceleration	Float
The amount the Z-axis acceleration needs to change by before it's updated	Float		

Phidgets Analog



Description

This primitive allows you to control a Phidgets Analog board.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

The board has 4 analog outputs. You can enable and set the voltage for each output independently. Voltages are in the range -10 to +10 volts.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String		
Enable output 0	Boolean		
Voltage for output 0 (+/-10v)	Float		
Enable output 1	Boolean		
Voltage for output 1 (+/-10v)	Float		
Enable output 2	Boolean		
Voltage for output 2 (+/-10v)	Float		
Enable output 3	Boolean		
Voltage for output 3 (+/-10v)	Float		

Phidgets Bridge

Description

This primitive allows you to control a Phidgets Bridge board.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

The board has 4 analog inputs. You can enable and set the gain for each input independently. Gains must be either 1,8,16,32,64 or 128. If you enter a value outside of this range it will be rounded to the nearest of these values. Higher gain gives you lower noise and higher resolution.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String	Value at input 0 in mV/V	Float
Enable input 0	Boolean	Value at input 1 in mV/V	Float
Gain at input 0	Int	Value at input 2 in mV/V	Float
Enable input 1	Boolean	Value at input 3 in mV/V	Float
Gain at input 1	Int		
Enable input 2	Boolean		
Gain at input 2	Int		
Enable input 3	Boolean		
Gain at input 3	Int		

Phidgets Encoder

Description

This primitive allows you to interface to the Phidgets Encoder board.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

You can set the sensitivity for the three axes. This is a value from 0 to 1 and represents the amount that the acceleration has to change in order for you to be sent an update.

The X, Y and Z outputs then give you the acceleration value for each axis in the range -1 to 1.

Connectors

Inputs	Type
Trigger to start the component	Trigger
Connection string (optional)	String
Zero based index of the encoder you want to use (some boards support multiple encoders)	Int
Power on or off the specified encoder	Boolean
Set the position of the specified encoder	Int

Outputs	Type
Whether the board is connected and on	Boolean
Number of encoders supported by the connected board	Int
Number of digital inputs supported by the connected board	Int
Position of the specified encoder	Int
The change since the last recorded encoder change	Int
The elapsed time in milliseconds since the last recorded encoder change	Int
The index position of the encoder if supported	Int
State of the boards digital input 0	Boolean
State of the boards digital input 1	Boolean
State of the boards digital input 2	Boolean
State of the boards digital input 3	Boolean

Phidgets Frequency Counter

Description

This primitive allows you to control a Phidgets Frequency Counter board.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

The board has two channels for measuring. For each one you can enable the channel. You can set the Filter to zero crossing or logic level (0=zero crossing, 1=logic level), depending on your sensor. You can set the timeout in milliseconds (the default is 1000 if you leave this input empty). The reset trigger will reset the counter.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String	Frequency calculated at channel 0 in Hz	Float
Enable channel 0	Boolean	Number of pulses counted for channel 0	Int
Set the filter for channel 0 (0=zero crossing, 1=logic level)	Int	Total time spent counting in microseconds for channel 0	Int
Timeout (ms) for channel 0	Int	Frequency calculated at channel 1 in Hz	Float
Reset the counter for channel 0	Trigger	Number of pulses counted for channel 1	Int
Enable channel 1	Boolean	Total time spent counting in microseconds for channel 1	Int
Set the filter for channel 1 (0=zero crossing, 1=logic level)	Int		
Timeout (ms) for channel 1	Int		
Reset the counter for channel 1	Trigger		

Phidgets GPS

Description

This primitive allows you to control a Phidgets GPS board.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

The component reads latitude, longitude, altitude, velocity and heading information from the board. It also reports whether a fix has been achieved.

Connectors

Inputs	Type
Trigger to start the component	Trigger
Connection string (optional)	String

Outputs	Type
Whether the board is connected and on	Boolean
Whether a fix has been achieved	Boolean
Latitude of position in degrees North	Float
Longitude of position in degrees East	Float
Altitude in metres above sea level	Float
Velocity in km/h	Float
Heading in degrees	Float

Phidgets IR Transmit and Receive

Description

This primitive allows you to interface to the Phidgets IR board.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

To send an IR code you need to attach a String to the Code input. This must contain the full NEC encoding for the code you want to send . An example code would be:

08b750af,32,2,2,108500,501,8980,4590,501,1761,501,626,8980,2333,501,|,0,0,0,0

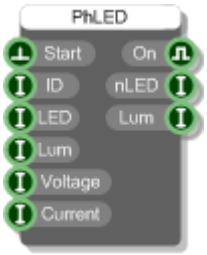
To send the code simply trigger the Send input.

When IR codes are received you'll get the short hex code at the Hex output and the full NEC encoding at the Code output. The Rept output gives you the repeat count for a code – the number of times it has been repeated if say a transmit button on a remote was held down.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String	Short hex code of any IR received	String
Full NEC encoding of the IR code you want to transmit	String	Full NEC encoding of any IR receieved	String
Trigger to send the code	Trigger	Repeat count of any IR code received	Int

Phidgets LED 64



Description

This primitive allows you to interface to the Phidgets LED 64 board.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

This is a very simple component to use. Simply specify the LED you want to control using the LED input. This should be a value from 0 to 63. Then set the luminance using the Lum input. Luminance is in the range 0 to 100 with 0 being off and 100 being fully on.

The current luminance of the specified LED is also given at the Lum output.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String	Number of LEDs supported by the board	Int
Index of the LED you want to control (0-63)	Int	Luminance of the currently selected LED	Int
Luminance level you want to set the specified LED to (0-100)	Int		
Set the voltage for ALL LEDs. You can only choose certain voltages so this input takes values 0-3 where 0=1.7v, 1=2.75v, 2=3.9v and 3=5v.	Int		
Set the current for ALL LEDs. You can only choose certain currents so this input takes values 0-3 where 0=20mA, 1=40mA, 2=50mA and 3=80mA.	Int		

Phidgets Motor Control

Description

This primitive allows you to interface to the two Phidgets Motor Control boards (both 2 motor HC and 4 motor LC).

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

Specify the index of the motor you want to control using the Motor input. This should be a value from 0 to 1 if using the two motor board and 0-3 for the 4 motor.

You can set the velocity and acceleration for the motor using the Vel and Accel inputs.

Current velocity is given at the Vel output as well as minimum and maximum acceleration capabilities.

You can set the braking from zero to 100% and switch back EMF sensing on or off. If the motor you're using has an encoder then you can set this using the Encoder input.

If the board has digital inputs then the status of these is reflected by the four boolean outputs labeled 0-3.

If the board has analog inputs then the status of these is reflected by the two integer outputs labeled A0 and A1.

Encoder and back EMF values can also be read if applicable.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String	Number of motors supported by the connected board	Int
Index of the motor you want to control (0-3)	Int	Current velocity of the selected motor	Float
Velocity you want to set for the selected motor	Float	Minimum supported acceleration for the selected motor	Float
Acceleration you want to set for the selected motor	Float	Maximum supported acceleration for the selected motor	Float
Braking amount 0-100%	Float	State of the boards digital input 0	Boolean
Set the encoder on the motor if there is one	Int	State of the boards digital input 1	Boolean
Switch back EMF sensing on or off	Boolean	State of the boards digital input 2	Boolean
		State of the boards digital input 3	Boolean
		Value at first analog input	Int
		Value at second analog input	Int
		Encoder position (if any)	Int
		Back EMF value	Float

Phidgets RFID

Description

This primitive allows you to interface to the Phidgets RFID board.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

The Ant input controls whether the RFID sensor is on or off. You can switch the on board LED on and off via the LED input.

The board has two digital outputs that you can control using the '0' and '1' inputs.

When an RFID tag passes close enough to the sensor it's id is sent to the TagId output. The Found output will change from false to true when a tag is in close proximity.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String	Whether the RFID sensor is on or off	Boolean
Switch the RFID sensor on or off	Boolean	Whether there is an RFID tag in close proximity	Boolean
Switch the on board LED on or off	Boolean	Id code of any detected RFID tag	String
Change the state of the first digital output	Boolean		
Change the state of the second digital output	Boolean		

Phidgets Servo Advanced

Description

This primitive allows you to interface to the two Phidgets Advanced servo boards (both 1 motor and 8 motor).

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

Specify the index of the servo you want to control using the Servo input. This should be a value from 0 to 7 depending on the board you have connected.

You can set the servo type using the Type input. This is optional and if left unconnected will adopt the default type settings. The available types are:

1 Default - originally based on the Futaba FP-S148

2 Raw US mode - all position, velocity, acceleration functions are specified in microseconds rather then degrees

3 HiTec HS-322HD Standard Servo

4 HiTec HS-5245MG Digital Mini Servo

5 HiTec HS-805BB Mega Quarter Scale Servo

6 HiTec HS-422 Standard Servo

7 Tower Pro MG90 Micro Servo

8 HiTec HSR-1425CR Continuous Rotation Servo

9 HiTec HS-785HB Sail Winch Servo

10 HiTec HS-485HB Deluxe Servo

11 HiTec HS-645MG Ultra Torque Servo

12 HiTec HS-815BB Mega Sail Servo

13 Firgelli L12 Linear Actuator 30mm 50:1

14 Firgelli L12 Linear Actuator 50mm 100:1

15 Firgelli L12 Linear Actuator 50mm 210:1

16 Firgelli L12 Linear Actuator 100mm 50:1

17 Firgelli L12 Linear Actuator 100mm 100:1

The other connectors allow you to control or inspect the state of the specified servo. There are all explained in the Connectors table below.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String	Number of servos supported by the connected board	Int
Index of the servo you want to use	Int	Whether the selected servo is moving or not	Boolean
Type of servo. This is optional. See above for supported types	Int	Current position of the selected servo	Float
Whether the servo is powered on or not	Boolean	Current velocity of the selected servo	Float
Whether to use ramping (ie. velocity and acceleration settings)	Boolean	Minimum supported velocity for the selected servo	Float
Set the position of the specified servo	Float	Maximum supported velocity for the selected servo	Float
Set the velocity of the specified servo (only used if ramping is on)	Float	Minimum supported acceleration for the selected servo	Float
Set the acceleration of the specified servo (only used if ramping is on)	Float	Maximum supported acceleration for the selected servo	Float
Set the minimum position of the specified servo	Float	Returns the minimum position of the specified servo	Float
Set the maximum position of the specified servo	Float	Returns the maximum position of the specified servo	Float

Phidgets Spacial

Description

This primitive allows you to interface to the two Phidgets Spacial boards (0/0/3 and 3/3/3).

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

Connectors

Inputs	Type
Trigger to start the component	Trigger
Connection string (optional)	String
Set the rate at which data is received from the board in milliseconds. Leave disconnected or set to zero for maximum rate	Float
Trigger to zero the gyroscope	Trigger

Outputs	Type
Whether the board is connected and on	Boolean
Number of acceleration axes supported by the board	Int
Number of gyroscope axes supported by the board	Int
Number of compass axes supported by the board	Int
X-axis acceleration (+/-5G)	Float
Y-axis acceleration (+/-5G)	Float
Z-axis acceleration (+/-5G)	Float
X-axis gyroscope (degrees)	Float
Y-axis gyroscope (degrees)	Float
Z-axis gyroscope (degrees)	Float
X-axis compass (gauss)	Float
Y-axis compass (gauss)	Float
Z-axis compass (gauss)	Float
Pitch (degrees)	Float
Roll (degrees)	Float
Bearing (degrees)	Float

Phidgets Stepper Controller

Description

This primitive allows you to interface to the two Phidgets Stepper Controller boards (both 1 motor bipolar and 4 motor unipolar).

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

Specify the index of the motor you want to control using the Motor input. This should be a value from 0 to 3 depending on the board. You can set the target position for the motor as well as velocity and acceleration using the Pos, Vel and Accel inputs.

The Curr input sets the current and so allows you to control torque. If the board has digital inputs then the status of these is reflected by the four boolean outputs labeled 0-3.

The Zero input will zero the position counter thus making the current motor position the initial reference point.

Connectors

Inputs	Type
Trigger to start the component	Trigger
Connection string (optional)	String
Index of the motor you want to control (0-3)	Int
Position you want to set for the selected motor	Float
Velocity you want to set for the selected motor	Float
Acceleration you want to set for the selected motor	Float
Current you want to set for the selected motor	Float
Trigger to zero the position	Trigger

Outputs	Type
Whether the board is connected and on	Boolean
Number of motors supported by the board	Int
Whether the selected motor is moving or not	Boolean
Current position of the selected motor	Float
Current velocity of the selected motor	Float
Minimum supported velocity for the motor	Float
Maximum supported velocity for the motor	Float
Minimum supported acceleration for the motor	Float
Maximum supported acceleration for the motor	Float
Minimum supported current for the selected motor	Float
Maximum supported current for the selected motor	Float
Minimum supported position for the selected motor	Float
Maximum supported position for the selected motor	Float
State of digital input 0	Boolean
State of digital input 1	Boolean
State of digital input 2	Boolean
State of digital input 3	Boolean

Phidgets Temperature

Description

This primitive allows you to interface to the two Phidgets Temperature Sensor boards (both 1 input and 4 input).

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

The four Type inputs allow you to specify the type of thermocouple you have attached to input. This is a string input and can be 'K','J','E' or 'T'. If no type is provided it is assumed to be type 'K'.

The four Snsy inputs allow you to set the sensitivity for each input i.e. how much the temperature needs to change before an update in temperature will be received. The default is 0.5 degrees Celcius.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String	Ambient temperature of the board in degrees Celcius	Float
Type of thermocouple attached to input 1. Either 'K','J','E' or 'T'.	String	Temperature at input 1 in degrees Celcius	Float
Type of thermocouple attached to input 2. Either 'K','J','E' or 'T'.	String	Temperature at input 2 in degrees Celcius	Float
Type of thermocouple attached to input 3. Either 'K','J','E' or 'T'.	String	Temperature at input 3 in degrees Celcius	Float
Type of thermocouple attached to input 4. Either 'K','J','E' or 'T'.	String	Temperature at input 4 in degrees Celcius	Float
Sensitivity of input 1	Float		
Sensitivity of input 2	Float		
Sensitivity of input 3	Float		
Sensitivity of input 4	Float		

Phidgets Text LCD

Description

This primitive allows you to interface to the Phidgets Text LCD and Text LCD Adapter boards.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

The Text LCD Adapter board allows you to control up to two screens. To choose the screen send either 0 (first screen) or 1 (second screen) to the Screen input.

To tell FlowStone what screen size you are using with your adapter, use the Size input. This takes a string of the format '*R*x*C*' where *R* is the number of rows and *C* is the number of columns. For example, '2x20' for a screen with 2 rows and 20 columns.

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String		
The text or character you want to show on the display. Note that this will wrap from the first to the second line if no line break is included in the string	String		
Position where you want to start displaying the text. 0-19 for the first row and 20-39 for the second row	Int		
Controls the cursor. 0=off, 1=on, 2=blinking	Int		
Whether to turn the back light on or off	Boolean		
The contrast level (0-255)	Int		
The brightness level (0-255)	Int		
The screen you want to address, 0 or 1 (adapter board only)	Int		
The size of screen you are using (adapter board only)	String		

Phidgets Touch Linear/Circular

Description

This primitive allows you to interface to the Phidgets Linear and Circular Touch Sensor boards.

Once you have the board connected to your PC you need to start the component by sending a trigger to the first input. The 'On' output will return True if a connection has been established (False otherwise).

If you have multiple boards connected to your PC then you should provide the unique serial number of the board you want to connect to at the 'Conn' (connection string) input. If you want to connect to a device on a remote server provide the device serial, IP and port or server name and also a password if necessary separated by commas. Use -1 as the serial if you want to connect to the first device found.

Example connection strings:

“196491” or “635824,myserver” or “-1,myserver,password” or “142567,192.168.1.1,800,password”

Connectors

Inputs	Type	Outputs	Type
Trigger to start the component	Trigger	Whether the board is connected and on	Boolean
Connection string (optional)	String	Returns True when in close proximity to the sensor	Boolean
		Returns True when touching the sensor	Boolean
		The value corresponding to the touch point	Int

Pitch to Frequency

Description

Converts a pitch value to a frequency. There are two versions of this component. The stream version should be used in Mono and Poly sections and performs the calculation at sampling rate. The Float version should be used for static calculations.

Connectors

Inputs	Type	Outputs	Type
Pitch	Stream/Float	Equivalent frequency	Stream/Float

Pixel to Grid

Description

The Pixel to Grid components convert values in Pixels to values in Grid Squares. There are two versions, one for Floats and one for Areas.

Connectors

Inputs	Type	Outputs	Type
Float or Area in pixel units	Float/Area	Float or Area in grid square units	Float/Area

Plugin Folder

Description

The plugin folder component gives you the folder where your exported exe resides.

When inside FlowStone the folder given is whatever you have set up as the target folder for exported executables. If you wish, you can provide a test folder that will be used only when working within FlowStone.

Connectors

Inputs	Type	Outputs	
Optional test folder path	String	Folder path for your exported exe	String

Point Array Lines

Description

The Point Array Lines component draws a line through a set of points. This is similar to the Graph Lines component. However, in this case both the x and y coordinates are supplied. As with the Graph Lines component, the y-coordinates are assumed to be in the range 0 to 1 (–1 to 1 if the graph is centred on zero). This means that you may need to use the Norm component if you want to make sure that your values are all visible. The x-coordinates are automatically normalised so that they fit within the area of the graph.

By default the line drawn through the points is made from straight line segments. You can also choose to draw a best fit curve through the points by setting the Curve input to True.

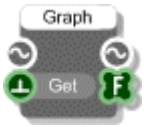
To generate a Point Array to use as input use the Graph to Point Array component.

Connectors

Inputs	Type	Outputs	Type
View to draw onto	View	The same View as the input, but anything connected here is drawn on top	View
The area of the view that the graph will be drawn into	Area		
The array of points	Point Array		
The pen defining the colour, thickness and style of the lines	Pen		

True if the graph is centred on zero, False otherwise	Boolean
True if you want a curved line, False otherwise	Boolean

Poly to Graph



Description

The Poly to Graph component takes the first 2048 samples from a Poly stream and puts them in an array. Poly sections are only active when voices are active and for this component the samples are the first ones generated by the first note played only.

The Get trigger determines when the data is sent to the Float Array output but does not affect which samples are taken – these are always the first 2048 in the signal no matter when the trigger is fired.

Connectors

Inputs	Type	Outputs	Type
Poly signal to take samples from	Poly	Poly signal passing through	Poly
Trigger to say when to send the samples to the Float Array output	Trigger	Array of samples taken	Float

Poly to Mono

Description

The Poly to Mono primitive is an essential part of any polyphonic synth. It acts as a voice combiner adding together the independent signals from each Poly channel and producing a single Mono stream of data.

Note: A Poly to Mono component must have a MIDI to Poly module (or a Voices to Poly) somewhere in the Poly section that precedes it. It can also only connect into one such section – no two MIDI to Poly modules can feed into the same Poly to Mono.

Connectors

Inputs	Type	Outputs	Type
Poly data	Poly	Single Mono stream	Mono
		Assembler code generated by the Poly section that connects into it	String

Poly to PolyInt

Description

The Poly to PolyInt primitive converts Poly signals to PolyInt signals by rounding float values to the nearest integer above or below.

Example

A float value of 1.4 would be converted to an integer value of 1. However, a float value of 1.6 would be converted to an integer value of 2 because 1.6 is closer to 2 than it is to 1.

Note that to round to the nearest integer below all you need to do is subtract 0.5 from the Poly signal before running it through the Poly to PolyInt component.

Connectors

Inputs	Type	Outputs	Type
Poly stream (which contains floating point numbers)	Poly	PolyInt stream containing the same numbers rounded to the nearest integer value	PolyInt

PolyInt to Poly

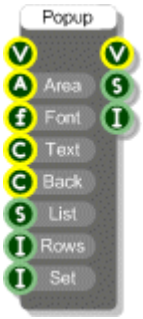
Description

The PolyInt to Poly primitive converts PolyInt signals to Poly signals by converting the integer values to their equivalent in floating point form.

Connectors

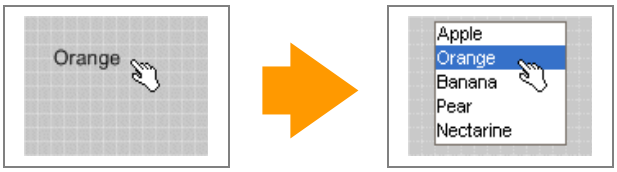
Inputs	Type	Outputs	Type
PolyInt stream (which contains integers)	PolyInt	Poly stream containing the same numbers in floating point format	Poly

Popup List Control



Description

The Popup List Control defines a popup list of selectable values. You need to define an area on the View where the control is to appear and supply a list of options (via a comma separated string). Having done this you can then click on the control and, as you hold the mouse down, the list will pop up. By moving the mouse up and down you can scroll through the options. You then release the mouse to accept the current selection.



The component outputs the selected string and the index of the selection (which is zero based).

Connectors

Inputs	Type	Outputs	Type
View on which to display the control	View	The same View as the input, but anything connected here is drawn on top	View

The position and size of the control	Area	The selected item	String
The font to use for displaying the selected item on the control	Font	The index of the selected item in the list	Int
The text colour	Colour		
The background colour of the pop up list	Colour		
A comma separated list of entries for the pop up list	String		
The maximum number of rows to show in the pop up list (default is 8)	Int		
Set the selected item	Int		

PPQ Pos

Description

The PPQ Pos primitive outputs the current songs quarter position when your VST is used within a host. This is a floating point value, for more details see the VST SDK documentation

Connectors

Inputs	Type	Outputs	Type
N/A		PPQ position	Stream

Preset Manager

Description

The Preset Manager component controls preset changes within a schematic.

The component stores the program names but the data is stored in Preset Parameter, Preset Parameter array and Preset String components. The Preset Manager notifies these parameter components via its Preset output connector. Mostly this information is transferred via use wireless links.

The Preset Manager also controls parameter updates. The Lock input determines whether changes to parameters are discarded between program changes. The Save input can be triggered regardless of the lock state in order to save any changes made.

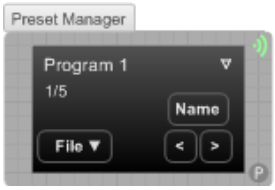
The Before and After Program Change trigger outputs are very useful if for delaying global calculations until after a program change has occurred. If the calculations depend on several parameters then you can wait until all parameters have been updated as a result of the program change before doing any calculations.

You must have one and only one preset manager per module if you want to have preset support. In the majority of cases the Preset Manager module will be all you need.

Connectors

Inputs	Type	Outputs	Type
Nprgs is the number of programs you want	Int	Connector for communicating with Preset Parameter components	Preset
Set the currently selected program	Int	Index of the currently selected program	Int
Set the name of the current program (24 chars max)	String	The current program name	String
Whether the programs are locked so that data changes are not altered permanently	Boolean	List of all program names	String
Trigger to save any parameter changes made regardless of lock state	Trigger	Trigger sent just before a program change occurs	Trigger
Set all the program names at once using a separated string	String	Trigger sent after a program change occurs	Trigger

Preset Manager (module)



Description

The Preset Manager module is the key to adding preset handling to your plugins. To add preset support simply drop a Preset Manager inside your plugin module.

The Preset Manager is wireless and will establish a Preset type connection with all the preset parameters and arrays inside the standard knob, slider and other built in modules. These connections are used to manage preset changes and other such operations.

The number of programs and their names are set via the properties panel. You can also save and load preset data and set program names using text files via the File menu (see Preset Text Files below).

Connectors

Inputs	Type	Wireless Outputs	Type
N/A		Preset connection	Preset

Preset Text Files

The software uses a very flexible system for managing preset data. The reason for using this system instead of the standard fxb/fxp system is that it is much more adaptive to the synth building process.

The fxb/fxp system assumes a fixed format. However, when you're developing a plugin the structure can often change. If we used a rigid format like fxb/fxp your data would be unusable after making any changes.

Using the system we have you can save preset data part way through development and still retain that data as your plugin evolves.

Using

The Preset Manager allows you to load and save preset data to and from text files for storage or for modification. These features are accessed via the File menu on the Preset Manager front panel. You can save/load all the programs or you can choose to save the current program or load a single program into the current one.

File Format

The preset data is saved in a simple tabular format and stored in a text file for easy editing externally. Best to use a spreadsheet program like Microsoft Excel™ to view or manipulate the data. The example below shows an excerpt from the preset text file for a synth.

```
Preset Table Format

Number of programs = 32

Number of parameters = 26


Parameter                 Type     Tekno Honk  Low Distant  Rez Sweeper …
OSC1-Detuner-Fine         Float      0.289429          0.5        0.2625 …
OSC1-Detuner-Octave       Float      0.444445     0.333333      0.666667 …
OSC2-Detuner-Fine         Float      0.752513          0.5        0.7875 …
OSC2-Detuner-Octave       Float      0.444445     0.333333      0.666667 …
OSC1-Waveform             Float           0.2          0.8           0.2 …
OSC2-Waveform             Float           0.2          0.4           0.2 …
OSC1-Volume               Float      0.644708          0.5         0.575 …
OSC2-Volume               Float           0.5          0.0           0.5 …
  ⋮                         ⋮             ⋮            ⋮             ⋮
```

The first 3 rows are always the same except for the number of programs and parameters which will obviously vary.

There is then a blank line followed by the table of preset data. The first column is the preset parameter name. **Parameter names must be unique**. If you find any that are the same then you need to go back to your schematic and change them.

The next column is the data type. Currently this can only be one of the following:

```
String
Float
Float Array (N)          (where N is the size of the float array)
```

For Float Arrays there is an entry in the parameter column for the name followed by a numbered entry for each element in the array. These numbered entries do not count when considering uniqueness of parameter names.

The rest of the columns define the parameter data for each program. If you've saved just the current program then you'll just have one column.

Manipulating

One advantage of the preset text file system is that you can manipulate the files before re-loading. Possible uses include rearranging program order or collating programs from separate files into one. You can also just update a selection of parameters by deleting the rows for the parameters you don't want to affect.

If you make any changes to the number of parameters or programs then you need to make sure you update the figures for these in the 2nd and 3rd lines of the file.

At the top of each program column is the program name. You can change these names in the text file and then load them back in so that all your program names get updated.

Loading

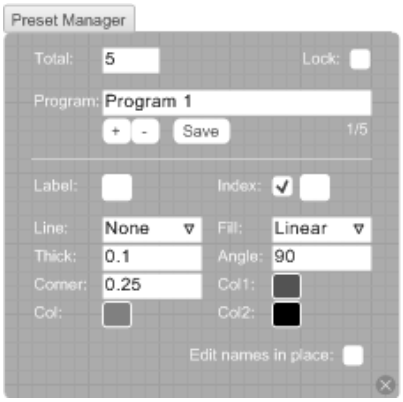
It's useful to understand what happens when a preset text file is loaded into a schematic as you can use this to your advantage under certain circumstances.

First the number of programs and number of parameters are read in. The software then cycles through each parameter in turn. It reads the parameter name from the Parameter column and then tries to find a preset parameter in your schematic that matches the name. If it fails to find a match the parameter data is ignored. This means that if you have deleted something from your schematic the preset data will still load in correctly.

If your schematic contains new preset parameters not included in the file then these will not be affected. Having loaded in your previous preset data you can then make adjustments to the new parameters and save out a new file to reflect the schematic changes you have made.

One other point to note is that the preset loading will never create new programs. If your data file has more programs than you have set in the Preset Manager in your schematic then the extra programs will be ignored. If you want to use these programs just go to your schematic and increase the number of programs as required.

Properties



The Total box is where you define how many programs you require. New programs are automatically assigned program names. You can cycle through the programs using the + and – buttons. Program names can then be changed in the Program box.

If you check the Lock box then the preset manager will only save changes when you press the save button (this includes changes to both preset parameters via knobs/sliders and to program names). If you move from one program to the next without saving any changes will be lost.

For the background you can set the outline style from the Line drop list. There are options for solid, dashed, dotted lines or you can choose to have no outline at all. Line thickness is specified in grid squares. The Corner size is also specified in grid squares and determines how rounded the corners are.

There are several background Fill options. You can have a solid, linear or radial gradient fills. You can also choose to have no fill colour. For a solid fill only the Col1 colour applies. For the two gradient fills Col1 and Col2 define the boundary colours. For linear gradients you can also specify the angle of the gradient.

Preset Text File

Description

The Preset Text File component allows you to transfer preset data to and from a text file. The format used is very flexible and has been designed with the plugin developer in mind as it allows the structure of the plugin to change without losing the preset data.

The component accesses and updates the data stored in Preset Parameter and Preset Parameter array components. It notifies these parameter components via its Preset output connector. Mostly this information is transferred via use wireless links.

Connectors

Inputs	Type	Outputs	Type
Path to the text file on the hard disk	String	Connector for communicating with Preset Parameter components	Preset
Trigger to save the preset data to the file	Trigger	All program names in the text file (use a Text component to view these)	String
Trigger to load preset data from the text file	Trigger		
Whether to save just the currently selected program	Boolean		

(otherwise all are saved)

Whether to save or load preset order information	Boolean
Program names to use when saving (usually come from the Preset Manager)	String

File Format

The file format used is very easy to understand. It is tab delimited and it's best viewed in a spreadsheet application like Microsoft Excel™. Here's an extract from an example synth:

Preset Table Format

Number of programs = 32

Number of parameters = 26

Parameter	Type	Tekno Honk	Low Distant
OSC1-Detuner-Fine	Float	0.289429	0.5
OSC1-Detuner-Octave	Float	0.555556	0.444444
OSC2-Detuner-Fine	Float	0.752513	0.5
OSC2-Detuner-Octave	Float	0.555556	0.444444
OSC1-Waveform	Float	0.2	0.8
OSC2-Waveform	Float	0.2	0.4
OSC1-Volume	Float	0.644708	0.5
OSC2-Volume	Float	0.5	0
Filter Envelope-Attack	Float	0.075	0
Filter Envelope-Decay	Float	0.760487	0

continues right

continues down

The first three lines give the table format, number of programs and number of parameters. After that comes a table which contains all the preset data. The first column shows the preset parameter name (this must be unique for each parameter). The second column shows the data type – either String, Float or Float Array. The columns after that show the data for each program and are headed by the program name.

Parameter Order

If the Preset Text File Order input was set to True there will be an additional column after the Type called Order. Initially this will show –1 for all parameters. To specify the order of parameters as they will appear in the host you should assign a number to each parameter. Parameters with the lowest numbers will appear at the top of the parameter list. Parameters left at –1 will be highest on the list.

Reloading Parameter Data

When the text file is loaded back in the software will take each parameter in turn and locate the Preset Parameter or Preset Parameter Array component with the same name in your schematic and load in the data for all the programs. If it can't find a match the data is ignored.

Program name data is also loaded and is sent to the Program Names output on the Preset Text File component.

PS2 Lynxmotion Controller

Description

The PS2 Lynxmotion Controller component allows you to receive input data from a connected Lynxmotion PS2 controller.

To use this you must have an appropriate PS2 controller attached to your PC. If you trigger the Connect input on the component it will pick up the controller and you can then use the outputs to respond to controller input.

NOTE: the Analog option on the controller must be selected in order for it to work with FlowStone.

Connectors

Inputs	Type	Outputs	Type
Trigger to connect to an attached Lynxmotion PS2 controller	Trigger	Whether the component is connected to a controller	Trigger
		Is the Triangle button pressed	Boolean
		Is the Circle button pressed	Boolean
		Is the Cross button pressed	Boolean
		Is the Square button pressed	Boolean
		Is the Lower Left trigger button pressed	Boolean
		Is the Lower Right trigger button pressed	Boolean
		Is the Upper Left trigger button pressed	Boolean
		Is the Upper Right trigger button pressed	Boolean
		Is the Select button pressed	Boolean
		Is the Start button pressed	Boolean
		Is Left Thumb stick pressed	Boolean
		Is Right Thumb stick pressed	Boolean
		Position of the D-pad as an angle in degrees clockwise from the Up position. -1 if nothing pressed.	Int
		Left Thumb Stick X (-1 to 1)	Float
		Left Thumb Stick Y (-1 to 1)	Float
		Right Thumb Stick X (-1 to 1)	Float
		Right Thumb Stick Y (-1 to 1)	Float

Ramp

Description

The Ramp primitive generates a non-bandlimited wave whose values increase linearly from 0 to 1 according to the normalised frequency input.

Connectors

Inputs	Type	Outputs	Type
Normalised frequency (0-1) with 1 meaning half sampling rate) of the wave to be generated.	Stream	The generated wave	Stream
Hard sync on transition from false to true	Stream Boolean		

Random Number

Description

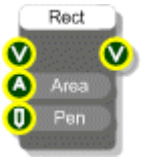
The Random Number primitive generates pseudo-random numbers. You can choose a fixed seed to reproduce a stream of numbers or leave the Seed input unconnected and the component will seed itself based on system time when the component is loaded or added to a schematic. A seed of 1 will reset the stream of numbers generated by the component.

Trigger the Get input to produce the next number in the current sequence.

Connectors

Inputs	Type	Outputs	Type
Trigger to produce the next number in the current sequence.	Trigger	The last generated random number	Int
Seed the random number sequence	Int		

Rectangle



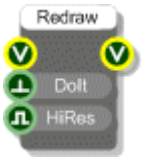
Description

Draws a rectangle on a View.

Connectors

Inputs	Type	Outputs	Type
View to display the rectangle on	View	The same View as the input, but anything connected here is drawn on top	View
Area defining the position and size of the rectangle	Area		
Pen defining the outline colour, thickness and style	Pen		

Redraw



Description

The Redraw component gives you low-level control over when redraws occur on a view. This particular version redraws everything on the view.

When inside an exported exe the Redraw component limits the redraw rate to 100Hz during automation for efficiency purposes. You can switch this off by setting the HiRes input to True.

Connectors

Inputs	Type	Outputs	Type
View to redraw	View	The same View as the input, but anything connected here is drawn on top	View
Trigger to do the redraw	Trigger		
Whether to enable instant redraws during automation in exported plugins	Boolean		

Redraw Area

Description

The Redraw Area component gives you low-level control over when redraws occur on a view. This particular version redraws everything inside a particular area of a view.

When inside an exported exe the Redraw component limits the redraw rate to 100Hz during automation for efficiency purposes. You can switch this off by setting the HiRes input to True.

Connectors

Inputs	Type	Outputs	Type
View to redraw	View	The same View as the input, but anything connected here is drawn on top	View
Area to redraw	Area		
Trigger to do the redraw	Trigger		
Whether to enable instant redraws during automation in exported plugins	Boolean		

Redraw Limiter

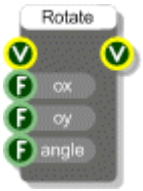
Description

The Redraw Limiter component is used to restrict the rate of flow of triggers through a schematic. It is primarily used for sections of schematic that are used for drawing whilst either interacting or automating. In such cases redraws do not need to occur at the same rate that values are updating so this component allows you to restrict data flow to something around 100Hz.

Connectors

Inputs	Type	Outputs	Type
Triggers at full rate	Trigger	Triggers at a rate no greater than 100Hz	Trigger

Rotate



Description

Applies a rotation transformation to the view. Any GUI components attached to this component are rotated according to the transformation.

Connectors

Inputs	Type	Outputs	Type
View to rotate	View	The same View as the input, but anything connected here is rotated according to the transformation	View
The x-coordinate of the centre point of the rotation	Float		
The x-coordinate of the centre point of the rotation	Float		
Rotation angle clockwise in degrees	Float		

Round Rectangle

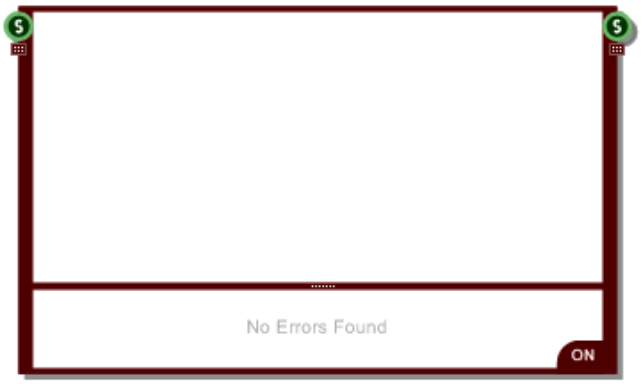
Description

Draws a round rectangle on a View, that is a rectangle with rounded corners.

Connectors

Inputs	Type	Outputs	Type
View to display the rectangle on	View	The same View as the input, but anything connected here is drawn on top	View
Area defining the position and size of the rectangle	Area		
Pen defining the outline colour, thickness and style	Pen		
Corner size in grid squares. A value of 1 will give you a corner that has an effective radius of 1 grid square	Float		

Ruby



Description

The Ruby component allows you to write standard Ruby code and use it in your schematic. The inputs and outputs can be changed interactively by dragging the grippers below the connectors and by right-clicking on the connectors themselves

There is a huge chapter dedicated to the Ruby component in the main user guide. All the information about it can be found there.

Connectors

Inputs	Type	Outputs	Type
User defined	User defined	User defined	User defined

Sample and Hold

Description

This component holds a value until the trigger input is hit at which point the current value is passed to the output.

The component has template connectors which means it can be used with multiple data types including Floats, Ints, Strings, Booleans, Float/Int/String arrays and Areas. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
Float value to sample and hold	Template	Value when the trigger was last fired	Template
Trigger to send the current value to the output	Trigger		

Sample Position

Description

The Sample Position primitive outputs the current songs sample position when your VST plugin is used within a host.

Connectors

Inputs	Type	Outputs	Type
N/A		Sample position	Stream

Sample Rate

Description

The Sample Rate primitive will give you the sample rate of the currently selected audio device either within FlowStone or when used in an exported exe.

This component is essential for ensuring that fixed time periods are translated into the correct number of samples for use with low-level components like delays and ADSR envelopes.

Connectors

Inputs	Type	Outputs	Type
N/A		Sample rate (in samples per second)	Float

Save Wave

Description

This component saves a memory buffer to a Wave file (.wav). You specify the path to the file, sampling rate and the sample format.

The sample format can be 16, 24 or 32 bit integer or 32 bit float. This is determined by a string supplied to the Format input. The string should be “int16”, “int24”, “int32” or “float32”.

Send a trigger to the Save input to execute the save.

Connectors

Inputs	Type	Outputs	
Memory buffer	Mem		
Trigger to execute the save	Trigger		
Full path to the file you want to save to	String		
Sampling rate (samples per second)	Int		
Sample format – one of int16, int24, int32 or float32	String		

Sawtooth

Description

The Sawtooth primitive produces the classic Sawtooth waveform. Its rich harmonic content makes it ideal for subtractive synthesis and this bandlimited form avoids aliased overtones. The phase input can be used for frequency modulation (phase modulation).

This Sawtooth HB primitive produces a Sawtooth wave with overtones limited to half the available bandwidth. Its inputs are the same as the basic Sawtooth and it is included primarily for educational purposes.

This Sawtooth DB produces a Sawtooth wave with overtones limited to twice the available bandwidth. It therefore produces aliasing. Its inputs are the same as the basic Sawtooth and it too is included primarily for educational purposes.

Connectors

Inputs	Type	Outputs	Type
Normalised frequency (0-1) with 1 meaning half sampling rate) of the wave to be generated.	Stream	The generated wave	Stream
Phase shift for the generated wave in the range 0-1 where 1 represents a shift of 2*pi	Stream		
Hard sync on transition from false to true	Stream Boolean		

Select

Description

The Select component will select between two inputs based on the value of the boolean input. The first input is selected if boolean input is False and the second input is selected if the boolean input is True.

The component has template connectors which means it can be used with multiple data types including Floats, Ints, Strings, Booleans, Float/Int/String arrays and Areas. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
First float value	Template	Value selected by the boolean input	Template
Second float value	Template		
Whether to pick the first input (False) or the second input (True)	Boolean		

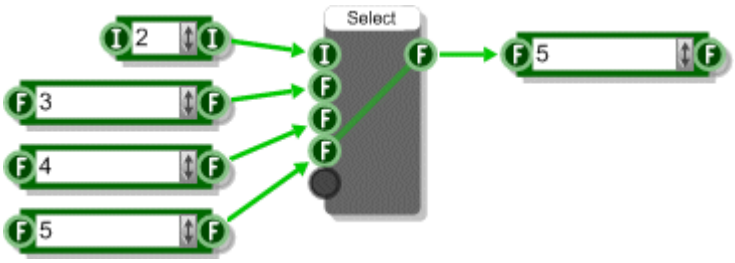
Selector

Description

The Selector component routes just one input from a set of inputs to a single output. The component has template connectors which means that you can use it with any connector type. The type is defined when you connect your first link to the component.

When you connect an input a new, unassigned input will appear below it. By continuously connecting unassigned inputs in this way you can build up to the number of inputs you require.

Selectors are useful for choosing between different options or for switching stream sections in and out of a schematic.



Connectors

Inputs	Type	Outputs	Type
Index of selected output (zero based)	Int	One single output	Any Type
Any number of other inputs	Any Type		

Set Pixel

Description

Sets the colour of a specified pixel in a bitmap.

Connectors

Inputs	Type	Outputs	Type
Bitmap to modify	Bitmap		
The x-coordinate of the pixel to be changed	Int		
The y-coordinate of the pixel to be changed	Int		
Colour to set the pixel to	Colour		
Trigger to set the pixel colour	Trigger		

Set Sample Rate

Description

Sets the sample rate for the currently selected audio device. You provide a rate and then send a trigger to the Set input in order to attempt the change. If the rate couldn't be set (usually because the audio device doesn't support it) then you'll get a trigger sent to the output.

Connectors

Inputs	Type	Outputs	Type
The sample rate you want to set (in samples per second)	Int	Trigger if the sample rate could not be set	Trigger
Trigger to attempt the change	Trigger		

SFZ

Description

The SFZ component parses an sfz format file creating a Bus from which you can read the parsed parameters. The sfz format was developed by rgc audio which is now owned by Cakewalk. It's a basic but very flexible concept using a simple text file to define a set of samples together with arrangement and performance parameters. (see http://www.cakewalk.com/devxchange/sfz.asp)

The SFZ component supports a large subset of the opcodes but not the full set. These are listed below. To read a parameter simply use a Bus Extract with the corresponding sfz opcode name(s).

group	pitch_keycenter	ampeg_delay	fileg_delay	fillfo_delay	loop_mode
sample	pitch_keytrack	ampeg_attack	fileg_attack	fillfo_freq	Loop_start
lochan	effect1	ampeg_hold	fileg_hold	fillfo_depth	loop_end
hichan	effect2	ampeg_decay	fileg_decay	amplfo_delay	volume
lokey	lovel	ampeg_sustain	fileg_sustain	amplfo_freq	pan
hikey	hivel	ampeg_release	fileg_release	amplfo_depth	cutoff
key	transpose		fileg_depth		resonance
	tune				

Connectors

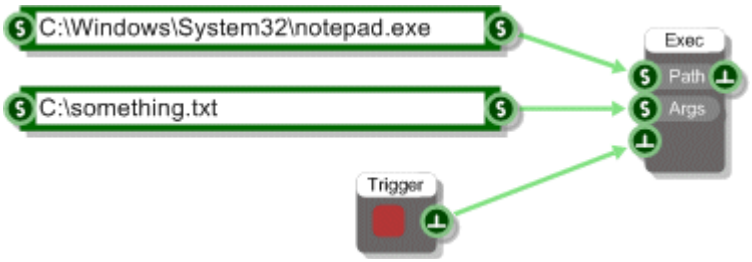
Inputs	Type	Outputs	Type
Path to the sfz file	String	Result: 0=success, 1=bad file, 2=parse failed	Int
Trigger to start parsing the file	Trigger	Bus containing the parsed data	Bus

Shell Execute

Description

The Shell Execute component will start an external program. You provide the path to the program executable and any command line arguments then send a trigger to launch the app.

For example:



Connectors

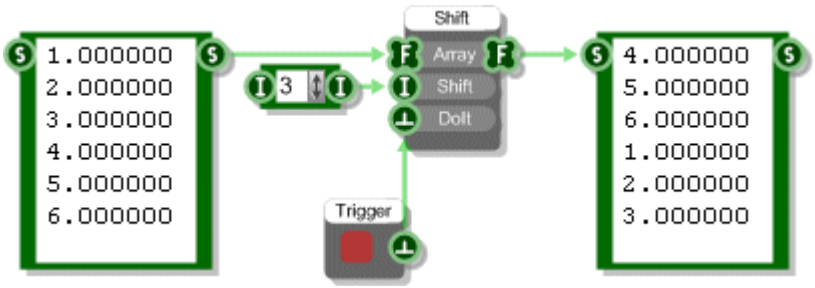
Inputs	Type	Outputs	
Full file path to the application you want to launch	String	Trigger when application has been launched	Trigger
Any command line arguments	String		
Trigger to launch the application	Trigger		

Shift Float Array

Description

This component will shift the elements in a float array round by a particular number of entries. Elements that get 'pushed off' the end of the array are wrapped round to the beginning of the array.

The example below shows how this works.



Connectors

Inputs	Type	Outputs	
The float array to be shifted	Float Array	The shifted array	Float Array
Number of elements to shift	Int		
Trigger to do the shifting	Trigger		

Show Cursor

Description

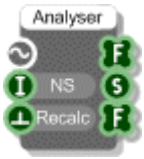
The Show Cursor component allows you to show or hide the mouse cursor. You can also choose whether to hold the mouse position when you hide the cursor and restore the mouse position when you show it again.

WARNING: use this component with care or you may find yourself without a cursor.

Connectors

Inputs	Type	Outputs	Type
Whether to show the cursor (so False will hide it)	Boolean	N/A	
Whether to maintain the mouse position while the cursor is hidden	Boolean		

Signal Analyser



Description

The Signal Analyser allows you to analyse the signal that would be produced by a section of Poly components. The Analyser simply runs the Poly code section for a defined number of samples. The output is an array containing the calculated samples. You can then plot this on a graph or use it for some other calculation.

The other outputs from the Signal Analyser are a String containing the compiled code for the attached Poly section and an array of the number of cpu cycles used for each calculated sample.

NOTE: You cannot connect an Analyser into a section of code that contains a Poly to Mono component.

Connectors

Inputs	Type	Outputs	Type
Section of Poly components to analyse	Poly	Array of calculated samples	Float Array
The number of samples to calculate	Int	The compiled code for the attached Poly section	String
Trigger to recalculate	Trigger	Array of cpu cycles used for each sample	Float Array

Sin

Description

Standard trigonometric Sine function with radians as the input units.

Connectors

Inputs	Type	Outputs	Type
Float value in radians	Float	Result of calculation	Float

Sin Inverse

Description

Standard trigonometric Inverse Sine function with radians as the output units.

Connectors

Inputs	Type	Outputs	Type
Float value	Float	Result of calculation in radians	Float

Sine

Description

The Sine primitive produces a sinusoid waveform. The phase input can be used for frequency modulation (phase modulation). It has no harmonic content of its own and is often used as modulation source.

Sinh

Description

Standard hyperbolic sine function with radians as the input units.

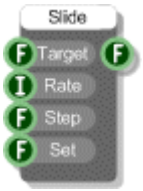
Connectors

Inputs	Type	Outputs	Type
Float value in radians	Float	Result of calculation	Float

Connectors

Inputs	Type	Outputs	Type
Normalised frequency (0-1) with 1 meaning half sampling rate) of the wave to be generated.	Stream	The generated wave	Stream
Phase shift for the generated wave in the range 0-1 where 1 represents a shift of 2*pi	Stream		
Hard sync on transition from false to true	Stream Boolean		

Slide



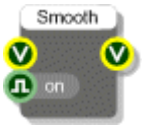
Description

The Slide component will slide from one float value to another at a particular rate. This can be useful for animation effects. The rate is specified as a time between steps in milliseconds. The component uses Windows timers so it can't be relied on for accuracy particularly below 15 milliseconds.

Connectors

Inputs	Type	Outputs	Type
The target value	Float	The current value	Float
The time between steps in milliseconds	Int		
The step value	Float		
For setting the current value	Float		

Smooth



Description

Changes the smooth mode for a View. By default smoothing (antialiasing) is applied to all GUI graphics. This component can be used to switch this smoothing off or back on again.

Smoothed Not Smoothed

Connectors

Inputs	Type	Outputs	Type
View to display the rectangle on	View	The same View as the input, but anything connected here is drawn on top	View
Whether to turn smoothing on or off	Boolean		

Sort Float Array

Description

The Sort Float Array primitive sorts the elements in the input array into ascending numerical order.

Connectors

Inputs	Type	Outputs	Type
Array to sort	Float Array	Sorted array	Float Array

Sort String Array

Description

The Sort String Array component sorts an array of strings into ascending alphabetical order.

Connectors

Inputs	Type	Outputs	
Source string array	String Array	The sorted array	String Array

Stream Add

Description

This component adds two Stream values together.

Connectors

Inputs	Type	Outputs	Type
Input signal 1	Stream	Sum of inputs	Stream
Input signal 2	Stream		

Stream Divide

Description

This component divides the first Stream value by the second one.

Connectors

Inputs	Type	Outputs	Type
Input signal	Stream	Division of inputs	Stream
Divisor signal	Stream		

Stream Greater Than

Description

This component compares the two inputs and returns a Stream Boolean based on whether the first input is greater than the second input.

Connectors

Inputs	Type	Outputs	Type
Input signal 1	Stream	Whether input signal 1 is greater than input signal 2	Stream Boolean
Input signal 2	Stream		

Stream Greater Than or Equal to

Description

This component compares the two inputs and returns a Stream Boolean based on whether the first input is greater than or equal to the second input.

Connectors

Inputs	Type	Outputs	Type
Input signal 1	Stream	Whether input signal 1 is greater than or equal to input signal 2	Stream Boolean
Input signal 2	Stream		

Stream Less Than

Description

This component compares the two inputs and returns a Stream Boolean based on whether the first input is less than the second input.

Connectors

Inputs	Type	Outputs	Type
Input signal 1	Stream	Whether input signal 1 is less than input signal 2	Stream Boolean
Input signal 2	Stream		

Stream Less Than or Equal to

Description

This component compares the two inputs and returns a Stream Boolean based on whether the first input is less than or equal to the second input.

Connectors

Inputs	Type	Outputs	Type
Input signal 1	Stream	Whether input signal 1 is less than or equal to input signal 2	Stream Boolean
Input signal 2	Stream		

Stream Max

Description

This component compares the two inputs and returns the greater of the two.

Connectors

Inputs	Type	Outputs	Type
Input signal 1	Stream	The greater of the two inputs	Stream
Input signal 2	Stream		

Stream Min

Description

This component compares the two inputs and returns the lower of the two.

Connectors

Inputs	Type	Outputs	Type
Input signal 1	Stream	The lower of the two inputs	Stream
Input signal 2	Stream		

Stream Multiply

Description

This component multiplies two Stream values together.

Connectors

Inputs	Type	Outputs	Type
Input signal 1	Stream	Product of inputs	Stream
Input signal 2	Stream		

Stream Subtract

Description

This component subtracts two Stream values from one another.

Connectors

Inputs	Type	Outputs	Type
Input signal 1	Stream	Input 1 minus Input 2	Stream
Input signal 2	Stream		

Stream to Double

Description

This component converts a standard Stream to a Double Stream.

Double Stream is similar to standard Float Stream except that it works at double the precision. It is particularly useful for counters.

Connectors

Inputs	Type	Outputs	Type
Standard Float Stream	Stream	Float Stream converted to Double Stream	Double Stream

String

Description

The String component allows you to enter and view character string data. To add a string just click on the main body of the component and type it in. Press Return, Tab or just click away to finish editing.

You can copy and paste data using the standard accelerator key combinations (CTRL+C,X and V).

The component can be resized horizontally for viewing long strings.

You can also change the type by right-clicking on the input or output. A pop-up menu will appear as shown below.

Simply click on the type you want to change to.

Connectors

Inputs	Type	Outputs	Type
Set the string	String	The current stored string	String

String Array

Description

The String Array component creates an array of strings by setting, inserting and deleting individual elements in the array. You define the index of the element you want to refer to and if needed the value you want to set or insert at that index. The array will resize automatically to accommodate elements set or inserted at indexes higher than the current size of the array.

Connectors

Inputs	Type	Outputs	
Value to use	String	Array of strings	String Array
The reference index used for set, insert and delete	Int	Trigger sent when the array has been changed	Trigger
Set the value at the index	Trigger	The number of array entries	Int
Clear the array	Trigger		
Array to make this equal to	String Array		
Insert the value at the index	Trigger		
Delete the entry at the index	Trigger		

String Array Find

Description

The String Array Find primitive will find the first occurrence of a given string in the array. You can choose whether comparisons are case sensitive. You can also choose whether to reverse the direction of the find and start at the end of the array working backwards towards the front. By default the find will start at the beginning of the array and work forwards.

Example

If the source array is Apple, Banana, Orange, Pear then finding “Orange” would return an index of 2. However, if you change the Rev input to True then the result would be 1.

Connectors

Inputs	Type	Outputs	Type
Source array to search through	String	Index of the string in the array (zero indexed) or -1 if the string was not found.	Int
String to find	String		
Whether comparisons should be case sensitive	Boolean		
Perform the find in reverse from beginning to start	Boolean		

String Array Get At

Description

The String Array Get At component extracts a particular entry from a String Array.

Connectors

Inputs	Type	Outputs	
The source array of text strings	String Array	The number at the given index	String
The index to get the value for	Int	Trigger sent when the string has been extracted	Trigger
Trigger to get the string	Trigger		

String Array Split

Description

The String Array Split primitive will break a given string array into two parts at a given point. Note that the split position starts at zero.

Connectors

Inputs	Type	Outputs	Type
Source string array	String Array	String array to the left of the split	String Array
Position at which to make the split (zero indexed)	Int	String array to the right of the split	String Array

String Array to String

Description

This component converts a String Array to a single String by appending the entries in the array to each other in order.

Example

The array “Apples”,”Oranges”,”Pears” would become “ApplesOrangesPears”.

Connectors

Inputs	Type	Outputs	Type
Array of strings	String Array	String of appended entries from the array	String

String Array to String

Description

This component converts a String Array to a single String by appending the entries in the array to each other in order and inserting the Sep input string in between entries to separate them.

Example

The array “Apples”,”Oranges”,”Pears” with Sep set to “,” would become “Apples,Oranges,Pears”.

Connectors

Inputs	Type	Outputs	Type
Array of strings	String Array	String of appended entries from the array	String
Separator to insert between entries	String		

String Extract

Description

The String Extract primitive will give you a sub-string of a particular size starting from a particular point in the string. Note that positions start at zero.

Example

If the string is ‘FlowStone’ then using a Pos of 4 and a Count of 5 you’d get ‘Stone’.

Connectors

Inputs	Type	Outputs	Type
Source string	String	Extracted sub-string	String
Position of the first character in the required sub-string (zero indexed)	Int		
Number of characters to extract	Int		

String Find

Description

The String Find primitive will find the first occurrence of a given sub-string inside a string. You can choose whether comparisons are case sensitive. You can also choose whether to reverse the direction of the find and start at the end of the string working backwards. By default the find will start at the beginning of the string and work forwards.

Example

If the source string is 'C:\Windows\Temp' then using a Find string of '\' the result would be 2. However, if you change the Rev input to True then the result would be 10.

Connectors

Inputs	Type	Outputs	Type
Source string	String	Position of the sub-string (zero indexed) or -1 if the sub-string was not found.	Int
Sub-string to find	String		
Whether comparisons should be case sensitive	Boolean		
Perform the find in reverse from beginning to start	Boolean		

String Length

Description

Calculates the length of a string by counting the number of characters. This includes all whitespace characters.

Example

If the source string is ‘Apples and Oranges’ then the length will be 18.

Connectors

Inputs	Type	Outputs	Type
Source string	String	Number of characters in the string.	Int

String Queue

Description

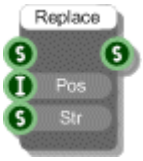
The String Queue component stores string values in a queue. Values are pushed in and popped out on a first in, first out basis (FIFO).

You can get the queue in String Array form from the third output. The array contains items in the order they would be popped out – so the first item is at the front of the queue, the second item is next and so on.

Connectors

Inputs	Type	Outputs	
Next number to be pushed onto the queue	Float	The string at the front of the queue	String
Trigger to push the next number onto the queue	Trigger	Number of entries in the queue	Int
Trigger to pop the next number off the queue	Trigger	The queue as a string array	String Array
Trigger to clear all entries from the queue	Trigger		

String Replace



Description

The String Replace primitive will replace the characters at a particular position in a given string with another sub-string. Note that positions start at zero.

Example

If the source string is ‘Apples and Oranges’ then using a Pos of 10 and sub-string of ‘Bananas’ you’d get ‘Apples and Bananas’.

Connectors

Inputs	Type	Outputs	Type
Source string	String	Modified string	String
Position of the first character to be replaced (zero indexed)	Int		
Replacement sub-string	Int		

String Split

Description

The String Split primitive will break a given string into two parts at a given point. Note that the split position starts at zero.

Example

If the source string is ‘FlowStone’ then using a Pos of 4 you’d get ‘Flow’ and ‘Stone’.

Connectors

Inputs	Type	Outputs	Type
Source string	String	String to the left of the split	String
Position at which to make the split (zero indexed)	Int	String to the right of the split	String

String Stack

Description

The String Stack component stores string values in a stack. Values are pushed in and popped out on a last in, first out basis (LIFO).

You can get the stack in String Array form from the third output. The array contains items in the order they would be popped out – so the first item is at the top of the stack, the second item is next and so on.

Connectors

Inputs	Type	Outputs	
Next number to be pushed onto the stack	Float	The number at the top of the stack	String
Trigger to push the next number onto the stack	Trigger	Number of entries in the stack	Int
Trigger to pop the next number off the stack	Trigger	Stack represented as an array of strings	String Array
Trigger to clear all entries from the stack	Trigger		

String to ASCII

Description

The String to ASCII primitive converts a string character to its corresponding ASCII character code (or more accurately the ISO Latin 1 character as ASCII is only defined through 127).

Connectors

Inputs	Type	Outputs	Type
String character	String	ASCII character code (0-255)	Int

String to Hex

Description

Converts a string of characters to a string of hex. Each character is first converted to Ascii and then the hex representation of that byte is used in the hex string.

For example, the character string “hello” is converted to the hex string “68656C6C6F”.

Connectors

Inputs	Type	Outputs	Type
String of characters	String	String of hex	String

String to String Array

Description

This component converts a String to an Array of Strings by splitting the string up character by character and inserting the characters into the array.

Example

The String “Apples” would become a String Array with entries: “A”,”p”,”p”,”l”,”e”,”s”.

Connectors

Inputs	Type	Outputs	Type
String to split	String	Array of characters	String Array

String to String Array

Description

This component converts a String to an Array of Strings by splitting the string up using the supplied delimiter or field width or both to determine where to make a split.

Examples

The String “Apples,Oranges,Pears” with a delimiter of “,“ would become a String Array with entries: “Apples”, “Oranges” and “Pears”.

The String “1011011101011110” with a Width of 4 would become a String Array with entries: “1011”,”0111”,”0101”,”1110”

Connectors

Inputs	Type	Outputs	Type
String to split	String	Array of individual strings	String Array
Delimiter to use when looking where to split the string	String		
Maximum length of each string in the resulting array	Int		

String to Sysex

Description

This component takes a string of hex and sends it out as MIDI System Exclusive data. Useful for controlling external MIDI hardware.

String Format

Description

Creates a String Format for use with a Text component. A String Format combines a text style, horizontal alignment and vertical alignment.

The style can be any combination of the following:

```
normal, righttoleft, nowrap, vertical
```

For example:

```
nowrapvertical, nowraprighttoleft.
```

The `righttoleft` style specifies that the reading order is right to left, `nowrap` prevents text from wrapping when it runs outside the defined area and `vertical` will cause text to be drawn vertically instead of horizontally.

For horizontal alignment you can use:

```
left, center or right
```

For vertical alignment you can use

```
top, center or bottom
```

Connectors

Inputs	Type	Outputs	Type
Formatting style	String	The String Format	String Format
Horizontal alignment	String		
Vertical alignment	String		

Connectors

Inputs	Type	Outputs	Type
String of hex	String	Sysex data	MIDI
Trigger to send the hex	Trigger		

Subtract

Description

This component subtracts two values from one another.

The component has template connectors which means it can be used with multiple data types including Floats, Ints and Float/Int arrays. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
Input 1	Template	Input 1 minus input 2	Template
Input 2	Template		

Subtract from Float Array

Description

Subtracts the same float value from every entry in a float array.

Connectors

Inputs	Type	Outputs	Type
Array to subtract from	Float Array	Resulting array	Float Array
Float value to be subtracted	Float		

Sum Float Array

Description

The Sum Float Array primitive adds all the elements in the input array together and outputs the result.

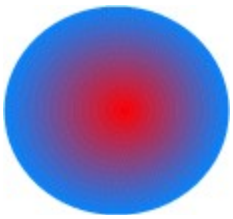
Connectors

Inputs	Type	Outputs	Type
Array to sum	Float Array	Sum of all entries	Float Array

Sunburst Gradient

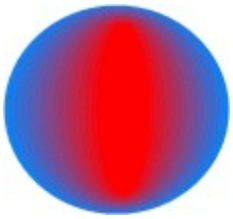
Description

The Sunburst Gradient component draws either an ellipse or a rectangle with a sunburst fill effect. This effect creates a seamless transition through increasing concentric circles from a centre to an outer colour. This component useful for creating lighting effects.



The gradient is defined by a bounding area inside which the fill can be rectangular or ellipsoid. You specify a centre colour and a colour at the boundary and you can set the coordinates of the centre point as well.

The H focus and V focus inputs determine the focus scale. These are float values in the range (0-1) and determine at which point the centre colour starts to blend into the boundary colour. A focus scale value of zero will begin transition immediately. A focus scale of 0.5 will have the centre colour for half the range of the fill before blending to the boundary colour for the other half.



The above picture shows a sunburst fill with H Focus of 0.2 and a V Focus of 0.8.

Connectors

Inputs	Type	Outputs	Type
View to draw on	View	The same View as the input, but anything connected here is drawn on top	View
Area defining the bounding area of the ellipse or rectangle	Area		
Either “Rectangle” or “Ellipse”	String		
First colour in the gradient	Colour		
Second colour in the gradient	Colour		
The x-coordinate of the centre point of the fill in grid squares	Float		
The y-coordinate of the centre point of the fill in grid squares	Float		
Horizontal focus scale (0-1)	Float		
vertical focus scale (0-1)	Float		

Sysex to String

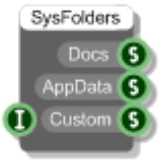
Description

This component receives MIDI System Exclusive data and outputs it as a string of hex. Useful for processing data from external MIDI hardware.

Connectors

Inputs	Type	Outputs	Type
Sysex data	MIDI	String of hex	String
		Trigger when data is received	Trigger

System Folders



Description

This component gives you the full paths to the Document and Application Data folders on the host PC. This is needed if you want to store local settings.

You can also get the values of other system specific folder paths from the Custom output by supplying the appropriate id to the Int input.

Connectors

Inputs	Type	Outputs	Type
		Path to Documents folder	String
		Path to Application Data folder	String
Required custom folder id	Int	Folder path defined by the Int input	String

The custom folder id's are as follows:

Id and Description	Default folder on Windows 7
-1 = FlowStone Install Folder	C:\Program Files\DSP Robotics\FlowStone
0 = Desktop	C:\Users\[username]\Desktop
2 = Start Menu Programs	C:\Users\[username]\AppData\Roaming\Microsoft\Windows\Start Menu\Programs
5 = Documents	C:\Users\[username]\Documents
6 = Favourites	C:\Users\[username]\Favorites
7 = Startup	C:\Users\[username]\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
8 = Recent	C:\Users\[username]\AppData\Roaming\Microsoft\Windows\Recent
9 = Send To	C:\Users\[username]\AppData\Roaming\Microsoft\Windows\SendTo 1
1 = Start Menu	C:\Users\[username]\AppData\Roaming\Microsoft\Windows\Start Menu
13 = Music	C:\Users\[username]\Music
14 = Videos	C:\Users\[username]\Videos
19 = Network Shortcuts	C:\Users\[username]\AppData\Roaming\Microsoft\Windows\Network Shortcuts
20 = Fonts	C:\Windows\Fonts
21 = Templates	C:\Users\[username]\AppData\Roaming\Microsoft\Windows\Templates
26 = AppData Roaming	C:\Users\[username]\AppData\Roaming
27 = Printer Shortcuts	C:\Users\[username]\AppData\Roaming\Microsoft\Windows\Printer Shortcuts
28 = AppData Local	C:\Users\[username]\AppData\Local
32 = Temporary Internet Files	C:\Users\[username]\AppData\Local\Microsoft\Windows\Temporary Internet Files
33 = Cookies	C:\Users\[username]\AppData\Roaming\Microsoft\Windows\Cookies

Id and Description	Default folder on Windows 7
34 = History	C:\Users\[username]\AppData\Local\Microsoft\Windows\History
35 = Program Data	C:\ProgramData
36 = Windows	C:\Windows
37 = System32	C:\Windows\system32
38 = Program Files	C:\Program Files
39 = Pictures	C:\Users\[username]\Pictures
40 = User	C:\Users\[username]
43 = Common Files	C:\Program Files\Common Files
45 = Windows Templates	C:\ProgramData\Microsoft\Windows\Templates
46 = Public Documents	C:\Users\Public\Documents
47 = Program Data Administrative Tools	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Administrative Tools
48 = AppData Administrative Tools	C:\Users\[username]\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Administrative Tools

System Fonts

Description

The System Fonts component gives you a list of all the fonts installed on the host system. The font names are provided as a string array and as a delimited string that you can look at using a Text component.

Connectors

Inputs	Type	Outputs	
Trigger to get the font list	Trigger	Array of font names	String Array
		Delimited string containing the font names	String
		Number of fonts found	Int

Tan

Description

Standard trigonometric Tangent function with radians as the input units.

Connectors

Inputs	Type	Outputs	Type
Float value in radians	Float	Result of calculation	Float

Tan Inverse

Description

Standard trigonometric Inverse Tangent function with radians as the output units.

Connectors

Inputs	Type	Outputs	Type
Float value	Float	Result of calculation in radians	Float

Tanh

Description

Standard hyperbolic tangent function with radians as the input units.

Connectors

Inputs	Type	Outputs	Type
Float value in radians	Float	Result of calculation	Float

Tempo

Description

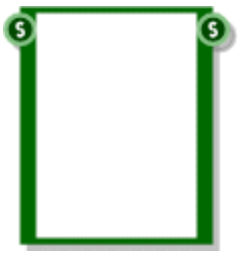
When your VST is used within a host these components will tell you the current host tempo in beats per minute (BPM). This is most often used when creating tempo synchronised effects.

There are two versions of the Tempo primitive a Stream version for use in Poly or Mono sections and a Float version for use in green data sections.

Connectors

Inputs	Type	Outputs	Type
N/A		Current tempo in BPM	Stream/Float

Text



Description

The Text component allows you to enter and view character string data that stretches over several lines. To add a string just click on the main body of the component and type it in. Press Return to start a new line. Press Tab or just click away to finish editing.

You can copy and paste data using the standard accelerator key combinations (CTRL+C,X and V). You can also use the mouse wheel, PGUP, PGDN, HOME and END keys to navigate text that spans many lines.

The Text component can be used for generating fixed arrays of Floats or Ints as a list of numbers separated by line breaks can be automatically converted to a Float or Int Array (and vice-versa).

The component can be resized horizontally and vertically for viewing larger amounts of text.

Connectors

Inputs	Type	Outputs	Type
Set the string	String	The current stored string	String

Text Draw

Description

Draws text on a view. You specify the area into which the text will be drawn together with any formatting options.

Connectors

Inputs	Type	Outputs	Type
View to display the text on	View	The same View as the input, but anything connected here is drawn on top	View
Area into which the text is to be drawn	Area		
The text colour	Colour		
The text to be displayed	String		
The font for the text	Font		
Text formatting options	String Format		

Text Load

Description

This component allows you to load text from a text file.

Connectors

Inputs	Type	Outputs	
Full path to the text file on your hard disk	String	The contents of the text file	String
Trigger to load the file	Trigger	Trigger when the file has loaded	Trigger

Text Save

Description

This component allows you to save text to a text file.

Connectors

Inputs	Type	Outputs	
Text to be saved	String	Trigger when the file has been saved	Trigger
Full path to the text file on your hard disk	String		
Trigger to save the file	Trigger		

Ticker 100

Description

The Ticker 100 component sends a trigger roughly 100 times per second. The component uses a Windows timer and therefore cannot be relied upon to be accurate. This component is useful for animation or for restricting data flow through a schematic.

Connectors

Inputs	Type	Outputs	
N/A		Trigger sent 100 times per second	Trigger

Ticker 25

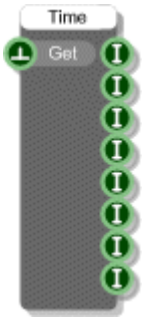
Description

The Ticker 25 component sends a trigger roughly 25 times per second. The component uses a Windows timer and therefore cannot be relied upon to be accurate. This component is useful for animation or for restricting data flow through a schematic.

Connectors

Inputs	Type	Outputs	
N/A		Trigger sent 25 times per second	Trigger

Time



Description

The Time primitive will give you any or all of the components of the current system time. The time is sampled when you trigger the Get input and this time is stored in the component.

Connectors

Inputs	Type	Outputs	Type
Get the current system time (and store it)	Trigger	Weekday. 0= Monday, 1=Tuesday etc.	Int
		Day of the month	Int
		Month. 0=January, 1=February etc.	Int
		Year	Int
		Hour	Int
		Minute	Int
		Second	Int
		Millisecond	Int

Timer

Description

The Timer component allows you to schedule a trigger to be sent some time in the future. You specify a time duration and then start the timer. After the duration has elapsed a trigger will be sent through the output. The timer will automatically schedule another trigger so that the triggers keep coming every N milliseconds (where N is the duration).

For a one single shot timer just connect the trigger output to the Stop input.

The component uses a Windows timer and therefore cannot be relied upon to be accurate, particularly for durations below 15 ms.

IMPORTANT: because of multi-threading you cannot trigger timers in response to MIDI events. This can cause unexpected behaviour or even crash the software.

Connectors

Inputs	Type	Outputs	
Duration between triggers in milliseconds	Int	Trigger sent every time a duration's worth of time passes	Trigger
Trigger to start the timer	Trigger		
Trigger to stop the timer	Trigger		

Time Signature

Description

The Time Signature primitive outputs the current songs time signature when your VST plugin is used within a host. This is represented by a numerator and denominator (for example, numerator=1 & denominator=4 for time signature of 1/4). For more details see the VST SDK documentation.

Connectors

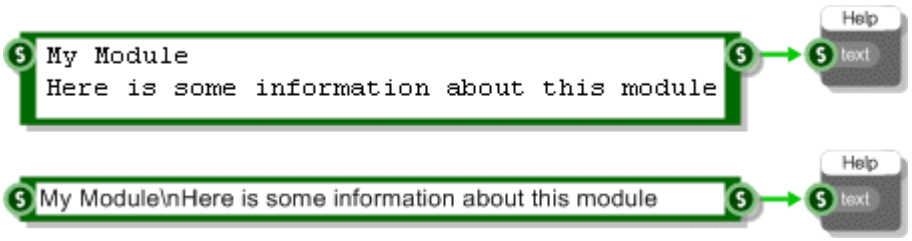
Inputs	Type	Outputs	Type
N/A		Numerator	Stream / Float
		Denominator	Stream / Float

Tooltip Help

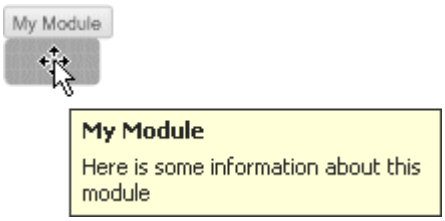
Description

The Tooltip Help component allows you to add tooltip help to a module. The help text should be connected to the String input. Paragraphs can be identified using the new line character '\n' if using a String component or by starting a new line in a Text component. The first line is assumed to be the name and is formatted in bold when the tool tip displays.

For example, either of the follow arrangements inside 'My Module':



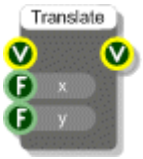
will produce the result shown below:



Connectors

Inputs	Type	Outputs	Type
Tooltip title and text	String	N/A	

Translate



Description

Applies a translation transformation to the view. Any GUI components attached to this component are translated accordingly.

Connectors

Inputs	Type	Outputs	Type
View to translate	View	The same View as the input, but anything connected here is translated according to the transformation	View
The translation in the x (horizontal) direction in grid squares	Float		
The translation in the y (vertical) direction in grid squares	Float		

Triangle

Description

The Triangle primitive produces a Triangle waveform. The phase input can be used for frequency modulation (phase modulation).

Connectors

Inputs	Type	Outputs	Type
Normalised frequency (0-1) with 1 meaning half sampling rate) of the wave to be generated.	Stream	The generated wave	Stream
Phase shift for the generated wave in the range 0-1 where 1 represents a shift of 2*pi	Stream		
Hard sync on transition from false to true	Stream Boolean		

Trigger Blocker

Description

The Trigger Blocker components prevents a change to a value from propagating through a schematic. Instead, changes at the input are stored until requested.

This can be useful when values are changing quicker than they are required further down in the schematic. By inserting a trigger blocker you can increase performance by saving the message flow into parts of your schematic where the updates are not required.

The component has template connectors which means it can be used with multiple data types including Floats, Ints, Strings, Booleans, Float/Int/String arrays and Areas. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
Value	Template	Current value but no trigger is sent	Template

Trigger Button

Description

The Trigger Button component sends a trigger when its red button is clicked. This component is incredibly useful and comes in handy in many, many situations.

You can get a Trigger Button using the “Q” shortcut key.

Connectors

Inputs	Type	Outputs	Type
N/A	N/A	Trigger sent when button clicked	Trigger

Trigger Div

Description

This primitive is used to reduce the rate that triggers pass through. The second input (the divisor) defines the reduction. If the divisor is one then all triggers pass through. If the divisor is two then only every other trigger passes through. If the divisor is three then only every third trigger passes through and so on.

Connectors

Inputs	Type	Outputs	Type
Trigger source	Trigger	Trigger output	Trigger
Divisor	Int		

Trigger Switch

Description

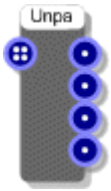
The trigger switch is used to control the flow of triggers through a schematic. When the boolean input is True then triggers pass through otherwise they are blocked.

The component has template connectors which means it can be used with multiple data types including Floats, Ints, Strings, Booleans, Float/Int/String arrays, Triggers and Areas. You can right-click on the connectors at any time to change the type.

Connectors

Inputs	Type	Outputs	Type
Trigger	Template	Trigger passing through	Template
Whether to let the trigger through	Boolean		

Unpack



Description

The Unpack component separates the original 4 Mono streams from a Mono4 stream. The Mono streams would have previously been packed together using the Pack component to create a Mono4 stream.

Using Unpack (and Pack) can radically increase the efficiency of a Mono section.

Connectors

Inputs	Type	Outputs	Type
First mono stream	Mono	All 4 mono streams ‘packed’ into one stream	Mono4
Second mono stream	Mono		
Third mono stream	Mono		
Fourth mono stream	Mono		

Video Delay

Description

The Video Delay component will delay the output of a video stream by a fixed number of frames.

All you need is a stream of video images at the first input and an integer number of frames to delay by at the second input.

Connectors

Inputs	Type	Outputs	Type
Video as a set of streaming bitmap images	Bitmap	Video as a set of streaming bitmap images	Bitmap
Number of frames to delay by	Int		

Video Save

Description

The Video Save component allows you to save to a .AVI movie file.

Connect your source video images to the first input. These will be recorded as they play.

At the Path input you need to supply the full path to the target .avi file that you want to save to.

To specify the codec attach a string with the appropriate 4 character code to the Codec input. Alternatively you can connect a String with -1 in it and the software will ask you to pick a codec from the standard dialog box.

The Rate input should be the frame rate in frames per second. If you leave this out then this will assume the default value of 30 fps.

Recording is started and stopped by triggering the bottom two inputs.

Connectors

Inputs	Type	Outputs	Type
Video as a set of streaming bitmap images	Bitmap	Whether recording is in progress or not	Boolean
Path to the .avi file you want to save to	String		
4 char codec code or -1	String		
Frame rate in frames per second	Int		
Start saving	Trigger		
Stop saving	Trigger		

Video Stream

Description

This component allows you to stream video from media file locally or across a network.

You must provide a valid URL or file path. Width and Height are optional and default to 640 x 480 pixels.

To begin streaming, trigger the Start input. The video arrives as constant stream of bitmaps. If you want to control when a frame comes out of the component you can set the Manual input to true and use the Grab trigger to get a frame exactly when you want it. This is useful if you don't want every single frame but maybe need them at a particular time interval or on demand.

You can Step through the video frame by frame by triggering the Step input. Note that this is not the same as manually grabbing a frame. When stepping the streaming is paused between steps whereas when grabbing frames streaming continues between grabs.

To resume play after stepping trigger the Pause/Play input. This can also be used to pause playback.

Triggering the Stop input will end streaming and you can only resume by re-starting from the beginning.

Most popular image formats are supported including AVI, MPEG, MP4, M4V, WMV, MOV.

Connectors

Inputs	Type	Outputs	Type
URL or file path for the source video	String	Video as a set of streaming bitmap images	Bitmap
Width of source video in pixels	Int		
Height of source video in pixels	Int		
Start streaming	Trigger		
Step through one frame at a time	Trigger		
Pause or resume play	Trigger		
Stop streaming	Trigger		
Use manual mode	Boolean		
Grab a frame (manual mode only)	Trigger		

View Area

Description

The View Area component will give you the dimensions of a View in the form of an area. This is useful if you're trying to draw something exactly within the bounds of a view or if you want to position elements relative to the size of the view.

Connectors

Inputs	Type	Outputs	Type
Source view	View	Area defining the size of the view	Area

View Size

Description

The View Area component will give you the dimensions of a View. This is useful if you're trying to draw something exactly within the bounds of a view or if you want to position elements relative to the size of the view.

Connectors

Inputs	Type	Outputs	Type
Source view	View	The width of the view in grid squares	Float
		The height of the view in grid squares	Float

Voices to Poly

Description

The Voices to Poly module generates voice managed Poly signals used to control a polyphonic synthesizer.

Note: Currently there is no reason to use this component on its own, instead you should use the MIDI to Poly module which combines the MIDI to Voices and Voices to Poly components.

Connectors

Inputs	Type	Outputs	Type
MIDI data	MIDI	The normalised frequency for each active voice in range (0-1) where 1 is half sampling rate	Poly
		The pitch number (0-127) for the note on each active voice	Poly
		The velocity of the note for each active voice	Poly
		A gate signal which is true if the note is on and false if the note is off for each active voice	Poly Boolean

VST Editor Open

Description

The Editor Open primitive will tell you whether the plugin editor window is open in a host. This is very useful for bypassing calculations that result in visual changes which will not be visible when the editor window is closed.

When the state of the window changes from open to closed or closed to open this component will send a trigger immediately so that you can respond to the change.

Connectors

Inputs	Type	Outputs	Type
N/A		Whether the editor window is open	Boolean

VST Parameter

Description

The VST Parameter primitive defines a preset parameter that can be automated or store preset data for each program. All the built in knob modules each contain a single VST Parameter component so that the knobs can be automated and store preset data in exported exes.

When a VST Parameter component is connected to a Preset Manager component it becomes part of the preset system. This connection is made through the Preset input connector. When the Preset Manager changes program the VST Parameter component will respond accordingly.

The VST preset parameter system requires values to be floats in the range 0 to 1 . You must make sure that you transform any values you want to store into this range before passing them to the VST Parameter component.

If you want the host to display something other than the 0-1 float value you can create a section of schematic that processes the float output of the VST Parameter component and connect it to the Disp input.

Examples

1. You could have switch that you want to show On for 1.0 and Off for 0.

2. You might have a multiplier that you want to show as Level 1, Level 2 or Level 3. You'd convert the input value to 0-1 by calculating (value-1)/2. This would give you 0,0.5 and 1. When the preset changes you'd process the result back to a number in the range 1-3, append it onto "Level " and pass the result to the Disp input.

You can choose whether the parameter is automatable and also whether it is a preset. For example, you might want to be able to automate a master volume control but not have it store a different value for each program.

Connectors

Inputs	Type	Outputs	Type
Connector for communicating with Preset Manager or Preset Text File components	Preset	The value of the parameter for the current program	Float
Set the current value (must be in range 0-1)	Float	Trigger sent just before a value change occurs	Trigger
Name of the preset parameter (as it will appear in the host)	String	Trigger sent after a value change occurs	Trigger
Optional display string to be used instead of the parameter value.	String		
Optional units string	String		
Whether the parameter can be automated	Boolean		
Whether the parameter is a preset	Boolean		
The default value for when the number of presets is increased	Float		
Whether to send automation recording information to the host when the value changes	Boolean		

VST Parameter Array

Description

The VST Parameter Array primitive defines an array of VST parameters that can be automated or store preset data for each program. This is a specialised component used for storing preset data for step sequencers and the like.

When a VST Parameter Array component is connected to a Preset Manager component it becomes part of the preset system. This connection is made through the Preset input connector. When the Preset Manager changes program the VST Parameter Array component will respond accordingly.

The VST preset parameter system requires values to be floats in the range 0 to 1 . You must make sure that you transform any values you want to store into this range before passing them to the VST Parameter Array component.

If you want the host to display something other than the 0-1 float value you can create a section of schematic that processes the float output of the VST Parameter component and connect it to the Disp input.

Examples

1. You could have switch that you want to show On for 1.0 and Off for 0.
2. You might have a multiplier that you want to show as Level 1, Level 2 or Level 3. You'd convert the input value to 0-1 by calculating (value-1)/2. This would give you 0,0.5 and 1. When the preset changes you'd process the result back to a number in the range 1-3, append it onto "Level " and pass the result to the Disp input.

You can choose whether the parameters in the array are automatable and also whether the array is a preset. If the array data applies globally and is large in size for example you may not want to maintain copies for each program so switching the Preset option off would save on memory.

Connectors

Inputs	Type	Outputs	Type
Connector for communicating with Preset Manager or Preset Text File	Preset	The array for the current program	Float Array
Number of entries in the array	Int	The value of the parameter for current program & index	Float
The current array index	Int	The current array index	Int
Set the current value for the above array index (must be in range 0-1)	Float	Trigger sent just before a value change occurs	Trigger
Name of the preset parameter (for host)	String	Trigger sent after a value change occurs	Trigger
Optional display string to be used instead of the parameter value.	String		
Optional units string	String		
Whether the parameter can be automated	Boolean		
Whether the parameter is a preset	Boolean		
The default value for when the number of presets is increased	Float		
Set the whole array for the current program	Float Array		
Trigger to reset the array for the current program to the default value for each entry	Trigger		
Whether to send automation recording information to the host when the value changes	Boolean		

VST Plugin Info

Description

The VST Plugin Info primitive allows you to provide default VST plugin information for a module. This is optional but is helpful if you want to use the same information every time you export a plugin. Any data not provided will use the global setting.

In order to use the component just drop it somewhere inside your VST plugin module.

Connectors

Inputs	Type	Outputs	Type
Type of plugin, 0 = effect, 1= instrument	Int	N/A	
Plugin name	String		
Vendor name	String		
Version number	Int		
Path to the folder where the exported plugin DLL should go			
Four character plugin id	String		
Whether to save support for all SSE types (improves loading speed but increases export time)	Boolean		

VST Preset String

Description

The VST Preset String primitive defines a VST parameter string that can be stored for each program. This works in a very similar way to the VST Parameter component except that this is string data not a number and so it can't be automated.

When a VST Preset String component is connected to a Preset Manager component it becomes part of the preset system. This connection is made through the Preset input connector. When the Preset Manager changes program the VST Preset String component will respond accordingly.

VST Preset Strings can be used for storing file paths or filenames for loading a particular sample but they could be used to store any other kinds of data, really anything that can be represented as a string.

Connectors

Inputs	Type	Outputs	Type
Connector for communicating with Preset Manager or Preset Text File components	Preset	The string for the current program	String
Set the string for the current program	String	Trigger sent just before a value change occurs	Trigger
Name of the parameter (must be unique)	String	Trigger sent after a value change occurs	Trigger
Default string for when the number of presets changes	String		

Wave Array Read

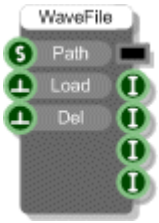
Description

This component will read data from a wave array at sample rate. There are two indexes, one to select the wave from the array and one for the current sample(s) within the selected wave. The outputs are the current samples for the left and right channels in the selected wave. If the selected wave is mono both outputs will give the same value.

Connectors

Inputs	Type	Outputs	Type
Array of waves	Wave Array	Left channel data	Stream
Index of the required wave in the array	Stream	Right channel data	Stream
Index of the required sample in the selected wave	Stream		

Wave File



Description

The Wave File component loads a wave file from your hard disk. You supply the complete path to the file then trigger the Load input. The wave file is loaded and stored. You can access the data using a Wave Read component.

The very last output on the component indicates the current state of the component using a flag. The possible flags are as follows:

0	Empty	The wave buffer is currently empty
1	Loading	The component is loading a file
2	Loaded	The component has loaded a file successfully
3	Bad File	The file does not exist
4	Bad Format	The format of the file is incorrect

Connectors

Inputs	Type	Outputs	Type
Full path to wave file	String	Wave file data in a buffer	Mem
Trigger to load the file	Trigger	Sampling rate in samples per second	Int
Trigger to delete the wave data	Trigger	Number of channels	Int
		Number of samples in each channel	Int
		Flag indicating the result of loading (see above)	Int

Wave File Array

Description

The Wave File Array component allows you to construct a wave array by loading in individual wave files one at a time. You specify a size for the array first and trigger the Resize input to create it.

To load in a file you need to specify an array index to say where it will go and then the complete path to the file. The Load trigger will then load the file.

By default all waves are stored in stereo format. If you are only using mono waves then you can set the Mono input to True and save some memory. If you set the Mono input when you have stereo wave files loaded they will be converted permanently to mono.

Connectors

Inputs	Type	Outputs	Type
Required size of the array	Int	Array of waves	Wave Array
Trigger to resize the array. Note that if you make the wave smaller it will be irreversibly truncated	Trigger	Array containing the sample rates for each wave in the array	Int Array
Trigger to clear the wave at the specified index	Trigger	Array containing the number of samples per channel for each wave in the array	Int Array
Index of the wave to be loaded or deleted	Int	The number of channels (2 for stereo waves, 1 for mono).	Int
Full path to wave file	String		
Trigger to load the file	Trigger		
Whether to store waves as mono files	Boolean		
Clear the array	Trigger		
Whether to ignore the wave files in the array when saving	Boolean		

Wave Read

Description

The Wave Read component will read data from a memory buffer at sample rate. The index input specifies which sample to read. For stereo buffers the samples for the left and right channels at the given index are sent to the two outputs. For mono buffers the same sample is sent to both outputs.

Connectors

Inputs	Type	Outputs	Type
Memory buffer containing the wave data	Mem	Left channel data	Stream
Index of the required sample in the wave	Stream	Right channel data	Stream

Wave Read Hop

Description

The Wave Read Hop component will read data from a memory buffer at sample rate. The index input specifies which sample to read. For stereo buffers the samples for the left and right channels at the given index are sent to the two outputs. For mono buffers the same sample is sent to both outputs.

This component is identical to the Wave Read except that, for efficiency, it allows you to provide a Hop. This saves cpu by only performing the lookup every 'Hop' number of samples.

The Hop is specified as a power of 2 so 1=lookup every sample, 2=lookup every 4 samples, 3= every 8 samples and so on.

Connectors

Inputs	Type	Outputs	Type
Memory buffer containing the wave data	Mem	Left channel data	Stream
Index of the required sample in the wave	Stream	Right channel data	Stream
Hop value as power of 2	Int		

Wave Table

Description

The Wave Table component will take any 256 sample wave and create a bandlimited wavetable. This allows you to define your own waveform and turn it into a playable sound source which does not alias. The component uses FFT analysis to remove unwanted frequencies from the wave and creates 256 bandlimited tables which can be read using the Wave Table Read component.

Connectors

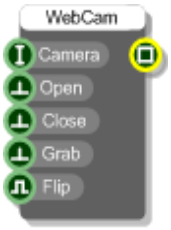
Inputs	Type	Outputs	Type
Array of 256 samples defining the waveform	Float Array	256*256 sample bandlimited wave table	Mem
Trigger to create the wave table	Trigger	Trigger when the wave table has been calculated	Trigger

Wave Table Read

Description

The Wave Table Read component is used to read wave data created by the Wave Table component. It works very much like the low-level oscillator components. You supply a frequency (in range 0-1, where 1 is nyquist) and a wave table and the component will read the data from the appropriate section of the table.

Web Cam



Description

This component allows you to capture an image from the default connected web cam.

Connectors

Inputs	Type	Outputs	Type
Id of the camera you want to use. Leave this blank or zero to use the default camera	Trigger	The captured image	Bitmap
Trigger to Open the web cam	Trigger		
Trigger to Close the web cam	Trigger		
Trigger to grab a frame from the web cam	Trigger		
Whether to flip the image horizontally to create a mirror image	Boolean		

Web URL

Description

The Web URL component allows you to open a web link by opening a web browser.

Connectors

Inputs	Type	Outputs	Type
The required web URL	String	N/A	
Trigger to launch the web link	Trigger		

Wii Nunchuck

Description

The Wii Nunchuk component allows you to receive input data from a Nunchuk controller attached to a Nintendo Wiimote.

The Wiimote input must be connected to the Wiimote output of a Wiimote component in order to work. Obviously that Wiimote component must also be associated with the relevant hardware.

Connectors

Inputs	Type	Outputs	Type
Wiimote component reference	Int	Whether the component is connected to applicable hardware	Boolean
		Is the C button pressed	Boolean
		Is the Z button pressed	Boolean
		The thumb stick X position (-1 to 1)	Float
		The thumb stick Y position (-1 to 1)	Float
		Pitch angle (-90 to 90 deg)	Float
		Roll angle (-90 to 90 deg)	Float
		Raw acceleration data in the X-axis	Float
		Raw acceleration data in the Y-axis	Float
		Raw acceleration data in the Z-axis	Float

Wiimote

Description

The Wiimote component allows you to receive input data from a Nintendo Wiimote that has been paired with your PC via Bluetooth.

To pair a Wiimote in Windows 7:

1. Hold 1 and 2 buttons on the Wiimote

2. Go to Devices and Printers under the Control Panel and click Add a Device

3. Select Nintendo RVL-CNT-01 and click Next

4. Select Pair Without Using Code

The process may be different for other operating systems.

Once the Wiimote is paired you can pick it up by triggering the Connect input to the component. The first Boolean output will change to True when the connection has been established.

Connectors

Inputs	Type	Outputs	Type
Trigger to connect to a Wiimote paired with the PC	Trigger	Whether the component is connected to a Wiimote	Boolean
Switch LED1 on or off	Boolean	Is the A button pressed	Boolean
Switch LED2 on or off	Boolean	Is the B button pressed	Boolean
Switch LED3 on or off	Boolean	Is the Up button pressed	Boolean
Switch LED4 on or off	Boolean	Is the Down button pressed	Boolean
Switch rumble on or off	Boolean	Is the Left button pressed	Boolean
		Is the Right button pressed	Boolean
		Is the minus button pressed	Boolean
		Is the plus button pressed	Boolean
		Is the Home button pressed	Boolean
		Is the '1' button pressed	Boolean
		Is the '2' button pressed	Boolean
		Pitch angle (-90 to 90 deg)	Float
		Roll angle (-90 to 90 deg)	Float
		Raw acceleration data in the X-axis	Float
		Raw acceleration data in the Y-axis	Float
		Raw acceleration data in the Z-axis	Float

Wiimote IR

Description

The Wiimote IR component allows you to receive input data from the IR camera of a Nintendo Wiimote.

The Wiimote input must be connected to the Wiimote output of a Wiimote component in order to work. That Wiimote component must also be connected to a Wiimote that is paired with your PC.

The infra red camera at the front of the Wiimote can detect and track up to 4 IR points or dots. The visibility state and position of these dots is returned by this component.

Connectors

Inputs	Type	Outputs	Type
Wiimote component reference	Int	Whether the component is connected to a wiimote	Boolean
		Whether dot 1 is visible	Boolean
		X position of dot 1 (0-1)	Float
		Y position of dot 1 (0-1)	Float
		Whether dot 2 is visible	Boolean
		X position of dot 2 (0-1)	Float
		Y position of dot 2 (0-1)	Float
		Whether dot 3 is visible	Boolean
		X position of dot 3 (0-1)	Float
		Y position of dot 3 (0-1)	Float
		Whether dot 4 is visible	Boolean
		X position of dot 4 (0-1)	Float
		Y position of dot 4 (0-1)	Float

Wireless Input

Description

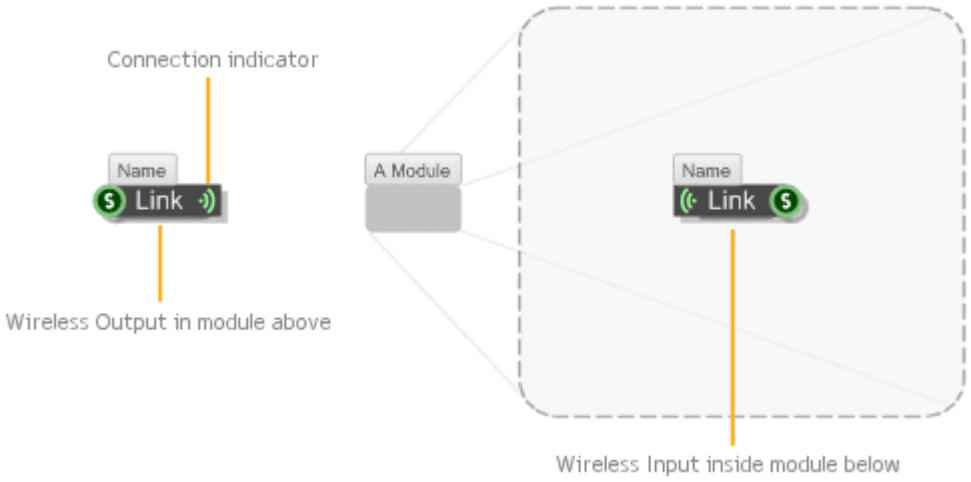
The Wireless Input and Wireless Output primitives provide two ends of a wireless link. Wireless links provide a mechanism for passing data through the module hierarchy without having to create any physical link.

The components have single Template input or output connectors. To change the type either right-click on the connector and choose a type from the pop-up menu or create a link from the connector to another component and the template will pick up the type.

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.

Connectors

Inputs	Type	Outputs	Type
N/A	N/A	Template connector which must be set to the type you want for the wireless link	Template

Wireless Output

Description

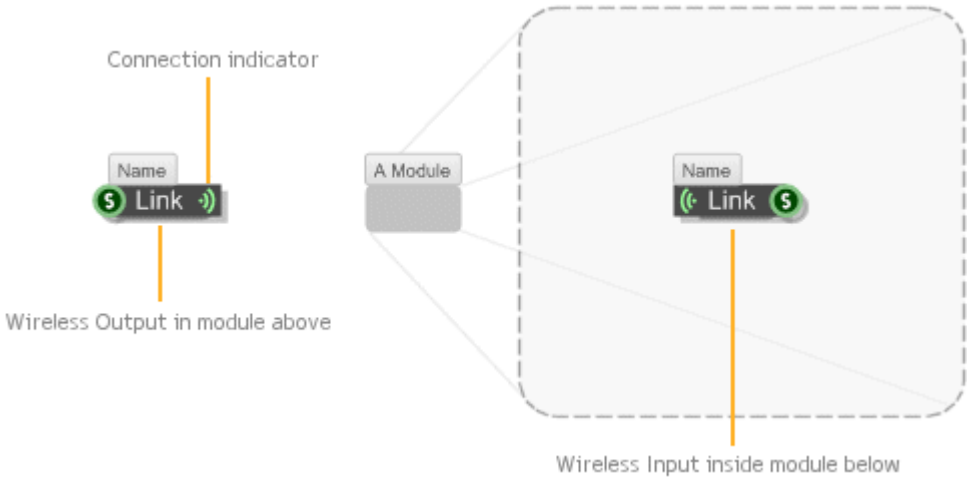
The Wireless Output and Wireless Input primitives provide two ends of a wireless link. Wireless links provide a mechanism for passing data through the module hierarchy without having to create any physical link.

The components have single Template input or output connectors. To change the type either right-click on the connector and choose a type from the pop-up menu or create a link from the connector to another component and the template will pick up the type.

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

4. The Wireless Input must appear in a module **below** the Wireless Output in the hierarchy
5. The Wireless Input and Output must have the **same label**
6. 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.

Connectors

Inputs	Type	Outputs	Type
Template connector which must be set to the type you want for the wireless link	Template	N/A	N/A

X Drag Accumulate

Description

The X Drag Accumulate component allows you to control a parameter by horizontal mouse drag operations on a mouse area. You define the minimum and maximum limits for the parameter then when a user drags horizontally on a mouse area the parameter will change between the limits accordingly.

You can define the scale at which the changes apply. A scale of 1 will move the parameter by 1 unit per grid square moved by the mouse. A scale of 0.1 will move the parameter by 0.1 units per grid square moved by the mouse.

The most common use of this component would be in slider type controls.

X10 Active Home

Description

The X10 Active Home component allows you to control and receive input data from a X10 home automation modules.

You must have an X10 computer interface connected to your PC.

X10 commands are sent as text. Simply connect a valid command string to the first input then trigger the Send input to send it.

An example X10 command string would be “sendplc A3 on”.

Data received from X10 modules like PIRs (passive infra red) are sent out through the “In” output.

Any errors that occur are sent through the Error output.

Connectors

Inputs	Type	Outputs	Type
X10 command string	String	Received strings from other X10 modules	String
Trigger to send the command	Trigger	Trigger when a command has been sent out	Trigger
		Any X10 error messages received	String

XBox 360

Description

The XBox 360 component allows you to receive input data from a connected XBox 360 controller.

To use this you must have an XBox 360 controller attached to your PC. If you trigger the Connect input on the component it will pick up the controller and you can then use the outputs to respond to controller input.

Connectors

Inputs	Type	Outputs	Type
Trigger to connect to a connected XBox controller	Trigger	Whether the component is connected to a controller	Boolean
Set left rumble level (0-1)	Float	Is the A button pressed	Boolean
Set right rumble level (0-1)	Float	Is the B button pressed	Boolean
		Is the X button pressed	Boolean
		Is the Y button pressed	Boolean
		Is the Up button pressed	Boolean
		Is the Down button pressed	Boolean
		Is the Left button pressed	Boolean
		Is the Right button pressed	Boolean
		Is Left Shoulder button pressed	Boolean
		Is Right Shoulder button pressed	Boolean
		Is the Start button pressed	Boolean
		Is the Back button pressed	Float
		Is Left Thumb stick pressed	Float
		Is Right Thumb stick pressed	Float
		Left Thumb Stick X (-1 to 1)	Float
		Left Thumb Stick Y (-1 to 1)	Float
		Right Thumb Stick X (-1 to 1)	Float
		Right Thumb Stick Y (-1 to 1)	Float
		Left trigger position (0-1)	Float
		Right trigger position (0-1)	Float

Connectors

Inputs	Type	Outputs	Type
Mouse messages from a	Mouse	The current value of the	Float

Mouse Area component		parameter	
Minimum value of the parameter	Float	Trigger sent before the parameter is about to change	Trigger
Maximum value of the parameter	Float	Trigger sent after the parameter has just changed	Trigger
Amount to move the parameter per grid square mouse movement	Float	Trigger sent when the drag operation ends (i.e. On mouse button release)	Trigger
Set the current value of the parameter	Float	Whether a drag operation is in progress	Boolean
Whether to hide the cursor during dragging (0=show, 1=hide and hold, 2=hide)	Int		

XY Drag Accumulate

Description

The XY Drag Accumulate component allows you to control a pair of parameters by mouse drag operations on a mouse area. You define the minimum and maximum limits for the parameters then when a user drags on a mouse area the parameters will change between the limits accordingly.

You can define the scales at which the changes apply. A scale of 1 will move the parameter by 1 unit per grid square moved by the mouse. A scale of 0.1 will move the parameter by 0.1 units per grid square moved by the mouse.

Use this component for controls where elements need to be moved around the display.

Connectors

Inputs	Type	Outputs	Type
Mouse messages from a Mouse Area component	Mouse	The current value of the horizontal parameter	Float
Minimum value of the parameter controlled by horizontal movement	Float	The current value of the vertical parameter	Float
Maximum value of the parameter controlled by horizontal movement	Float	Trigger sent before the parameter is about to change	Trigger

Amount to move the horizontal parameter per grid square mouse movement	Float	Trigger sent after the parameter has just changed	Trigger
Set the current value of the horizontal parameter	Float	Trigger sent when the drag operation ends (i.e. On mouse button release)	Trigger
Minimum value of the parameter controlled by vertical movement	Float	The current value of the vertical parameter	Float
Maximum value of the parameter controlled by vertical movement	Float	Trigger sent before the parameter is about to change	Trigger
Amount to move the vertical parameter per grid square mouse movement	Float	Trigger sent after the parameter has just changed	Trigger
Set the current value of the vertical parameter	Float	Trigger sent when the drag operation ends (i.e. On mouse button release)	Trigger
Whether to hide the cursor during dragging (0=show, 1=hide and hold, 2=hide)	Int	Whether a drag operation is in progress	Boolean

Y Drag Accumulate

Description

The X Drag Accumulate component allows you to control a parameter by vertical mouse drag operations on a mouse area. You define the minimum and maximum limits for the parameter then when a user drags horizontally on a mouse area the parameter will change between the limits accordingly.

You can define the scale at which the changes apply. A scale of 1 will move the parameter by 1 unit per grid square moved by the mouse. A scale of 0.1 will move the parameter by 0.1 units per grid square moved by the mouse.

The most common use of this component would be in slider type controls or knobs.

Connectors

Inputs	Type	Outputs	Type
Mouse messages from a Mouse Area component	Mouse	The current value of the parameter	Float
Minimum value of the parameter	Float	Trigger sent before the parameter is about to change	Trigger
Maximum value of the parameter	Float	Trigger sent after the parameter has just changed	Trigger
Amount to move the parameter per grid square mouse movement	Float	Trigger sent when the drag operation ends (i.e. On mouse button release)	Trigger
Set the current value of the parameter	Float	Whether a drag operation is in progress	Boolean
Whether to hide the cursor during dragging (0=show, 1=hide and hold, 2=hide)	Int		