Sedona Framework – Best Opportunity for Open Control

Introduction

Zachary Netsov,

Product Specialist, Sedona Alliance

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

Four Traits of an Open Controller

- 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

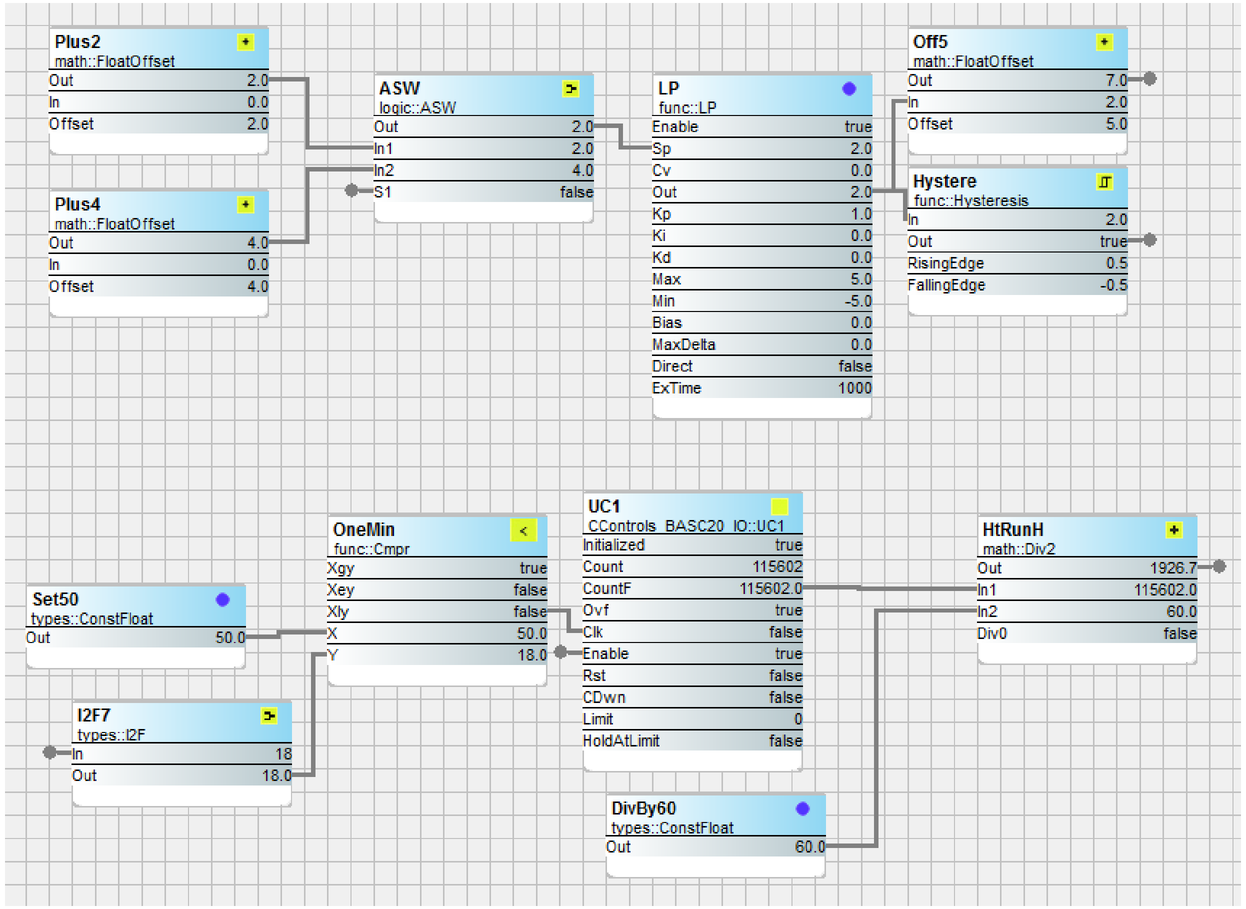
Open Protocol for Network Communications – BACnet

- 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 non-programmers 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.

Open-Source Sedona Framework

- 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 Framework^{tm} 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 Framework^{tm}

Built on Sedona FRAMEWORK^{tm}

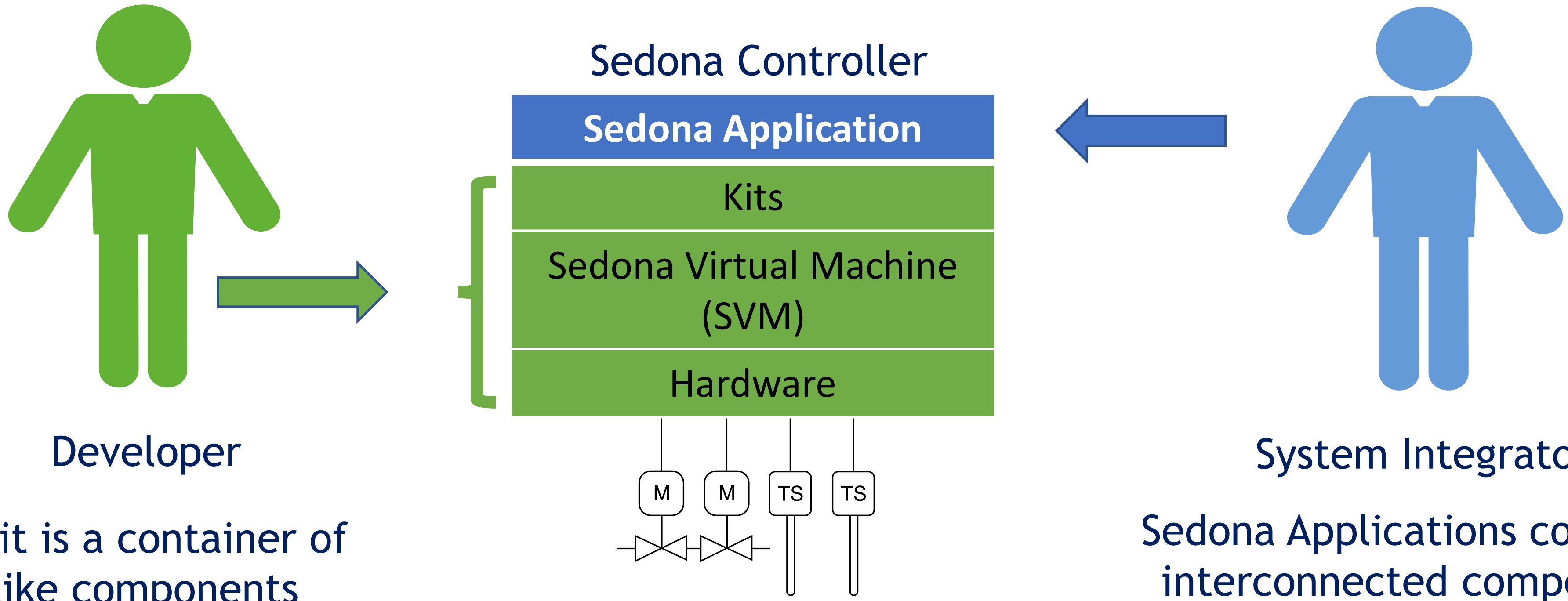
Programming Tool Available without Restriction – Sedona Applications Editor

- 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

Fosters a Community of Developers and Integrators

- 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

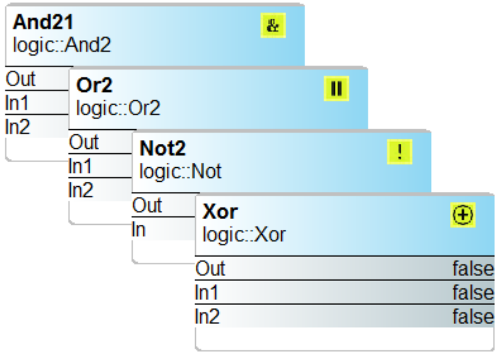
How are Sedona HVAC applications produced?



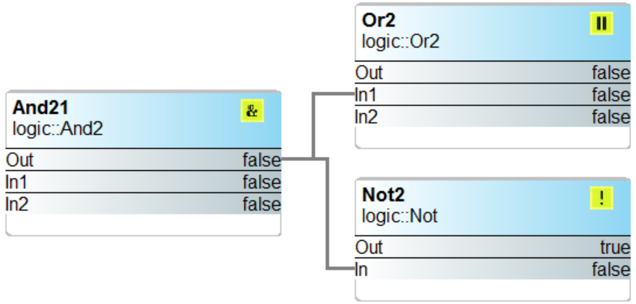
Developer

System Integrator

A Kit is a container of like components



Sedona Applications consist of interconnected components

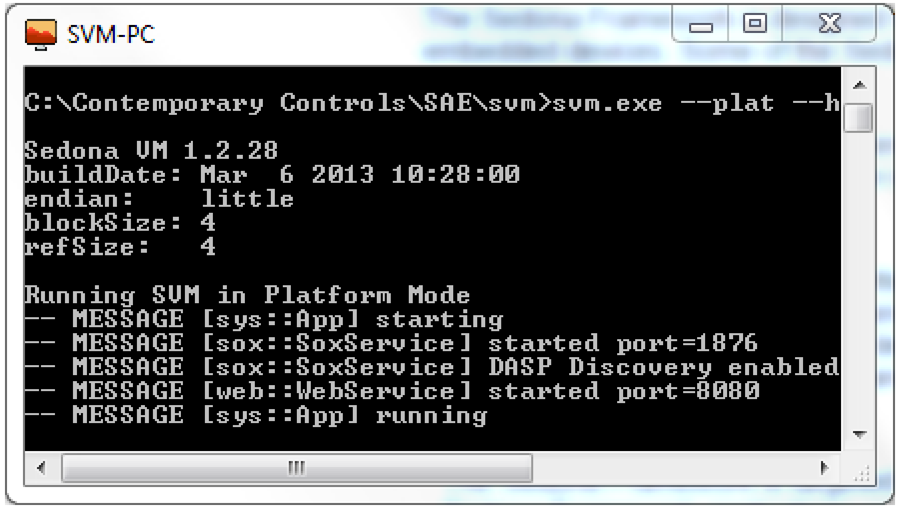


What is the Role of the Developer?

- 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

What is a Sedona Virtual Machine?

- 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

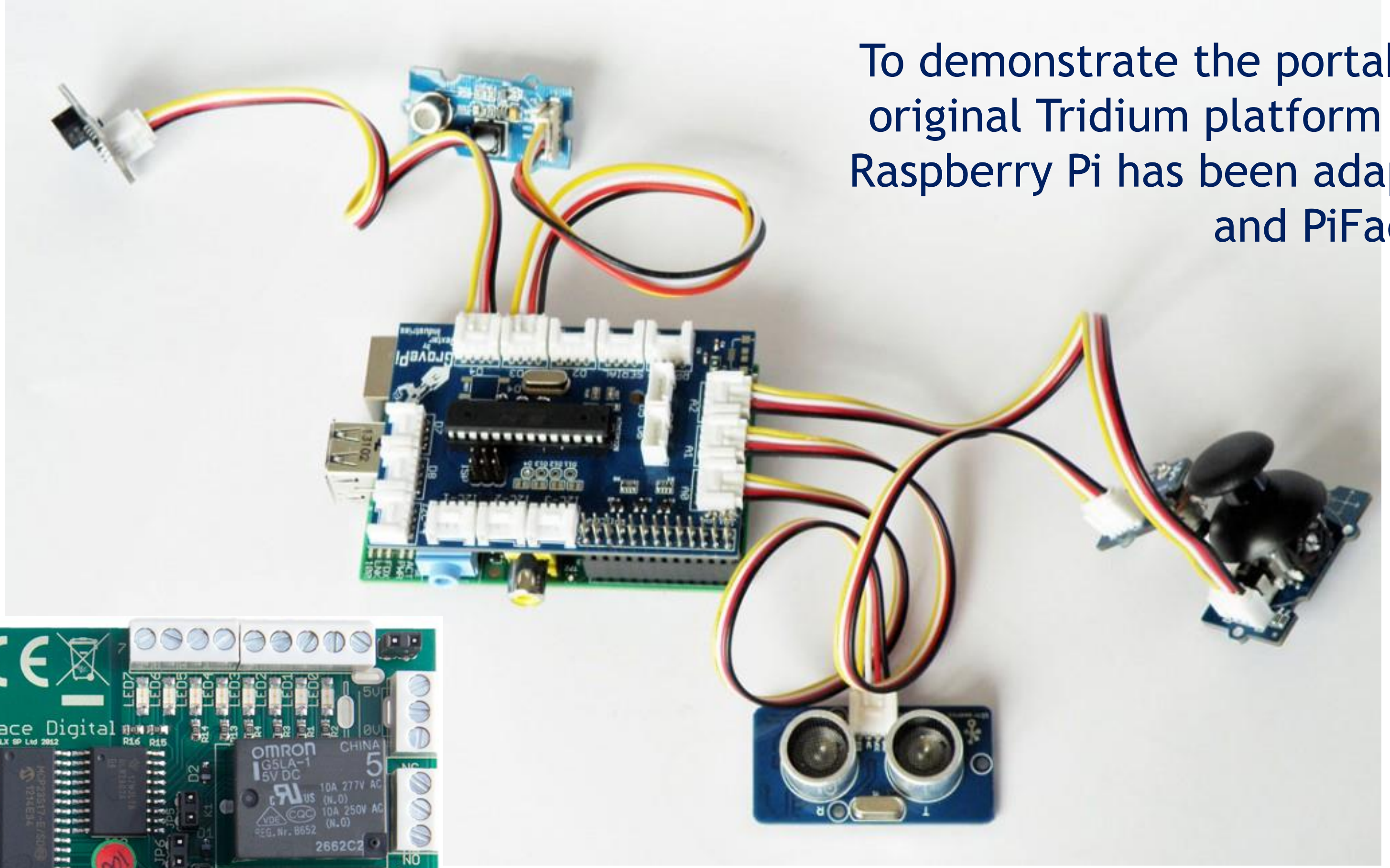


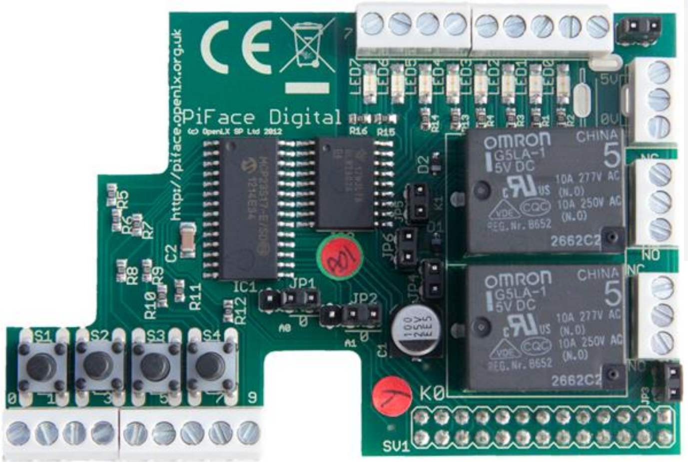
This SVM runs on a Windows PC

AHR EXPO

SVMs for Raspberry Pi Extensions

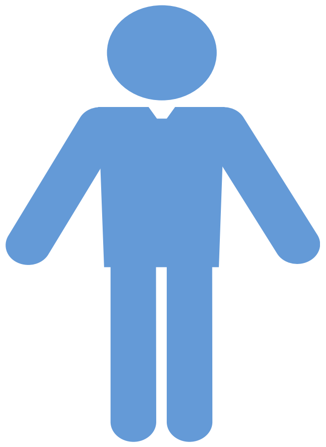
To demonstrate the portability of Sedona, the original Tridium platform implementation for Raspberry Pi has been adapted to both GrovePi and PiFace.





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



Sedona Application

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

Tridium-Release Kits

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

DateTim	
datetimeStd::DateTimeServiceStd	
Nanos	538011125000000000
Hour	23
Minute	32
Second	5
Year	2017
Month	1
Day	17
DayOfWeek	2
UtcOffset	0
OsUtcOffset	false
Tz	

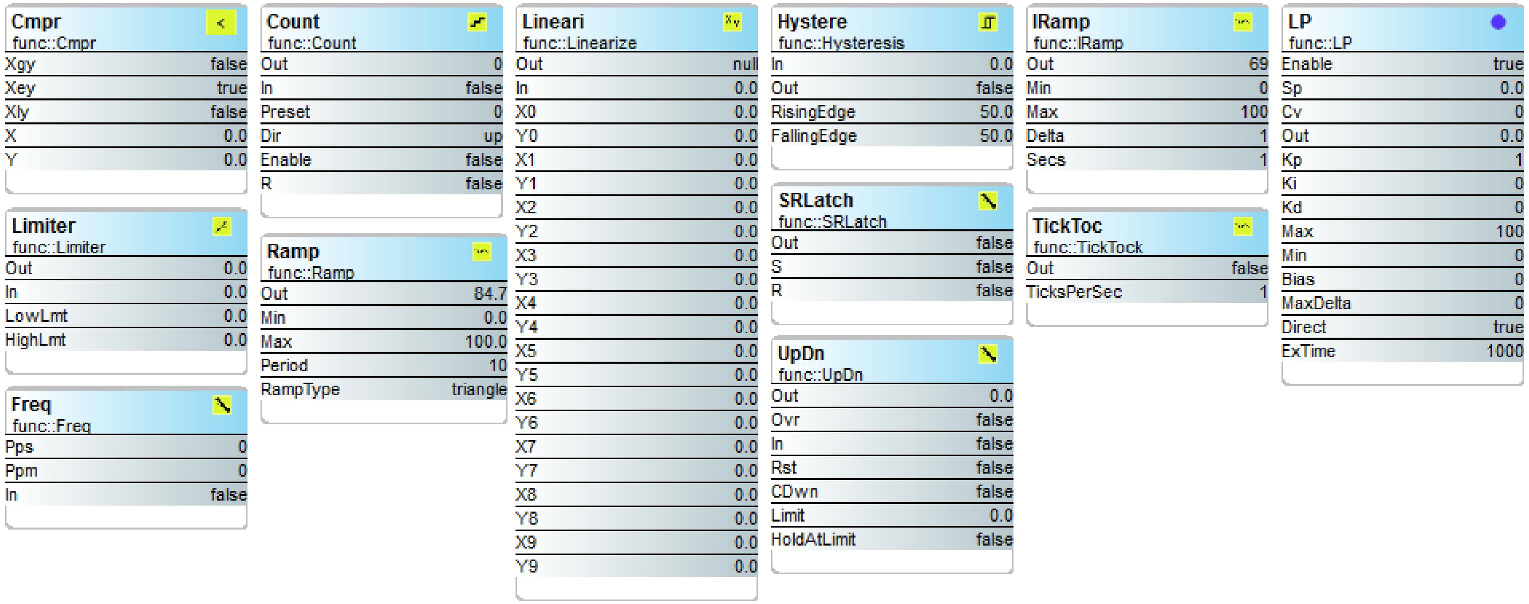
DailySc	
basicSchedule::DailyScheduleBool	
Start1	0:0
Dur1	0:0
Start2	0:0
Dur2	0:0
Val1	false
Val2	false
DefVal	false
Out	false

DailyS1	
basicSchedule::DailyScheduleFloat	
Start1	0:0
Dur1	0:0
Start2	0:0
Dur2	0:0
Val1	0.0
Val2	0.0
DefVal	0.0
Out	0.0

Tridium Function Kit – func

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
Hysteresis	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



Tridium HVAC Kit – hvac

The HVAC Group
operations that facilitate control

LSeq	Linear Sequencer — bar graph representation of input value
ReheatSeq	Reheat sequence — linear sequence up to four outputs
Reset	Reset — output scales an input range between two limits
Tstat	Thermostat — on/off temperature controller

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

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

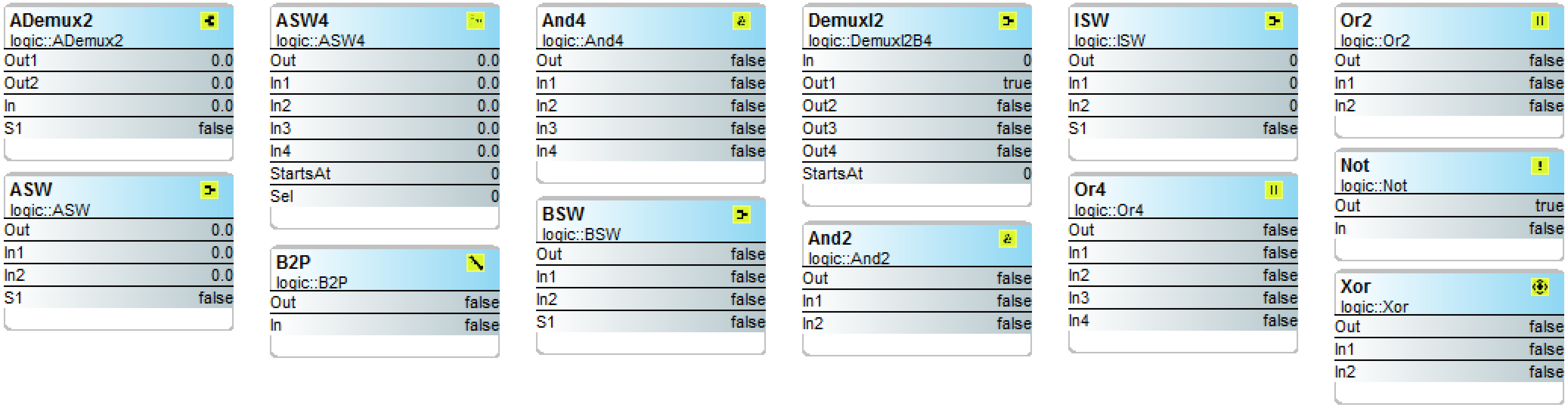
In	0.0
InMin	0.0
InMax	4095.0
OutMin	0.0
OutMax	100.0

IsHeating	false
Sp	0.0
Cv	0.0
Out	false
Raise	false
Lower	false

Tridium Logic Kit – logic

The Logic Group
logical operations using Boolean variables

ADemux2	Analog Demux — Single-input, two-output analog de-multiplexer
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
DemuxI2B4	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

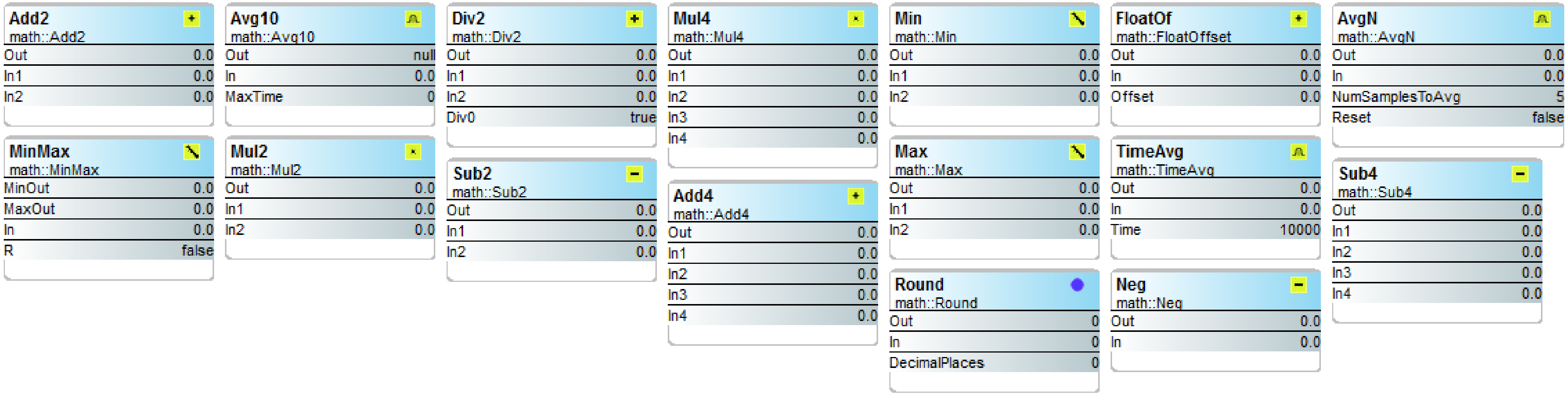


Tridium Math Kit – math

The Math Group

math-based components

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
Div2	Divide two — results in the division of two float variables
FloatOffset	Float offset — float shifted by a fixed amount
Max	Maximum selector — selects the greater of two inputs
Min	Minimum selector — selects the lesser of two inputs
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



Tridium Priority Kit – pricomp

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

PrioritizedBool	Prioritized Boolean output — highest of sixteen outputs
PrioritizedFloat	Prioritized float output — highest of sixteen outputs
PrioritizedInt	Prioritized integer output — highest of sixteen outputs

Priorit
pricomp::PrioritizedBool

SourceLevel	fallback
OverrideExpTime	0
In1	null
In2	null
In3	null
In4	null
In5	null
In6	null
In7	null
In8	null
In9	null
In10	null
In11	null
In12	null
In13	null
In14	null
In15	null
In16	null
Fallback	null
Out	null
MinActiveTime	0
MinInactiveTime	0

Priori1
pricomp::PrioritizedFloat

SourceLevel	fallback
OverrideExpTime	0
In1	null
In2	null
In3	null
In4	null
In5	null
In6	null
In7	null
In8	null
In9	null
In10	null
In11	null
In12	null
In13	null
In14	null
In15	null
In16	null
Fallback	null
Out	null

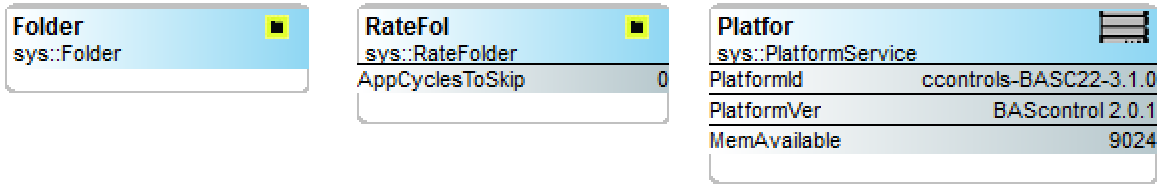
Priori2
pricomp::PrioritizedInt

SourceLevel	fallback
OverrideExpTime	0
In1	min
In2	min
In3	min
In4	min
In5	min
In6	min
In7	min
In8	min
In9	min
In10	min
In11	min
In12	min
In13	min
In14	min
In15	min
In16	min
Fallback	min
Out	min

Tridium System Kit – sys

The System Group
platform and folder components

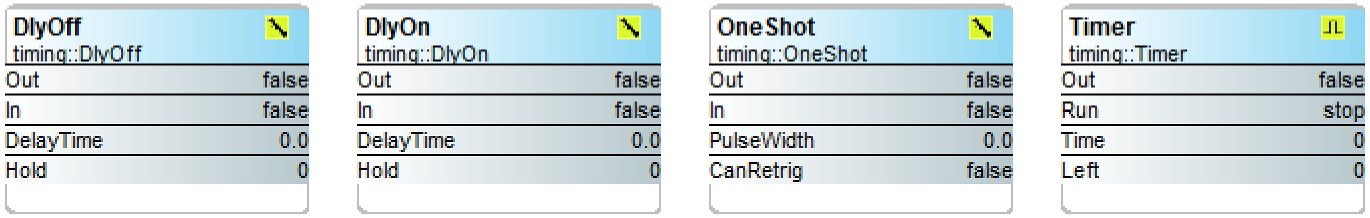
PlatformService	Platform service — indicates platform and available memory
Folder	Folder — when accessed opens to another wire sheet
RateFolder	Rate Folder — a folder that can be used for background tasks



Tridium Timing Kit – timing

The Timing Group
time-based components

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

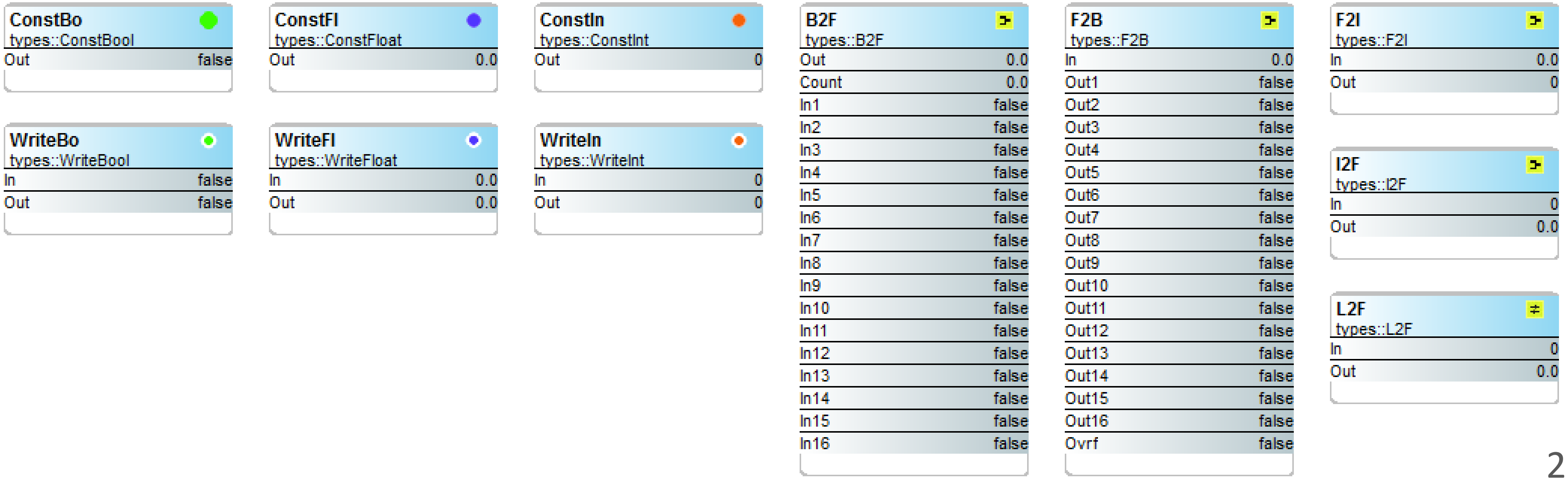


Tridium Types Kit – types

The Types Group

variable types and conversion between types

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



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

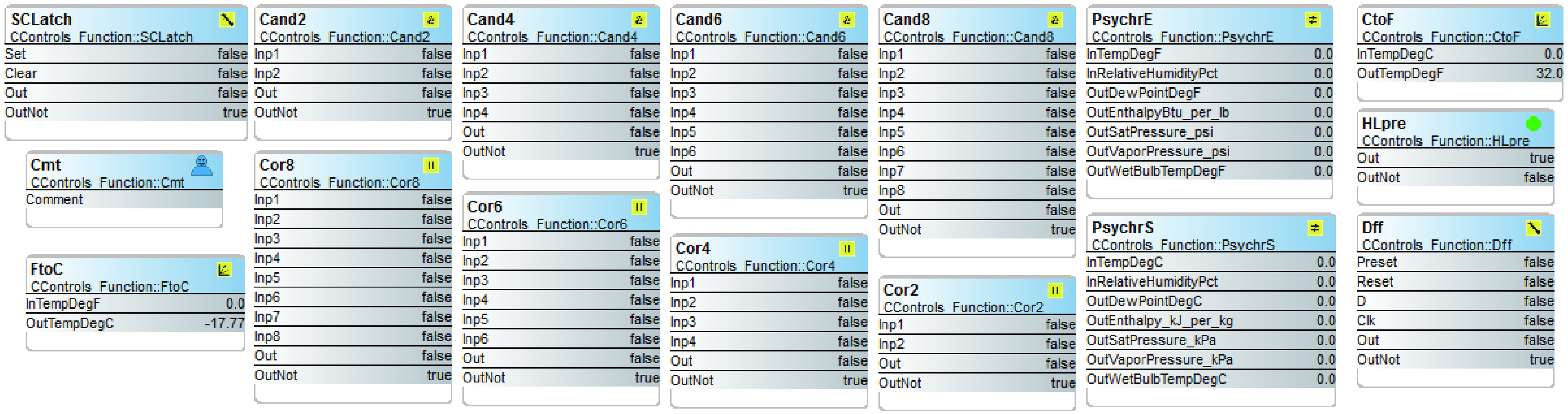
There are numerous custom kits and components from the Sedona community

Custom Hardware – Independent Kit Function – CControls_Function

Custom Functions
collection of helpful components

Cand2	Two-input Boolean product — two-input AND/NAND gate with complementary outputs
Cand4	Four-input Boolean product — four-input AND/NAND gate with complementary outputs
Cand6	Six-input Boolean product — six-input AND/NAND gate with complementary outputs
Cand8	Eight-input Boolean product — eight-input AND/NAND gate with complementary outputs
Cmt	Comment — comment field up to 64 characters
Cor2	Two-input Boolean sum — two-input OR/NOR gate with complementary outputs
Cor4	Four-input Boolean sum — four-input OR/NOR gate with complementary outputs
Cor6	Six-input Boolean sum — six-input OR/NOR gate with complementary outputs
Cor8	Eight-input Boolean sum — eight-input OR/NOR gate with complementary outputs
CtoF	°C to °F — Celsius to Fahrenheit temperature conversion
Dff	"D" Flip-Flop — D-style edge-triggered single-bit storage
FtoC	°F to °C — Fahrenheit to Celsius temperature conversion
HLpre	High-Low Preset — defined logical true and false states
PsychrE	Psychrometric Calculator — English units
PsychrS	Psychrometric Calculator — SI units
SCLatch	Set/Clear Latch — level-triggered single-bit data storage

This custom kit was developed by Contemporary Controls



Custom Hardware – Independent Kit HVAC Kit – CControls_HVAC

Custom HVAC
advanced HVAC components

AnaHiLo	Analog High/Low — analog variable out-of-range limit or detection
AntiSCY	Anti-Short Cycle — minimum run time and minimum start time limiter
BTUh	BTU/Hour Calculator — calculates energy usage based on temperature difference and flow
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
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 CControls_HVAC::AnaHiLo	
LimitDelay	1
HiLimit	10
LoLimit	-10
Differential	0.1
HoldAtLimit	false
LimitOutEnable	false
In	0
Out	0
OverLimit	false
UnderLimit	false

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

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

NumDamp CControls_HVAC::NumDamp	
UpdateInterval	5
RiseIncrement	0.5
FallDecrement	0.5
RiseDampInhibit	false
FallDampInhibit	false
In	0.0
Out	0.0

AntiSCY CControls_HVAC::AntiSCY	
MinRunTime	1
MinOffTime	1
In	false
Out	false
Reset	false

BTUh CControls_HVAC::BTUh	
ExeDelay	0
OffCal	0.0
InGPM	0.0
InTemp	0.0
OutTemp	0.0
Out	0.0
TonOutR	0.0
TonOutC	0.0

TockTic CControls_HVAC::TockTic	
Period	1.0
Enable	true
Out	true

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

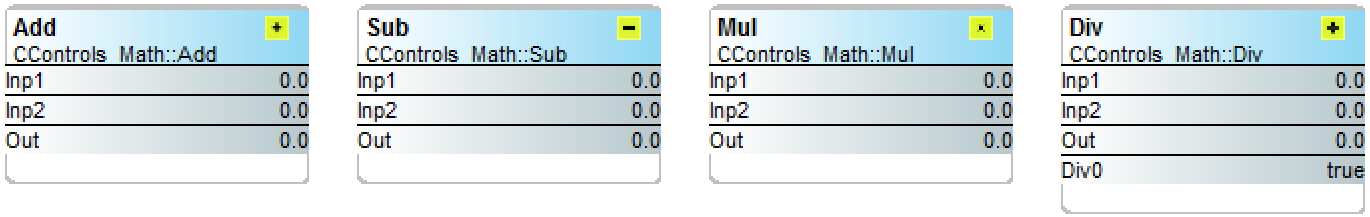
OATrueB CControls_HVAC::OATrueB	
ExeDelay	1
OffCal	0.0
OutsideAT	0.0
ReturnAT	0.0
MixedAT	0.0
Output	0.0
Fault	true

This custom kit was developed by Contemporary Controls

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



This custom kit was developed by Contemporary Controls

Hardware – Dependent Components BAScontrol20

AO1-AO4	Analog output — analog output voltage point
BI1-BI4	Binary input — binary input point
BO1-BO4	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	CControls_BASC20_IO::UI6
ChnType	Input10V
OutF	0.00
OutB	false
OutI	0

AO4	CControls_BASC20_IO::AO4
InpF	0.0
Enable	true

UC4	CControls_BASC20_IO::UC4
Initialized	true
Count	0
CountF	0.0
Ovf	true
Clk	false
Enable	true
Rst	false
CDwn	false
Limit	0
HoldAtLimit	false

VT8	CControls_BASC20_IO::VT08
Initialized	true
ChnType	FloatInput
Reset	false
FloatV	0.0
BinaryV	false
WireSheet	InputTo

BI4	CControls_BASC20_IO::BI4
OutB	true

WC01	CControls_BASC20_Web::WC01
WcType	Input
MinVal	0.0
MaxVal	100.0
FltVal	0.0
IntVal	0
BinVal	false

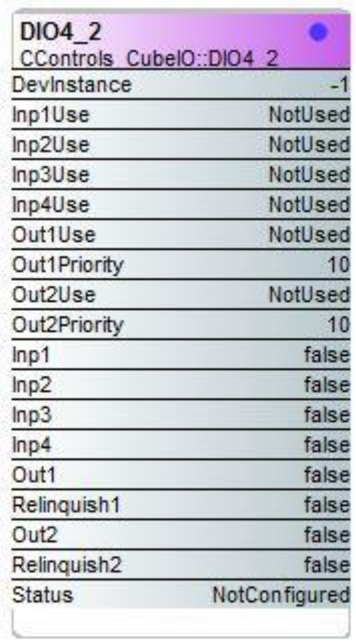
BO2	CControls_BASC20_IO::BO2
InpB	false
Enable	true

ScanTim	CControls_BASC20_IO::ScanTim
SampleSize	10
TimeMs	44
MinimumMs	43
MaximumMs	49
AverageMs	43
Reset	false

plat	CControls_BASC20_Platform::BASC20PlatformService
PlatformId	ccontrols-BASC20-3.1.0
PlatformVer	BAScontrol 2.0.1
MemAvailable	19560

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

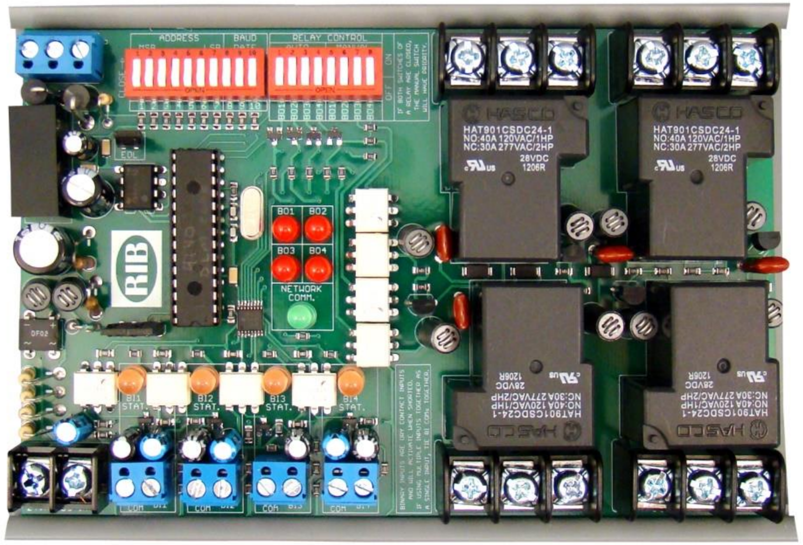
Hardware – Dependent Component for the Metz DIO 4/2 MS/TP I/O Module



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::MW24B	
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	NotConfigured

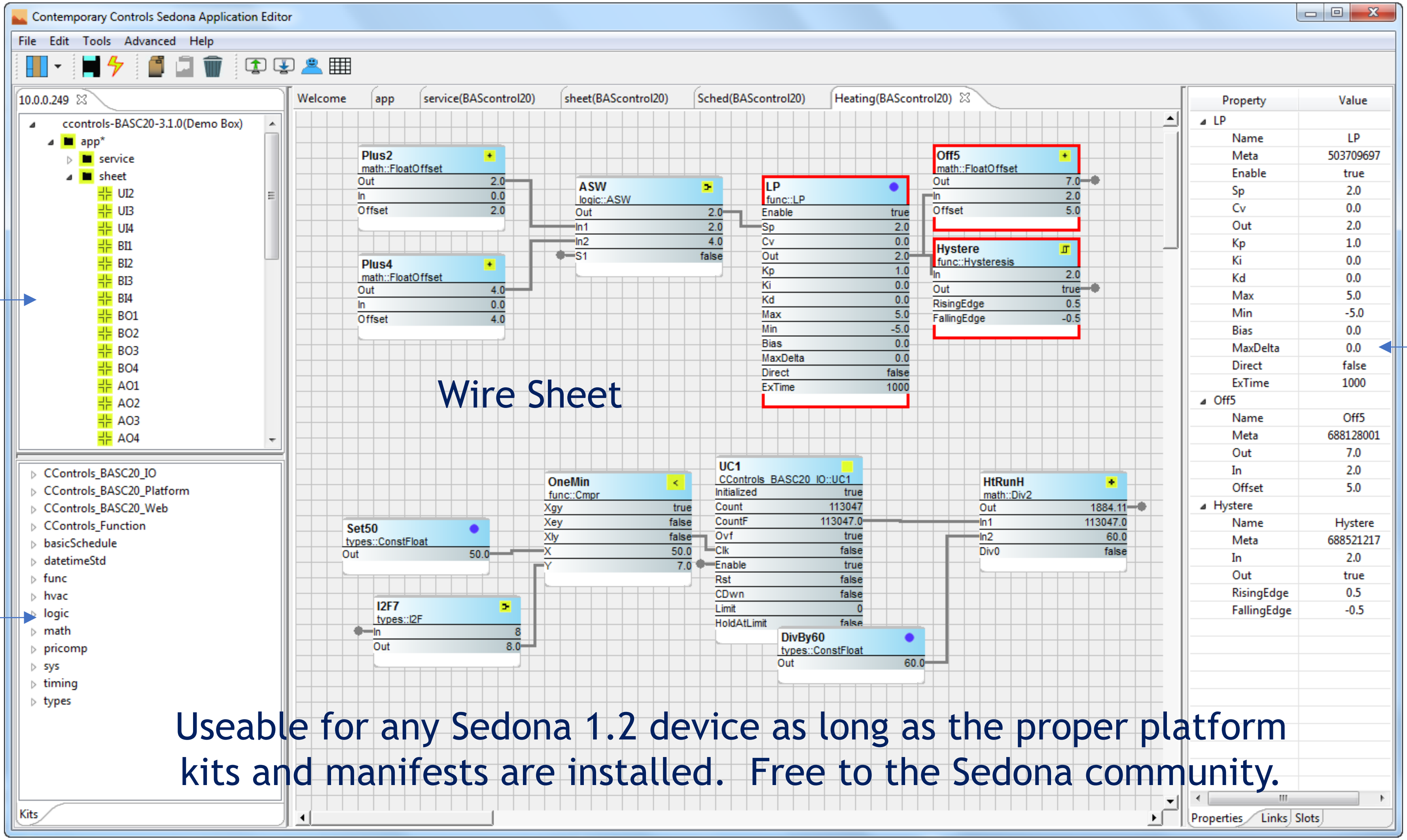


Although BACnet compliance is not necessary with Sedona, the combination can be advantageous.

Sedona Tool
Sedona Application Editor (SAE)

Navigation Pane

Kits Pane



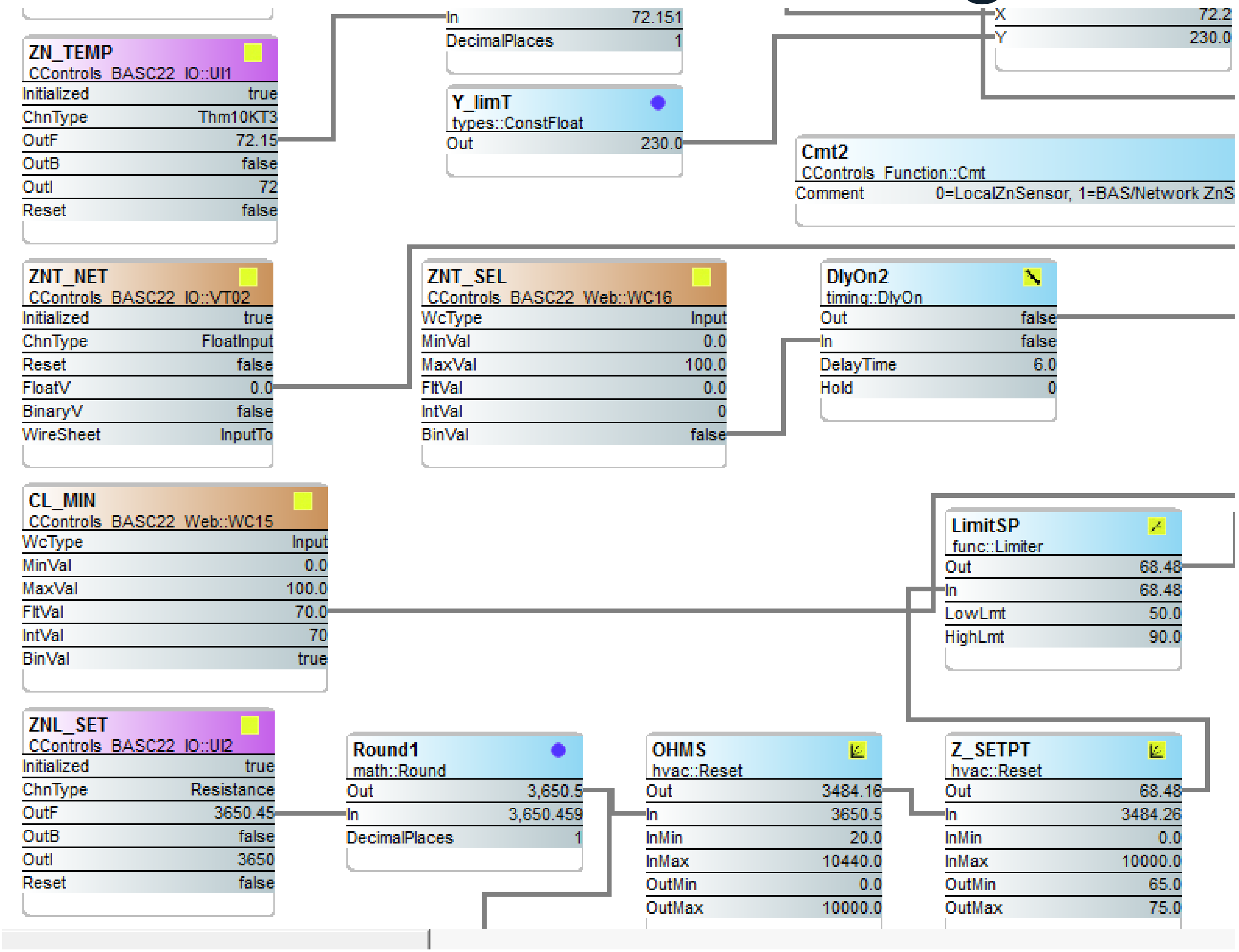
Properties Pane

Useable for any Sedona 1.2 device as long as the proper platform kits and manifests are installed. Free to the Sedona community.

Example HVAC Application
Adding an Economizer to an RTU

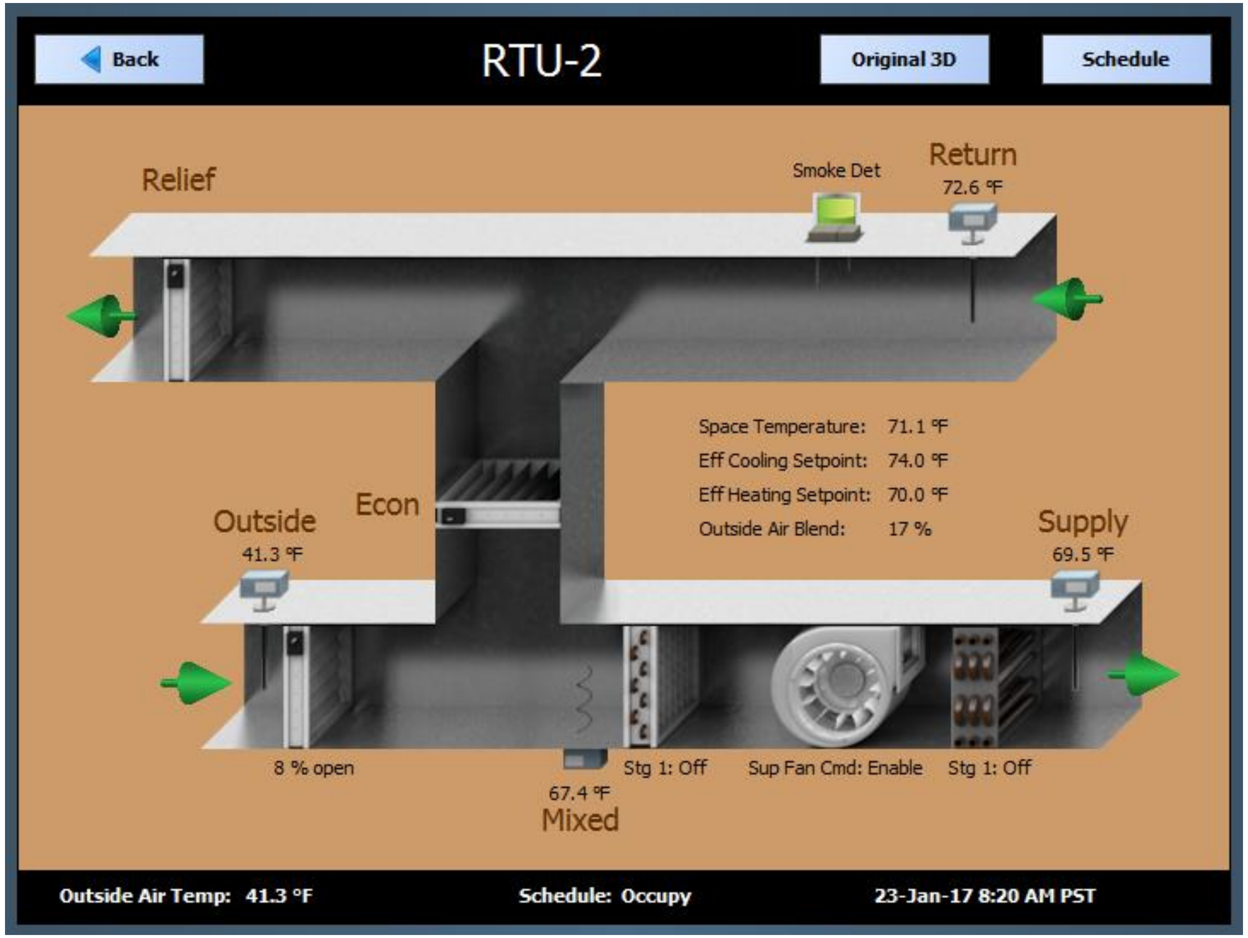
- With Sedona you have a freely-programmable controller that is capable of implementing several HVAC applications.
  - Multi-stage heating/cooling rooftop unit (RTU) with economizer
  - Air-handing unit (AHU) with analog heating/cooling valves
  - Fan-coil unit (FCU)
  - Make-up air unit (MAU)
  - Energy 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



Hardware-dependent, hardware-independent 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

HVAC Application Using Sedona

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

What is There to Like in Sedona?

- 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.

Teach Yourself Sedona

- 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 a

Thank You


AHR
EXPO®
THE WORLD'S LARGEST HVACR MARKETPLACE
JAN 30-FEB 1
LAS VEGAS
2017
CO-SPONSORS
ASHRAE
AHRI