Sedona Framework – Best Opportunity for Open Control

Introduction

Zachary Netsov, Product Specialist, Sedona Alliance



THE WORLD'S LARGEST HVACR MARKETPLACE

JAN 30-FEB 1 LAS VEGAS 2017

The Need for Open Controllers

- When we mention open controllers we immediately think of BACnet, but BACnet is only a protocol and does not address control
- Even with BACnet compliance, a system integrator is not assured access to a BACnet site
 - The programming language may be proprietary to the controller manufacturer
 - The programming tool may only be available to the controller manufacturer's sales channel

Therefore, an open protocol like BACnet is necessary for an open controller but it is not sufficient



- Utilizes an open protocol for network communications
 - BACnet is an ISO standard with international acceptance
- Utilizes an open programming language for implementing control strategies
 - Sedona Framework is open source, and due to its similarity to Niagara Framework it is familiar to many integrators
- Utilizes a programming tool available without restriction
 - Those without access to Niagara Workbench can use Sedona Application Editor from Contemporary Controls or Sedona tools from others
- Fosters a community of developers and integrators that share technology for the public good
 - A Sedona community of developers and integrators exist using the resources at SedonaDev.org and the Sedona Alliance



Open Protocol for Network Communications – BACnet

- BACnet a communications protocol for Building Automation and Control Networks
- Intended to provide "interoperability" among different vendor's equipment
- Frees the building owner of being dependent upon one vendor for system expansion
- Allows BAS devices to be modeled such that they are "network viewable"
- BACnet devices are modeled using an object-oriented structure of ...
 - Objects
 - Properties
 - Services





Open Programming Language for Control – Sedona

- The Sedona language is similar to Java or C# allowing developers the opportunity to create custom components
- These components can be assembled into applications by nonprogrammers using simple graphical methods
- A Sedona Virtual Machine (SVM) on the Sedona device executes the application program
- Sedona applications can be made to be portable to other Sedona devices
- Sedona is open source there are no royalties or commercial licenses required to develop and use Sedona components



Creating Applications by Linking Components



Using a drag-and-drop methodology, Sedona components are placed onto a wire sheet, configured, and linked together to create an application. Once placed on the wire sheet, components immediately begin execution thereby allowing for application debugging in real-time. 6



- Originally developed by Tridium as a software framework for embedded controllers operating with less than 100kB of memory, the technology is accessible from the SedonaDev.Org web site
- Tridium owns the trademark Sedona Frameworktm but the technology is available to the public licensed under the Academic Free License version 3.0 with numerous products in existance
- The public has the right to use, develop and sell products based upon the Sedona Framework without royalties or commercial licenses but should acknowledge the copyright owner along with stating that the product was built on the Sedona Frameworktm





Programming Tool Available without Restriction – Sedona Applications Editor

- For those without access to Niagara Workbench, the Sedona Application Editor (SAE) is available free via download from the Contemporary Controls website
- Includes a Sedona virtual machine (SVM-PC) that runs on a PC that can be programmed with the SAE for testing
- Includes Tridium-Release kits and components
- Can be used with other Sedona devices as long as the proper platforms, kits and manifests are installed
- Intended for the Sedona community





- The Sedona community consists of developers and integrators
- A developer is a skilled software professional or manufacturer who can
 - Create custom components beyond the standard components from Tridium some of which can be shared with others
 - Can modify the sample Sedona Virtual Machine to meet the hardware requirements of the target Sedona device
 - Can develop software tools for editing Sedona applications
- The integrator is a non-programmer with knowledge of control applications
 - Can assemble components onto a wire sheet to create a control strategy meeting a defined Sequence of Operation
 - May share with other integrators proven applications to benefit all integrators



How are Sedona HVAC applications produced?





- A Sedona developer is either a hardware manufacturer or a software developer
- Physical hardware such as CPU, memory and I/O need to be designed
- The Sedona Virtual Machine must be modified to accommodate the hardware platform
- Custom kits called hardware-dependent kits need to be developed that support the native functions of the platform
- Once all elements are put together you will have a Sedona device awaiting an application



Kits Sedona Virtual Machine (SVM) Hardware



- A Sedona Virtual Machine (SVM) is a small portable fast interpreter that can reside on most any hardware platform or operating system while executing the same Sedona application
- The original Tridium SVM has been modified by developers to run on limited resource microcontrollers, Linux platforms, and powerful Windows workstations
- Intended to operate over IP networks



This SVM runs on a Windows PC

SVMs for Raspberry Pi Extensions





What is the Role of the System Integrator?

- The system integrator translates the required sequence of operation (SOO) into a Sedona application that executes the sequence
- Applications are created by extracting components from kits, placing them onto a wire sheet, configuring the components if necessary, and interconnecting the components with links
- Because of the system integrators' knowledge, the SI recommends to the developer any custom components that need to be developed that can be shared by all





What is the Difference Between Sedona Kits and Components?

- Components are the fundamental building blocks for creating applications
- However, components are deployed into a Sedona device in a container called a kit
- Similar types of components are assigned to kits with relevant names such as Math, Logic, HVAC and so on.
- There are three types of kits
 - Original Sedona 1.2 kits provided by Tridium available to all
 - Custom hardware-independent kits by Sedona developers that can be shared
 - Custom hardware-dependent kits by Sedona developers that cannot be shared
- The spirit of the Sedona Community is to share kits if possible



- With the Sedona 1.2 release, Tridium restructured their Control kit into several smaller manageable kits which we call the Tridium-release kits
- It is recommend that they not be modified from their release form so that they can be shared by the community

basicSchedule	math
datetimeSTD	pricomp
func	sys
hvac	timing
logic	types

There are 69 unique components in these kits



Tridium Time and Schedule Kits – datetimeSTD, basicSchedule

The Scheduling Group scheduling operations based on time of day DailyScheduleBoolDaily Schedule BoolDailyScheduleFloatDaily Schedule FloatDateTimeServiceSTDTime of Day — time

Daily Schedule Boolean — two-period Boolean scheduler
Daily Schedule Float — two-period float scheduler **D** Time of Day — time, day, month, year

0:0 0:0 0:0 0:0 0.0 0.0 0.0

DateTim	
datetimeStd::Dat	teTimeServiceStd
Nanos	53801112500000000
Hour	23
Minute	32
Second	Į.
Year	2017
Month	1
Day	17
DayOfWeek	2
UtcOffset	(
OsUtcOffset	false
Tz	

DailySc		Daily S1
basicSchedule::DailyScheduleBool		basicSchedule::DailyScheduleFloat
Start1	0:0	Start1
Dur1	0:0	Dur1
Start2	0:0	Start2
Dur2	0:0	Dur2
Val1	false	Val1
Val2	false	Val2
DefVal	false	DefVal
Out	false	Out



The Function Group

convenient functions for developing control schemes

Cmpr	Comparison math — comparison (<=>) of two floats
Count	Integer counter — up/down counter with integer output
Freq	Pulse frequency — calculates the input pulse frequency
Hysteresi	s Hysteresis — setting on/off trip points to an input variable
IRamp	IRamp — generates a repeating triangular wave with an integer output
Limiter	Limiter — Restricts output within upper and lower bounds
Linearize	Linearize — piecewise linearization of a float
LP	LP — proportional, integral, derivative (PID) loop controller
Ramp	Ramp — generates a repeating triangular or sawtooth wave with a float output
SRLatch	Set/Reset Latch — single-bit data storage
TickToc	Ticking clock — an astable oscillator used as a time base
UpDn	Float counter — up/down counter with float output

Cmpr	<	Count 🗾	Lineari	×v	Hystere	л	IRamp	105 C	LP	•
func::Cmpr		func::Count	func::Linearize		func::Hysteresis		func::IRamp		func::LP	
Xgy	false	Out (Out	nul	In	0.0	Out	69	Enable	true
Xey	true	In false	e In	0.0	Out f	false	Min	0	Sp	0.0
Xly	false	Preset (X0	0.0	RisingEdge	50.0	Max	100	Cv	0
Х	0.0	Dir u	YO	0.0	FallingEdge	50.0	Delta	1	Out	0.0
Y	0.0	Enable false	X1	0.0			Secs	1	Кр	1
		R fals	Y1	0.0					Ki	0
	_		X2	0.0	SRLatch	1			Kd	0
Limiter	× .		Y2	0.0	func::SRLatch		TickToc	145	Max	100
func::Limiter		Ramp 🚾	X3	0.0	Out t	alse	func::TickTock		Min	0
Out	0.0	func::Ramp	- Y3	0.0	s t	alse	Out	false	Bias	0
In	0.0	Out 84.	7 ×4	0.0	R f	alse	TicksPerSec	1	MaxDelta	0
LowLmt	0.0	Min 0.	0 Y4	0.0	L				Direct	true
HighLmt	0.0	Max 100.	0 X5	0.0	UpDn	1			ExTime	1000
		Period 1	0 Y5	0.0	func::UnDn	•				
From		RampType triang	e X6	0.0	Out	0.0				
Freq	<u> </u>		V6	0.0	Ovr f	false				
Pns	0		X7	0.0	In f	false				
Pom	0		V7	0.0	Rst f	false				
In In	falee		V8	0.0	CDwn f	alse				
	10130		×8	0.0	Limit	0.0				
<u></u>			Y9	0.0	HoldAtl imit f	false				
			×9	0.0		4.50				
			19	0.0	<u></u>					



LSeq

Reset

ReheatSeq

The HVAC Group

operations that facilitate control

InMin	0.0
InMax	100.0
NumOuts	16
Delta	5.88
DOn	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
Ovfl	false

Tstat	Thermos
000	10100
Out2	false
Out3	false
Out4	false
In	0.0
Enable	false
DOn	0
Hysteresis	0.0
Threshold1	0.0
Threshold2	0.0
Threshold3	0.0
Threshold4	0.0

In InMin InMax OutMin OutMax

Linear Seque	encer — bar gra	ph rep	oresentatio	n of input value
Reheat sequ	ence — linear s	seque	nce up to fo	our outputs
Reset — out	put scales an in	put ra	nge betwee	en two limits
Thermostat -	- on/off temper	ature	controller	
14104		0.0	with the second s	0.0

0.0	200	0.0
0.0	IsHeating	false
0.0	Sp	0.0
4095.0	Cv	0.0
0.0	Out	false
100.0	Raise	false
	Lower	false



The Logic Group

logical operations using Boolean variables

ADemux2	Analog Demux — Single-input, two-output analog de-multiplexe
And2	Two-input Boolean product — two-input AND gate
And4	Four-input Boolean product — four-input AND gate
ASW	Analog switch — selection between two float variables
ASW4	Analog switch — selection between four floats
B2P	Binary to pulse — simple mono-stable oscillator (single-shot)
BSW	Boolean switch — selection between two Boolean variables
Demuxl2B4	Four-output Demux — integer to Boolean de-multiplexer
ISW	Integer switch — selection between two integer variables
Not	Not — inverts the state of a Boolean
Or2	Two-input Boolean sum — two-input OR gate
Or4	Four-input Boolean sum — four-input OR gate
Xor	Two-input exclusive Boolean sum — two-input XOR gate

ADemux2		ASW4	24	And4	ê	Demuxl2	-	ISW	-	Or2	Ш
logic::ADemux2		logic::ASW4		logic::And4		logic::Demuxl2B4		logic::ISW		logic::Or2	
Out1	0.0	Out	0.0	Out	false	In	0	Out	0	Out	false
Out2	0.0	In1	0.0	in1	false	Out1	true	In1	0	In1	false
In	0.0	In2	0.0	ln2	false	Out2	false	In2	0	In2	false
S1	false	In3	0.0	In3	false	Out3	false	S1	false		
		In4	0.0	In4	false	Out4	false				
		StartsAt	0			StartsAt	0			Not	<u>.</u>
ASW	-	Sel	0					Or4	<u>11</u>	logic::Not	
logic::ASW				BSW	-	·		logic::Or4		Out	true
Out	0.0			logic::BSW		And2	2	Out	false	In	false
in1	0.0	B2D	X	Out	false	logic::And2	-	In1	false		
In2	0.0	logic::B2P	· · · ·	In1	false	Out	false	In2	false	Vor	
S1	false	Out	false	In2	false	In1	false	In3	false	AUI Ingie::Xor	
		In	false	S1	false	In2	false	In4	false	Out	false
										In1	false
		-								ln2	false



	Add2	Two-input addition — results in the addition of two floats
	Add4	Four-input addition — results in the addition of four floats
	Avg10	Average of 10 — sums the last ten floats and divides by ten to provide a running average
	AvgN	Average of N — sums the last N floats and divides by N to provide a running average
The Math	Div2	Divide two — results in the division of two float variables
Group	FloatOffset	Float offset — float shifted by a fixed amount
Group	Max	Maximum selector — selects the greater of two inputs
matn-based	Min	Minimum selector — selects the lesser of two inputs
components	MinMax	Min/Max detector — records both the maximum and minimum values of a float
	Mul2	Multiply two — results in the multiplication of two floats
	Mul4	Multiply four — results in the multiplication of four floats
	Neg	Negate — changes the sign of a float
	Round	Round — rounds a float to the nearest N places
	Sub2	Subtract two — results in the subtraction of two floats
	Sub4	Subtract four — results in the subtraction of four floats
	TimeAvg	Time average — average value of float over time

Add2	٠	Avg10 🔨		Div2 +		Mul4 💌	Μ	Ain 🦄	× .	FloatOf +		AvgN	л	
math::Add2		math::Avg10		math::Div2		math::Mul4	m	nath::Min		math::FloatOffset		math::AvgN		_
Out	0.0	Out n	านไ	Out 0.0	.0	Out 0.0	0 Ou	ut	0.0	Out (0.0	Out	0.	0
In1	0.0	In O	0.0	In1 0.0	.0	In1 0.0	0 In1	1	0.0	In (0.0	In	0.	0
In2	0.0	MaxTime	0	In2 0.0	.0	ln2 0.0	0 In2	2	0.0	Offset (D.O	NumSamplesToAvg	1	5
				Div0 true	ie	In3 0.0	0			L		Reset	fals	e
						In4 0.0	0 -]
MinMax	N	Mul2 🔹			1		M	lax 🎽	N	TimeAvg 🗛			_	
math::MinMax		math::Mul2		Sub2 -			m	nath::Max		math::TimeAvg		Sub4	-	
MinOut	0.0	Out 0	0.0	math::Sub2		Add4 +	Ou	ut	0.0	Out (0.0	math::Sub4	_	
MaxOut	0.0	ln1 0	0.0	Out 0.0	.0	math::Add4	In1	1	0.0	In (0.0	Out	0.0	
In	0.0	In2 0	0.0	In1 0.0	.0	Out 0.0	0 ln2	2	0.0	Time 100	00	In1	0.0	
R	false			In2 0.0	.0	In1 0.0	0					In2	0.0	
				L		In2 0.0	0 🗖	Nexuel 1		Nee		In3	0.0	
						In3 0.0	0 8	cound	•	methulas		In4	0.0	
						In4 0.0	0	ut	0	MathNeg	10			
								01			2.0			
						<u></u>	_ in		0	in (0.0			~ 1
							De	ecimalPlaces	0	L				21



The Priority Group prioritizing actions of Boolean, Float and Integer variables PrioritizedBool PrioritizedFloat PrioritizedInt

Prioritized Boolean output — highest of sixteen outputs Prioritized float output — highest of sixteen outputs Prioritized integer output — highest of sixteen outputs

Priorit		Priori1		Priori2	
pricomp::PrioritizedBool	fallback	SourceLevel	fallback	pricomp::Prioritizedint SourceLevel	fallback
OverrideEvel		OverrideEvel		OverrideEvoTime	Iailuack
In1	null	In1	null	In1	min
In2	null	In2	null	In2	min
In3	null	In3	null	In3	min
In4	null	In4	null	In4	min
In5	null	In5	null	In5	min
In6	null	In6	null	In6	min
In7	null	In7	null	In7	min
In8	null	In8	null	In8	min
In9	null	In9	null	In9	min
In10	null	In10	null	In10	min
In11	null	In11	null	In11	min
In12	null	In12	null	In12	min
In13	null	In13	null	In13	min
In14	null	In14	null	In14	min
In15	null	In15	null	In15	min
In16	null	In16	null	In16	min
Fallback	null	Fallback	null	Fallback	min
Out	null	Out	null	Out	min
MinActiveTime	0				
MinInactiveTime	0				



The System Group

platform and folder components

Folder **RateFolder**

PlatformService Platform service — indicates platform and available memory Folder — when accessed opens to another wire sheet Rate Folder — a folder that can be used for background tasks

Folder sys::Folder	RateFol sys::RateFolder		Platfor sys::PlatformSer	vice
	AppCyclesToSkip	0	PlatformId	ccontrols-BASC22-3.1.0
			PlatformVer	BAScontrol 2.0.1
			MemAvailable	9024



	DlyOff	Off delay timer — time of
The Timing Group	DiyOn	On delay timer — time c
time-based components	OneShot	Single Shot — provides
	Timer	Timer — countdown tim

Off delay timer — time delay from a "true" to "false" transition of the input On delay timer — time delay from an "false" to "true" transition of the input Single Shot — provides an adjustable pulse width to an input transition Timer — countdown timer

DlyOff timing::DlyOff	N	DlyOn timing::DlyOn	N	OneShot timing::OneShot	N	Timer timing::Timer	Л
Out	false	Out	false	Out	false	Out	false
In	false	In	false	In	false	Run	stop
DelayTime	0.0	DelayTime	0.0	PulseWidth	0.0	Time	0
Hold	0	Hold	0	CanRetrig	false	Left	0



The Types Group variable types and conversion between types	B2F ConstBool ConstFloat ConstInt F2B F2I I2F L2F WriteBool WriteFloat WriteInt	Binary to float encoder — 16-bit binary to float conversion Boolean constant — a predefined Boolean value Float constant — a predefined float variable Integer constant — a predefined integer variable Float to binary decoder — float to 16-bit binary conversion Float to integer — float to integer conversion Integer to float — integer to float conversion Long to float — long integer to float conversion Write Boolean — setting a writable Boolean value Write Float — setting a writable float value Write integer — setting an integer value
--	---	---

ConstBo	•	ConstFl	•	Constin	•	B2F	•	F2B	-	F2I	-
types::ConstBool		types::ConstFloat		types::ConstInt		types::B2F		types::F2B		types::F2I	
Out	false	Out	0.0	Out	0	Out	0.0	In	0.0	In	0.0
						Count	0.0	Out1	false	Out	0
						In1	false	Out2	false		
WriteDo		WriteEl		Writeln		In2	false	Out3	false		
types::WriteBool	•	types::WriteEleat	•	types::WriteInt	•	In3	false	Out4	false	IOF	-
In In	false	In In	0.0	In In	0	In4	false	Out5	false	IZF	2
Out	false	Out	0.0	Out	0	In5	false	Out6	false	lypesizr	0
- Cut	Talao	- Cut	0.0	- Cut	Ŭ	In6	false	Out7	false	Out	0.0
						In7	false	Out8	false	Uut	0.0
						In8	false	Out9	false		
						In9	false	Out10	false		
						In10	false	Out11	false	L2F	≠
						In11	false	Out12	false	types::L2F	
						In12	false	Out13	false	In	0

In13

In14

In15

In16

0.0

Out

false

false

false

false

Out14

Out15

Out16

Ovrf

false

false

false

false



Custom Hardware – Independent Kits Developer Supplied

- All non-Tridium-Release kits are called custom kits
- Custom kits that operate independent of specific hardware are called hardware-independent kits
- Unlike Tridium-release kits, custom kits must be identified by their developer
- It is encouraged that custom hardware-independent kits be shared by the Sedona community

There are numerous custom kits and components from the Sedona community



Custom Hardware – Independent Kit Function – CControls_Function

Custom Functions collection of helpful components	Cand2 Cand4 Cand6 Cand8 Cmt Cor2 Cor4 Cor4 Cor6 Cor8 CtoF Dff	Two-input Boolean product — two-input AND/NAND gate with complete Four-input Boolean product — four-input AND/NAND gate with complete Six-input Boolean product — six-input AND/NAND gate with complete Eight-input Boolean product — eight-input AND/NAND gate with complete Comment — comment field up to 64 characters Two-input Boolean sum — two-input OR/NOR gate with complement Four-input Boolean sum — four-input OR/NOR gate with complement Six-input Boolean sum — six-input OR/NOR gate with complement Completer Six-input Boolean sum — eight-input OR/NOR gate with complement Eight-input Boolean sum — eight-input OR/NOR gate with complement Completer Six-input Boolean sum — eight-input OR/NOR gate with complement Completer Six-Input Boolean sum — eight-input OR/NOR gate with complement Completer Six-Input Boolean sum — eight-input OR/NOR gate with complement Six-Input Boolean sum — eight-input OR/NOR gate with complement Completer Six-Input Boolean sum — eight-input OR/NOR gate with complement Six-Input Boolean sum — eight-input OR/NOR gate with complement Completer Six	lementary outputs olementary outputs mentary outputs nplementary outputs tary outputs ntary outputs ry outputs entary outputs
	FtoC HLpre PsychrE PsychrS	°F to °C — Fahrenheit to Celsius temperature conversion High-Low Preset — defined logical true and false states Psychrometric Calculator — English units Psychrometric Calculator — SI units	This custom kit was developed by Contemporary Controls
	JULALCH	Serviear Laton — level-triggered Single-bit data storage	

SCLatch CControls Function::SCLatch	N	Cand2 & CControls Function::Cand2	Cand4 2 CControls Function::Cand4		Cand6 &	Cand8 2 CControls Function::Cand8	PsychrE ≠ CControls Function::PsychrE	CtoF ECCOntrols Function::CtoF
Set	false	Inp1 false	Inp1 false	e Ir	np1 false	Inp1 false	InTempDegF 0.0	InTempDegC 0.0
Clear	false	Inp2 false	Inp2 false	e Ir	np2 false	Inp2 false	InRelativeHumidityPct 0.0	OutTempDegF 32.0
Out	false	Out false	Inp3 false	e Ir	np3 false	Inp3 false	OutDewPointDegF 0.0	
OutNot	true	OutNot true	Inp4 false	ie Ir	np4 false	Inp4 false	OutEnthalpyBtu_per_lb 0.0	111
			Out false	ie Ir	np5 false	Inp5 false	OutSatPressure_psi 0.0	Controls Euloction: HL pre
A			OutNot true	ie Ir	np6 false	Inp6 false	OutVaporPressure_psi 0.0	Out true
Cmt		Cor8		0	Out false	Inp7 false	OutWetBulbTempDegF 0.0	OutNot false
Controls Function::Cmt		CCONTROLS FUNCTION::CORS		Ō	DutNot true	Inp8 false		
Comment		inpi iaise	Cor6 II			Out false		
		inp2 Taise	CControls Function::Cor6	- 0		OutNot true	PsychrS =	Dff 🔉 🔪
		inp3 faise	Inp1 false	ie (Cor4 II		CControls Function::PsychrS	CControls Function::Dff
FtoC	12	Inp4 taise	Inp2 false	e (CControls Function::Cor4		InTempDegC 0.0	Preset false
CControls Function::FtoC	-	Inp5 false	Inp3 false	ie Ir	np1 false	Cor2	InRelativeHumidityPct 0.0	Reset false
InTempDegF	0.0	Inp6 false	Inp4 false	ie Ir	np2 false	CControls Function::Cor2	OutDewPointDegC 0.0	D false
OutTempDegC -1	7.77	Inp7 false	Inp5 false	e Ir	np3 false	Inp1 false	OutEnthalpy_kJ_per_kg 0.0	Clk false
		Inp8 false	Inp6 false	ie Ir	np4 false	Inp2 false	OutSatPressure_kPa 0.0	Out false
-		Out false	Out false	e C	Out false	Out false	OutVaporPressure_kPa 0.0	OutNot true
		OutNot true	OutNot true	ie C	OutNot true	OutNot true	OutWetBulbTempDegC 0.0	
		L,		11				



Custom Hardware – Independent Kit HVAC Kit – CControls_HVAC

	AnaHiLo	Analog High/Low — analog variable out-of-range limit or detection
	AntiSCY	Anti-Short Cycle — minimum run time and minimum start time limiter
Custom	BTUh	BTU/Hour Calculator — calculates energy usage based on temperature difference and flow
HVAC	NumDamp	Numeric Dampener — digital filter dampens amplitude and rate changes
	EnhPID	Enhanced PID Loop Controller — same as LP component except with better output control
	LeadLag	Lead Lag Sequence Controller — lead/lag control for up to four devices
components	OATrueB	Outside Air True Blend — percentage of outside air based on OAT, MAT and RAT
	RnProof	Run Proving — verifies that a commanded device indeed executes
	TockTic	Period Driven Clock — similar to TickToc component but with period control

AnaHiLo	N	EnhPID	LeadLag	N.C. LeadLag	NumDamp CControls_HVAC::NumDamp	N	BTUh CControls HVAC.BTUh	N	RnProof	
LimitDelay	1	Enable tr	ue RunTime	10	UpdateInterval	5	ExeDelay	0	ProofDelay	1 ExeDelay 1
HiLimit	10	Sp (.0 ProofDelay	1	RiseIncrement	0.5	OffCal	0.0	In fals	e OffCal 0.0
LoLimit	-10	Cv	0 OverlapTime	0	FallDecrement	0.5	InGPM	0.0	Proof fals	OutsideAT 0.0
Differential	0.1	Out (.0 OutQty	Two	RiseDampInhibit	false	InTemp	0.0	Out fals	e ReturnAT 0.0
HoldAtLimit	false	Кр	1 In	false	FallDampInhibit	false	OutTemp	0.0	OutNot true	MixedAT 0.0
LimitOutEnable	false	Ki	0 OutA	false	In	0.0	Out	0.0	Fail fals	e Output 0.0
In	0	Kd	0 OutB	false	Out	0.0	TonOutR	0.0	Faillnhibit fals	e Fault true
Out	0	Max 1	00 OutC	false			TonOutC	0.0		
OverLimit	false	Min	0 OutD	false						
UnderLimit	false	Bias	0 ProofA	false	AntiSCY 🔨			_		
		MaxDelta	0 ProofB	false	CControls HVAC::AntiSCY		TockTic	200	This cus	tom kit was
		Direct tr	ue ProofC	false	MinRunTime 1		CControls HVAC::TockTi	c 📃	THIS CUS	LUIII KIL WAS
		ExTime 10	00 ProofD	false	MinOffTime 1		Period	1.0	dovo	loned by
			Alarm	false	In false		Enable	true	ueve	toped by
					Out false		Out	true	Contempo	rary Controls
			1		Reset false				contempt	



Custom Hardware – Independent Kit Math Kit – CControls_Math

Custom MATH

accommodate configurable inputs

Add Add two with configurable inputs — results in the addition of two floats
Sub Subtract two with configurable inputs — results in the subtraction of two floats
Mul Multiply two with configurable inputs — results in the multiplication of two floats
Div Divide two with configurable inputs — results in the division of two float variables

Add	+	Sub	-	Mul	×	Div	+
CControls Math::Add		CControls Math::Sub		CControls Math::Mul		CControls Math::Div	
Inp1	0.0	Inp1	0.0	Inp1	0.0	Inp1	0.0
Inp2	0.0	Inp2	0.0	Inp2	0.0	Inp2	0.0
Out	0.0	Out	0.0	Out	0.0	Out	0.0
						Div0	true

This custom kit was developed by Contemporary Controls



Hardware – Dependent Components BAScontrol20

UT1	UIS		LED Power 24 VDC 110% 4W 24 VAC 110% 6VA 47-63 Hz HI: DC+ or AC HI COM: DC COM or AC LD Class 2 Circuits Only
UI3	U17	813	
A01	_ 801	BAScontrol2	C.(U).us
A02	_ B02 _ B03	IP	ND. CONT EQ. 4EA4 Ethernet
	_ 804 04	Reset IP B	Solid + Link Flashing = Data

A01-A04	Analog output — analog output voltage point
BI1-BI4	Binary input — binary input point
B01-B04	Binary output— binary output point
UI1-UI8	Universal input — binary, analog, thermistor, resistance or accumulator
ScanTim	Scan time monitor— records the min, max and average scan times
UC1-UC4	Retentive universal counters — up/down retentive counters
VT01-VT24	Virtual points — share wire sheet data with BACnet/IP clients
WC01-WC48	Web components — share wire sheet data with controller web pages

UI6		UC4		VT8		WC01			ScanTim	
CControls BAS	C20 IO::UI6	CControls BASC	20 IO::UC4	CControls BAS	C20 IO::/VT08	CControls BASC	20 Web::WC	01	CControls BASC20 II	D::ScanTim
ChnType	Input10V	Initialized	true	Initialized	true	WcType		Input	SampleSize	10
DutF	0.00	Count	0	ChnType	Floatinput	MinVal		0.0	TimeMs	44
DutB	false	CountF	0.0	Reset	false	MaxVal		100.0	MinimumMs	43
)utl	0	Ovf	true	FloatV	0.0	FltVal		0.0	MaximumMs	49
		Clk	false	BinaryV	false	IntVal		0	AverageMs	43
		Enable	true	WireSheet	InputTo	BinVal		false	Reset	false
		Rst	false							
404		CDwn	false				_			
Controls BAS	C20 IO::A04	Limit	0	BI4		BO2		plat		
pF	0.0	Hold∆tl imit	false	CControls BAS	C20 IO::BI4	CControls BASC20	IO::BO2	CControls	BASC20 Platform::BAS	C20PlatformService
nable	true		10.00	OutB	true	InpB	false	Platformld	(ccontrols-BASC20-3
		<u></u>				Enable	true	PlatformVer		BAScontrol 2
								MemAvailab	le	19

Hardware-dependent components cannot be shared because they use native functions.



Hardware – Dependent Component for the Metz DIO 4/2 MS/TP I/O Module

CControls Cub	el0DI04_2
Devinstance	-1
Inp1Use	NotUsed
Inp2Use	NotUsed
Inp3Use	NotUsed
Inp4Use	NotUsed
Out1Use	NotUsed
Out1Priority	10
Out2Use	NotUsed
Out2Priority	10
Inp1	false
Inp2	false
Inp3	false
Inp4	false
Out1	false
Relinquish1	false
Out2	false
Relinquish2	false
Status	NotConfigured
L	



A custom component can be made to drive a remote I/O module from a BACnet client controller over MS/TP.



Hardware – Dependent Component for the RIBMW24B-44 MS/TP I/O Module

MW24B	•
CControls RIB::MW2	4B
Devinstance	-1
Inp1Use	NotUsed
Inp2Use	NotUsed
Inp3Use	NotUsed
Inp4Use	NotUsed
Out1Use	NotUsed
Out1Priority	10
Out2Use	NotUsed
Out2Priority	10
Out3Use	NotUsed
Out3Priority	10
Out4Use	NotUsed
Out4Priority	10
Inp1	false
Inp2	false
Inp3	false
Inp4	false
Out1	false
Relinquish1	false
Out2	false
Relinquish2	false
Out3	false
Relinquish3	false
Out4	false
Relinquish4	false
Status I	NotConfigured
10	





Although BACnet compliance is not necessary with Sedona, the combination can be advantageous.



Sedona Tool Sedona Application Editor (SAE)





Example HVAC Application Adding an Economizer to an RTU

- With Sedona you have a freely-programmable controller that is capable of implementing several HVAC applications.
 - Multi-stage heating/cooling rooftop unit (RTU) with economizer
 - Air-handing unit (AHU) with analog heating/cooling valves
 - Fan-coil unit (FCU)
 - Make-up air unit (MAU)
 - Energy Recovery Ventilation (ERV) unit
- In this example a 22-point Sedona controller was installed during an RTU retrofit of an economizer requiring the installation of mixed-air and outside air sensors
- By having a BACnet compliant controller, performance of the economizer was easy to monitor with a BACnet client



Example RTU Application Work of an Integrator

ZN_TEMP CControls BASC22 Initialized true ChnType Thm10KT3 OutF 72.15 OutB false OutI 72	In 72.151 DecimalPlaces 1 Y_limT types::ConstFloat Out 230.0	Cmt2 Controls Function::Cmt Comment 0=LocalZoSensor 1=BAS/Network ZoS
ZNT_NET CControls BASC22 IO::VT02 Initialized true ChnType FloatInput Reset false FloatV 0.0 BinaryV false WireSheet InputTo	ZNT_SEL CControls BASC22 Web::WC16WcTypeInputMinVal0.0MaxVal100.0FitVal0.0IntVal0BinValfalse	DiyOn2 N timing::DiyOn Out false DelayTime 6.0 Hold 0
CL_MIN CControls BASC22 Web::WC15 WcType MinVal MaxVal FitVal IntVal BinVal	nput 0.0 00.0 70.0 70 70	LimitSP // func::Limiter Out 68.48 In 68.48 LowLmt 50.0 HighLmt 90.0
ZNL_SET CControls BASC22 Initialized true ChnType Resistance OutF 3650.45 OutB false OutI 3650 Reset false	Round1 OHMS math::Round hvac::Reset Out 3,650.5 In 3,650.459 DecimalPlaces In InMin InMax OutMin OutMax	Z_SETPT Image: Reset 3484.16 Out 68.48 3650.5 In 3484.26 20.0 InMin 0.0 10440.0 InMax 10000.0 0.0 OutMin 65.0 10000.0 OutMax 75.0

Hardware-dependent, hardwareindependent and Tridium-release components were assembled onto wire sheets and interconnected to create the logic for setpoint, mode, heating and cooling, as well as economizer control. A BACnet client provided an occupancy schedule. By adding an economizer, demand control ventilation was obtained.



"H" Diagram of Typical Rooftop Unit w/Economizer



Sedona provides the control while a BACnet client provides the supervision and graphics





Rooftop unit (RTU) with two-stages of heating and cooling plus economizer was upgraded to Sedona when the economizer was installed



- The graphical experience of selecting components, configuring parameters, and linking components to create applications is easy to do and to explain to others
- The technology is open source and supported by several companies so the opportunity exists to share experiences
- A community exists of users who create applications and developers who make components and virtual machines
- The technology is portable to other platforms and will run on a small micro-controller or a powerful computer
- The opportunity exists to share in the exchange of custom components and kits within the community
- Program debugging is fast because the affect of any change is seen instantly

For those familiar with Tridium's Niagara Framework, learning Sedona Framework will require minimal effort.



- The best way to learn Sedona is to try it by downloading SAE and connecting to the SVM-PC that will run on your computer and then create a program
- Community member Contemporary Controls has a multi-part video series on its website devoted to SAE
- There is ample help files in SAE that explain the functioning of the components

SAE Part 1: Introduction Video (8:50)

Introduction to the Sedona Application Editor (SAE) which allows graphical and BASremote.

SAE Part 2: Variable Types Video (6:48)

This video introduces users to the different variable types in the Sedona A

SAE Part 3: Logic Kit Video (9:07)

This video introduces users to the different components located within the components.

SAE Part 4: Math Kit Video (9:11)

This video introduces users to the different components located within the components.

SAE Part 5: Timers and Counters Video (13:28)

This video introduces users to the different timers and counters available t time-critical routine can be implemented.

SAE Part 6: HVAC Kit Video (13:24)

This video introduces users to the different components located within the as example applications created using the components.

SAE Part 7: Introduction to the Kit Manager Video (9:37)

This video introduces users to the Kit Manager and details how to install ar

Thank You



THE WORLD'S LARGEST HVACR MARKETPLACE

JAN 30-FEB 1 LAS VEGAS 2017