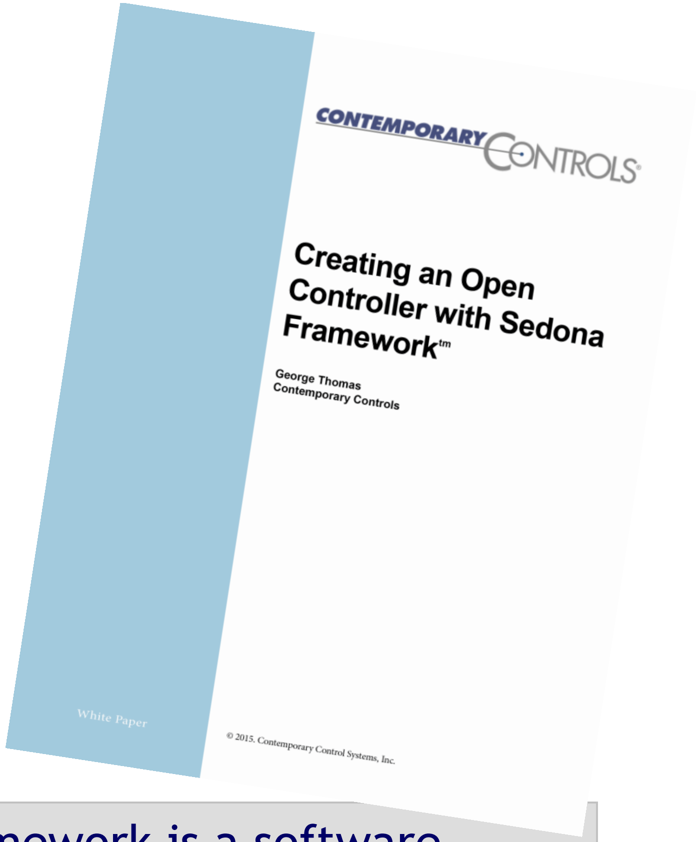
Built on the Sedona Framework™

Using Sedona to Create an Open Controller

The Need for an Open Control Technology

Having just BACnet is not good enough when you are locked out of a job due to a proprietary programming language and tool. What is needed is an open control technology and unrestricted programming tool.


CONTEMPORARY CONTROLS

Creating an Open Controller with Sedona Framework™

George Thomas
Contemporary Controls

White Paper

© 2015. Contemporary Control Systems, Inc.

Developed by Tridium, 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. Contemporary Controls is a Sedona community member and views this technology as the best hope in creating a truly open controller.

Contemporary Controls Defines an Open Controller

- Utilizes an open protocol for network communications
  - *BACnet is an ISO standard with international acceptance*
- Supports 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*
- Provides a programming tool that is available to systems integrators without restriction
  - *Those without access to Niagara Workbench can use Sedona Application Editor from Contemporary Controls*
- 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*

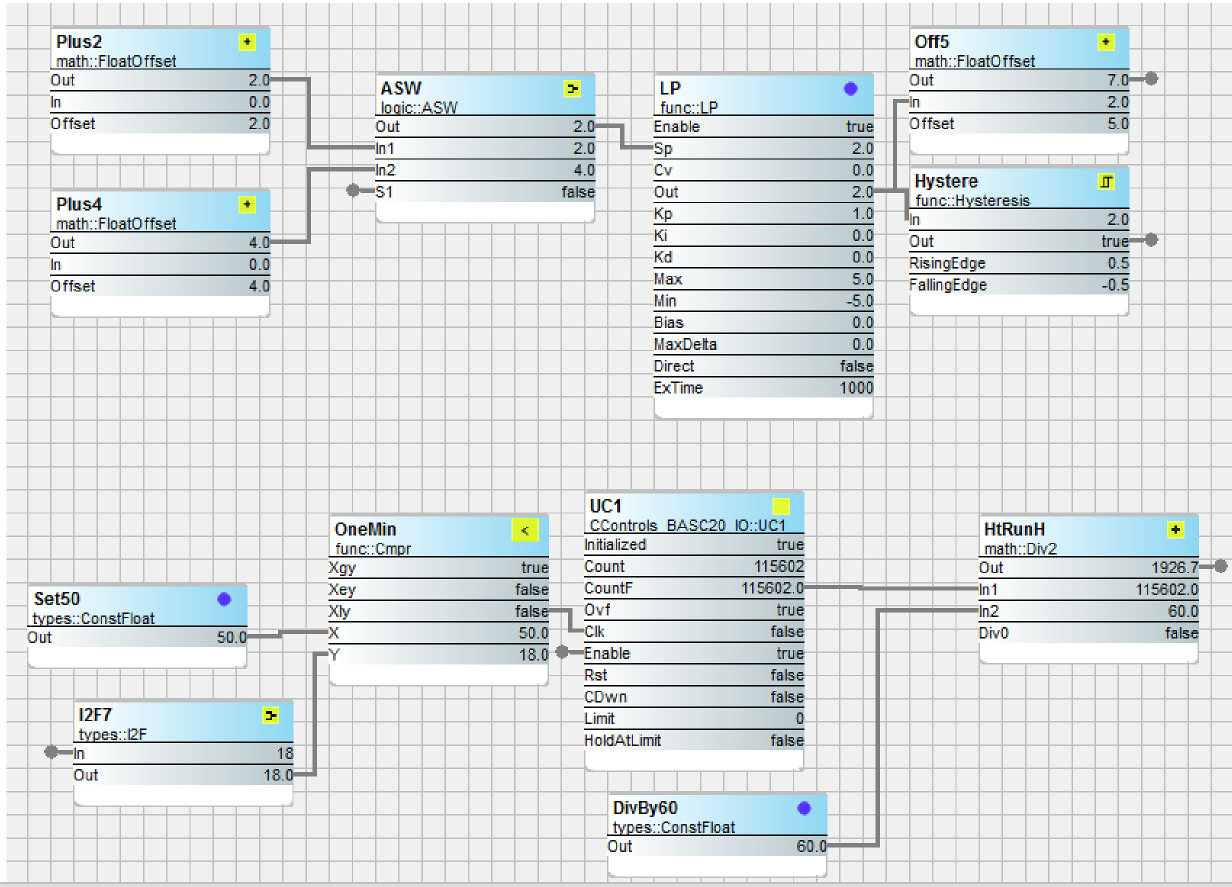
Open Protocol for Network Communications

- BACnet - a communications protocol for **B**uilding **A**utomation and **C**ontrol **Net**works
- 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

- 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.

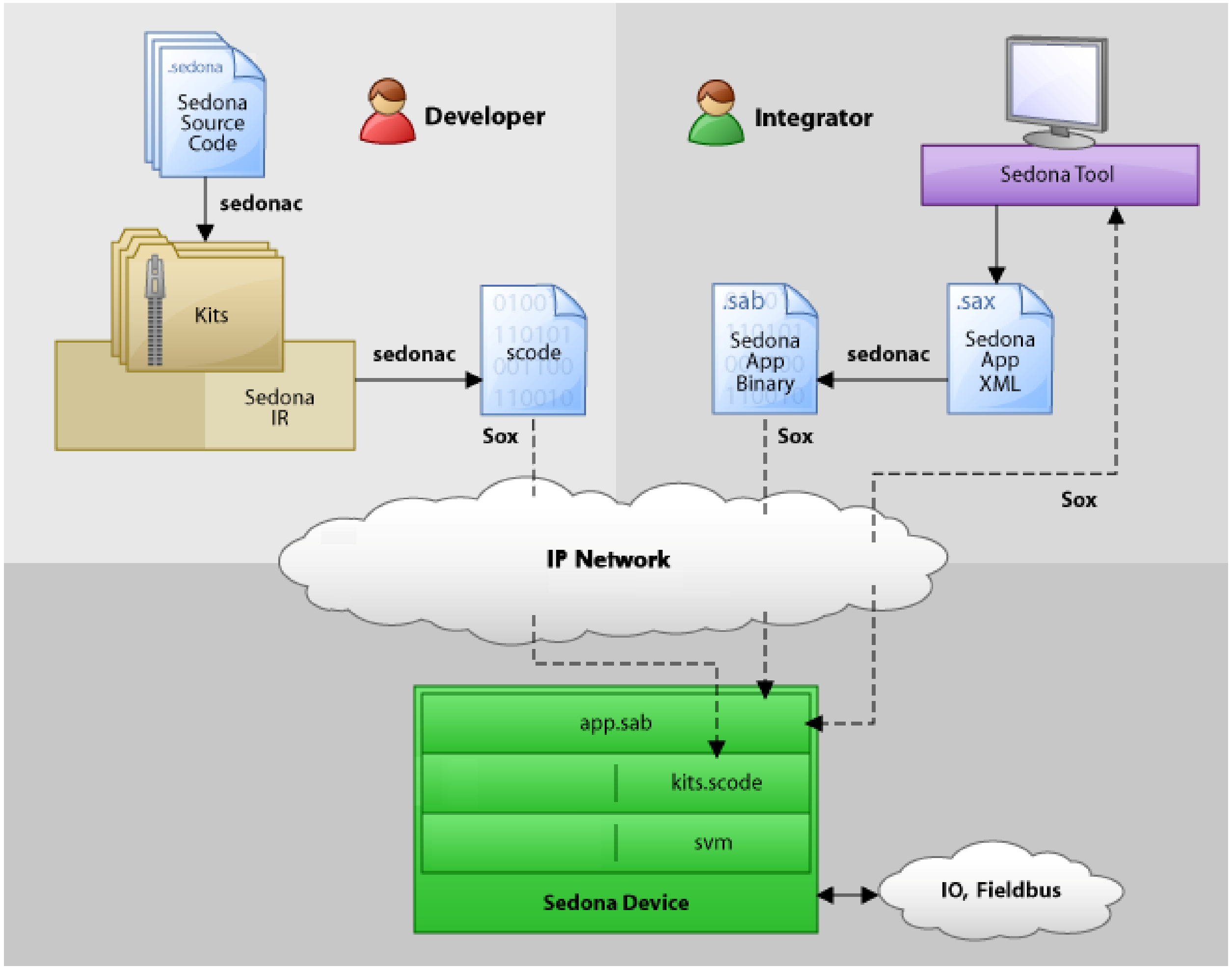
Programming Tool Available without Restriction

- Available via download from the Contemporary Controls website – *Sedona Application Editor (SAE)*
- Includes all the necessary platforms, kits and manifests required for Contemporary Controls' controllers
- Includes a Sedona virtual machine (SVM-PC) that runs on a PC that can be programmed with the SAE for testing
- Can be used with other Sedona devices as long as the proper platforms, kits and manifests are added to the Sedona Data Folder
- Requires Java Runtime Environment 1.7
- 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 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

Sedona Workflow Model

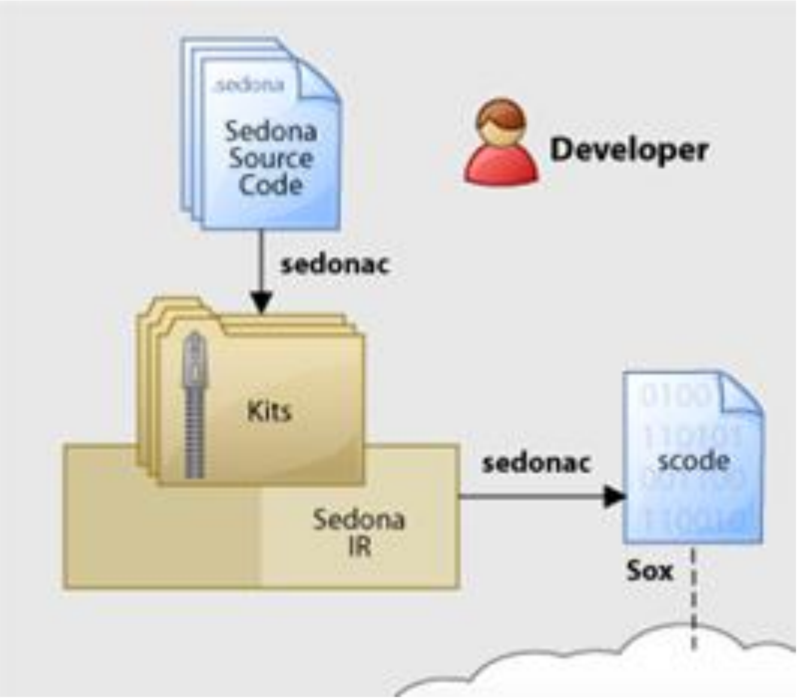


The roles or developer and integrator differ in this model.

Developer’s Role

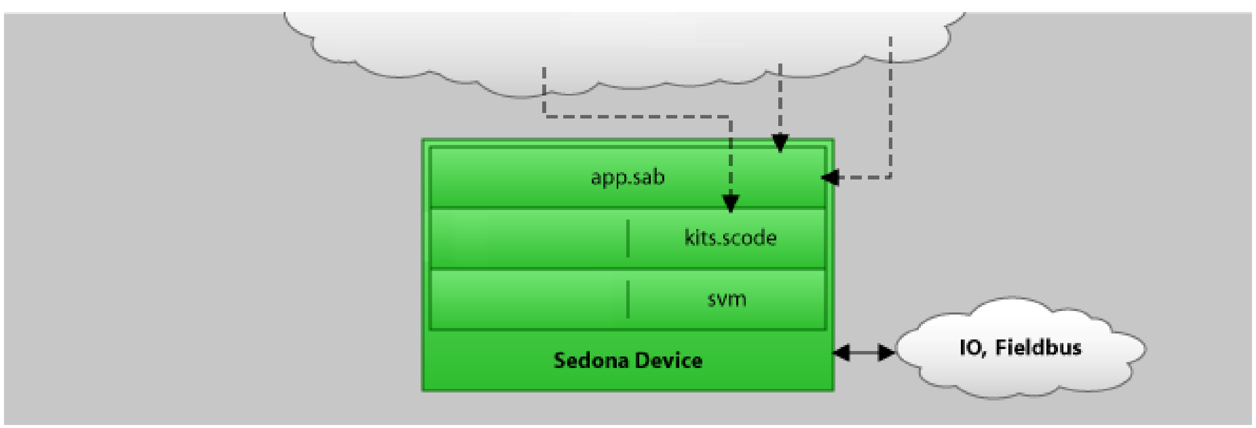
- *Components* are developed using the Sedona language and deployed as *kits*
- All the *kits* are then complied first to an intermediate language for portability and then into an *kits.scode* image suitable for the Sedona device

The Sedonac complier is available from the SedonaDev.org site.



Contemporary Controls as a Developer

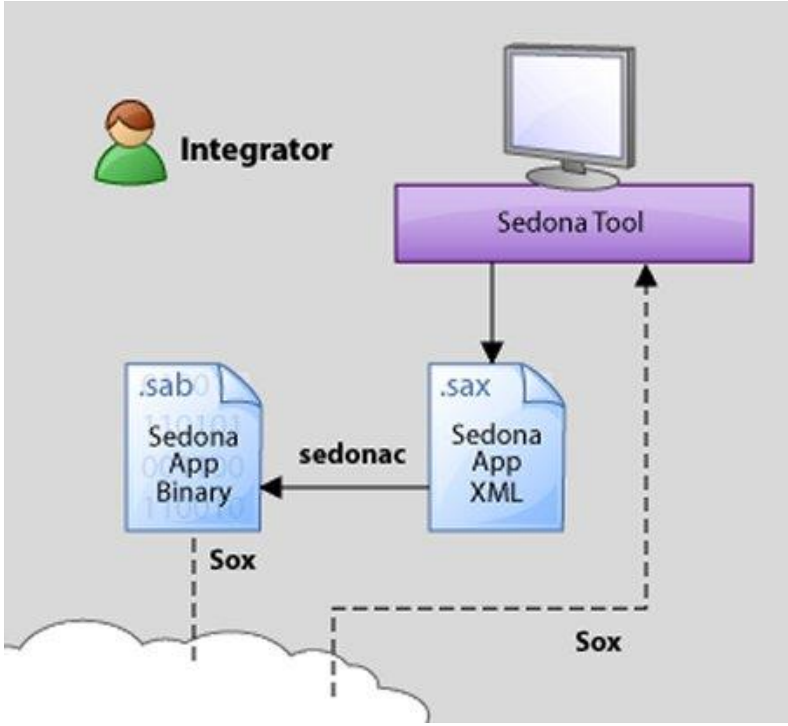
- Uses the Sedona language to develop custom *components* that are unique to the BAScontrol or BAS Remote
- Creates the *Sedona Virtual Machine* (SVM) that resides in the controller



Integrator's Role

- Drags-and-drops components from the various kits onto a wire sheet and configures the components accordingly
- Using *links*, interconnects the components to create an application called an *app.sab* file and tested in real-time
- The application is saved to flash on the Sedona device for auto-execution upon power-up

Either Niagara Workbench or a Sedona tool such as Contemporary Controls' Sedona Application Editor can be used to create Sedona applications.



Some Sedona Definitions That Might Help

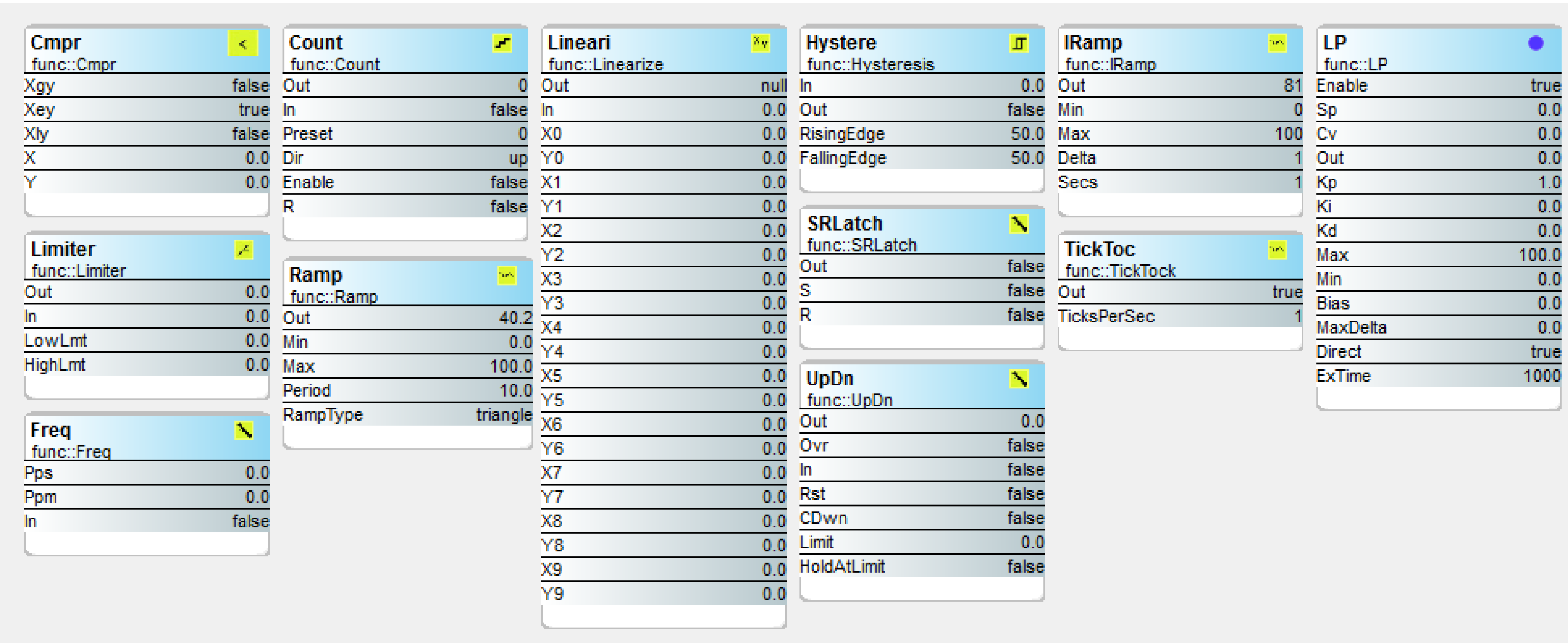
- **Component** – basic building block for creating logic. Components have slots for interconnecting links and proprieties that can be configured.
- **Kit** – a grouping of components by some common trait such as math, logic or IO. A kit file has the executable code for each component in the kit in binary form.
- **Manifest** – a XML file which describes the code within the kit by listing each component along with characteristics such as slots. Needed when drawing components on wire sheets.
- **Platform** – a XML manifest file contains a list of services the Sedona device provides.
- **kits.scode** – a single binary file of all kits in a Sedona device
- **sax** file – a textual representation of the application in XML
- **sab** file – a binary representation of the application complied from the SAX file and executed on the Sedona device.

Hardware Dependent and Independent Kits

- There are three types of kits
- The original Tridium kits built for the 1.2.28 platform had grouped components by function – they do not have a leading developer name in front of the kit names and can run on any Sedona 1.2.28 platform
- All other kits are custom kits requiring the developer to append its name in front of the kit name
- Some kits are hardware dependent (involve the addressing of physical I/O) and require an appended product name to the kit
  - e.g. CControls_BASC22_IO
- Hardware independent custom kits just carry the developer name and some meaningful name for the kit
  - e.g. CControls_Function

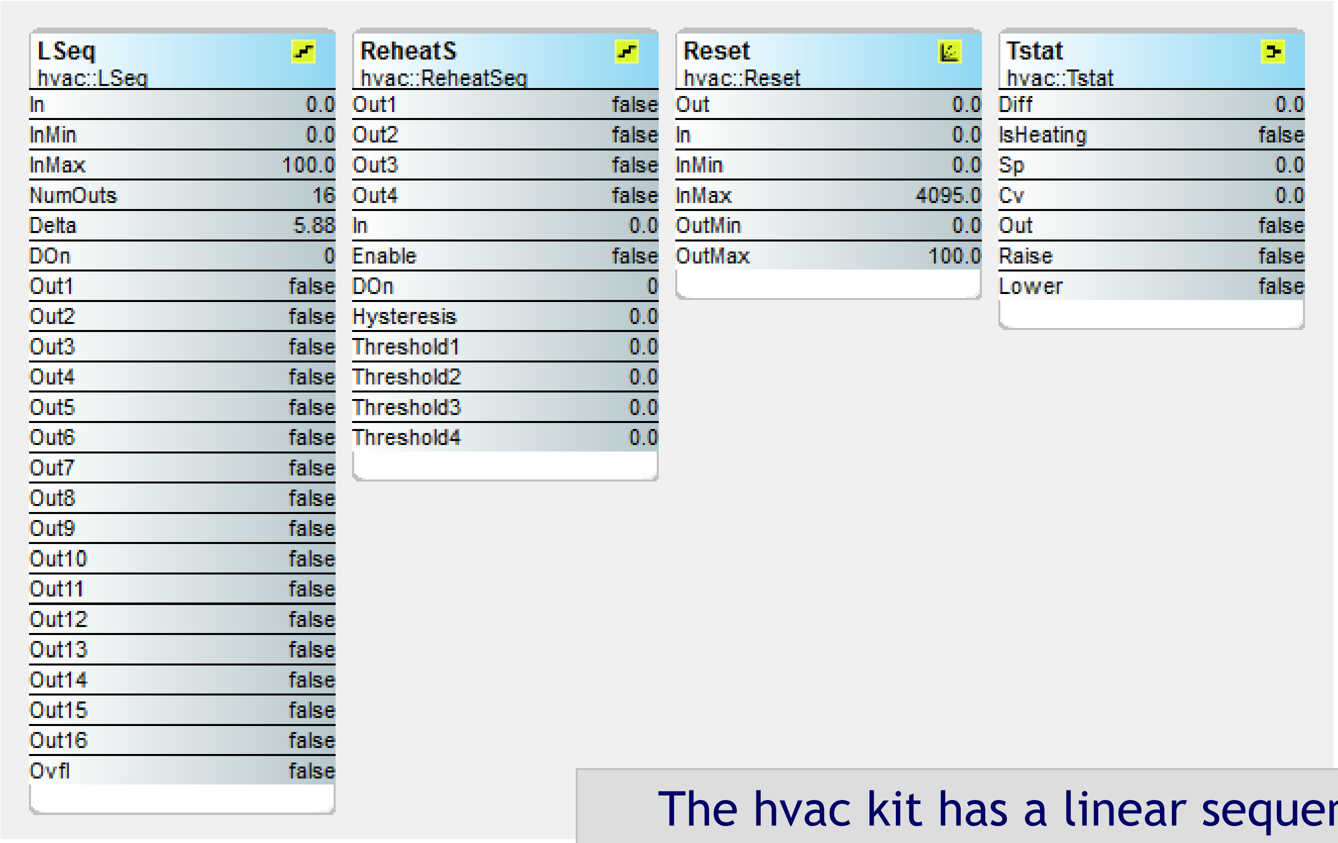
It is encouraged that hardware independent kits be shared.

Function Kit - func



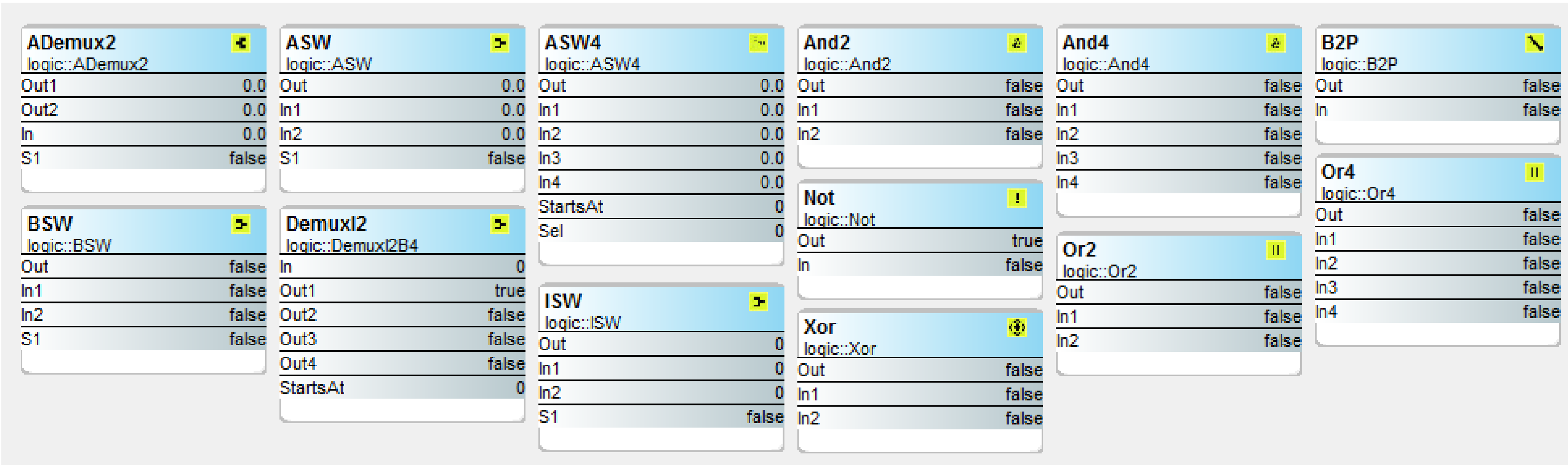
The func kit from Tridium provides a comparator, limiters, counters, ramp generators, a clock, a linearizer, a latch and a loop component that implements proportional, integral and derivative (PID) control.

HVAC Kit – hvac



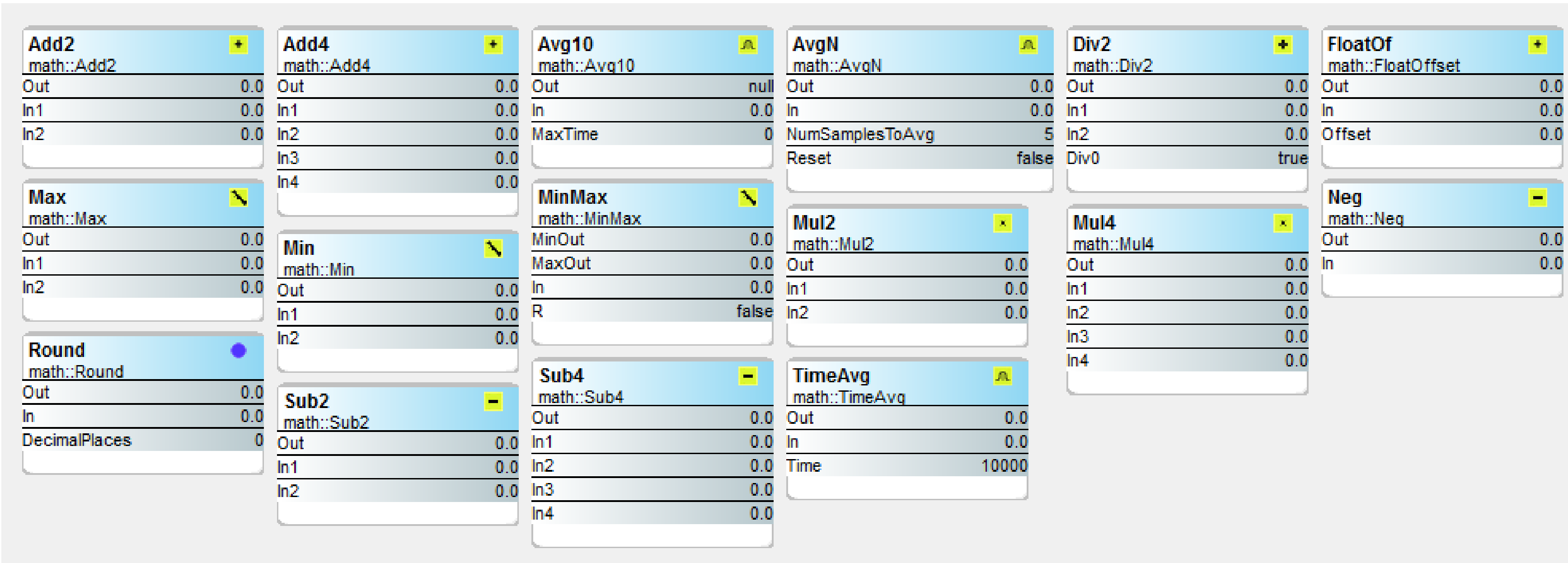
The hvac kit has a linear sequencer, a reheat sequencer, a reset component that can scale inputs, and a thermostat controller.

Logic Kit – logic



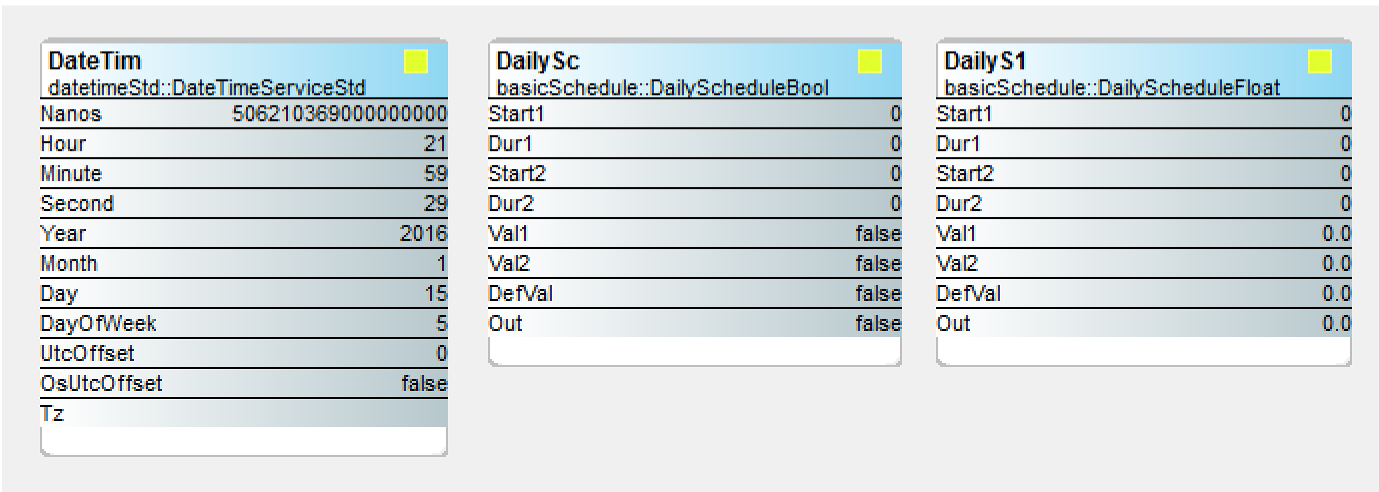
The logic kit includes common Boolean AND, OR, XOR, NOT components, binary and analog switches, de-multiplexers and a binary to pulse converter.

Math Kit – math



Besides standard Add, Subtract, Multiply and Divide functions, the math kit has components to determine the minimum and maximum of a variable, and its average.

Time and Schedule Kits – dateTime, basicSchedule



Both time and date are maintained in order to drive schedules of either binary or analog variables.

Priority Kit – pricomp

Priorit — pricomp::PrioritizedBool

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

Priori1 — pricomp::PrioritizedFloat

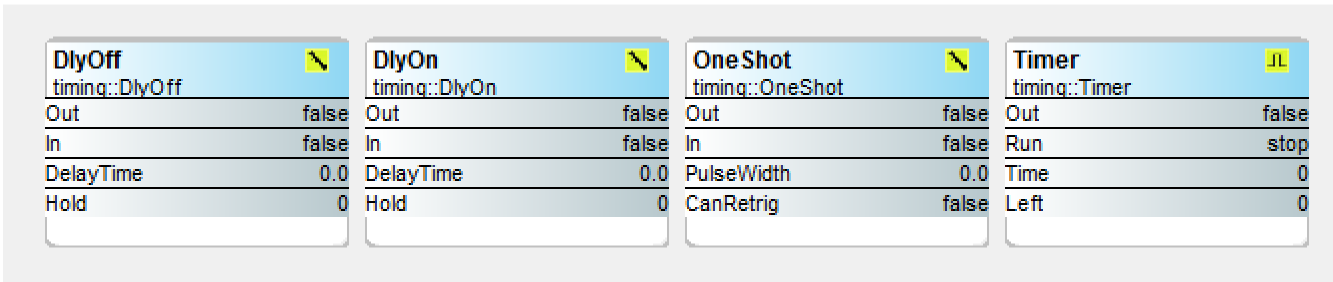
SourceLevel	fallback
OverrideExpTime	0
In1	null
In2	null
In3	null
In4	null
In5	null
In6	null
In7	null
In8	null
In9	null
In10	null
In11	null
In12	null
In13	null
In14	null
In15	null
In16	null
Fallback	null
Out	null

Priori2 — pricomp::PrioritizedInt

SourceLevel	fallback
OverrideExpTime	0
In1	-2147483648
In2	-2147483648
In3	-2147483648
In4	-2147483648
In5	-2147483648
In6	-2147483648
In7	-2147483648
In8	-2147483648
In9	-2147483648
In10	-2147483648
In11	-2147483648
In12	-2147483648
In13	-2147483648
In14	-2147483648
In15	-2147483648
In16	-2147483648
Fallback	-2147483648
Out	-2147483648

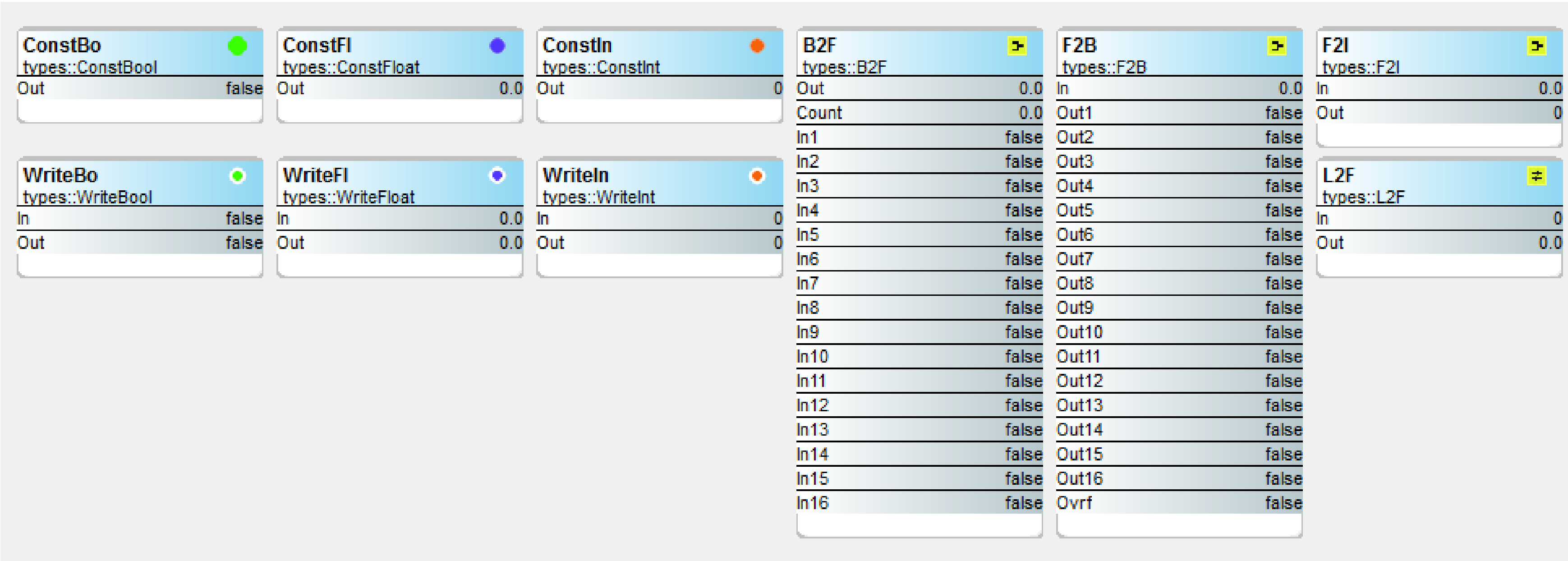
Priority components exist to handle 16 levels of priority for binary, integer and float variables.

Timing Kit – timing



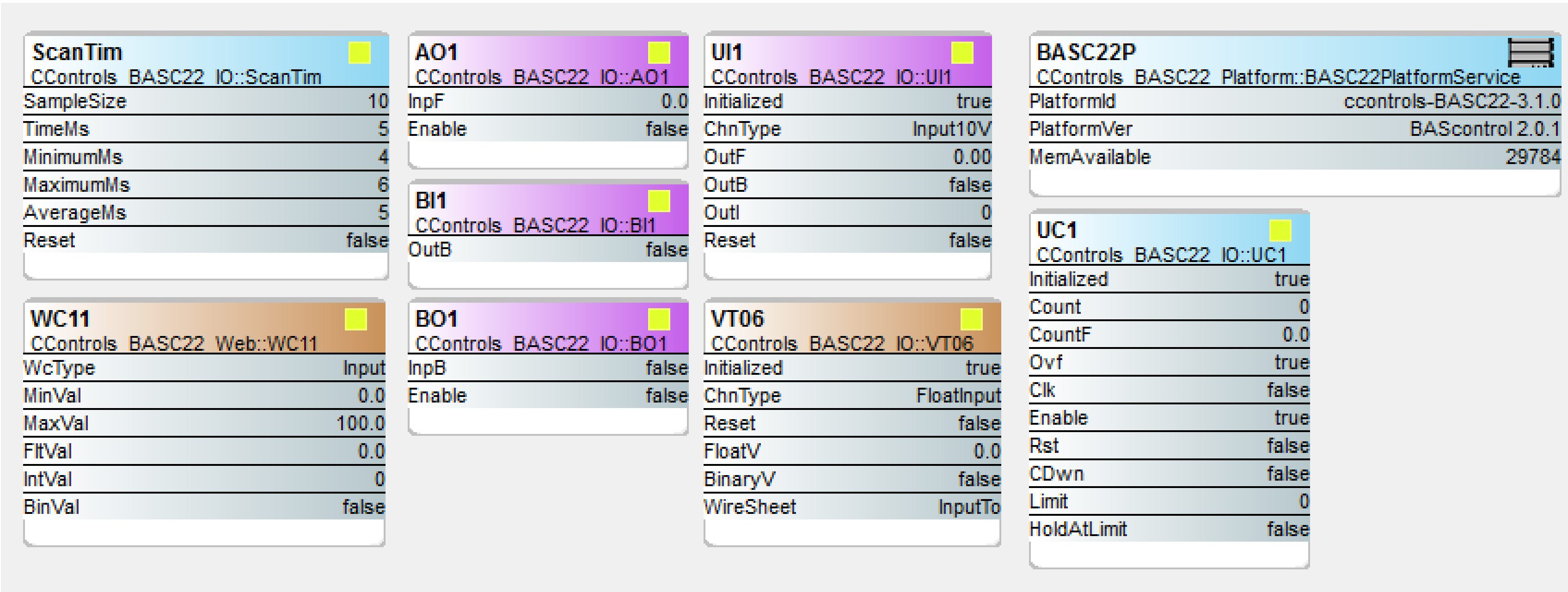
On-delay, off-delay, and interval counters are available in the timing kit along with a settable single-shot.

Types Kit – types



Variable types include Boolean, integer, long (long integer), and float. Components exist to introduce constant values and the ability to convert between variable types.

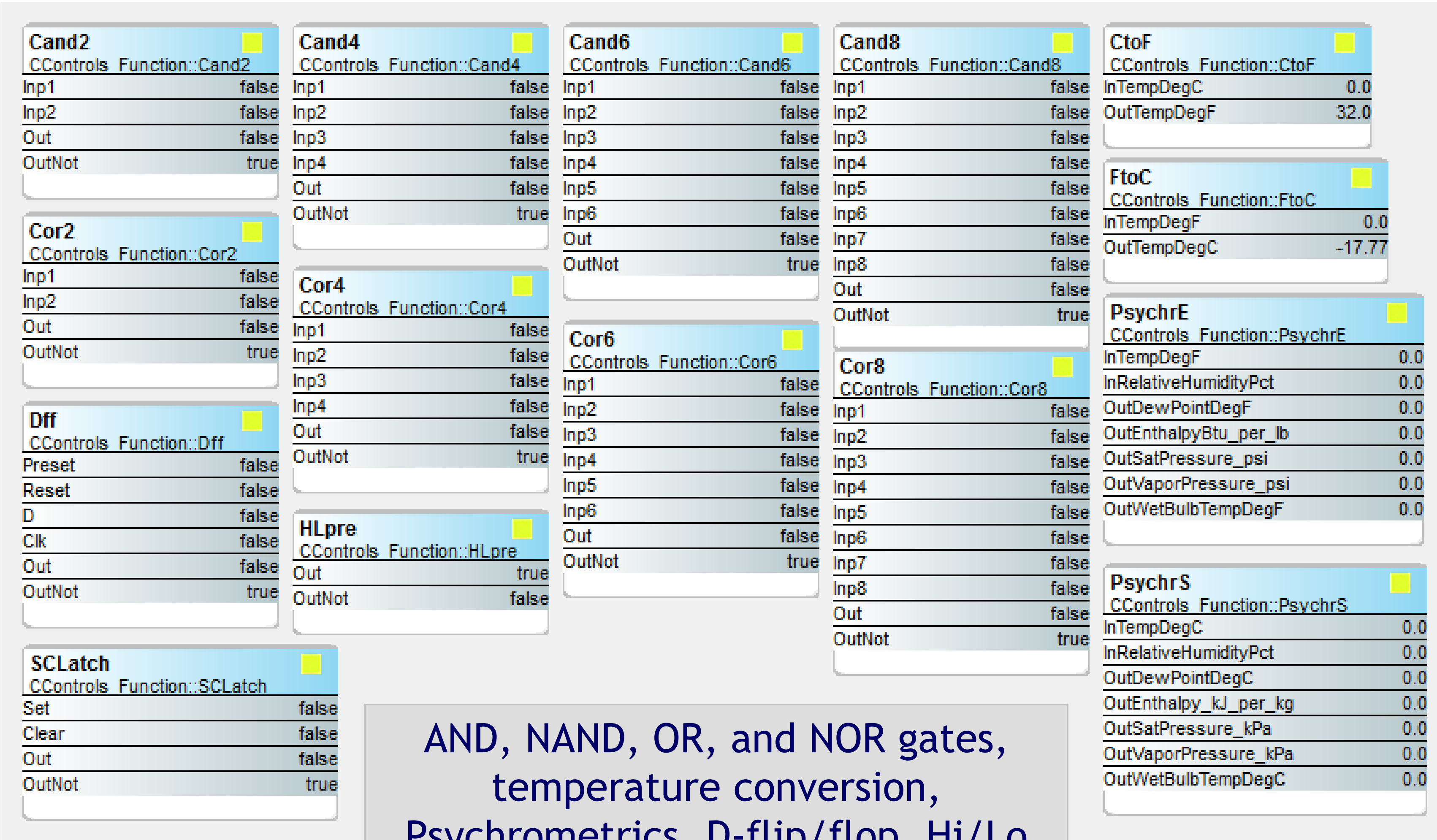
CControls Hardware Dependent Kits



Universal inputs, binary inputs, binary outputs, analog outputs, virtual points, web components, scan timer, platform and universal counter address particular platforms.

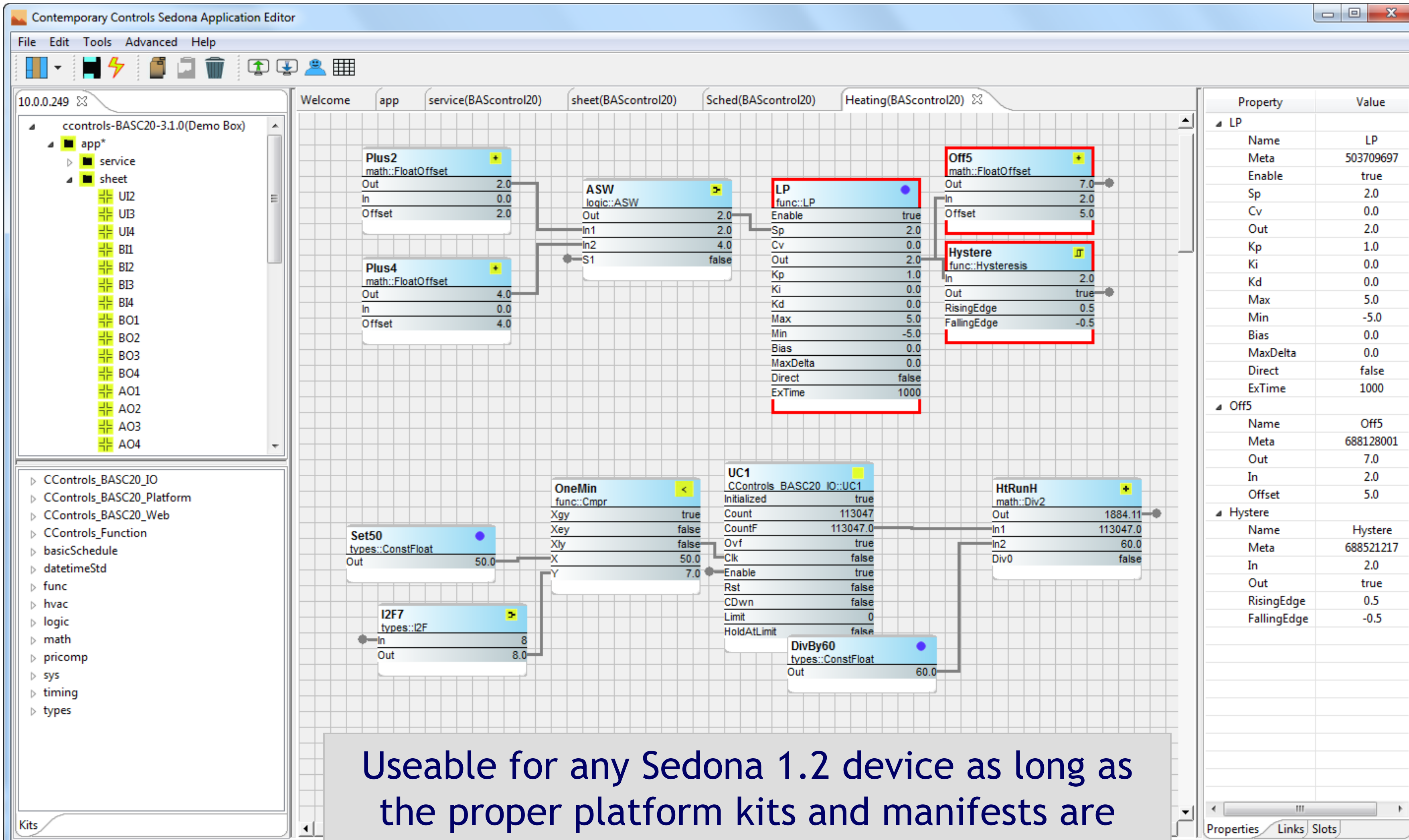
CONTEMPORARY CONTROLS

CControls_Function Kit



AND, NAND, OR, and NOR gates, temperature conversion, Psychrometrics, D-flip/flop, Hi/Lo Preset, and SCLatch. This hardware independent kit can be shared.

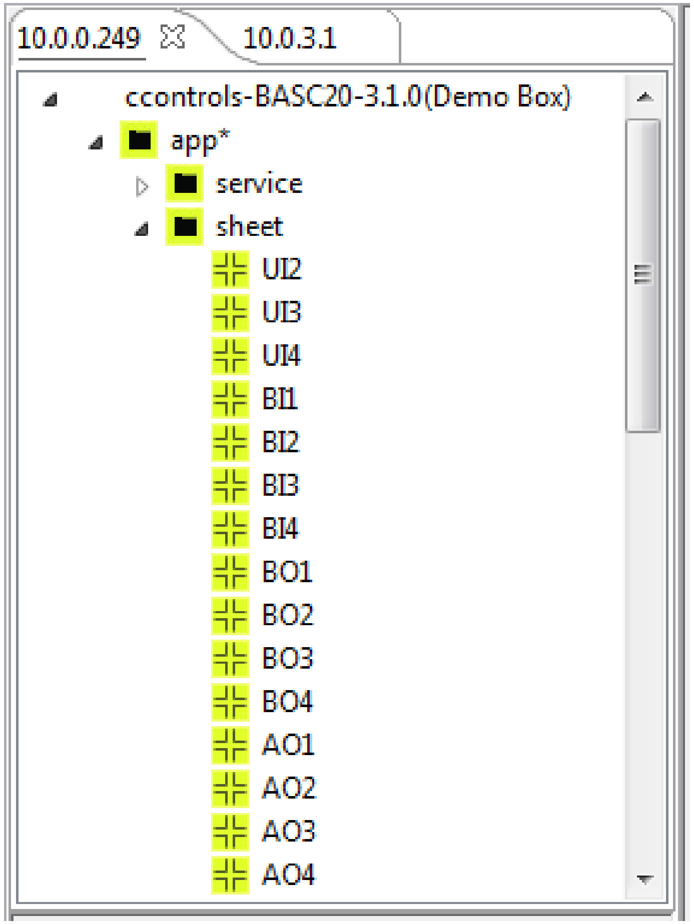
Our Sedona Tool – *Sedona Application Editor*



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

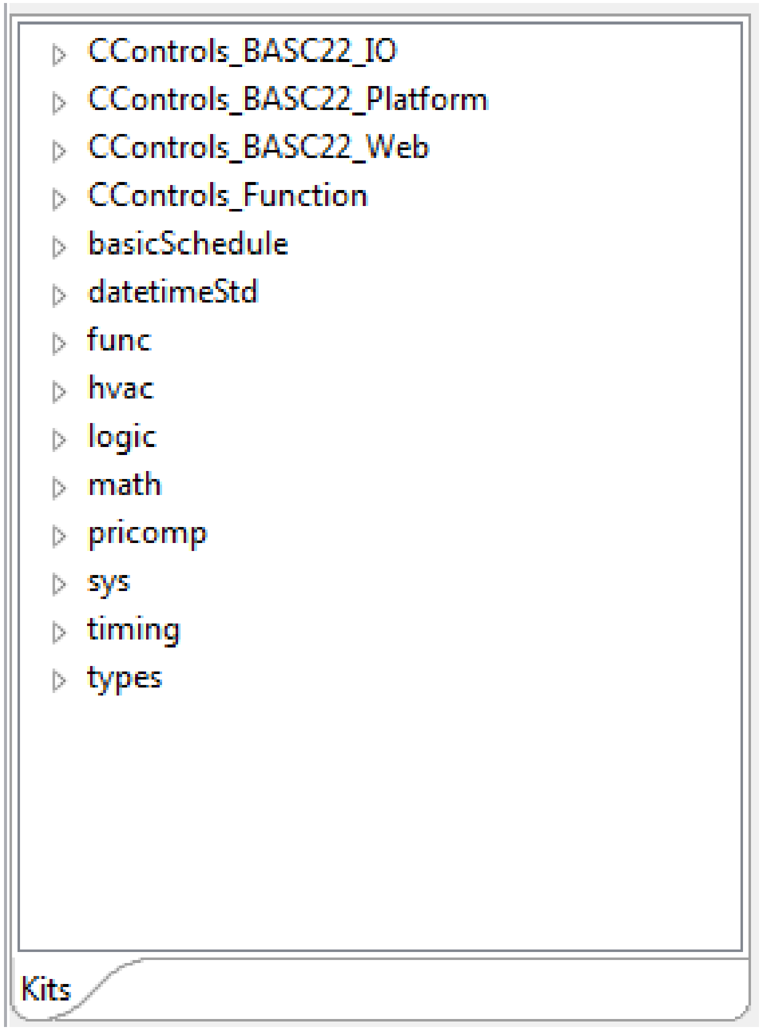
Navigation Pane – Showing Connected Devices

- Multiple IP address tabs for copying of programs between connected controllers or for simply viewing multiple devices
- Sedona platform ID and application name shown at the top
- Asterisk above App indicates program has been changed and needs to be saved to flash memory
- Navigation tree can be expanded down to individual components



Kits Pane – Showing Available Kits of Components

- The kits shown are the kits from the attached Sedona device and not all those available in the tool
- Tool must have installed all kits available in attached Sedona device
- Three types of kits:
  - Tridium 1.2 – no vendor name but just a group name
  - Hardware dependent – vendor, product and group names
  - Hardware independent – vendor and group names



Properties Pane – Showing Property Values

- Individual or multiple components can be highlighted to observe their slot names and property values
- Property values can be changed for configuration or testing
- Property values change on the screen as the wire sheet logic is executed

Property	Value
LP	
Name	LP
Meta	503709697
Enable	true
Sp	2.0
Cv	0.0
Out	2.0
Kp	1.0
Ki	0.0
Kd	0.0
Max	5.0
Min	-5.0
Bias	0.0
MaxDelta	0.0
Direct	false
ExTime	1000

Properties Pane – Showing the Slots

- A detailed view of component slots can be obtained showing the variable type and their facets

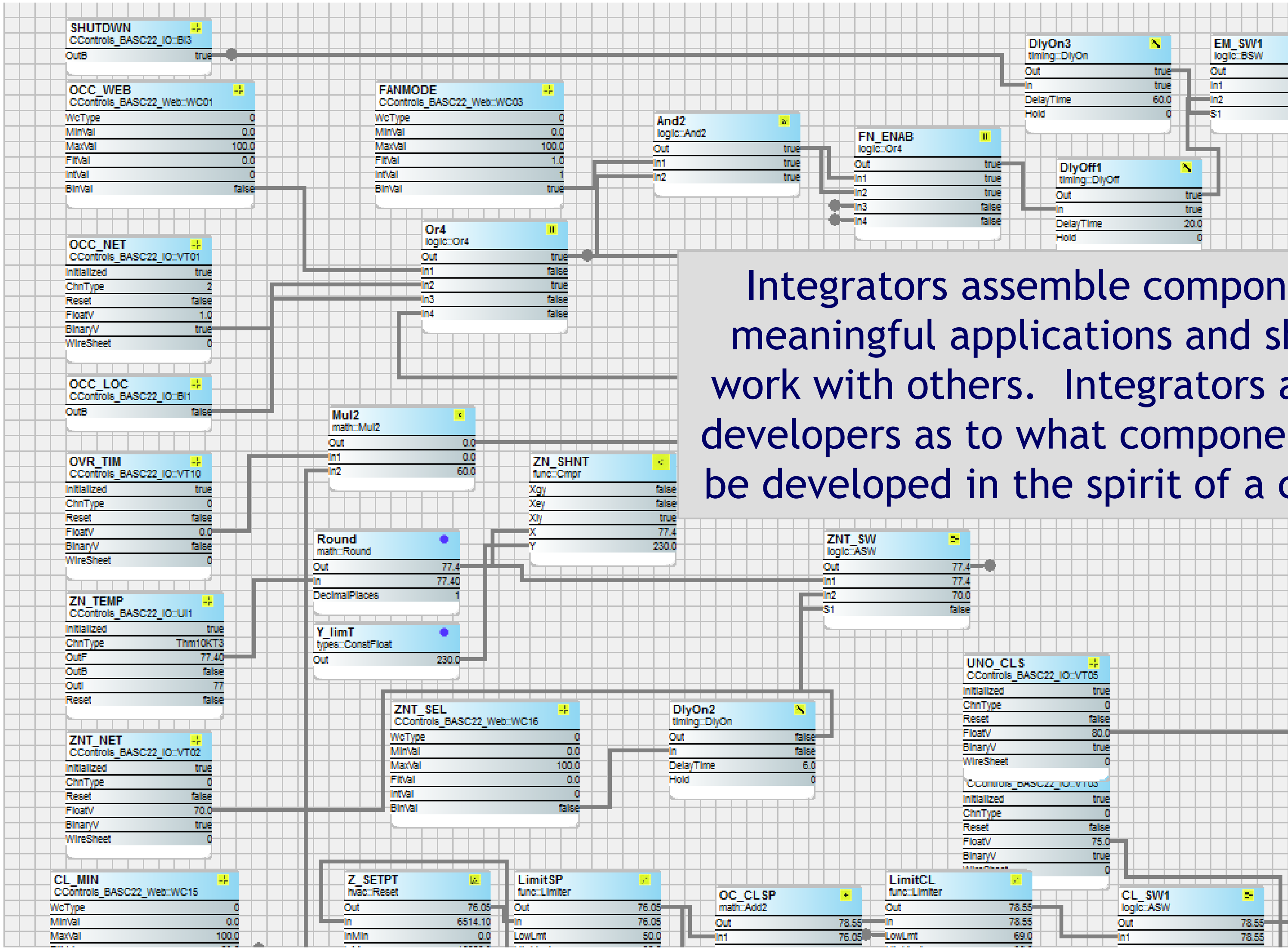
Name	Type	Facets
LP		
meta	int	[config]
enable	bool	[config]
sp	float	[summary, config]
cv	float	[precision=3]
out	float	[readonly]
kp	float	[min=0.0, config, precision=6]
ki	float	[unit="per_minute", min=0.0, config, precision=6]
kd	float	[unit="second", min=0.0, config, precision=6]
max	float	[config, precision=6]
min	float	[config, precision=6]
bias	float	[config, precision=6]
maxDelta	float	[min=0.0, config, precision=6]
direct	bool	[config]
exTime	int	[unit="millisecond", min=0, config]

Properties Pane – Showing the Links

- The connections between components are called Links and they can be identified by their:
  - Folder/component name/component type/slot name

From	To
LP	
/sheet/Heating/ASW/enable	/sheet/Heating/LP/sp
/sheet/Heating/LP/out	/sheet/Heating/Off5/in
/sheet/Heating/LP/out	/sheet/Heating/Hystere/in
Off5	
/sheet/Heating/LP/	/sheet/Heating/Off5/in
/sheet/Heating/Off5/out	/sheet/AO1/inpF
Hystere	
/sheet/Heating/LP/fallingEdge	/sheet/Heating/Hystere/in
/sheet/Heating/Hystere/out	/sheet/Heating/UC1/enable
/sheet/Heating/Hystere/out	/sheet/BO4/inpB

Generic RTU Application - *Work of an Integrator*



Integrators assemble components into meaningful applications and share their work with others. Integrators also inform developers as to what components need to be developed in the spirit of a community.

Conclusion – *an Open Controller and a Community*

- An open controller is defined as follows:
  - Open networking protocol – *BACnet*
  - Open programming language – *Sedona Framework*
  - Programming tool available without restriction – *SAE*
  - Community of developers and integrators –*Sedona community*
- Contemporary Controls is a Sedona community developer
  - Develops Sedona virtual machines for target hardware
  - Develops hardware dependent and independent components
  - Develops Sedona tools that aid in the creation of applications
- Integrators contribute to the community with their knowledge
  - Understand control strategies and sequence of operations
  - Can implement applications using components
  - Feedback to developers what components are needed

CONTEMPORARY CONTROLS®

Thank You

Visit our web site at http://www.ccontrols.com