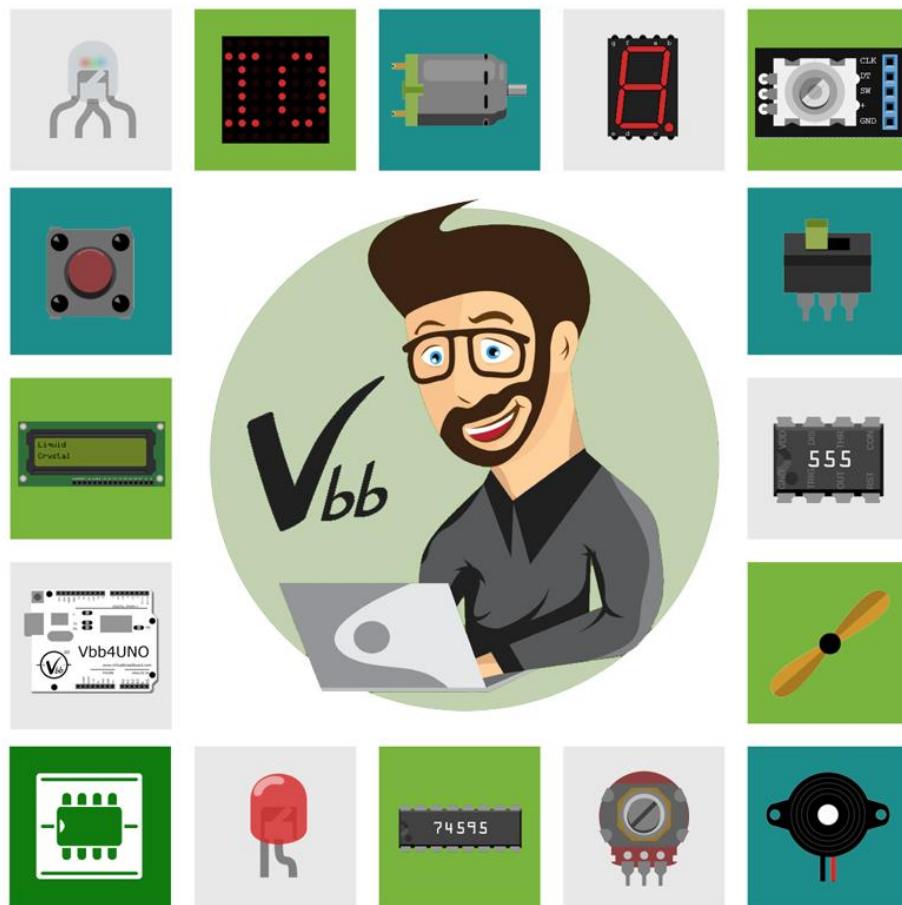


Vbb/Vbb4Arduino Classic

User Manual



Prototype Virtually, Make for Real

1 Contents

2	Introduction	11
2.1	 Virtual Breadboard (VBB)	11
2.2	 Vbb4Arduino	11
3	Breadboard Components.....	11
3.1	Behaviour Virtualization	11
3.2	Signal Level Components.....	12
3.3	API 'Smart' Level Components	12
3.4	Generic Layout Components	12
3.5	Function Block Component	13
4	Virtual Breadboard Circuit Modelling.....	13
5	Download and Installation.....	14
5.1	Download	14
5.1.1	Installation VBB	14
5.1.2	Dependencies	14
6	Licensing Extensions	15
6.1	 VBB 'InApp' Market Place	15
6.2	Activating a VBB Feature	15
6.2.1	Activating a feature	17
6.3	 VBB4Arduino 'InApp' Splash Screen.....	18
6.3.1	Activating the VBB4Arduino license	18
6.4	4. Enter the activation code in the splash screen	19
6.5	TroubleShooting	20
6.5.1	Firewalls.....	20
7	VBB Application	21
7.1	Solutions	21
7.1	Projects Files.....	21
7.2	Design Sheets	21

7.2.1	Breadboard.....	21
7.2.2	Java Source Projects	21
7.2.3	Logic Analyser.....	22
7.3	Project Select.....	22
7.3.1	TIPS	22
7.3.2	New Tab.....	23
7.3.3	New Project Template	23
7.3.4	'Built-in' Example Projects.....	24
7.4	Existing Tab.....	25
7.5	Recent.....	26
7.6	Design Mode.....	28
7.7	 VBB Design Mode.....	28
7.1	 VBB4Arduino Design Mode	29
7.1.1	App Window	29
7.1.2	App Menu	29
7.1.3	App Toolbar	31
7.1.4	Solution Explorer	32
7.1.5	View Mode.....	33
7.1.6	View Panes	34
7.1.7	Design Sheets	34
7.1.8	DesignSheet Toolbox	35
7.2	The properties panel	36
7.2.1	The property description panel.....	36
7.2.2	Empty panel.....	36
7.2.3	Status Toolbar.....	36
7.3	Runtime Virtualization.....	37
8	Design Sheets.....	37
8.1	Logic Analyser.....	39
8.1.1	Trace Log (*.VLG)	39
8.1.2	Adding a Logic Analyser.....	39
VBB 'Classic' User Manual	www.virtualbreadboard.com	6.0

8.1.3	Drag the Design Sheet into a View	39
8.1.4	3. Sampling Control Panel	42
8.2	Breadboard DesignSheet.....	45
8.2.1	Toolbar : Breadboard.....	45
8.2.2	Note on Zoom factor:	46
8.2.3	Placing a component from the Toolbox	47
8.2.4	Component Editing.....	47
8.2.5	Placing a component from the Toolbox	47
8.2.6	Select a component.....	47
8.2.7	Select a group of components.....	48
8.2.8	Append a component to a selection	48
8.2.9	Move a group of selected components.....	49
8.2.10	Shrink and grow a selection	49
8.2.11	Wiring Essentials.....	50
8.2.12	All Nets (--).....	56
8.2.13	Virtual Links	56
8.2.14	Show Links 	57
8.3	Toolbox : Breadboard	60
8.3.1	Toolbox Groups	60
9	Micro Controller Components.....	62
9.1	Arduino Family (with DUO).....	62
9.2	AVR ATMega328p.....	62
9.3	PicMicro 12/16/18 Family	62
9.1	Firmata Arduino UNO Host.....	63
9.2	ICEShield Arduino UNO Host	63
10	Generic Layout Components	64
10.1.1	Generic Layout Only	64
10.2	Generic DIP	64
10.3	Parametric	66
10.3.1	Using the Library Component.....	67

10.4	Marking up a generic component with a function block	70
10.5	 Radial Component.....	73
10.5.1	Parametric Model.....	73
10.5.2	Properties	74
10.6	 Axial Component.....	74
10.6.1	Parametric Model.....	74
10.6.2	Properties	75
10.1	 Generic Header Component.....	75
10.1	 Generic Male Header.....	76
10.1	 Generic Female Header	77
10.1	 Screw Terminal.....	78
11	Circuit Model Components.....	79
11.1	 Function Block	79
11.1.1	The Function Block expression	80
11.2	Function Block Expression.....	81
11.3	Function Block Expression Grammar.....	81
11.3.1	Function Block Editor Dialog.....	84
11.4	Function Block Usage	84
11.4.1	Dynamic Voltage Divider	85
11.4.2	Multi Tap Voltage Divider.....	87
11.4.3	Resistors	92
11.4.4	Resistors	92
11.5	 Diode	95
11.5.1	Parametric Model.....	95
11.5.2	Properties	96
11.5.3	Diodes	96

	11.6 PNP Transistor.....	97
11.6.1	Properties:	98
11.6.2	Useage	98
	11.7 NPN Transistor.....	99
11.7.1	Properties:	99
11.7.2	Useage	99
	11.8 Ceramic / Tantalum / Electrolytic Capacitor.....	100
11.8.1	Properties	100
12	IO Components	101
12.1	DIP1, DIP4, DIP8.....	101
12.1.1	Pinout	102
12.1.2	Properties	102
12.1.3	Usage	102
12.2	LEDN	103
12.2.1	Pinout	104
12.2.2	Properties	104
12.2.3	Usage	104
12.3	DotMatrixLED8x8.....	105
12.3.1	Pinout	105
12.3.2	Properties	106
12.4	Usage	106
12.5	JUMPN	107
12.5.1	Pinout	108
12.5.2	Properties	108
12.5.3	Usage	108
12.6	NumericKeyPad	110
12.6.1	Pinout	110
12.6.2	Properties	110

12.6.3	Usage	111
12.7	Seg7	112
12.7.1	Pinout	112
12.7.2	Properties	113
12.7.3	Usage	113
12.8	DigitalPort.....	116
12.8.1	Pinout	116
12.9	Properties	116
12.9.1	Usage	116
12.10	PushButton.....	117
12.10.1	Pinout	118
12.10.2	Properties	119
12.11	Usage	119
12.12	Switch	120
12.12.1	Pinout	120
12.12.2	Properties	120
12.12.3	Usage	121
12.13	PanelMeter.....	121
12.13.1	Pinout	121
12.13.2	Properties	121
12.13.3	Usage	122
12.14	SlidePot.....	123
12.14.1	Pinout	123
12.14.2	Properties	123
12.14.3	Usage	123
12.15	RotaryPot.....	124
12.15.1	Pinout	124
12.15.2	Properties	124
12.16	Usage	124
12.17	JoyStick	125

12.17.1	Pinout	125
12.17.2	Properties	125
12.17.3	Usage	126
12.18	MiniTerminal	126
12.18.1	Pinout	126
12.18.2	Properties	127
12.18.3	Usage	127
13	Breadboardable Components.....	128
13.1	LiquidCrystal	128
13.1.1	Properties	128
13.1.1	Usage	129
13.2	LDR – Light Dependent Resistor	129
13.2.1	Description	129
13.2.2	Properties	129
13.2.3	Usage	130
13.1	 Relay.....	130
13.1.1	Pins	131
13.1.2	Properties	131
13.1.3	Usage	132
13.2	 Solenoid.....	132
13.2.1	Pins	133
Properties	133	
13.2.2	Usage	133
13.1	 Trimmer	134
13.1.1	Pins	135
13.1.2	Usage	135
13.2	 Potentiometer	135
13.2.1	Pins	136

13.2.2	Usage	136
13.3	 Segment7	137
13.3.1	Pins	137
13.3.2	Usage	138
13.4	 Buzzer	138
13.4.1	Pins	139
13.4.2	Usage	139
13.5	Sharp Proximity Sensor	139
13.5.1	Pins	140
13.5.2	Usage	140
	555 Timer IC	141
13.6	Astable	142
13.7	ASTABLE Frequency Calculator.....	143
13.8	Calculating the values for AStable	144
13.9	Viewing the AStable Output.....	145
13.10	Monostable	146
13.11	Calculating the values for Monostable.....	147
13.12	Viewing the Monostable Output.....	148
14	API 'Smart' Components.....	150
14.1	WS2812	150
14.2	API.....	151
14.3	DS1307.....	152
14.3.1	API.....	153
14.4	24LC256 EEPROM	155
14.4.1	API.....	155
15	Appendix Named Colors	157

2 Introduction

VBB is a software platform designing ‘Breadboard’ form-factor electronic circuits and developing the microcontroller firmware that drive them.

Virtual Breadboard (VBB) and VBB4Arduino share the same codebase and this User Manual covers both versions of the software highlighting the differences with the different icons.



2.1 Virtual Breadboard (VBB)

VBB is the main software version with base functionality that can be extended by licensing additional modules

Available extensions (documented separately)

- **VBB4Arduino** – Arduino simulator for Arduino IDE and Java debugger
- **PicMicro** – PIC12/16/18 simulators and assembly debuggers
- **Firmata** – Remoting protocol host/client
- **Fritzing** – Importer for extended footprint documentation support

2.2 Vbb4Arduino

Vbb4Arduino is a version of VBB dedicated to the Arduino learner. It is an intentionally limited standalone ‘sandbox’ version of VBB ideal for taking the first steps learning about physical computing. It has reduced components, a simplified user interface and support only for the VBB4Arduino license.

The Vbb4Arduino license can be used with either VBB or VBB4Arduino software.

3 Breadboard Components

In a perfect world every single component in the world would be reverse engineered and virtualized at the lowest level with complete perfection including firmware bugs of the various revisions of all the millions of components. Since this is impossible for any price let alone for the enthusiast market VBB uses innovative techniques to virtualize complex components in a useful yet simplified way.

3.1 Behaviour Virtualization

VBB components are models of the way real world electronic components behave from the point of view of the microcontroller driving them.

Important! VBB does not support analog circuit simulation like SPICE. Sorry but VBB just doesn't work that way.

Instead VBB is a circuit modelling tool with a focus on the microcontroller application code and uses simplified electronic models to virtualize the electronics for testing your code.

Nevertheless, a huge number of circuits can be modelled this way especially if you take the time to learn a few VBB tricks.

There are several types of Components that can be placed in a Breadboard

- The Microcontroller – the central component to any layout
- Low Level ‘Signal Level’ components that virtualize using circuit signals
- High Level ‘API’ components that virtualize using an API interface
- Generic Layout – Documentation components which do not virtualize
- Function Block – Circuit modelling component to supplement circuit behaviour

3.2 Signal Level Components

Signal level components are modelled based on the signals and protocols received at their pins. There are many classes of signal level components.

The simplest are digital HIGH/LOW response components such as switches and lights. Then there are simple wave function components such LED’s that respond to PWM signals, Servos’ to periodic Pulsed Signals, Steppers to quadrature encoded signals. These are relatively simple components to implement and the VBB CDK can make some of these in a drag-and-drop way.

Then there are complex protocol decoding components such as the Liquid Crystal Display and CMOS4000 / TTL74 series logic. To implement these types of components requires the implementation of the underlying protocol based on the timing specification of the logic controller. It also requires the implementation of the logic controller, for example the HD44780U for the Liquid Crystal Display.

This is a highly technical task, but the advantage is the component can then be driven by ‘bit-banging’ the IO with the appropriate protocol.

3.3 API ‘Smart’ Level Components

In contrast to Low Level Components, API components are components that only virtualize when driven via the Java API. The wiring of the component only links the component to the microcontroller. The actual communication to the device is done in an abstract communications channel rather than using signal level protocols. This makes it practical to implement the component virtualization but requires the use of the API to make it work.

3.4 Generic Layout Components

These are components that are ‘pictures’. Many other circuit layout programs, fritzing for example, work only with pictures so they remain a useful part of the design.

The generic components can be used to complete the circuit for make-ability by the Virtual Breadboard FAB or for documenting a layout to share with others.

3.5 Function Block Component

The function block component is a special component which can be used to mock a variety of circuit behaviours. When used in conjunction with Generic Layout Components it can be used as a type of component development kit.

4 Virtual Breadboard Circuit Modelling

Virtual Breadboard can be used as a Circuit Emulator for some types of Circuits. In particular VBB is not a SPICE simulation and does not currently resolve circuit current so you cannot use it for analog circuit-analysis or parameter tuning. However for a wide variety of ‘physical computing’ circuits VBB emulations work just fine.

Whats the difference between a Simulation and an Emulation?

Emulation models behaviour whereas simulation emerges behaviour. A behaviour might be to turn on a LED connected via a resistor to a microcontroller pin. A simulation might compute the circuit resistance and equivalent resistance of the power driver of the pin along with the voltage curve and forward voltage of the diode to resolve the instantaneous current and then lookup the luminosity curve of the diode to render a faithful representation of the color and intensity of the LED the user might expect to see. An emulation on the other hand just draws a LED as on when the PIN is HIGH and off when the PIN is LOW. The resulting behaviour from the user and microcontroller are the same. A LED is *on* when the pin is *HIGH* and the LED is *off* when the pin is *LOW*. Naturally emulations are faster to calculate and easier to implement. VBB is a circuit *emulator* and SPICE is a circuit *simulator*.

Its important to understand this difference when using VBB because you cannot place capacitors and resistors and expect for example behaviours such as an oscillator to emerge. Instead you would add a oscillator function block to model the behaviour you expect from the circuit block.

Several work-arounds to support common behaviours using properties of the discrete components are available but you should be aware of the limits.

5 Download and Installation

All VBB is free to download. This makes it easy to manage updates etc. As a minimum VBB can be used to layout Breadboard designs for free. This is already useful and there are other popular Breadboard layout design tools that are limited to design only.

In addition VBB has virtualization features which become enabled when you license an extension.

5.1 Download

VBB is available for download from

<http://www.virtualbreadboard.com/DocView.html?doc=Downloads/Downloads>

5.1.1 Installation VBB



VBB.exe



VBB4Arduino.exe

VBB/VBB4Arduino ships as a standalone executable packaged in a ZIP file.

Unzip the executable to the location where you would like to run it from. For example the desktop. Execute .exe to start the application. Note this is not an installer and will not create shortcuts. You will need to create your own shortcuts if required

5.1.2 Dependencies

VBB requires .NET 4.0 runtime to be installed and depending on your installation you may also require to install .net 2.0

5.1.2.1 *Administrator Permissions*

When you first run VBB you may need to use the Administrator account to give permission to install its configuration files in the AppData.

VBB/VBB4Arduino will install its examples etc in the follow directory



<User>AppData\Local\Virtual Breadboard\VBB <version> directory



User>AppData\Local\Virtual Breadboard\VBB4Arduino <version> directory

6 Licensing Extensions

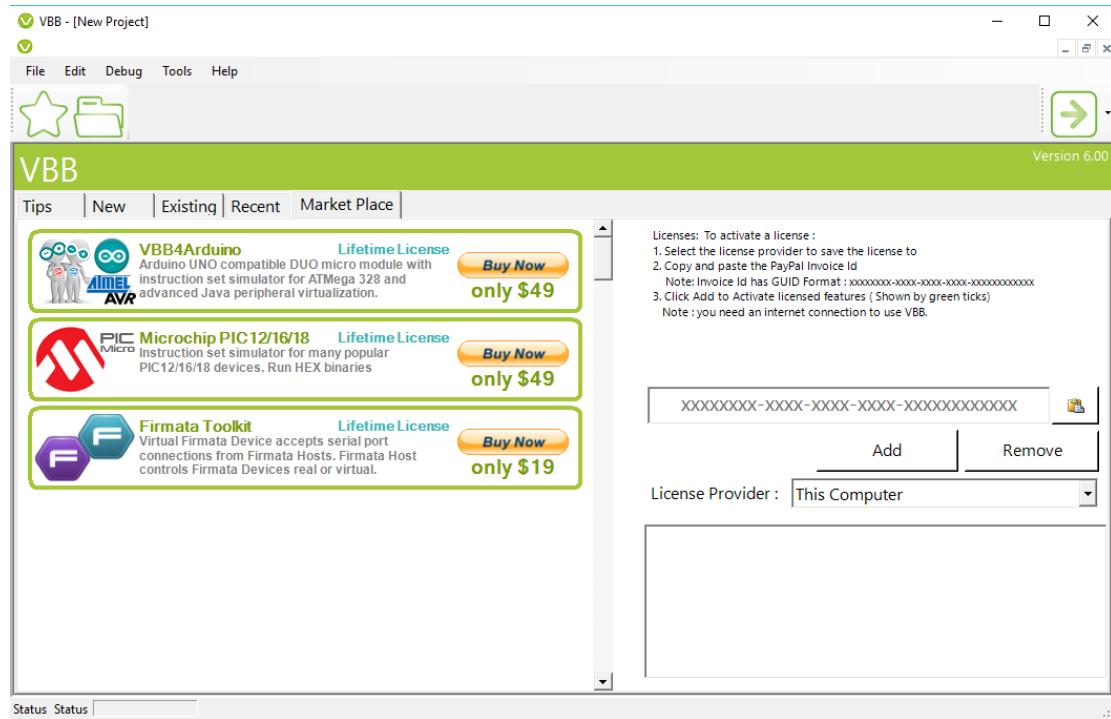
Extensions can be purchased online from the WebShop or ‘In-App’.

<http://www.virtualbreadboard.com/DocView.html?doc=WebShop/WebShop>

There is no evaluation version for VBB/VBB4Arduino ‘Classic’. The Microsoft App Store version of VBB has free evaluation options.

6.1 VBB ‘InApp’ Market Place

The VBB *Market Place* Tab shows a list of products and current activation status. You can purchase products directly from the *Market Place* by clicking the BUY NOW button of the feature to purchase. As more features are added they will appear in the marketplace



6.2 Activating a VBB Feature

When you buy a VBB Feature using PayPAL the PayPAL receipt contains an invoice number. This invoice number is the *activation code* for the purchased feature.

Here is a sample receipt, the invoice number is highlighted in red.

You can now ship the items. To see all the transaction details, log in to your PayPal account.

It may take a few moments for this transaction to appear in your account.

Seller Protection - Not Eligible

Buyer Jim Bloggs jimbo@hotmail.com	Instructions to merchant The buyer hasn't entered any instructions.																				
Shipping address Jim Bloggs Enterprises 1 Blogs Drive 101 AnyLands	Shipping details You haven't added any shipping details.																				
<table><thead><tr><th>Description</th><th>Unit price</th><th>Qty</th><th>Amount</th></tr></thead><tbody><tr><td>VBB#26 : VBB4Arduino</td><td>\$49,00 USD</td><td>1</td><td>\$49,00 USD</td></tr><tr><td></td><td>Subtotal</td><td></td><td>\$49,00 USD</td></tr><tr><td></td><td>Total</td><td></td><td>\$49,00 USD</td></tr><tr><td></td><td>Payment</td><td></td><td>\$49,00 USD</td></tr></tbody></table>		Description	Unit price	Qty	Amount	VBB#26 : VBB4Arduino	\$49,00 USD	1	\$49,00 USD		Subtotal		\$49,00 USD		Total		\$49,00 USD		Payment		\$49,00 USD
Description	Unit price	Qty	Amount																		
VBB#26 : VBB4Arduino	\$49,00 USD	1	\$49,00 USD																		
	Subtotal		\$49,00 USD																		
	Total		\$49,00 USD																		
	Payment		\$49,00 USD																		

Payment sent to sales@virtualbreadboard.com

Invoice ID: 987f8e5e-a173f-464b-a1f1-0f7f0774651b 

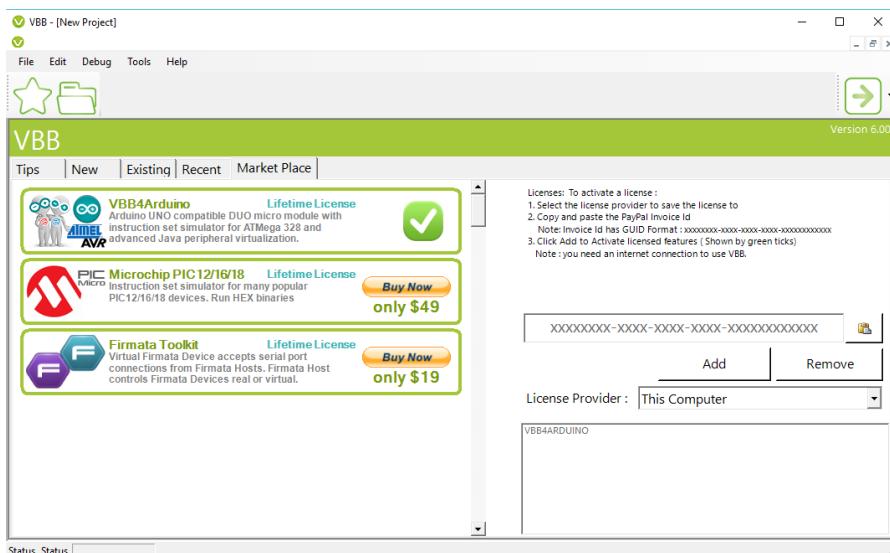
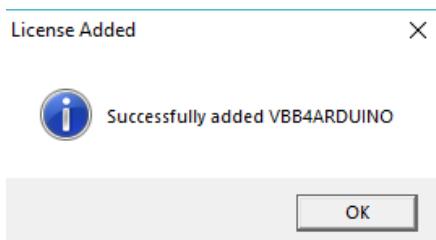
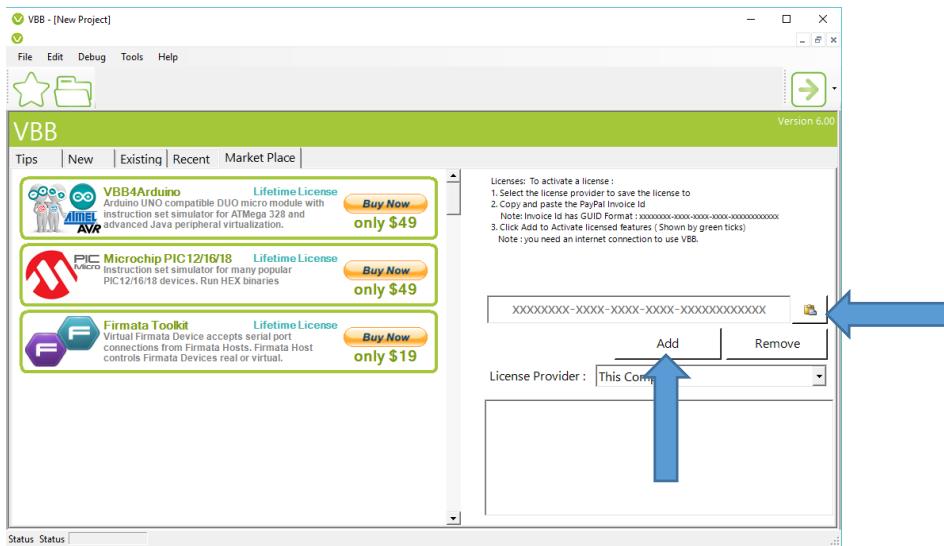
 Questions? Go to the Help Center at www.paypal.com/nl/help.

Lift your withdrawal and receiving limits. Log in to your PayPal account and click **View limits** on your Account Overview page.

Please do not reply to this email. This mailbox is not monitored and you will not receive a response. For assistance, log in to your PayPal account and click **Help** in the top right corner of any PayPal page.

6.2.1 Activating a feature

To activate a feature you have purchased in VBB you first start VBB and locate the MarketPlace tab. Then you copy the invoice number from the PayPal receipt and click the Paste Button to enter it into the Activation TextBox and click the Add button. The activation code will be checked online and when valid the feature will become ticked with a green box to indicate its activated and ready to use.



6.3 VBB4Arduino 'InApp' Splash Screen

VBB4Arduino only supports the VBB4Arduino license key.



You can purchase the license by clicking the PayPal button on the on the splash screen



6.3.1 Activating the VBB4Arduino license

When you buy the VBB4Arduino license the PayPAL receipt contains an invoice number. This invoice number is the *activation* code for the purchased feature.

Here is a sample receipt, the invoice number is highlighted in red.

The invoiceId is the activation code for VBB4Arduino. The licenseKey on PayPal invoice so you can always look it up in the future in your paypal account.

You can now ship the items. To see all the transaction details, log in to your PayPal account.
It may take a few moments for this transaction to appear in your account.
Seller Protection - Not Eligible

Buyer Jim Bloggs jimb@hotmail.com	Instructions to merchant The buyer hasn't entered any instructions.			
Shipping address Jim Bloggs Enterprises 1 Blogs Drive 101 AnyLands	Shipping details You haven't added any shipping details.			
Description		Unit price	Qty	Amount
VBB#26 : VBB4Arduino		\$49,00 USD	1	\$49,00 USD
		Subtotal		\$49,00 USD
		Total		\$49,00 USD
		Payment		\$49,00 USD

Payment sent to sales@virtualbreadboard.com
Invoice ID: 987f8e5e-a173f-464b-a1f1-0f7f0774651b

Questions? Go to the Help Center at: www.paypal.com/ml/help.
Lift your withdrawal and receiving limits. Log in to your PayPal account and click **View limits** on your Account Overview page.
Please do not reply to this email. This mailbox is not monitored and you will not receive a response. For assistance, log in to your PayPal account and click **Help** in the top right corner of any PayPal page.

6.4 4. Enter the activation code in the splash screen

When VBB4Arduino starts it will begin with a splash screen

Enter the activation code into the textbox and the code will be checked using the internet and the software activated. You should see a big Green tick once activated.



Press Start to continue using the software

6.5 Troubleshooting

Some of the features of VBB are now hosted in the cloud in the Google AppEngine server and accessed by license keys that are generated when features are purchased. It all works pretty seamlessly as long as you have a quality full time internet connection.

Fortunately, the internet has become more or less ubiquitous but there are some situations where it doesn't work out

6.5.1 Firewalls

Internet Access is made via standard Http WebServices on Port 80. This looks just like Browser Access and usually there are no problems with firewalls.

However rarely, maybe 1 in 100 users find their configurations are not compatible with VBB. The primary problem is firewalls. There are 2 firewall that can cause issues.

6.5.1.1 Chinese Firewall

For some reason China blocks access to the Google App Engine host url. It's quite a clever block too because I have tried to route via a proxy but it didn't work. So for now the only solution is a VPN or we just refund Chinese customers and say sorry.

6.5.1.2 Work

Sometimes there is a domain/password proxy that needs to be configured. The 'Classic' versions have a setting for this now in the Tool→Options dialog. The Windows App libraries don't offer this same setting and rely on the Browser settings however this seems a bit indirect so a source of potential issues.

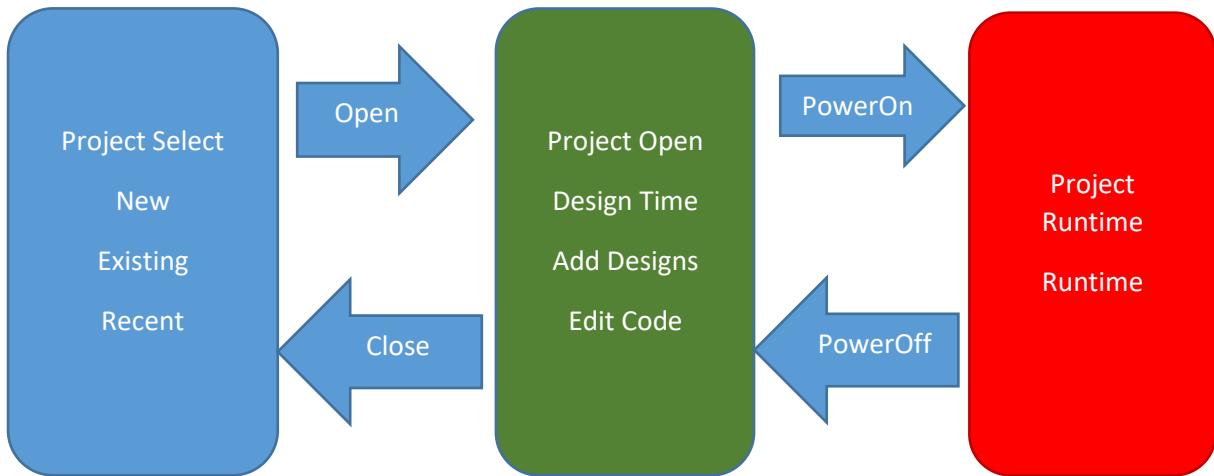
6.5.1.3 Antivirus/ Internet Security

Applications such as Norton Internet security block internet access preventing authentication and pay-for-feature usage. You might have to add VBB.exe to the exception list of your Internet monitor and/or Firewall in order for Activation to function correctly.

7 VBB Application

VBB is an Integrated Development Environment for the purpose of designing, simulating and building microcontroller based electronics hardware systems.

There are 3 modes, Project Select, Design Mode and runtime mode



The fundamental concept is the DesignSheet. Designsheets are contained in projects and projects are contained Solutions.

7.1 Solutions

A solution The VSM '*Virtual System Model*' Solution file is the root file for VBB. It contains references to a collection of Projects and contains global settings for the solution.
Vbb4Arduino solutions have only 1 project.

7.1 Projects Files

A project is a collection of Design Sheets.

7.2 Design Sheets

7.2.1 Breadboard

The Breadboard is the primary DesignSheet in VBB and consists of a collection of components wired together to create circuits

7.2.2 Java Source Projects

The Source Project DesignSheet contains a collection of Java source files

7.2.3 Logic Analyser

The Logic Analyser is a special function DesignSheet containing a Logic Analyser instrument for circuit analysis.

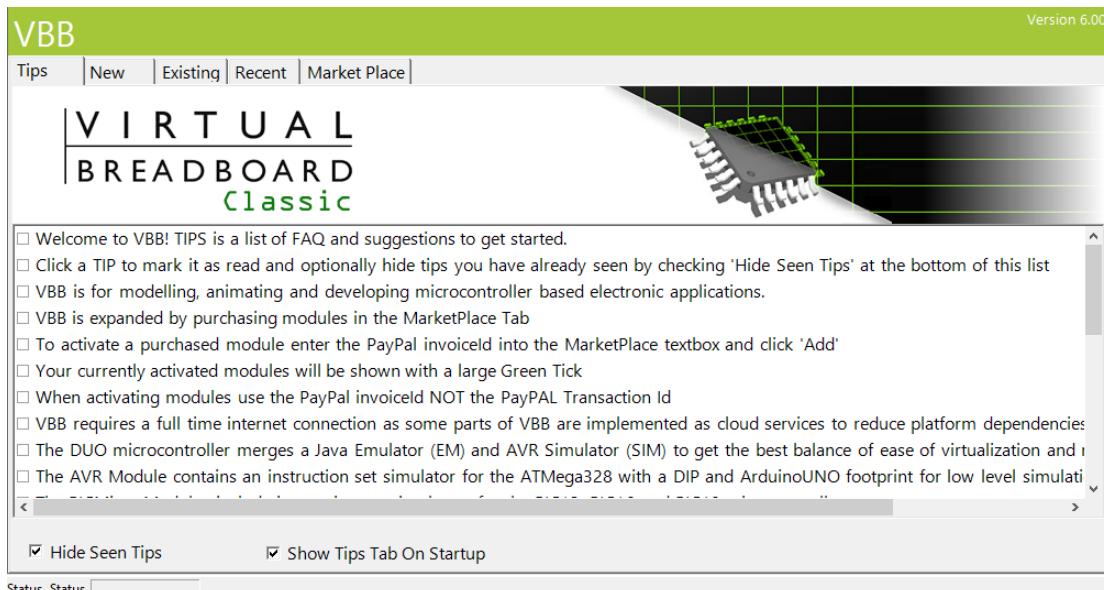
7.3 Project Select

The project select form helps you create a new project or open an existing project. Select a tab to perform the task

-  **Tips** – The Tips tab contains a list of tips that can be useful when getting started
-  **New** : Create a new project from scratch or with a helper
-  **Existing** : Open an existing project from the file system
-  **Recent** : Open an existing project from a list of projects recently worked on
-  **MarketPlace**: License VBB extensions

7.3.1 TIPS

The Tips tab contains a list of tips that can be useful when getting started



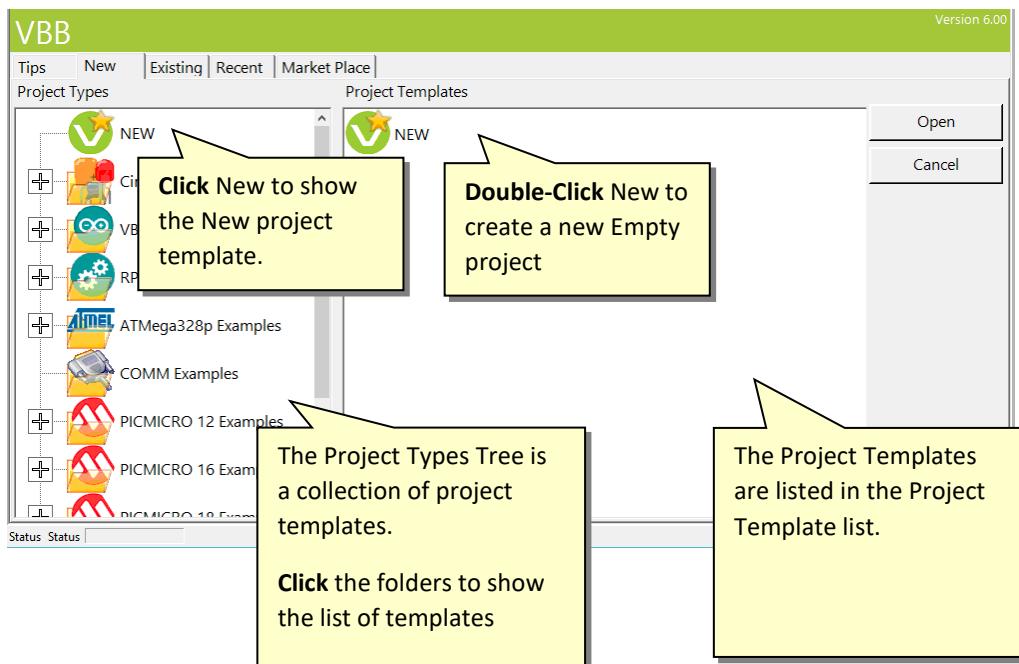
Check Tip Items – Check each tip once you have read as a note to yourself that it has been read

Hide Seen Tips – When checked the tips you have already read and checked as read will not display in the list

Show Tips Tab On Startup – When checked the TIPS Tab will be selected as the default Tab when the project select dialog is show. Unselect to default to NEW or Existing

7.3.2 New Tab

The New Tab is used for creating New projects from project templates. A project template is a complete solution based on an example or project type which you can use to quickly get started. The templates are organized into a tree of project types to help you locate a template that best suits your embedded application challenge. The *Project Types Tree* on the left hand side selects the project type and the *Project Templates* lists the available templates for the selected type.

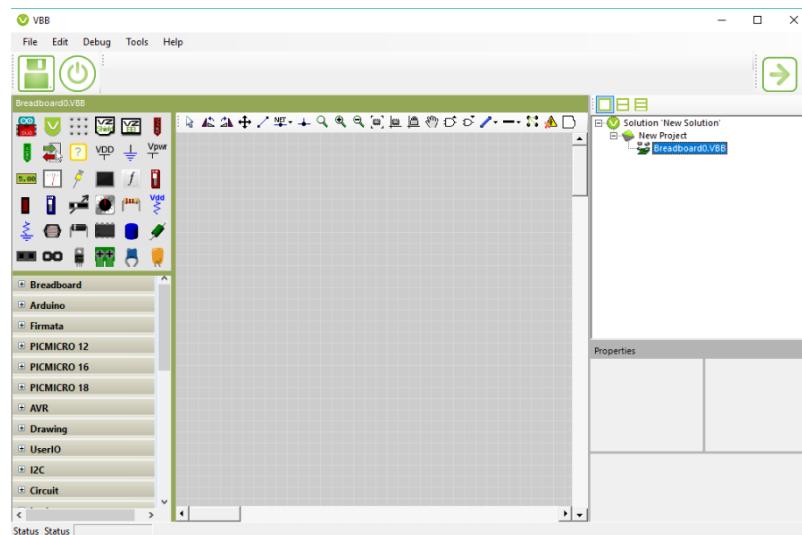


7.3.3 New Project Template



Select/Open or DoubleClick the New project template to create a new Blank project. The blank project created will contain a project called 'New Solution' with a single project called 'New Project' with a single Breadboard called 'Breadboard0.VBB'.

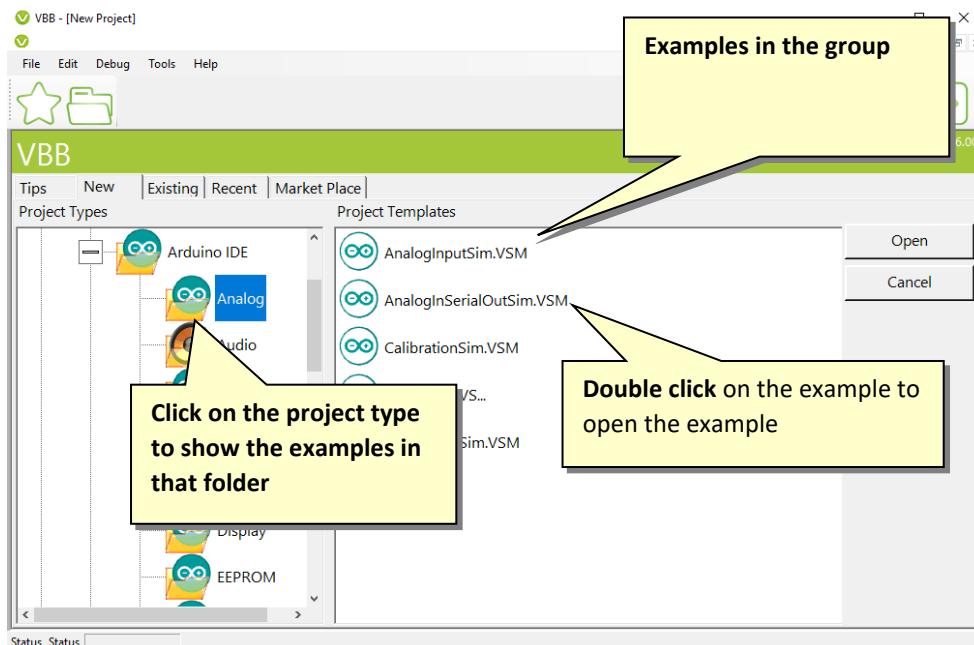
The empty project is created in scratch disk.



7.3.4 ‘Built-in’ Example Projects

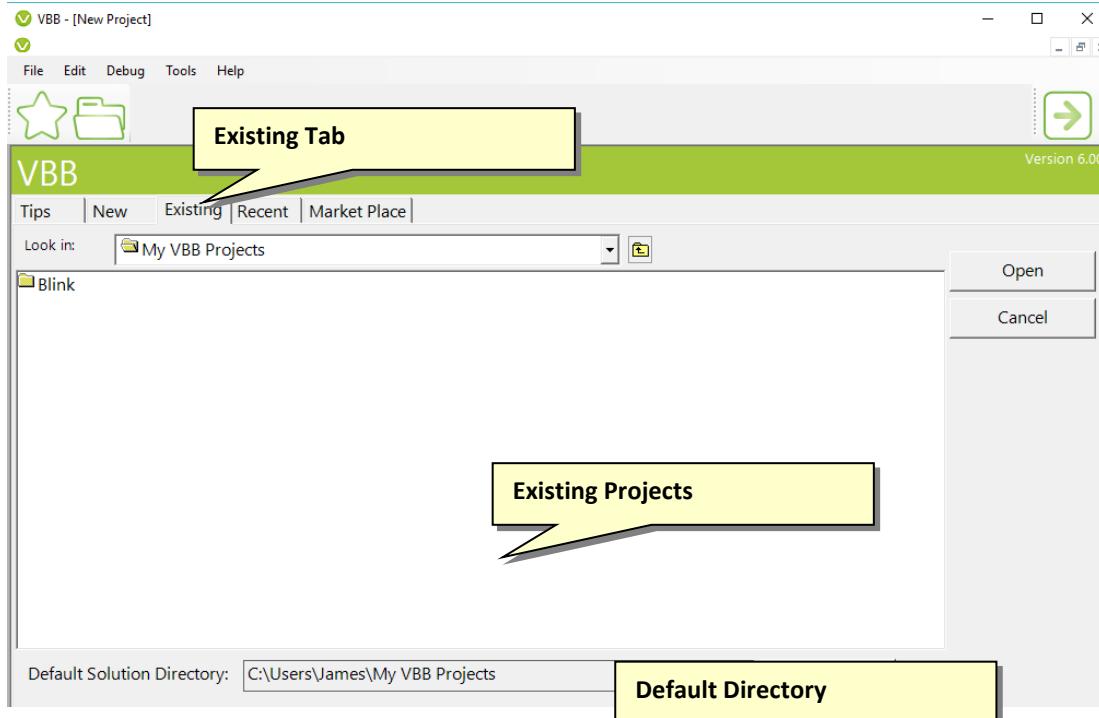
The examples are organised in a tree of example ‘project types’

To open a reference example click the folder on the left to show the available examples in the group then Select/Open or double click the example in the right Project Templates panel to open the example

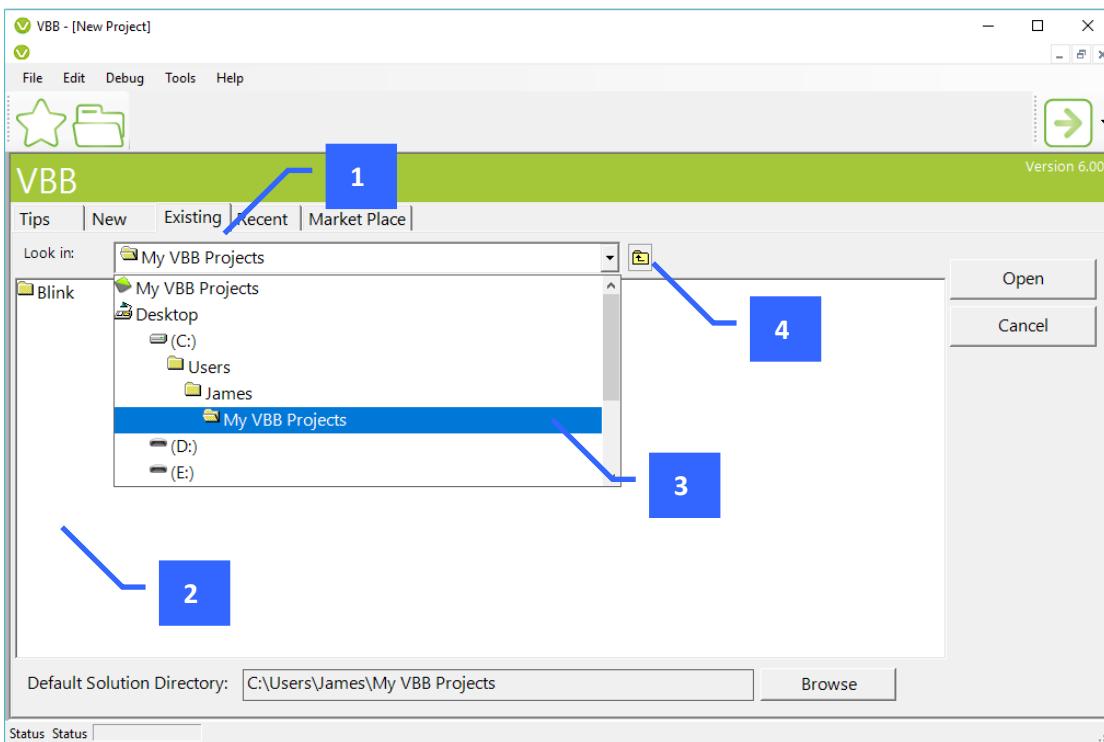


7.4 Existing Tab

The Existing Tab allow projects that already exist to be located and opened using the inbuild file browser

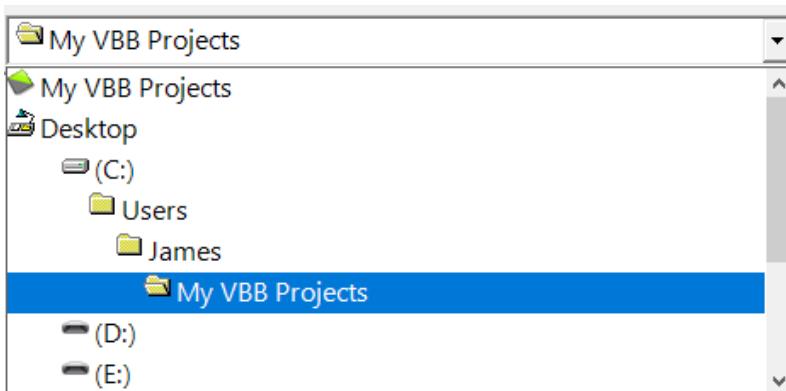


The Default Directory is the location that is first shown... By default this is <User>My Vbb Projects but you can change the default location using the Browse Button to navigate to a new folder location.



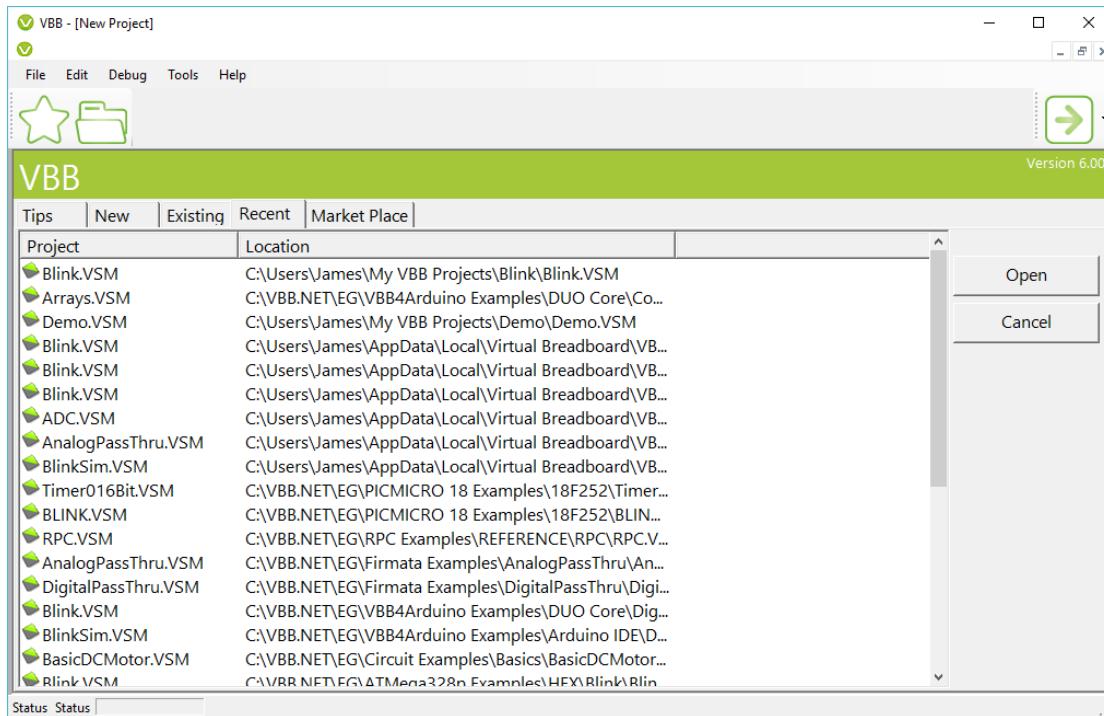
1. Existing Tab
2. List of VBB Solutions in the current directory
3. Drop down navigator
4. Move up to the previous directory

The navigation dropdown list allows you to search the computer for directories containing an existing VBB solution to open



7.5 Recent

The recent tab hold the most recently used project list for quick access to recently used projects.



7.6 Design Mode

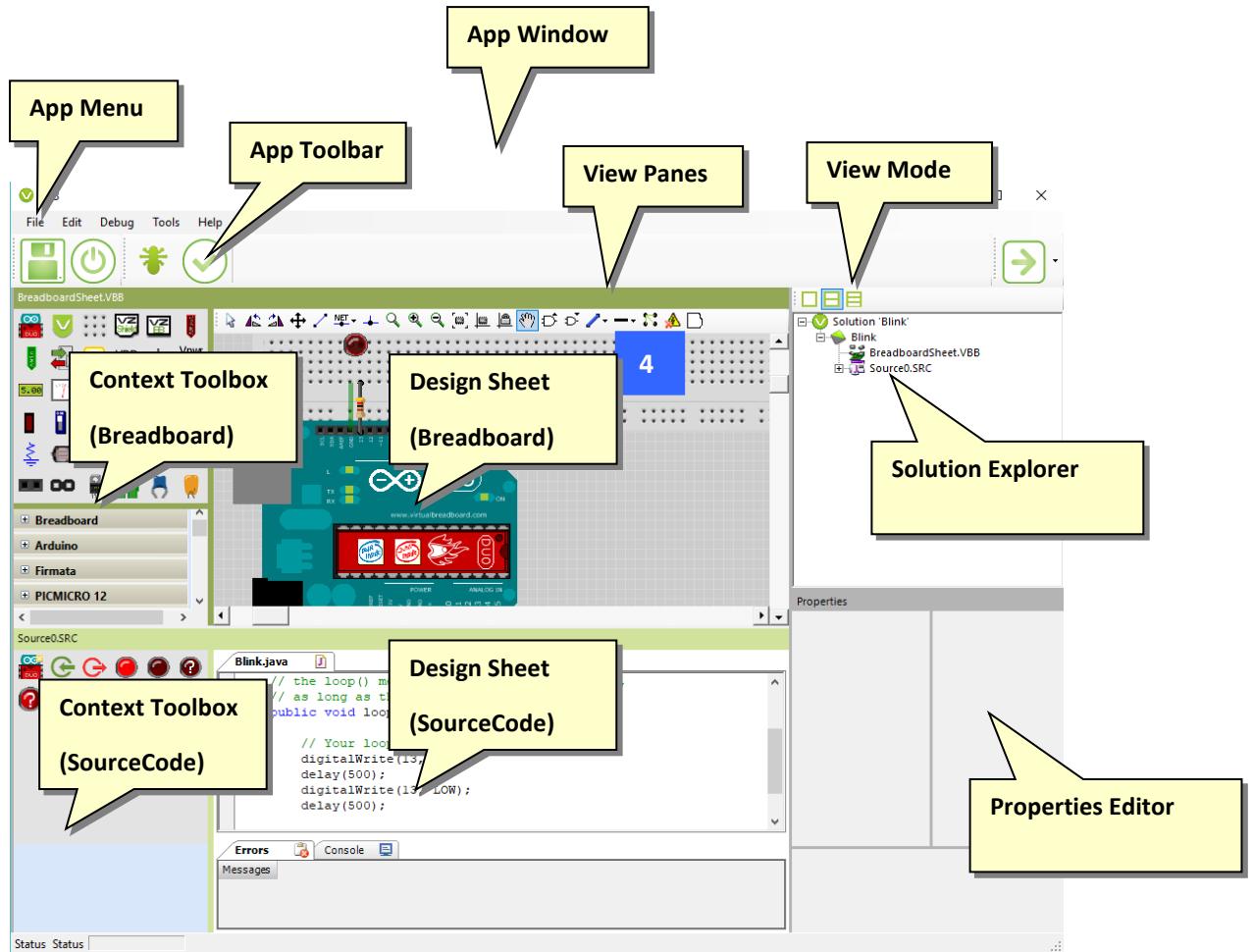
When the project opens the state of the VBB UI moves into DesignMode with the following environment features

The organisation is based around the Solution Explorer which is a tree of projects containing design sheets as leaves.

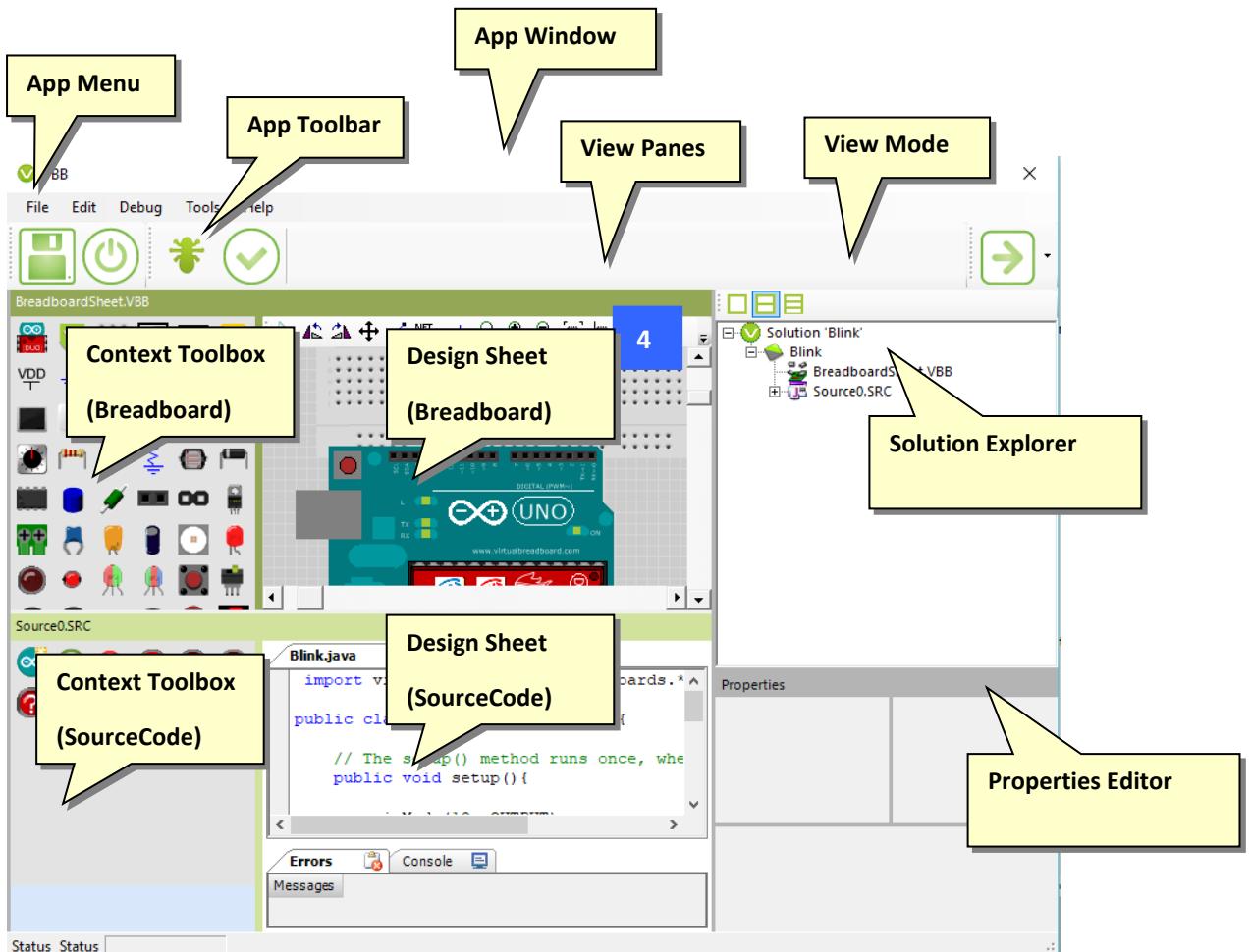
DesignSheets are Viewed in ViewPanes and to move a design sheet to a view pane it is dragged and dropped from the solution explorer tree into the design sheet.

Properties of Design Sheets and components in designs are shown in the Properties Editor

7.7 VBB Design Mode



7.1 VBB4Arduino Design Mode



7.1.1 App Window

The VBB Application is a Multiple Document Interface (MDI) style application where the document being managed is the DesignSheet.

7.1.2 App Menu

The App Menu

- File : Access functions for opening and saving files
- Edit : Access functions for editing
- Tools : Access tools
- Help : Access application information

7.1.2.1 File

Icon	Name	ShortCut	Description
	New	Ctrl + N	Opens the New Project Dialog with the New Tab selected. Only available when no solution is loaded.
	Open	Ctrl + O	Opens the New Project Dialog with the Existing Tab selected. Only available when no solution is loaded.
	Close Solution		Closes the current solution and returns to the closed solution state. Only available when a solution is loaded.
	Save	Ctrl + S	Saves the current solution
	Save As		Opens the Save Dialog prompting for the new project name. Only available when a solution is loaded.

7.1.2.2 Edit

Icon	Name	ShortCut	Description
	Undo	Ctrl + Z	Undo's the previous action (s)
	Redo	Ctrl+Y	Redo's the previous action (s)
	Cut	Ctrl + X	Copies and deletes the current selection to the clipboard and deletes the selection from the current DesignSheet. Only available when a DesignSheet is active and components or code have been selected.
	Copy	Ctrl + C	Copies the current selection of the current DesignSheet to the clipboard Only available when a DesignSheet is active and components have been selected.

	Paste	Ctrl + P	Pastes the components copied to the clipboard to the current DesignSheet. Only available when a DesignSheet is active and components have been copied to the clipboard.
	Delete	Del	Deletes the current selection from the current DesignSheet. Only available when a DesignSheet is active and components have been selected.

7.1.2.3 Help Menu

Name	Description
Update ICEShield Firmware	Starts the ICEShield firmware update dialog. See ICEShield
About	Opens the About dialog to display version information
Virtual Breadboard HomePage	Launches a Browser and opens the Virtual Breadboard HomePage

7.1.3 App Toolbar

The VBB application Toolbar gives quick access to the core Vbb4Arduino features

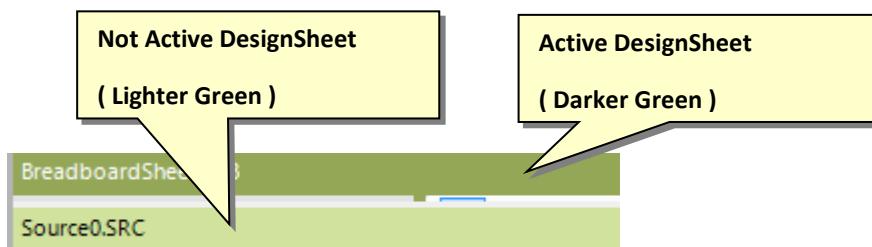


Name	Name	Description
	Save	Saves the current project. If the project has not yet been saved it will launch the SaveAs dialog
	PowerUp	Powers Up virtualization of the active circuit model . Only available if you have purchased a license and from Design Mode
	Debug	Start Code in Debug Step Mode

	Verify	Performs a code check on source code projects
	PowerDown	Powers down the circuit virtualization and returns to Design Mode

7.1.3.1 Active DesignSheet

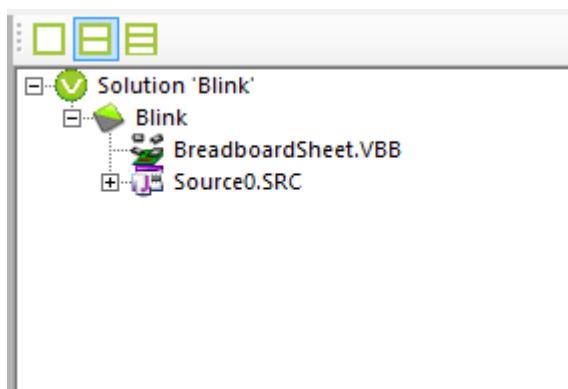
The Active design sheet is the currently selected sheet which is the design sheet highlighted darker green



1. You can always click into the TitleBar to select the DesignSheet as the current DesignSheet
2. Some DesignSheets allow you click anywhere on the DesignSheet to select it

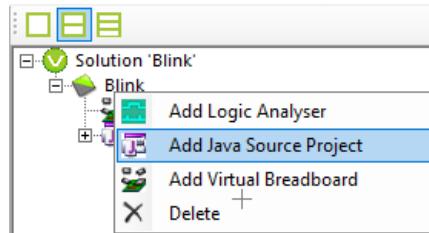
7.1.4 Solution Explorer

The solution Explorer contains a tree representing the DesignSheets in the solution.

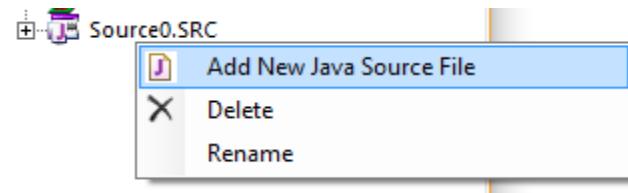


The Solution Explorer is context sensitive.

Right Click on a Project to add Logic Analyser, Breadboard, Arduino Java Source or Arduino Sketch Source DesignSheet



Right Click on a Java Source Project to add a Java source file to a Source Code Project

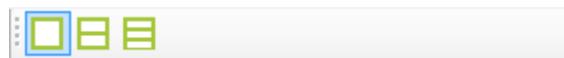


Deleting : Right click on the element to delete and select the delete drop down menu

Rename : Right click on the element to rename and in-place edit the new name

7.1.5 View Mode

The design panel has up to three design panes. Each pane may contain a different arrangement of DesignSheet9s)



	One Panel	Show the design panel as a single design pane. Any selected DesignSheet is shown in the single pane
	Two Panel	Splits the design panel into two separate design panes
	Three Panel	Splits the design panel into three separate design panes.

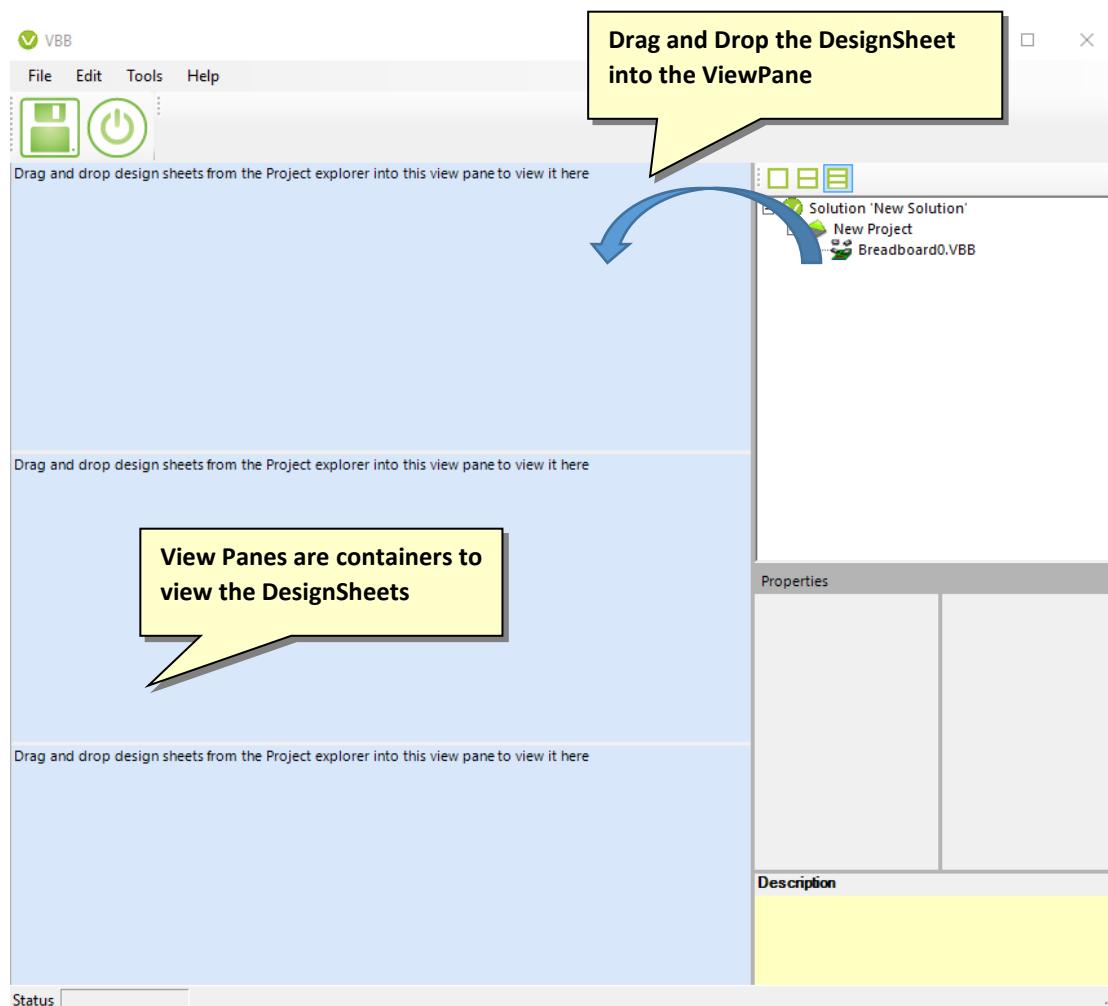
When you double click a DesignSheet in the navigation tree it is selected into one of the available panes according to its default viewing position.

You can override the default location by dragging and dropping the design from the navigation tree to the desired panel.

When switching between layouts the previous layout remembers which DesignSheets are in which panel.

7.1.6 View Panes

The individual view panes are MDI child panes!



7.1.7 Design Sheets

DesignSheets are the documents managed by VBB. They have a design area and an optional Toolbox. There are standard and extended design sheets.

- Breadboard
- Logic Analyser
- Source Code Project (extension)

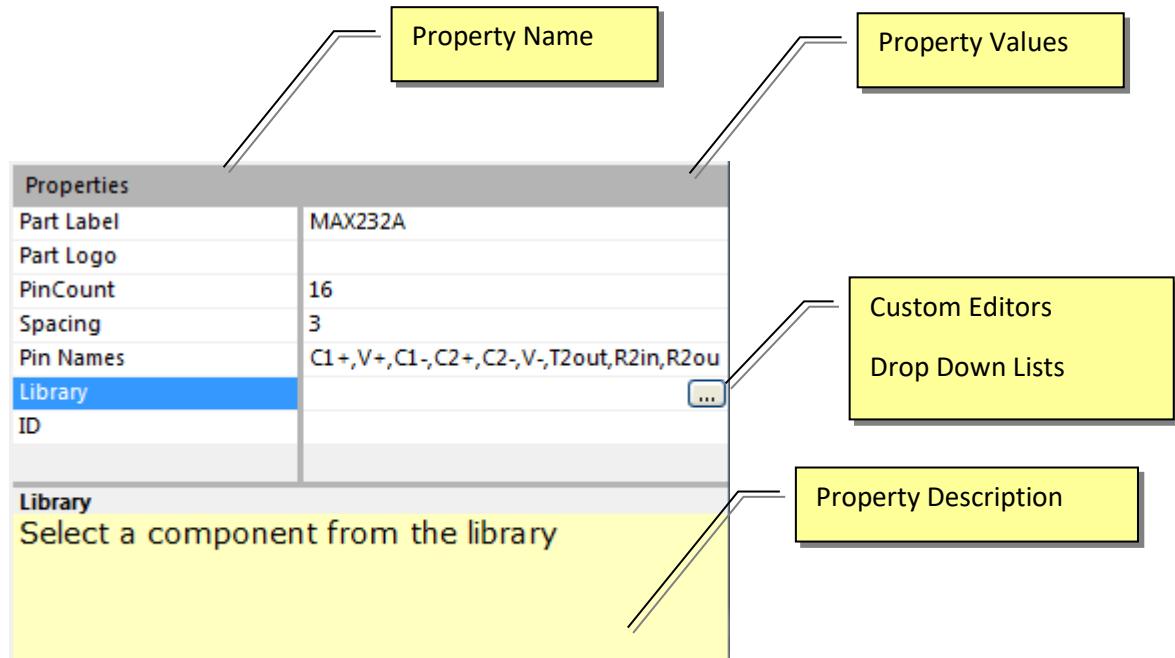
7.1.8 DesignSheet Toolbox

The toolbox contains the collections of components specific to the DesignSheet.

7.2 The properties panel

Many items, DesignSheets and Components have Properties

The properties editor is populated when components are selected with DesignSheets. The properties are specific to individual components.



When you select a component the properties box is populated with the component properties.

7.2.1 The property description panel

When a component is selected the description of the property is displayed in the property description panel

7.2.2 Empty panel

When a panel has no DesignSheet it appears blue. You can fill the panel with a DesignSheet by dragging and dropping a design into the panel

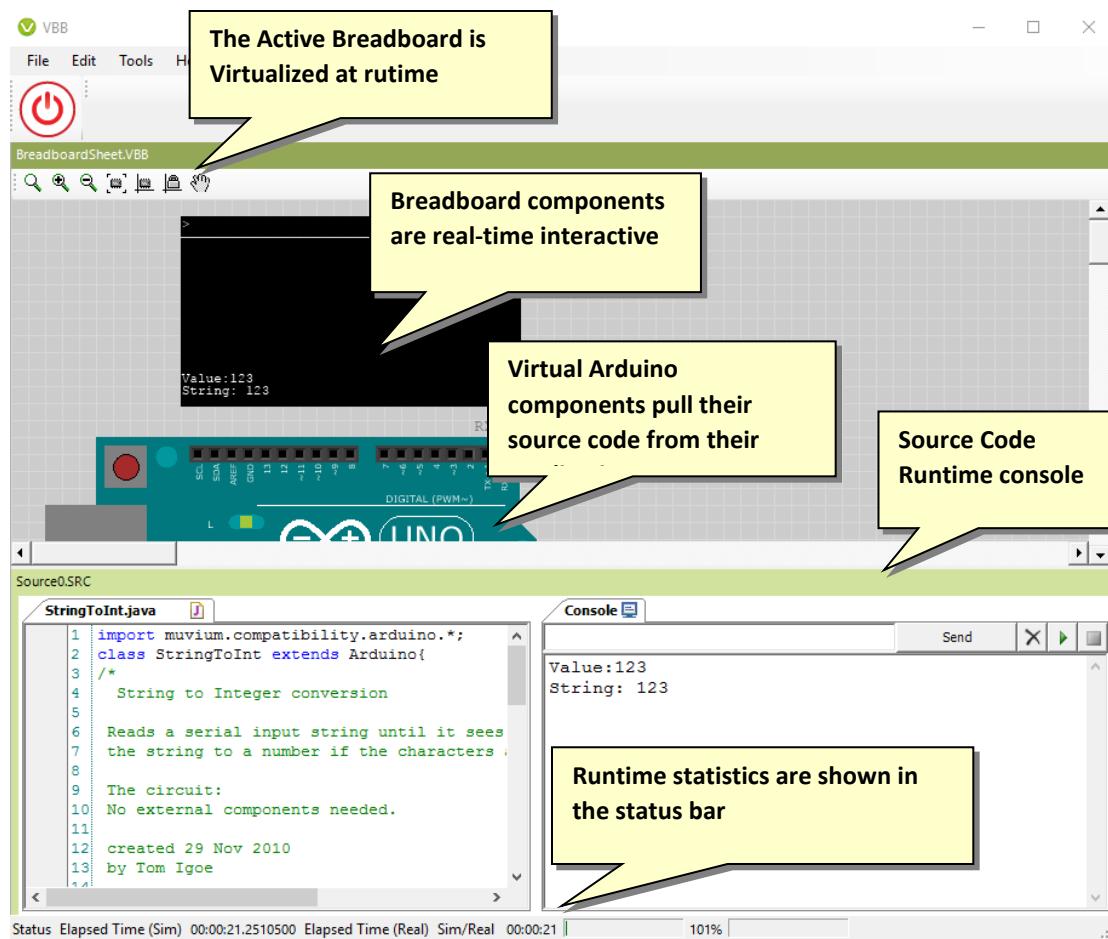
7.2.3 Status Toolbar

Shows status information including runtime timing and debug information

7.3 Runtime Virtualization

When a circuit is powered up it moves into Runtime mode where the circuit is virtualized and can be explored for testing or learning purposes. Clicking the PowerDown button stops virtualization and returns Vbb4Arduino to design mode.

If there is more than one Breadboard in the project the Active Breadboard is launched.



8 Design Sheets

DesignSheets are the main document managed by the VBB system. The Breadboard and LogicAnalyser design sheet are available to all projects. Licensing expansion modules add additional design sheets.

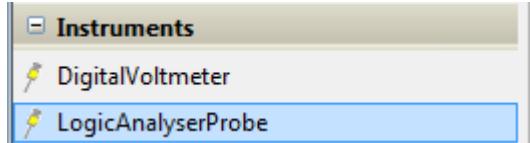
Icon	Module	Name	Ext	Description
------	--------	------	-----	-------------

	All	Logic Analyser	VLI	Virtual Multichannel Logic Analyser Instrument that is very useful for analysing logic signals for debugging and testing
	All	Breadboard	.VBB	The Breadboard is a DesignSheet supporting the VBB emulation framework. Components designed using VBB can be dropped into a Breadboard wired up and simulated
	Vbb4Arduino	Java Source Project	.SRC	The source code project is a special type of DesignSheet which contains a collection of java source code files

8.1 Logic Analyser

8.1.1 Trace Log (*.VLG)

The Logic Analyser traces signal events for logic analysis of circuits useful for debugging. The Logic Analyser works with Logic Probes that are placed onto Breadboard sheets and linked to the Logic Analyser Instrument from the Instrument property of the LogicProbe.

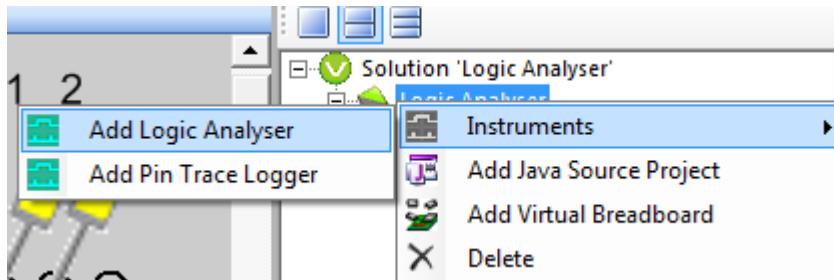


Design Sheet – The design sheet is a form which graphs signal events

Toolbar – None

Toolbox – None

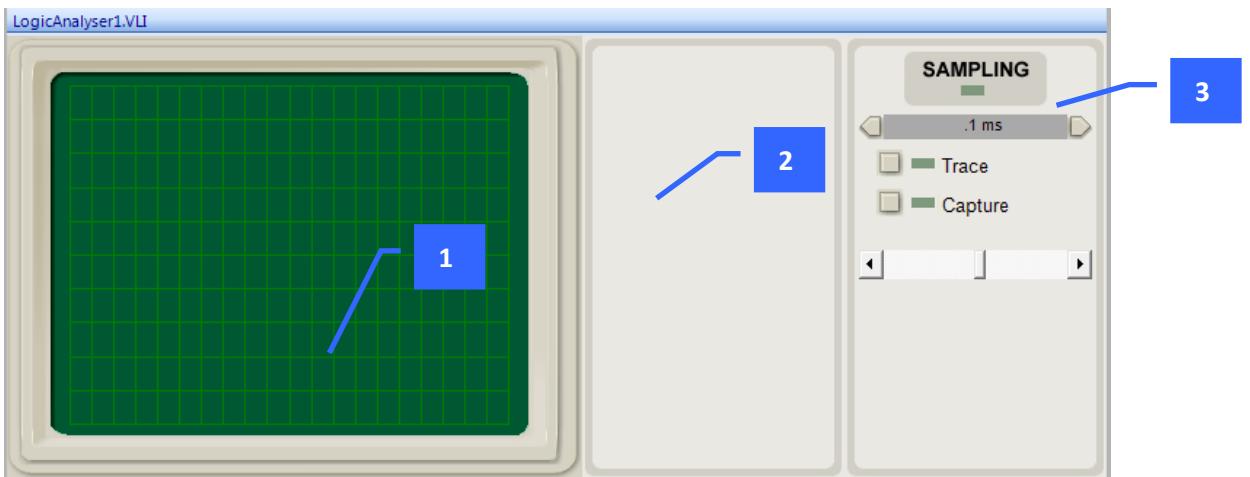
8.1.2 Adding a Logic Analyser



The trace log is found in the instruments project context sub menu. Right click on the project and select the Add Logic Analyser from the drop down context menu

8.1.3 Drag the Design Sheet into a View

The Logic Analyser doesn't have a default view location so wont appear until you drag it from the Project tree into a view pane

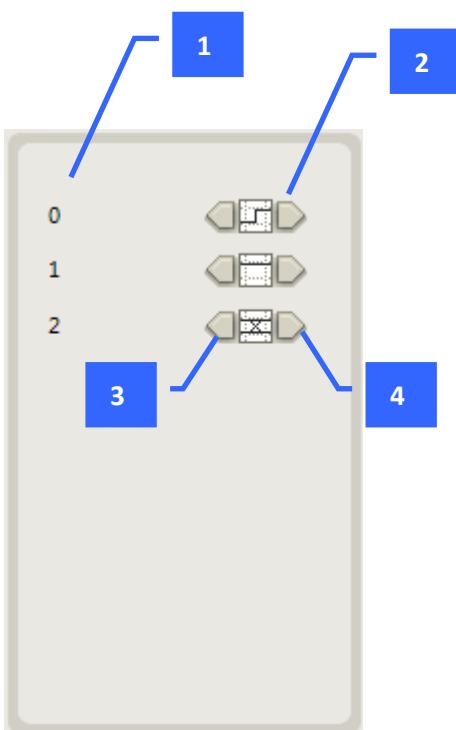


8.1.3.1 1. View Window

The trace grid is where the signal chart will be drawn. The probes sample the signal from the Breadboard sheet and are drawn into the TraceGrid. Up to 10 probes can be sampled

8.1.3.2 2. Probe List

The Probe list is initially empty when the circuit emulation is first run the probe list becomes populated with a probe list and trigger elements for each probe.



8.1.3.2.1 1 Probe name

The name of the probe is taken from the name property of the probe on the Breadboard sheet

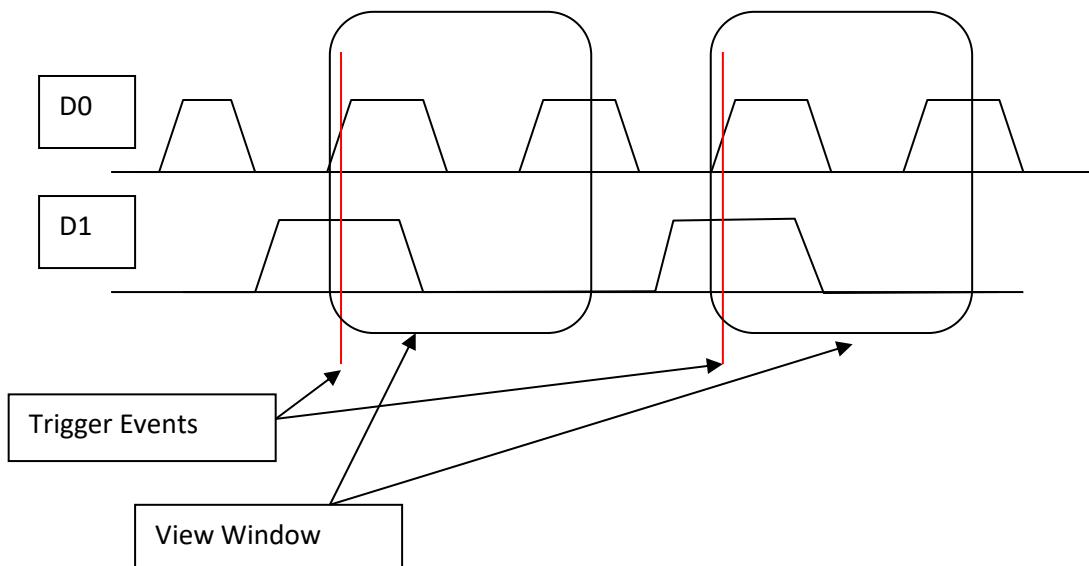
8.1.3.2.2 2. Trigger Element

The trigger element for each probe is combined to make a trigger event which is detected by the logic analyser to begin recording. This makes it possible to capture specific parts of a signal

There are 5 possible trigger options

Icon	Trigger	Description
	Don't-Care	Not used in the trigger detection
	Rising Edge	Trigger occurs when signal moves from low to high
	Falling Edge	Trigger occurs when signal moves from high to low
	HIGH	Trigger occurs when signal is HIGH.
	LOW	Trigger occurs when signal is LOW

A trigger is formed by AND the trigger elements together. For example the above trigger is ISRISING(0) AND ISHIGH(0). The left hand edge of the view window is marked by the trigger



3. Previous Filter Element

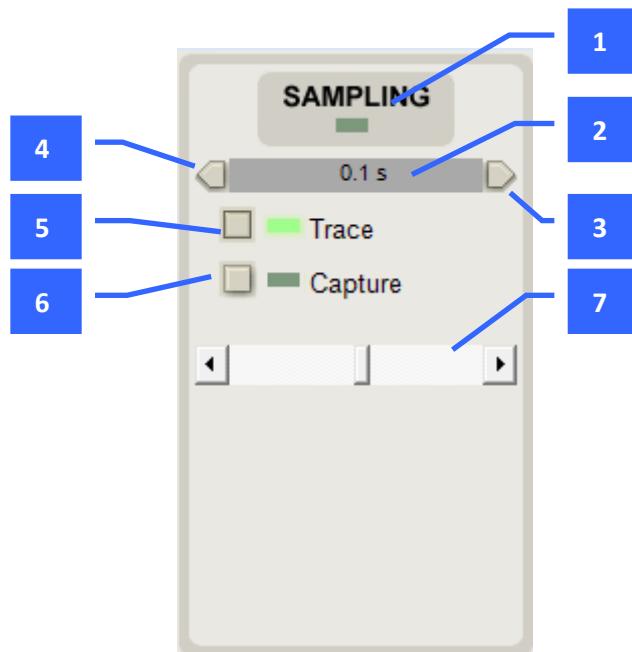
Click to switch to the previous filter element

4. Next Filter Element

Click to switch to the next filter element

8.1.4 3. Sampling Control Panel

The sampling control panels determines how signals are captured and viewed in the View Window

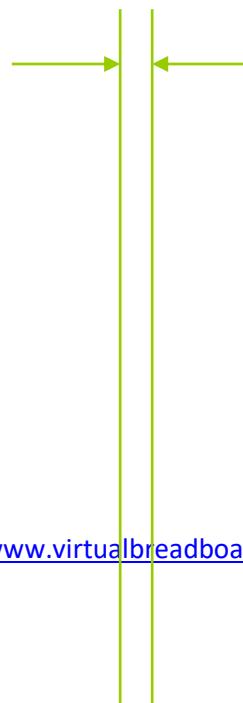


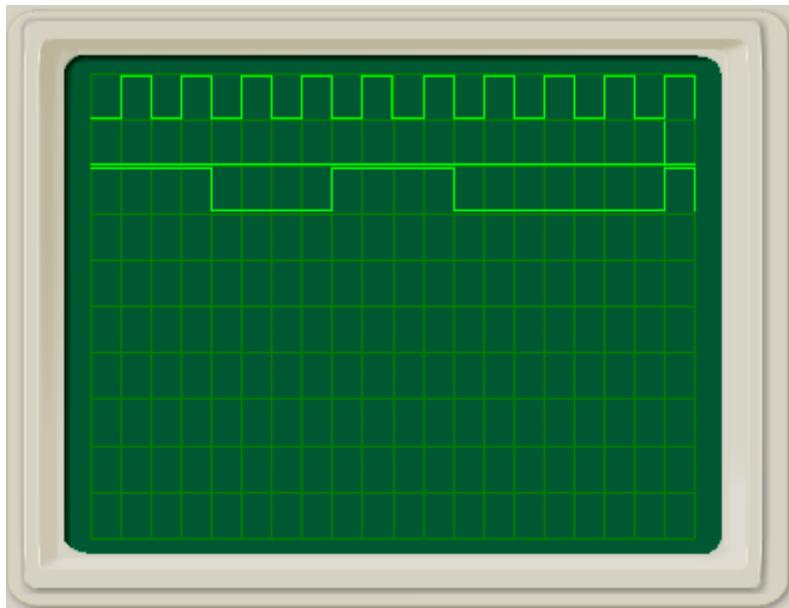
1. Sampling LED

The sampling LED flashes when a trigger event occurs and a sample is made

2. TimeBase

The Timebase is the displayed time per view per horizontal grid in the View Window from 0.1 microsecond through to 1 second.





8.1.4.1.1 3. Next Time Base

Decreases the timebase

8.1.4.1.2 4. Previous Time Base

Increases the timebase

8.1.4.1.3 5. Trace Mode Toggle Button

Click to enable trace mode. In trace mode the signal is sampled continuously and refreshes each time a trigger event occurs

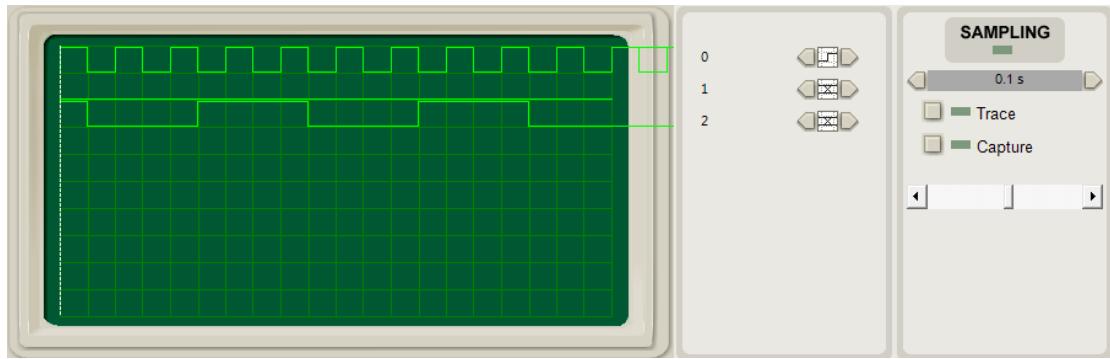
8.1.4.1.4 6. Capture Mode Toggle Button

Click to arm capture. In capture mode the first trigger event is captured and viewed. The capture mode button becomes disabled. You can then analyse the resulting signal trace without it changing. To sample another signal you need to arm the capture mode again.

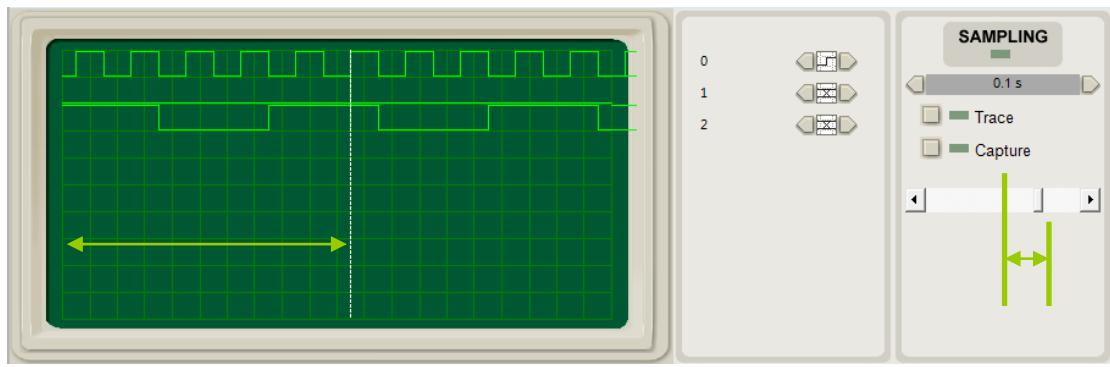
8.1.4.1.5 7. View Window Offset

Adds an offset to the view window left or right from the trigger event so you can view different parts of the signal in and around the trigger event

Offset = 0



Offset = ~+50% Shifts the left offset along to view the signal before the trigger



8.2 Breadboard DesignSheet

The Breadboard DesignSheet is a circuit emulation enabled drag and drop design sheet.

- DesignSheet – the DesignSheet a design graphic containing the component models
- Toolbar – the Breadboard toolbar is populated with graphical manipulation tools to zoom pan and manipulate the graphical design
- Toolbox – The Breadboard toolbox is populated with electronic component models which can be dragged and dropped on to Breadboard design

8.2.1 Toolbar : Breadboard

The graphics toolbar allows the manipulation of the Breadboard graphics, components and links.



Toolbar	Button	Description
Select Mode		Enters Select Mode. The cursor becomes an arrow. In select mode click on components to select them. Selecting a component populates the property box with the component properties.
Rotate Left		Rotate Left. Rotates a component 90 degrees left.
Rotate Right		Rotate Right. Rotates a component 90 degrees right.
Move Mode		Enters move Mode. The cursor becomes a NSEW pointer. In move mode you can drag and drop component to new location .
Link Mode		Enters link Mode. The cursor becomes a cross-hair and you can draw links between component pins.
Junction		Merges two links by joining with a junction.
Net		Merges links with the same net name
Zoom		Zoom Mode.

		Click the Zoom button to enter zoom mode. When over the Breadboard layout the cursor will change into a Zoom graphic. Click and hold the mouse down and then moving vertically up will zoom out and vertically down will zoom in. Stays in zoom mode until another graphic mode is selected
Zoom In		Zoom in by factor 2
Zoom out		Zoom out by factor $\frac{1}{2}$
Zoom region		Zoom to a region. <i>Not functional @v0.1</i>
Zoom Extents		Zoom to the extents. <i>Not functional @v0.1</i>
Restore Origin		Restore the origin to offset 0,0 and zoom =1
Lock origin		Lock the origin when zooming. When locked the offsets don't change only the zoom factor. When not locked the offset changes to keep the center of the screen fixed
Pan		Pan Mode. Click the Pan button to enter Pan mode. When over the Breadboard layout the cursor will change into a Hand Zoom graphic. Click and hold the mouse down and then drag the display to pan around. Stays in Pan mode until another graphic mode is selected
Grow		Grow the selected components by factor 2
Shrink		Shrink the selected components by factor $\frac{1}{2}$
Link Color		Set the link color. Sets the currently selected links to the selected color. Future links will be created with the new color.
Link Weight		Set the link width. Sets the currently selected links to the selected weight. Future links will be created with the new weight.
Show Nets		Show the virtual nets between named nets in the circuit board.

8.2.2 Note on Zoom factor:

The snap to grid has a problem when the zoom factor is not a multiple of 2. You should not try to select links or draw links at arbitrary zoom factors. The zoom, zoom extents, zoom region are best used to inspect the design and select components but you should restore the origin and use the zoom in, zoom out when drawing links to ensure the zoom is a multiple of 2.

8.2.3 Placing a component from the Toolbox

Placing a component is an important skill – not quite drag and drop. You need to click on the Toolbox icon releasing the mouse button. This attaches the component to the mouse pointer when you move over the Breadboard design. You can then drag the component into position in the DesignSheet and click a second time to place the component.

8.2.4 Component Editing

There are several ways you can work with components.

- Placing a component
- Select a component
- Select a group of components
- Append a component to a selection
- Move a group of selected components
- Copy and paste selected components
- Delete selected components

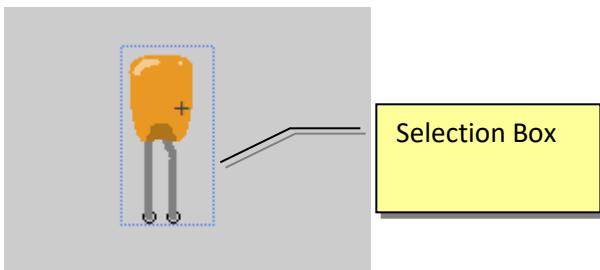
8.2.5 Placing a component from the Toolbox

Placing a component is an important skill – not quite drag and drop. You need to click on the Toolbox icon releasing the mouse button. This attaches the component to the mouse pointer when you move over the DesignSheet. You can then drag the component into position in the DesignSheet and click a second time to place the component.

8.2.6 Select a component

To select a component enter Select Mode  and left click on the component

Selected components are bounded by a dashed blue line.

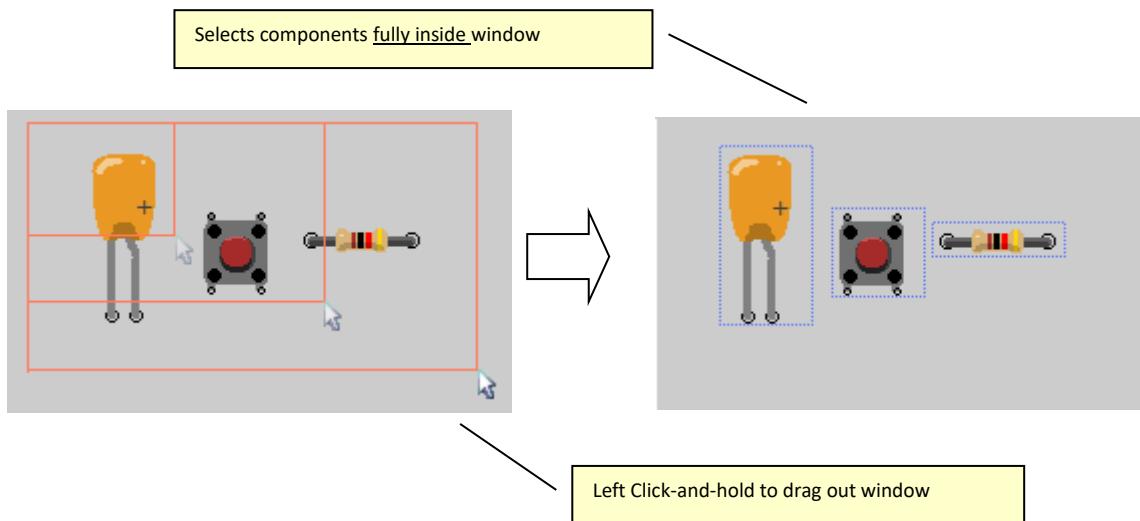


8.2.7 Select a group of components

Select a group by drawing a window around the components to select.

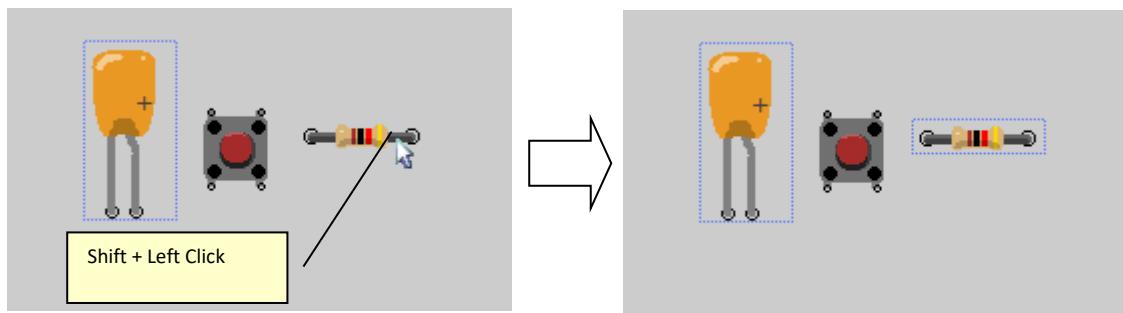
Draw a window with a Left Click to anchor the window, and holding down the mouse button drag out a window.

Release the mouse to select the components fully inside the window.



8.2.8 Append a component to a selection

Append a component to a selection by holding shift and left clicking the component.

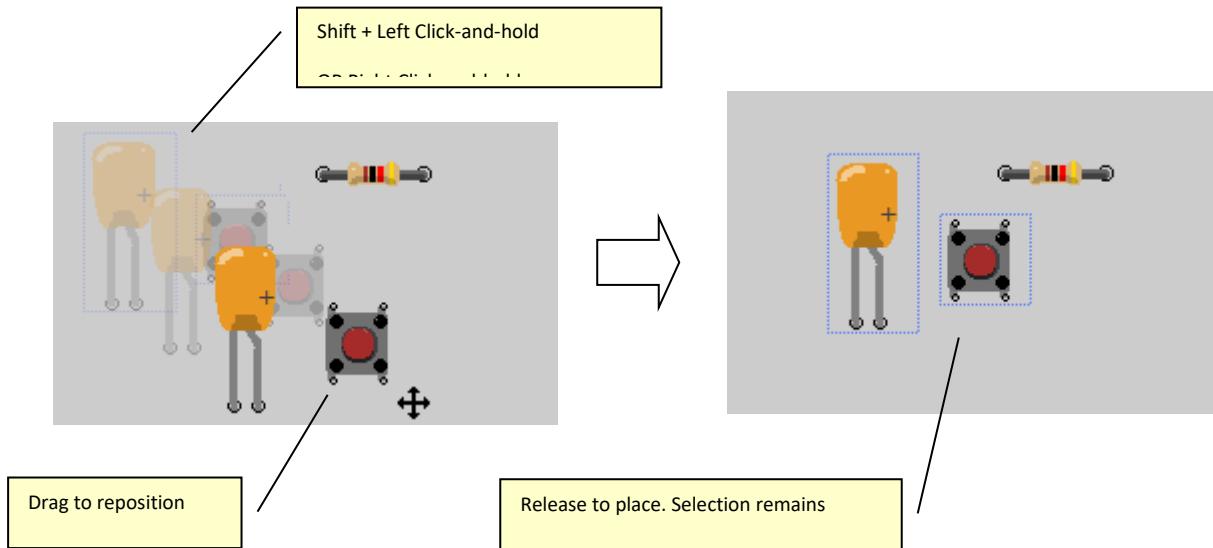


8.2.9 Move a group of selected components

To move a selected group of components

Select Move Mode by clicking the move state button

Shift + Left Button drag and drop or Right button drag and drop.

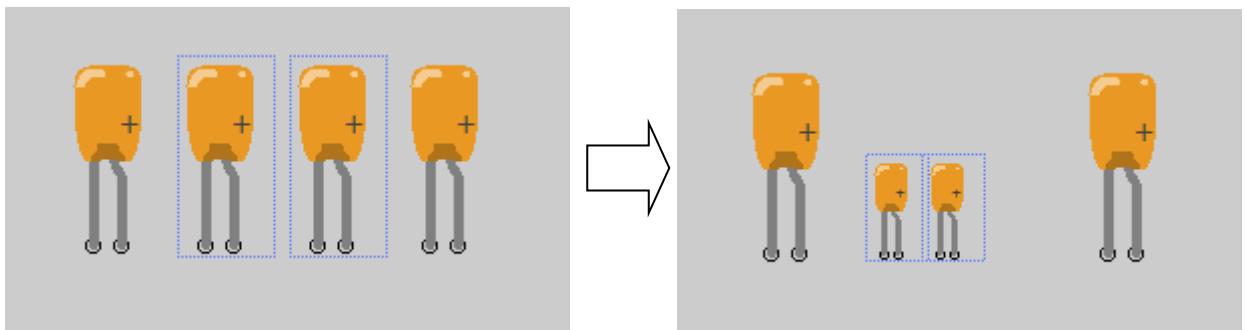


8.2.10 Shrink and grow a selection

To increase density of certain parts of a graphical design it can be useful to scale down a component or group of components. Equally it is useful to be able to restore the size of the components and scale components back up.

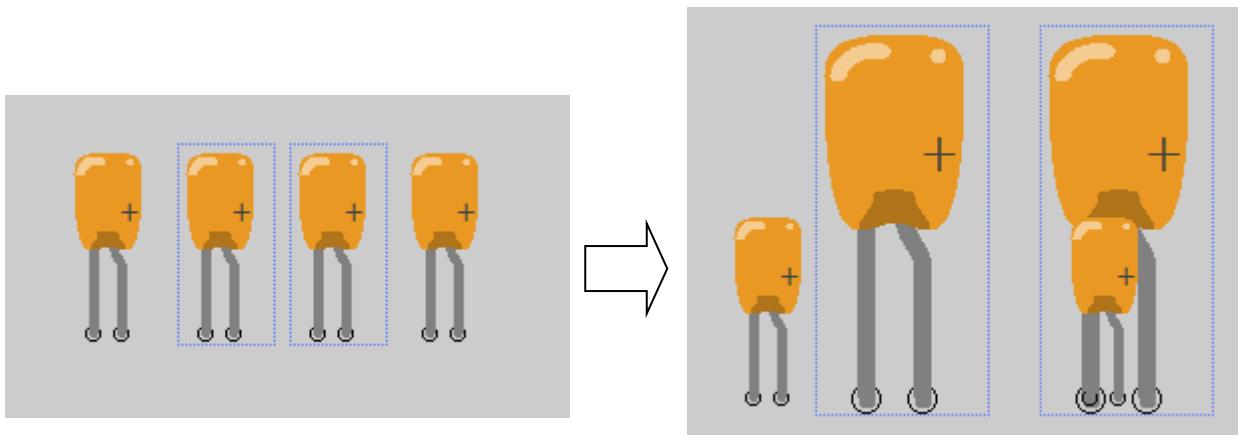
8.2.10.1 To scale down a component selection

- Click the scale down component toolbar button 



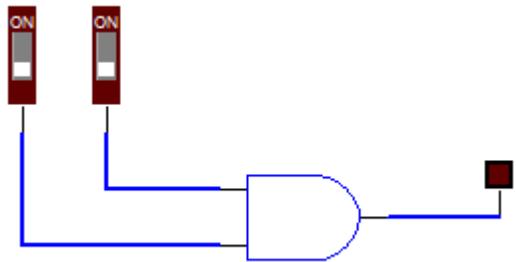
8.2.10.2 To scale up a component selection

- Click the scale up component toolbar button 



8.2.11 Wiring Essentials

Wires are links from component pin to component pin.

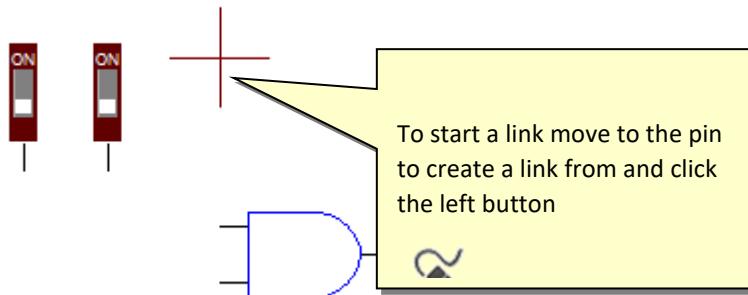


To begin wiring click the wiring button

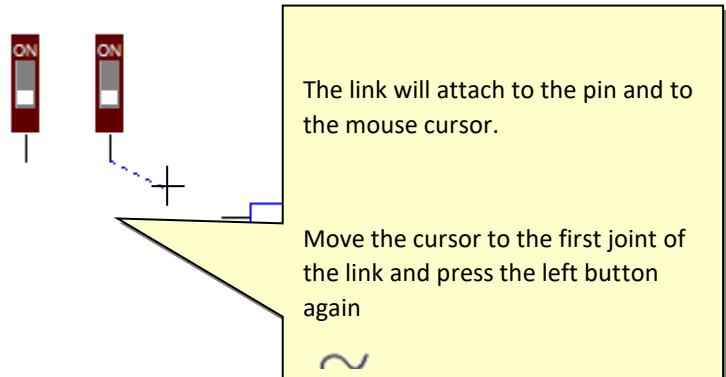
The cursor will become a crosshair



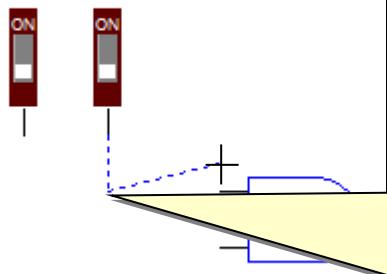
Breadboard0.VBB



Breadboard0.VBB



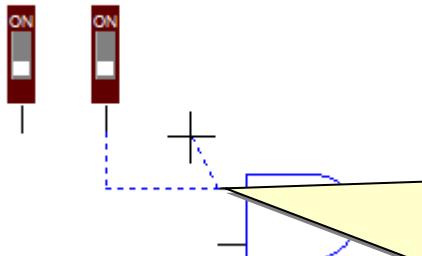
Breadboard0.VBB



Now the link is anchored to the last joint and the cursor

Click again with the left mouse on the next joint location or the destination pin **or Double Click to finish the pin**

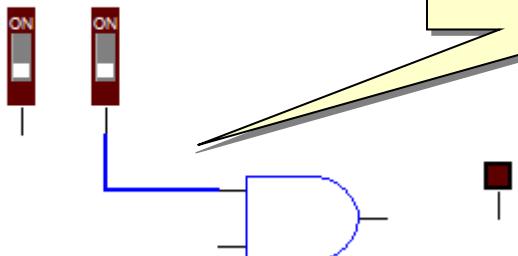
Breadboard0.VBB



Now the link is anchored to the destination pin and the cursor

To finish the link right mouse button or double click

Breadboard0.VBB



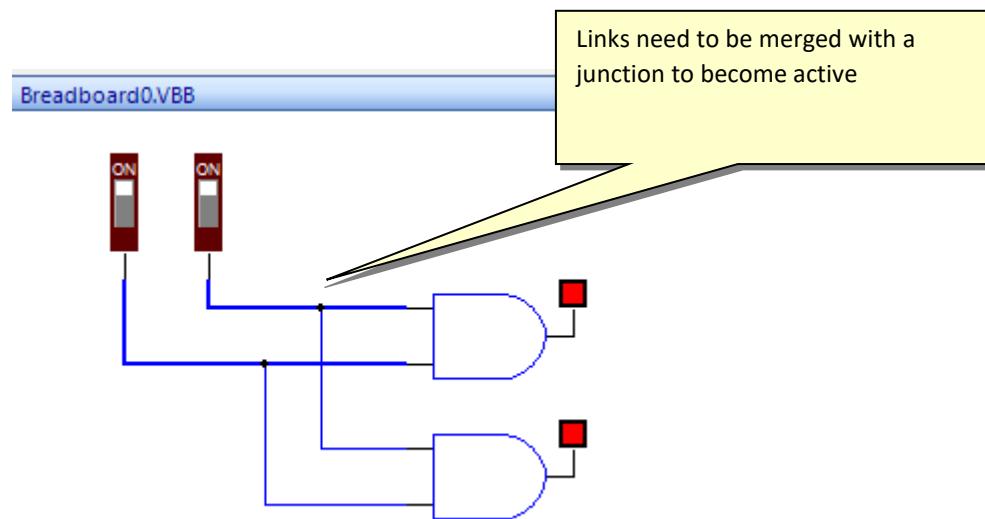
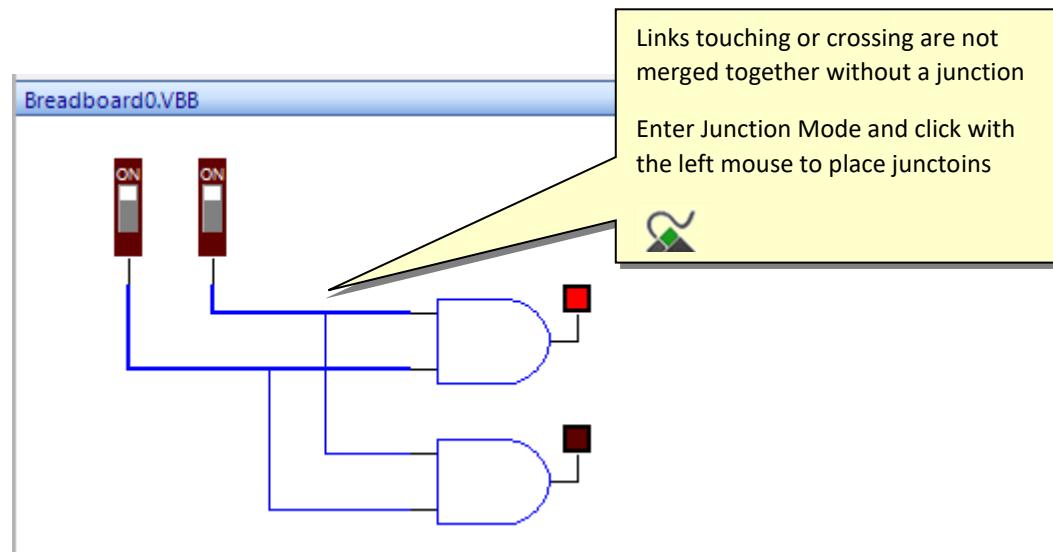
When a link is made between two pins the link becomes thicker to indicate the link is an active link

8.2.11.1 Working with Junctions

Junctions are used to merge links together.

Note: Links overlayed over each other are not considered a single link

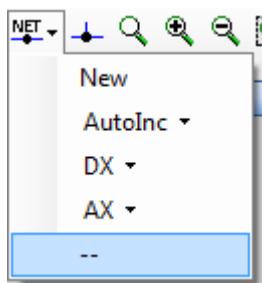
To create a junction click the junction mode  and then



8.2.11.2 Working with Net Labels

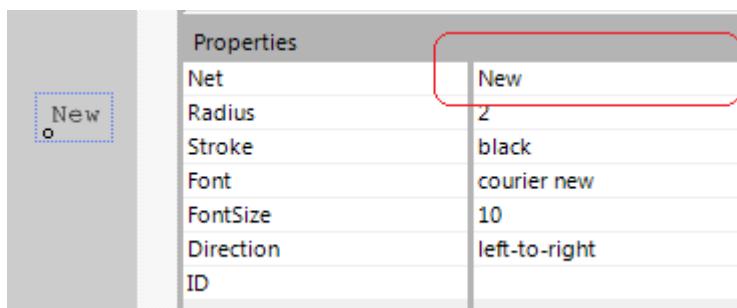
Net Labels allow connections to be made between pins by using a name instead of a wire. This can be used to better organise the connection layout.

The Net Toolbar option is used to place nets and features some helper options to add nets



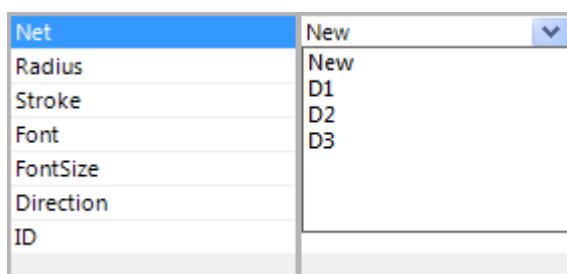
8.2.11.2.1 New

New is the standard option to add a new net. When you add a new net it is placed with the Name new. You can edit the name by clicking the net name and editing the name in the property box



Drop Down List of Named nets

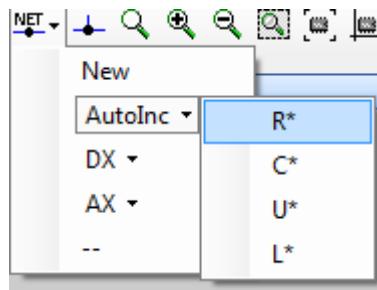
You can also select a net from the drop-down list in the property box. The dropdown list contains the names of all the nets currently on the Breadboard.



8.2.11.2.2 Autolnc

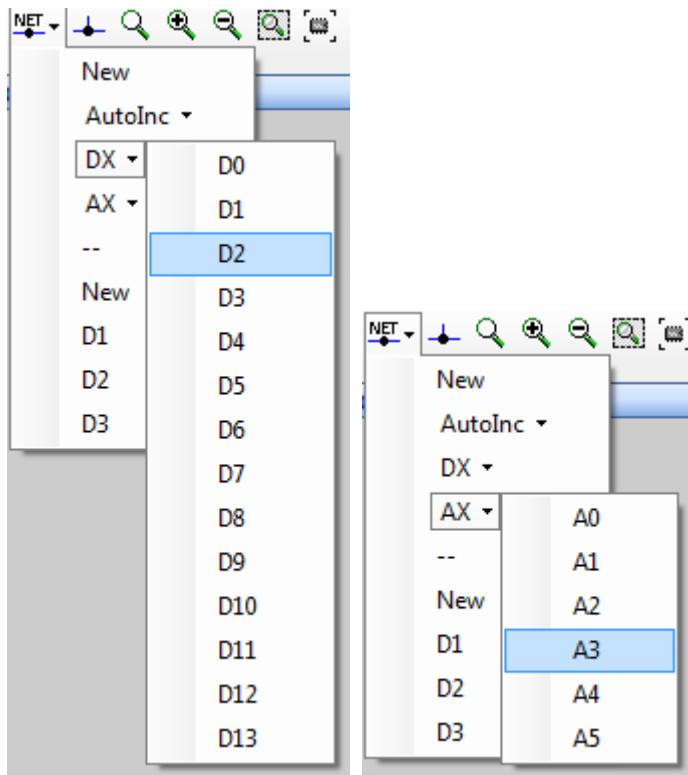
The AutoInc dropdown option has a further sub dropdown menu which allows you to select a net using the common prefixes (R = Resistor, C = Capacitor, L = Inductor , U = Integrated Circuit)

When you select from the submenu the net will be named +1 to the current highest net name. So for example if you have R1,R2,R2 on your breadboard already and you select R* then the net will be named R3.



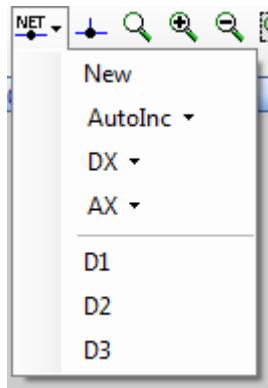
8.2.11.2.3 DX / AX

The V18'0, Arduino and others have a naming convention for digital pins of D0, D1, D2 .. and Analog Pin A0,A1,.. These nets are also automatically added when using the V18'0, Arduino components so these are very commonly used net names. The AX and DX submenus gives you a shortcut for selecting these names.



8.2.12 All Nets (--)

All the current nets on the Breadboard are listed below the – toolbar separator giving a shortcut to select a matching net for a net to be placed.

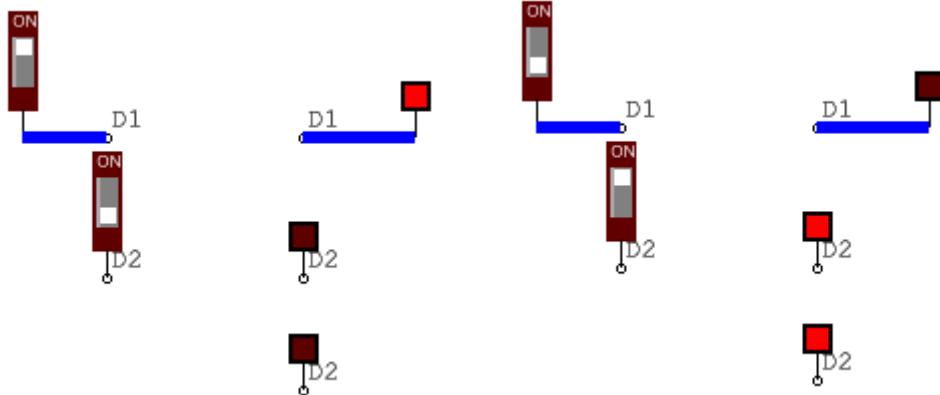


8.2.13 Virtual Links

Nets can be thought of as acting a virtual link between the pins that are connected by the Net

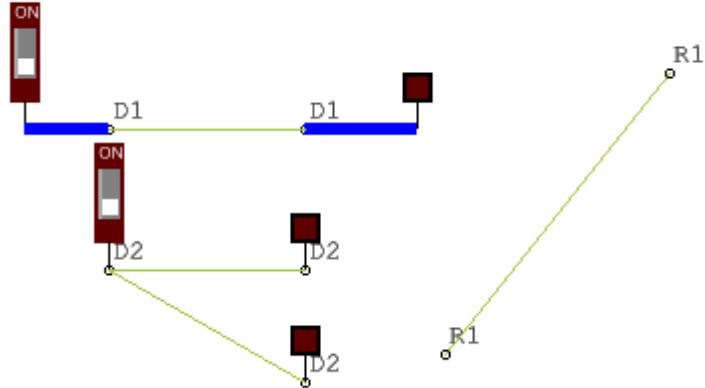
Nets can be

- Connected to the end of links or midway along links like junctions
- Placed over pins directly



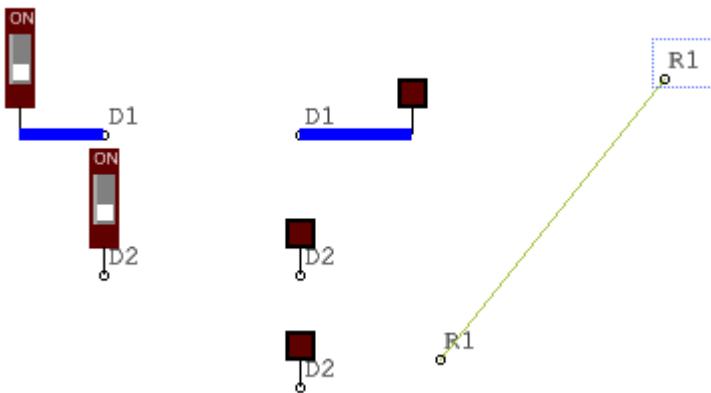
8.2.14 Show Links

You can view the links between the nets using the ShowLinks button. When you click the button it will show the links. It is a state button and when you unselect the button it will hide the links again.



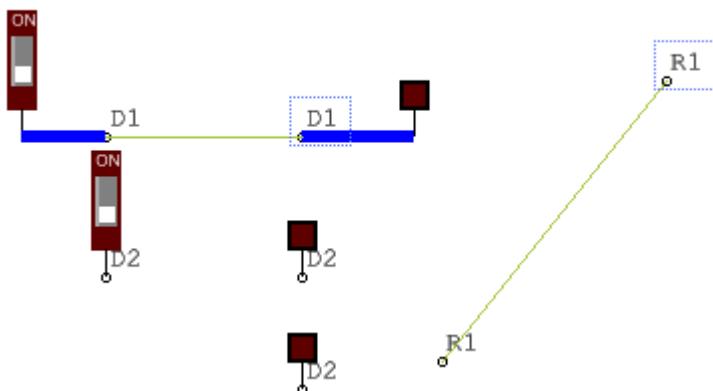
8.2.14.1.1 Showing specific Net Links

To show a specific net link you can select a member of the net and click to ShowLinks button



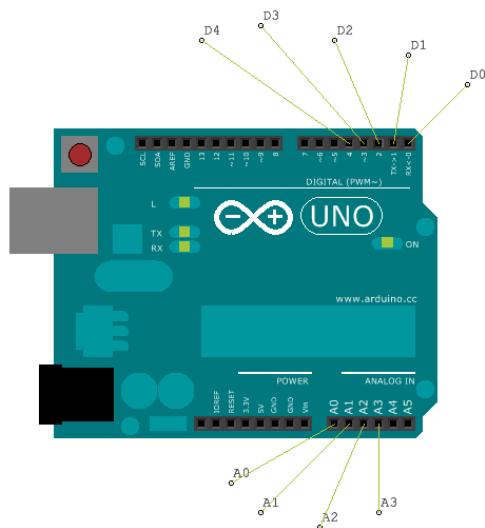
8.2.14.1.2 Show subgroups of links

If you select members of more than one net then the links belonging to all the selected nets will be shown.

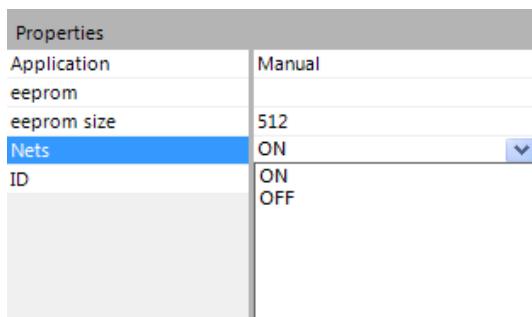


8.2.14.1.3 Predefined Nets

Arduino footprint boards have predefined nets associated with the pins



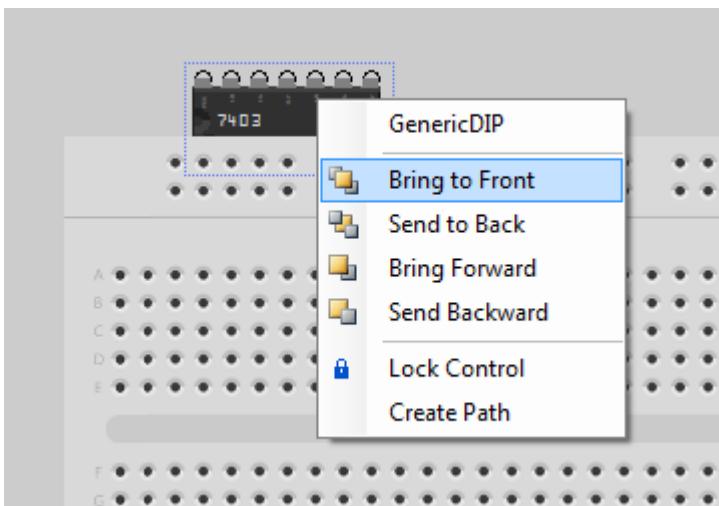
By default the nets for these are on but you can switch off the default nets using the components Nets property



8.2.14.2 Component Layering

Components are layered , you move components up and down the layering using the context menu.

Select the component and press right click to start the context menu.



8.2.14.3 Locking a Component

The context menu contains an option for locking components. Select the component, right click to bring up the context menu and click Lock Control to lock the component. A locked component cannot be accidentally moved.

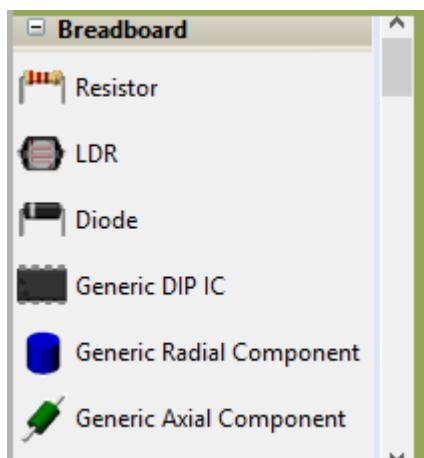
8.3 Toolbox : Breadboard

The Breadboard Toolbox is the most important Toolbox to VBB as it contains the component models that constitute the circuits



8.3.1 Toolbox Groups

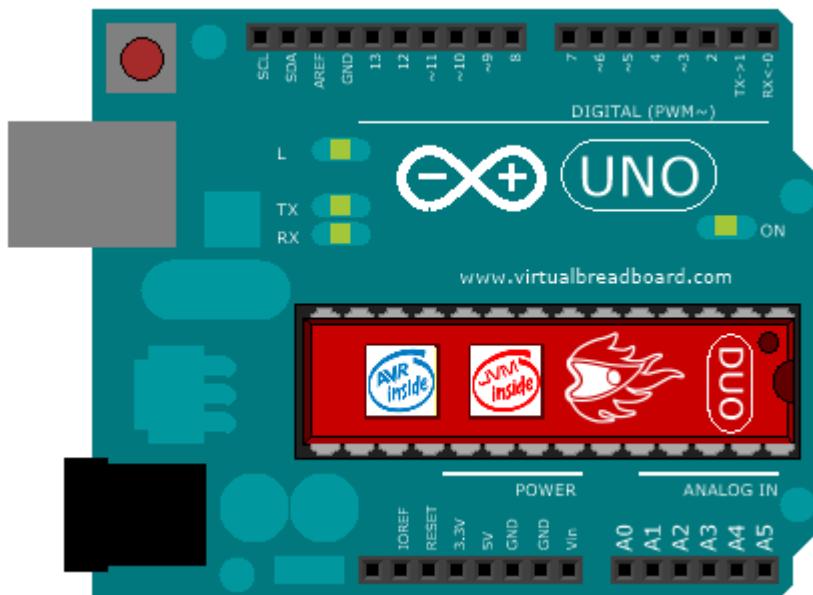
Components are grouped into function



9 Micro Controller Components

9.1 Arduino Family (with DUO)

The DUO is an Arduino UNO R3 compatible board with the socketed ATmega328p microcontroller replaced with the DUO module which also contains a ATmega328p but also contains a PIC18F46K22 implementing a Java Virtual Machine (JVM) device wired as the master to the AVR slave. The DUO is dual boot device which will transparently connect the ATmega328p when used with the Arduino IDE but the JVM will be configured as the master when programmed in Java mode from VBB.



The Arduino with DUO is enabled with the VBB4Arduino license.

See [VBB Vbb4Arduino Documentation](#)

9.2 AVR ATMEGA328p

The Arduino with DUO is enabled with the VBB4Arduino license.

See [**VBB Vbb4Arduino Extension** Documentation](#)

9.3 PicMicro 12/16/18 Family

The PicMicro with DUO is enabled with the PicMicro license.

See [**VBB PicMicro Extension** Documentation](#)

9.1 Firmata Arduino UNO Host

A remotely controlled microcontroller enabled with the Firmata license.

See **VBB Firmata Extension** Documentation

9.2 ICEShield Arduino UNO Host

An hardware remoted interface enabled by using the ICEShield hardware with remote physical microcontroller .

See **VBB ICEShield Extension** Documentation

10 Generic Layout Components

Layout Components are generic graphics only components and will not emulate. However when used in combination with a function block they can appear to emulate. These components are parametric models of the common hole-through components. Layout components help you create a circuit board which can be fabricated by our PCB service or as a guide when assembling a real circuit board even where specific components models are not available.

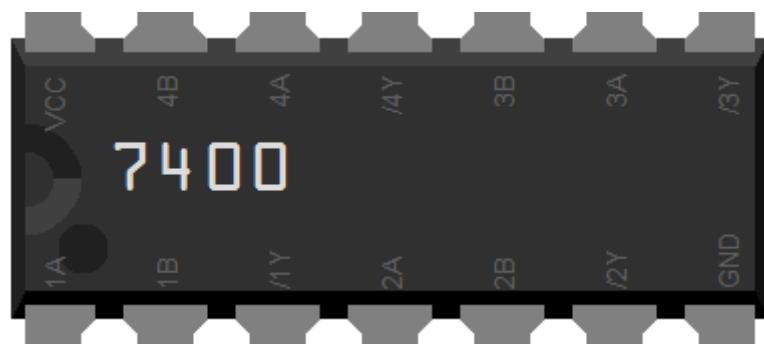
10.1.1 Generic Layout Only

- Generic DIP
- Axial
- Radial

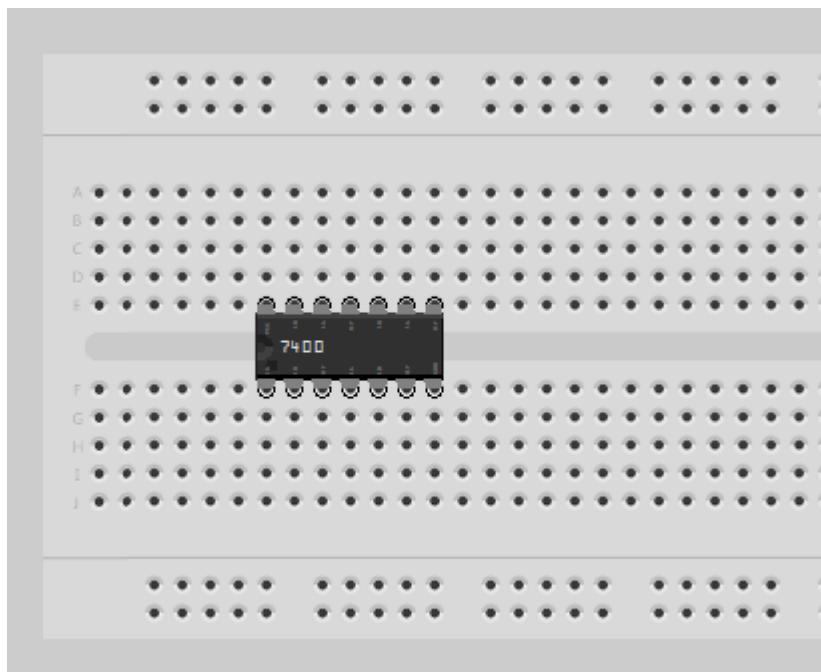
10.2 Generic DIP



Breadboards are useful because they have pin sockets that match the spacing of the standard DIP packaging of electronic circuits. The Generic DIP component allows you to easily create a DIP component package.

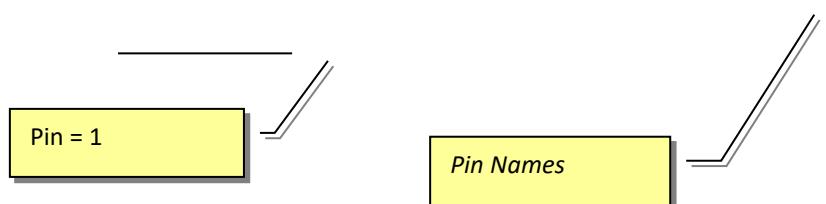
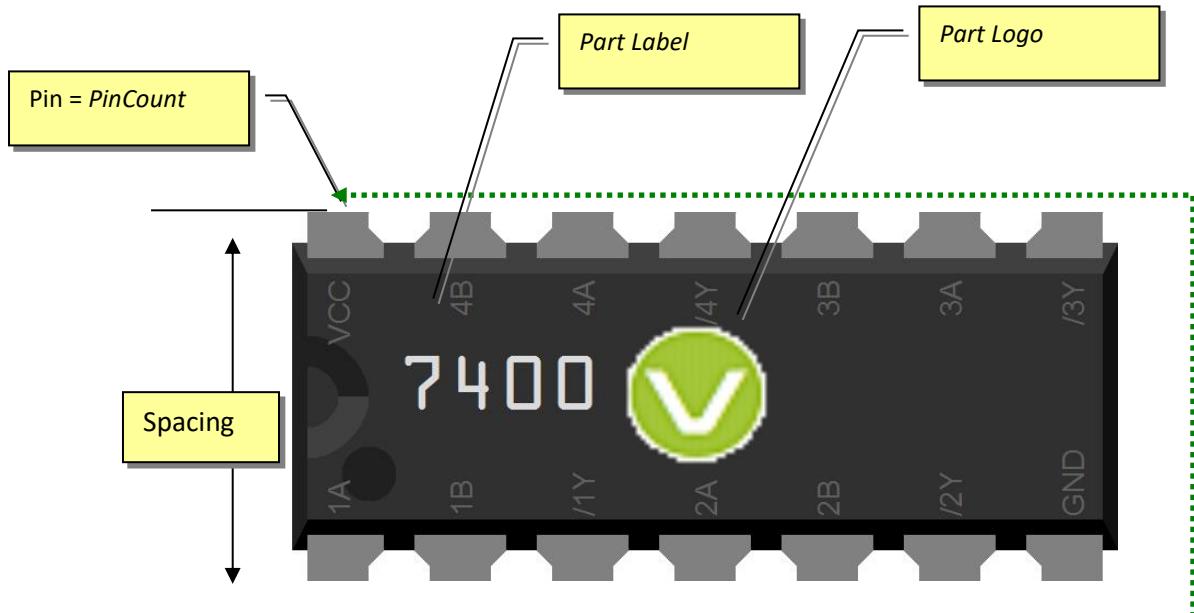


DIPs components are typically placed across the center section.



10.3 Parametric

The DIP component is parametric so you can create any DIP by setting the component properties accordingly.



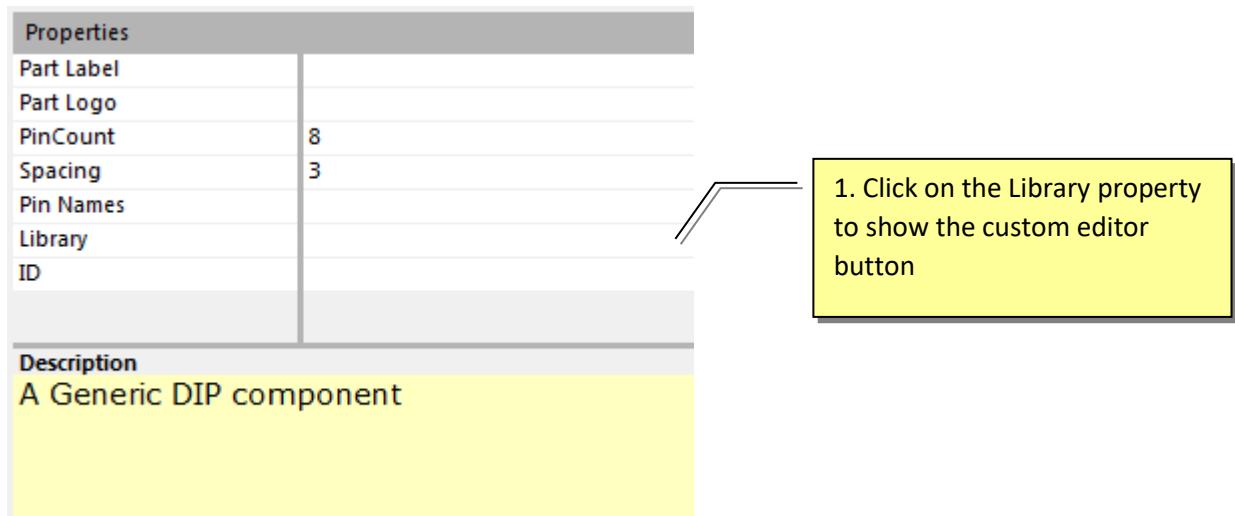
Library

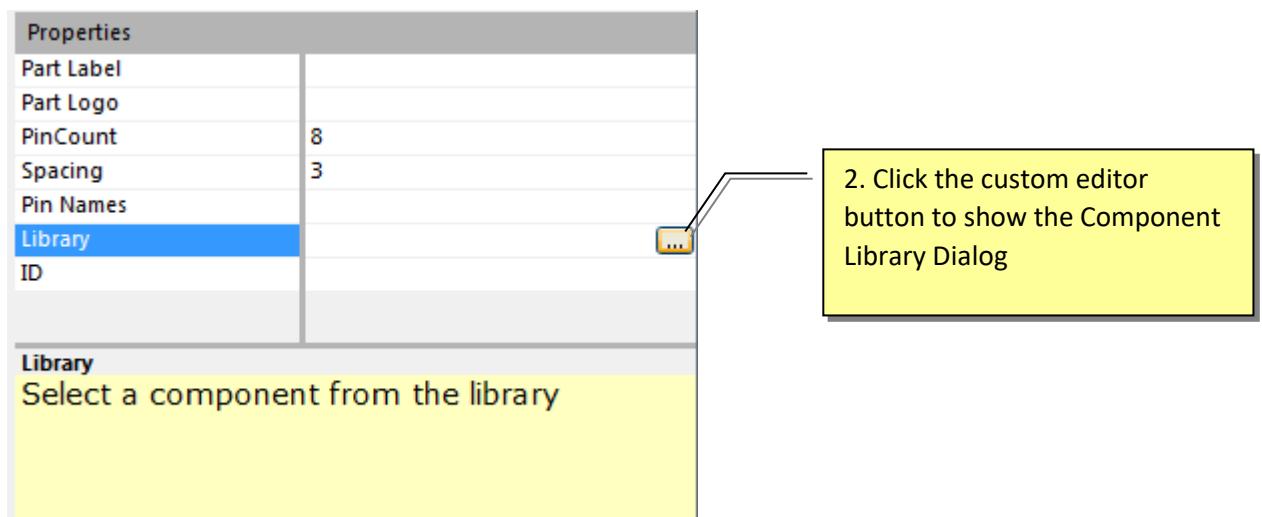
Property	Description
ID	Identifier used in BOM
Part Label	The part name shown on the packaging

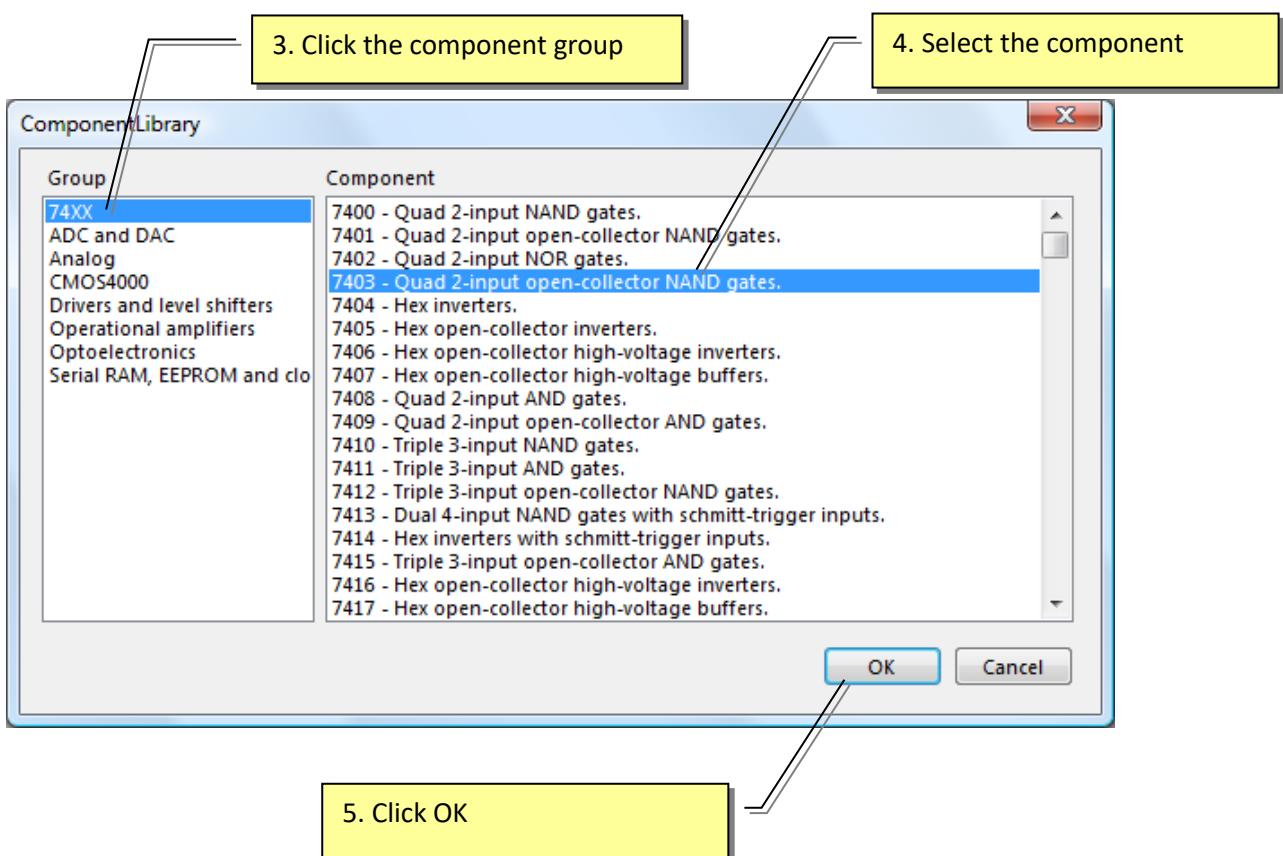
Part Logo	An optional logo that is shown center of the package. Uses magenta for transparent color
PinCount	The number of pins (multiple of 2)
Spacing	The size of the spacing between pins – in hundreds of mil – typically 3 or 5
Pin Names	A CSV(comma separated array) of names starting from pin 1 to pin pin pinCount of the pins labels. This is shown as an overlay as a form of schematic for the part
Library	The Library is a special property which actives as Custom Dialog to select a component from a library which then sets the properties

10.3.1 Using the Library Component

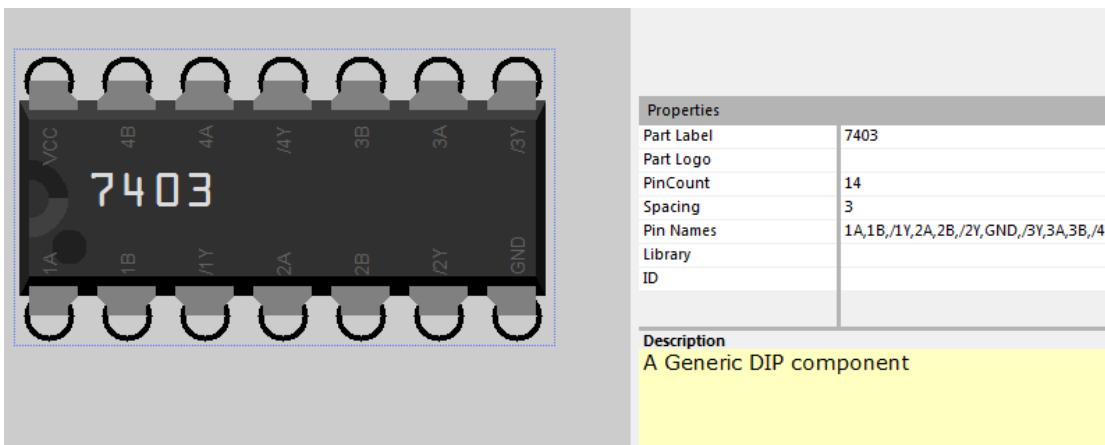
When you first place a Generic DIP the properties are blank. You can enter your own values or select from the library. To use the library







This sets the properties for the component creating the component.



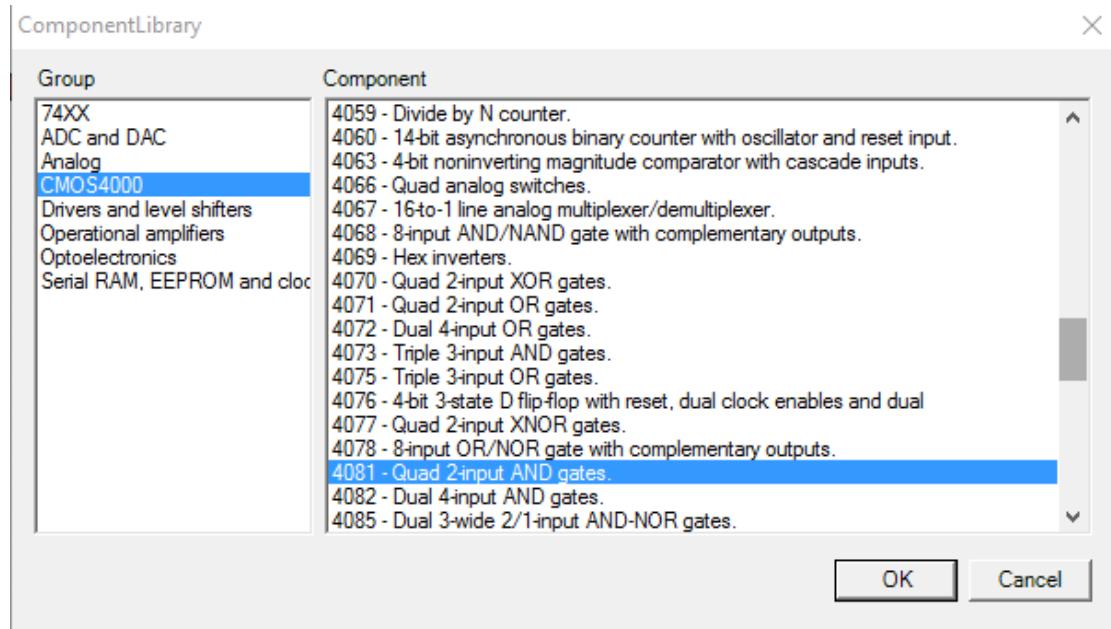
10.4 Marking up a generic component with a function block

You can ‘add virtualization’ to a generic component footprint by marking it up with one or more function blocks.

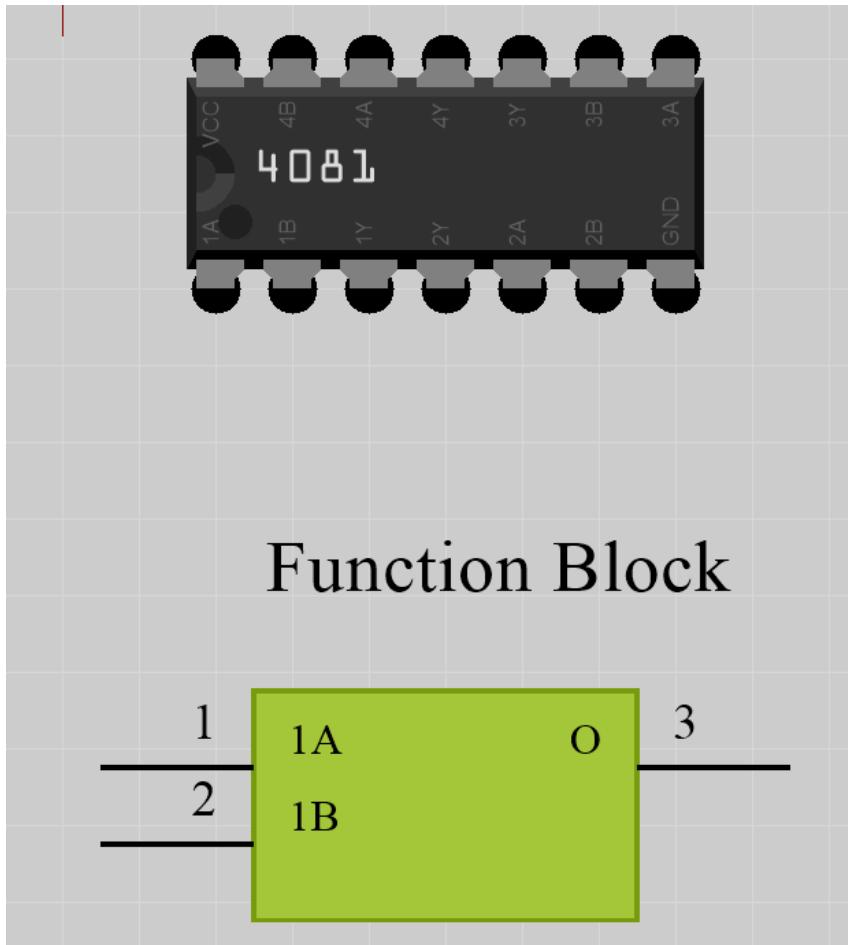
For example you might want to virtualize a dual AND gate

1. Place a suitable generic component

Select a 4081 Quad 2-input AND gate using the generic library



2. Place a Function block and give it 2 inputs



Referring to the 4081 datasheet or by inspection of the pins on the top of the DIP component we want to assign the function block the pin logic of the 4081

So Function Block Output 1Y - = AND(1A,1B)

The easiest way to do this is to use the netlist. From the properties of the 4081 copy the pin names over to the netlist field to assign pin names to the known pin positions

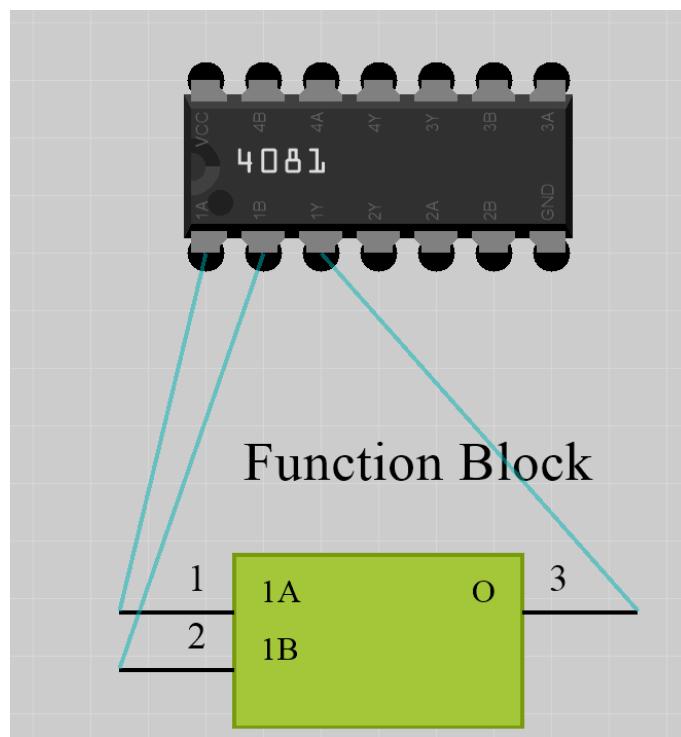
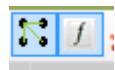
Properties	
Part Label	4081
Part Logo	
PinCount	14
Spacing	3
Pin Names	1A,1B,1Y,2Y,2A,2B,GND,3A,3B,3Y,4Y,4A,4B,VCC,
Library	
ID	
netlist	1A,1B,1Y,2Y,2A,2B,GND,3A,3B,3Y,4Y,4A,4B,VCC,

Now in the function block properties copy over the relevant net names to the match the input and outputs. Conveniently in this example they are aligned at pin 1,2,3 so it's just a matter of copying over the first 3 net names from the 4081 to the function block

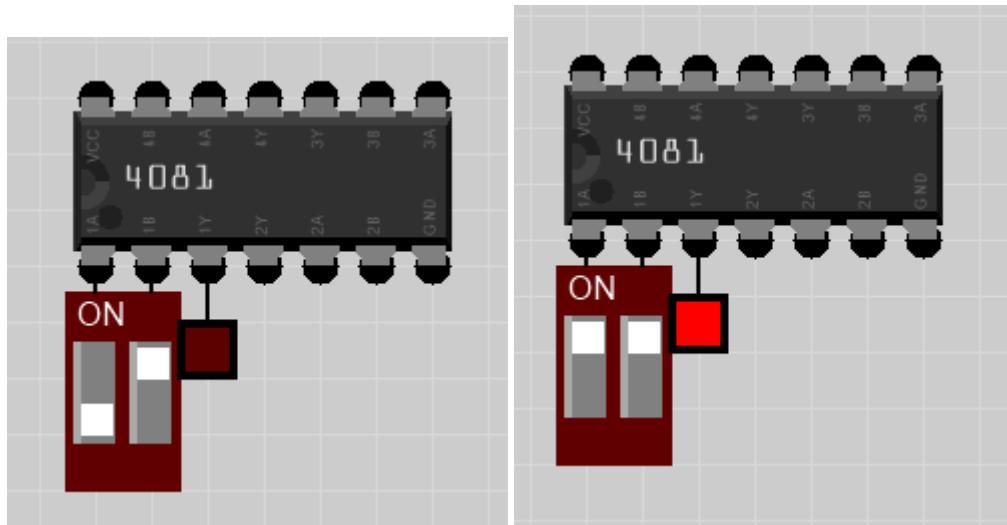
Properties	
Name	Function Block
Inputs	2
Expression	AND(P1,P2)
ID	
netlist	1A,1B,1Y

Finally assign the Expression AND(P1,P2) to create an AND gate from pins 1 and 2 with the output on pin 3, which is how the 4081 is specified

You can preview the virtual links using the show links and functionblocks options

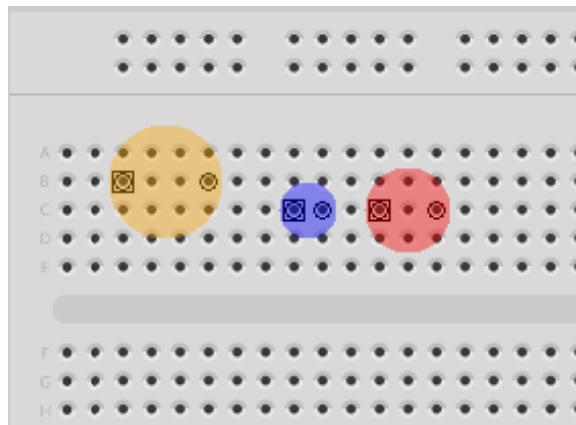


At runtime you can hide the links and function block to make it appear the 4081 generic component footprint is virtualized

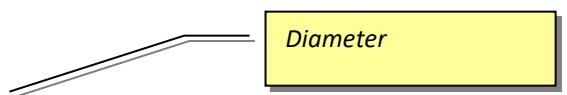


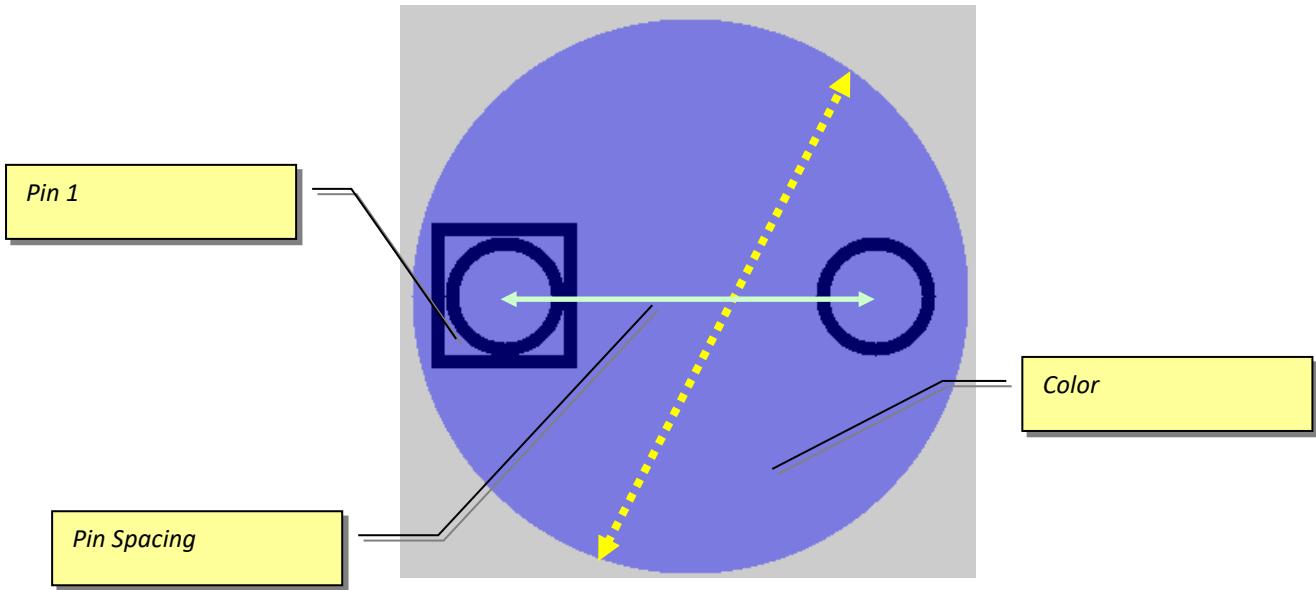
10.5 Radial Component

Generic Radial component can be used to model components with a 2 pin Radial configuration for which there is no matching visual component.



10.5.1 Parametric Model



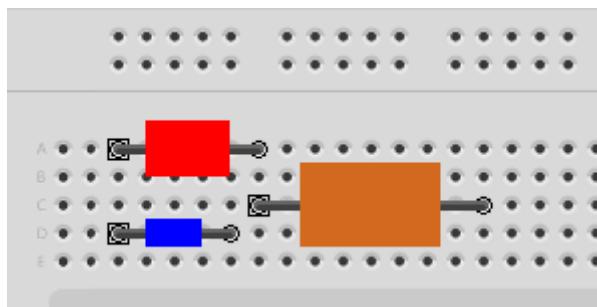


10.5.2 Properties

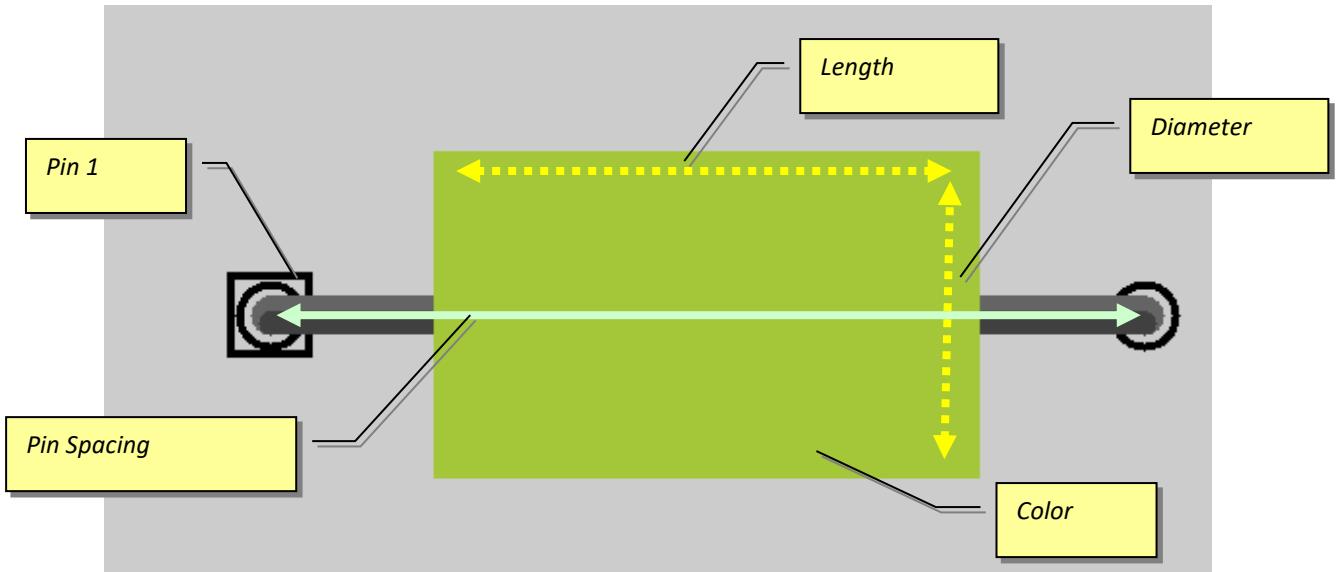
Property	Description
ID	Identifier used in BOM
Spacing	The number of grid spacings between pins
Diameter	The diameter of the radial component in grid spacings (0.1")
Color	The color of the Radial packaging
Value	The value of the part to appear in the BOM. Eg Capacitance or special part no.

10.6 Axial Component

Generic Axial component can be used to model components with a 2 pin Axial configuration for which there is no matching visual component.



10.6.1 Parametric Model

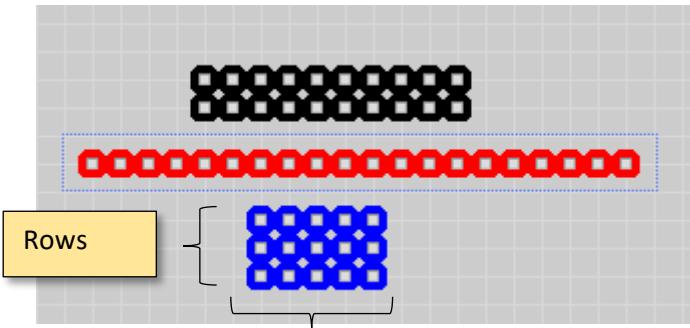


10.6.2 Properties

Property	Description
ID	Identifier used in BOM
Spacing	The number of grid spacings between pins
Diameter	The diameter of the axial component in grid spacings (0.1")
Length	The length of the axial component in grid spacings (0.1")
Color	The color of the Radial packaging
Value	The value of the part to appear in the BOM. Eg Capacitance or special part no.

10.1 Generic Header Component

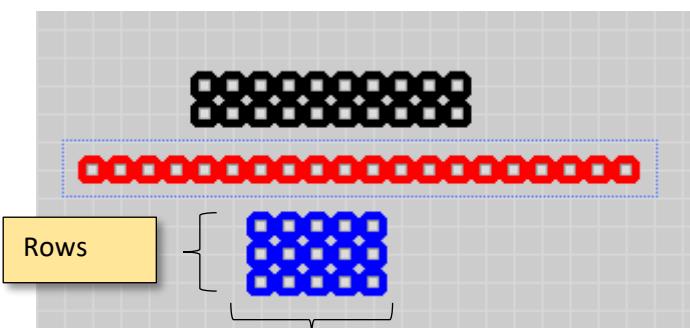
Generic Radial component can be used to model all different header configurations.



Name	Default	O Pins	Description
Pins	2	2 3 4 8 User Enter	The number of horizontal pins
Rows	1	1 2	The number of rows of the header block
Color	Black	Black Blue Red Named Color	Size of the LED array

10.1 Generic Male Header

Generic male header component can be used to model many different header configurations.

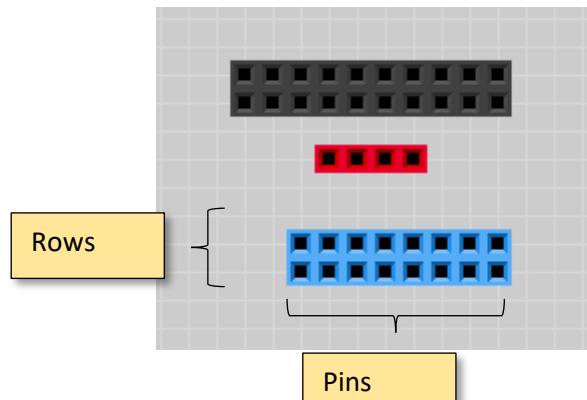


Name	Default	O Pins	Description

Pins	2	2 3 4 8 User Enter	The number of horizontal pins
Rows	1	1 2	The number of rows of the header block
Color	Black	Black Blue Red Named Color	Color of packaging. Black, Blue, Red or Name User Color – See Appendix

10.1 Generic Female Header

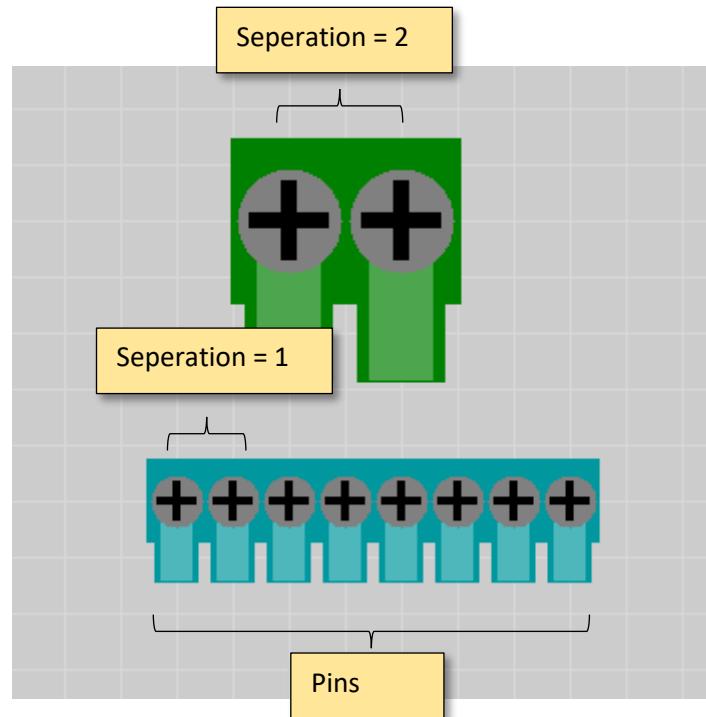
Generic male header component can be used to model many different header configurations.



Name	Default	Options	Description
Pins	2	2 3 4 8 User Enter	The number of horizontal pins
Rows	1	1 2	The number of rows of the header block
Color	Black	Black Blue Red	Color of the housing. Black Blue Red only.

10.1 Screw Terminal

Generic male header component can be used to model many different header configurations.



Name	Default	Options	Description
Pins	2	2 3 4 8 User Enter	The number of horizontal pins
Separation	.1	0.1 0.2	The separation or the pins, 1 or 2 slots. You can also use scale to increase the size of the
Color	Green	Names Color	Named Color of the packaging. See the Appendix

11 Circuit Model Components

VBB has some special circuit model components which can be used to model or ‘mock’ many different types of circuits relevant to the Arduino user.

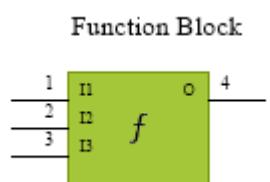
- **Function Blocks** are programmable logic blocks that can overlay circuits to mock circuit behaviour
- **Resistors** are modelled either as a fuse or an open-circuit
- **Capacitors** are modelled as an open-circuit.
- **Diodes** are modelled as switched pull-up or switched pull-down resistor.
- **NPN Transistors** are modelled as a switch which turns on when the base voltage is HIGH
- **PNP Transistors** are modelled as a switch which turns on when the base voltage is LOW

11.1 Function Block

The Function Block is a powerful circuit modelling block and can be used in Co-Virtualization of many different circuits.

When the function block is used in combination with generic layouts there is special support to hide the function block to make it look like the circuit block is virtualizing. It’s a type of component development kit.

The purpose of the Function Block component is to model certain circuit behaviour not directly supported by VBB.



Property	Description
Name	The Name of the Function Block for circuit documentation
Inputs	The number of inputs. When this value changes the Function Block diagram will automatically resize.
Expression	The FunctionBlock Expression.
ID	The device ID in the BOM list

netlist	A CSV list of nets I1,I2..,IN,O which map to the input pin array and output pin. Use the netlist to add FunctionBlocks without wiring
---------	---

11.1.1 The Function Block expression

The function block expression calculates an output voltage based on it's input pins. It works in a similar way to an Excel function.

$$V_{out} = f(P_1, P_2, \dots, P_n)$$

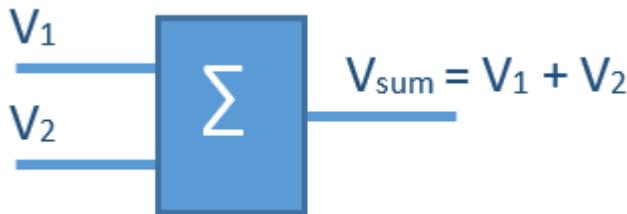
For example

- ADD(P1,P2) '*Voltage Summer*'
- MUL(SUB(P1,P2),2) '*Differential Voltage Multiplier*'

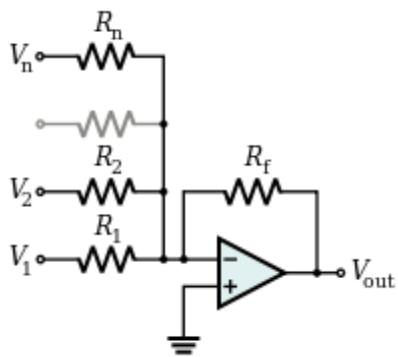
When used the Function Block graphic will change to reflect the names for the input pins

11.1.1.1 Summer Concept Example

A voltage summer is a circuit function that sums two or more voltage inputs.

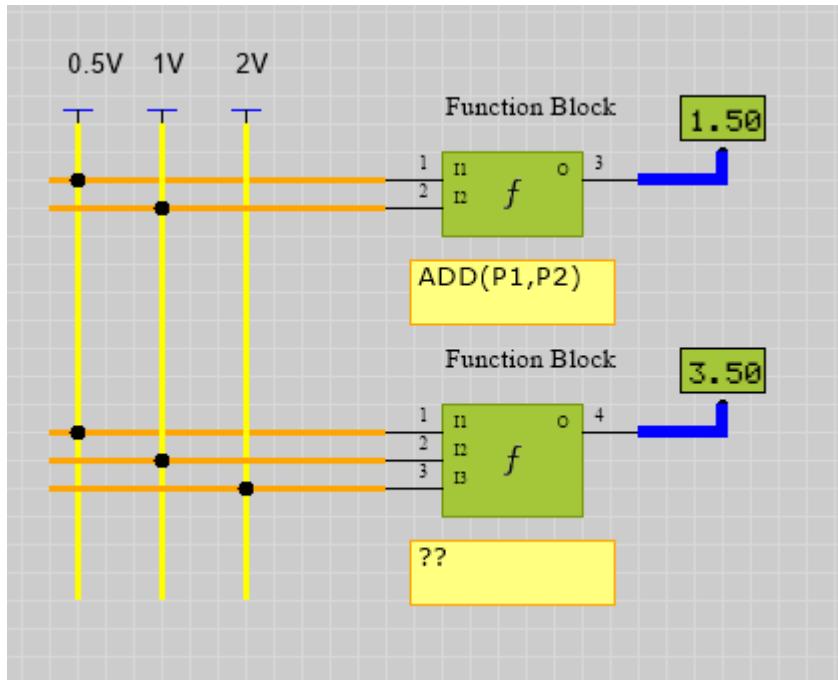


This is a common circuit block often implemented with operational amplifiers



However there are other ways to sum voltages such as by using digital sampling.

The focus of VBB is on *equivalent circuits* so it doesn't matter how the circuit implements a voltage adder VBB can model it using a function block.



To create a voltage summer you use the **ADD function** in the function block expression.

You could use a function block alongside Generic components in a VBB circuit and it would work the same and look the same as if it's behaviour we evolved from the discrete circuit elements the way SPICE does it but the end result is the same.

11.2 Function Block Expression

11.3 Function Block Expression Grammar

```

Expression = Primitive | Function
Primitive ::= Numeric | Pn | PINn
Function ::= functionName( expression [, expression]* )
functionName ::= CONST | RISING | FALLING | ISHIGH | ISLOW | PULSIN |
CINT | NAND | NOR | XOR | XOR | MIN | MAX | ABSDIFF | CSIGN | CEN |
SIGN | PWM | NEG | IF | MUL | ADD | SUB | RANGE | MAP | OR | NOT | GT |
LT | VOLTS | DAC
  
```

Function Block Expressions

FunctionName::=Pin Functions | Logic Functions |Math Functions | Special Functions

Pin Functions ::= ISHIGH | ISLOW | RISING | FALLING

Logic Functions ::= AND, OR, XOR, NOT, NAND, NOR, XNOR

Math Functions ::= ABSDIFF, ADD, CEN, CONST, CSIGN, DIV, GT, IF, LT, MAP, MIN, MUL, NEG, RANGE, SIGN, SUB

Special Functions::= DAC, PULSIN,PWM,VOLTS

Expressions can often contain other functions as parameters. For example:

AND(OR(P1,P2),OR(P3,P4) *Complex Logic*

ADD(P1,P2) 'Voltage Summer

MUL(SUB(P1,P2),2)' Differential amplifier gain = 2

Pins

PN return a value 0 for logic level GND and 1 for Vdd which is nominally 5V

Pin Functions:

ISHIGH (PIN) : Returns 1 if the PIN is HIGH else returns 0

ISLOW(PIN) : Returns 1 if the PIN is LOW else returns 0

RISING(PIN) : Returns 1 if the PIN has a rising edge

FALLING(PINS) : Returns 1 if the PIN has a falling edge

Logic Functions:

AND(F1,F2,...,F3): Returns 1 if all FN return 1

OR(F1,F2,...,FN): Returns 1 if any FN return 1

XOR(F1,F2,...,FN): Returns 1 if equal numbers of FN are 1 and 0

NAND(F1,F2,...,F3): Returns 1 if any FN is 0

NOR(F1,F2,...,FN): Returns 1 if no FN is 1

XNOR(F1,F2,...,FN): Returns 1 if odd numbers of FN are 1 and 0

NOT(F1): Return 1 if F1 is 0

Math Functions:

ABSDIFF(FA,FB): Evaluates a $|FB-FB|$

ADD(FA,FB,..,FN): Evaluates $FA+FB+\dots+FN$

CEN(FA) : Evaluates $|FA-FB|-0.5$

CINT(FA): Converts floating FA to integer

CONST(string) : Evaluates string as numeric

CSIGN(FA): if FA > 0.5 returns 1 else returns 0

DIV(FA,FB,..,FN) : Evaluates $FA\div FB\div\dots\div FN$

GT(FA, FB): If FA > FB returns 1 else returns 0

IF(FA, FB, FC): If FA != 0 returns RB else returns FC

LT(FA, FB): If FA < FB returns 1 else returns 0

MAP(FA, FB, FC): Evaluates $FA-FBFC-FB$ clipped to range 0 to 1

MIN(FA,FB,..,FN) : returns the lowest value in the parameter list FA,.. ,FN

MAX(FA,FB,..,FN) : returns the highest value in the parameter list FA,.. ,FN

MUL(FA,FB,..,FN) : Evaluates $FA*FB*\dots*FN$

NEG(FA) : Evaluates $-FA$

RANGE(FA, FB, FC) : Evaluates $FA+(FB-FA)*FC$

SIGN(FA, FB) : if(FA > FB) returns 1 else returns 0

SUB(FA,FB,..,FN) : Evaluates $FA-FB-\dots-FN$

Special Functions

DAC(FA,FB,..FN): Digital To Analog converter using N bits to create analog range between 0 and 1

PULSIN(FA): Measure the time in seconds of a pulse where the time is measured between the rising and falling edge

PWM (FA) : Digital to Analog converter using the PWM pulse duty on FA to create an analog value between 0 and 1

PWM(FA,FB) : Digital to Analog converter using the PWM pulse duty on Differential FA - FB to create an analog value between 0 and 1

VOLTS (FA) : Converts the result of FA into a volts value by dividing by the nominal voltage value of 5.

11.3.1 Function Block Editor Dialog

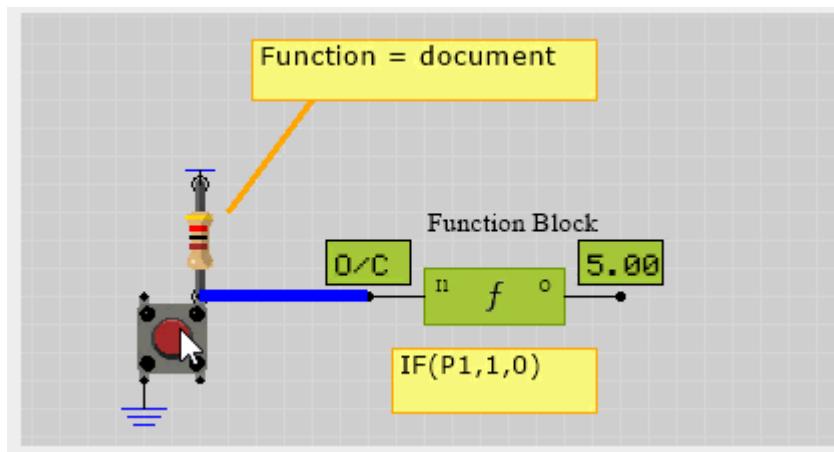
The function block editor dialog is a

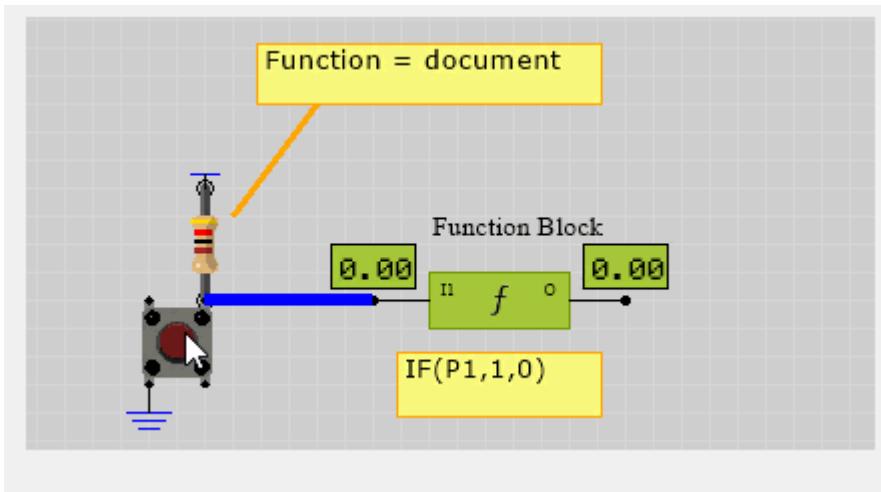
11.4 Function Block Usage

Pullup Resistor

A pullup resistor is a very common circuit block very often used to create voltage dividers or convert button presses into high/low values.

For example when a momentary button which is connected to ground is connect to a resistor is connected to





The first trick is to set the resistor in *open-circuit* mode. This means the resistor is just a picture and its value is not used in the resulting circuit behaviour.

The output of the momentary button is the input to the function block and it has two values. O/C *open-circuit* when the switch is off and 0.0 *ground* when the switch is on.

To model the O/C open-circuit needs to be converted to 1 = HIGH = 5V

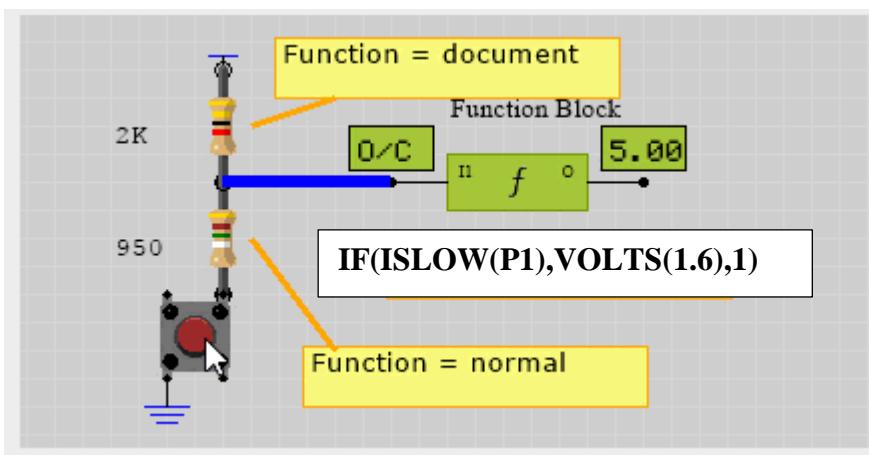
$$\text{Pullup resistor} := \text{IF}(\text{ISLOW}(P1), 0, 1) = \text{NOT}(\text{ISLOW}(P1))$$

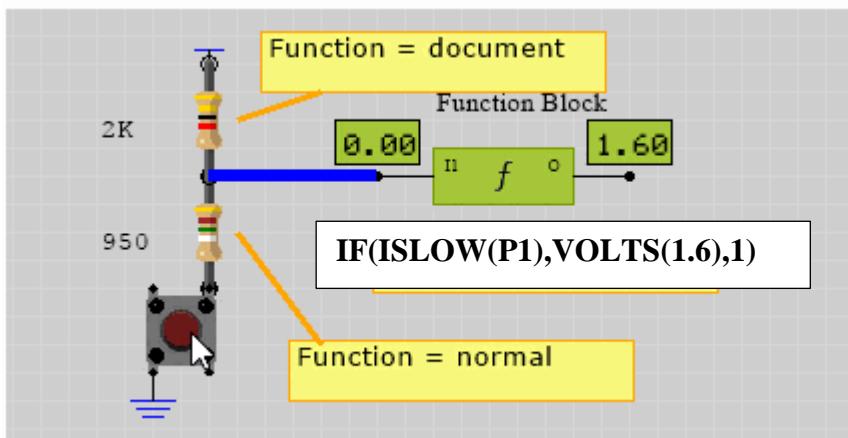
`ISLOW(P1)` returns 1 when the voltage at P1 is LOW = 0.0 and otherwise it returns 1. However the output of the function block needs to invert this which can be done with an NOT or IF

11.4.1 Dynamic Voltage Divider

A voltage divider is a common circuit configuration. Consider a voltage divider which is tapped between two resistors and momentary switch connected to ground on the bottom of the ladder. The voltage divider is has the well known formula

$$V_{out} = \frac{R_2}{R_1+R_2} \cdot V_{in}$$





So for resistors 2K and 950Ohm with 5V rail we can derive the output we expect when the button is pressed to be $V = (950) (2K + 950) \times 5 \approx 1.6V$

IF(ISLOW(P1),VOLTS(1.6),1)

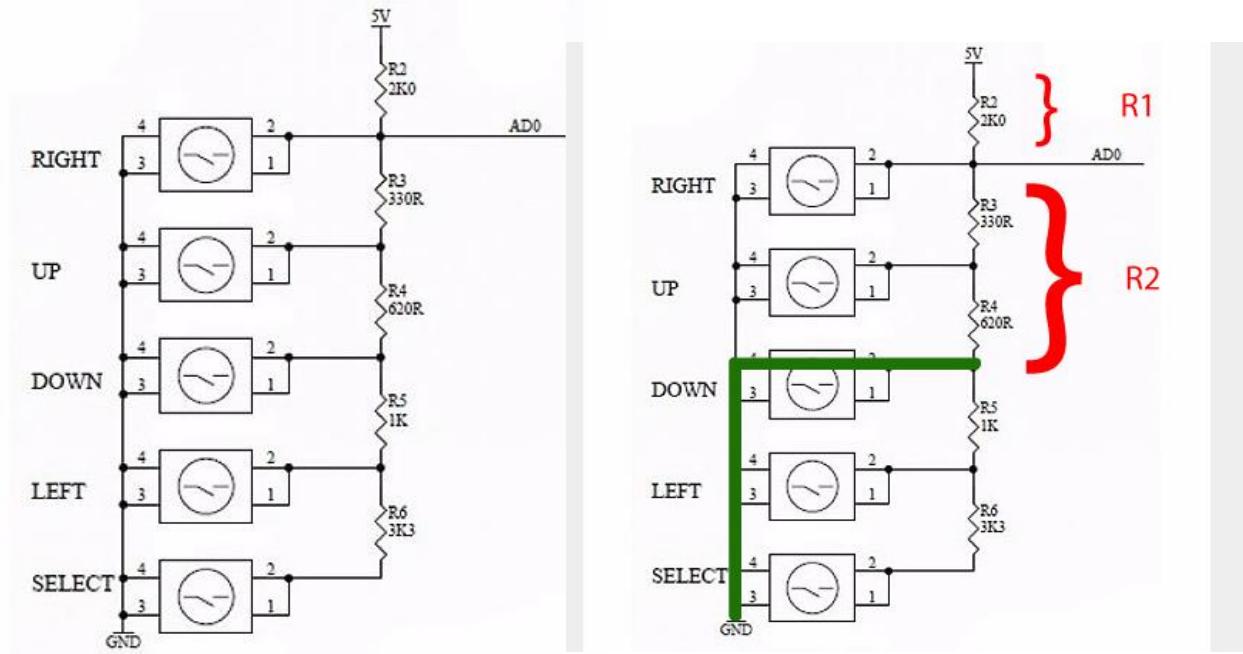
If the button is pressed P1 becomes 0.0 so ISLOW is true and becomes 1 which IF evaluates and returns the first term VOLTS(1.6). VOLTS is a function that converts 1.6 volts into the nominal range which is 5V by default so VOLTS(1.6) return (1.6 / 5).

If the button is not pressed P1 is O/C so ISLOW is false and becomes 0 which IF evaluates as false returns the second term 1 which is nominally HIGH or 5V

11.4.2 Multi Tap Voltage Divider

If your *microcontroller* doesn't have many spare pins it can be useful to decode multiple buttons using a single analog pin.

The way you can do this is to use the principle of Voltage Division to create different voltage levels depending on the button pressed. This voltage can then be read as an analog value and compared with calculated values to decode the button pressed.



The trick is to realise this is just a voltage divider

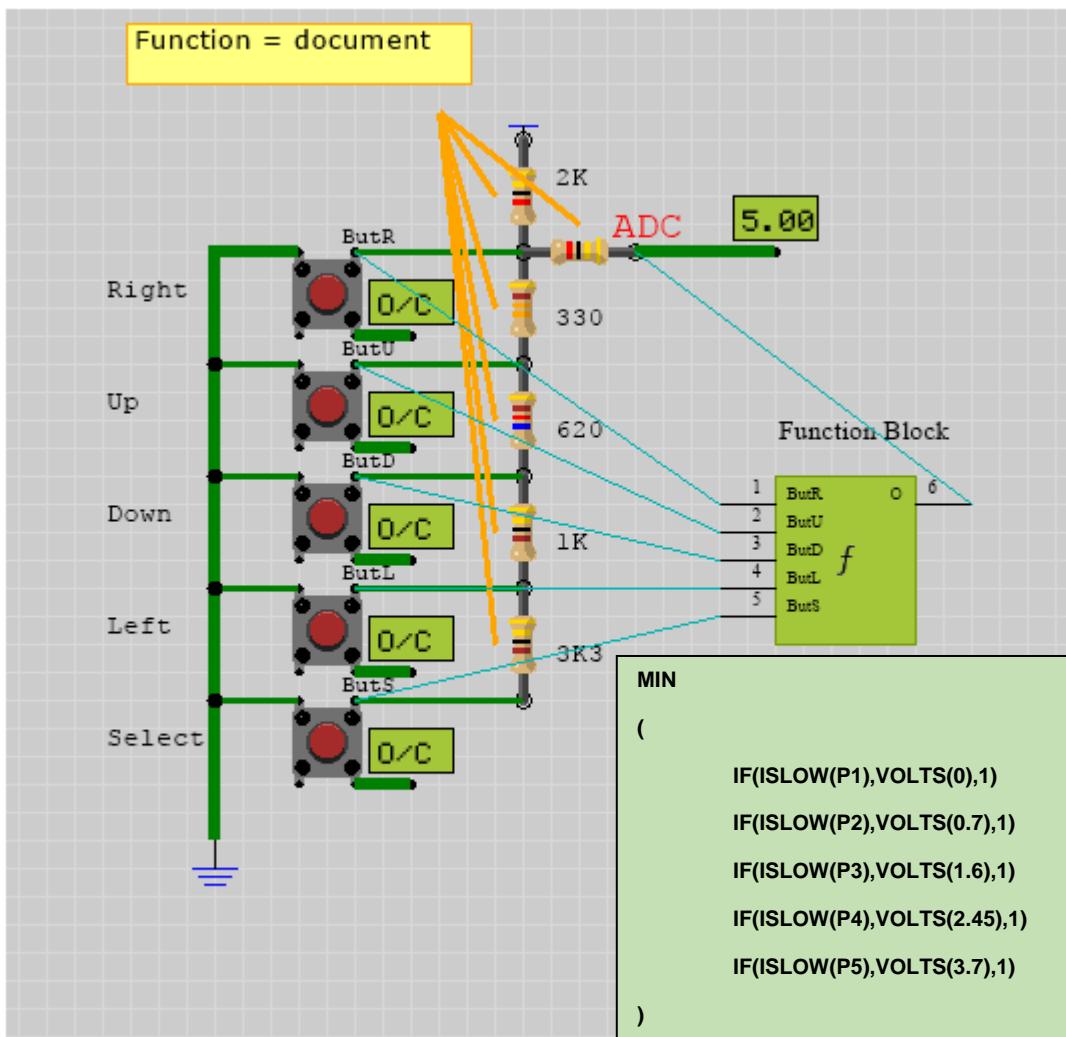
Voltage Divider R1 = 2K

Voltage Divider R2 = 330 + 620

$$V_{ButtonDown} = \frac{R_2}{R_1+R_2} \cdot V_{in} = \frac{330+620}{330+620+2K} \cdot 5V = \frac{950}{2950} \cdot 5V = 1.61V$$

So we can see each button press gives a unique voltage level we can decode

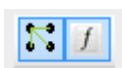
- VRight = 0.0V
- VUp = 0.7V
- VDown = 1.6V
- VLeft = 2.45V
- VSelect = 3.7V



To model this circuit block what we end up with is a FunctionBlock that merges the 5 buttons

It is intended that FunctionBlocks can be hidden behind the scenes and a good way to do that and keep the circuit models organised is to use Net Labels and netlists. You can see in the example the net labels on each button and these labels are matched by the netlist names of the FunctionBlock.

TIP: The hidden links are shown as the Jade coloured lines and the FunctionBlock is visible these can be toggled visible or hidden using the toolbar netlist toggle buttons



The expression uses the **MIN** function to select the minimum voltage from the switched button dividers created by the 5 separate buttons which gives us the final expression.

MIN

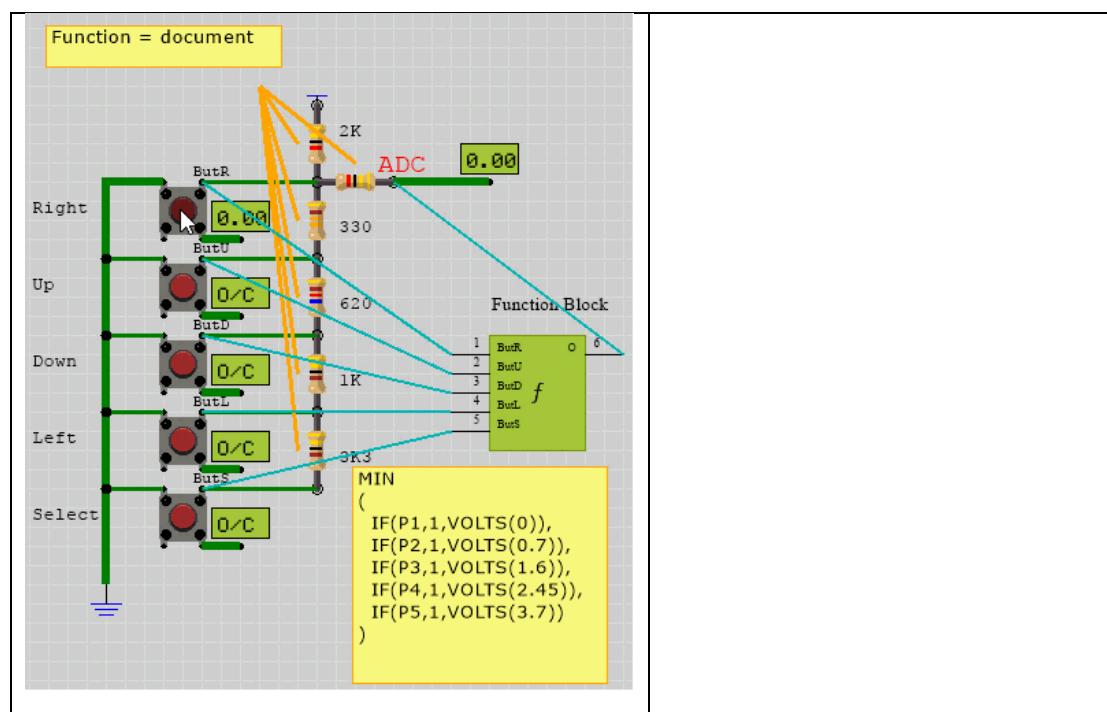
(

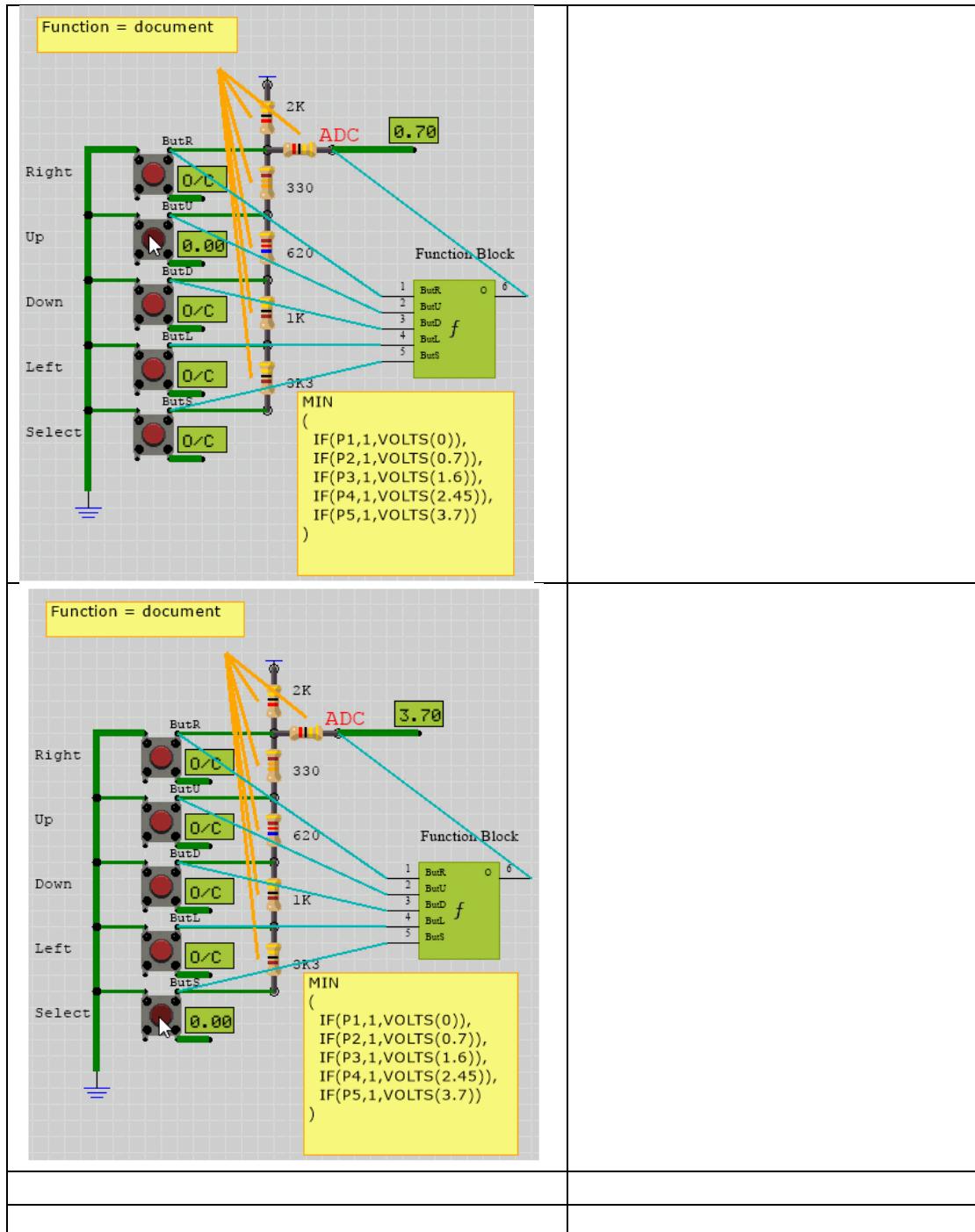
```
IF(ISLOW(P1),VOLTS(0),1)
IF(ISLOW(P2),VOLTS(0.7),1)
IF(ISLOW(P3),VOLTS(1.6),1)
IF(ISLOW(P4),VOLTS(2.45),1)
IF(ISLOW(P5),VOLTS(3.7),1)
```

)

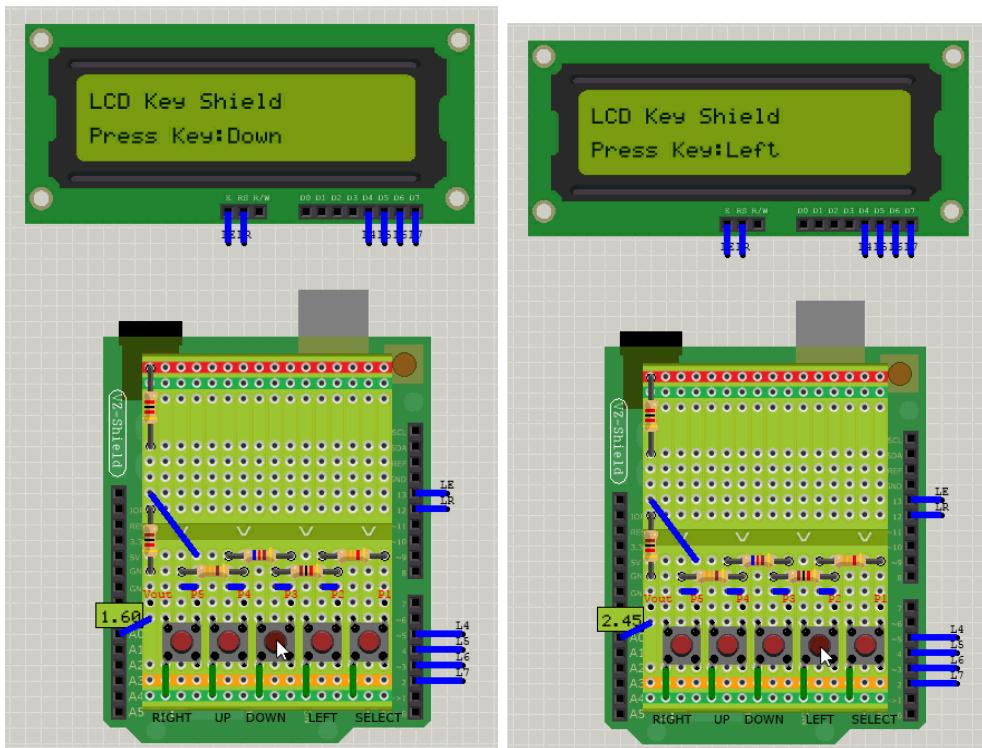
One difference to the original schematic is the use of the protection resistor to the ADC output of the function block. This is used actually as a trick to separate the output of the topmost right button to the output of the function block. Without it VBB would signal a short circuit but it's a useful circuit addition that doesn't change the voltage output.

Virtualization





Practical Application



VBB is primarily about firmware development support so it's only important that the circuit model behaves like the real circuit does from the point of view of the microcontroller application code.

```

int x;
x = analogRead(0);
lcd.setCursor(10, 1);
if (x < 60) {
    lcd.print("Right ");
} else if (x < 200) {
    lcd.print("Up ");
} else if (x < 400){
    lcd.print("Down ");
} else if (x < 600){
    lcd.print("Left ");
} else if (x < 800){
    lcd.print("Select");
}else{
    lcd.print("None ");
}

```

As you can see the microcontroller decoder code can't tell the difference and the decoder code works just the same as it does in real world.

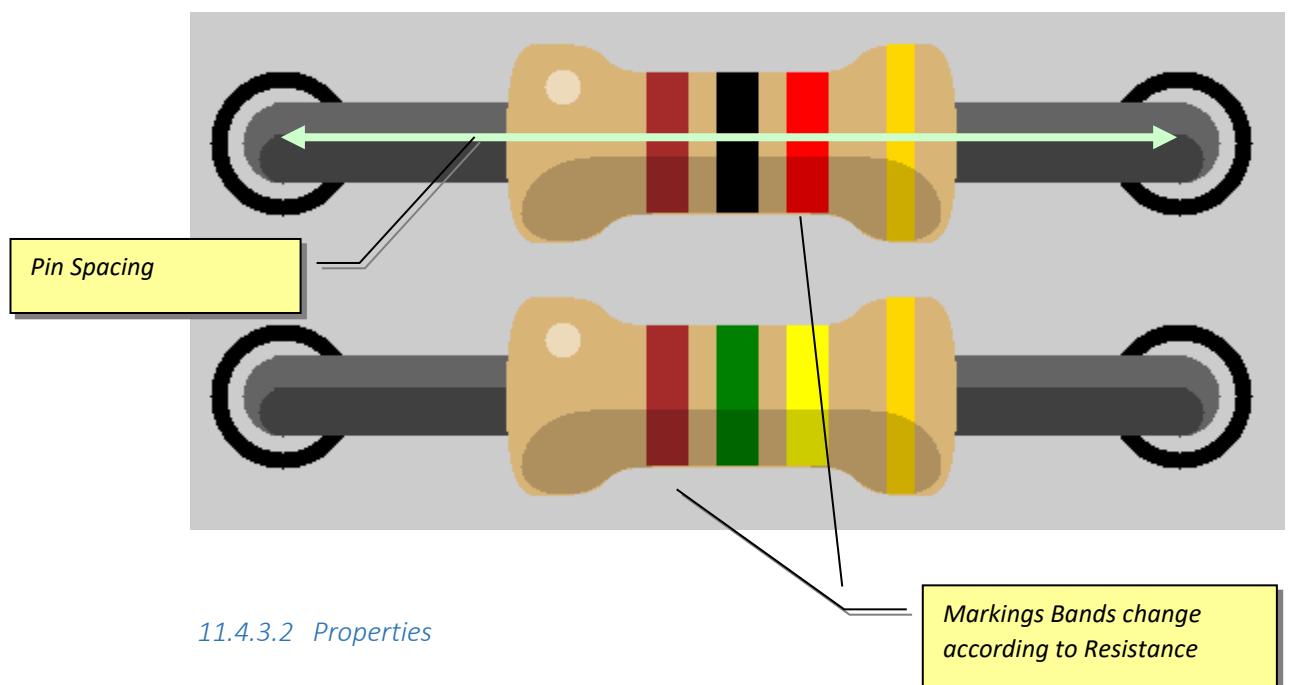
So to sum up. Use VBB function blocks to model circuit behaviour such that the microcontroller code can't tell the difference between the virtual model or a real

circuit. Then you can happily develop and test your application virtually before committing to hardware.

11.4.3 Resistors

Generic resistor component models the standard axial resistor with parametric spacing and resistance color bands.

11.4.3.1 Parametric Model



11.4.3.2 Properties

Property	Description
ID	Identifier used in BOM
Resistance	The resistance of resistor – the color markings update according to resistance
Spacing	The number of grid spacings between pins
Function	<i>Emulation Mode Fuse / wire / pullup / pulldown / Open-Circuit</i>

11.4.4 Resistors

Resistors have three modes which can be selected from the Function property of the resistor.

- fuse (Default)
- wire

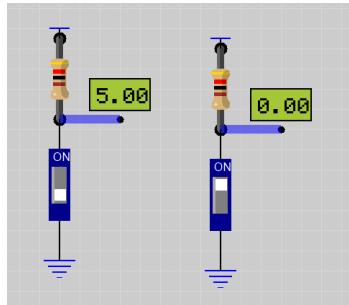
- Pullup
- Pulldown
- open-circuit (Document

11.4.4.1 Fuse Mode

In fuse mode the resistor acts as a poly fuse which if there is a short circuit will break its connection else it will make a wired connection. Fuse mode is the most default mode and the most versatile as it can represent a regular resistor, pullup and pulldown.

As a polyfuse when a resistor is open circuit one end of the resistor by the other end is 5V then the polyfuse is a wire and the other side of the resistor sees the 5V

When one end is 5V and other end is 0V, a short-circuit, the polyfuse breaks and an open-circuit is created



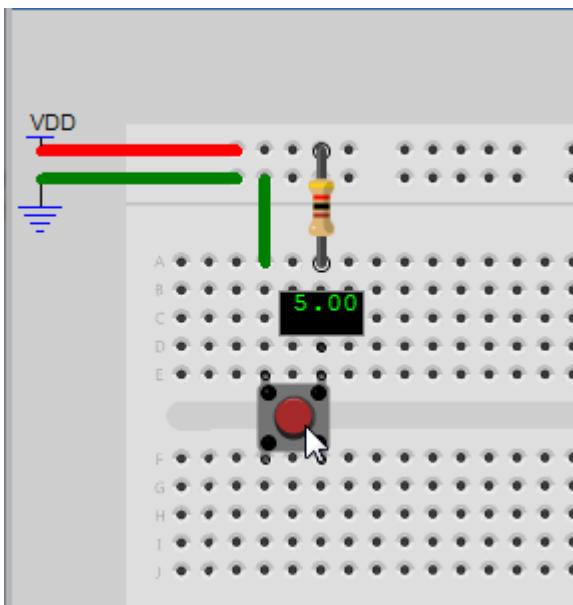
11.4.4.2 Pullup Mode

Pullup mode can be used for resistors that are dedicated as pullups. The main reason to use a resistor in pullup mode is when used as a basic voltage divider in conjunction with a pulldown resistor. Also there can be a minor performance improvement over using fuse mode

A Pullup resistor pulls a voltage HIGH if its not driven LOW.

 *One end of the pull-up resistor is assumed to be connected to Vdd even if its not actually wired that way.*

A common example shown here is to attach a pullup to a momentary switch so that the output is normally HIGH and switched LOW when the button is pressed.

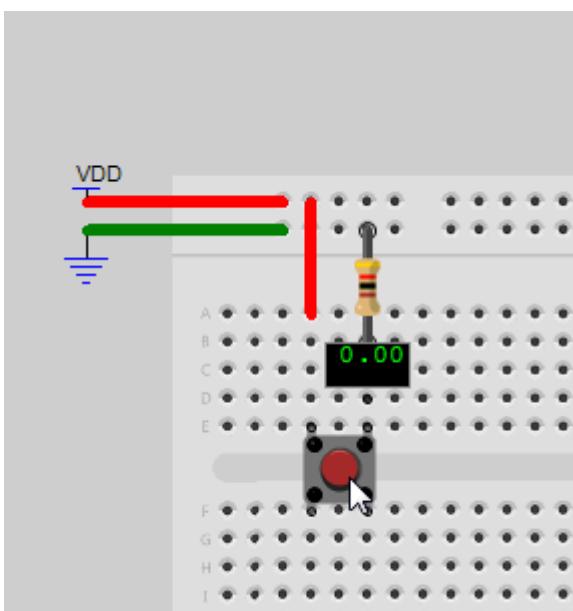


11.4.4.3 Pulldown Mode

A Pullup resistor pulls a voltage HIGH if its not driven LOW. The main reason to use a resistor in pullup mode is when used as a basic voltage divider in conjunction with a pulldown resistor. Also there can be a minor performance improvement over using fuse mode.

 One end of the pull-down resistor is assumed to be connected to Gnd even if its not actually wired that way.

A common example shown here is to attach a pulldown to a momentary switch so that the output is normally LOW and switched HIGH when the button is pressed.

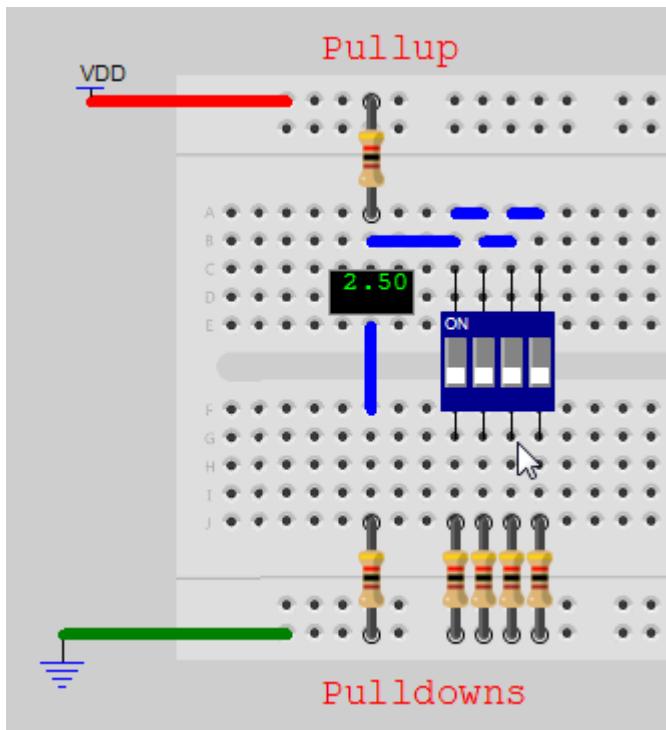


11.4.4.4 Voltage Divider

Pullup and Pulldown resistors can also be used in some Voltage Divider configurations.

Voltage dividers are used for creating reference voltages, or even digital to analog converters. In this example a 4-bit analog to digital converter is created by ladder. The resistors are switched together using a DIP switch array to create a network of resistor values. This circuit will emulate because the pullups are connected to VDD and the pulldowns are connected to GND. There are

other Voltage divider networks which dynamically drive the resistors, these cannot be emulated directly.



11.4.4.5 Wire Mode

In normal mode a resistor function the same as a regular wire. This can be used instead of fuse mode as there are minor performance improvement compared to a fuse

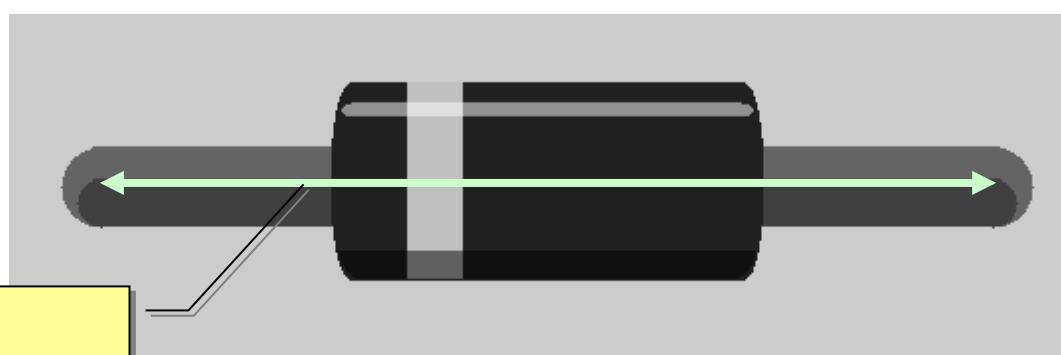
11.4.4.6 Document Mode

In document mode a resistor does not participate in the circuit

11.5 Diode

Generic diode component models the standard axial diode with parametric spacing.

11.5.1 Parametric Model



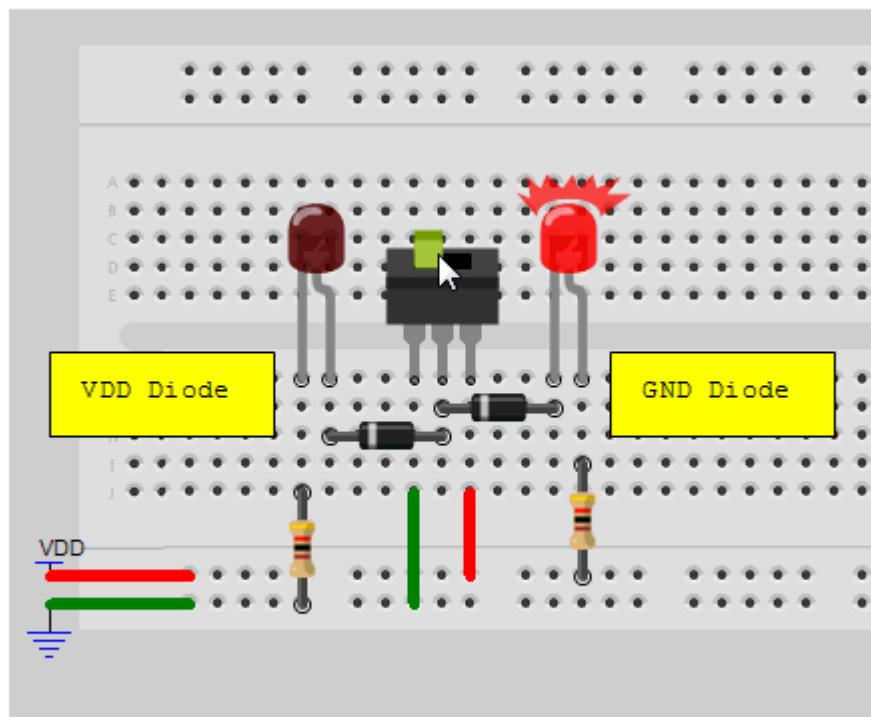
11.5.2 Properties

Property	Description
ID	Identifier used in BOM
Spacing	The number of grid spacings between pins
Function	<p><i>Emulation Behaviour</i></p> <p><i>Document / VDD / GND</i></p>

11.5.3 Diodes

Diodes are modelled as switched pull-up or switched pull-down resistor.

- document *No connection in circuit*
- VDD *Switched Pullup*
- GND *Switched Pulldown*



VDD and GND Mode

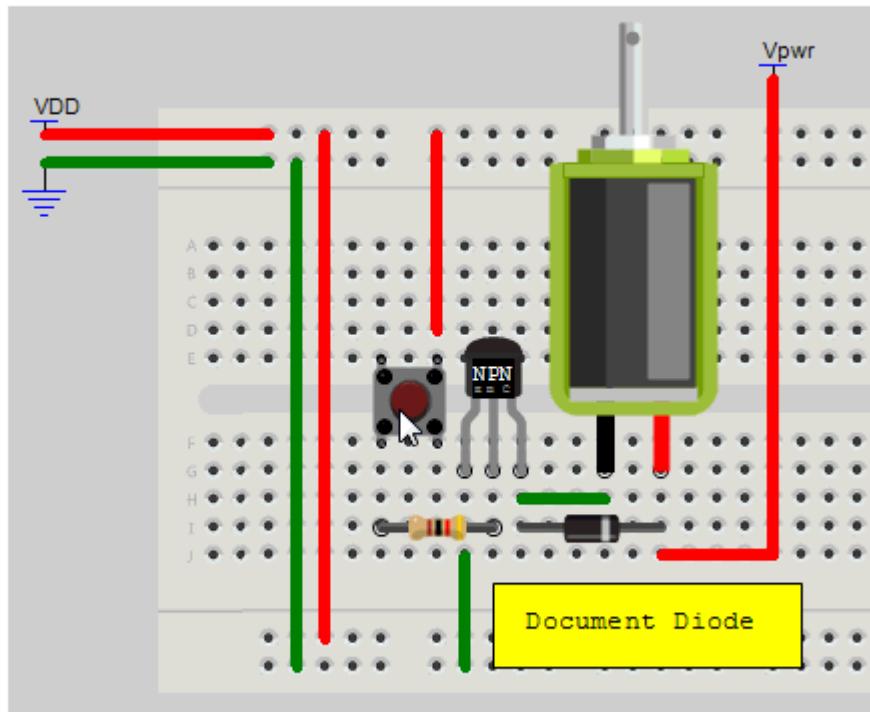
There are two common modes supported

In GND mode the diode acts as a pulldown resistor when connected to ground.

In VDD mode the diode acts as a pullup resistor when connected to VDD.

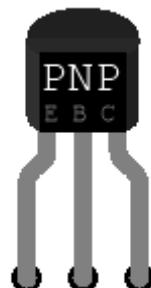
11.5.3.1 Document mode

Diodes can also be used in document mode. A common example is a protection diode you often see in circuits across solenoid and motors. The protection diodes can be placed using document mode so they appear in the circuit but don't participate in the emulation.



There are lots of other circuit tricks that use diode non-linearity and other diode characteristics. These won't work in VBB and require the use of document mode and functionblocks

11.6 PNP Transistor



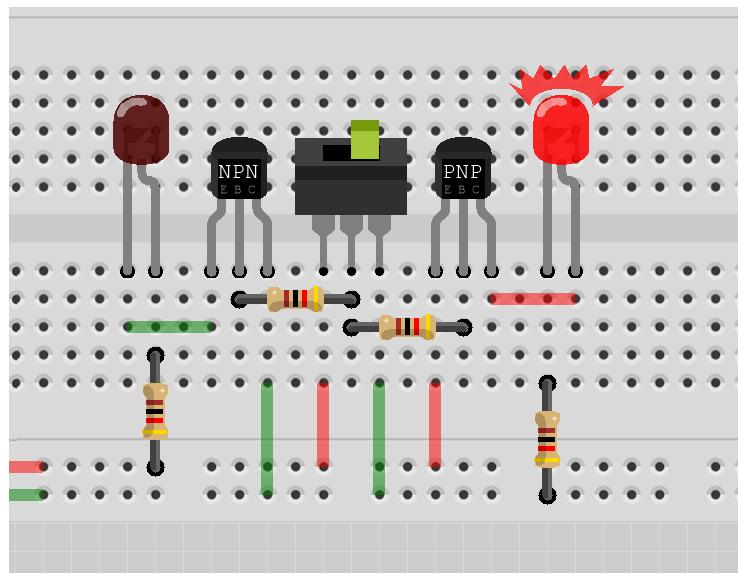
PNP Transistors are modelled as a switch which turns on when the base voltage is LOW

11.6.1 Properties:

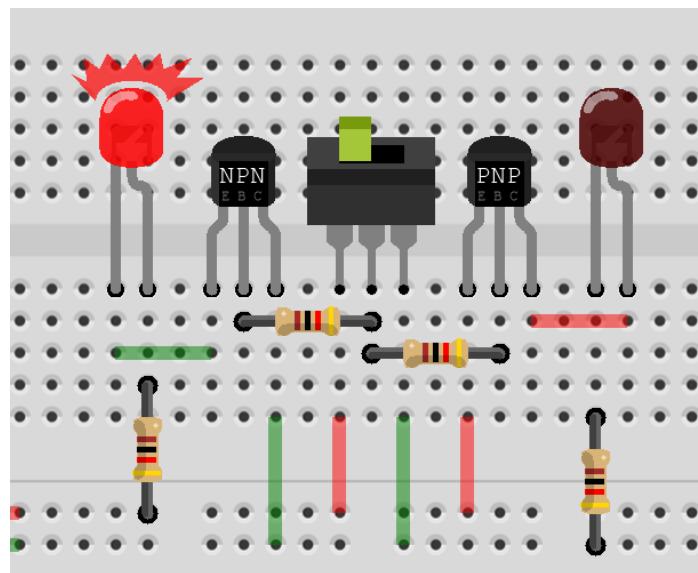
ID,Netlist

11.6.2 Usage

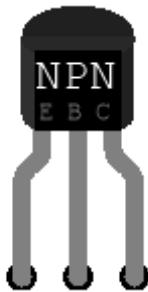
Switch connects the PNP B base to ground (through the base resistor) causing it to active it's internal switch connecting (E)mitter to (C)ollector which connects the anode(+) of the LED to the 5V rail causing it to flash on.



Switch connects the PNP B base to Open Circuit (through the base resistor) causing it to disconnect it's internal switch connecting (E)mitter to (C)ollector which disconnects the anode(+) of the LED from the 5V rail causing it to flash off



11.7 NPN Transistor



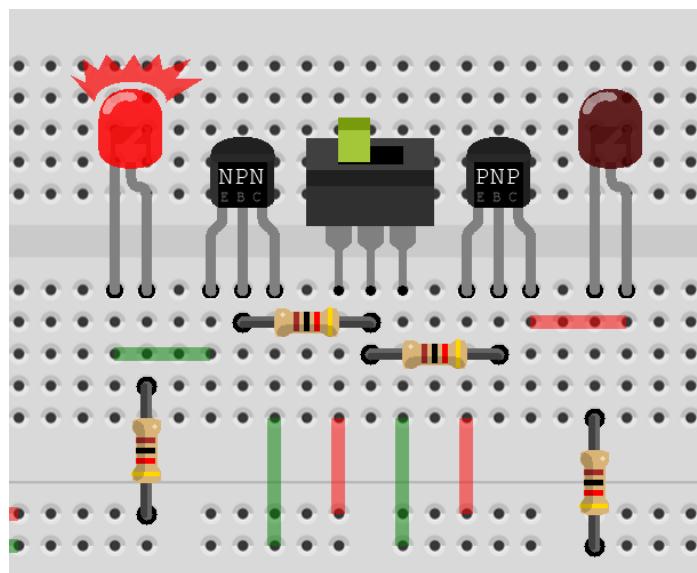
NPN Transistors are modelled as a switch which turns on when the base voltage is HIGH

11.7.1 Properties:

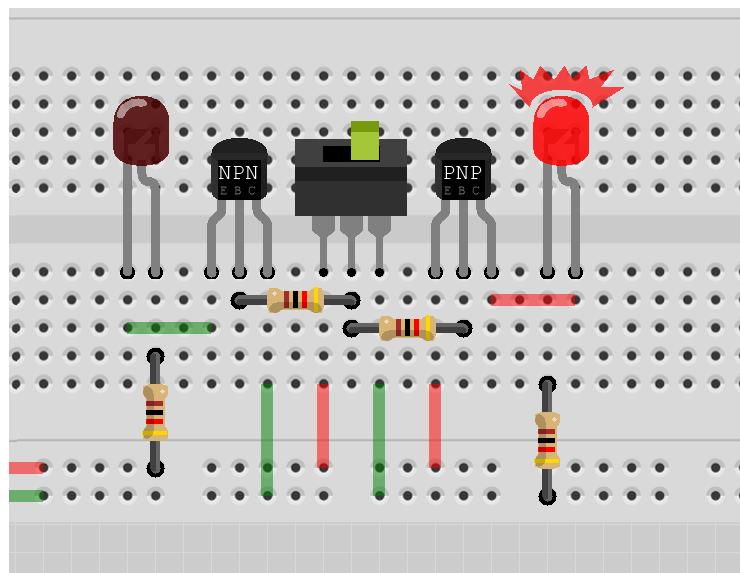
ID,Netlist

11.7.2 Usage

Switch connects the NPN B base to Vcc(5V) (through the base resistor) causing it to activate its internal switch connecting (E)mitter to (C)ollector which connects the anode(+) of the LED to the 5V rail causing it to flash on.

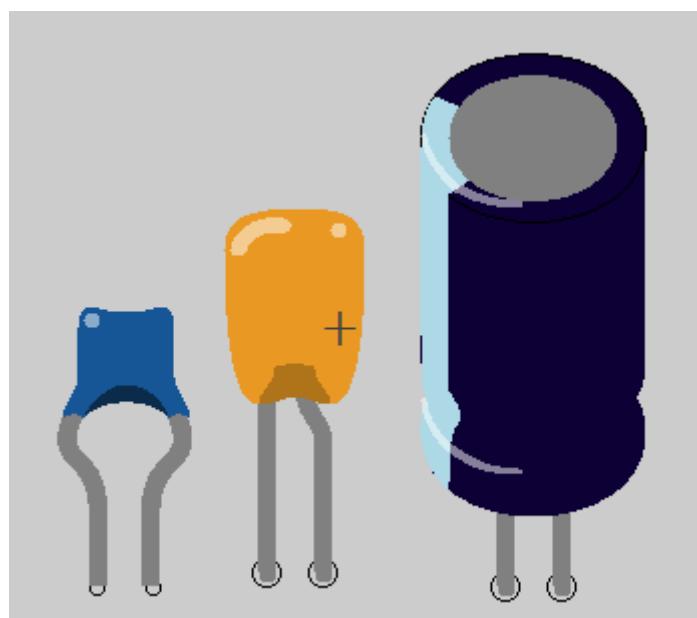


Switch connects the NPN B base to Open Circuit (through the base resistor) causing it to disconnect its internal switch connecting (E)mitter to (C)ollector which disconnects the anode(+) of the LED from the 5V rail causing it to flash off



11.8 Ceramic / Tantalum / Electrolytic Capacitor

Generic capacitor components



11.8.1 Properties

Property	Description
ID	Identifier used in BOM
Capacitance	The capacitance value of the capacitor to appear in BOM

12 IO Components

UserIO components don't match directly to physical devices but they provide common functions useful for quickly modelling and testing a circuit.

12.1 DIP1, DIP4, DIP8

The DIP component is an N array of DIP switches which are connected to 5V when Switch = ON and 0V (GND) when Switch = OFF. DIPS are an interactive component. Clicking on the DIP switch, the white square region, toggles the value of the switch. The SETTINGS property is an N Binary string representing the values of each of the DIP switches. Clicking the respective DIP switch will change the value of SETTINGS, alternatively the value can be directly edited in the SETTINGS property textbox.

Name	VBB Graphic	Equivalent Circuit
DIP1		
DIP4		
DIP8		

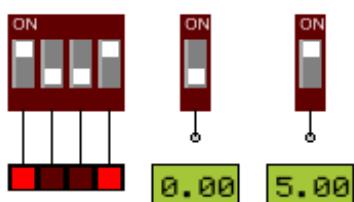
12.1.1 Pinout

Pin	Name	Description
N	<i>OUTPUT[N]</i>	DIP Output. Nth Pin is 5V if the Nth switch is ON. Nth Pin is 0V – GND if the Nth switch is OFF

12.1.2 Properties

Name	Default	Options	Description
OnState	HIGH	HIGH LOW	The value of the DIP output when in the ON state. Either driven HIGH or driven LOW when the DIP is on ON
DIP	0	<User Entry>	The decimal value of the DIP. DIP1 0 or 1, DIP4 0 to 15, DIP4 0 to 255. You can type the value into the setting box or click the dip switches at designtime or runtime to change the value
Pins	Bottom	Top Bottom	The pins are rendered either on the Top or Bottom of the DIP body as per this setting

12.1.3 Usage



12.2 LEDN

The LED component is an N array of Light Emitting Diode (LED) indicators. The Nth LED is ON when the Nth input pin is driven by greater than 2.5V otherwise the LED is OFF. The color of the LED is determined by the *COLOR* property. When the LED is ON the LED color is lighter than when OFF giving a visual clue to voltage level at the LED input pin. LEDs are a fundamental indicator of circuit status.

Name	VBB Graphic	Equivalent Circuit
LED1		<pre> graph LR In1(()) --- R1(()) R1 --- LED1[LED1] LED1 --- GND1(()) </pre>
LED4		<pre> graph LR In1(()) --- R1(()) R1 --- LED1[LED1] In2(()) --- R2(()) R2 --- LED2[LED2] In3(()) --- R3(()) R3 --- LED3[LED3] In4(()) --- R4(()) R4 --- LED4[LED4] LED1 --- GND1(()) LED2 --- GND1 LED3 --- GND1 LED4 --- GND1 </pre>
LED8		<pre> graph LR In1(()) --- R1(()) R1 --- LED1[LED1] In2(()) --- R2(()) R2 --- LED2[LED2] In3(()) --- R3(()) R3 --- LED3[LED3] In4(()) --- R4(()) R4 --- LED4[LED4] In5(()) --- R5(()) R5 --- LED5[LED5] In6(()) --- R6(()) R6 --- LED6[LED6] In7(()) --- R7(()) R7 --- LED7[LED7] In8(()) --- R8(()) R8 --- LED8[LED8] LED1 --- GND1(()) LED2 --- GND1 LED3 --- GND1 LED4 --- GND1 LED5 --- GND1 LED6 --- GND1 LED7 --- GND1 LED8 --- GND1 </pre>

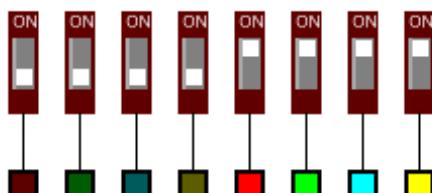
12.2.1 Pinout

Pin	Name	Description
N	<i>INPUT[2N]</i>	LED Input pins. Nth LED is ON when the Nth <i>INPUT</i> pin is > 2.5V else is OFF

12.2.2 Properties

Name	Default	Options	Description
COLOR	RED	RED GREEN BLUE YELLOW	Color of LED. LED is lighter when ON than OFF giving a visual clue to the status of the LED.
Cathode	Ground	Ground Pin	Expose the cathode to make it switchable
PinCount	1	N	Size of the LED array

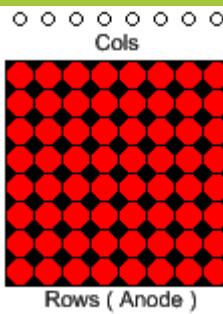
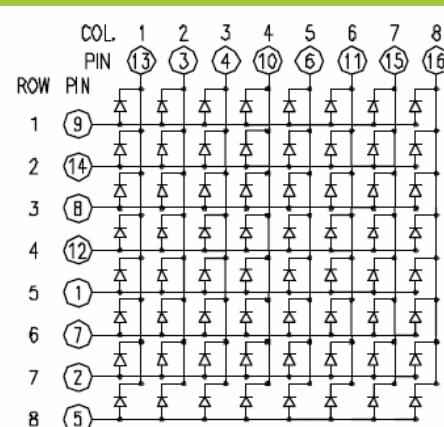
12.2.3 Usage



12.3 DotMatrixLED8x8

The DotMatrixLED8x8 component is a Dot Matrix LED array. The array consists of 8 Rows and 8 Columns. Each Row Anode pin drives 8 LEDs each attached to one of the cathode column pins.

<http://sigma.octopart.com/140413/datasheet/Lumex-LDM-24488NI.pdf>

Name	VBB Graphic	Equivalent Circuit
DotMatrixLED8x8		

By switching quickly between each row and using the persistence of vision effect each dot in the matrix can be individually addressed. To emulate the persistence of vision effect VBB only changes the state of the LED when COL is LOW. In this way the COL pin becomes like a latch operation which latches the value of LED.

COL	ROW	Description
LOW	HIGH	LED ON.
LOW	LOW	LED OFF
HIGH	<i>Don't Care</i>	No Change.

12.3.1 Pinout

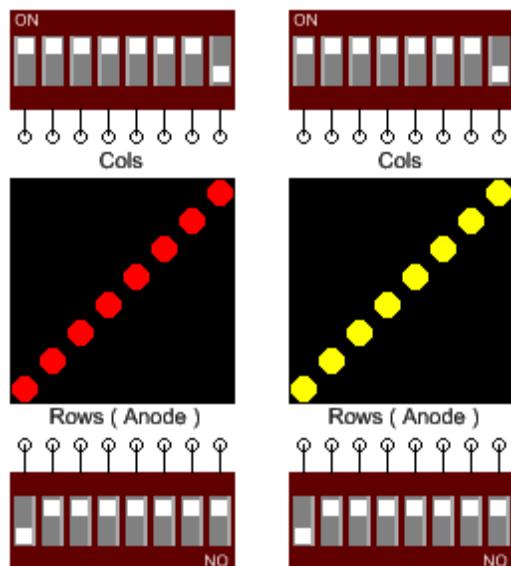
Pin	Name	Description

Dot Matrix Properties		
Col	Row	Description
Cols	Col 0..7	Column Cathode pins of the Dot Matrix array.
Rows	Row 0..7	Row Anode pins of the Dot Matrix array

12.3.2 Properties

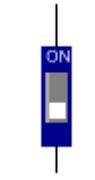
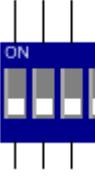
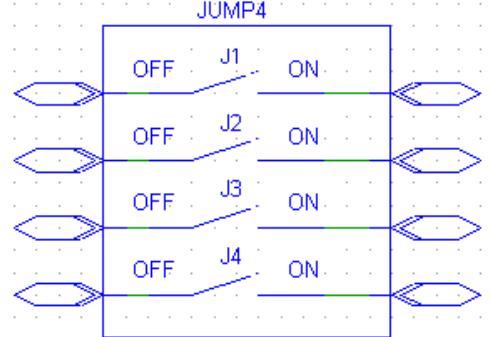
Name	Default	Options	Description
COLOR	RED	RED GREEN BLUE YELLOW	Color of LED in the array. The default color is red. The options are red,green,blue and yellow but you can also type in any named color

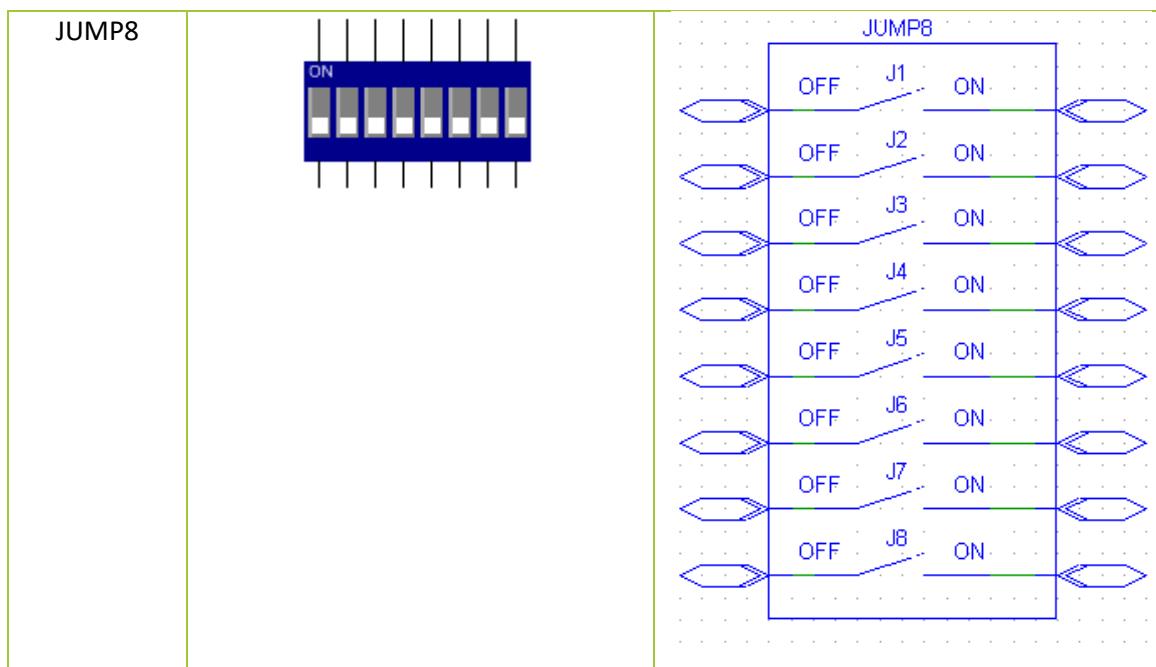
12.4 Usage



12.5 JUMPN

The JUMPER component is an N array of switches which form a short circuit between the Nth input/output pin pair when the Nth Switch = ON and open-circuit O/C when Nth Switch = OFF. JUMPERS are an interactive component. Clicking on the JUMPER switch, the white square region, toggles the value of the JUMPER switch. The SETTINGS property is an N Binary string representing the values of each of the JUMPER switches. Clicking the respective JUMPER switch will change the value of SETTINGS. Alternatively the value can be directly edited in the SETTINGS property textbox.

Name	VBB Graphic	Equivalent Circuit
JUMP1		
JUMP4		



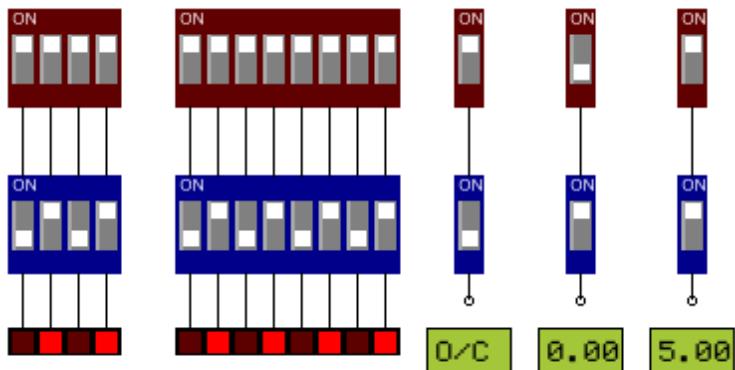
12.5.1 Pinout

Pin	Name	Description
N	<i>IO[2N]</i>	JUMPER IO Switch pair. Nth IO pair is short-circuit where Nth switch is ON, and open-circuit (O/C) when the Nth switch is OFF

12.5.2 Properties

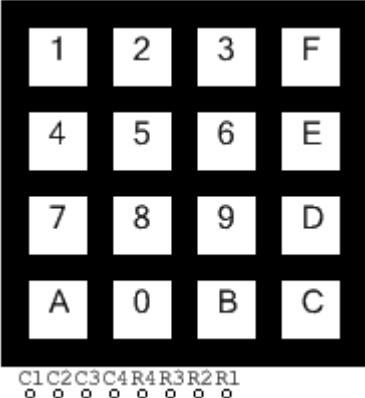
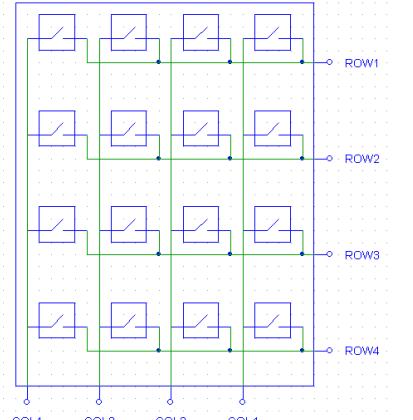
Name	Default	Options	Description
SETTINGS	[JUMP1] 1 [JUMP4] 1111 [JUMP8] 11111111	<User Entry>	N Bit binary string holding JUMPER switches status. Is modified by clicking the component in-place or direct editing in the SETTINGS property textbox.
N			

12.5.3 Usage



12.6 NumericKeyPad

The KEYPAD4x4 component consists of a Col x Row array of touch switches. The KEYPAD has column connect pins (C1,C2,..Cols-1) and row pins (R1,R2,..Rows-1). Each switch is connected to one column and one row connection pin such that each (row, column) combination is unique for each switch. The appearance of the KEYPAD component is determined by the KEYMASK property which contains a string of the key characters used on the key faces. The currently activated switch is determined by the KEYON property where **KEYON** = 0 when no switch is connected and **KEYON** = 1 to (Cols*Rows) when a valid switch is activated. KEYPAD is a user interactive component. Clicking on individual keys will activate the switch for that key. When active the background of the key becomes light green in color giving a visual clue to the state of the key. Only one switch can be active at any one time. Clicking on the currently active switch will toggle the switch to off resulting in no key being selected and **KEYON** = 0. When a key is on its switch becomes active creating a short-circuit across the unique row, column connection pin combination for that key.

Name	VBB Graphic	Equivalent Circuit
NumericKeyPad Rows=4 Cols = 4	 <pre> C1 C2 C3 C4 R4 R3 R2 R1 o o o o o o o o </pre>	

12.6.1 Pinout

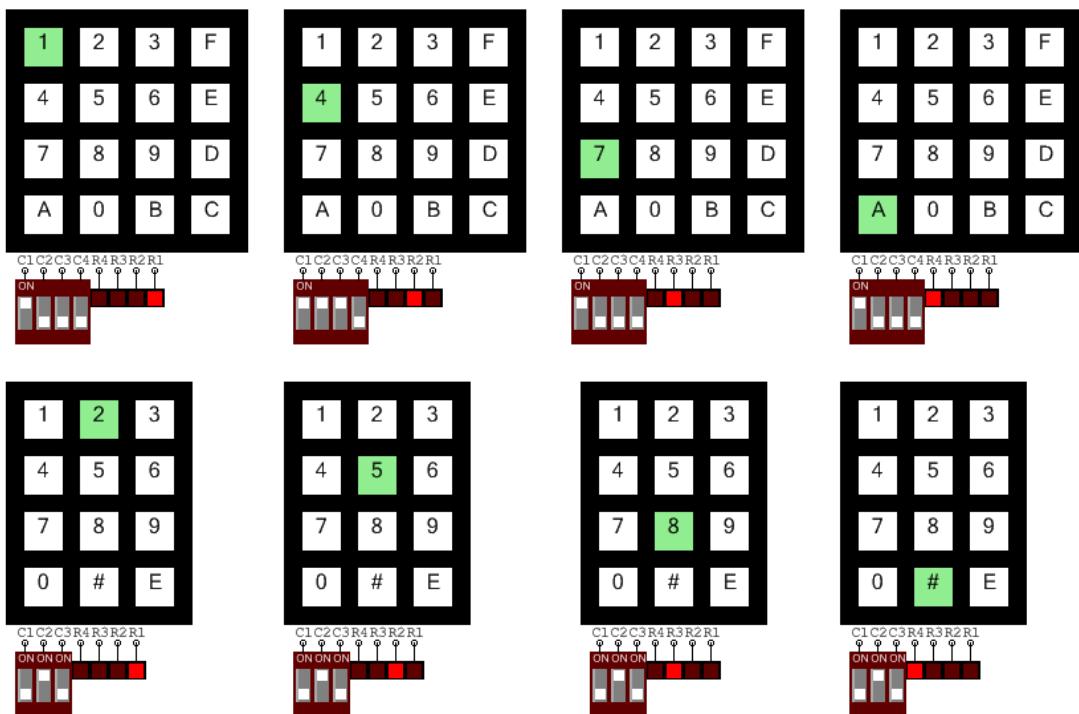
Pin	Name	Description
	C1, C2,..CN	Column Connections
	R1, R2,..RN	Row Connections

12.6.2 Properties

Name	Default	Options	Description
<hr/>			
Rows	4	<User Entry>	The number of Rows of the KeyPad
Cols	4	<User Entry>	The number of columns of the KeyPad
KeyMask	123F456E789DA0BC	<User Entry>	Col * Row character string containing the key-face character for each KEY

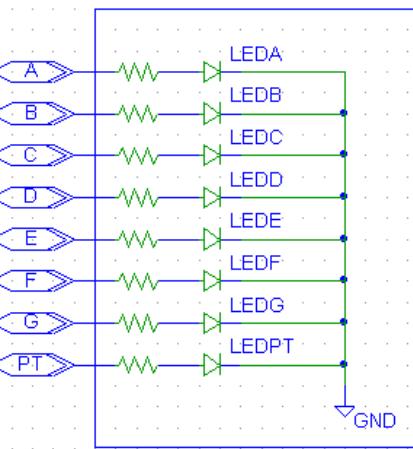
12.6.3 Usage

The switch is connected by pressing a key joining a column and a row. Typically the user scans all columns and row combinations to detect the keypress



12.7 Seg7

The SEG7 component is a 7-Segment Display module, which consists of a versatile array of LEDS which can be sequenced to form numbers and a limited form of alphanumeric character. 8 LEDS A, B, C, D, E, F, G, PT are physically arranged on the 7-Segment device. A LED Segment is ON when 5V HIGH is applied to its corresponding input pin and is OFF when 0V LOW is applied. When the appropriate patterns are applied to the inputs, 7-Segment display modules are able to display numbers and ASCII characters.

Name	VBB Graphic	Equivalent Circuit
		

12.7.1 Pinout

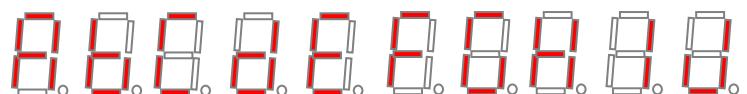
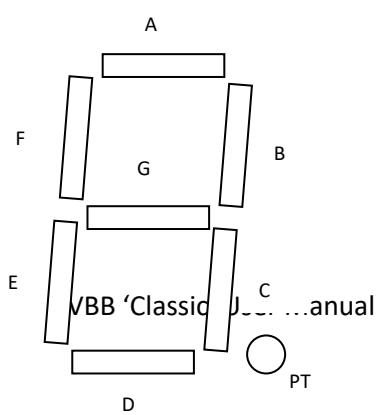
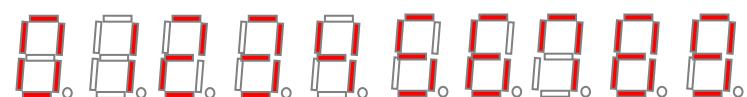
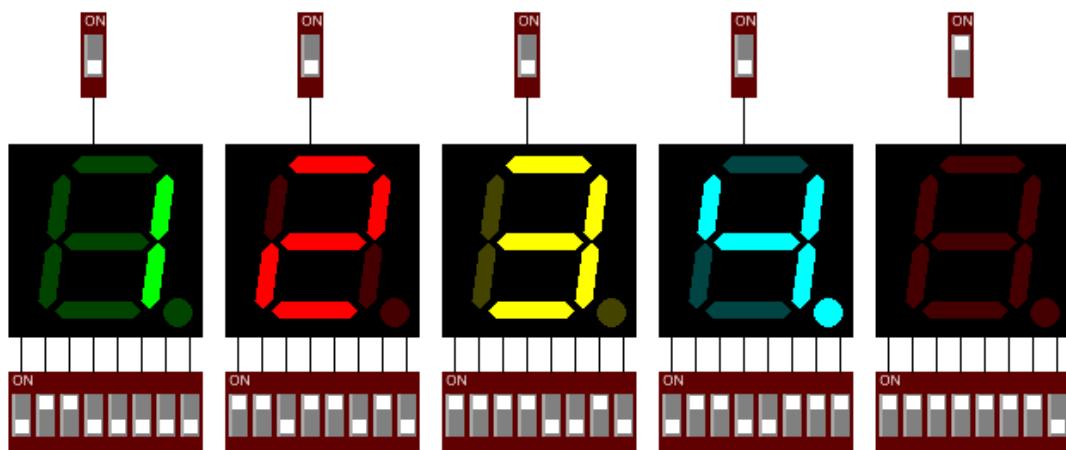
Pin	Name	Description
1	A	A LED Segment Input. HIGH = ON, LOW = OFF
2	B	B LED Segment Input. HIGH = ON, LOW = OFF
3	C	C LED Segment Input. HIGH = ON, LOW = OFF
4	D	D LED Segment Input. HIGH = ON, LOW = OFF
5	E	E LED Segment Input. HIGH = ON, LOW = OFF
6	F	F LED Segment Input. HIGH = ON, LOW = OFF

7	G	G LED Segment Input. HIGH = ON, LOW = OFF
8	PT	PT LED Segment Input. HIGH = ON, LOW = OFF
9	AN	Common Cathode. LOW = ON, HIGH = All Off

12.7.2 Properties

Name	Default	Options	Description
Color	red	red green blue yellow	Sets the color of the LED elements

12.7.3 Usage

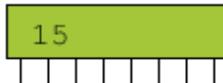


Char	A	B	C	D	E	F	G	PT	HEX
0	1	1	1	1	1	1	0	0	FC
1	0	1	1	0	0	0	0	0	60
2	1	1	0	1	1	0	1	0	DA
3	1	1	1	1	0	0	1	0	F2
4	0	1	1	0	0	1	1	0	66
5	1	0	1	1	0	1	1	0	B6
6	1	0	1	1	1	1	1	0	BE
7	1	1	1	0	0	1	0	0	E4
8	1	1	1	1	1	1	1	0	FE
9	1	1	1	1	0	1	1	0	F6
A	1	1	1	0	1	1	1	0	EE
B	0	0	1	1	1	1	1	0	3E
C	1	0	0	1	1	1	0	0	9C
D	0	1	1	1	1	0	1	0	7A
E	1	0	0	1	1	1	1	0	9E
F	1	0	0	0	1	1	1	0	8E
G	1	0	1	1	1	1	0	0	BC
H	0	1	1	0	1	1	1	0	6E
I	0	1	1	0	0	0	0	0	60
J	0	1	1	1	1	0	0	0	78
K	0	1	0	1	1	1	1	0	5E
L	0	0	0	0	1	1	0	0	06
M	1	0	1	0	1	0	1	0	M
N	0	0	1	0	1	0	1	0	N
O	0	0	1	1	1	0	1	0	O
P	1	1	0	0	1	1	1	0	P
Q	1	1	1	0	0	1	1	0	Q
R	0	0	0	0	1	0	1	0	R
S	1	0	1	1	0	1	1	0	S
T	1	0	0	0	1	1	0	0	8C
U	0	1	1	1	1	1	0	0	7C
V	0	1	1	1	1	1	1	0	7E

W	1	1	0	1	1	0	0	0	D8
X	0	0	0	0	1	1	1	0	07
Y	0	1	1	1	0	1	1	0	76
Z	1	0	0	1	0	0	1	0	92
.	0	0	0	0	0	0	0	1	01
+	0	1	1	0	0	0	1	0	62
-	0	0	0	0	0	0	1	0	02
(0	0	0	1	1	0	1	0	1A
)	0	0	1	1	0	0	1	0	32
-	0	0	0	1	0	0	0	0	10
SPACE	0	0	0	0	0	0	0	0	00

12.8 DigitalPort

The DigitalPort Component is an instrument which displays the decimal value of the binary value represented by its input pins

Name	VBB Graphic	Equivalent Circuit
DigitalPort		

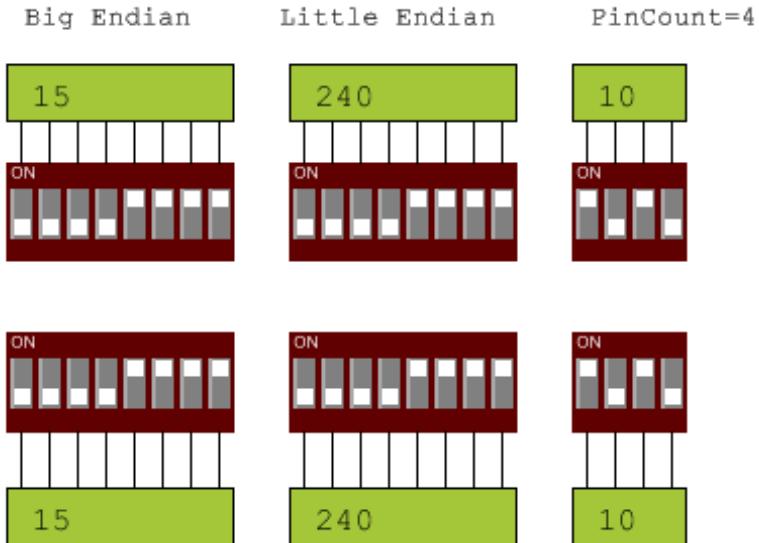
12.8.1 Pinout

Pin	Name	Description
1.. PinCount	<i>INPUT[PinCount]</i>	Port input pins

12.9 Properties

Name	Default	Options	Description
Endian	Big	Big Little	Endian determines the direction of calculation of the value of port
PinCount	8	4 8 16	The number of bits of the port.
Pins	Bottom	Bottom top	Determines if the pins are rendered on the top or the bottom

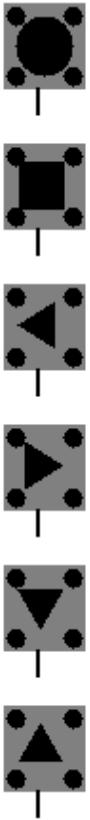
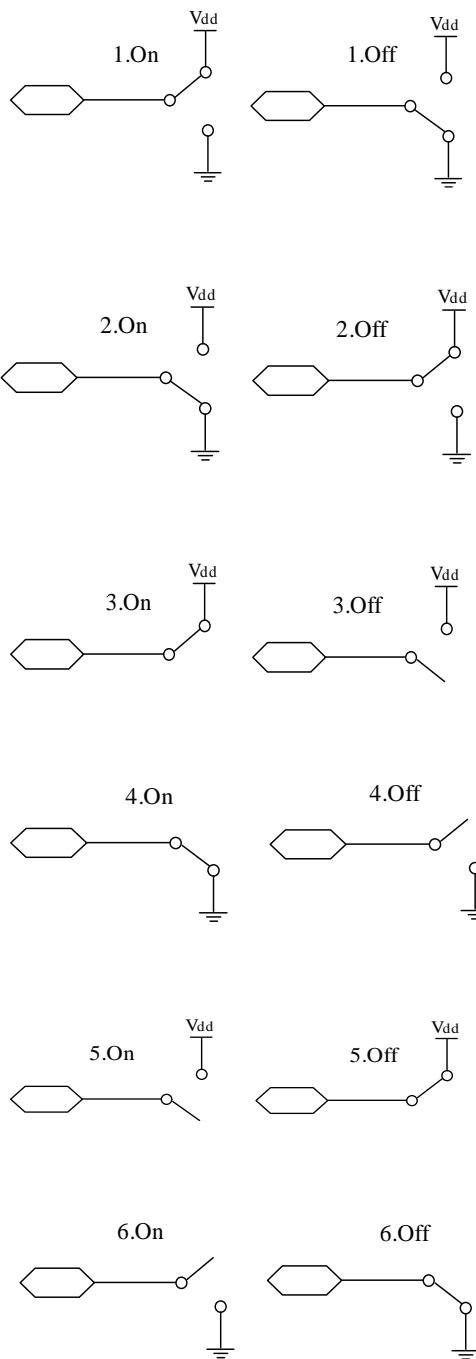
12.9.1 Usage



12.10 PushButton

The Button component is an interactive switch. Clicking the color panel of the button toggles the value of the POSITION property. The contact type determines the circuit made when the POSITION is ON or OFF. The color of the button is determined by the COLOR property. When POSITION = ON the color is set by ColorOn property and when POSITION =OFF the color is set by the ColorOff property. When these colors are not the same the color can be used as a visual clue to the state of the button.

Name	VBB Graphic	Equivalent Circuit

<p>Button</p> 	
---	---

12.10.1

Pinout

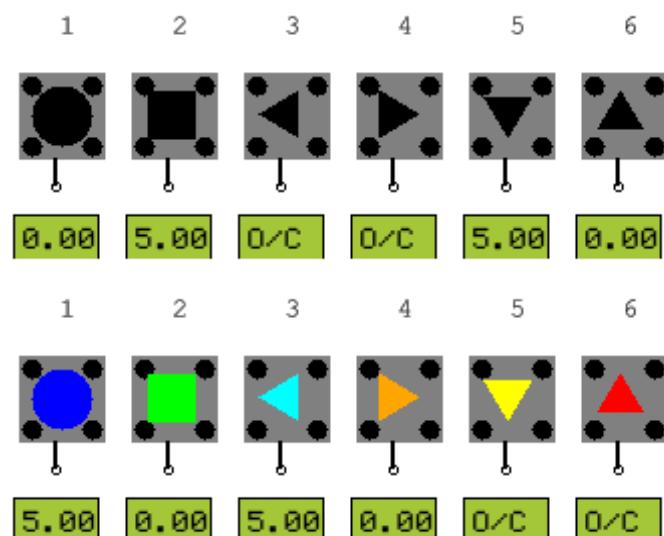
Pin	Name	Description
1.	B1	The contact pin for the button

12.10.2 Properties

Name	Default	Options	Description
Contact Type	1. On=VDD, Off=GND	1. On=VDD, Off=GND 2. On=GND, Off=VDD 3. Make VDD 4. Make GND 5. Break VDD 6. Break GND	The contact Type
Button	1. circle	1. circle 2. square 3. left arrow 4. right arrow 5. down arrow 6. up arrow	The graphic of the button.
ColorOn	blue	Red green blue yellow NamedColor	Sets the graphic color of the On button. Can be a named color in addition to the option colors
ColorOff	black	Red green blue yellow NamedColor	Sets the graphic color of the Off button. Can be a named color in addition to the option colors

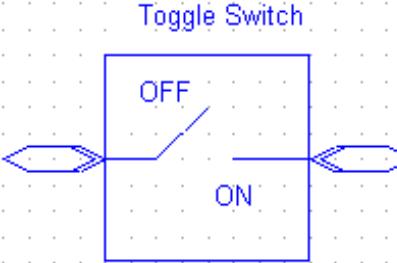
12.11 Usage

Not you can change the size of the buttons using the Scale Up 



12.12 Switch

The Switch component is an interactive switch. Clicking the switch image toggles the value of the POSITION property. When POSITION = ON a short circuit is present across the IO pins. When POSITION = OFF there is an open-circuit across the IO pins. To give a visual clue to the state of the switch, when POSITION = ON the switch lever in the ON Position or the Bitmap On is shown and when POSITION = OFF the switch lever in the OFF position or Bitmap Off is shown.

Name	VBB Graphic	Equivalent Circuit
ToggleSwitch		

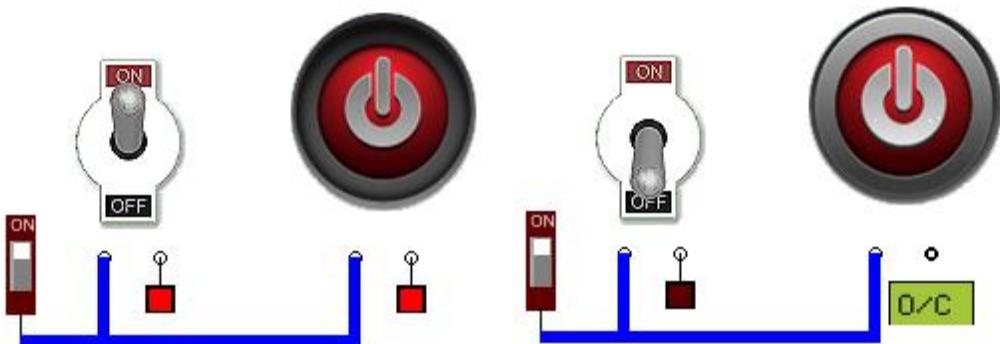
12.12.1 Pinout

Pin	Name	Description
1	IO	Switch input/output pin
2	IO	Switch input/output pin

12.12.2 Properties

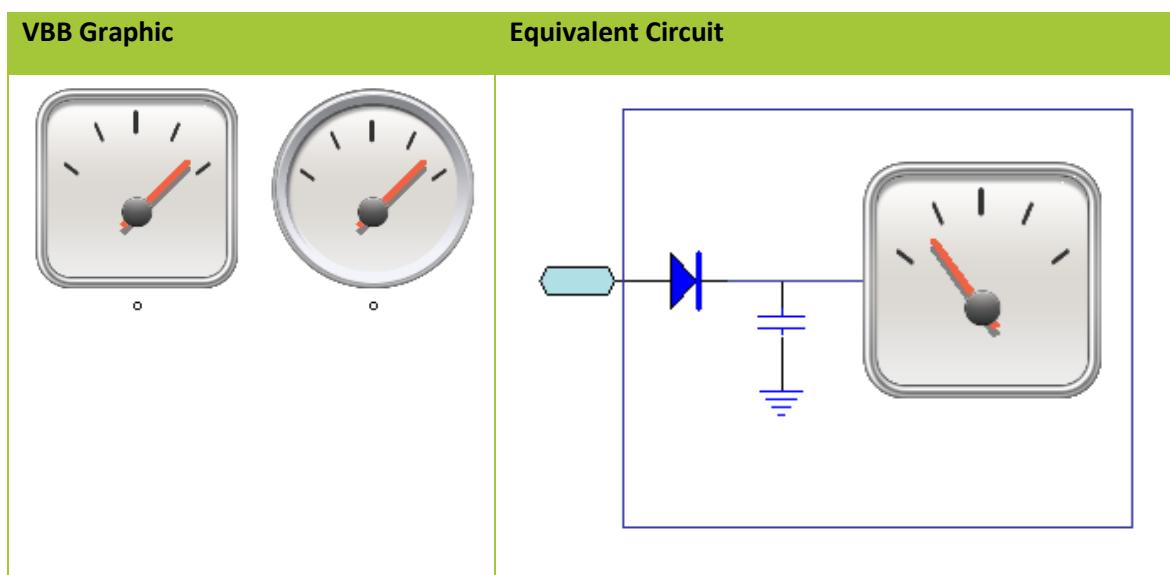
Name	Default	Options	Description
Switch	1. Off	1. Off 2. On	The state of the ToggleSwitch
Bitmap On		FileDialog for Bitmap .png .bmp	If set use this Bitmap in the On position
Bitmap Off		FileDialog for Bitmap .png .bmp	If set use this Bitmap in the Off position

12.12.3 Usage



12.13 PanelMeter

The PanelMeter is a Voltage Meter Instrument with a dial representation of the voltage on its input pin between Vss(left) and Vdd(right). The Panel Meter has the equivalent of a LOW PASS filter on the input pin so can display voltage levels for both absolute voltage levels and PWM equivalent analog voltages



12.13.1 Pinout

Pin	Name	Description
1		The voltage sense input pin

12.13.2 Properties

Name	Default	Options	Description
Shape	Square	Square Round	The shape of the Panel Meter

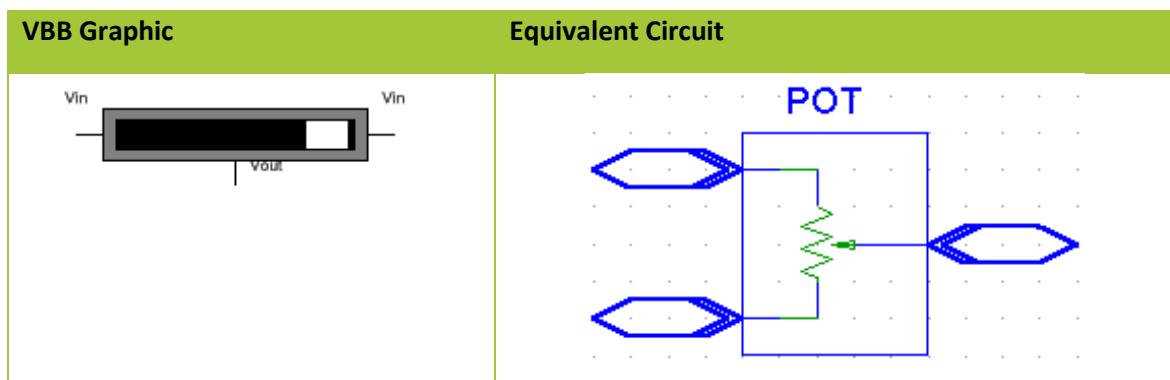
12.13.3

Usage



12.14 SlidePot

The SlidePOT (potentiometer) is a variable voltage divider and consists of two inputs and a single output. The *OUTPUT* voltage is determined by the input voltages and the value of the *POSITION* property where $OUTPUT = (\text{POSITION} \times V_{INA}) + ((1 - \text{POSITION}) \times V_{INB})$. Where Property *POSITION* has a valid range from range from [0..1]. POT is an interactive component. Clicking on POT will set the *POSITION* property. Clicking on the far left of the POT component corresponds to position 0. Clicking on the far right of the POT component corresponds to position 1. Clicking in-between the left and right will set the POT to a position linearly between 0,1. The *POSITION* property can also be set using the property editor. You can also drag the POT Knob with the mouse to continuously change the value.



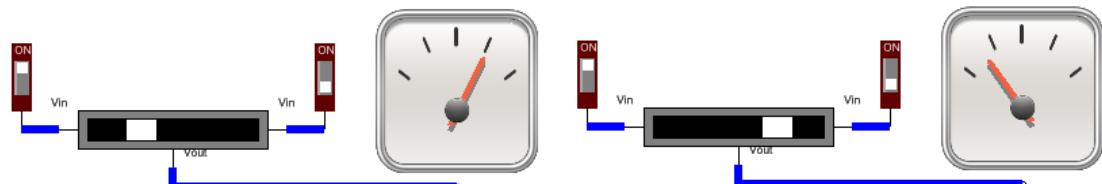
12.14.1 Pinout

Pin	Name	Description
1	<i>INA</i>	Input voltage reference
2	<i>INB</i>	Input voltage reference
3	<i>OUTPUT</i>	Variable output voltage

12.14.2 Properties

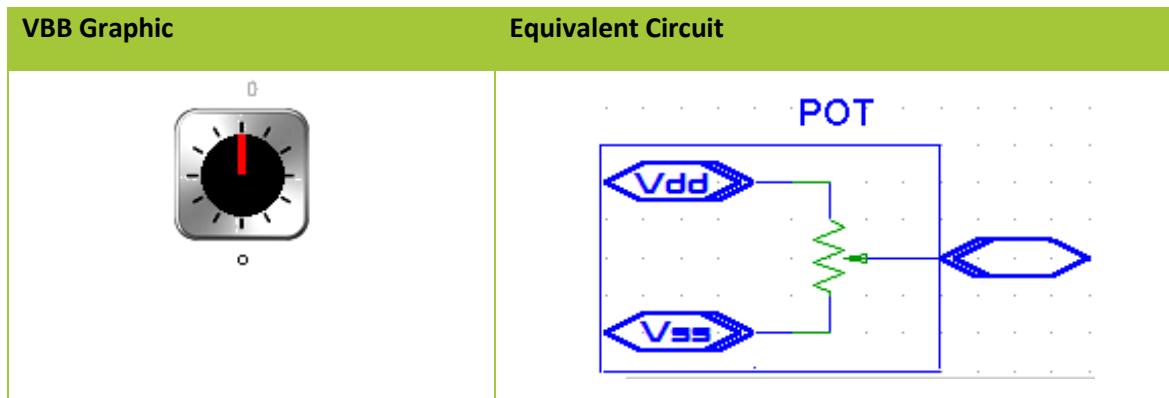
Name	Default	Options	Description
pot	0	0 25 50 75 100 <User Entry>	Potentiometer setting between 0,1 which determines the ratio of <i>INA</i> : <i>INB</i> using in <i>OUTPUT</i> voltage where $OUTPUT = (\text{POSITION} \times V_{INA}) + ((1 - \text{POSITION}) \times V_{INB})$.

12.14.3 Usage



12.15 RotaryPot

The RotaryPOT (potentiometer) is a variable voltage divider connected between Vdd and Vss with a single output. The *OUTPUT* voltage is determined by the rotation position where $\text{OUTPUT} = \text{Vdd} \times (\text{POSITION} / \text{Positions})$. To rotate the POT you use the mouse to drag the rotation dial.



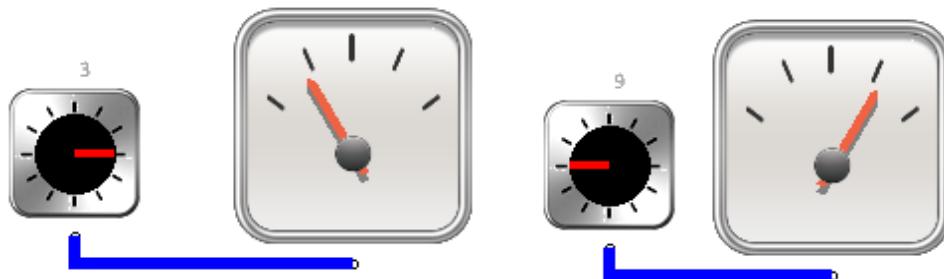
12.15.1 Pinout

Pin	Name	Description
1	<i>OUTPUT</i>	Variable output voltage

12.15.2 Properties

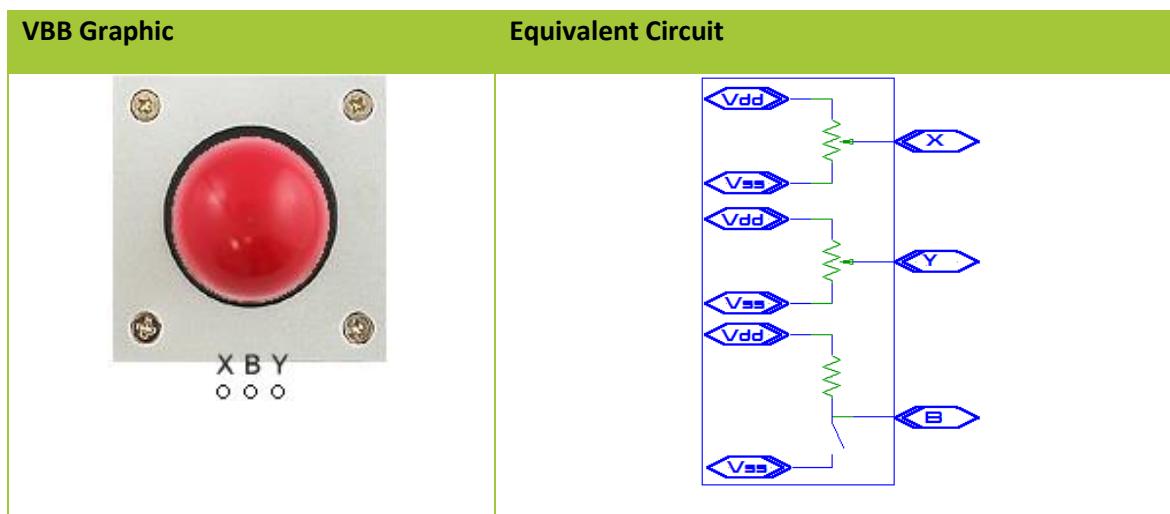
Name	Default	Options	Description
Positions	12	10 12 24 100 <User Entry>	The number of positions or steps of the Potentiometer
Position			Potentiometer setting between 0,Positions-1 which determines the $\text{OUTPUT} = \text{Vdd} \times (\text{POSITION} / \text{Positions})$

12.16 Usage



12.17 JoyStick

The JoyStick component is a model of a regular PC JoyStick with 2 potentiometers on the X and Y axis and button with a pullup resistor. Move the mouse over the keypad to move the JoyStick in X, Y and click the mouse to press the button pulling the button output to ground.



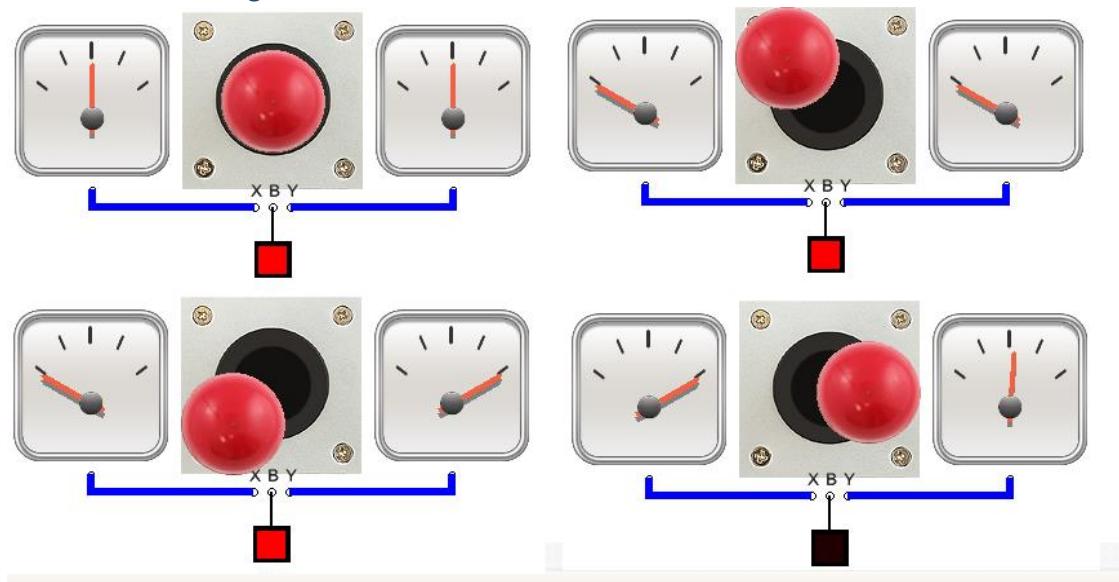
12.17.1 Pinout

Pin	Name	Description
1	X	X-Axis output Vcc (left) through Vdd (right)
2	B	Pulled up button. House mouse down to pull down
3	Y	Y-Axis output Vcc (top) through Vdd (bottom)

12.17.2 Properties

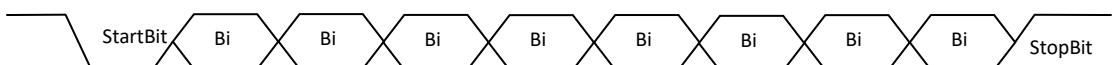
None

12.17.3 Usage



12.18 MiniTerminal

The mini terminal is a RS232 terminal which can send a receive asynchronous UART communications at TTL levels ie Vdd to Vcc at BAUD. Click on the Green screen segment and type to send messages. When pressing ENTER the characters set in the Enter property are sent. Communications is standard 8,N,1.



VBB Graphic	Equivalent Circuit
<pre>Line Text 0 Line Text 1 Line Text 2</pre> <pre>Line Text 0 Line Text 1 Line Text 2</pre> <p>TX (Out) RX (In)</p> <p>o o</p>	

12.18.1 Pinout

Pin	Name	Description
-----	------	-------------

1	<i>RX(In)</i>	The input into which the communications is received
2	<i>TX(Out)</i>	The output from which communications is sent

12.18.2 Properties

Name	Default	Options	Description
BAUD	9600	2400 9600 19200 57600 115200	The baud rate of the communications
Enter	LF(13)	CR(10) LF(13) CR(10)+LF(13)	The characters sent when pressing the enter key

12.18.3 Usage



13 Breadboardable Components

Breadboardable components are components that have both virtualization functionality and match up with real components so when you layout a circuit with Breadboardable components it becomes a makeable circuit.

13.1 LiquidCrystal

The LiquidCrystal component is a model of a 2 line HD44780 Liquid Crystal Display

Supports HD44780 emulation in 4-bit mode..

Fonts: If available the HD44780 font is used else the default is used.

Limitations

- 4 Bit mode only supported
- Custom Bitmaps not supported
- R/W read not support



Pin	Name	Description
1	Enable	Variable output voltage
2	Register Select Pin	
3	Read/Write Pin – Not functional	
4-11	D0-7	Data Pins. Only 4 bit mode using D4-D7 is used

13.1.1 Properties

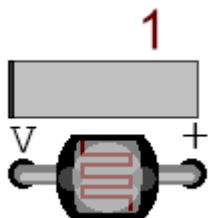
Name	Default	Options	Description
------	---------	---------	-------------

Display Cols	16	8 10 16 20 32 40	The number of columns of text the display will show can be used to model different displays
--------------	----	----------------------------	---

13.1.1 Usage



13.2 LDR – Light Dependent Resistor



13.2.1 Description

The Light Dependent Resistor, (LDR), is a useful photonic sensor that converts light into a difference in resistance. Of course in a virtual environment you can't actually change the light so a way is needed to vary the resistance. This is done by the LDR slider widget which you can slide from left to right to model the way light is varied.

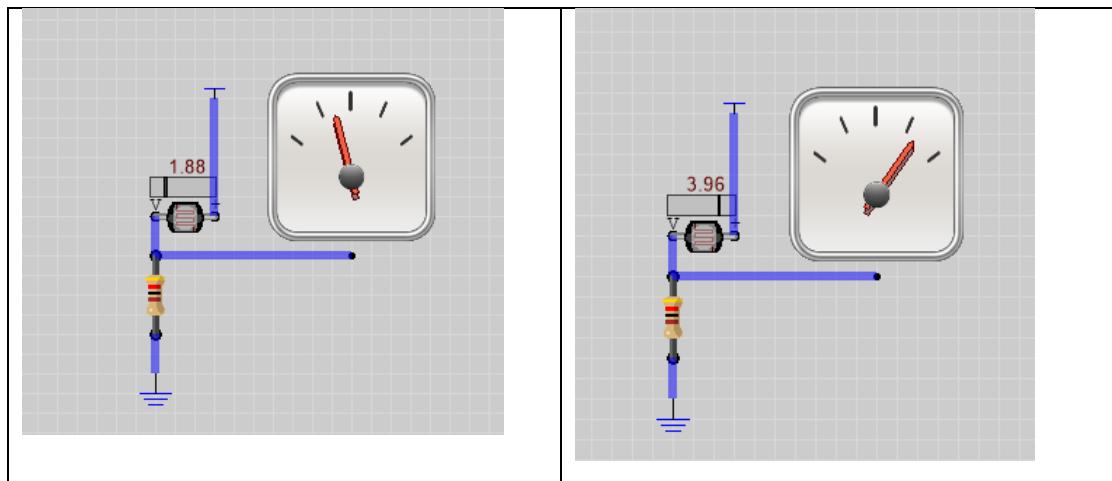
Normally a LDR forms a voltage divider and the variable resistance causes the voltage at the voltage divider tap to vary. The LDR models this by being a variable voltage source in the range MINVOLT to MAXVOLT as set by the properties.

13.2.2 Properties

Name	Default	Options	Description
MINVOLT	1	<User Entry> 1 to 5	Sets the value of the minimum voltage when the slider is all the way to the left
MAXVOLT	5	<User Entry> 1 to 5	Sets the value of the maximum voltage when the slider is all the way to the right

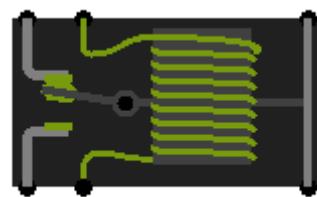
13.2.3 Usage

A voltage divider is formed with the resistor. By dragging the LDR widget different light conditions are simulated resulting in different output voltage levels.

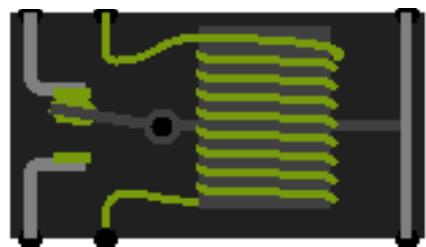


13.1 Relay

The relay is a common configuration for a solenoid relay. A relay uses a smaller powered solenoid to mechanically switch a high current circuit. Relays are useful for driving higher voltage and current systems. When the solenoid circuit is powered the switch changes position connecting the alternate path.



1 4 3a



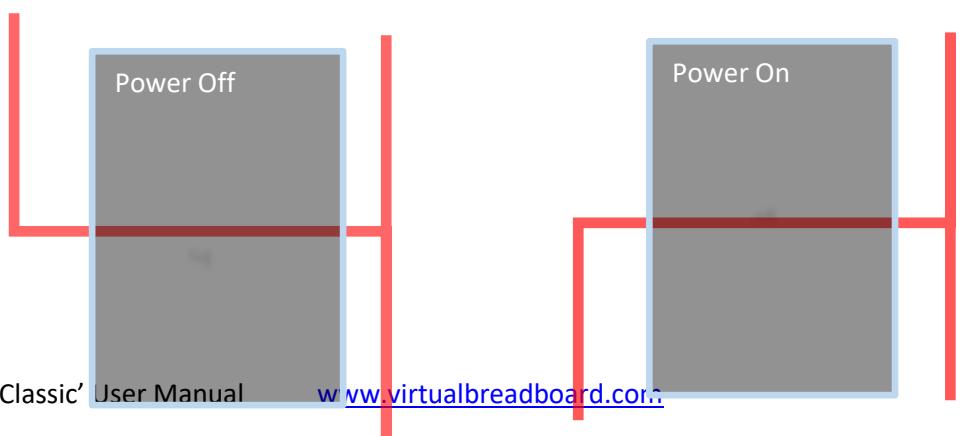
2 5 3b

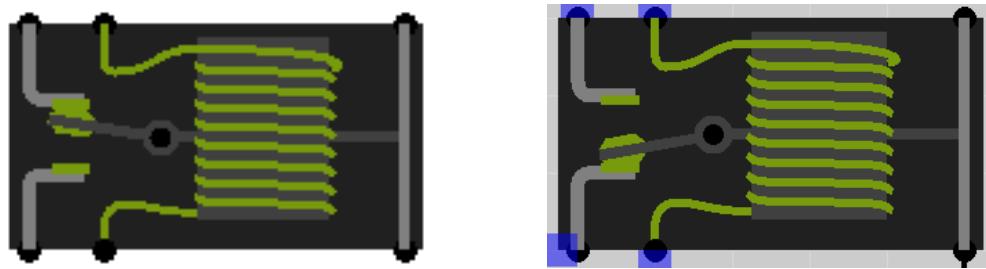
13.1.1 Pins

Pin	Name	Description
1	Switched A	Switched A connects to common when solenoid circuit is NOT powered
2	Switched B	Switched A connects to common when solenoid circuit IS powered
3	Common	The common pin
4	Solenoid A	Solenoid is powered when a current flows between Solenoid A and Solenoid B
5	Solenoid B	Solenoid is powered when a current flows between Solenoid A and Solenoid B

13.1.2 Properties

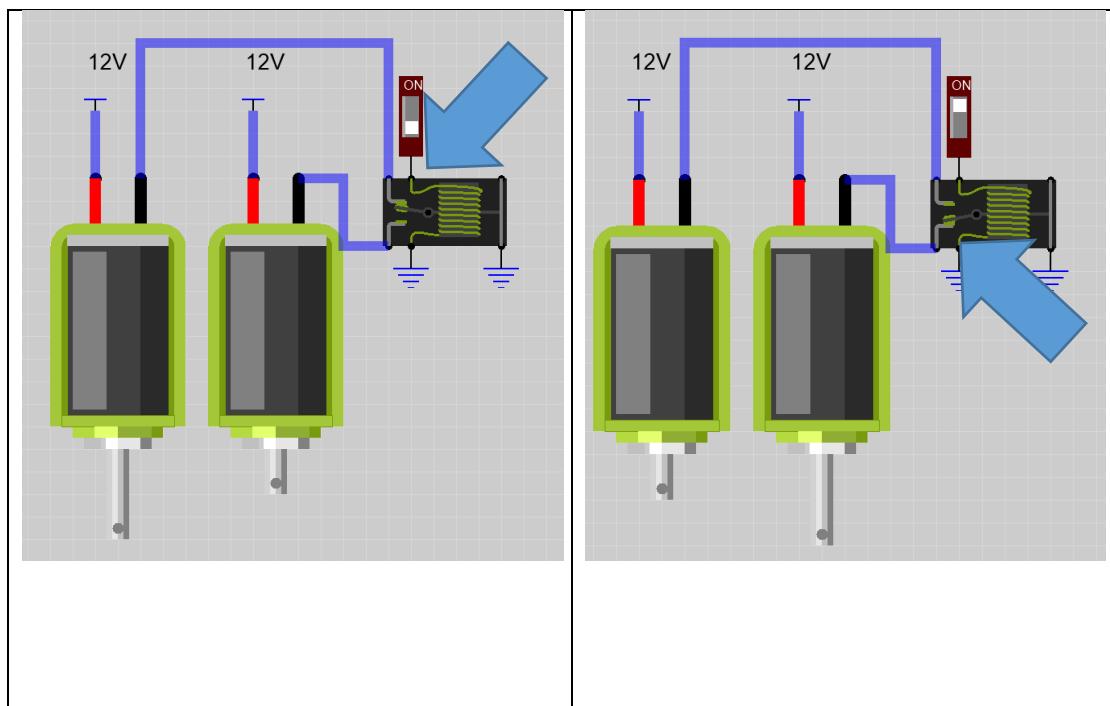
Common Properties only





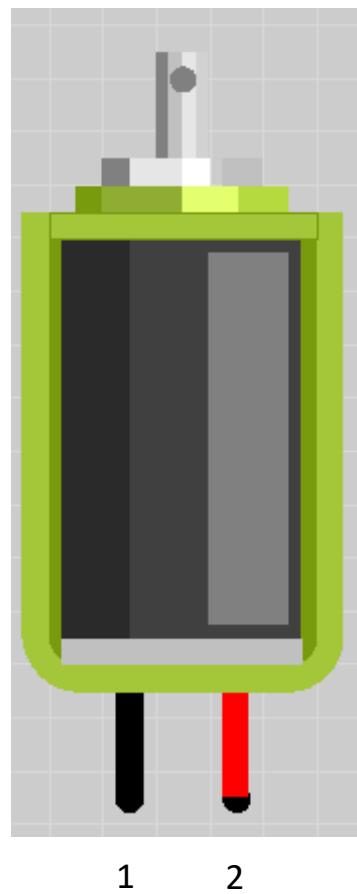
13.1.3 Usage

Relays are used to drive larger loads, typically mechanical loads such as larger solenoids. In this example you can see two solenoids are driven by a 12V supply to ground via the Relay. You can also visualise the action of the relay with the small switch internal to the relay



13.2 Solenoid

The solenoid is a mechanical actuator that when a sufficient current passes from PWR to GND pint the centre rod actuates. Solenoids are useful for many different mechanisms such as locking.



13.2.1 Pins

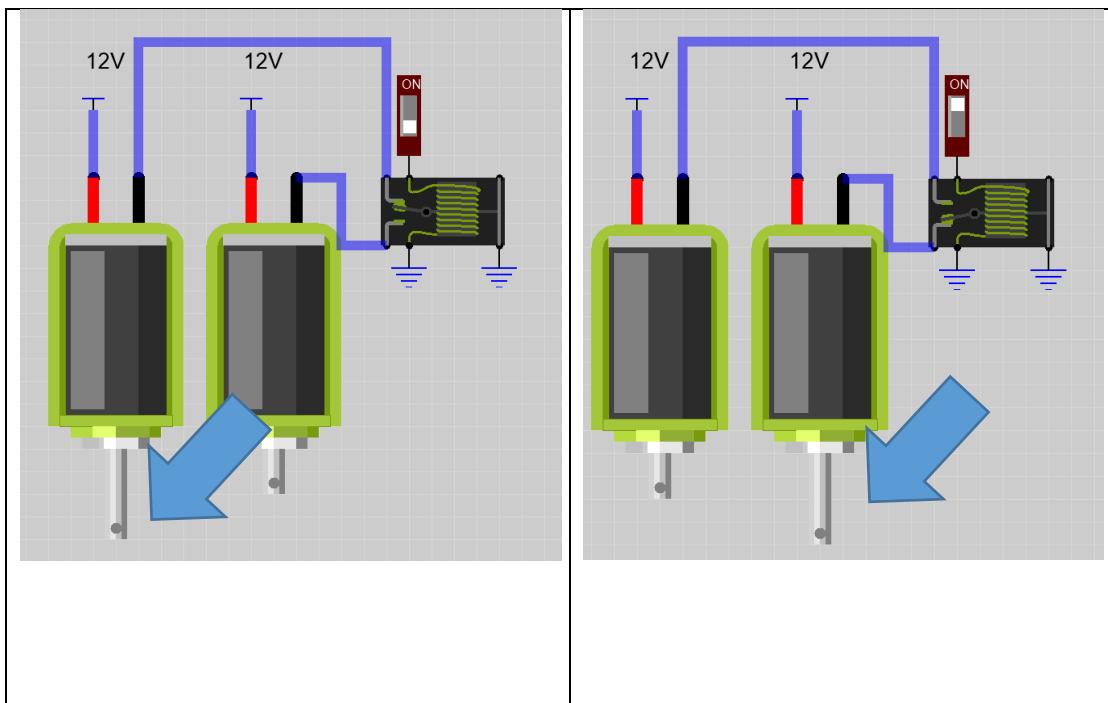
Pin	Name	Description
1	Ground	Switched A connects to common when solenoid circuit is NOT powered
2	Power	Switched A connects to common when solenoid circuit IS powered

Properties

Common Properties only

13.2.2 Usage

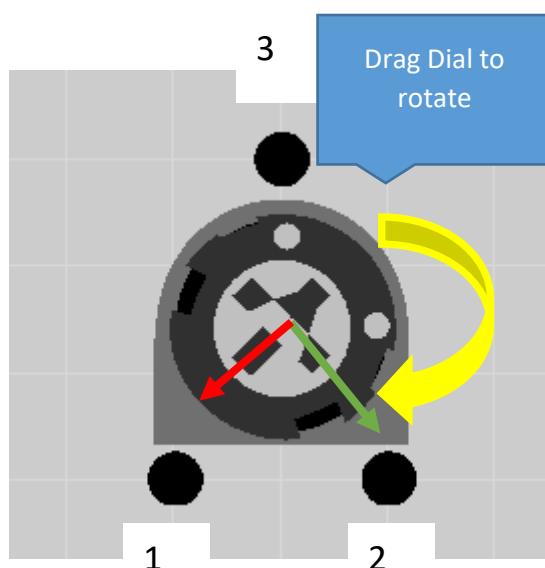
The solenoid actives when it is driven otherwise it is in the retracted position



13.1 Trimmer

The trimmer is a voltage divider potentiometer in the common 3 leg 'tripod' configuration with an adjustable centre. Dragging the trimmer dial changes the position of the center tap and hence changes the value of the resistor proportions of the voltage divider changing the output voltage proportionally to the rotation position.

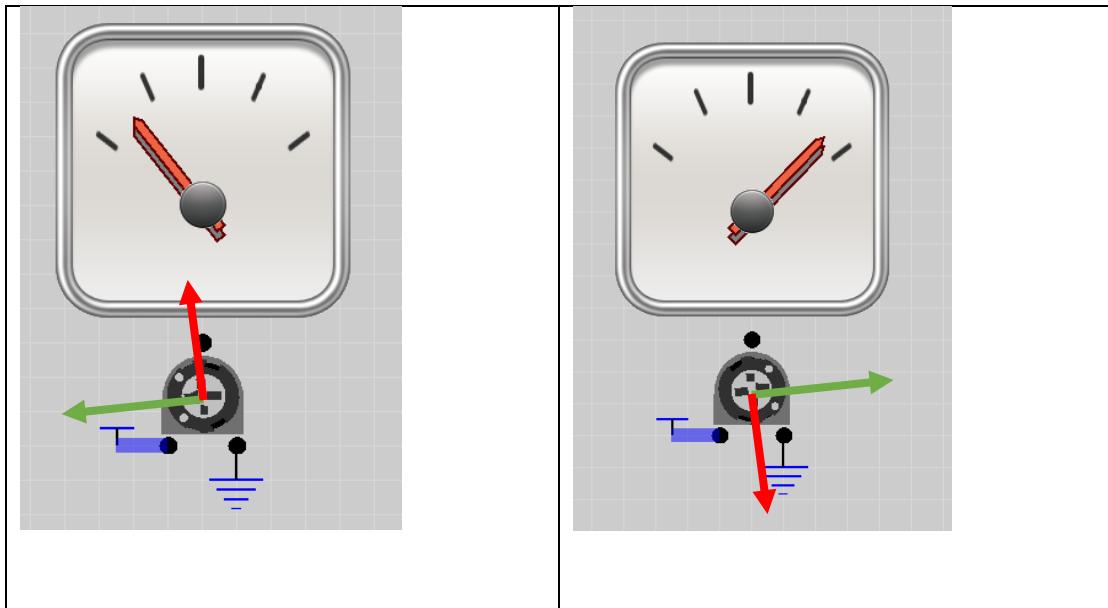
The position of the dial is persisted when saved and restored on open and run.



13.1.1 Pins

Pin	Name	Description
1	A	Variable resistor pin
2	B	Variable resistor pin
3	Tap	The voltage at TAP is $V1 = (Va-Vb) * \text{position} / \text{MaxPosition}$

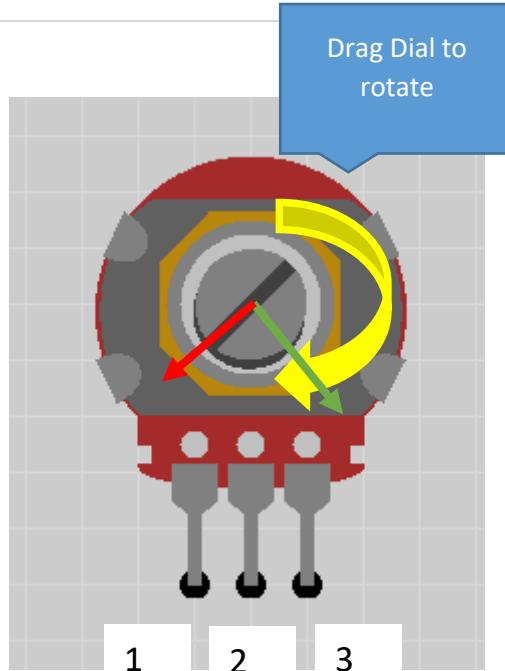
13.1.2 Usage



13.2 Potentiometer

The potentiometer is a voltage divider potentiometer in the common 3 inline pins configuration with an adjustable centre. Dragging the dial changes the position of the centre tap and hence changes the value of the resistor proportions of the voltage divider changing the output voltage proportionally to the rotation position.

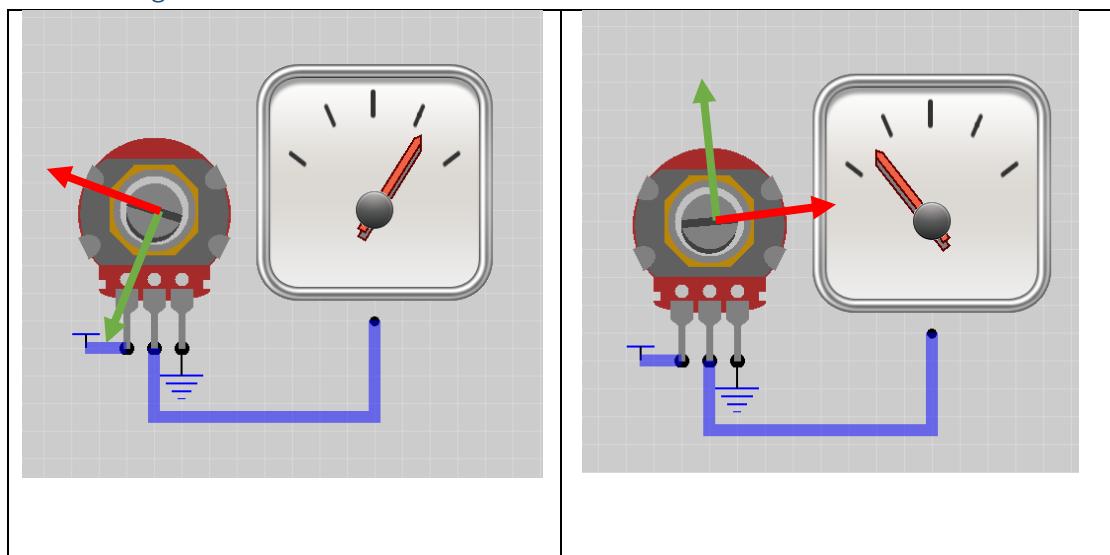
The position of the dial is persisted when saved and restored on open and run.



13.2.1 Pins

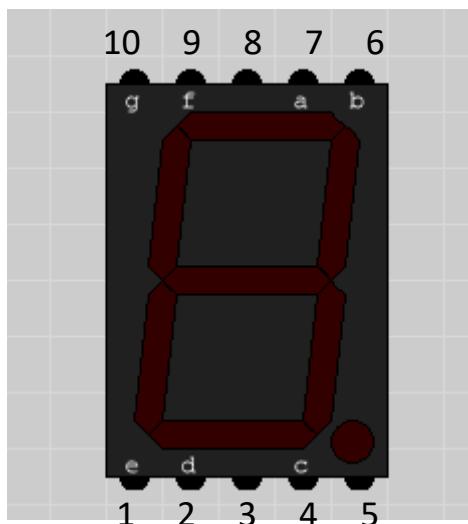
Pin	Name	Description
1	A	Variable resistor pin
2	B	Variable resistor pin
3	Tap	The voltage at TAP is $V1 = (Va-Vb) * \text{position} / \text{MaxPosition}$

13.2.2 Usage



13.3 Segment7

The Segment7 is a Breadboardable version of the Segment7 from the user IO segment 7

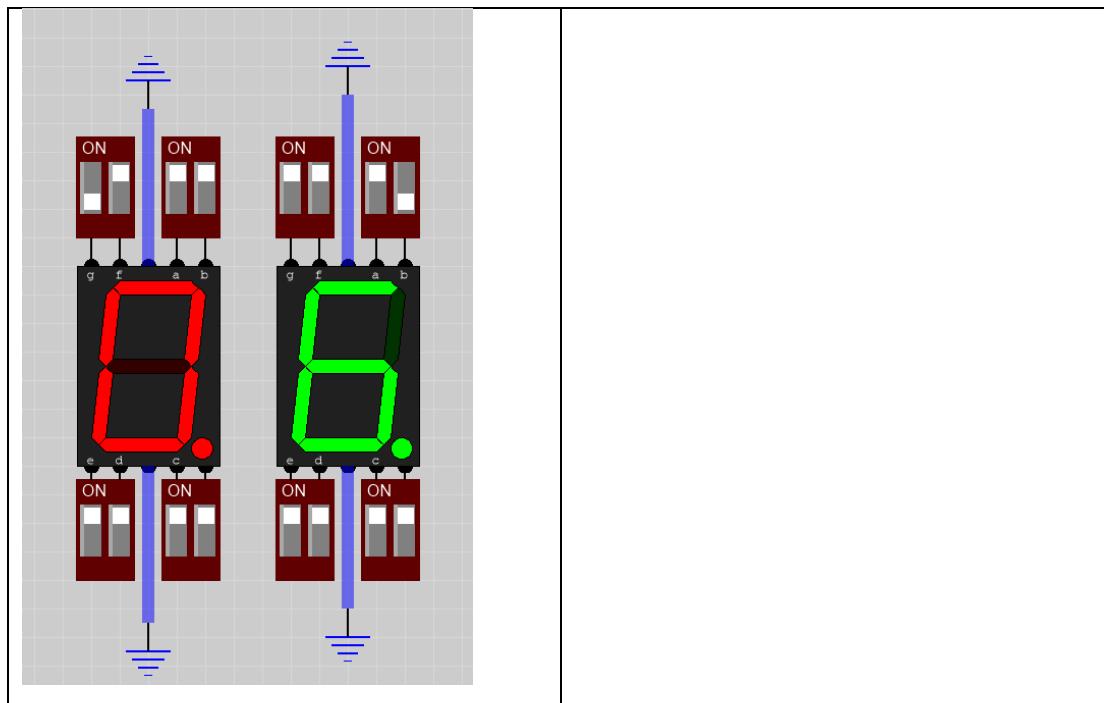


13.3.1 Pins

Pin	Name	Description
1	e	Variable resistor pin
2	d	Variable resistor pin
3		a,d,c,pt common cathode
4	c	
5	pt	
6	B	
7	a	
8		a,d,c,pt common cathode
9	F	
10	g	

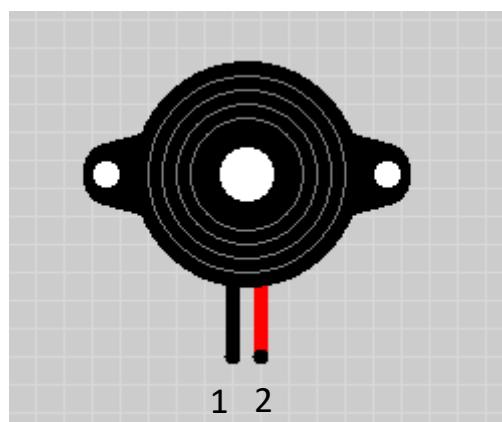
13.3.2 Usage

Setting the common cathode pins to ground you can create different numbers by powering the individual segment pins



13.4 Buzzer

The Buzzer component is an audio component that converts frequencies into audio. Commonly used with tone() or a 555 square wave source.

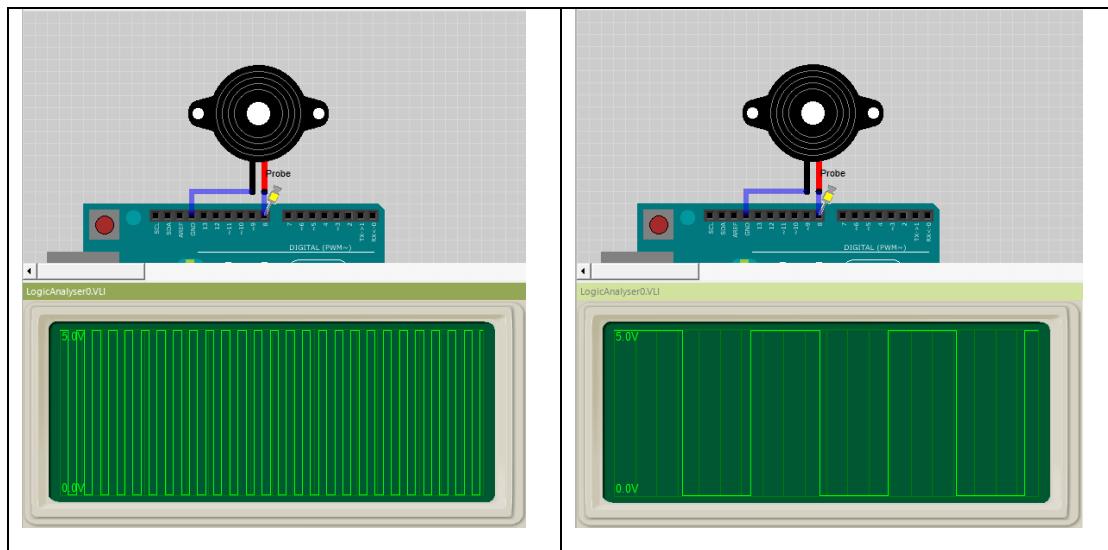


13.4.1 Pins

Pin	Name	Description
1	+	Power pin, connect to frequency source
2	GND	ground

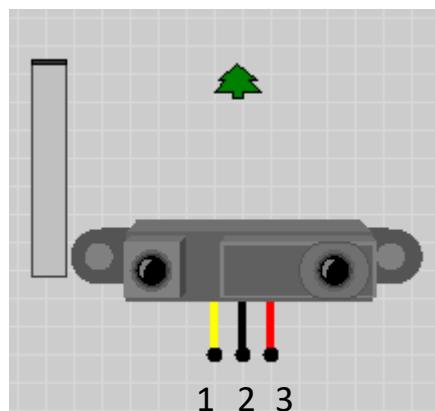
13.4.2 Usage

When used with a frequency source changing the frequency will change the audio tone heard for feedback



13.5 Sharp Proximity Sensor

The sharp proximity sensor is a popular robotics sensor for detecting how close an object is. It is easy to use as it generates a voltage which changes according to how close the senso

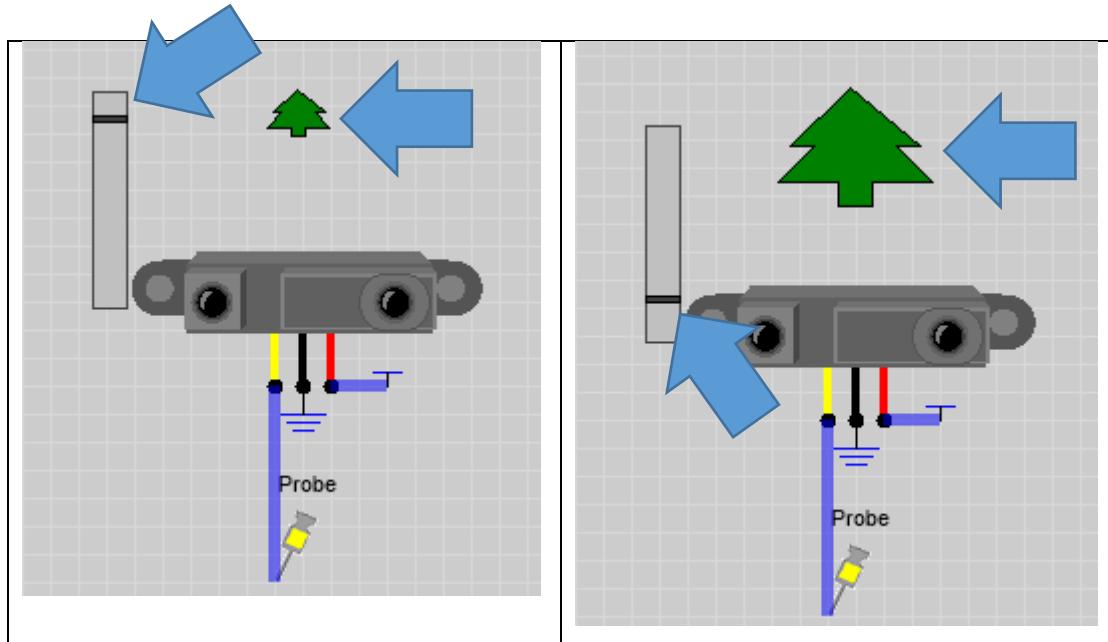


13.5.1 Pins

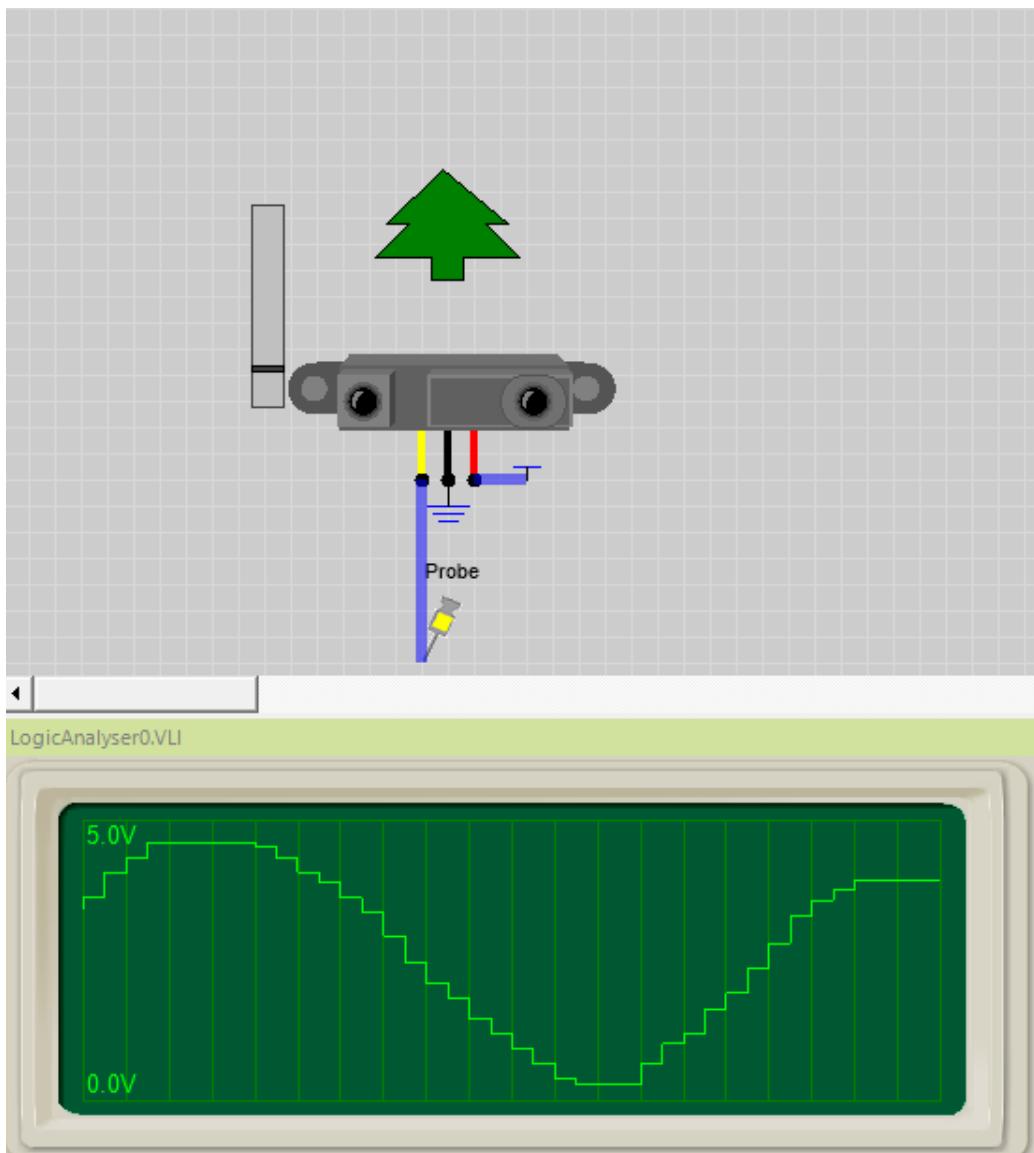
Pin	Name	Description
1	Sig	Analog signal between proportion to the distance to the object. The proportion is not linear in the real world sensor and requires calibration tables for some applications
2	GND	Ground
3	PWR	Power at 5V

13.5.2 Usage

By dragging the widget the sensor simulates proximity which can be visualized by the changing size of the tree which represents how close or far the sensor is from the tree.



Plotted over time you can generate signals that a robot might see as it approaches and moves away from a wall



555 Timer IC

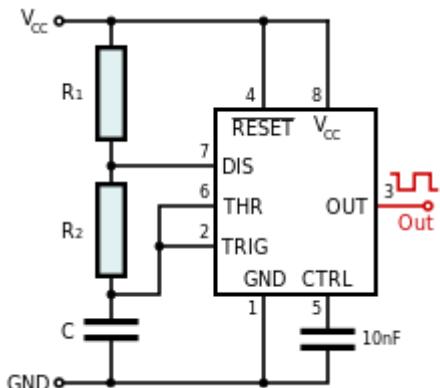
The [555](#) is a versatile chip used in a huge variety of timing, pulse generating and oscillator applications. Although modern microcontrollers have provided alternative ways to implement some of these type of circuits the 555 remains an invaluable part of any tinkerers toolbox.

There are two basic circuit configurations used by the 555 to create distinct functions

Astable - in this mode the 555 operates as an oscillator. Uses include LED and lamp flashers, pulse generation, logic clocks, tone generation and so on.

Monostable - in this mode the 555 operates as a triggered pulse generator. Uses include timers, pulse detection, debouncing, pulse generation and so on.

13.6 Astable

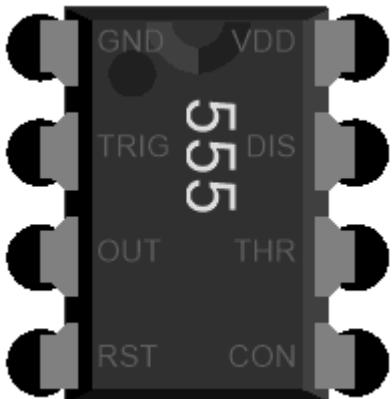


In astable mode, the 555 timer puts out a continuous stream of rectangular pulses having a specified frequency.

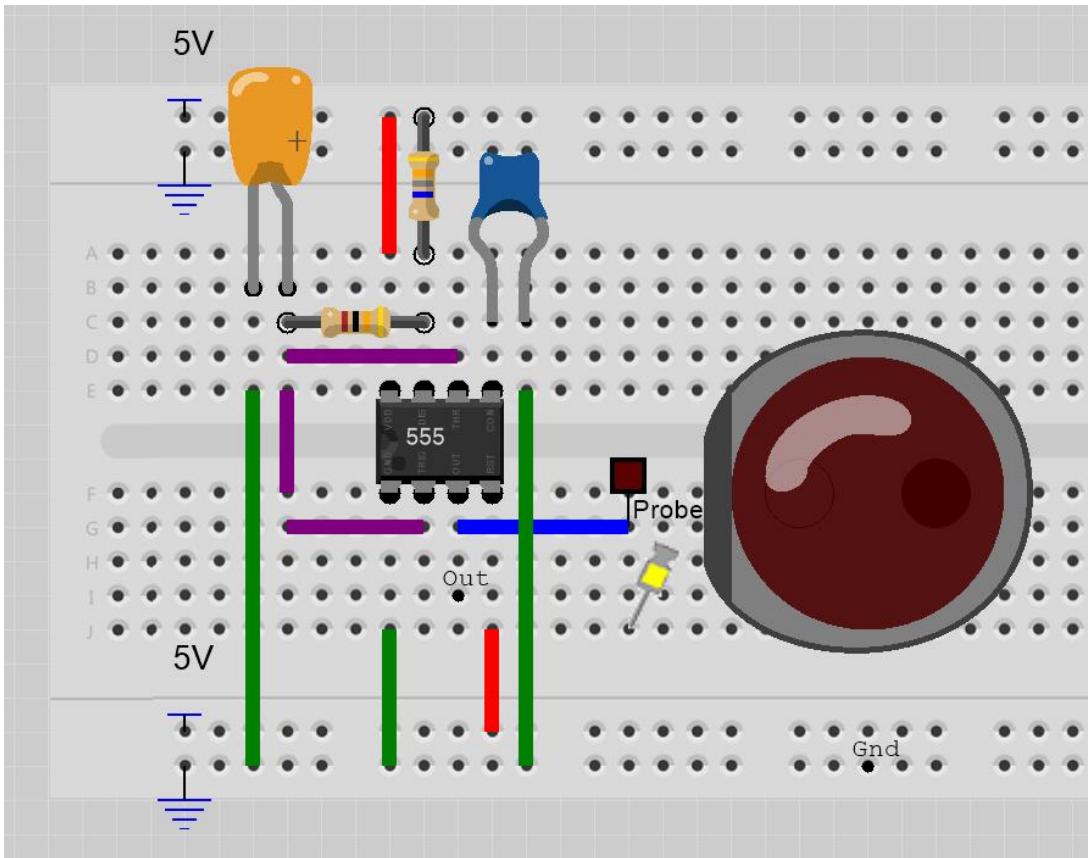
The frequency depends on the values of R1, R2 and C in the above schematic.

There are plenty of articles covering the theory of the AStable 555 operation ([WIKIPEDIA](#) is a good place to start) so I am not going to reproduce that here.

Instead I am going to show you how to use VirtualBreadboard (VBB) to calculate these values for you and show you how to view the resulting output signal.



If you zoom into the VBB 555 component you will see the pins are labelled with the schematic labels. So the Breadboard is a hybrid schematic and Breadboard layout. This makes it easier to wire up a schematic as a Breadboard layout.



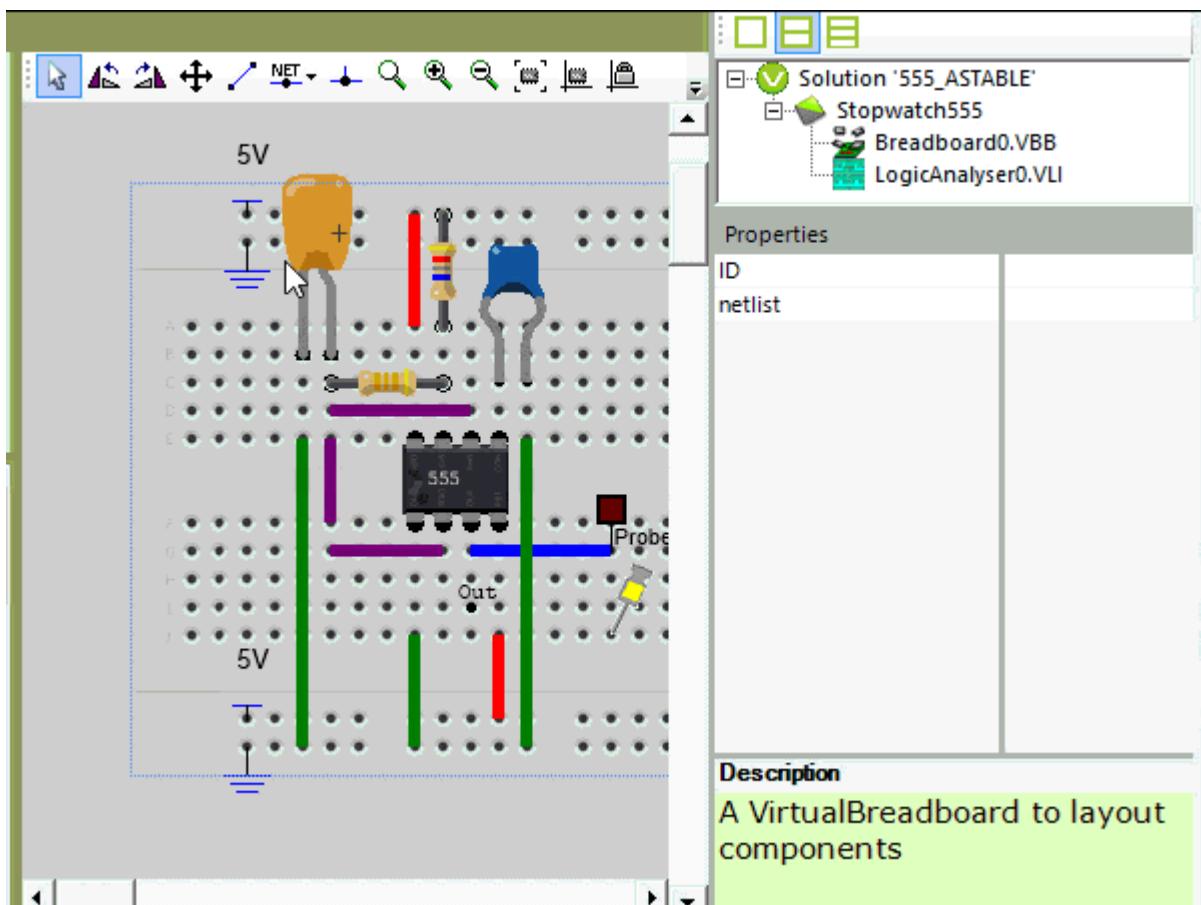
TIP: You can use '**Import from Clipboard**' to Load this project directly into VBB. See 'ASTABLE PROJECT' in the Source Code section below.

13.7 ASTABLE Frequency Calculator

The VBB 555 component has several properties which can be used to determine the emulation behavior of the component

- C
- R1-STABLE
- R2-ASTABLE
- C-MONOSTABLE
- Calculator

To make the 555 generate emulate a astable frequency generator you need to set the values of C,R1,R2 to match the component ID values of the corresponding circuit elements. This will set the value of the calculate property to ASTABLE(..).

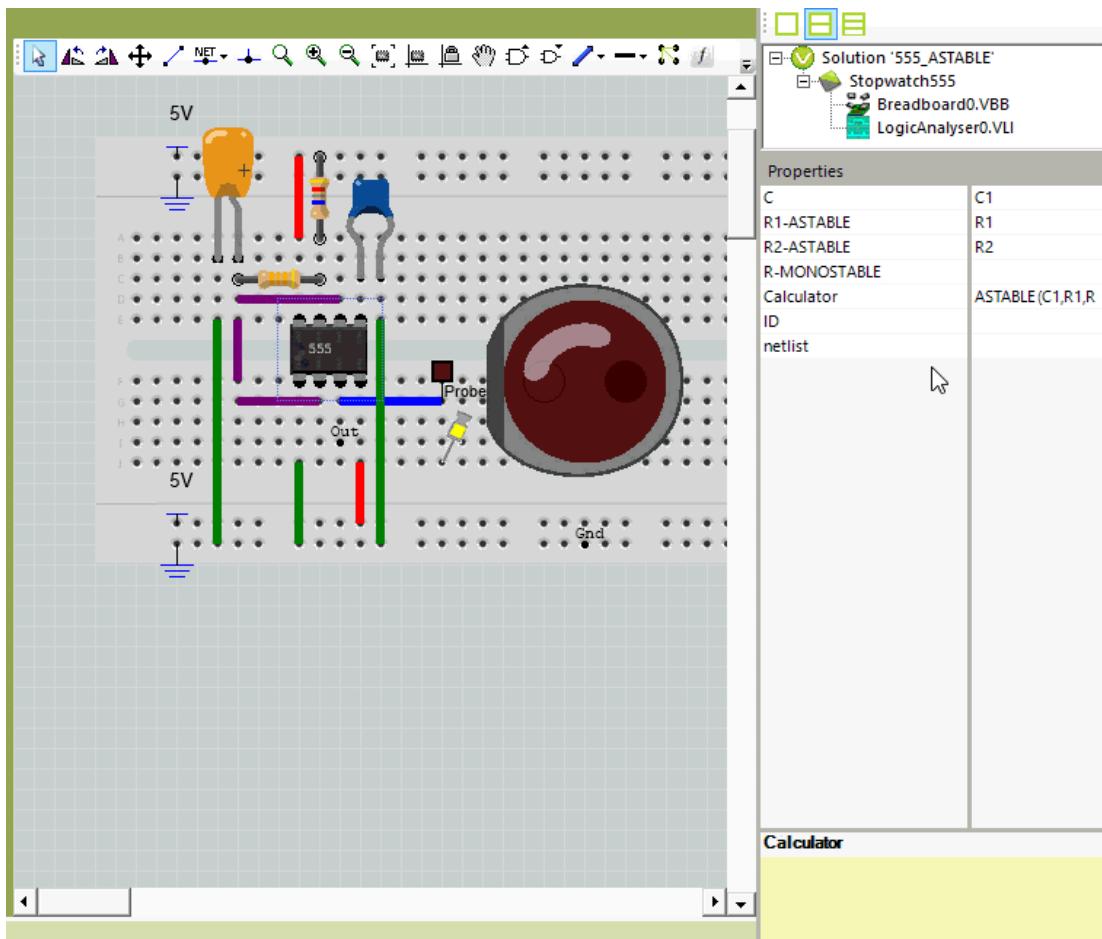


13.8 Calculating the values for AStable

At runtime VBB uses the values for C,R1,R2 are used to calculate the frequency and duty of the 555. You can set the values manually yourself for example if are wiring up and existing design. Alternatively you can use the Calculator properties custom editor to calculate and the set the values for you.

To use the calculator

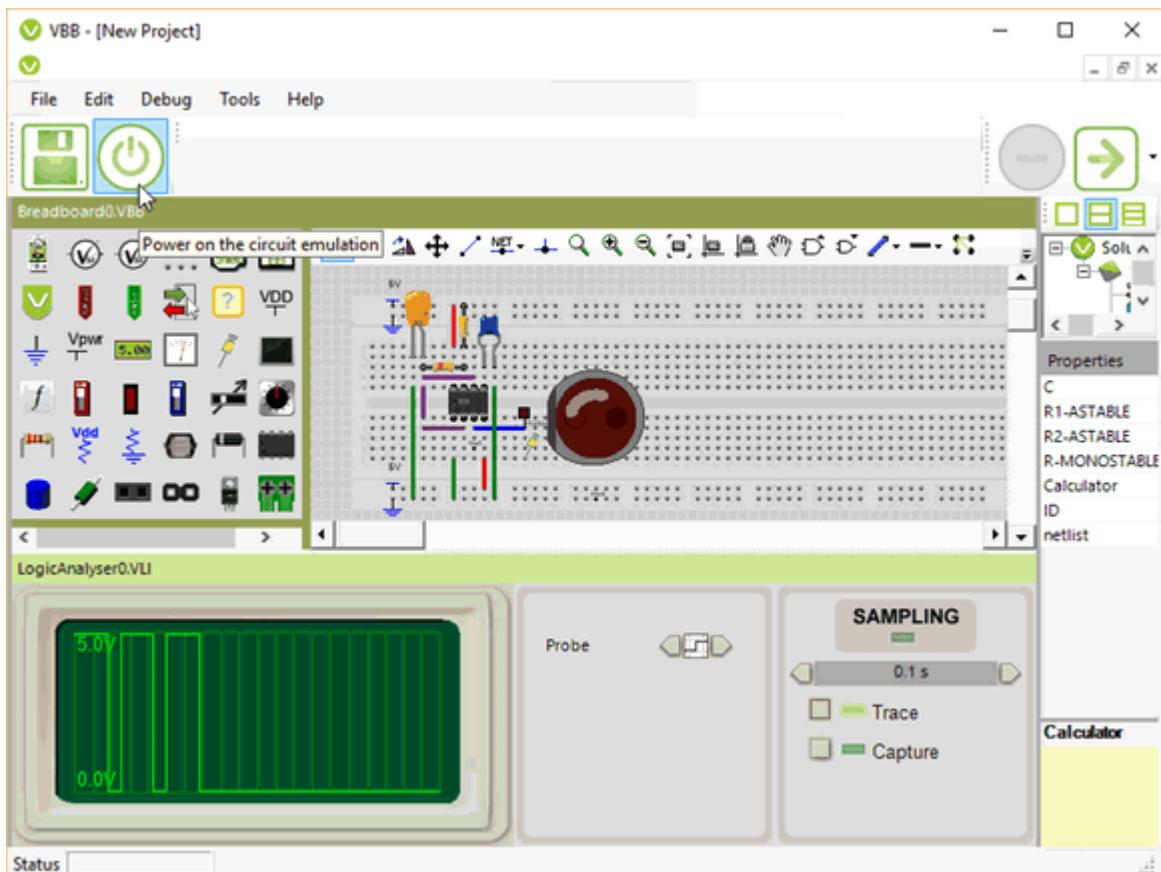
- Click the calculate property and then click the edit button to open the custom calculator dialog
- Enter the target frequency or period and click calculate
- Select your preferred C,R1,R2 values and click OK



13.9 Viewing the AStable Output

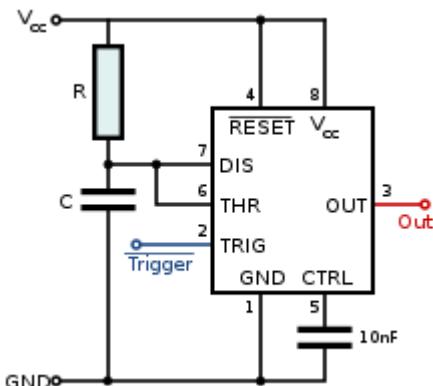
VBB emulations are 'real time' so when you 'Power Up' the circuit you can get a rough idea of the output frequency of the 555 just by looking at a flashing L.E.D if you attach one to the output pin.

This visual check is only useful to about 10 Hz after which the easiest way is to attach a logic probe and view the output waveform in the Logic Analyser. You can change the timebase to the period of the waveform best fits the grid lines making it a simple calculation to find the period.

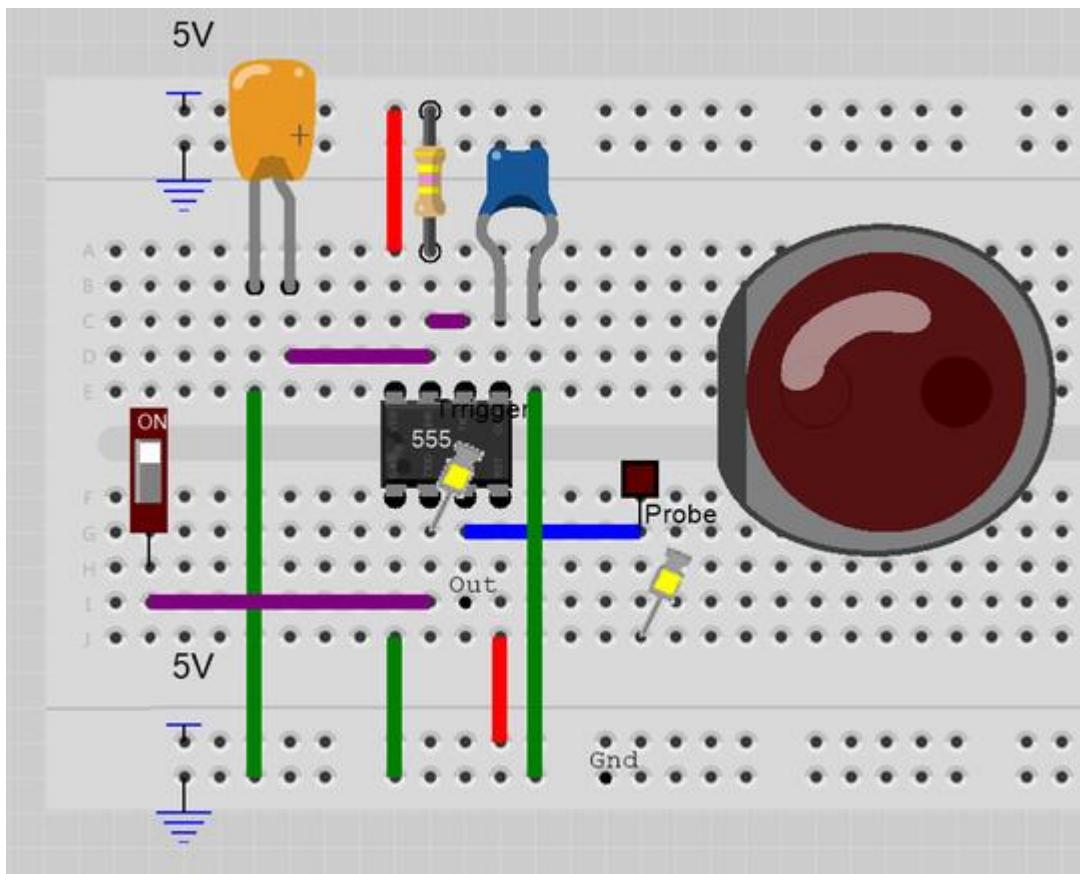


13.10 Monostable

In monostable mode, the 555 timer puts out a triggered 'one shot' pulse.



The duration of the pulse depends on the value of R and C in the above schematic.



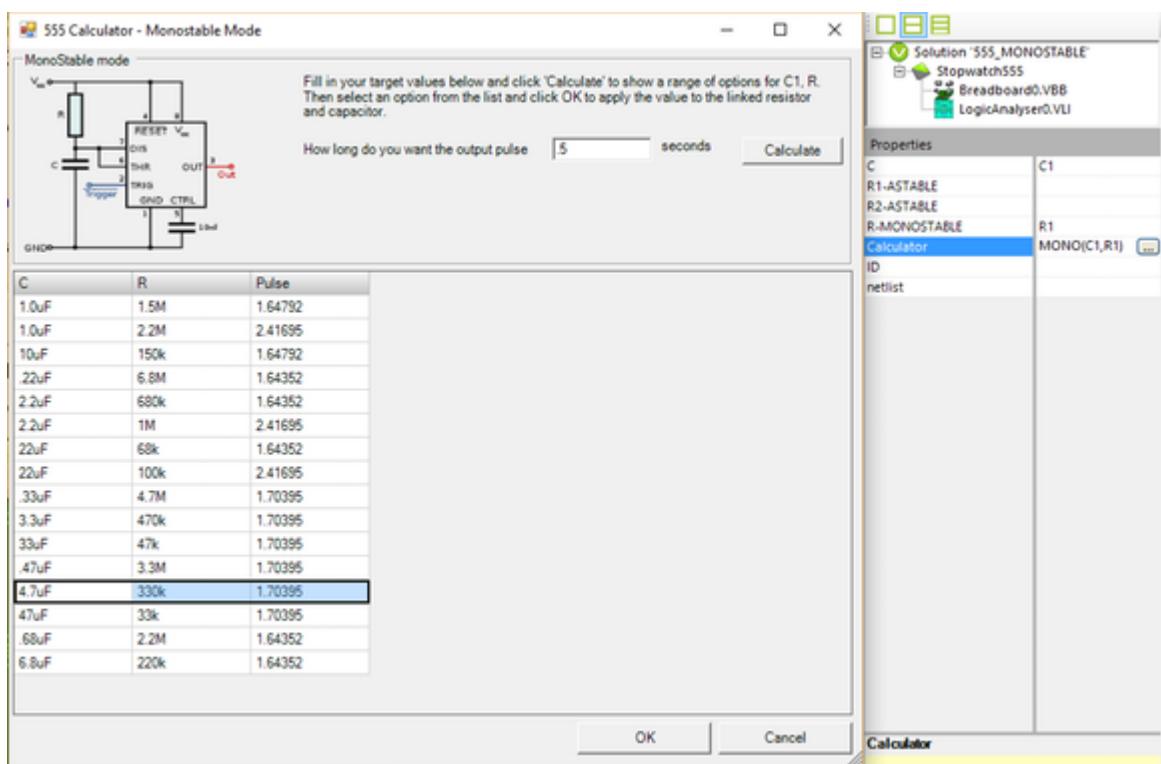
TIP: You can use 'Import from Clipboard' to Load this project directly into VBB. See 'MONOSTABLE PROJECT' in the Source Code section below.

13.11 Calculating the values for Monostable

To make the 555 generate emulate a monostable pulse generator you need to set the values of C and R to match the component ID values of the corresponding circuit elements. This will set the value of the Calculate property as MONO(..)

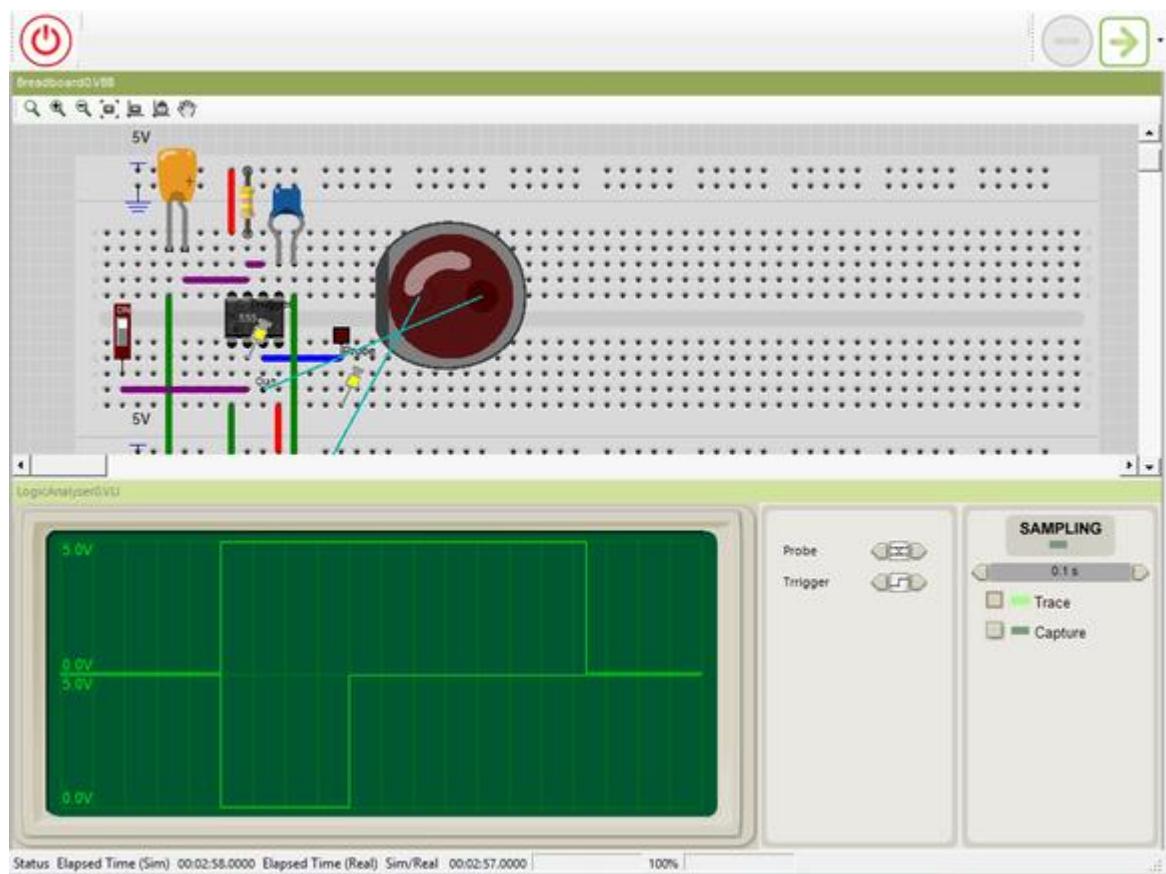
To use the calculator

- Click the calculate property and then click the edit button to open the custom calculator dialog
- Enter the target pulse period and click calculate
- Select your preferred C and R values and click OK.



13.12 Viewing the Monostable Output

You can interact with the virtual circuit. To generate the trigger input click on the DIP switch to create a **falling edge** at the 555 trigger input pin. You should see the L.E.D attached to the output power on for a fixed period. You can also capture the pulse length using the Logic-analyser to get a more accurate view of the pulse width.



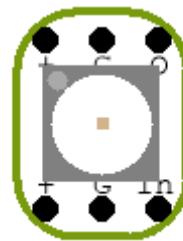
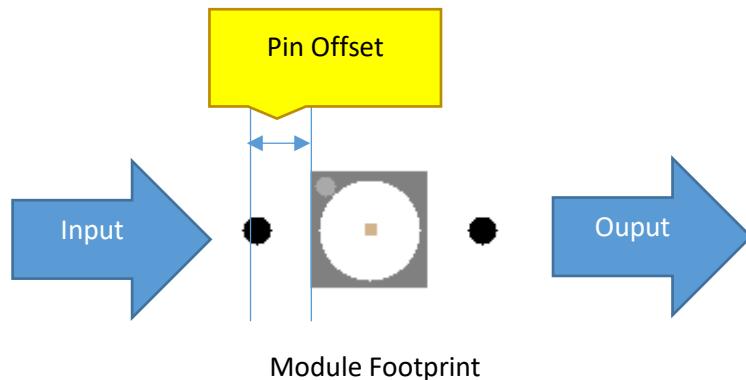
14 API 'Smart' Components

There are two types of components. Signal level components and API components. A signal level component such as Liquid Crystal is modelled based on the signals it receives at its pins. This requires implementing the underlying timing specification of the component.

API components are components that only virtualize when driven via the Java API. The wiring of the component only links the component to the microcontroller. The actual communication to the device is done in an Abstract communications channel. This makes it easier to implement the component virtualization but requires the use of the API to make it work.

14.1 WS2812

The WS2812 is a 'smart' 3 color LED. It can be 'daisy chained' by connecting the output of a module to the input of the next module



Breadboard footprint

Name	Default	Options	Description
------	---------	---------	-------------

Footprint	Breadboard	Breadboard Module	Sets the value of the minimum voltage when the slider is all the way to the left
Pin Offset	0	0 1 2	Sets the pin offset of the module to space out the module layout

14.2 API

[JavaDoc Online](#)

Constructor Summary

Constructors

Constructor and Description

`Adafruit_NeoPixel(Arduino arduino, int n, int pin, int ledType)`

Method Summary

Methods

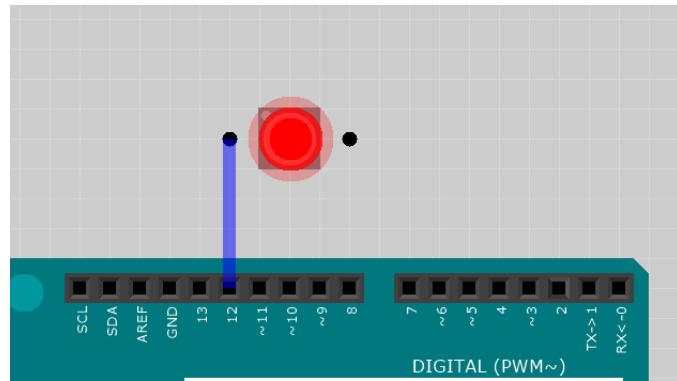
Modifier and Type	Method and Description
void	<code>begin()</code>
void	<code>clear()</code>
long	<code>Color(int r, int g, int b)</code>
int	<code>numPixels()</code>
void	<code>setBrightness(int n)</code>
void	<code>setPin(int p)</code>
void	<code>setPixelColor(int n, int r, int g, int b)</code>
void	<code>setPixelColor(int n, long rgb)</code>
void	<code>show()</code>

```
///////////////
// NEO PIXEL Constants
///////////////
public const int NEO_RGB      = 0x00 ;// Wired for RGB data order
public const int NEO_GRB      = 0x01 ;// Wired for GRB data order
public const int NEO_BRG      = 0x04;

public const int NEO_COLMASK   = 0x01;
public const int NEO_KHZ800    = 0x02; // 800 KHz datastream
public const int NEO_SPDMASK   = 0x02;
```

Usage

1. Place a WS2812
2. Switch to Module footprint
3. Wire the input pin to an output pin, 12
4. Include the API :
5. Instantiate and call



```

import muvium.compatibility.arduino.*;
import muvium.compatibility.arduino.Adafruit_NeoPixel;

public class WS2812 extends Arduino{

    Adafruit_NeoPixel pixels = new Adafruit_NeoPixel( this, 1,
12, NEO_GRB + NEO_KHZ800);

    // The setup() method runs once, when the sketch starts
    public void setup(){

        // Your setup code goes here
        pixels.begin();
        pixels.setPixelColor(0, 255, 0, 0);
        pixels.show();
    }

    // the loop() method runs over and over again,
    // as long as the Arduino has power
    public void loop(){

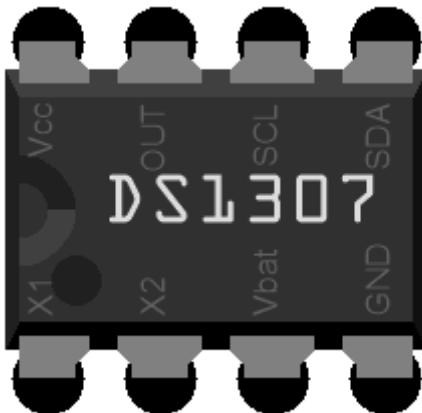
        // Your loop code goes here
    }
}

```

14.3 DS1307

64 x 8, Serial, I2C Real-Time Clock

The DS1307 serial real-time clock (RTC) is a lowpower, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially through an I2C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12hour format with AM/PM indicator. The DS1307 has a built-in power-sense circuit that detects power failures and automatically switches to the backup supply. Timekeeping operation continues while the part operates from the backup supply.



14.3.1 API

Wire

In VBB Wire is an API. The I2C signals are not transmitted over the actual wire but are handled abstractly.

```

import muvium.compatibility.arduino.*;
import muvium.compatibility.arduino.Wire;
import muvium.compatibility.arduino.StringUtils;

class TestDS1307 extends Arduino{

    private static final int DS1307_ADDRESS = 0x68;

    private static final int POS_DAY = 0;
    private static final int POS_MONTH = 3;
    private static final int POS_YEAR = 8;
    private static final int POS_HOUR = 11;
    private static final int POS_MINUTE = 14;
    private static final int POS_SECOND = 17;

    char timeFormat[] = StringUtils.getChars( "dd/mm/yyyy
hh:MM:ss\r\n" );

    // The setup() method runs once, when the sketch starts
    public void setup(){

        Wire.begin(); // initialise the connection
        Serial.begin(9600);

        Serial.write(StringUtils.castChars(timeFormat),
        sizeof(timeFormat));

        timeFormat[POS_YEAR - 2] = '2';
        timeFormat[POS_YEAR - 1] = '0';

        TimeNow();
        Serial.write(StringUtils.castChars(timeFormat),
        sizeof(timeFormat));
        UpdateTime();
        TimeNow();
    }
}

```

```

        Serial.write(StringUtils.castChars(timeFormat),
sizeof(timeFormat));
    }

private void TimeNow(){

    Wire.beginTransmission(DS1307_ADDRESS);
    Wire.write(0);
    Wire.endTransmission();

    Wire.requestFrom(DS1307_ADDRESS, 7);
    writeBCD( Wire.read() & 0x7F, POS_SECOND);
    writeBCD( Wire.read(), POS_MINUTE );
    writeBCD( Wire.read(), POS_HOUR);
    Wire.read();
    writeBCD( Wire.read(), POS_DAY );
    writeBCD( Wire.read(), POS_MONTH );
    writeBCD( Wire.read(), POS_YEAR );

}

private void UpdateTime(){

    Wire.beginTransmission(DS1307_ADDRESS);
    Wire.write(0);
    Wire.write(0); //Seconds
    Wire.write(1); //Minute
    Wire.write(2); //Hour
    Wire.write(0); //Day of Month
    Wire.write(3); //Day
    Wire.write(4); //Month
    Wire.write(5); //Year (2005 )
    Wire.write(0); //Control..
    Wire.endTransmission();

}

private void writeBCD(int val, int pos){
    timeFormat[pos++] = (char)( '0' + ( val >> 4 ) );
    timeFormat[pos++] = (char)( '0' + (val & 0xF ) );
}

// the loop() method runs over and over again,
// as long as the Arduino has power
public void loop(){

    TimeNow();
    Serial.write(StringUtils.castChars(timeFormat),
sizeof(timeFormat));

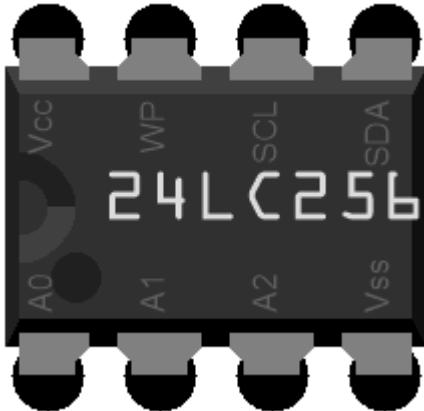
    delay(2000);

}

}

```

14.4 24LC256 EEPROM



The 24LC256 is a 256Kb (32K x 8) Serial Electrically Erasable PROM (EEPROM),

14.4.1 API

Wire

In VBB Wire is an API. The I2C signals are not transmitted over the actual wire but are handled abstractly.

```
import muvium.compatibility.arduino.*;
import muvium.compatibility.arduino.Wire;
import muvium.compatibility.arduino.StringUtils;

public class TestEEPROM extends Arduino{

    void i2c_eeprom_write_byte( int deviceaddress,    int eeaddress,
int data ) {

        Wire.beginTransmission(deviceaddress);
        Wire.write((int) (eeaddress >> 8)); // MSB
        Wire.write((int) (eeaddress & 0xFF)); // LSB
        Wire.write(data);
        Wire.endTransmission();
    }

    // WARNING: address is a page address, 6-bit end will wrap
    // around
    // also, data can be maximum of about 30 bytes, because the
    // Wire library has a buffer of 32 bytes
    void i2c_eeprom_write_page( int deviceaddress,    int
eeaddresspage, byte[] data, int length ) {
        Wire.beginTransmission(deviceaddress);
        Wire.write((int) (eeaddresspage >> 8)); // MSB
        Wire.write((int) (eeaddresspage & 0xFF)); // LSB

        for (int c = 0; c < length; c++)
            Wire.write(data[c]);
        Wire.endTransmission();
    }
}
```

```

    byte i2c_eeprom_read_byte( int deviceaddress,    int eeaddress )
{
    byte rdata = (byte) 0xFF;
    Wire.beginTransmission(deviceaddress);
    Wire.write((int) (eeaddress >> 8)); // MSB
    Wire.write((int) (eeaddress & 0xFF)); // LSB
    Wire.endTransmission();
    Wire.requestFrom(deviceaddress, 1);
    if (Wire.available() !=0) rdata = (byte)Wire.read();
    return rdata;
}

// maybe let's not read more than 30 or 32 bytes at a time!
void i2c_eeprom_read_buffer(int deviceaddress, int eeaddress,
byte[] buffer, int length) {
    Wire.beginTransmission(deviceaddress);
    Wire.write((int) (eeaddress >> 8)); // MSB
    Wire.write((int) (eeaddress & 0xFF)); // LSB
    Wire.endTransmission();
    Wire.requestFrom(deviceaddress, length);
    int c = 0;
    for ( c = 0; c < length; c++ )
        if (Wire.available() !=0) buffer[c] = (byte)
Wire.read();

}

// The setup() method runs once, when the sketch starts
public void setup(){

    // Your setup code goes here
    char eeprom1[] =StringUtils.getChars( "this is data
from the first eeprom" ); // data to write
    char eeprom2[] =StringUtils.getChars( "this is data
from the second eeprom" ); // data to write

    Wire.begin(); // initialise the connection
    Serial.begin(9600);
    i2c_eeprom_write_page(0x50, 0, StringUtils.castChars(
eeprom1 ), sizeof( eeprom1 ) ); // write to EEPROM
    i2c_eeprom_write_page(0x51, 0, StringUtils.castChars(
eeprom2 ), sizeof(eeprom2)); // write to EEPROM

    delay(10); //add a small delay

    Serial.println("Memory written");
}

private void readOutEEPROM(int address){
    int addr = 0; //first address
    byte b = i2c_eeprom_read_byte(address, 0); // access the
first address from the memory

    while (b != 0) {

        Serial.print((char) b); //print content to serial
port
}

```

```

        addr++; //increase address

        b = i2c_eeprom_read_byte(address, addr); //access
an address from the memory
    }

    Serial.println(" ");
}

// the loop() method runs over and over again,
// as long as the Arduino has power
public void loop() {

    readOutEEPROM(0x50);
    readOutEEPROM(0x51);

    delay(2000);

}
}

```

15 Appendix Named Colors

You can enter named colors into any property that requires color.

```

"background" RGB(204,204,204)
"vbb" RGB(164,198,57)
"darkvbb" RGB(120,154,13)
"comment" RGB(255,255,128)
"arduino" RGB(0,152,158)
"darkarduino" RGB(0,120,125)
"aliceblue" RGB(240,248,255)
"antiquewhite" RGB(250,235,215)
"aqua" RGB(0,255,255)
"aquamarine" RGB(127,255,212)
"azure" RGB(240,255,255)
"beige" RGB(245,245,220)
"bisque" RGB(255,228,196)
"black" RGB(0,0,0)
"blanchedalmond" RGB(255,235,205)
"blue" RGB(0,0,255)
"blueviolet" RGB(138,43,226)
"brown" RGB(165,42,42)
"burlywood" RGB(222,184,135)
"cadetblue" RGB(95,158,160)
"chartreuse" RGB(127,255,0)
"chocolate" RGB(210,105,30)
"coral" RGB(255,127,80)
"cornflowerblue" RGB(100,149,237)
"cornsilk" RGB(255,248,220)
"crimson" RGB(220,20,60)
"cyan" RGB(0,255,255)
"darkblue" RGB(0,0,139)
"darkcyan" RGB(0,139,139)
"darkgoldenrod" RGB(184,134,11)

```

```

"darkgray" RGB(169,169,169)
"darkgreen" RGB(0,100,0)
"darkgrey" RGB(169,169,169)
"darkkhaki" RGB(189,183,107)
"darkmagenta" RGB(139,0,139)
"darkolivegreen" RGB(85,107,47)
"darkorange" RGB(255,140,0)
"darkorchid" RGB(153,50,204)
"darkred" RGB(96,0,0) 'getSVGColor=RGB(139,0,0)
"darksalmon" RGB(233,150,122)
"lightpink" RGB(255,182,193)
"lightsalmon" RGB(255,160,122)
"lightseagreen" RGB(32,178,170)
"lightskyblue" RGB(135,206,250)
"lightslategray" RGB(119,136,153)
"lightslategrey" RGB(119,136,153)
"lightsteelblue" RGB(176,196,222)
"lightyellow" RGB(255,255,224)
"lime" RGB(0,255,0)
"limegreen" RGB(50,205,50)
"linen" RGB(250,240,230)
"magenta" RGB(255,0,255)
"maroon" RGB(128,0,0)
"mediumaquamarine" RGB(102,205,170)
"mediumblue" RGB(0,0,205)
"mediumorchid" RGB(186,85,211)
"mediumpurple" RGB(147,112,219)
"mediumseagreen" RGB(60,179,113)
"mediumslateblue" RGB(123,104,238)
"mediumspringgreen" RGB(0,250,154)
"mediumturquoise" RGB(72,209,204)
"mediumvioletred" RGB(199,21,133)
"midnightblue" RGB(25,25,112)
"mintcream" RGB(245,255,250)
"mistyrose" RGB(255,228,225)
"moccasin" RGB(255,228,181)
"navajowhite" RGB(255,222,173)
"navy" RGB(0,0,128)
"oldlace" RGB(253,245,230)
"olive" RGB(128,128,0)
"olivedrab" RGB(107,142,35)
"orange" RGB(255,165,0)
"orangered" RGB(255,69,0)
"lightcoral" RGB(240,128,128)
"lightcyan" RGB(224,255,255)
"lightgoldenrodyellow" RGB(250,250,210)
"lightgray" RGB(211,211,211)
"lightgreen" RGB(144,238,144)
"lightgrey" RGB(211,211,211)
"violet" RGB(238,130,238)
"wheat" RGB(245,222,179)
"white" RGB(255,255,255)
"whitesmoke" RGB(245,245,245)
"yellow" RGB(255,255,0)
"yellowgreen" RGB(154,205,50)
"darkseagreen" RGB(143,188,143)
"darkslateblue" RGB(72,61,139)
"darkslategray" RGB(47,79,79)
"darkslategrey" RGB(47,79,79)
"darkturquoise" RGB(0,206,209)
"darkviolet" RGB(148,0,211)

```

```

"deeppink" RGB(255,20,147)
"deepskyblue" RGB(0,191,255)
"dimgray" RGB(105,105,105)
"dimgrey" RGB(105,105,105)
"dodgerblue" RGB(30,144,255)
"firebrick" RGB(178,34,34)
"floralwhite" RGB(255,250,240)
"forestgreen" RGB(34,139,34)
"fuchsia" RGB(255,0,255)
"gainsboro" RGB(220,220,220)
"ghostwhite" RGB(248,248,255)
"gold" RGB(255,215,0)
"goldenrod" RGB(218,165,32)
"Gray" RGB(128,128,128)
"grey" RGB(128,128,128)
"green" RGB(0,128,0)
"greenyellow" RGB(173,255,47)
"honeydew" RGB(240,255,240)
"hotpink" RGB(255,105,180)
"indianred" RGB(205,92,92)
"indigo" RGB(75,0,130)
"ivory" RGB(255,255,240)
"khaki" RGB(240,230,140)
"lavender" RGB(230,230,250)
"lavenderblush" RGB(255,240,245)
"lawngreen" RGB(124,252,0)
"lemonchiffon" RGB(255,250,205)
"lightblue" RGB(173,216,230)
"orchid" RGB(218,112,214)
"palegoldenrod" RGB(238,232,170)
"palegreen" RGB(152,251,152)
"paleturquoise" RGB(175,238,238)
"palevioletred" RGB(219,112,147)
"papayawhip" RGB(255,239,213)
"peachpuff" RGB(255,218,185)
"peru" RGB(205,133,63)
"pink" RGB(255,192,203)
"plum" RGB(221,160,221)
"powderblue" RGB(176,224,230)
"purple" RGB(128,0,128)
"red" RGB(255,0,0)
"rosybrown" RGB(188,143,143)
"royalblue" RGB(65,105,225)
"saddlebrown" RGB(139,69,19)
"salmon" RGB(250,128,114)
"sandybrown" RGB(244,164,96)
"seagreen" RGB(46,139,87)
"seashell" RGB(255,245,238)
"sienna" RGB(160,82,45)
"silver" RGB(192,192,192)
"skyblue" RGB(135,206,235)
"slateblue" RGB(106,90,205)
"slategray" RGB(112,128,144)
"slategrey" RGB(112,128,144)
"snow" RGB(255,250,250)
"springgreen" RGB(0,255,127)
"steelblue" RGB(70,130,180)
"Tan" RGB(210,180,140)
"teal" RGB(0,128,128)
"thistle" RGB(216,191,216)
"tomato" RGB(255,99,71)

```

"turquoise" RGB(64,224,208)