Using Sedona 1.2 Components from Tridium's Kits

Introduction

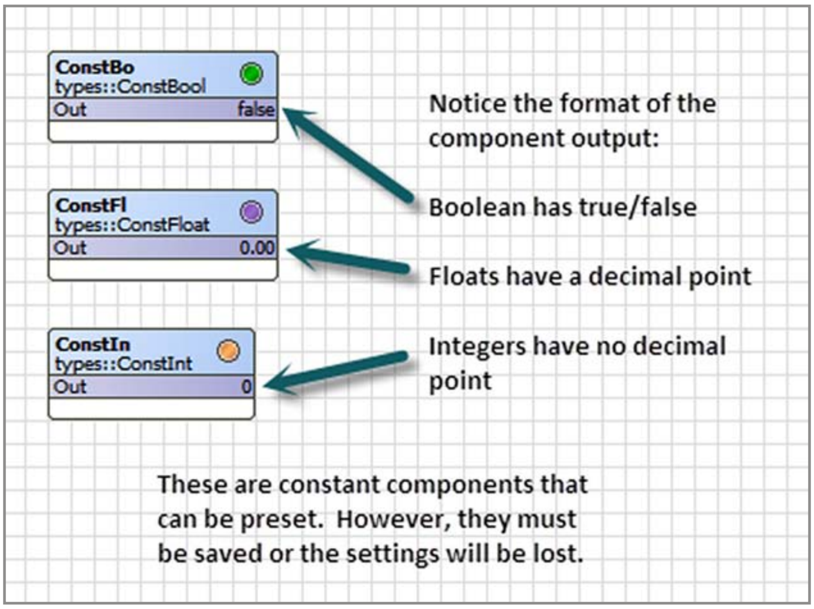
This application note assists in the understanding of the Sedona components provided in Tridium's Sedona-1.2.28 release. Some of the Sedona components were changed or added since the previous release. New with the 1.2 release is that the Sedona components, previously concentrated in one Control kit, are now organized in smaller kits under a functional name. Components discussed in this document can be found in the following kits:

- basicSchedule
- datetimeSTD
- func
- hvac
- logic
- math
- pricomp
- timing
- types

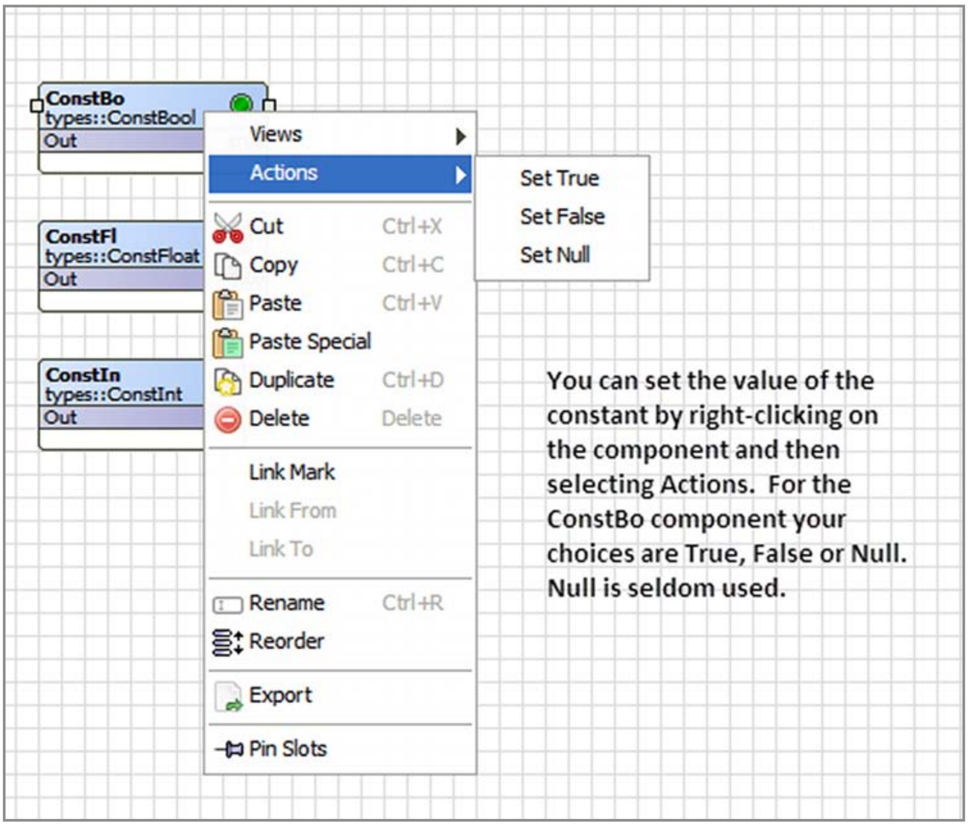
The intent of this document is to explain the potential use of those components supplied by Tridium in their Sedona 1.2 release. All are included in Contemporary Controls' BASremote and BAScontrol product families. They have not been modified for use in these products. Contemporary Controls has product specific Sedona kits that address the uniqueness of the IO structure in the BASremote and BAScontrol products. These kits are not mentioned in this document. It is Contemporary Controls' policy to provide all Sedona kits to the Sedona Framework community without charge or license. This includes kits obtained from Tridium, kits with modified Tridium components, kits developed solely by Contemporary Controls to improve the control options available to systems integrators, and kits specific to Contemporary Controls' hardware. Any feedback is welcomed.

Variable Types

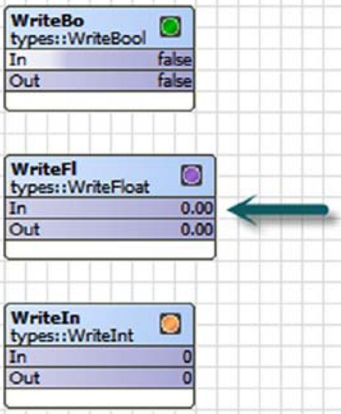
Although there are several variable types used by Sedona, three are the most interesting — Boolean, Float and Integer. You can define constants for each type and use converting components to change the data representation to a different type.



Configuring Constants

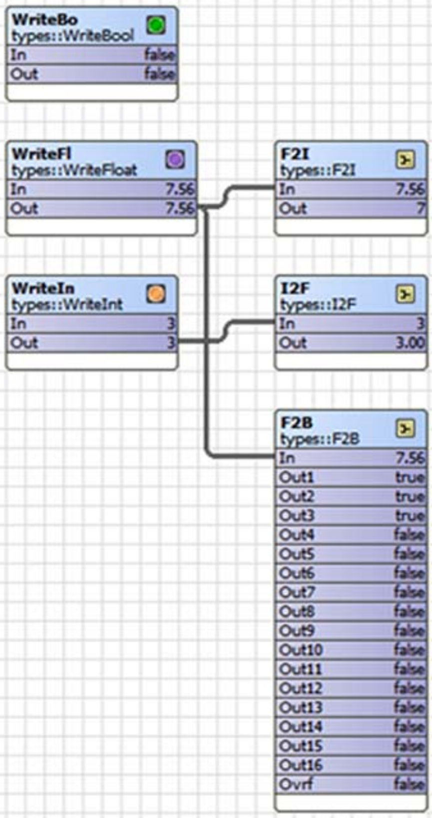


Using Write Components



In a similar manner there are write components for each variable type. Unlike the constant components, these write components have an input pin. The value of the input will be saved if the application program is saved. Other than the input pin difference, the constant components and the write components function the same.

Converting Between Component Types



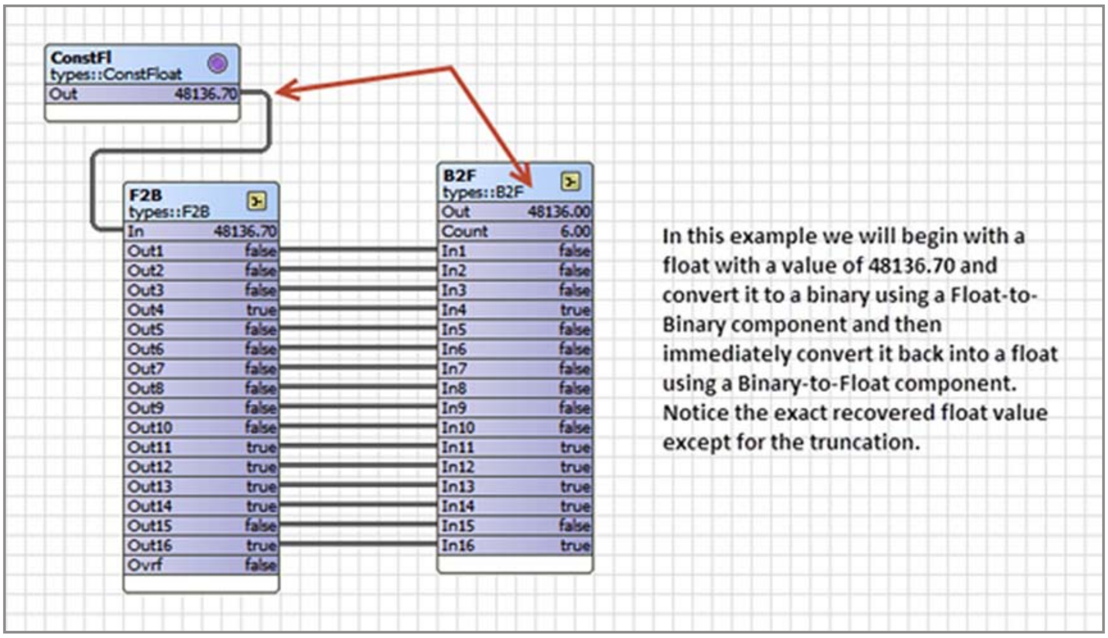
Float-to-Integer and Integer-to-Float components exist. Notice that when we converted from a float to an integer that the Float-to-Integer component truncated the original value during the conversion.

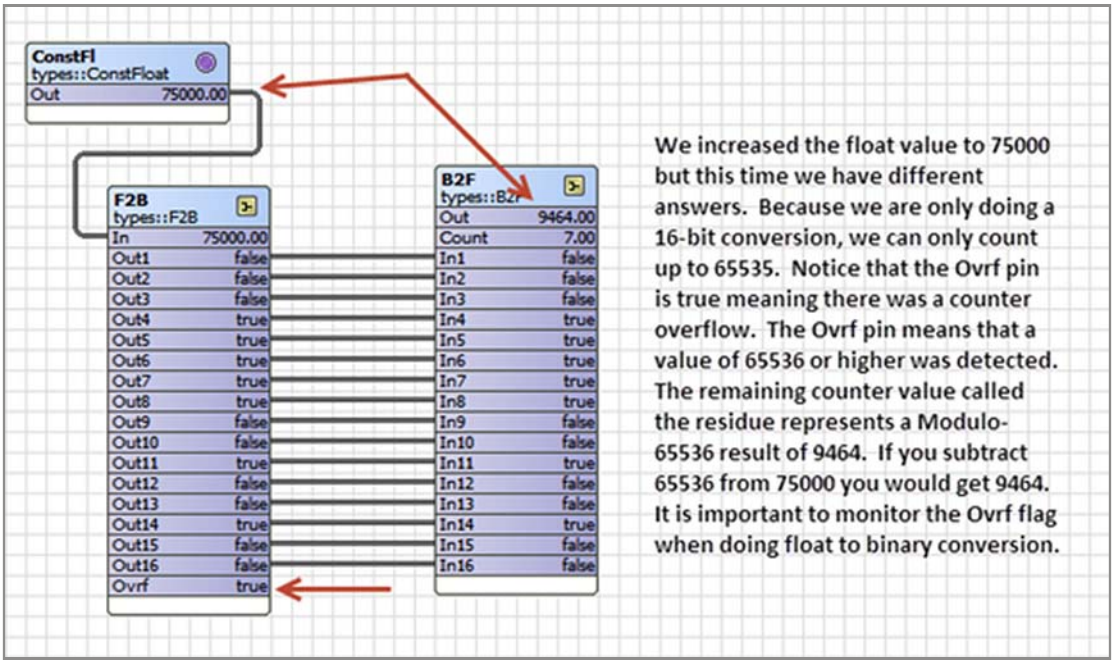
Although it appears that an Integer-to-Float conversion created a much higher accuracy of the original value, this is not the case. The ability to convert variable types is necessary because not all Sedona components exist for each variable type

You can also convert a float to a binary using the Float-to-Binary component. However, notice that the resulting 0000 0000 0000 0111 binary representation is actually a decimal 7 and that again the original float value was truncated.

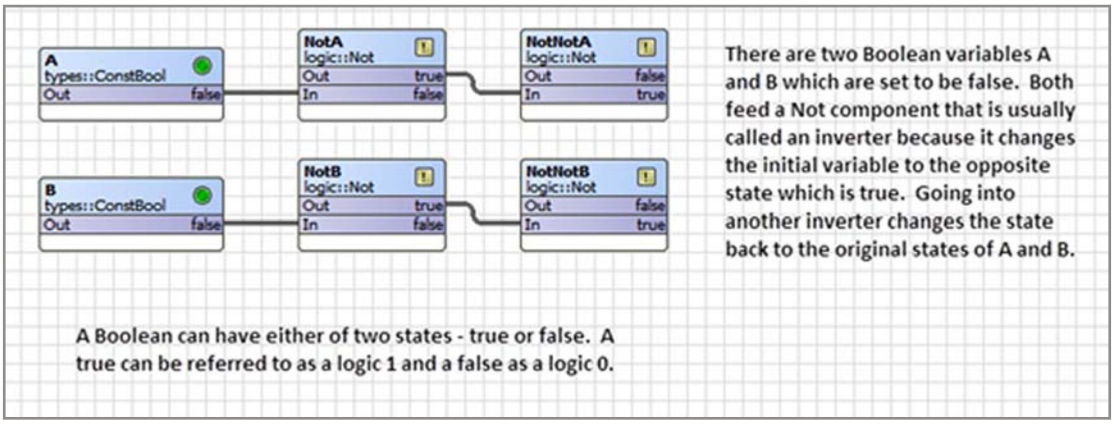
There is no Integer-to-Binary component but this could be accommodated by using an Integer-to-Float ahead of the F2B component.

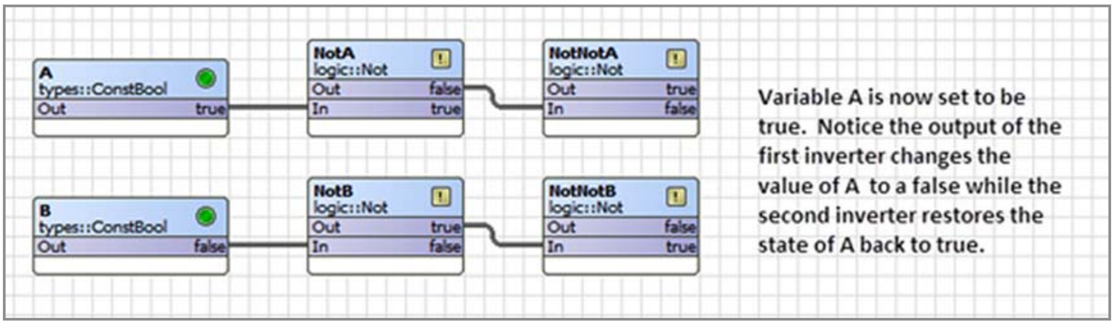
Float-to-Boolean and Boolean-to-Float Conversion



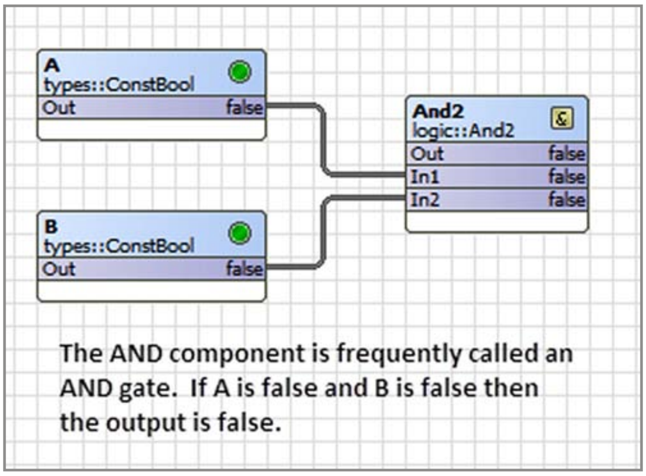


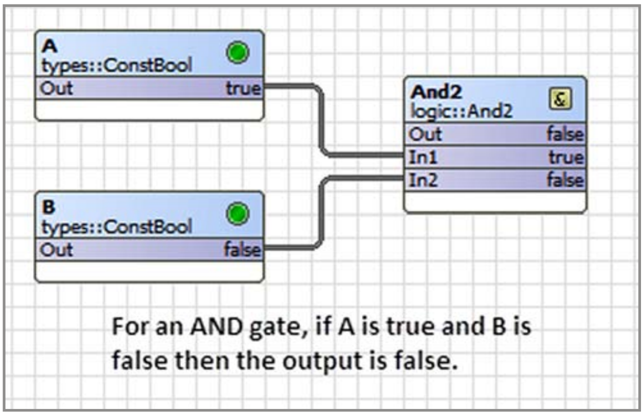
Negating a Boolean Variable — Inverting Your Logic

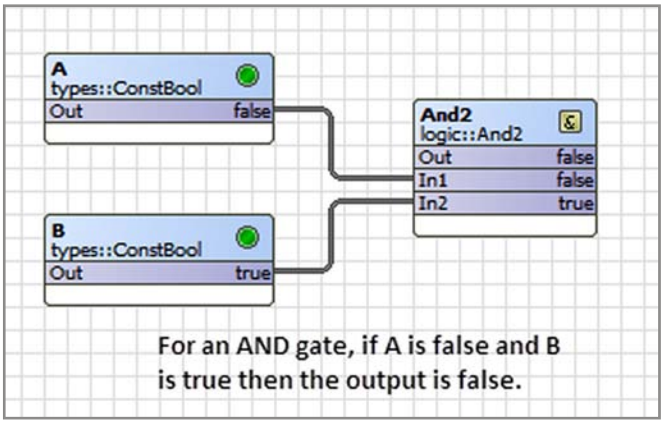


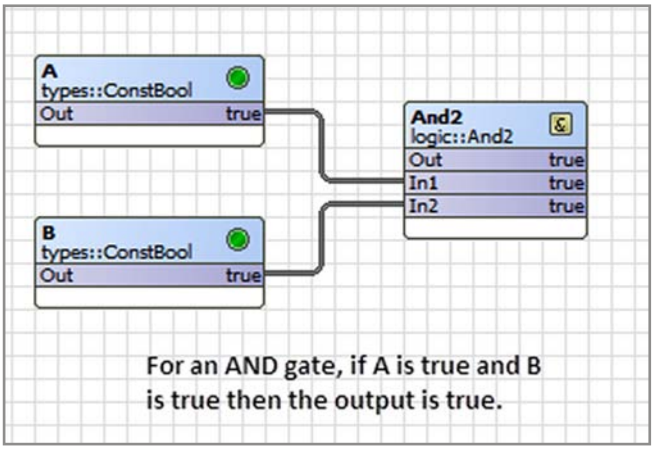


Boolean Product — "ANDing" Boolean Variables

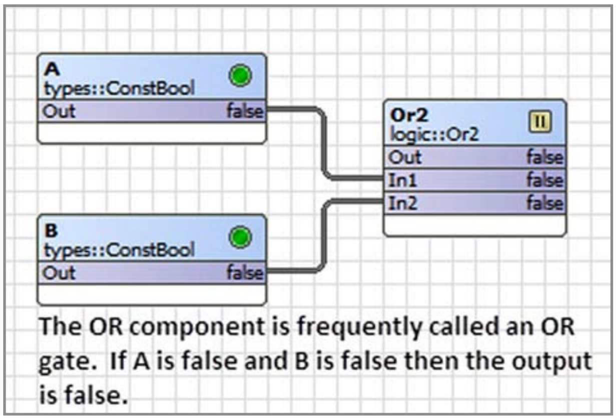


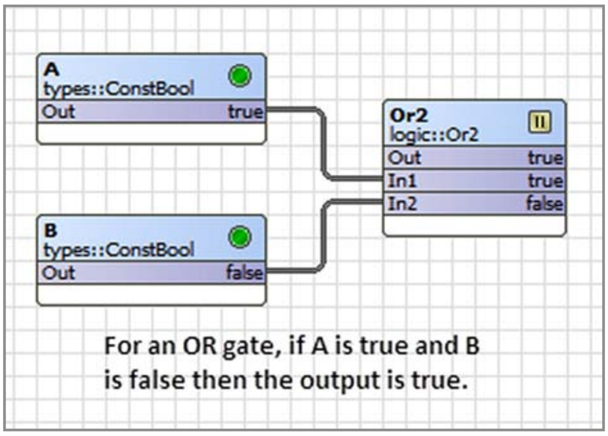


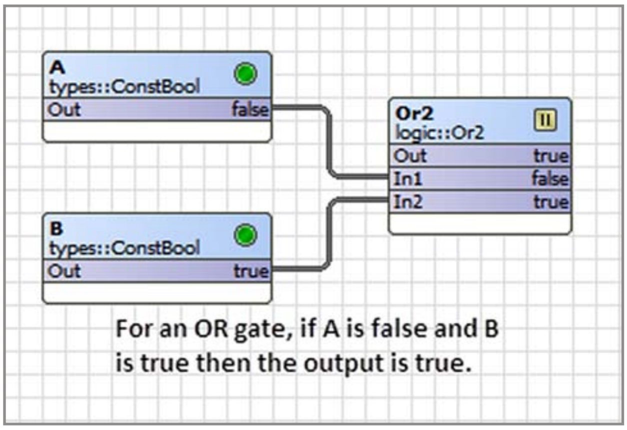


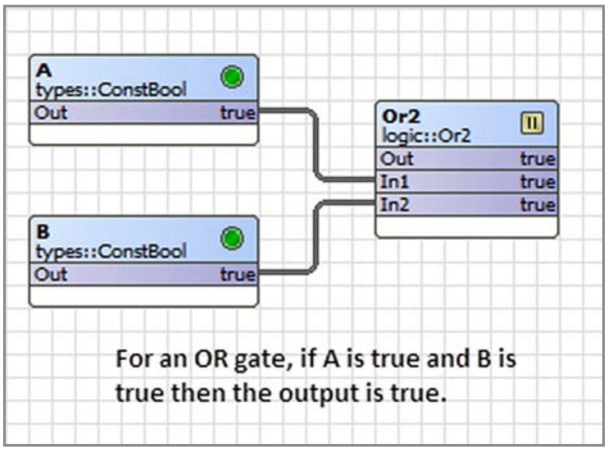


Boolean Sum — "Oring" Boolean Variables

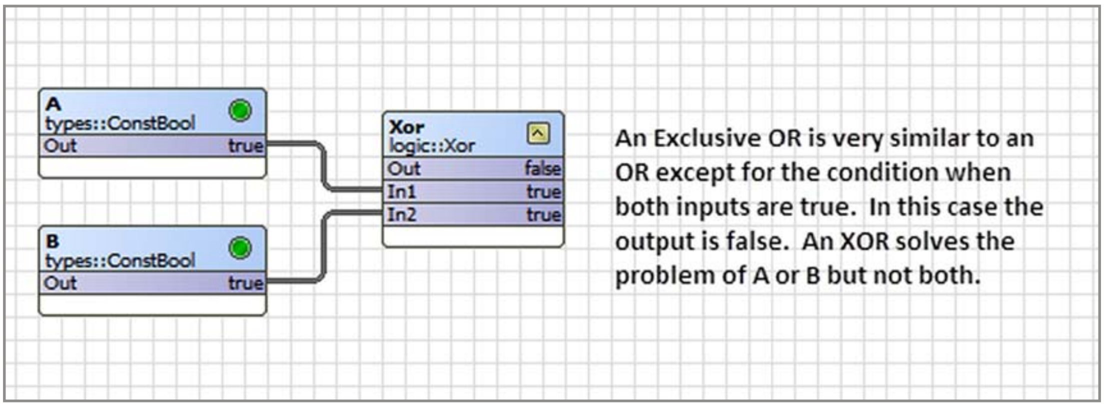




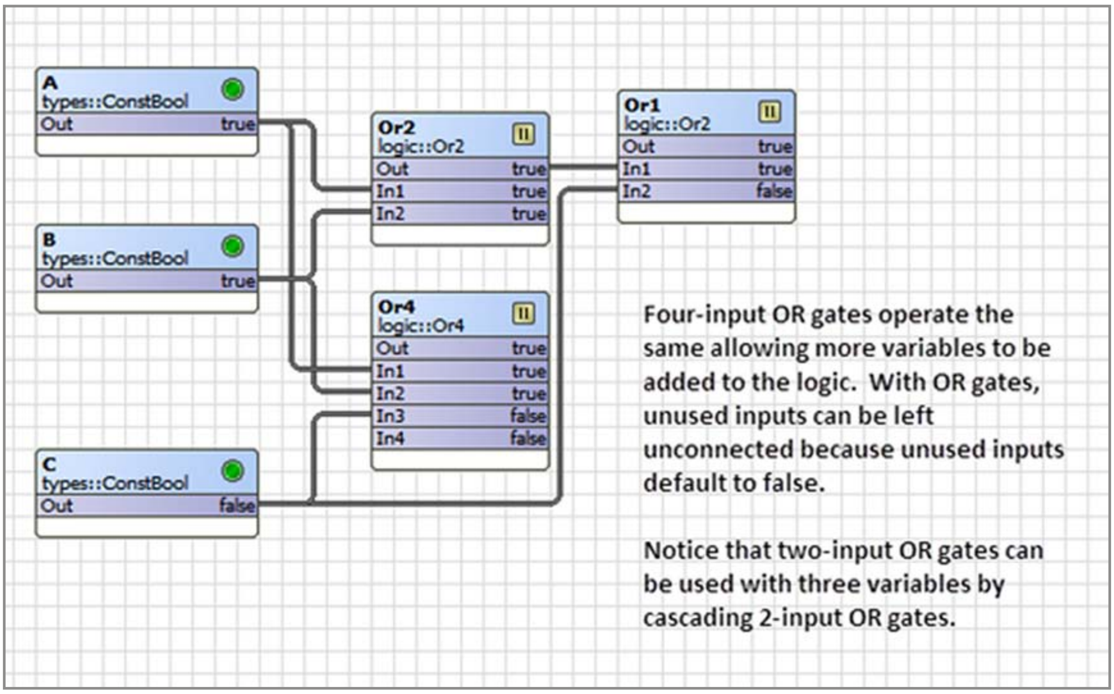




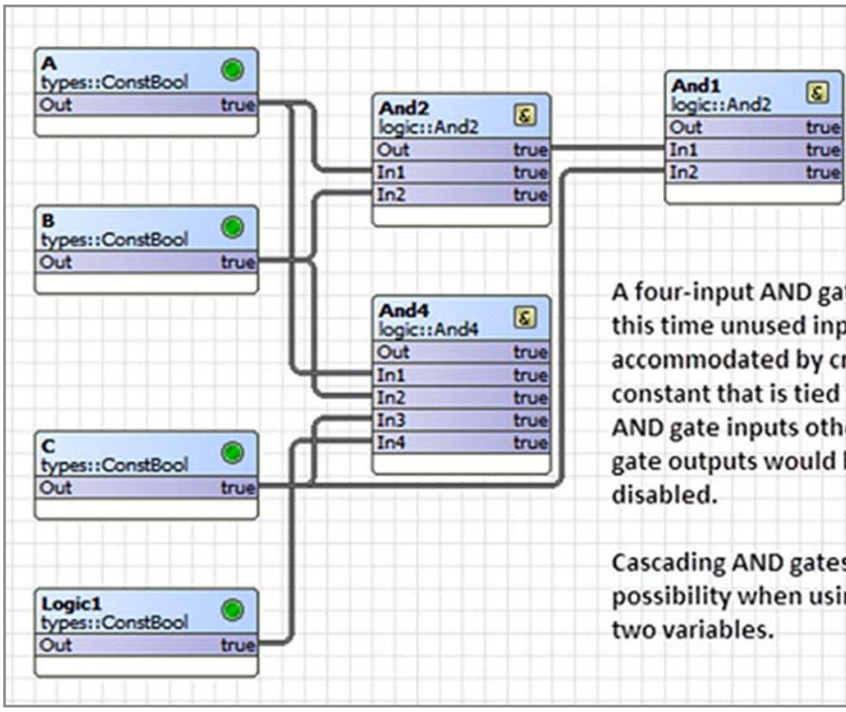
Exclusive OR — A OR B but Not Both



Cascading Logic Blocks and Unused Inputs

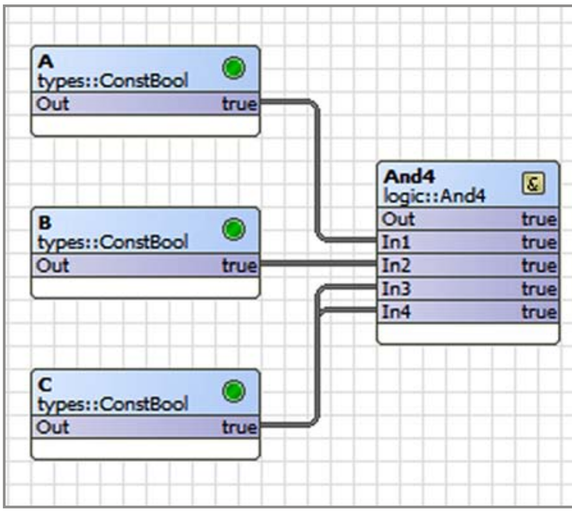


Cascading Logic Blocks and Unused Inputs (continued)



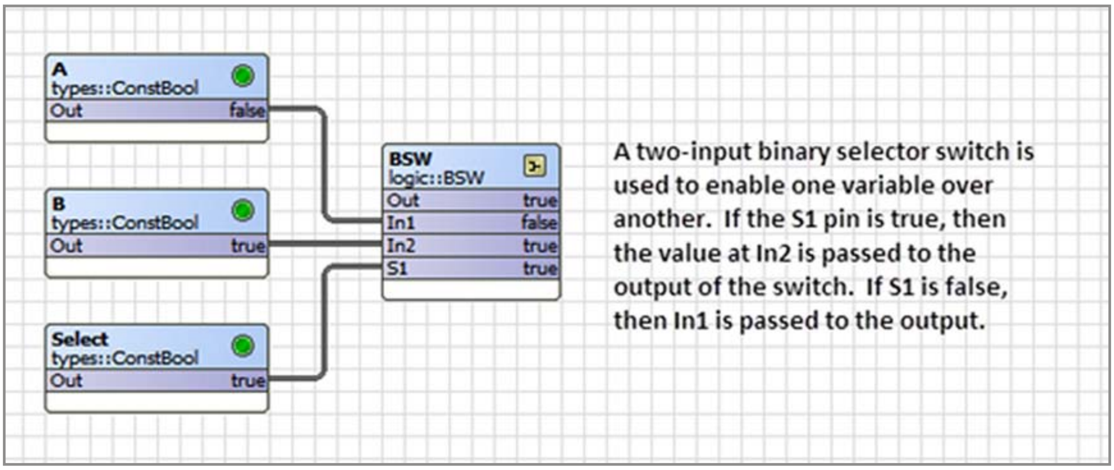
A four-input AND gate exists but this time unused inputs must be accommodated by creating a Logic1 constant that is tied to all unused AND gate inputs otherwise the AND gate outputs would be permanently disabled.

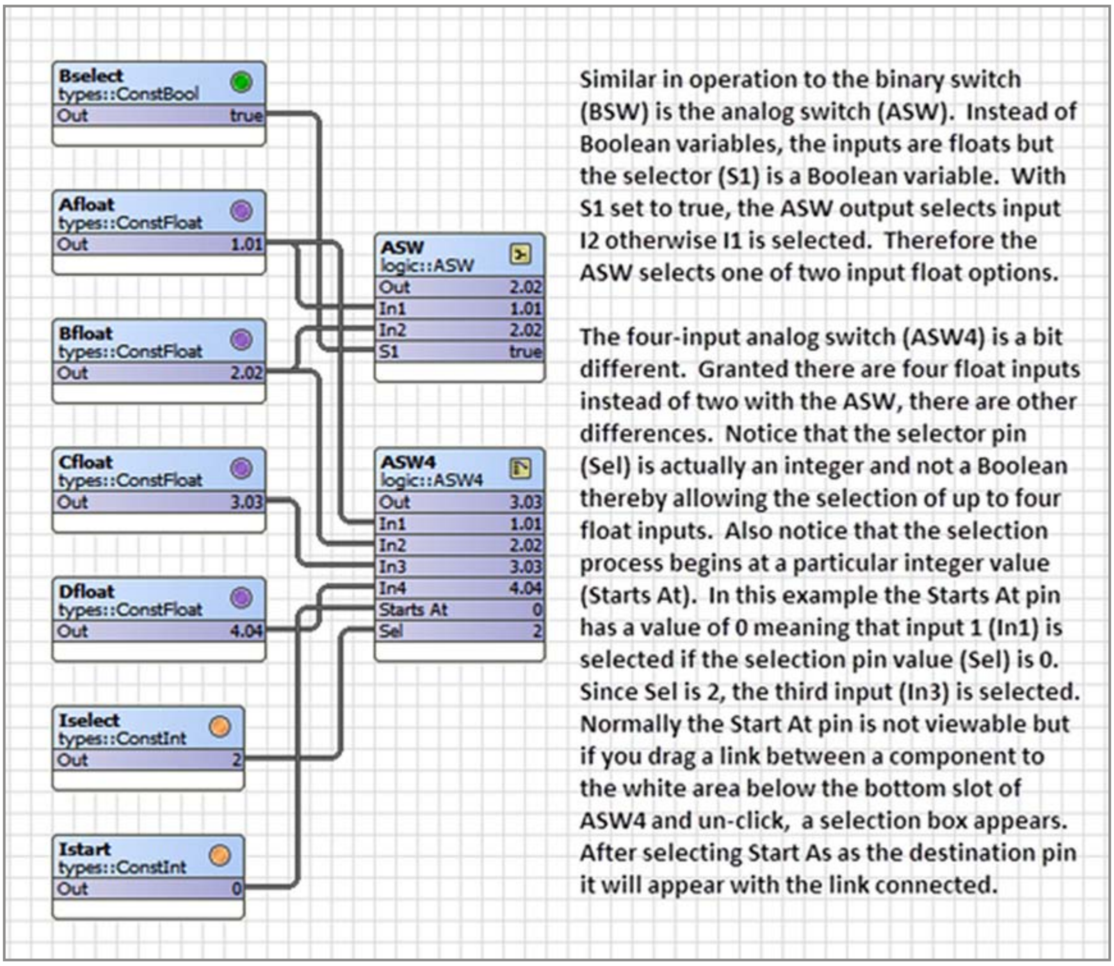
Cascading AND gates is also a possibility when using more than two variables.



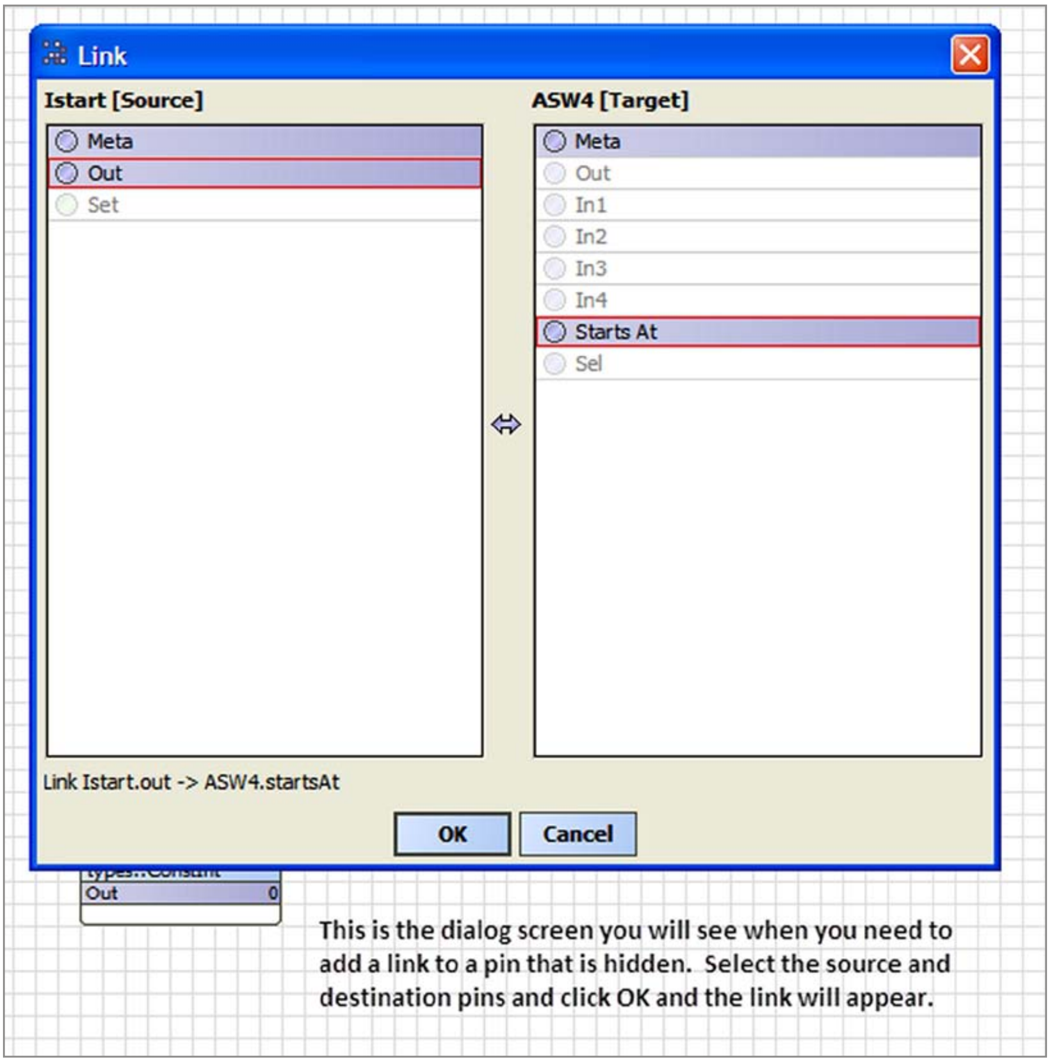
Here is another way of handling an unused input when three variables are connected to a four-input AND gate. Attach the unused input to one of the variables. It does not matter which one is used.

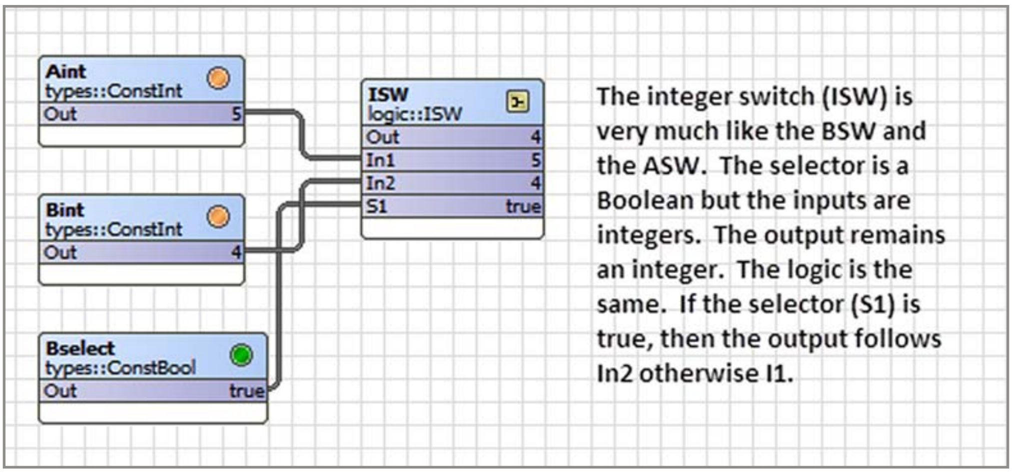
Boolean, Float or Integer Selection



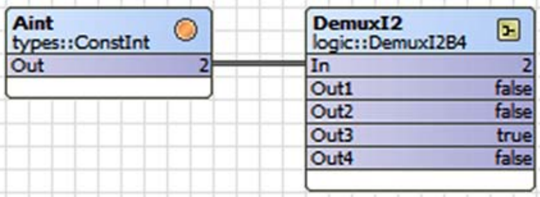


Boolean, Float or Integer Selection (continued)

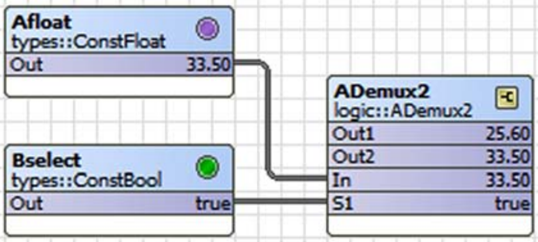




De-Multiplexing

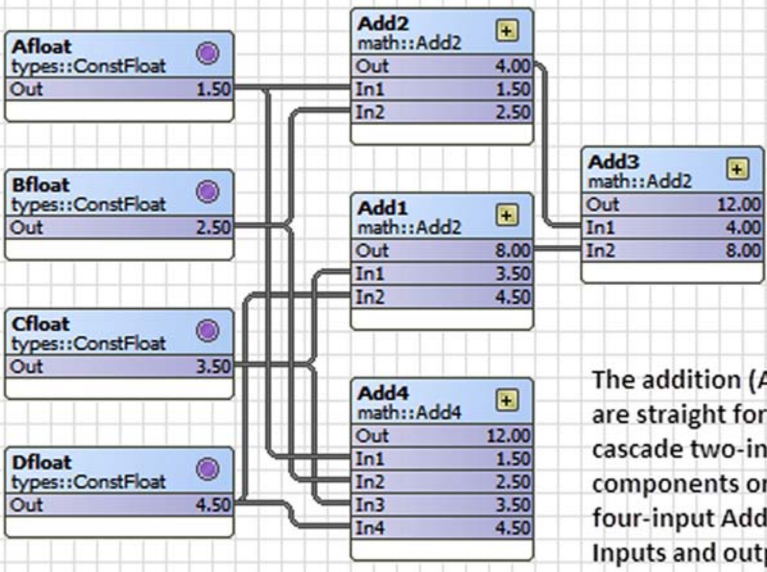


The de-multiplexer (DemuxI2) operates on integer values and provides a linear selection of the outputs based upon the value of the input. If the input is 0, the first output (Out1) is set to true. If the input is 2, the third output (Out3) is set to true.



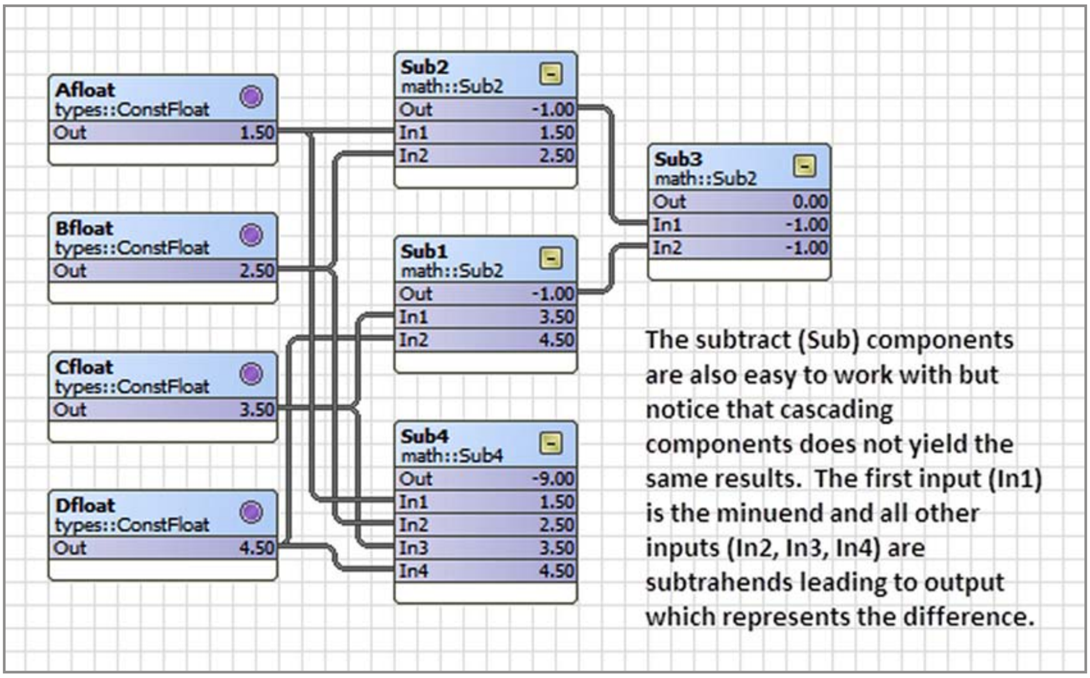
The analog de-multiplexer has only one input (In) and one selector (S1) but two outputs. When the selector is false, Out1 reflects the input. When the selector is true, Out 1 will reflect the input just before the selector changed state and Out2 will reflect the instantaneous value of the input. This component can be treated as a sample-and-hold detector.

Float Addition

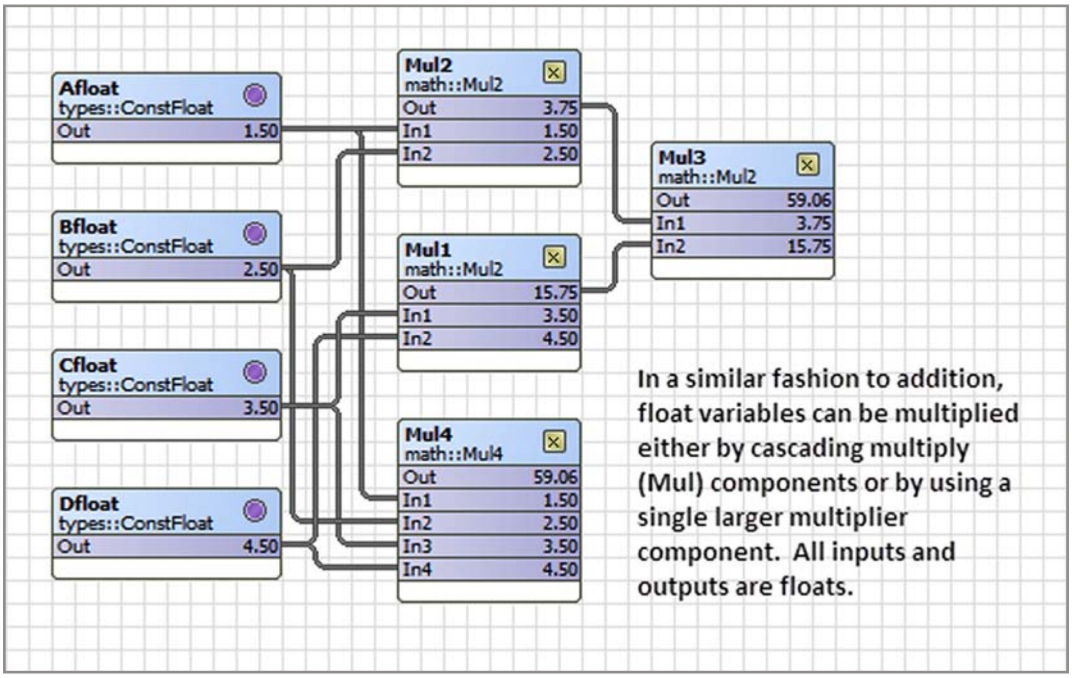


The addition (Add) components are straight forward. You can cascade two-input Add components or you can use a four-input Add component. Inputs and outputs are all floats.

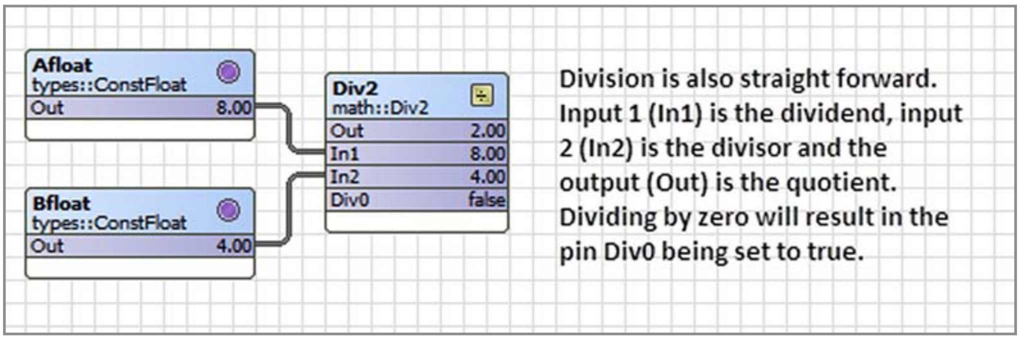
Float Subtraction



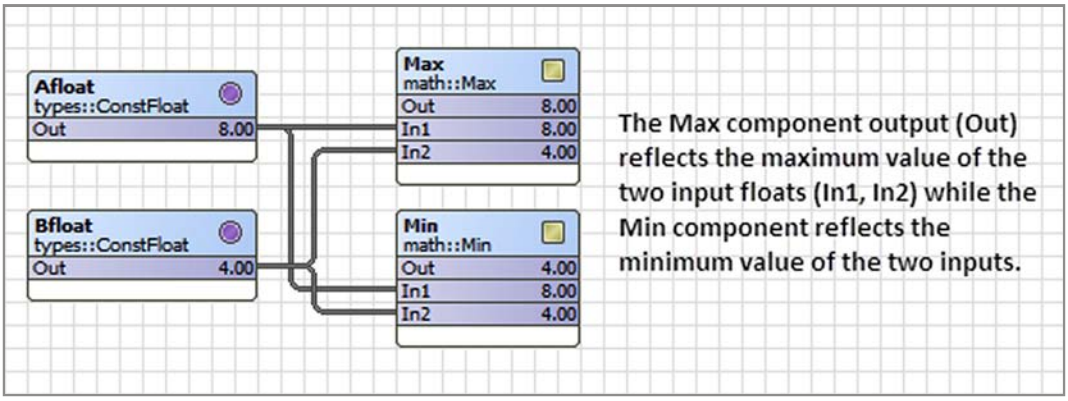
Float Multiplication

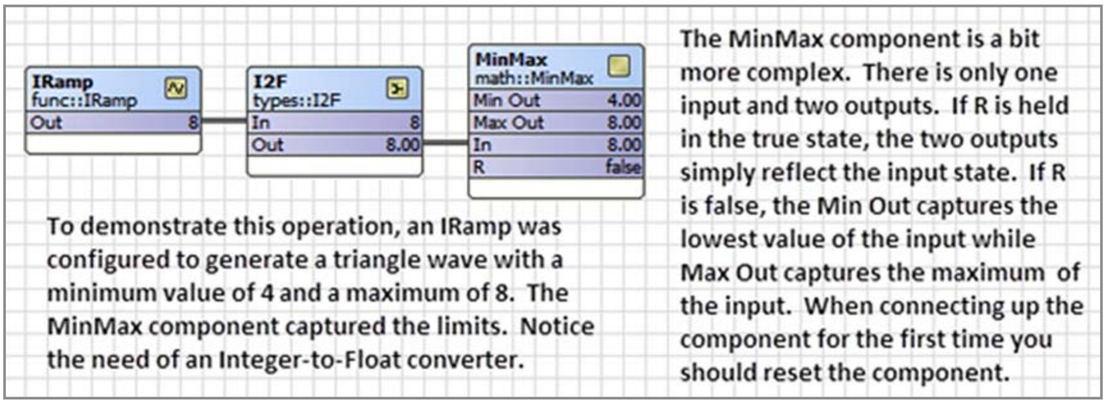


Float Division

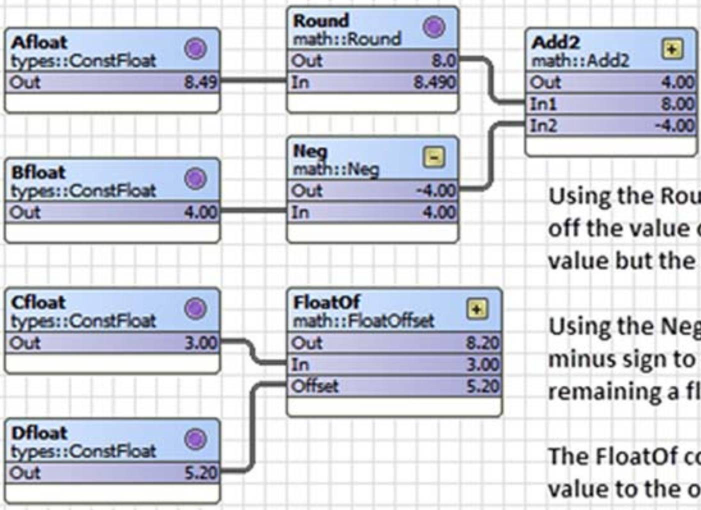


Finding Minimums and Maximums





Rounding Off Floats

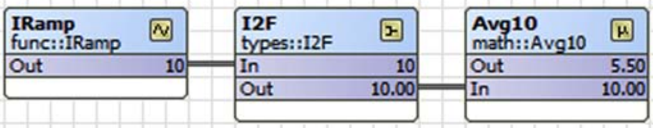


Using the Round component you can round-off the value of a float to the closest integer value but the output will remain a float.

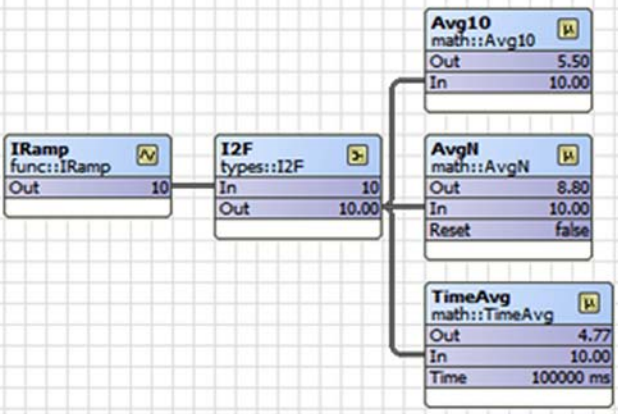
Using the Neg component you can append a minus sign to a float with the output remaining a float.

The FloatOf component appends an offset value to the output. It is not necessary to use a constant component for establishing the offset amount. The offset amount can be configured within the FloatOf component.

Averaging Successive Readings

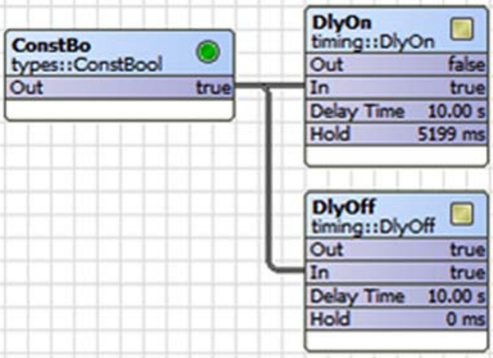


The Avg10 component averages the last ten input values and sends the result to the output. To demonstrate this, an IRamp is configured to create a triangle wave with a minimum value of 0 and a maximum value of 10. Increments are set to 1. When the IRamp reaches 10, the Avg10 component would have averaged 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 for a value of 5.5 which appears in the output.



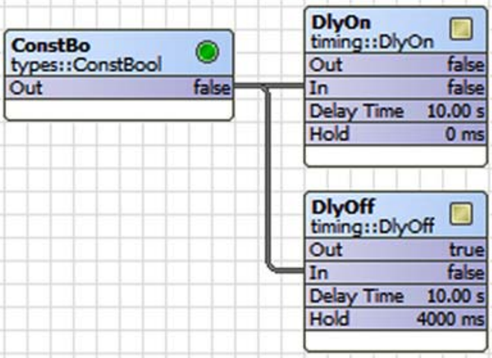
In this example three averaging components are compared. The Avg10 averages over 10 samples but the data must change to trigger a new sample. The AvgN component can be configured for the the number of samples but it samples every scan and not on just a change in value. The TimeAvg averages over a fixed period of time which is configurable. The output does not change until all samples are obtained.

On-Delays and Off-Delays



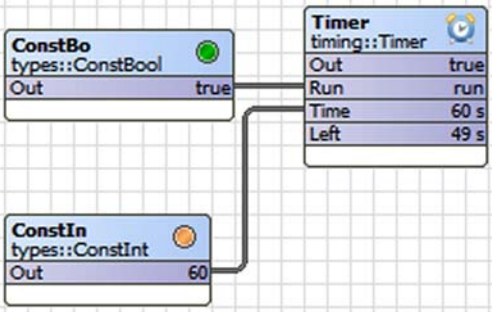
The DlyOn component is an on-delay timer which begins timing on the false to true transition of the input. Once the time (as shown in the Hold slot) goes to 0 the output will become true. The delay time is configurable. In this example the delay timer is still timing after the input went true 4.8 seconds ago.

The DlyOff component operates the same except it is triggered by a true to false transition of the input.



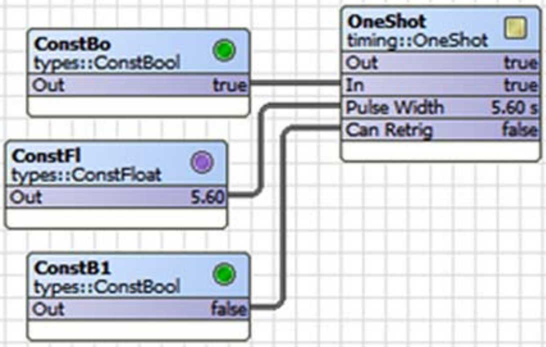
The input to the two timers made a true to false transition 6 seconds ago. The DlyOn component had immediately transitioned from true to false with the input but the DlyOff timer is still timing. In another 4 seconds its output will become false.

Using the Timer

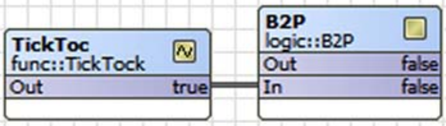


The Timer component will count down from a predetermined amount when the Run input is true. A constant integer component was used to set the time although the Timer component can be internally configured. The output will remain true during timing and transition false upon completion or if the Run input goes false. To begin a new timing period, the Run input must be cycled.

Using One-Shots — Mono-Stable Multivibrators

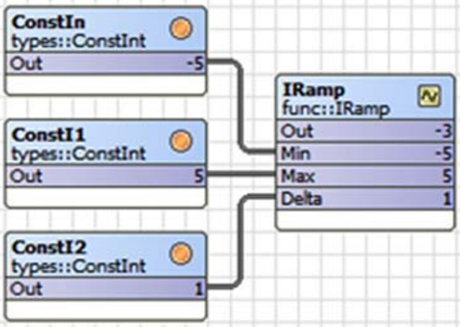


The OneShot provides a single pulse of determined value upon the false to true transition of the input signal. The output immediately goes true on its input's false to true transition and then returns to false when timing is complete. The Pulse Width can be configured or externally programmed as shown. A re-triggerable one-shot will renew timing if the input transitions from true to false to true during the timing period. A non-retriggerable will not. Notice that the Pulse Width is a float.

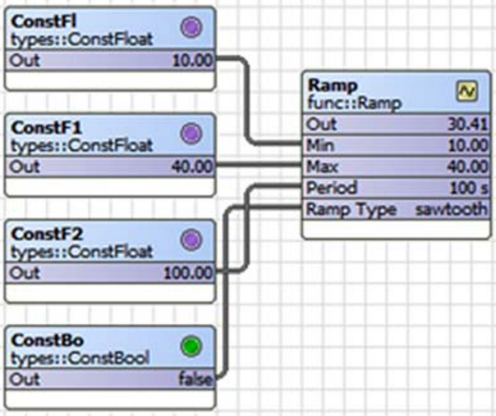


The Boolean-to-Pulse (B2P) converter is actually a very simple single-shot in that it outputs a true for only one scan time when its input goes from false to true. There are no time settings. It is used when a pulse is required after detection of an event instead of a logic level.

Creating Ramps — A-Stable Multivibrators

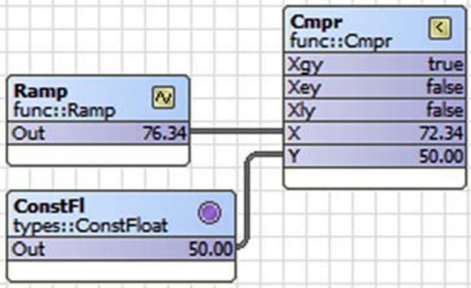


The IRamp provides a triangular output ranging from Min to Max with increments of Delta. These parameters can be configured or programmed as shown. The time increment must be configured having the units of seconds. Notice that all the configurable settings are integers as is the output.



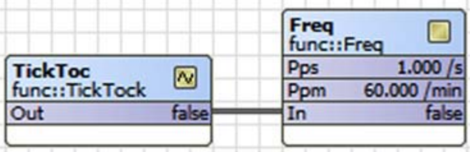
The Ramp is similar to the IRamp but the Ramp has mostly float settings and a float output. The output of the Ramp can be configured or programmed for either a sawtooth or triangle wave. Increasing the period slows down the speed of the ramp within the limits of Min and Max. Notice that although the Period is a float, the input to Period will be rounded to the nearest integer.

Comparing Two Floats



The comparator component (Cmpr) compares the the X input to that of the Y input. If X is less than Y, then the Xly output is true. If X equals Y then Xey is true. If X is greater than Y then Xgy is true. Both inputs are floats and the outputs are Booleans. In this example the output of the Ramp is compared to that of a constant. Using the default values of the Ramp, the input X varies as a triangle between 0 and 100 every 10 seconds. You can watch how the comparator outputs change over this range.

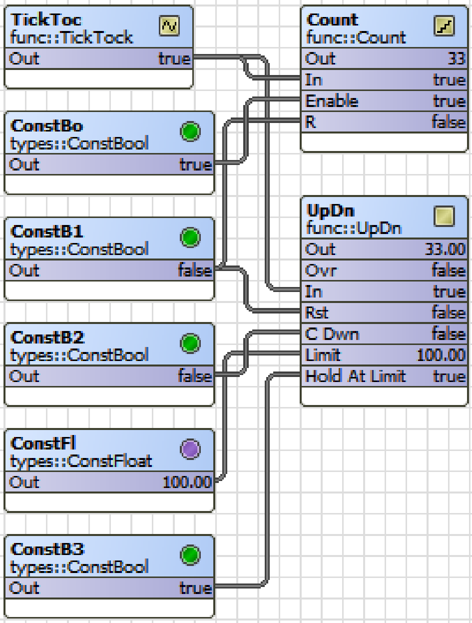
A Simple Clock — the TickToc



The TickToc component provides a convenient clock from 1 to 10 pulses per second. However, because of the controller scan time and other processing overhead it is recommended to use its default value of one second. More accurate timing is available from a real-time clock.

The Freq component can provide output values in pulses-per-second (Pps) or pulses-per-minute (Ppm). Because of the low-speed nature of these two components, the Ppm calculation will probably be the most useful.

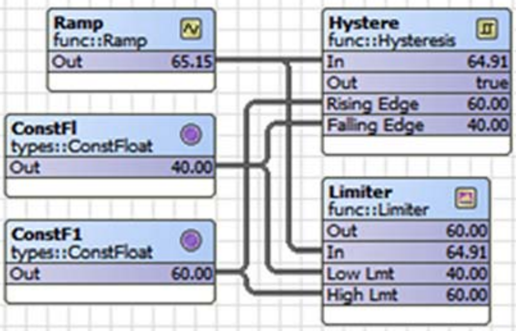
Introducing Counters



There are two counters. Count is an up/down counter with an integer output. It must be enabled to count. It can be reset to zero or preset to a postive value as long as the Enable pin is true. The direction pin (Dir) can be connected for programmable up/down counting.

UpDn is an up/down counter with a programmable direction input (C Dwn) which can also be configured. Although counters are inherently integer devices, the output of this component is a float. In this example a limit of 100 has been programmed. Once the limit is hit the overflow bit (Ovr) will be set. If Hold At Limit is true, the counter will not go pass 100. If it is false, the counter will continue to count past the limit but the overflow bit will remain set. Resetting the counter returns the component to the start position while clearing the counter and overflow bit.

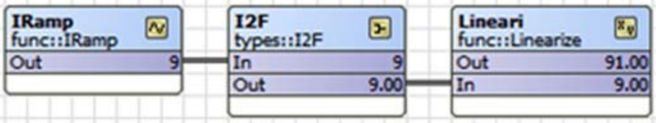
Operating on Real-World Signals — Hysteresis and Limiting



The hysteresis component (Hystere) has separate rising-edge and falling-edge trip points when setting a trigger on a float variable. It is ideal for creating a digital event from a real-world analog input. Its output is Boolean.

The Limiter component restricts the range of a float variable by outputting a float that does not exceed the configurable low-limit (Low Lmt) or high-limit (High Lmt). The Limiter only limits the range of its output and does not scale the input float.

Handling Non-Linear Signals



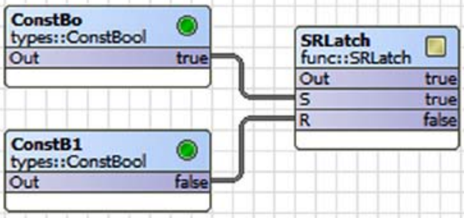
The linearize component (Lineari) operates on a float input and creates a piece-wise linear representation of a non-linear input (such as a thermistor) or it can create a non-linear piece-wise representation of a linear input. There is complete flexibility in defining the ten X,Y coordinates along the output curve. The component determines the approximate output between the ten coordinates using linear interpolation.

Meta	Group [1] »
Out	56.50
In	7.50
X0	0.00
Y0	0.00
X1	1.00
Y1	1.00
X2	2.00
Y2	4.00
X3	3.00
Y3	9.00
X4	4.00
Y4	16.00
X5	5.00
Y5	25.00
X6	6.00
Y6	36.00
X7	7.00
Y7	49.00
X8	8.00
Y8	64.00
X9	9.00
Y9	81.00

In this example we will do the reverse of what is commonly done. We will use a linear input and create a non-linear output that approximates the equation Y=X*X over the range of X values from 0 to 9. We need to input corresponding values of Y that obey the desired equation. To make it easy we will use integer values but this is not a restriction. For example, the square of 4 is 16 and the square of 5 is 25. We enter the X values as an independent variable and then the Y value as the dependent variable. We need to be careful that the input does not exceed 9 in this example because we do not define a corresponding value for Y above 9.

You can test the interpolation by entering a value for X in the In slot assuming no link is connected to the linearize component. This is done here. Notice that the result is 56.50 for an input value of 7.5. The correct value would have been 56.25 which is very close.

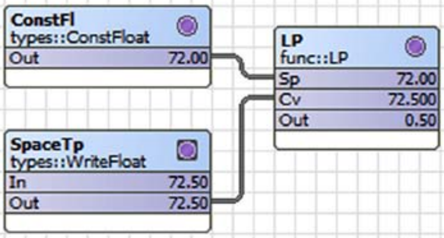
Simple Set-Reset Flip Flop — Bi-Stable Multivibrator



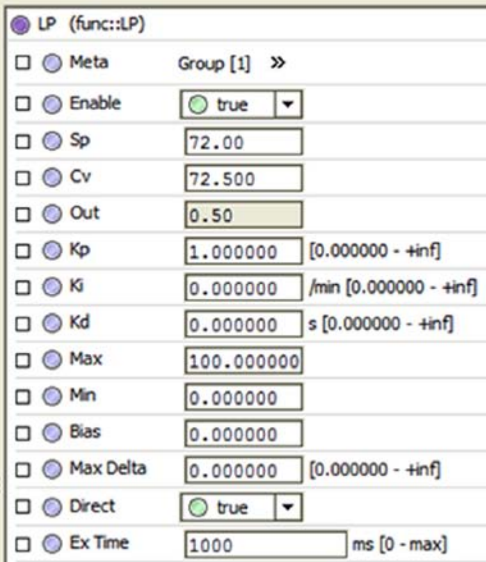
The SRLatch appears to be straight-forward logic block. The output would become true if the set (S) pin is high and would go low if the reset (R) pin goes high. However, both the S and R pins are positive leading-edge sensitive. Regardless of their steady-state condition, the output (Out) will only change on the false-to-true transition of either input. If this occurs on the S pin the output goes high and will remain high until the R pin does its transition.

On the rare condition that both S and R transition from false-to-true during the same logic scan, R will take precedence because its state is tested last in the logic and therefor the output will be false.

The Loop Component — Basic PID Controller



The LP or loop component is one of the most complex components. It can provide three modes of control P-proportional, I-integral, and D-derivative. In this example we will assume a temperature loop with a setpoint (Sp) of 72 degrees and a controlled variable (Cv) currently at 72.5 degrees which is the space temperature which we want to control.



Enable must be configured true otherwise there is no control.

Kp is the proportional gain which defaults to 1. Notice that the error signal is Cv-Sp or 0.5. The error multiplied by the proportional gain of 1 yields an output of 0.50. If the Ki and Kd factors are used, their contributions are also multiplied by the proportional gain factor. Ki is the integral gain in units of resets per minute. It is multiplied by the error signal. Kd is the derivative gain in seconds and it is also multiplied by the error signal.

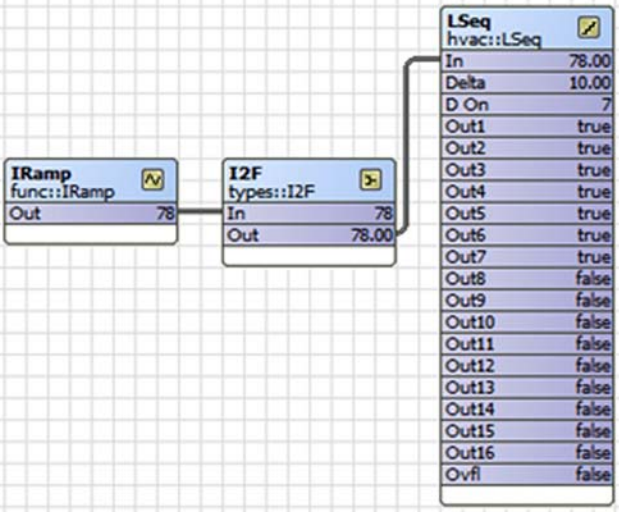
Min and Max are the limits of the output signal. They can be set to any value. Bias can offset the output regardless of the error. Max Delta sets the rate of change of the output within the output limits. This will slow the output swing.

Bias only applies to proportional-only (P) control. When using a PI controller, reset-windup can be minimized by limiting the output range.

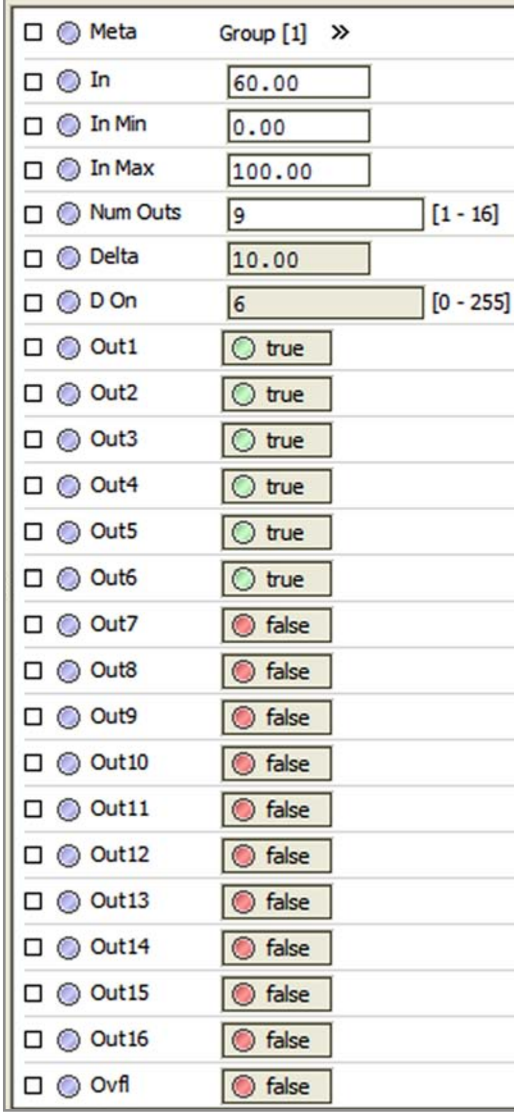
For a cooling application set Direct to true. For heating set it for false.

The loop equation is solved each execute time (Ex Time) in milliseconds.

Linear Sequencer — Bar-Graph Representation of a Float



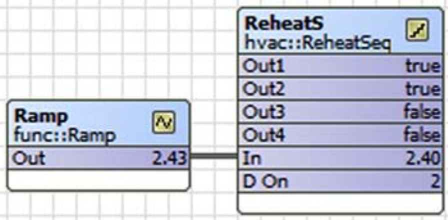
The linear sequencer (LSeq) provides a digital representation of an input float similar in operation to a bar graph on audio equipment. It is easier to understand its operation using an integer input. There are 16 possible Boolean outputs plus one overflow (Ovfl) flag. The input ramp provides a triangle wave from 0 to 100. The sequencer was configured for a 0 minimum input and a 100 maximum input. The maximum number of outputs was configured for 9 yielding a Delta of 10.



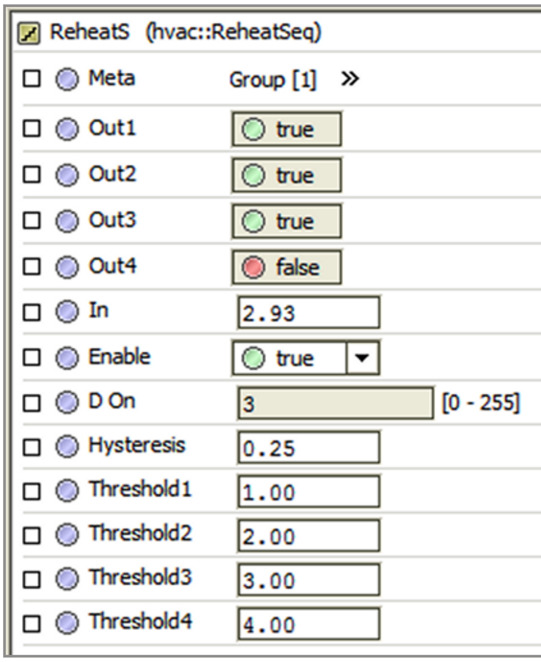
The range of the linear sequencer is configured using In Min at the low-end and In Max at the high-end. Selecting the number of outputs (Num Outs) determines the difference (Delta) between successive outputs turning on. In this case the range is 100 and the number of desired outputs is 9. Divide 100 by Num Outs + 1 and you will get a Delta of 10.

You will notice that the input (In) is at 60 and D On is indicating that six outputs are on. With an input between 0-9, there are no outputs on but once you hit a decade such as 10, 20 on up to 90, successive outputs will come on. At the maximum of 100, 9 lights will be on. If the input exceeds the maximum intended, the overflow flag will set but the number of outputs will remain as specified by Num Outs.

Reheat Sequencer — Four Staged Outputs from a Float Input



The reheat sequencer (ReheatS) provides a linear sequence of up to four outputs based upon the input float (In). The threshold for the four outputs can be configured for increasing values of the input. As the input increases to each threshold, the corresponding output will go on. As the input decreases below the threshold, the corresponding output will remain on until the Hysteresis value is exceeded.

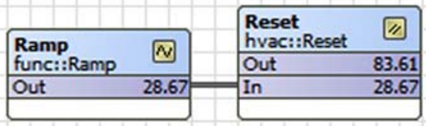


Enable must to true otherwise the outputs to be false.

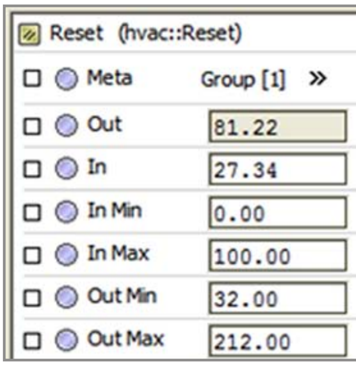
There are four possible threshold settings corresponding to four outputs. As the input signal increases to each threshold its corresponding output goes on and stays on until the input drops below the threshold plus the value of the hysteresis.

The input signal is decreasing but it has not exceeded the amount of the threshold so output 3 (Out3) remains set. Once the signal is below 2.75, output 3 will go off.

Reset — Scaling a Float Input between Two Limits

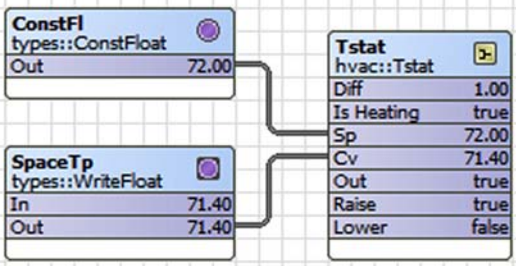


The reset component (Reset) will scale the output linearly between two limits. The input range must be configured by setting In Min and In Max. The corresponding output for those two points must be configured as Out Min and Out Max. If the input signal exceeds the defined input range, the output will be clamped to one of the two output limits.



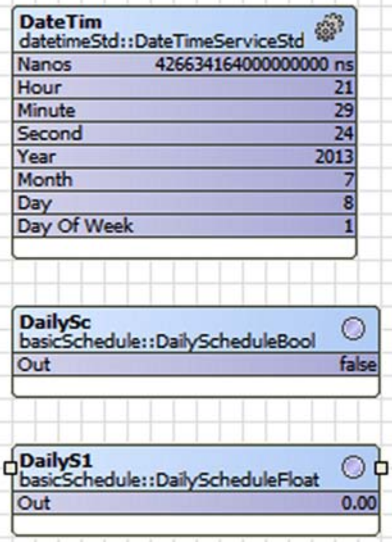
In this example we are converting degrees Celsius to degrees Fahrenheit within the 0-100 degree Celsius range. Therefore we set Out Min and Out Max to the corresponding Fahrenheit values. All Celsius input values between these two limits will be interpolated thereby providing the correct Fahrenheit values.

Tstat — Basic On/Off Temperature Controller



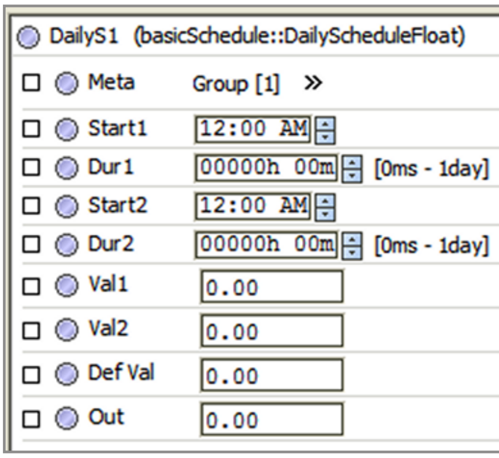
The Tstat is an on/off temperature controller for either heating or cooling. For heating configure the Is Heating bit to true. The deadband can be set by the Diff value. If the controlled variable (Cv) deviates from the setpoint (Sp) by half the Diff value, the output (Out) will become true and stay set until Cv deviates from the setpoint by a like amount in the other direction. In this way Diff also provides hysteresis. The Raise and Lower outputs are a function of the Is Heating setting. If Is Heating is true, Out=Lower, otherwise Out= Raise.

Real-Time Clock and Scheduling



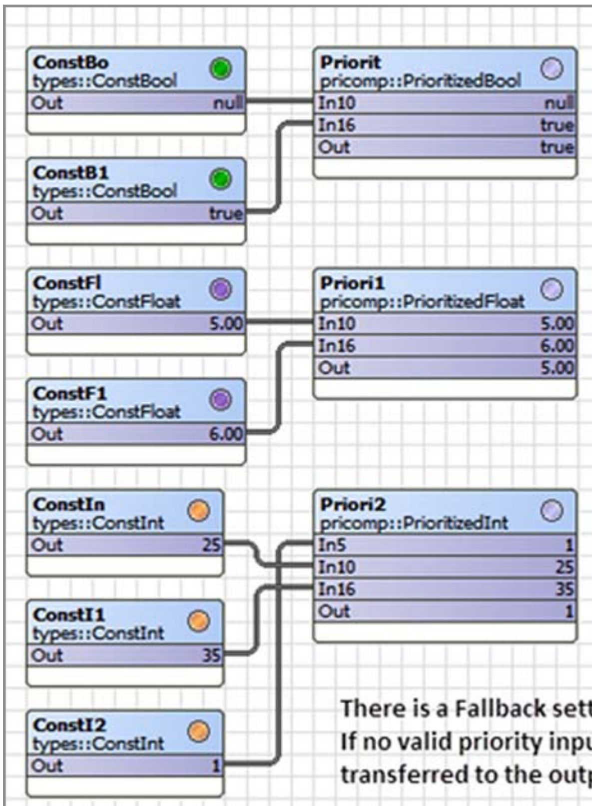
The DateTim component provides real-time information. There is no need to place it on the wiresheet. However, if you need specific information from the component for driving logic, you can connect to the various integer outputs such as Hour, Minute and Second.

There are two schedule components which have different output types. One is for Boolean and the other for float. There is no need to connect the DateTim component to either of the schedulers. Each scheduler can handle two events over the 24 hour period by configuring the time and duration of each event. The output of each schedule will change with each event. If more events or more outputs are needed, multiple schedulers can be placed on the wiresheet.



Configuration of the two scheduler components is similar. For the float version, Val1 and Val2 need to be specified along with the start times (Start1 and Start2) and the durations (Dur1 and Dur2). The output (Out) will assert either Val1 or Val2 during the scheduled times. If neither are programmed, the Def Val should be configured.

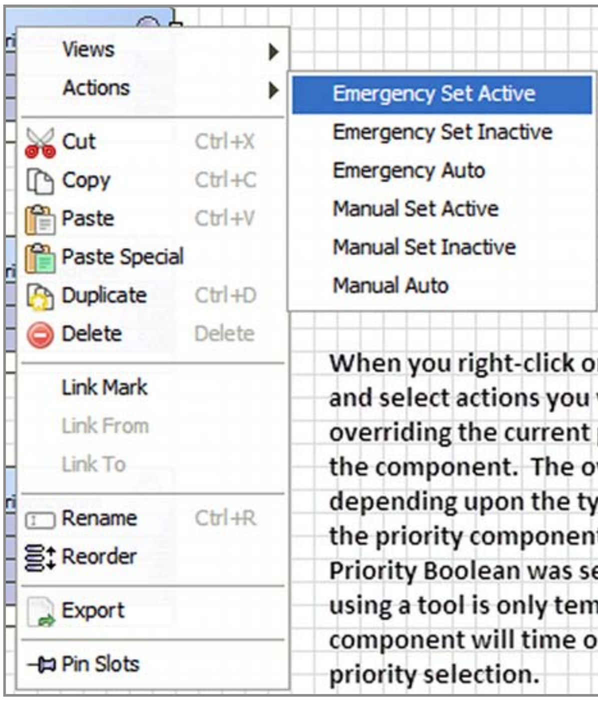
Priority Arrays



Priority array (Priorit) components exist for Boolean, float and integer variables. Up to 16 levels of priority from In1 to In16 can be assigned. In1 has the highest priority and In16 the lowest. With few exceptions, all can be pinned out. If a priority level is not assigned it is marked as a Null and therefor ignored. If a Null is inputted to the priority array as shown in the top-most example, the priority array will ignore it and choose the next in line input. The Boolean version of the priority array has two timer settings - one for minimum active time and minimum inactive time. If the highest priority device changes from false to true and then back to false, the priority component will maintain the event for the configured times.

There is a Fallback setting in each array that can be specified. If no valid priority input exists, the Fallback value is transferred to the output.

The Override Exp Time guards against the possibility of an indefinite override condition.



When you right-click on the priority component and select actions you will have several choices for overriding the current priority selection made by the component. The override choices vary depending upon the type of variable supported by the priority component. In this example, the Priority Boolean was selected. Setting an override using a tool is only temporary. Eventually, the component will time out and revert to normal priority selection.

United States
Contemporary Control Systems, Inc.
2431 Curtiss Street
Downers Grove, IL 60515
USA

Tel: +1 630 963 7070
Fax:+1 630 963 0109

info@ccontrols.com
www.ccontrols.com

China
Contemporary Controls (Suzhou) Co. Ltd
11 Huoju Road
Science & Technology Industrial Park
New District, Suzhou
PR China 215009

Tel: +86 512 68095866
Fax: +86 512 68093760

info@ccontrols.com.cn
www.ccontrols.asia

United Kingdom
Contemporary Controls Ltd
14 Bow Court
Fletchworth Gate
Coventry CV5 6SP
United Kingdom

Tel: +44 (0)24 7641 3786
Fax:+44 (0)24 7641 3923

ccl.info@ccontrols.com
www.ccontrols.eu

Germany
Contemporary Controls GmbH
Fuggerstraße 1 B
04158 Leipzig
Germany

Tel: +49 341 520359 0
Fax: +49 341 520359 16

ccg.info@ccontrols.com
www.ccontrols.eu