

Information SHEET



Component Descriptions for the Contemporary Controls' HVAC Kit

What follows are component descriptions for Sedona components in the CControls_HVAC kit. This is a custom hardware independent kit that can be used with any Sedona 1.2.28 platform.

Analog High/Low (AnaHiLo)

This component has two independent usages. It can be used to initiate an alarm or event for an analog point, or it could be used to limit the output range of an analog point. Both capabilities can be used at the same time.

If *HoldAtLimit* is false, the output (Out) will follow the input (In). If *HoldAtLimit* is true, Out will be clamped to the value of *HiLimit* if the input exceeds this high limit or clamped to the value of *LoLimit* if the input becomes less than the low limit. Both *HiLimit* and *LoLimit* are configurable float values while *HoldAtLimit* is a configurable Boolean value.

This same component can be used to create an alarm or an event independent of the setting of *HoldAtLimit*. If *LimitOutEnable* is true, then *OverLimit* will become true if In remains above the high limit for a time greater or equal than the value set in configurable float slot *LimitDelay*. Once *OverLimit* is true, it will remain true until In decreases below the high limit by an amount greater or equal than the value set in configurable float slot *Differential*. A similar action will occur if In remains below the low limit for a time greater or equal than the value set in *LimitDelay*. Once *UnderLimit* is true, it will remain true until In increases above the low limit by an amount equal or greater than the value set in *Differential*.

If the configurable slot *LimitOutEnable* is false, then both *OverLimit* and *UnderLimit* are false.

AnaHiLo	
CControls HVAC::AnaHiLo	
LimitDelay	1
HiLimit	10
LoLimit	-10
Differential	0.1
HoldAtLimit	false
LimitOutEnable	false
In	0
Out	0
OverLimit	false
UnderLimit	false

Anti-Short Cycle (AntiSCY)

This component is used to protect mechanical equipment from running for too short of a time or restarting without a sufficient delay. Usually used with compressors, it can be used to protect any HVAC piece of equipment.

There are two configurable time slots – *MinRunTime* and *MinOffTime*. Both are set in units of seconds. Assume Reset is false. Once the command signal to the input slot (In) transitions from a false to true state, the output (Out) will immediately become true assuming that the previous command signal did not go false and then true within the time set in *MinOffTime*. *Out* will remain true for the entire time that the command signal is asserted and will continue to remain in the true state after the command signal returns to false for the amount of time set in *MinRunTime*. Once *Out* transitions to the false state, it will remain in the false state for an amount of time that is set in the *MinOffTime* slot regardless of the state of In. Once this timer times out, the *In* slot will be re-enabled.

If Reset is true, *Out* becomes false regardless of the state of In, or the state of the minimum run timer. All timers are cleared. Normal component operation occurs once Reset returns to the false state.

AntiSCY	
CControls HVAC::AntiSCY	
MinRunTime	1
MinOffTime	1
In	false
Out	false
Reset	false

BTU/Hour Calculator (BTUh)

This component makes it easy to calculate the amount of heating or cooling of water that is occurring by monitoring the temperature differential across a piece of equipment and the flow rate. Although the result is BTU per hour, the industry would just say BTU.

This component executes the equation:

$$\text{BTU/hr} = 500 * \text{flow rate} * (\text{Outlet temperature} - \text{Inlet temperature})$$

$$\text{TonOutR} = \text{GPM} / 12,000$$

$$\text{TonOutC} = \text{GPM} / 15,000$$

Where;

Flow rate is in gallons per minute

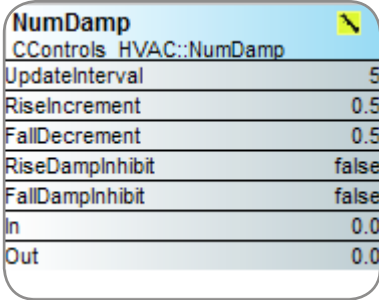
Temperatures are in °F

The output slot (*Out*) follows the equation above with inputs of flow rate (*InGPM*), Outlet temperature (*OutTemp*) and Inlet temperature (*InTemp*). In order to ensure a positive result, the temperature differential is treated as an absolute number. *OffCal* is a configurable float slot that will add positive or negative bias to the calculation if the equation needs to be tweaked. *ExeDelay* is a configurable float that specifies the time between calculations. *TonOutR* reflects the refrigeration tons conversion from BTU/Hr. while *TonOutC* reflects cooling tower tons conversion. *InGPM* is a configurable float that can be pre-set for constant flow calculations.

BTUh	
CControls HVAC::BTUh	
ExeDelay	0.0
OffCal	0.0
InGPM	0.0
InTemp	0.0
OutTemp	0.0
Out	0.0
TonOutR	0.0
TonOutC	0.0

Numeric Dampener (NumDamp)

This component functions as a digital filter to dampen the volatility of an input signal by creating an output version of the input signal but with modifications to the signal's rate-of-change, and amplitude change. Although the output (Out) will attempt to track the input (In), the rate-of-change and amplitude of the output will lag the input depending upon configurable settings.



Property	Value
UpdateInterval	5
RiseIncrement	0.5
FallDecrement	0.5
RiseDampInhibit	false
FallDampInhibit	false
In	0.0
Out	0.0

UpdateInterval is a configurable float in units of seconds. It sets the update rate of the output. The output will not change until the update interval is satisfied. If sampling is to be instantaneous, set *UpdateInterval* to zero.

RiseIncrement and *FallDecrement* are also configurable floats that set the incremental rising rate and falling rate of the output respectively. The output will not change more than the increments entered assuming that rise and fall rate inhibits are false.

There are two configurable Boolean inputs *RiseDampInhibit* and *FallDampInhibit*. In the false state, incremental changes to the output are allowed for rising and falling outputs. Depending upon which of these two inhibits are set to true, the rising (or falling) rates will not be modified by the component. These two inhibits do not impact the sampling rate.

If *UpdateInterval* is zero and both *RiseDampInhibit* and *FallDampInhibit* are true, Out will equal In.

Enhanced PID Loop Controller (EnhPID)

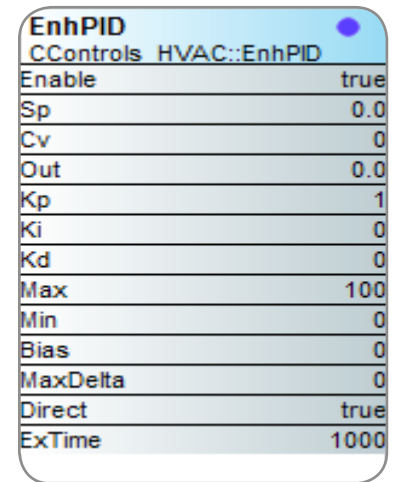
This component appears as an exact replica of the LP component in Tridium's Func kit but it has one significant change in the way it handles the configurable Boolean slot *Enable*.

When *Enable* is false in the LP slot, the LP component ceases to function and holds its output (*Out*) at its last state. When *Enable* returns to true, the LP picks up where it left off. This may not be acceptable in all applications.

With the Enhanced PID Loop Controller, if *Enable* becomes false, proportional control ceases and the internal timers and accumulators for integral and derivative action are cleared. *Out* will return to the *Bias* setting at a rate based upon *MaxDelta*. When *Enable* returns to true, the component returns to normal operation as if it began from a power-up condition.

If the intended application does not programmatically utilize the *Enable* slot, there is no need to use the Enhanced PID Loop Controller over the LP component. However, if this is not the case the action of the LP component under a disable state should be studied for acceptability.

To learn more about Enhanced PID Loop Controller, study the description for the LP component which is included in Tridium's 1.2 Func kit.



EnhPID	
CControls HVAC::EnhPID	
Enable	true
Sp	0.0
Cv	0
Out	0.0
Kp	1
Ki	0
Kd	0
Max	100
Min	0
Bias	0
MaxDelta	0
Direct	true
ExTime	1000

Lead Lag Sequence Controller (LeadLag)

This component monitors and controls up to four devices (usually pumps) dedicated to one critical process. If a pump fails to come on-line or goes off-line when it should be on-line, the next-in-line pump will take over. In order to even out running times on all pumps, the next-in-line pump is selected on a time basis but before the primary pump (lead pump) is taken off-line, the next-in-line pump (lag pump) must be proven to be on-line before the changeover. Configuration can be for 2, 3 or 4 pumps.

RunTime is a configurable float that sets the desired run time for all pumps. Its time is in minutes. *ProofDelay* is a configurable float that sets the time limit before declaring a commanded pump has failed to come on line. *OverlapTime* is a configurable float that sets the time where two pumps will run together during the proving time and beyond. The last two slots are in seconds. *OutQty* is a configurable multi-state that can be two, three or four. It sets the pump sequence to A-B, A-B-C or A-B-C-D before repeating. Each pump in the sequence must have a proving signal such as an auxiliary switch on a motor contactor, current switch or a flow switch. These are applied to *ProofA* through *ProofD*. Only those proving signals used in the sequence need to be connected.

A single command signal set to true is applied to the input (In) and starts the sequence. If this signal goes false all pumps stop. Assuming it is pump A's turn to come on, its proving signal is monitored to verify that it is on-line and will continuously be monitored throughout its run time cycle. At the end of run time, the lag pump (pump B) is commanded on and its proving signal is monitored for activation. If proving occurs, the lead pump will shut down after the overlap time and the lag pump now becomes the lead pump (pump B). If the lag pump fails to come on, the Alarm slot will become true. The next-in-line pump (pump C – in a three or four pump sequence) is commanded on and its proving signal is monitored for activation. If it is proven, the Alarm slot will return to false. If a lead pump failure is sensed during run time, the Alarm slot becomes true, and the lag pump will be commanded on and proving initiated. Upon successful proving the Alarm slot will be cleared. The sequence continues to repeat based on the run time setting in the component.

LeadLag	
CControls HVAC::LeadLag	
RunTime	10
ProofDelay	1
OverlapTime	0
OutQty	Two
In	false
OutA	false
OutB	false
OutC	false
OutD	false
ProofA	false
ProofB	false
ProofC	false
ProofD	false
Alarm	false

Outside Air True Blend (OATrueB)

This component makes a dynamic calculation of the percentage opening of an outside air damper which is helpful when creating economizers or for demand control ventilation to reduce CO2. In order to use this component it will be necessary to monitor outside-air temperature (OAT), mixed-air temperature (MAT) and return-air temperature (RAT).

The component solves the following equation for Outside Air %:

$$\text{OAT\%} = 100 * (\text{MAT} - \text{RAT}) / (\text{OAT} - \text{RAT})$$

Where; MAT, RAT, and OAT are in either °C or °F.

The output (*Out*) is OAT%. Execution delay (*ExeDelay*) is the delay between calculations in seconds. *OffCal* is a bias that can be applied to the output calculation to tweak the result.

OutsideAT, *ReturnAT* and *MixedAT* are OAT, RAT and MAT respectively.

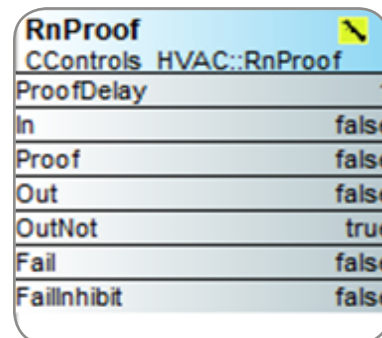
The value of MAT should always be between OAT and RAT and therefore the Output value should be between 0 and 100%. If the calculated output is below zero, the Output value will be clamped at zero and the Fault slot will be set to true. Likewise, if the calculated output exceeds 100 then the Output will be clamped to 100 and the Fault slot will be set to true. If OAT=RAT then the Output will be zero and the Fault slot will be set to true.

OATrueB	
CControls HVAC::OATrueB	
ExeDelay	1
OffCal	0.0
OutsideAT	0.0
ReturnAT	0.0
MixedAT	0.0
Output	0.0
Fault	true

Run Proving (RnProof)

This component is a simple adaptation of the more complex *LeadLag* component. With this component, one device (pump or motor) is commanded on and proven to be on otherwise a failure is noted.

When the command signal applied to the input (*In*) becomes true, the output (*Out*) is driven true and the *OutNot* is driven false. *ProofDelay* is a configurable float that sets the time limit before declaring the commanded pump has failed to come on-line. The proving signal can come from an auxiliary switch on a motor contactor, current switch or a flow switch and is applied to the Proof input. If the pump is determined to have failed to come on-line, the slot Fail becomes true as long as *FailInhibit* is false and will stay on until the command signal returns to the false state. If *FailInhibit* is true, the alarm slot Fail is disabled.

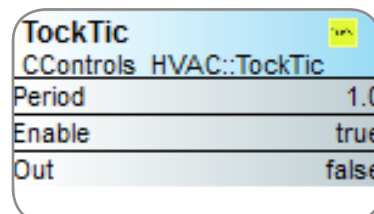


RnProof	
CControls HVAC::RnProof	
ProofDelay	1
In	false
Proof	false
Out	false
OutNot	true
Fail	false
FailInhibit	false

Period Driven Clock (TockTic)

TockTic should not be confused with the TickToc component in Tridium's Func kit. With the Tridium component, the clock frequency is set in steps per second between 1 and 10. With TockTic, the frequency is set by the period allowing for much slower frequencies thereby providing a much slower counting source.

Enable defaults to true and must be true to enable the output (*Out*). *Period* is a configurable float slot with a minimum value of 0.5 seconds. When enabled, *Out* will provide a square wave with a frequency of one divided by the specified period.



TockTic	
CControls HVAC::TockTic	
Period	1.0
Enable	true
Out	false

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