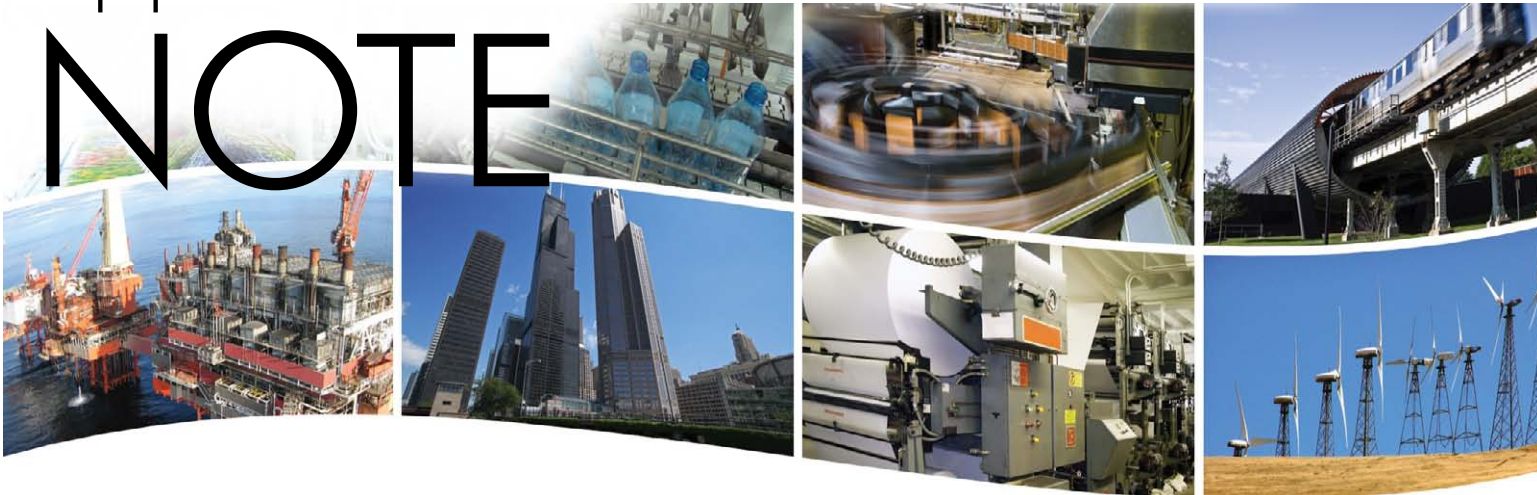


# application NOTE



## **Sedona Component Descriptions**

### **Introduction**

Developed by Tridium Inc., Sedona Framework™ is a software environment designed to make it easy to build smart, networked, embedded devices which are well suited for implementing control applications. The Sedona language is a component-oriented programming language similar to Java or C#. For those familiar with Niagara Framework®, understanding Sedona Framework will be easy. The system integrator's role is to create an application by assembling components on a wire sheet and connecting and configuring these components using a graphical programming tool such as Sedona Workbench. Applications can be developed live on a target device such as the BAS Remote, or offline, and then saved and uploaded via an IP connection. The Sedona Virtual Machine (SVM) resident in the device executes the application.

Components are deployed in kits. Kits are available from Tridium and Contemporary Controls. As more components are developed, revised kits will be made available. What follows are descriptions of components that will be of the most use to system integrators when developing control applications. These components are organized by the kits in which they can be found.

### **Control Kit Components**


The following components are found in the Control Kit. Boolean variables are assumed if there is a false/true state. Integers (32-bit signed integers) are shown as whole numbers while floats (32-bit floating point) are shown with a decimal point. Many of the following components may have been expanded in order to show internally configurable parameters. The default view of a component may not show the same level of detail. In order to show more detail, select the desired component on the wire sheet and then right click. Select Pin Slots and in the Pin Slot menu, select all the parameters you want to view. An expanded view will then appear when exiting the Pin Slot menu.

<b>Add2</b>	
control::Add2	
Out	0.0
In1	0.0
In2	0.0

**Two-input addition — results in the addition of two floats**


$$Out = In1 + In2$$

## Application Note — BAS Remote and Sedona Component Descriptions

<b>Add4</b>	
control::Add4	
Out	0.0
In1	0.0
In2	0.0
In3	0.0
In4	0.0


**Four-input addition — results in the addition of four floats**

$$Out = In1 + In2 + In3 + In4$$

<b>And2</b>	
control::And2	
Out	false
In1	false
In2	false


**Two-input Boolean product — two-input AND gate**

$$Out = In1 \cdot In2$$

<b>And4</b>	
control::And4	
Out	false
In1	false
In2	false
In3	false
In4	false

**Four-input Boolean product — four-input AND gate**

$$Out = In1 \cdot In2 \cdot In3 \cdot In4$$

<b>ASW</b>	
control::ASW	
Out	0.0
In1	0.0
In2	0.0
S1	false

**Analog switch — selection between two float variables**

*If S1 is false then Out = In1*

*If S1 is true then Out = In2*

<b>ASW4</b>	
control::ASW4	
Out	0.0
In1	0.0
In2	0.0
In3	0.0
In4	0.0
Starts At	0
Sel	0

**Analog switch — selection between four floats**

Configurable integer parameter *Starts At* sets the base selection.


*If integer Sel <= Starts At then Out = In1*

*If integer Sel = Starts At + 1 then Out = In2*

*If integer Sel = Starts At + 2 then Out = In3*

*If integer Sel = Starts At + 3 then Out = In4*

*For all other values of Sel then Out = In4*


<b>Avg10</b>	
control::Avg10	
Out	nan
In	0.0
Max Time	0

**Average of 10 — sums the last ten floats while dividing by ten thereby providing a running average**

$$Out = (sum\ of\ the\ last\ ten\ values) / ten$$

The float input *In* is sampled once every scan and stored. If the input does not change in value on the next scan, it is not sampled again — unless sufficient time passes that exceeds the internal integer *Max Time* with units of milliseconds. In this instance the input is sampled and treated as another value.


## Application Note — BAS Remote and Sedona Component Descriptions

<b>B2F</b>	
control::B2F	
Out	0.0
Count	0.0
In1	false
In2	false
In3	false
In4	false
In5	false
In6	false
In7	false
In8	false
In9	false
In10	false
In11	false
In12	false
In13	false
In14	false
In15	false
In16	false

### Binary to float encoder — 16-bit binary to float conversion


*Out = encoded value of binary input with In16 being the MSB and In1 being the LSB*

*Count = sum of the number of active inputs*

<b>B2P</b>	
control::B2P	
Out	false
In	false

### Binary to pulse — simple mono-stable oscillator (single-shot)


*Out = true for one scan on the raising edge of In*

<b>BSW</b>	
control::BSW	
Out	false
In1	false
In2	false
S1	false

### Boolean Switch — selection between two Boolean variables

*If S1 is false then Out = In1*

*If S1 is true then Out = In2*

<b>Cmpr</b>	
control::Cmpr	
Xgy	false
Xey	true
Xly	false
X	0.0
Y	0.0

### Comparison math — comparison ( $\leq$ ) of two floats

*If  $X > Y$  then Xgy is true*

*If  $X = Y$  then Xey is true*

*If  $X < Y$  then Xly is true*

<b>ConstBo</b>	
control::ConstBoo	
Out	false

### Boolean Constant – a predefined Boolean value

*Out = a Boolean value that is internally configurable*

## Application Note — BAS Remote and Sedona Component Descriptions

<b>ConstFl</b>	
control::ConstFlo:	
Out	0.0


**Float Constant — a predefined float value**

*Out = a float value that is internally configurable*

<b>ConstIn</b>	
control::ConstInt	
Out	0

**Integer Constant — a predefined integer value**


*Out = an integer value that is internally configurable*

<b>Div2</b>	
control::Div2	
Out	0.0
In1	0.0
In2	0.0
Div0	true

**Divide two — results in the division of two floats**

*Out = In1/In2*

*Div0 = true if In2 is equal to zero*


<b>DlyOff</b>	
control::DlyOff	
Out	false
In	false
Delay Time	0.0
Hold	0

**Off delay timer — time delay from a true to false transition of the input**

*For input transitions from false to true, Out = true*

*For input transitions from true to false, Out = false after a delay*

*Hold is a read-only integer that counts down the time. Delay time is in seconds.*


<b>DlyOn</b>	
control::DlyOn	
Out	false
In	false
Delay Time	0.0
Hold	0

**On delay timer — time delay from a false to true transition of the input**

*For input transitions from true to false, Out = false*

*For input transitions from false to true, Out = true after a delay*

*Hold is a read-only integer that counts down the time. Delay Time is in seconds.*

<b>F2B</b>	
control::F2B	
In	0.0
Out1	false
Out2	false
Out3	false
Out4	false
Out5	false
Out6	false
Out7	false
Out8	false
Out9	false
Out10	false
Out11	false
Out12	false
Out13	false
Out14	false
Out15	false
Out16	false
Ovrf	false

**Float to binary decoder — float to 16-bit binary conversion**

*Out1 to Out16 = the 16-bit decoded value of In — with Out16 representing the MSB and Out1 representing the LSB*

*Ovrf = true when In > 65535*

*Although the input requires a float, fractional amounts are ignored during the conversion.*

## Application Note — BAS Remote and Sedona Component Descriptions

<b>F2I</b>	
control::F2I	
In	0.0
Out	0

### Float to integer — float to integer conversion

*Out = In except that the output will be a whole number*

The fractional amount of the float input will be truncated at the output.

<b>FloatOf</b>	
control::FloatOffs	
Out	0.0
In	0.0
Offset	0.0

### Float offset — float shifted by a fixed amount

*Out = In + Offset*

*Offset is a configurable float.*

<b>Freq</b>	
control::Freq	
Pps	0.0
Ppm	0.0
In	false

### Pulse frequency — calculates the input pulse frequency

*Pps = number of pulses per second of In*

*Ppm = number of pulses per minute of In*

<b>Hystere</b>	
control::Hysteresi	
In	0.0
Out	false
Rising Edge	50.0
Falling Edge	50.0

### Hysteresis — setting on/off trip points to an input variable

There are two internal floats called *Rising Edge* and *Falling Edge* which are configurable. If *Rising Edge* is greater than *Falling Edge*, then the following is true.

*If In > Rising Edge then Out = true and will remain in that state until In < Falling Edge*

*If Rising Edge is less than Falling Edge then the action is inverted.*

<b>I2F</b>	
control::I2F	
In	0
Out	0.0

### Integer to Float — integer to float conversion

*Out = In except that the output will become a float*

<b>ISW</b>	
control::ISW	
Out	0
In1	0
In2	0
S1	false

### Integer switch — selection between two integer variables

*If S1 is false then Out = In1*


*If S1 is true then Out = In2*

<b>L2F</b>	
control::L2F	
In	0
Out	0.0

### Long to Float — 64-bit signed integer to float conversion

*Out = In except that the output will become a float from a 64-bit signed integer*

## Application Note — BAS Remote and Sedona Component Descriptions

<b>Limiter</b>	
control::Limiter	
Out	0.0
In	0.0
Low Lmt	0.0
High Lmt	0.0


### Limiter — Restricts output within upper and lower bounds

*High Lmt* and *Low Lmt* are configurable floats.

*If In > High Lmt then Out = High Lmt*

*If In < Low Lmt then Out = Low Lmt*


*If In < High Lmt and > Low Lmt then Out = In*

<b>Linearize</b>	
control::Linearize	
Out	0.0
In	0.0
X0	0.0
Y0	0.0
X1	0.0
Y1	0.0
X2	0.0
Y2	0.0
X3	0.0
Y3	0.0
X4	0.0
Y4	0.0
X5	0.0
Y5	0.0
X6	0.0
Y6	0.0
X7	0.0
Y7	0.0
X8	0.0
Y8	0.0
X9	0.0
Y9	0.0

### Linearize — piecewise linearization of a float

For piecewise linearization of a non-linear input, there are ten pairs of x,y parameters that must be configured into this component. The x,y pairs indicate points along the input curve. For an x value of the input, there should be a corresponding y value of the output. For input values between these points, the component will estimate the output based upon the linear equation:

*Out = Y = mx + b where m is the slope between adjacent points and b is the Y intercept*

<b>LP</b>	
control::LP	
Enable	false
Sp	0.0
Cv	0.0000
Out	0.0
Kp	0.0000
Ki	0.0000
Kd	0.0000
Max	0.0000
Min	0.0000
Bias	0.0000
Direct	false
Ex Time	0 ms

### LP — proportional, integral, derivative (PID) loop controller

The LP component is much more complex component requiring an explanation of the numerous configurable parameters. *Sp* is the *setpoint* or the desired outcome. *Cv* is the *controlled variable* which we are trying to make equal to the setpoint. The difference between *Cv* and *Sp* is the *error signal (e)* that drives the *output variable Out* used to manipulate the *controlled variable*. There are three gain factors *Kp*, *Ki*, *Kd* — called *tuning parameters* — for each of the three modes of the controller: *proportional*, *integral* and *derivative*. Setting a gain factor to zero effectively disables that particular mode. Typical controller operation is either:

*Proportional-only (P)*

*Proportional plus reset (integral) (PI)*

*Proportional plus reset plus rate (PID)*

## Application Note — BAS Remote and Sedona Component Descriptions

In HVAC applications, P and PI are the most common. PID is seldom used.

*Enable* must be set true if loop action is to occur. If *Enable* is set to false, control action ceases and the output will remain at its last state.

If *Direct* is equal to true then the output will increase if the Cv becomes greater than Sp. If this was a temperature loop, this would be considered being in *cooling mode*. If *Direct* is equal to false, then the output will decrease if the Cv becomes less than the Sp. If this was a temperature loop, this would be considered being in *heating mode*. Notice that by entering negative gain factors, the action of the controller is reversed.

*Max* and *Min* are limits on the output's swing and are considered the absolute boundaries to the controller's throttling range (proportional control range). Basically, the LP component includes Limiter functionality.

*Bias* sets the output's offset. Sometimes *bias* is called manual reset to correct an output error with a large proportional band. It is usually only used with proportional-only control. The amount of bias is not influenced by the proportional gain *Kp*. Bias is also used on split-range control systems that will be discussed shortly.

*Ex Time* is the amount of time in milliseconds that the control loop is solved. Typical times are from 100–1000 ms. Most HVAC loops are slow acting and therefore solving loops faster brings no benefit.

In the following discussion on setting the gain factors, assume we need a temperature controller enabled for direct action and that the output can swing from 0–100%. When the output ranges from 50–100%, a proportional cooling valve is modulated. When the output ranges from 0–50%, a proportional heating valve is modulated. At 50% output no valve is open. This is called a split range control system — so we set the bias of the controller to 50%. *Max* and *Min* are set to 100 and 0 respectively. When we force the controller output from maximum heat to maximum cooling (0–100% output change), we notice that we can effect a change in our process temperature of 20°. This becomes our throttling range. In the real world, conducting this test might be difficult.

Now we need to set the three tuning parameters. We first begin by setting *Ki* and *Kd* to zero — thereby creating a proportional-only controller. The controller equation therefore becomes:

$$Out = Kp(e) + Bias \text{ where } e = Cv - Sp$$

Our first guess at *Kp* is 5 because we know that a 100% change in output yielded a 20° change in process temperature. This assumes that we can cool with the same efficiency as we can heat which may not be the case. By having a *Kp* of 5, the output will remain linear over this wide range. Notice that if there is no error signal (*Cv-Sp* is equal to zero), the output will then equal the *bias*. The value 5 is entered into *Kp* and a disturbance is introduced into the process such as a step change in the setpoint. If the process continues to oscillate between heating and cooling and never settles down, then we must reduce our proportional gain *Kp* which increases our proportional band ( $1/Kp$  times 100% is the proportional band). Assume we achieve a stable system with *Kp* at 5 (proportional band at 20%) but based on the load on the system. We notice that the output reached 70%. Our setpoint is at 70°, but our controlled temperature is 74°. Temperature is stable, but we have a 4° offset. This is the inherent difficulty with proportional-only control — there is an offset depending upon the value of the output. We have two choices. We can increase the proportional gain to 10 because we do not need a 20° range in input, but we risk oscillation. The second approach is to “reset the output manually” by increasing the bias. Approach one will never solve the problem but will minimize it, and there is a better method to approach two and that is called

## Application Note — BAS Remote and Sedona Component Descriptions

*automatic reset* — or adding reset action by adding a  $K_i$  term. The new controller equation becomes:

$$Out = K_p(e + K_i \int e dt) + Bias$$

If there remains an error signal ( $e \neq 0$ ), then the integral of the error over time will continue to drive the output until the error is driven to zero. The amount of action is determined by the  $K_i$  term. Notice that the integral term in the equation is also multiplied by the proportional gain before being applied to the output. The  $K_i$  coefficient is defined in units of repeats per minute. Too large a value can cause overshoot while too small a value will make the control system sluggish. The final setting  $K_p$  and  $K_i$  is done in the field based upon system response.

The third parameter is the rate parameter  $K_d$  which acts upon the rate of change of the error signal. Adding this term changes the controller equation as follows:

$$Out = K_p(e + K_i \int e dt + K_d de/dt) + Bias$$

For processes with extremely long reaction times, derivative control could be helpful in reducing overshoot.  $K_d$  is entered in seconds. As mentioned before, it is seldom used because tuning a control loop with three parameters can be challenging.

LSeq	
control::LSeq	<input type="checkbox"/>
In	0.0
In Min	0.0
In Max	100.0
Num Outs	16
D On	0
Ovfl	false
Out1	false
Out2	false
Out3	false
Out4	false
Out5	false
Out6	false
Out7	false
Out8	false
Out9	false
Out10	false
Out11	false
Out12	false
Out13	false
Out14	false
Out15	false
Out16	false

### Linear Sequencer — bar graph representation of input value

There are two internally configurable floats called *In Min* and *In Max* that set the range of input values. An internal configurable integer — called *Num Outs* — specifies the intended number of active outputs. By dividing the input range by the number of active outputs, the *delta* between outputs is determined. Outputs will turn on sequentially from Out1 to Out16 within the input range as a function of increasing input value.

For example: *In Min* is set to 100, *In Max* to 1100, and *Num Outs* is set to 10. The following is true:

*If In = 50 then Out1-16 are false and D On is zero.*

*If In = 899 then Out1-7 are true and Out8-16 are false. D On is 7.*


*If In = 1501 then Out1-14 are true and Out15-16 are false. D On is 14 and Ovfl is true.*

Mul2	
control::Mul2	<input type="checkbox"/>
Out	0.0
In1	0.0
In2	0.0

### Multiply two — results in the multiplication of two floats

$$Out = In1 * In2$$

## Application Note — BAS Remote and Sedona Component Descriptions

<b>Mul4</b>	
control::Mul4	
Out	0.0
In1	0.0
In2	0.0
In3	0.0
In4	0.0


**Multiply four** — results in the multiplication of four floats

$$Out = In1 * In2 * In3 * In4$$

<b>Neg</b>	
control::Neg	
Out	0.0
In	0.0


**Negate** — changes the sign of a float

$$Out = - In$$

<b>Not</b>	
control::Not	
Out	true
In	false


**Not** — inverts the state of a Boolean

$$Out = \overline{In}$$

<b>OneShot</b>	
control::OneShot	
Out	false
In	false
Pulse Width	0.0
Can Retrig	false


**Single Shot** — provides an adjustable pulse width to an input transition

Upon the input transitioning to true, the output will pulse true for the amount of time set in the configurable parameter *Pulse Width*. Time is in seconds. If the configurable parameter *Can Retrig* is set to true, the component will repeat its action on every positive transition of the input. For example in retrigger mode, a one-second *TickToc* connected to a *OneShot* with a 2 second pulse width setting will have the *OneShot* output in a continuous true state due to constant retriggering at a rate faster than the *OneShot* pulse width.

<b>Or2</b>	
control::Or2	
Out	false
In1	false
In2	false


**Two-input Boolean sum** — two-input OR gate

$$Out = In1 | In2$$

<b>Or4</b>	
control::Or4	
Out	false
In1	false
In2	false
In3	false
In4	false

**Four-input Boolean sum** — four-input OR gate


$$Out = In1 | In2 | In3 | In4$$

<b>Ramp</b>	
control::Ramp	
Out	93.0
Min	0.0
Max	100.0
Delta	1.0

**Ramp** — generates a repeating triangular wave

There are two configurable float parameters — *min* and *max*. For every scan cycle, the output increments by one unit until the output equals the *max* value at which time it decrements until *min* is reached. The result is a triangular wave with limits of *max* and *min* and an incremental rate of one unit per scan cycle.

## Application Note — BAS Remote and Sedona Component Descriptions

<b>ReheatS</b> 	
control::ReheatSe	
Out1	False
Out2	False
Out3	False
Out4	False
In	0.0
Enable	False
D On	0
Hysteresis	0.0
Threshold1	0.0
Threshold2	0.0
Threshold3	0.0
Threshold4	0.0

### Reheat Sequence — linear sequence up to four outputs

There are four configurable threshold points — *Threshold1* through *Threshold4* — that determine when a corresponding output will become true as follows:


*Out1 = true when In ≥ Threshold1*

*Out2 = true when In ≥ Threshold2*

*Out3 = true when In ≥ Threshold3*

*Out4 = true when In ≥ Threshold4*

These outputs will remain true until the input value falls below the corresponding threshold value by an amount greater than the configurable parameter *Hysteresis*. Output signal *D On* indicates how many outputs are true. Configurable parameter *Enable* must be true otherwise all outputs will be false.

<b>Reset</b> 	
control::Reset	
Out	0.0
In	0.0
In Min	0.0
In Max	4095.0
Out Min	0.0
Out Max	100.0


### Reset — output scales an input range between two limits

There are four configurable float parameters — *In Max*, *In Min*, *Out Max* and *Out Min* — which determine the input and output ranges respectively of the input and output. The output of this component will scale linearly with the value of the input if the input is within the input range. The input range (IR) is determined by *In Max-In Min* while the output range (OR) is determined by *Out Max-Out Min*. If the input is within the input range then the following is true:

$$Out = (In + In Min)(OR/IR) + Out Min$$

If the input exceeds, *In Max* then *Out = Out Max*.

If the input is less than, *In Min* then *Out = Out Min*.

<b>SRLatch</b> 	
control::SRLatch	
Out	False
S	False
R	False

### Set/Reset Latch — single-bit data storage


The following logic applies:

*If S is true and R is false then Out = true*

*If S is false and R is true then Out = false*

*If S is false and R is false then Out = Out from the last scan*


*If S is true and R is true then Out = false*

<b>Sub2</b> 	
control::Sub2	
Out	0.0
In1	0.0
In2	0.0

### Subtract two — results in the subtraction of two floats


$$Out = In1 - In2$$

## Application Note — BAS Remote and Sedona Component Descriptions

<b>Sub4</b>	
control::Sub4	
Out	0.0
In1	0.0
In2	0.0
In3	0.0
In4	0.0

### Subtract four — results in the subtraction of four floats


$Out = In1 - In2 - In3 - In4$

<b>TickToc</b>	
control::TickTock	
Out	false
Ticks Per Sec	1

### Ticking clock — an astable oscillator used as a time base

There is one configurable float parameter — *Ticks Per Sec* — which can range from a low of 1 to a high of 10 pulses per sec.

*Out = a square wave between 1 and 10 Hz*

<b>Tstat</b>	
control::Tstat	
Diff	0.0
Is Heating	false
Sp	0.0
Cv	0.0
Out	false
Raise	false
Lower	false

### Thermostat — on/off temperature controller


The configurable float parameter — *Diff* — provides hysteresis and deadband. Another configurable parameter — *Is Heating* — indicates a heating application. *Sp* is the *setpoint* input and *Cv* is the *controlled variable* input. *Raise* and *lower* are outputs.

If  $Cv > (Sp + Diff/2)$  then *Lower* is true and will remain true until  $Cv < Sp$

If  $Cv < (Sp - Diff/2)$  then *Raise* is true and will remain true until  $Cv > Sp$

If *Is Heating* is true then *Out = Lower*

If *is Heating* is false then *Out = Raise*


<b>UpDn</b>	
control::UpDn	
Out	0.0
Ovr	false
In	false
Rst	false
C Dwn	false
Limit	0.0
Hold At Limit	false

### Up/down counter — up/down float counter

The counter range is between zero and a value that can be set with configurable parameter *Limit*. To cease counting at the limit set the configurable parameter *Hold at Limit* to true. To count down instead of up, set *C Dwn* to true. To reset the counter to zero set *Rst* to true. *Ovr* is the overflow indicator. *In* is the Boolean count input.

*Out = the current count*


*If Out ≥ Limit then Ovr is true*

<b>WriteBo</b>	
control::WriteBool	
In	false
Out	false

### Write Boolean — setting a writable Boolean value

*Out = In*

Unlike *ConstBo*, this component has an input.

<b>WriteFl</b>	
control::WriteFLoa	
In	0.0
Out	0.0

### Write Float — setting a writable float value

*Out = In*

Unlike *ConstFl*, this component has an input


## Application Note — BAS Remote and Sedona Component Descriptions

<b>WriteIn</b>	
control::WriteInt	
In	0
Out	0

### Write Integer — setting an integer value

$$Out = In$$

Unlike *ConstIn*, this component has an input.

<b>Xor</b>	
control::Xor	
Out	false
In1	false
In2	false

### Two-input exclusive Boolean sum — two-input XOR gate

$$Out = In1 \oplus In2 = In1 \cdot \overline{In2} + \overline{In1} \cdot In2$$

## BAS Remote Kit

The BAS Remote kit allows Sedona applications to tie into real world inputs and outputs via the BAS Remote. All components must first be configured for an Object Instance that is supported by the BAS Remote hardware. Access the property sheet for the component being commissioned after first dragging it onto the wire sheet. Enter the Object Instance which is the same one used for a BACnet point. Refer to the BAS Remote User Manual for details. In order for the online status to revert to true, the point must be properly configured, must be actively scanned by the hardware and not in a forced state.

<b>InpBool</b>	
basremote::InpBc	
Out	false
Online	false

### Input Boolean — BAS Remote binary input

*Out = value of the real world binary input*

<b>InpFlea</b>	
basremote::InpFl	
Out	0.0
Online	false

### Input Float — BAS Remote analog input or value

*Out = value of the real world analog input*

<b>OutBool</b>	
basremote::OutB	
In	false
Online	false

### Output Boolean — BAS Remote binary output

*In = Boolean variable to be written to a real world output*

<b>OutFlea</b>	
basremote::OutFl	
In	0.0
Online	false

### Output Float — BAS Remote analog output

*In = Float variable to be written to a real world output*

## Basic Schedule Kit

*DailySchedule* represents a simple daily schedule with up to two active periods. Each active period is defined by a start time and duration. If the duration is zero, the period is disabled. If the periods overlap, then the first period (defined by *Start1* and *Dur1*) takes precedence. If the duration extends past midnight, then the active period will span two separate calendar days. There are two components in the kit — one for Boolean outputs and the other for floats. Both kits rely upon the time being set in the target hardware.

Duration periods — *Dur1* and *Dur2* — are configured in minutes from zero to 1440 minutes.

<b>DailySc</b>	
basicSchedule::D:	
Start1	0 min
Dur1	0 min
Start2	0 min
Dur2	0 min
Val1	False
Val2	False
Def Val	False
Out	False

### Daily Schedule Boolean — two-period Boolean scheduler

Configure *Def Val* to the intended output value if there are no active periods. Configure *Val1* and *Val2* for the desired output values during period 1 and period 2 respectively.

*Out = Def Val if no active periods*  
*Out = Val1 if period 1 is active*  
*Out = Val2 if period 2 is active*

<b>DailyS1</b>	
basicSchedule::D:	
Start1	0 min
Dur1	0 min
Start2	0 min
Dur2	0 min
Val1	0.0
Val2	0.0
Def Val	0.0
Out	0.0

### Daily Schedule Float — two-period float scheduler

Configure *Def Val* to the intended output value if there are no active periods. Configure *Val1* and *Val2* for the desired output values during period 1 and period 2 respectively.

*Out = Def Val if no active periods*  
*Out = Val1 if period 1 is active*  
*Out = Val2 if period 2 is active*

#### 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](mailto:info@ccontrols.com)  
[www.ccontrols.com](http://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](mailto:info@ccontrols.com.cn)  
[www.ccontrols.asia](http://www.ccontrols.asia)

#### United Kingdom

**Contemporary Controls Ltd**  
 Sovereign Court Two  
 University of Warwick  
 Science Park  
 Sir William Lyons Road  
 Coventry CV4 7EZ  
 United Kingdom

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

[info@ccontrols.co.uk](mailto:info@ccontrols.co.uk)  
[www.ccontrols.eu](http://www.ccontrols.eu)

#### Germany

**Contemporary Controls GmbH**  
 Fuggerstraße 1 B  
 04158 Leipzig  
 Germany

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

[info@ccontrols.de](mailto:info@ccontrols.de)  
[www.ccontrols.eu](http://www.ccontrols.eu)