BASC20

20-point BACnet/IP Sedona Field Controller

BAScontrol20

User Manual

Firmware Version 3.1

TD100700-0MB

CONTEMPORARY CONTROLS®

Trademarks

BASautomation, Contemporary Controls and CTRLink are registered trademarks of Contemporary Control Systems, Inc. BACnet is a registered trademark of the American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. Powered by Sedona Framework is a trademark of Tridium, Inc. Other product names may be trademarks or registered trademarks of their respective companies.

Copyright

Contemporary Control Systems, Inc. 2431 Curtiss Street Downers Grove, Illinois 60515 USA	Tel: Fax: E-mail: Web:	+1-630-963-7070 +1-630-963-0109 info@ccontrols.com http://www.ccontrols.com
Contemporary Controls Ltd 14 Bow Court Fletchworth Gate Coventry CV5 6SP UK	Tel: Fax: E-mail: Web:	+44 (0)24 7641 3786 +44 (0)24 7641 3923 info@ccontrols.co.uk http://www.ccontrols.co.uk
Contemporary Controls (Suzhou) Co. Ltd 11 Huoju Road Industrial Park — Science & Technology New District, Suzhou PR China 215009	Tel: Fax: E-mail: Web:	+44 (0)24 7641 3786 +44 (0)24 7641 3923 info@ccontrols.co.uk http://www.ccontrols.eu
Contemporary Controls GmbH Fuggerstraße 1 B D-04158 Leipzig Deutschland	Tel: Fax: E-mail: Web:	+49 (0)341 520359 0 +49 (0)341 520359 16 info@ccontrols.de http://www.ccontrols.eu

Disclaimer

Contemporary Control Systems, Inc. reserves the right to make changes in the specifications of the product described within this manual at any time without notice and without obligation of Contemporary Control Systems, Inc. to notify any person of such revision or change.

WARNING — This is a Class A product as defined in EN55022.
In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

Contents

Appendix

1 Introduction

The BAScontrol20 (BASC20) is a 20-point Sedona Field controller with a direct connection to an Ethernet network. The BASC20 was built on the Sedona Framework™. Ideally suited for structured wiring systems, the BASC20 is BACnet/IP compliant with a B-ASC device profile. Having a resident Sedona Virtual Machine (SVM), the unit is freely programmable using tools such as Niagara Workbench or a third-party Sedona tool. For remote Ethernet I/O applications, the unit can be configured via web pages.

The BASC20 provides a convenient mix of universal inputs, binary inputs and outputs as well as analog outputs. Models exist for both triac and relay binary outputs. The unit is ideal for unitary control or for expanding I/O points in the field via an Ethernet connection.

The BASC20 utilizes a powerful 32-bit ARM7 processor with 512 kB of flash memory plus a 16 Mbit serial flash file system for storing configuration data and an application program. By operating at the BACnet/IP level, the BASC20 can share the same Ethernet network with supervisory controllers and operator workstations. The unit can be configured for a fixed IP address or can operate as a DHCP client receiving its IP address from a DHCP server. A real-time clock with a super-cap backup allows for creating local schedules.

A 10/100 Mbps Ethernet port supports protocols such as BACnet/IP, Sedona Sox, HTTP and FTP. Configuration of universal inputs and virtual points can be accomplished using web pages. Type II and type III 10 k thermistors curves and a 20 k curve are resident in the unit. Current inputs can be measured using external resistors. Contact closures require a voltage-free source. Binary inputs and outputs as well as analog outputs require no configuration. The unit is powered from either a 24VAC/VDC source.

1.1 Features and Benefits

Versatile Control Device — field controller or remote Ethernet I/O

- BACnet/IP compliant
- B-ASC device profile
- Configurable by Workbench or third-party Sedona tool
- Direct connection to an Ethernet network
- Powered by a Sedona Virtual Machine

Flexible Input/Output — 20-points of I/O

- Eight configurable universal inputs:
- Thermistor, analog voltage, binary input, resistance, contact closure, pulse inputs (4 max)
- Four contact closure inputs
- Four analog voltage outputs
- Four relay or triac output (model specific)

1.2 Product Image and Main Features

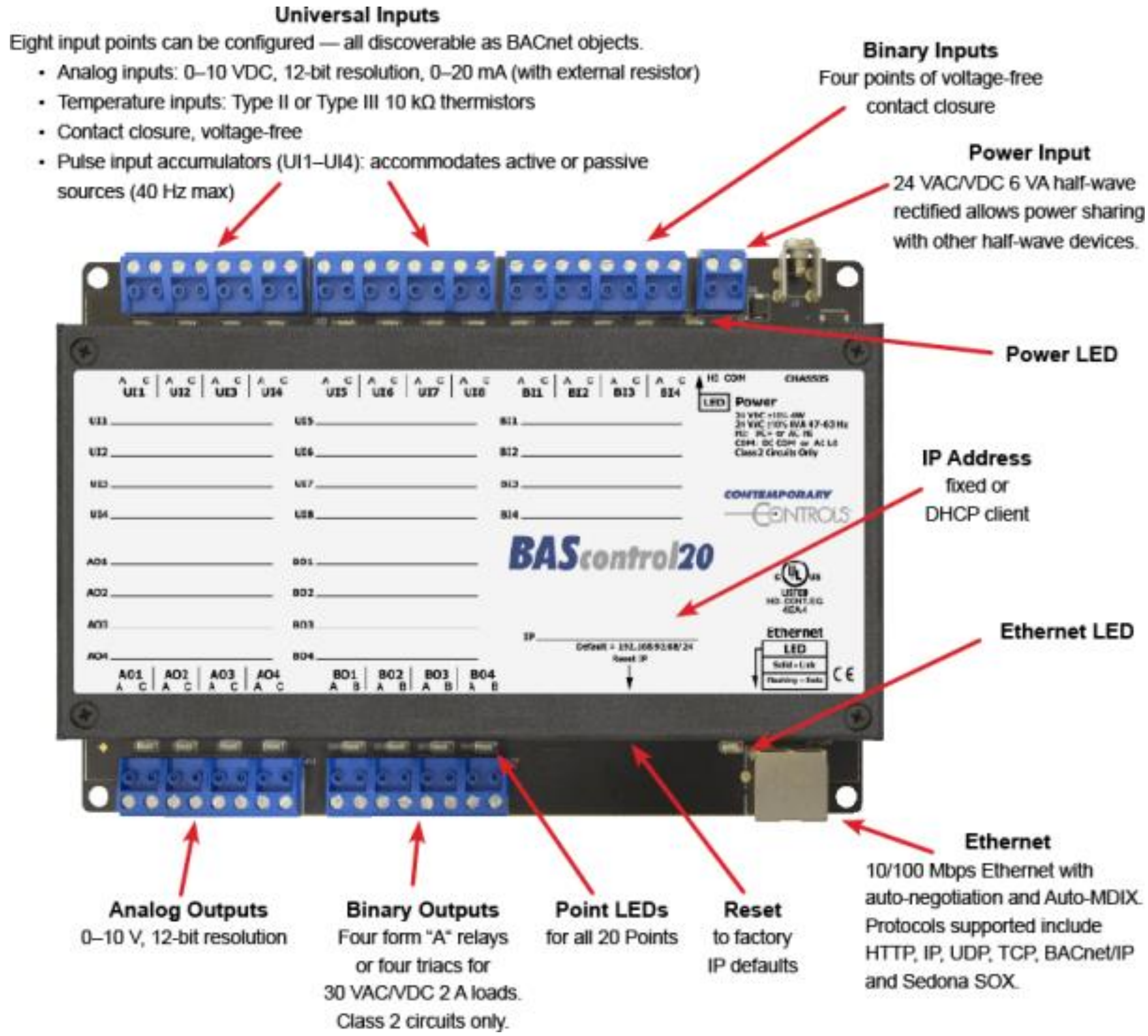


Figure 1 — BASC20 Main Features

2 Specifications

2.1 Universal Input (Channels UI1–UI8)

Configured As	Limits
Analog Input	0–10 VDC or 0–20 mA (with external resistor). 12-bit resolution. Input impedance 1 MΩ on voltage.
Temperature Input	Type II 10 kΩ thermistor –10° to +190 °F (–23.3° to +87.8°C) Type III 10 kΩ thermistor –15° to +200 °F (–26.1° to +93.3°C) Type 20 kΩ thermistor 15º to 215º F (-9º to +101º C)
Contact Closure Input	Excitation current 0.25 mA. Open circuit voltage 12 VDC. Sensing threshold 0.3 VDC. Response time 20 ms.
Pulse Input (points UI1–UI4)	0–10 VDC for active output devices. 0–12 VDC for passive devices (configured for internal pull-up resistor). 40 Hz maximum input frequency with 50% duty cycle.
Resistance	1 kΩ -100 kΩ range

2.2 Binary Inputs (Channels BI1–BI4)

Type	Limits
Contact Closure	Excitation current 0.25 mA. Open circuit voltage 12 VDC. Sensing threshold 0.3 VDC. Response time 20 ms.

2.3 Analog Outputs (Channels AO1–AO4)

Type	Limits
Analog Output	0—10VDC. 12-bit resolution. 4 mA maximum.

2.4 Binary Outputs (Channels BO1–BO4)

(Class 2 circuits only - requires external power source)

Type	Limits
Model BASC-20R	Normally Open contacts. 30 VAC/VDC 2 A.
Model BASC-20T	Isolated triac. 30 VAC 0.5 A.

2.5 Communications

Protocol	Data Link and Physical Layers
Ethernet	ANSI/IEEE 802.3 10/100 Mbps Ethernet. 10BASE-T, 100BASE-TX, auto-negotiation of speed and duplex. Auto-MDIX. 100 m maximum segment length. Default IP address is 192.168.92.68/24.

2.6 Protocol Compliance

Protocol	Compliance
BACnet/IP	ASHRAE 135-2008 annex J. Application specific controller device profile B-ASC.

2.7 Power Requirements

Item	Limits
Input power	24 VAC/VDC ± 10%, 47–63 Hz, 6 VA

2.8 General Specifications

Item	Description
Protection	All inputs and outputs (except for relay outputs and communications ports) are over-voltage protected up to 24 VAC and short-circuit protected.
Environmental	Operating temperature 0° to +60°C. Storage temperature –40°C to +85°C. Relative humidity 10–95%, non-condensing.
Weight	0.6 lbs. (0.27 kg).

2.9 LED Indicators

LED Indicator	Indication
UI1–UI8 Configured as Analog Input	Green: > 1% of range, otherwise off
UI1–UI8 Configured as Temperature Input	Green: sensor detected
UI1–UI8 Configured as Contact Input	Green: contact closed, otherwise off
UI1–UI8 Configured as Pulse Input	Green: pulse sensed, otherwise off
BI1–BI4 Contact Closure	Green: contact closed, otherwise off
AO1–AO4 Analog Output	Green: commanded output
BO1-BO4 Binary Output	Green: commanded output
Ethernet	Green: Link established; flashes with activity

2.10 Electromagnetic Compatibility

Standard	*Test Method*	*Description*	*Test Levels*
EN 55024	EN 61000-4-2	Electrostatic Discharge	6 kV contact
EN 55024	EN 61000-4-3	Radiated Immunity	10 V/m, 80 MHz to 1 GHz
EN 55024	EN 61000-4-4	Fast Transient Burst	1 kV clamp & 2 kV direct
EN 55024	EN 61000-4-5	Voltage Surge	1 kV L-L & 2 kV L-Earth
EN 55024	EN 61000-4-6	Conducted Immunity	10 V (rms)
EN 55024	EN 61000-4-11	Voltage Dips & Interruptions	1 Line cycle, 1–5 s @100% dip
EN 55022	CISPR 22	Radiated Emissions	Class A
EN 55022	CISPR 22	Conducted Emissions	Class B
CFR 47, Part 15	ANSI C63.4	Radiated Emissions	Class A

2.11 Field Connections

Terminal	Universal Inputs 1–8
UI1 A	Universal Input Point 1 High
UI1 C	Universal Input Point 1 Common
UI2 A	Universal Input Point 2 High
UI2 C	Universal Input Point 2 Common
UI3 A	Universal Input Point 3 High
UI3 C	Universal Input Point 3 Common
UI4 A	Universal Input Point 4 High
UI4 C	Universal Input Point 4 Common
UI5 A	Universal Input Point 5 High
UI5 C	Universal Input Point 5 Common
UI6 A	Universal Input Point 6 High
UI6 C	Universal Input Point 6 Common
UI7 A	Universal Input Point 7 High
UI7 C	Universal Input Point 7 Common
UI8 A	Universal Input Point 8 High
UI8 C	Universal Input Point 8 Common

Terminal	Relay Outputs (BASC-20R)
BO1 A	Output 1 normally-open contact
BO1 B	Output 1 common contact
BO2 A	Output 2 normally-open contact
BO2 B	Output 2 common contact
BO3 A	Output 3 normally-open contact
BO3 B	Output 3 common contact
BO4 A	Output 4 normally-open contact
BO4 B	Output 4 common contact

Terminal	Analog Outputs 1–4
AO1 A	Output Point 1 High
AO1 C	Output Point 1 Common
AO2 A	Output Point 2 High
AO2 C	Output Point 2 Common
AO3 A	Output Point 3 High
AO3 C	Output Point 3 Common
AO4 A	Output Point 4 High
AO4 C	Output Point 4 Common

Terminal	Binary Inputs 1–4
BI1 A	Input Point 1 High
BI1 C	Input Point 1 Common
BI2 A	Input Point 2 High
BI2 C	Input Point 2 Common
BI3 A	Input Point 3 High
BI3 C	Input Point 3 Common
BI4 A	Input Point 4 High
BI4 C	Input Point 4 Common

Terminal	Triac Outputs (BASC-20T)
BO1 A	Output 1 Isolated Triac
BO1 B	Output 1 Isolated return
BO2 A	Output 2 Isolated Triac
BO2 B	Output 2 Isolated return
BO3 A	Output 3 Isolated Triac
BO3 B	Output 3 Isolated return
BO4 A	Output 4 Isolated Triac
BO4 B	Output 4 Isolated return

2.12 Power Connection

Terminal	Power
HI	High AC or DC +
COM	AC or DC common

2.13 Ordering Information

Model	Description
BASC-20R	BAScontrol with 20 I/O points, includes 4 relay outputs
BASC-20T	BAScontrol with 20 I/O points, includes 4 triac outputs

2.14 Dimensional Drawing

All units are in mm.

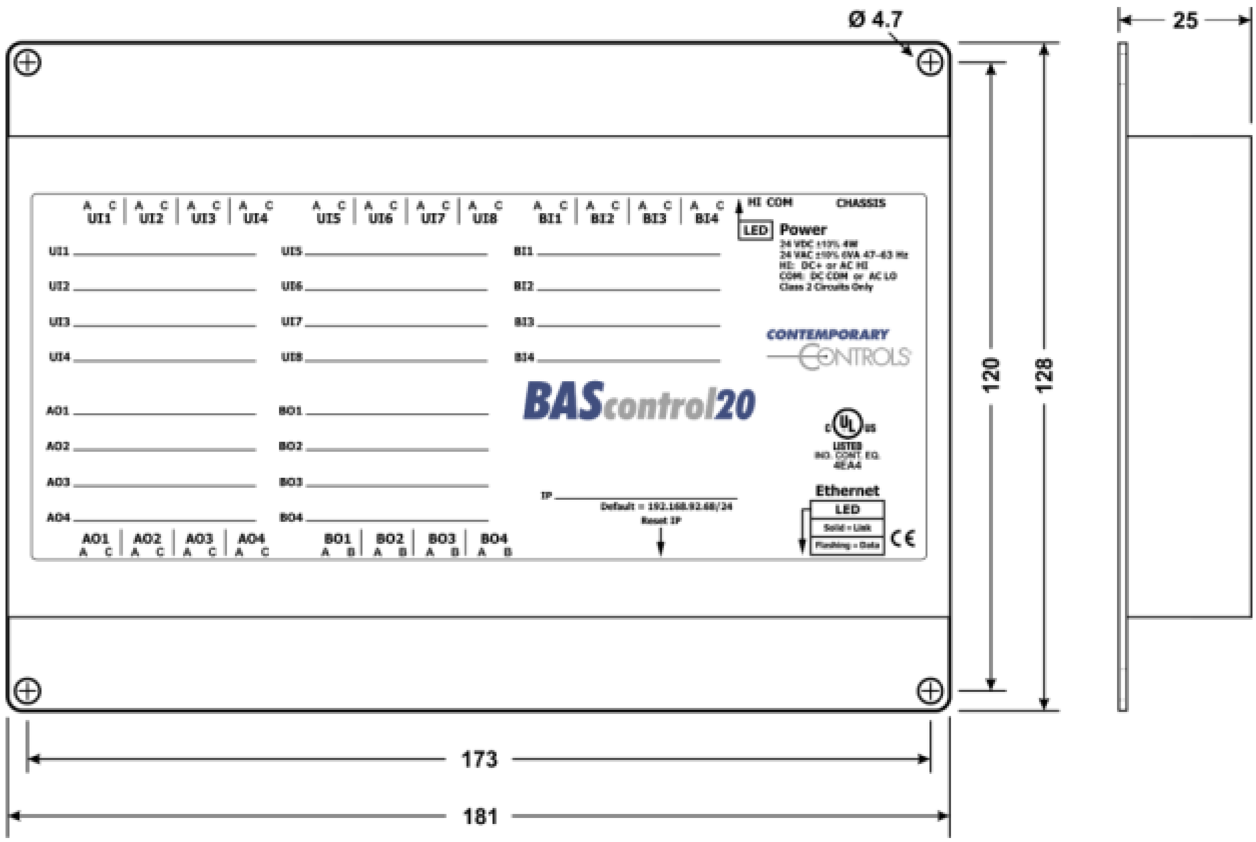


Figure 2 — BASC20 Dimensions

2.15 PICS Statement

CONTEMPORARY CONTROLS

BAScontrol20

20-point BACnet/IP Sedona Field Controller

BACnet Protocol Implementation Conformance Statement (Annex A)

Date: June 1, 2015
Vendor Name: Contemporary Controls
Product Name: BAScontrol20
Product Model Number: BASC-20R and BASC-20T
Applications Software Version: 1.2.28 **Firmware Revision: 3.1.1** **BACnet Protocol Revision:** 3
Product Description: BACnet/IP compliant 20-point field controller or remote I/O that allows a direct connection to Ethernet without the need of a BACnet router.

BACnet Standardized Device Profile (Annex L):
- ☐ **BACnet Operator Workstation (B-OWS)**
- ☐ **BACnet Building Controller (B-BC)**
- ☐ **BACnet Advanced Application Controller (B-AAC)**
- ☒ **BACnet Application Specific Controller (B-ASC)**
- ☐ **BACnet Smart Sensor (B-SS)**
- ☐ **BACnet Smart Actuator (B-SA)**

List all BACnet Interoperability Building Block Supported (Annex K):
- DS-RP-B Data Sharing — ReadProperty – B
- DS-WP-B Data Sharing — WriteProperty – B
- DS-RPM-B Data Sharing — ReadPropertyMultiple – B
- DS-COV-B Data Sharing — ChangeOfValue – B
- DM-DDB-B Device Management — Dynamic Device Binding – B
- DM-DOB-B Device Management — Dynamic Object Binding – B
- DM-DCC-B Device Management — Device Communication Control – B
- DM-TS-B Device Management — Time Synchronization – B

Segmentation Capability:
- ☐ Able to transmit segmented messages Window Size:
- ☐ Able to receive segmented messages Window Size:

Standard Object Types Supported:

Object Type Supported	Can Be Created Dynamically	Can Be Deleted Dynamically
Analog Input	No	No
Analog Output	No	No
Analog Value	No	No
Binary Input	No	No
Binary Output	No	No
Binary Value	No	No
Device	No	No

No optional properties are supported.

Data Link Layer Options:
- ☒ BACnet IP, (Annex J)
- ☒ BACnet IP, (Annex J), Foreign Device
- ☐ ISO 8802-3, Ethernet (Clause 7)
- ☐ ANSI/ATA 878.1, EIA-485 ARCNET (Clause 8), baud rate(s):
- ☐ MS/TP master (Clause 9), baud rate(s):
- ☐ MS/TP slave (Clause 9), baud rate(s):
- ☐ Point-To-Point, EIA 232 (Clause 10), baud rate(s):
- ☐ Point-To-Point, modem, (Clause 10), baud rate(s):
- ☐ LonTalk, (Clause 11, medium:
- ☐ Other:

Device Address Binding:
Is static device binding supported? (This is currently necessary for two-way communication with MS/TP slaves and certain other devices.) ☐ Yes ☒ No

Networking Options:
- ☐ Router, Clause 6 – List all routing configurations, e.g., ARCNET-Ethernet-MS/TP, etc.
- ☐ Annex H, BACnet Tunnelling Router over IP
- ☐ BACnet/IP Broadcast Management Device (BBMD)
  - Does the BBMD support registrations by Foreign Devices? ☐ Yes ☐ No

Character Sets Supported:
Indicating support for multiple character sets does not imply that they can all be supported simultaneously.
- ☒ ANSI X3.4
- ☐ IBM™/Microsoft™ DBCS
- ☐ ISO 8859-1
- ☐ ISO 10646 (UCS-2)
- ☐ ISO 10646 (UCS-4)
- ☐ JIS C 6226

If this product is a communication gateway, describe the types of non-BACnet equipment/network(s) that the gateway supports:
No gateway support.

June 1, 2015 TD100701-0XE

3 Installation

The BASC20 is intended to panel-mounted with screws (not provided).

3.1 Power Supply

The power source for the internal supply is applied via the two terminals labelled HI and COM. COM is for the power source return and also serves as the common ground connection. Primary 24 VAC/VDC (± 10%) power is applied to HI and COM. HI connects to a diode accomplishes half-wave rectified power — while providing reverse input voltage protection. The recommended power conductor size s 16–18 AWG (solid or stranded). Ground is directly connected to zero volts. Input connections are reverse-polarity protected.

WARNING: Powering devices can present hazards. Read the next two sections carefully.

3.1.1 Power Supply Precautions

Internally, the BASC20 utilizes a half-wave rectifier and therefore can share the same AC power source with other half-wave rectified devices. Sharing a common DC power source is also possible. Sharing AC power with full-wave rectified devices is NOT recommended. Full-wave rectified devices usually require a dedicated AC power source that has a secondary elevated above ground. Both secondary connections are considered HOT. AC power sources that power several half-wave devices have a common secondary connection called COMMON, LO, or GROUND. This connection might be tied to earth. The other side of the secondary is considered the HOT or HI side of the connection. Connect the HOT side of the secondary to the HI input on the BASC20 and the LO side to COM on the BASC20. All other half-wave devices sharing the same AC power source need to follow the same convention. When using a DC power source, connect its positive terminal to the HI input on the BASC20 and the negative terminal to COM on the BASC20. Reversing polarity to the BASC20 will not damage the BASC20.

WARNING: Devices powered from a common AC source could be damaged if a mix of half-wave and full-wave rectified devices exist. If you are not sure of the type of rectifier used by another device, do not share the AC source with it.

3.1.2 Limited Power Sources

The BASC20 should be powered by a limited power source complying with the requirements of the National Electric Code (NEC) article 725 or other international codes meeting the same intent of limiting the amount of power of the source. Under NEC article 725, a Class 2 circuit is that portion of the wiring system between the load side of a Class 2 power source and the connected equipment. For AC or DC voltages up to 30 volts, the power rating of a Class 2 power source is limited to 100 VA. The transformer or power supply complying with the Class 2 rating must carry a corresponding listing from a regulatory agency such as Underwriters Laboratories (UL).

3.2 Cabling Considerations

Function	Signalling and Data Rate	Minimum Required Cable	Maximum Segment Distance
Ethernet	10BASE-T 10 Mbps	Category 3 UTP	100 m (328 ft)
Ethernet	100BASE-TX 100 Mbps	Category 5 UTP	100 m (328 ft)
I/O	Unspecified	Solid: 16–22 AWG Stranded: 16–18 AWG	Unspecified

Table 1 — Cabling Considerations

* If using shielded cable, connect to chassis at only one point.

NOTE: Wire size may be dictated by electrical codes for the area where the equipment is being installed. Consult local regulations.

Observe in Table 1 that 10BASE-T segments can successfully use Category 3, 4 or 5 cable — but 100BASE-TX segments must use Category 5 cable. Category 5e cable is highly recommended as the minimum for new installations.

The Ethernet port of the BASC20 employs Auto-MDIX technology so that either straight-through or crossover cables can be used to connect to the network.

4 Field Connections

4.1 Sample BASC20 Wiring Diagram

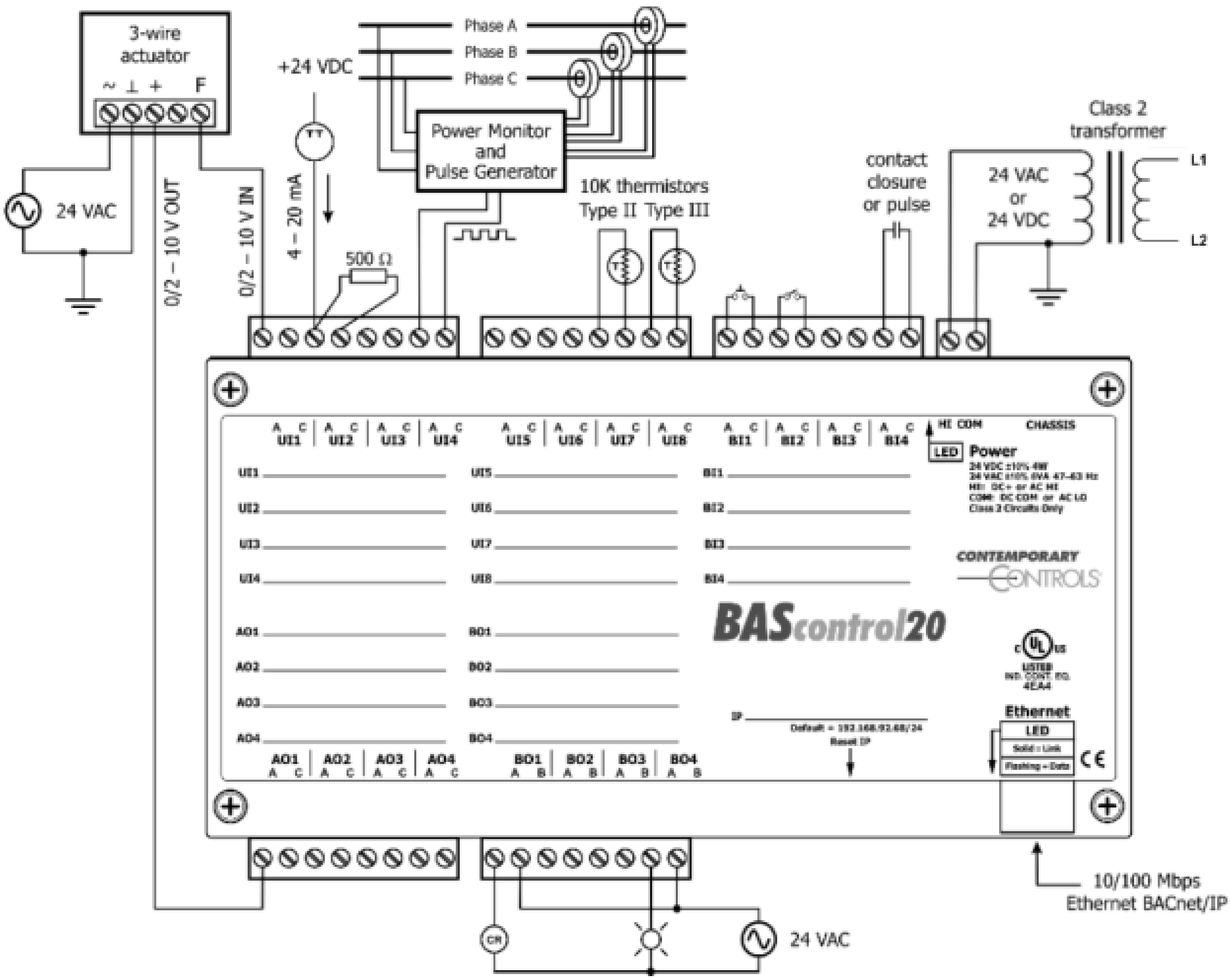


Figure 3 — Sample BASC20 Wiring Diagram

4.2 Universal Input — Configured as Analog Input

An analog input can measure voltage in the range of 0–10 VDC or it can measure current in the range of 0–20 mA with a 500 Ω external resistor. Transmitters that produce an elevated "zero" such as 2–10 VDC or 4–20 mA can be measured as well. Using the web page, configure the input for voltage. When set as a voltage input, the input impedance is 1 MΩ.

With voltage measurement, connect the more positive voltage to point ***A*** and the less positive to common ***C*** as shown in (Figure 4). On three-wire devices such as damper actuators, the output signal is referenced to the damper's power supply common. That common must be at the same reference as the BASC20 common. Notice the connections in the diagram. In this situation it is only necessary to attach the transmitter output to point ***A*** on the BASC20 input.

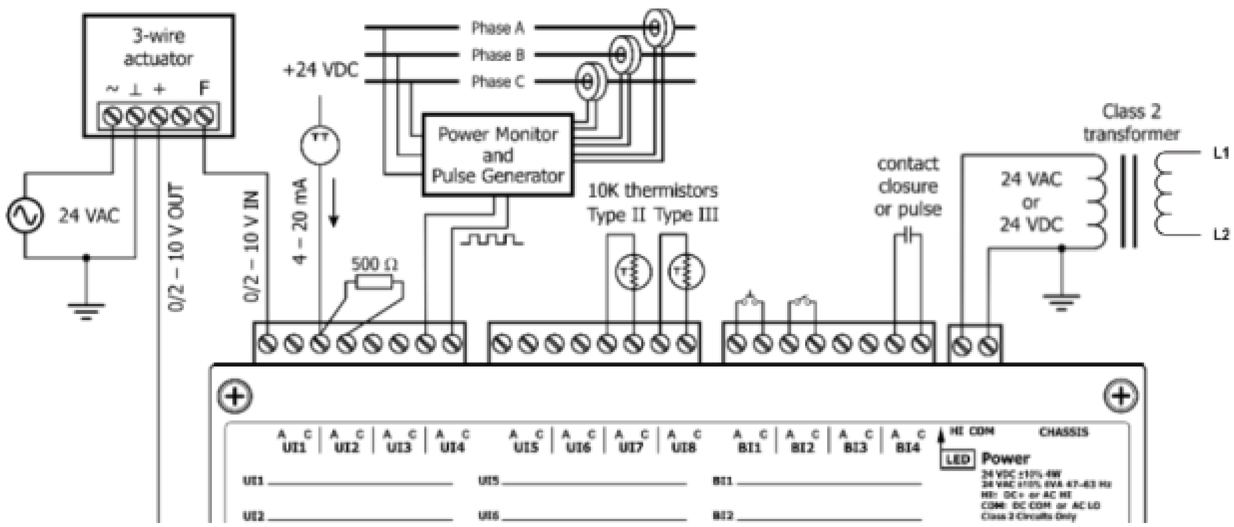


Figure 4 — Analog Input Connections

When measuring current from two-wire transmitters, remember the BASC20 sinks current to ground. A 500 Ω resistor is applied between points A and C on the input. To measure current, it must be driven into point A with respect to point C.

Care should be exercised when connecting to a three-wire current transmitter.

These are usually non-isolated devices between the power source and signal output. The BASC20 will sink current from its input to ground so the transmitter must source current from a positive potential to ground. If the three-wire transmitter works in this manner, it can be accommodated.

Four-wire transmitters usually have isolation between power supply and signal output so their output stage can usually be treated as a two-wire transmitter.

4.3 Universal Input — Configured as Temperature or Resistance Input

The BASC20 has built-in calibration curves for 10 kΩ Type II or Type III thermistors and 20 kΩ thermistors. These devices have a non-linear negative coefficient of resistance to temperature and provide a nominal resistance of 10 kΩ or 20 kΩ at 25°C. With a web browser, configure an input Channel Type for either Type II or Type III thermistor or 20 kΩ. As shown in (Figure 5), connect the two-wire thermistor to points ***A*** and ***C***. Polarity is not an issue. If averaging of temperature is desired, connect multiple thermistors in a series-parallel combination so that the nominal resistance remains at 10 kΩ or 20 kΩ as shown. Make sure that all devices are of the same type. The effective range of measurement varies by type. Type II 10 kΩ thermistors range from –10° to +190 °F (–23.3° to +87.8°C). Type III 10 kΩ thermistors range from –15° to +200 °F (–26.1° to +93.3°C). 20 kΩ thermistors range from 15º to 215º F (-9º to +101º C). An open input results in a fault condition and no LED indication for that point.

Two-wire potentiometers used as setpoint stations can be read by the universal input by selecting resistance on the drop-down menu. The resistance range is from 1kΩ to 100kΩ. Connections are made just like thermistors but no non-linear curves are used during resistance measurement. If unique curve-fitting is required, this could be accomplished using the Linearize component in the Sedona component family.

10 kΩ Type II or Type III Thermistors or 20 kΩ Thermistors

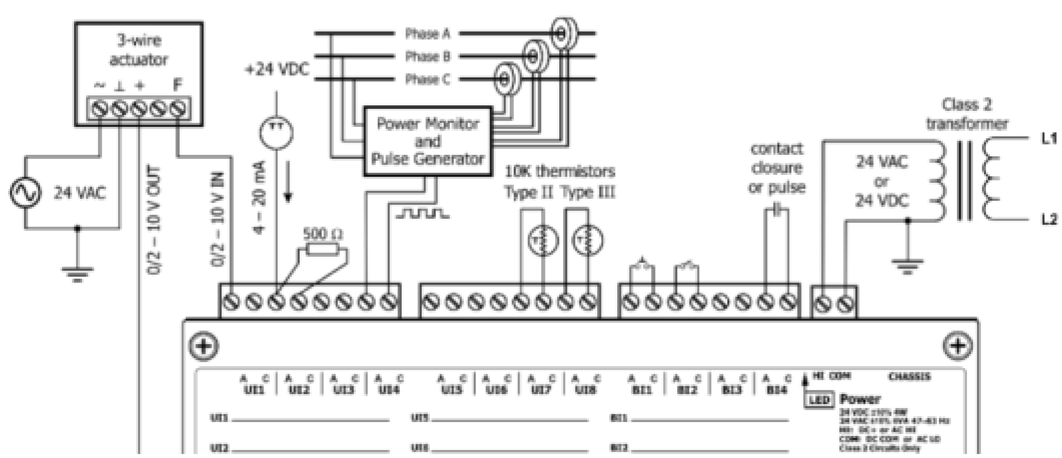


Figure 5 — Thermistor Connections

4.4 Universal Input — Configured as a Binary Input

To sense the action of a push-button or relay, the contacts must have no applied energy, and be rated for low-voltage, low-current switching. The BASC20 provides the energy to be sensed. With a web browser, access the Main Screen, click the title link of any channel UI1–UI8. Set the Channel Type to Binary Input and the Units to NO_UNITS. As shown in (Figure 6), connect the contacts between points A and C. For common mechanical contacts, polarity is not an issue. The open-circuit voltage is 12 VDC and the short-circuit current is 0.5 mA.

For solid-state switch sensing, we recommend that an attached solid-state device have an opto-isolated open-collector NPN transistor output stage with a collector-emitter output voltage (Vce) of at least 30 V. Output sinking current should be greater than 5 mA. The collector-emitter saturation voltage should be less than 0.2 V when sinking 2 mA. The emitter must be connected to point C and the collector to point A (the more positive point). The BASC20 sets the low-threshold to 3 V and the high-threshold to 6 V. When a contact is made or the solid-state switch is on (resulting in a saturated output), the voltage at point A is close to zero volts. The corresponding LED for that channel will be on. If the contact is opened or the solid-state switch is turned off, the voltage at point C quickly rises towards 12 V. Once the voltage passes the 6 V high-threshold, the "off" state is sensed. To return to the "on" state, this voltage must fall below 3 V. The three-volt difference is called hysteresis. There is no need to add an external pull-up resistor when using a contact closure input.

Contact closure inputs are sampled every 10 ms and for a change of state to be recognized, the input state must be stable for two consecutive samples. Therefore, contact closure response is 20 ms.

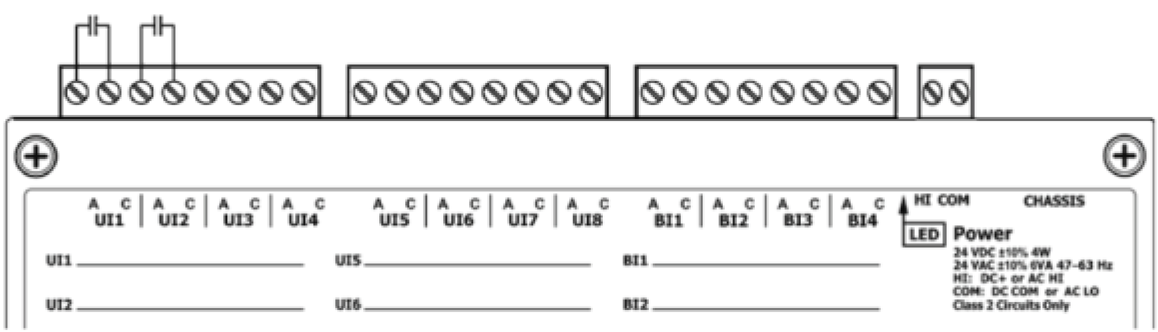


Figure 6 — Binary Input Connections

4.5 Universal Input — Configured as Pulse Input

When an input (UI1 – UI4) is configured for Pulse Input, a pulse rate up to 40 Hz can be measured, assuming a 50% duty cycle. The pulse device could have an active output or a passive output requiring a pull-up resistor. Both situations can be accommodated.

The input voltage range is 0–10 VDC and the installer can set both the low-threshold and high-threshold on the Pulse Input web page. The difference in the two thresholds is the hysteresis. You can detect a sinusoidal input by setting the high threshold below the positive peak and the low threshold above the negative peak. Setting both thresholds well away from the sinusoidal waveform peaks offers some noise immunity. It is not necessary for the input to swing from zero to 10 V. Any substantial swing within this range can be detected. The input impedance using Pulse Input is 100 kΩ when using active sensors. Connect the output of the pulse device to point A and the common to point C as shown in (Figure 7).

If the pulse device has a passive output requiring a pull-up resistor, the BASC20 can provide a 10 kΩ resistor to +12 VDC by checking a box on the configuration page. The two threshold values can still be set as needed.

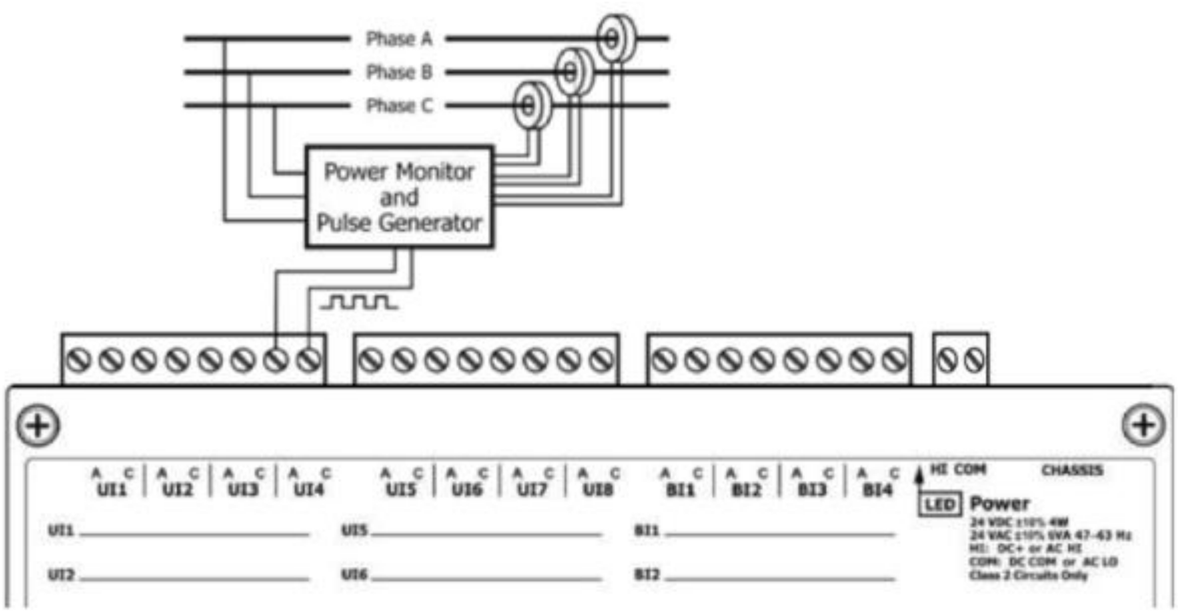


Figure 7 — Pulse Input Connections

4.6 Analog Outputs

Voltage in the range of 0–10 VDC can be outputted by assigning analog outputs (AO1–AO4). For analog output DC voltage, the output voltage is applied to point ***A*** with respect to ***C*** (common). There is no configuration necessary for analog outputs.

(Figure 8) illustrates connections to a three-wire damper actuator. The damper requires a 0–10 V command signal which can easily be accomplished by the BASC20. If position feedback is to be measured, connect the actuator output signal to UI1 and configure the universal input for analog input.

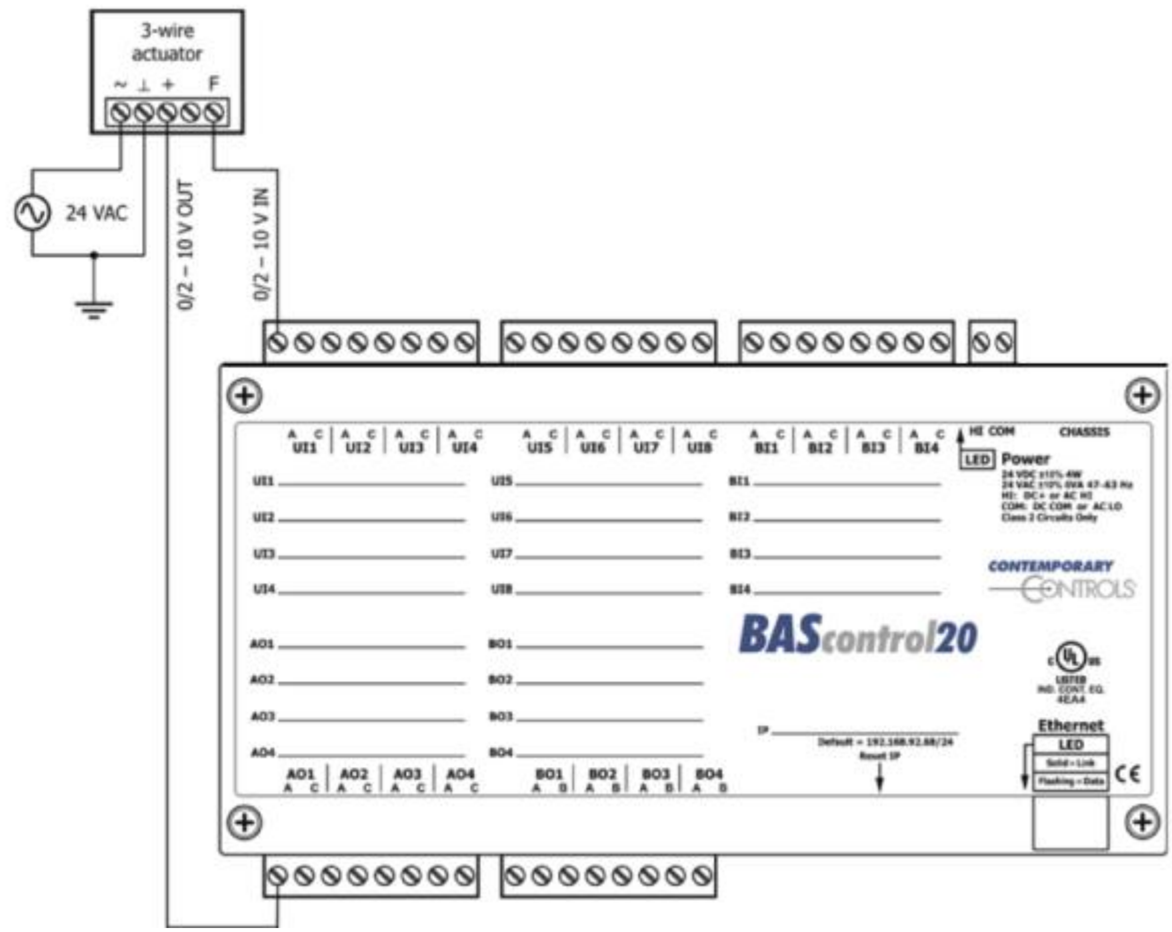


Figure 8 — Analog Output Connections

4.7 Binary Outputs

As shown in (Figure 9), four binary outputs (BO1 – BO4) are available. Each output requires an external power source. Two types of binary devices can be controlled. The BASC-20R provides four normally-open form “A” relay contacts that are rated at 30 VAC/VDC and 2 A. The BASC-20T provides isolated triac outputs that can drive loads up to 30 VAC and 0.5 A.

Each output voltage is applied to point ***A*** with respect to point ***B*** and is intended for Class 2 circuits only.

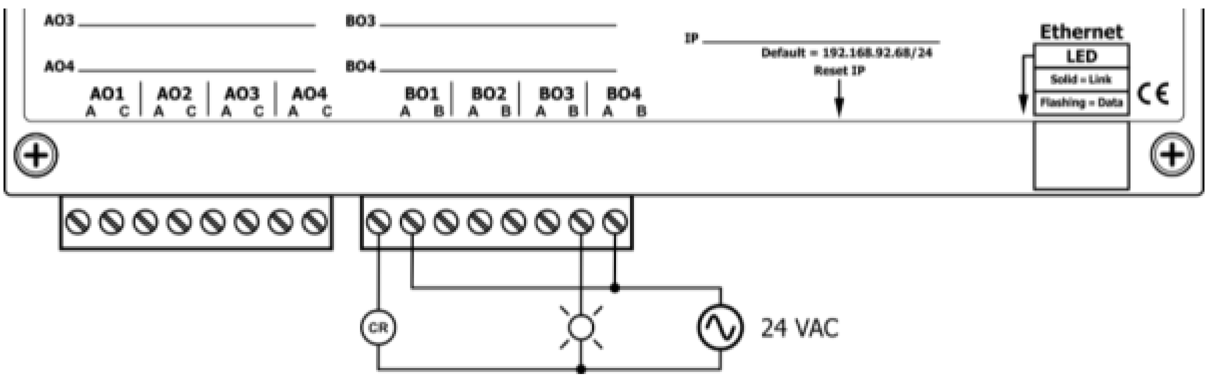


Figure 9 — Binary Output Connections

4.8 Binary Inputs

To sense the action of a push-button or relay, the contacts must have no applied energy, and be rated for low-voltage, low-current switching. The BASC20 provides the energy to be sensed. With a web browser, access the Main Screen, click the title link of any channel UI1–UI8. Set the Channel Type to Binary Input and the Units to NO_UNITS. As shown in (Figure 10), connect the contacts between points ***A*** and ***C***. For common mechanical contacts, polarity is not an issue. The open-circuit voltage is 12 VDC and the short-circuit current is 0.5 mA.

For solid-state switch sensing, we recommend that an attached solid-state device have an opto-isolated open-collector NPN transistor output stage with a collector-emitter output voltage (Vce) of at least 30 V. Output sinking current should be greater than 5 mA. The collector-emitter saturation voltage should be less than 0.2 V when sinking 2 mA. The emitter must be connected to point ***C*** and the collector to point ***A*** (the more positive point). The BASC20 sets the low-threshold to 3 V and the high-threshold to 6 V. When a contact is made or the solid-state switch is on (resulting in a saturated output), the voltage at point ***A*** is close to zero volts. The corresponding LED for that channel will be on. If the contact is opened or the solid-state switch is turned off, the voltage at point ***C*** quickly rises towards 12 V. Once the voltage passes the 6 V high-threshold, the "off" state is sensed. To return to the "on" state, this voltage must fall below 3 V. The three-volt difference is called hysteresis. There is no need to add an external pull-up resistor when using a contact closure input.

Contact closure inputs are sampled every 10 ms and for a change of state to be recognized, the input state must be stable for two consecutive samples. Therefore, contact closure response is 20 ms.

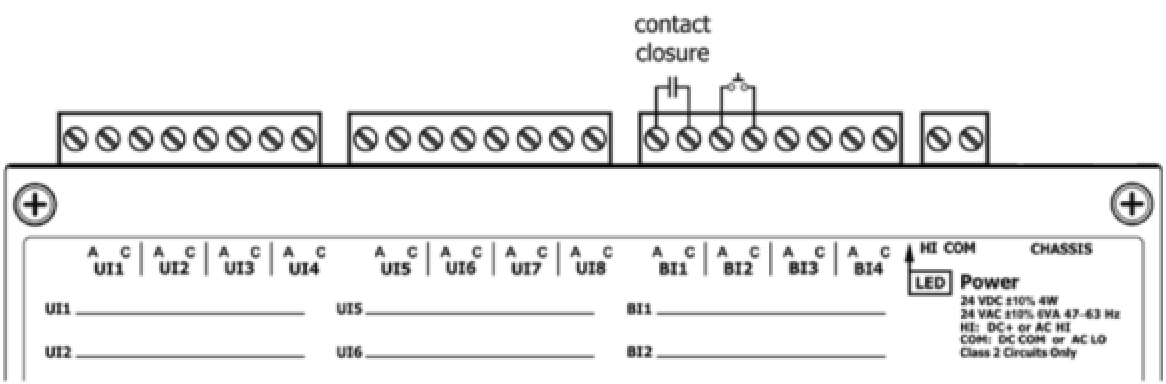


Figure 10 — Binary Input Connection

4.9 LEDs

To aid in troubleshooting, several LEDs have been provided.

The BASC20 has an Ethernet LED that glows green when properly linked to equipment operating at 10/100 Mbps and indicates activity by flashing.

LEDs to indicate I/O status follow the behaviour described in Table 2 below:

If the I/O channel is …	Green indicates …
a **Relay** output	the coil or triac is energized.
an **Analog** output	the command is greater than zero.
a **Contact** input	the contact is made.
a **Pulse** input	the input state changed.
a **Thermistor**	thermistor is connected
a **Resistor**	Resistor is connected
an **Analog** input	the signal is greater than 1% of span.

Table 2 — LED Behaviour

5 Configuration via a Web Browser

5.1 General Considerations

Some configuration of the BASC20 is required. This is accomplished while the unit is connected to a computer running a web browser (Java-enabled) that accesses the unit's built-in web server.

5.1.1 Ethernet Port

Auto-Negotiation

The Ethernet port on the BASC20 offers full auto-negotiation. A single cable links two Ethernet devices. When these devices auto-negotiate, the data rate will be 100 Mbps only if both are capable of that speed. Likewise, full-duplex will only be selected if both can support it. If only one device supports auto-negotiation, then it will default to half-duplex mode and match the data rate of the non-auto-negotiating device.

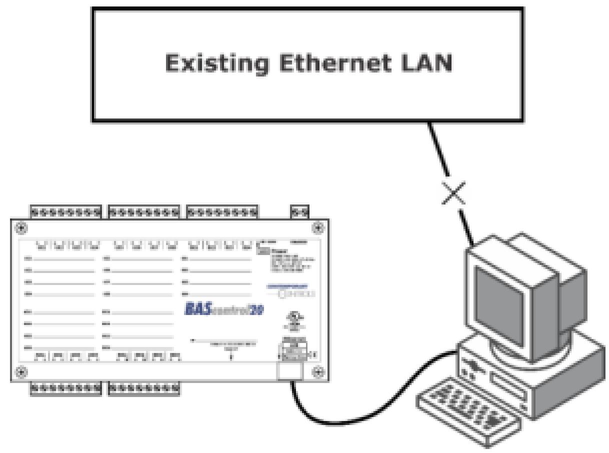


Figure 7 — Setup for Initial IP Address Configuration by Web Browser

Auto-MDIX (Auto-Crossover)

When interconnecting two Ethernet devices, a straight-through cable or crossover cable can be used — but if one device uses Auto-MDIX, the cable wiring does not matter; Auto-MDIX adjusts for either type.

Reset Switch

To reset the BASC20 to its default values of the IP address (192.168.92.68) and netmask (/24 or 255.255.255.0), press the reset switch (see Figure 10 for location) while the unit is powered. Follow the instructions under the section 5.1.2.

5.1.2 Secure Login and Reset (Recovery Mode)

To reset the unit to its default IP values and login credentials, press the reset switch for over 4 seconds. (See Figure 11 for the switch location.) This forces the **recovery mode** — confirmed by alternate flashing of UI1-UI4 and AO1-AO4 channel LEDs. This action restores the default settings for the user ID (admin), password (admin), IP address (192.168.92.68) and subnet mask (255.255.255.0). Access the main web page and make changes to the IP configuration and login credentials, and then click ***Restart Controller*** to exit recovery mode.

5.1.3 Web Server Initial Access

5.1.3.1 Web Server

The BASC20 contains an interactive web server, accessible from any Internet-compatible PC on the local network. It is compatible with all recent browsers. It is factory programmed with a default IP address of 192.168.92.68 and a Class C subnet mask of 255.255.255.0. Once configured, changing its IP address is strongly encouraged.

5.1.3.2 Initial Access

The hardware arrangement for initially setting the BASC20 IP address appears in (Figure 7). The PC should be temporarily disconnected from the Ethernet LAN in case the BASC20's default address matches that of a device on the existing LAN. The procedure for altering the IP address creates a temporary LAN composed of nothing but the BASC20, the PC used to configure it and a CAT5 cable connecting the two. Since the BASC20 supports Auto-MDIX, either straight-through or crossover cable can be used.

For initial configuration, the PC chosen for the procedure should temporarily have its IP address modified as shown in (Figure 12) — which employs a Windows® 7 example.

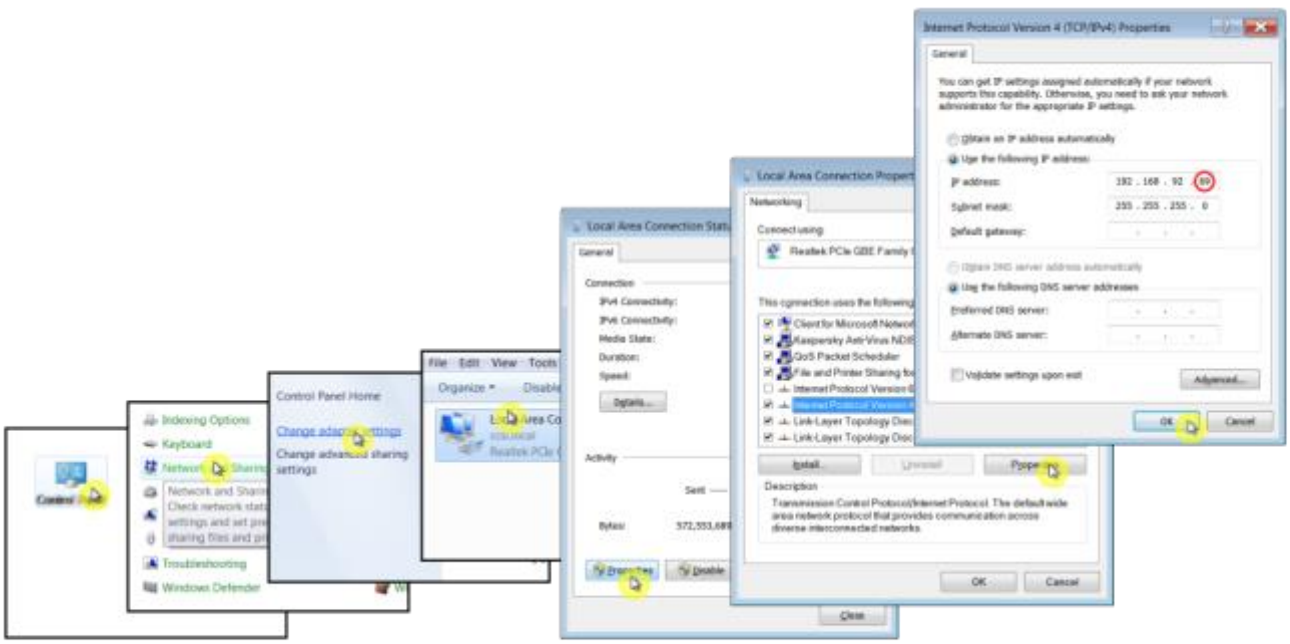


Figure 12 — Steps for Changing the IP Address of the PC Used for Setup

(Figure 12) uses an IP address for the PC of 192.168.92.69, but the final quad of the address could be any value 1–254 except for 68 which is used by the BASC20. After setting the IP address of the PC to the same LAN as the BASC20, a browser can access the BASC20 default IP address.

When first accessing the BASC20, you must provide the default login credentials. We strongly advise you to change these values as discussed in Section 5.1.4.4.

(Figure 13) displays the Main Page that appears after you first login to the BASC20. This page displays channel data in five columns:

- Universal Inputs (Channels UI1–UI8)
- Binary Inputs (Channels BI1–BI4)
- Analog Outputs (Channels AO1–AO4)
- Binary Outputs (Channels BO1–BO4)

Each of the 28 channels has three features:

- title link — If clicked, it displays a configuration screen (see Figure 18).
- data field* — You can read a value or enter one if forced (see Section 5.1.10).
- checkbox* — If checked, you can force the channel value (see Section 5.1.10).

*** You need to check the box before making a change.**

Figure 13— Main Page

Six buttons occupy the bottom of the Main Page. They function as follows:

- **System Configuration** described in Section 5.1.4
- **System Status** described in Section 5.1.5
- **Set Time** described in Section 5.1.6
- **Virtual Points** described in Section 5.1.10
- **Web Components** described in Section 5.1.7
- **Restart Controller** described in Section 5.1.8
- **Auto Refresh (On/Off)** described in Section 5.1.9

5.1.4 System Configuration

Clicking the *System Configuration* button shown in the lower-left area of (Figure 13) opens the window depicted in (Figure 14) — where you can configure the settings discussed in the next four sections.

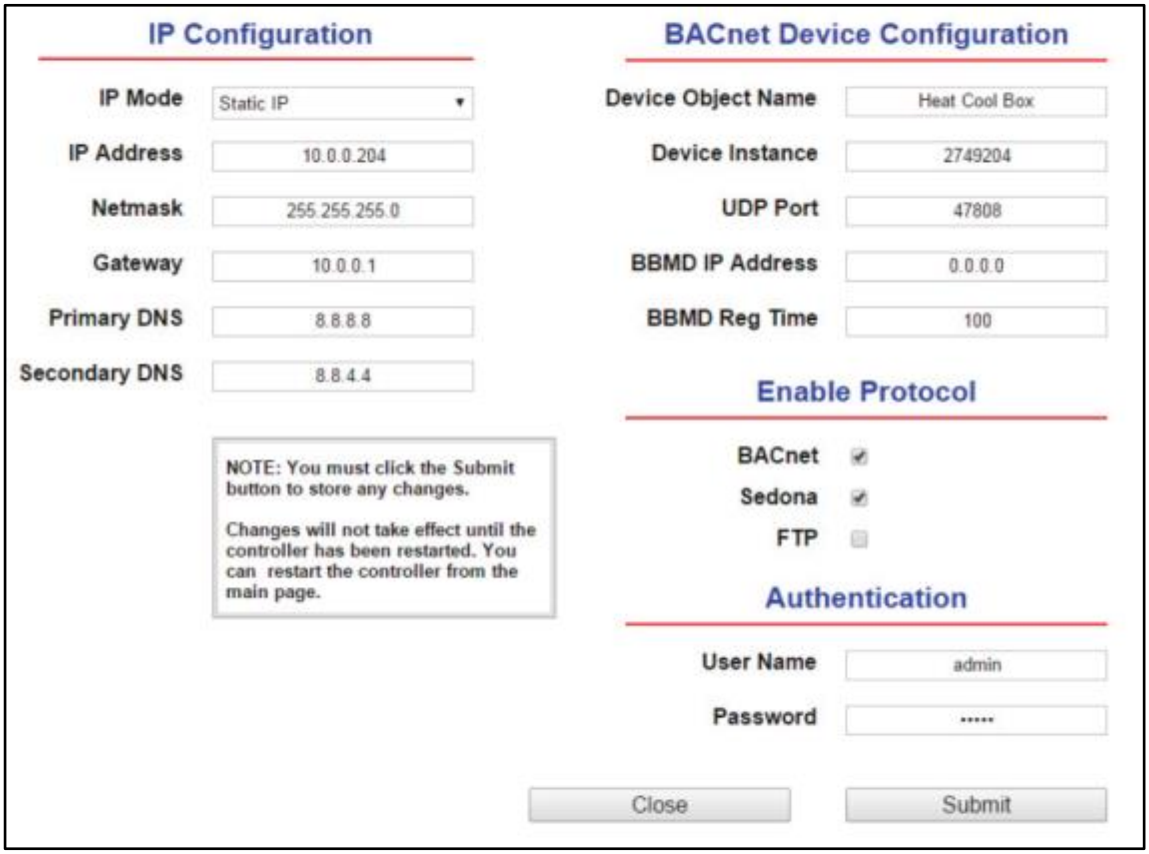


Figure 14 — System Configuration Window

Four sections and two special buttons exist on the System Configuration screen:

- IP Configuration is discussed in Section 5.1.4.1.
- BACnet Device Configuration is discussed in Section 5.1.4.2.
- Enable Protocol is discussed in Section 5.1.4.3.
- Authentication is discussed in Section 5.1.4.4.

5.1.4.1 IP Configuration

As shown in (Figure 14) the following parameters can be adjusted, followed by a ***Submit***:

- **IP Mode** Choose either ***Static IP*** (the default) or ***DHCP***.
- **IP Address** Changing the default value of 192.168.92.68 is recommended.
- **Netmask** The default value of 255.255.255.0 is adequate for most users.
- **Gateway** If your Ethernet LAN has a gateway (router) enter its IP address here.
- **Primary DNS** Enter your primary domain name service address
- **Secondary DNS** Enter your secondary domain name service address

After the BASC20 has been given its initial configuration, it will be ready for use in the full original Ethernet network. The temporary network constructed in (Figure 7) should be dismantled and the PC re-configured to restore its original IP address.

5.1.4.2 BACnet Configuration

As shown on the right side of (Figure 14), the following parameters can be adjusted, followed by a ***Submit***:

- **Device Object Name** You must change the default name (BAScontrol System) to be ***unique*** throughout the ***entire BACnet internetwork***.
- **Device Instance** This 22-bit value (0–4,194,303) ***must be unique*** throughout the ***entire BACnet internetwork***. It defaults to ***2749***.
- **UDP Port** The default of 47808 should usually not be changed.
- **BBMD IP Address** Enter the address of the BBMD with which the BASC20 will perform Foreign Device Registration (FDR) — if the BBMD is not in the same subnet as the BASC20.
- **BBMD Reg Time** Specify the seconds between successive FDR registrations. Default is 100.

5.1.4.3 Enable Protocol

On the right side of (Figure 14), three functions can be adjusted, followed by a ***Submit***:

- **BACnet** Disabling BACnet (on by default) will free more memory for Sedona.
- **Sedona** Disabling Sedona (on by default) will free more memory for BACnet.
- **FTP** If needed, enable FTP (which by default is unchecked). If you select FTP, BACnet and Sedona are automatically de-selected.

5.1.4.4 Authentication

On the right side of (Figure 14), you can use up to 63 characters to specify ***User Name*** and ***Password***, followed by ***Submit***:

- **User Name** You can change the default *admin* to any *User Name* you wish.
- **Password** You can change the default *admin* to any *Password* you wish.

NOTE: After checking the submit button after any change you must restart the controller from the main web page.

5.1.4.5 Kit Update

Consult the BASC20 support page for detailed instructions on using this feature:

5.1.5 System Status

This read-only screen is displayed in (Figure 15) and reports the three items:

- **Firmware Revision** Your firmware version is listed in the upper-left corner.
- **MAC ID** The Ethernet MAC address in the middle.
- **Available Memory** This value in the upper-right corner will vary often.
- **System Message Log** is discussed in Section 5.1.5.1.

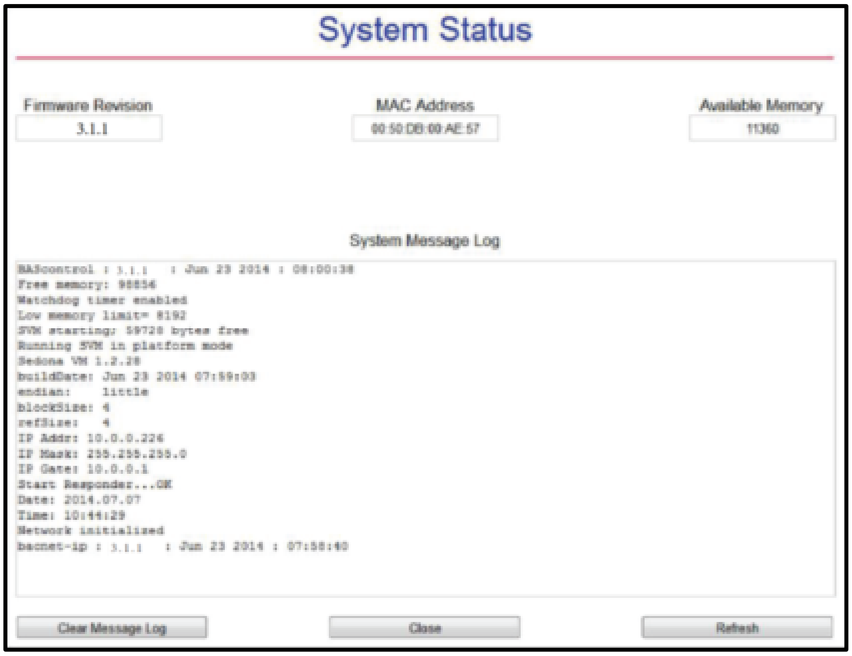


Figure 15 — System Status Window

5.1.5.1 System Message Log

Various items are reported in Figure 15 after a power up cycle. Information is used by technical support at Contemporary Controls. The information can be cleared by checking the Clear Message log button. To refresh the page, click on Refresh.

5.1.6 System Time

Clicking the *Set Time* button shown in the lower-right area of (Figure 13) opens the window depicted in (Figure 16) — where you can configure these settings:

- System Time — Here you can ***read*** the date and time or *manually* set them — ***but only if you disable*** the NTP option.
- NTP Configuration — is discussed in Section 5.1.6.1.
- DST Configuration — is discussed in Section 5.1.6.2.

Note: Refer to Date Time STD Kit on page 60 regarding the use of System Offset.

5.1.6.1 NTP (Network Time Protocol)

NTP is a protocol which synchronizes clocks to UTC (Coordinated Universal Time). By default as shown in the upper-right portion of (Figure 16), NTP is disabled, but an NTP server IP address is shown. When NTP is enabled, the NTP server will be queried and the BASC20 time will be synchronized at startup — and at midnight during each refresh period.

- **NTP Enable** — You can enable Network Time Protocol (disabled by default).
- **NTP Server** — Change the default IP address (130.149.17.21), if needed.
- **Time Zone** — Set the Time Zone to match that of your location.
- **NTP Refresh (Days)** — Change the default value (1) if needed.

NTP does not support local time zone changes such as for DST (Daylight Saving Time, aka Summer Time).

5.1.6.2 DST (Daylight Saving Time, aka Summer Time)

DST Configuration *is provided as displayed in the lower-right portion of (Figure 16), because NTP cannot adjust them. Drop-down menus allow you to set the date and the time after midnight for enabling and disabling DST. Be sure to click Update* NTP & DST *after making changes.*

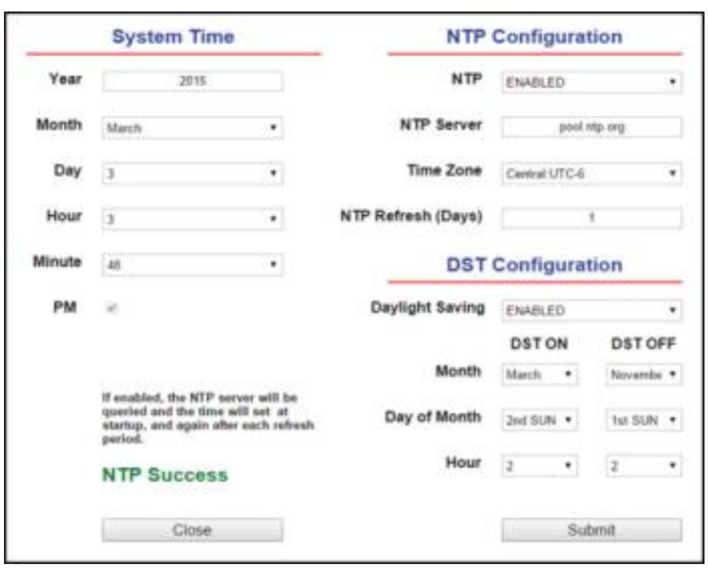


Figure 16 — System Time Window

5.1.7 Web Components (WC01–WC04)

Web components provide a means of interacting with the Sedona wire sheet via a web browser versus using a Workbench tool. These are custom components developed by Contemporary Controls which are provided in the Ccontrols_BASC20_Web kit. Configuring the 48 web components is accomplished from eight web pages. But first, each web component must be configured in the wire sheet as a wire sheet input or a wire sheet output using a drop-down box. In addition, high and low limits can be entered for wire sheet inputs. Returning to the web pages, for every web component (WC), a description and value can be entered. The description field is only used as an aid to the systems integrator in understanding the function of the component. If the component is configured as a wire sheet input (float, integer or binary), the assigned limits will restrict the range of the variable's entry. This eliminates the need to add limit logic on the wire sheet. For wire sheet outputs, limits are ignored. A green tag means that the web component has been placed on the wire sheet.

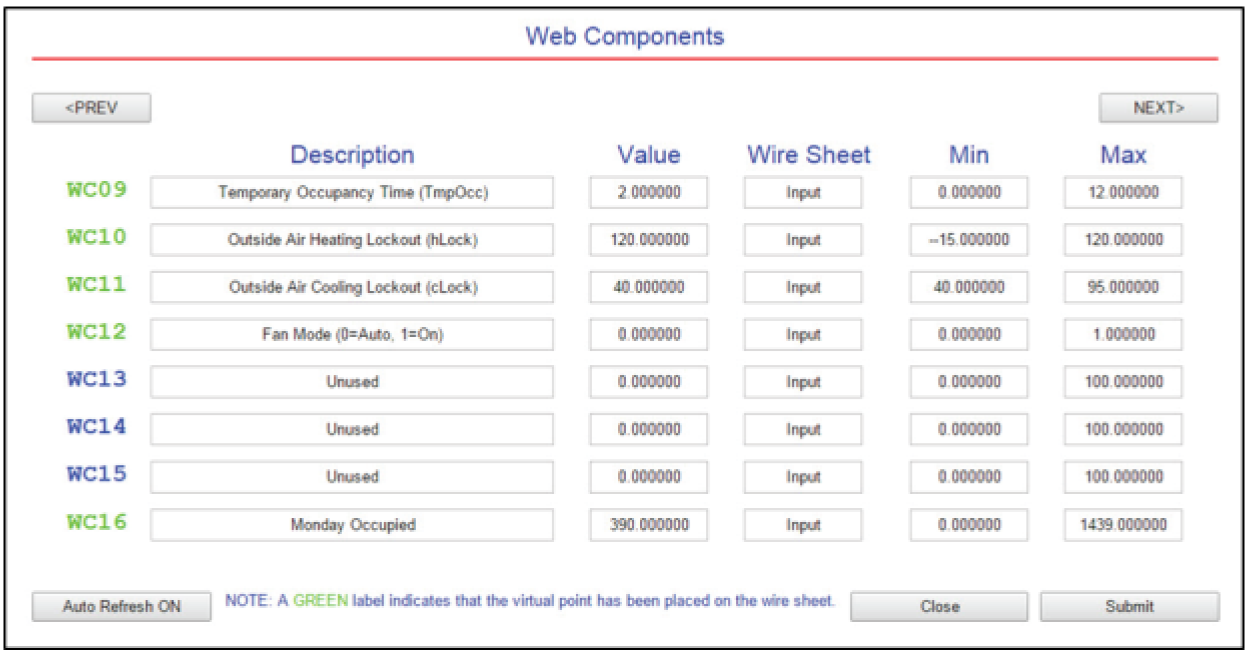


Figure 17 — Web Components Screen Showing Example Data

5.1.8 Restart Controller

Click this button to reboot the BASC20 that is currently targeted by your browser. Extreme care should be exercised when resetting a commissioned controller.

5.1.9 Auto Refresh (On/Off)

Click this button to update the BASC20 values currently displayed by your browser. With ***Auto Refresh*** ON, values periodically update. If OFF, there is no updating.

5.1.10 Virtual Points

The 24 virtual points have their own web page. Virtual points allow communication to and from a BACnet client to the BASC20 wire sheet. Virtual points are usually setpoints, calculated data or status points that do not impact the real input/output points that exist on the controller. The first eight virtual points (VT01-VT08) reside in persistent memory for up to seven days when power is removed. The remaining points are not retentive. A GREEN tag means that the virtual point has been placed on the wire sheet. The label hover text indicates if the point is configured as "Read from Wire Sheet" or "Write to Wire Sheet".

5.1.11 Forcing I/O Points from the Main Web Page

There is one feature available on the main web page that could be useful for checkout but must be done with great care. Both input and output points can be forced to states and values different from program generated values. Looking at the main web page, it is possible to both read and write values for the 20 real I/O points and 24 virtual points. There is no issue with reading points — only writing points. Just to the right of the value field is a checkbox. If you **hover your cursor** over this checkbox, this tool tip will display: ***Click to Force Channel***. To change an input or output value, check this box before making a value change. This override value will remain until the checkbox is unchecked. The same can be done to outputs.

Caution: Use great care when forcing an input or output on a commissioned system to avoid damage to equipment or process or injury to personnel.

5.2 Channel Configuration

To configure a real input/output channel, access the Main Page (Figure 13), click on the title link for the channel of interest and make adjustments in the new screen that appears (Figure 18). The upper section of the new screen displays BAS Channel Configuration options; the lower section displays BACnet Object Configuration options. Only the universal inputs must be configured in the upper portion of the screen. The channel identity is confirmed by the large channel tag near in the upper-right corner of the new screen. Clicking the ***Submit*** button registers your changes which become effective immediately. If you close the configuration screen without clicking the ***Submit*** button, ***your changes will be lost***.

The **BAS Channel Configuration** (upper) section of each configuration screen displays:

- ***Channel Type***—If more than one option is available, choose the desired type.
- ***Channel Number***—This *read-only* value confirms the selected channel.

The **BACnet Object Configuration** (lower) section of the screen displays:

- **Object Instance—** This is the **read-only** value automatically assigned for this channel.
- **Object Name**—assign the channel a **unique** name, using up to 63 characters.
- **Object Type**—This will match the selected **Channel Type** (see above) except for Virtual Points which must be either Analog Value or Binary Value.
- **Object Description—**Describe the device as you wish, using up to 63 characters.
- **Units**—Choose the appropriate unit from the list of standard BACnet units.
- **COV Increment**— Enter the amount of change (**0** for **any** change) at which a COV message will be sent to subscribers. (Ignored for binary objects.) You can subscribe to 14 binary and 2 analog channels. Additional subscription requests will be denied.
- **Submit** button—This will immediately apply your configuration.
- **Close** button— The window closes whether or not the configuration is saved.

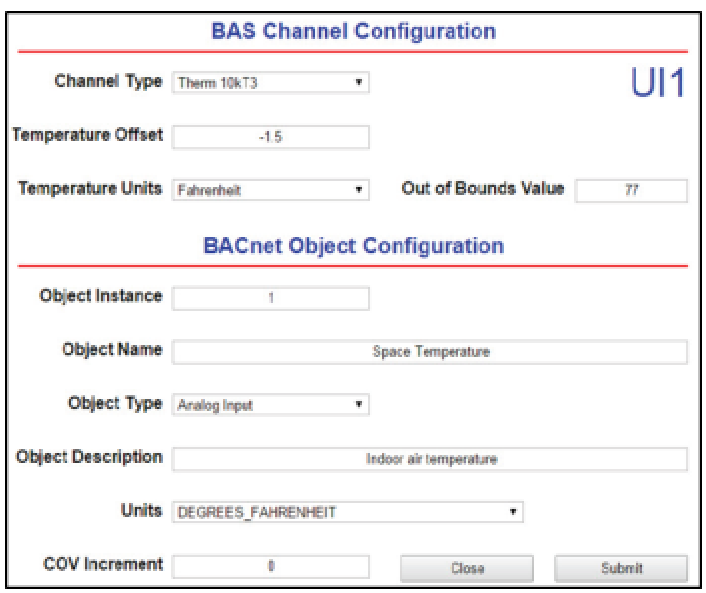


Figure 18 — Sample Configuration Screen

5.2.1 Universal Input — Configured as Analog Input (Channels UI1–UI8)

You can measure 0–10 V with UI1–UI8 as follows:

- Access the Main Page (Figure 13) and click a title link from among UI1–UI8.
- Under **BAS Channel Configuration** in the new page that appears, set the ***Channel Type*** to *Analog Input*. An example appears in (Figure 19).
- Under **BACnet Object Configuration**, the ***Units*** value defaults to *VOLTS*. Change if necessary.

Attach your device to the pair of BASC20 pins for the chosen channel — so

- that the more positive connection is to pin ***A*** and the more negative to pin ***C***.

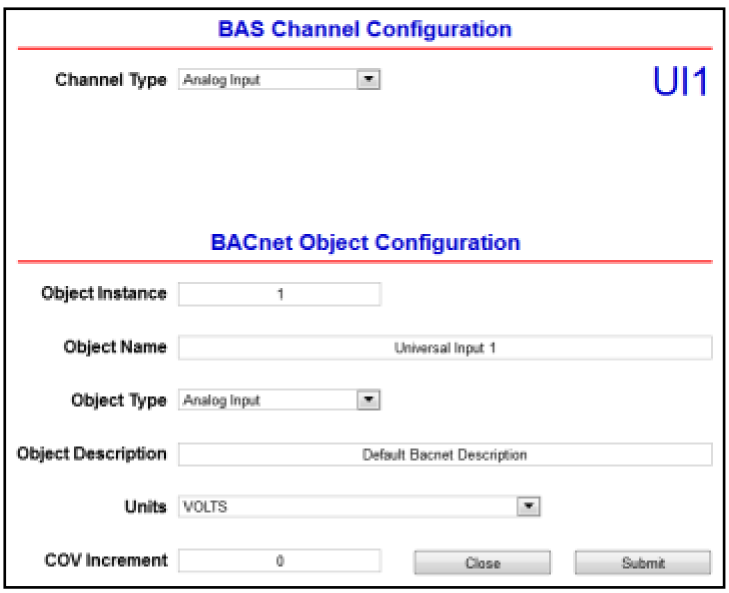


Figure 19 — Universal Input Configured as Analog Input

5.2.2 Universal Input — Configured as Binary Input (Channels UI1–UI8)

You can accept a binary input with any channel UI1–UI8 as follows:

- On the Main Page (Figure 13), click a title link from among UI1–UI8.
- Under **BAS Channel Configuration** in the new page that appears (Figure 20**)**, set the ***Channel Type*** to *Binary Input*.
- In the **BACnet Object Configuration** (lower) section of the screen, all items are as described in Section 5.2 above — but ***Units*** defaults to *NO_UNITS*.
- Attach your device to the pair of BASC20 pins for the chosen channel — so that the more positive connection is to pin **A** and the more negative to pin ***C***.

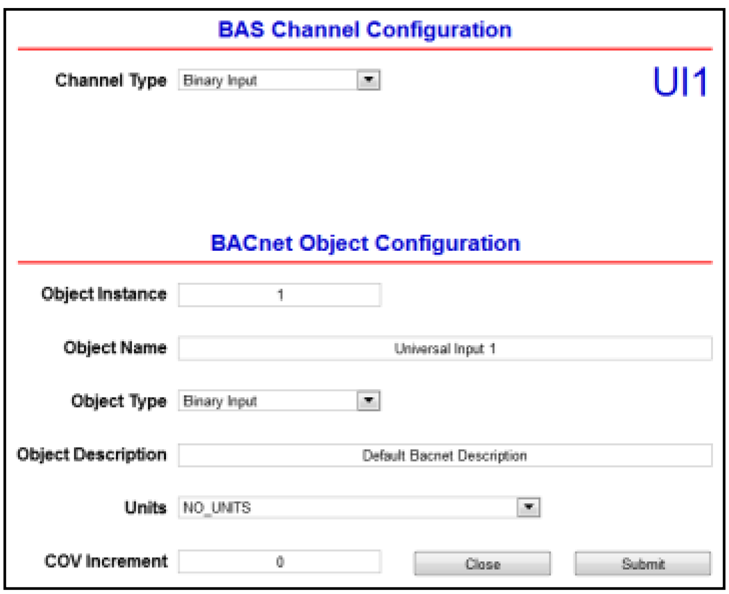


Figure 20 — Universal Input Configured as Binary Input

5.2.2 Universal Input — Configured as Pulse Input (Channels UI1–UI4)

Any channel UI1–UI14 can be a **Pulse Input** for pulse trains in the range of 0–40 Hz. You can accept a pulse input with any channel UI1–UI4 as follows:

- On the Main Page (Figure 13), click a title link from among UI1–UI4.
- Under **BAS Channel Configuration** in the new page that appears (Figure 21), set the ***Channel Type*** to *Pulse Input*. Additional fields will appear ...
- In the ***Maximum Value*** field, set the desired limit for the accumulated pulse count. It defaults to the absolute maximum of *16,777,215*. To **reset** the accumulator value to zero, set Reset = true in the universal input Sedona component.
- Set the ***Pull Up Resistor*** parameter to *Enabled*, if used with a passive device.

Note: In order The BAS ***Channel Type*** is *Pulse Input*, but the BACnet ***Object Type*** is *Analog Input*. This is because the BACnet object is an accumulator. ***Units*** can be changed from the default *NO_UNITS.*

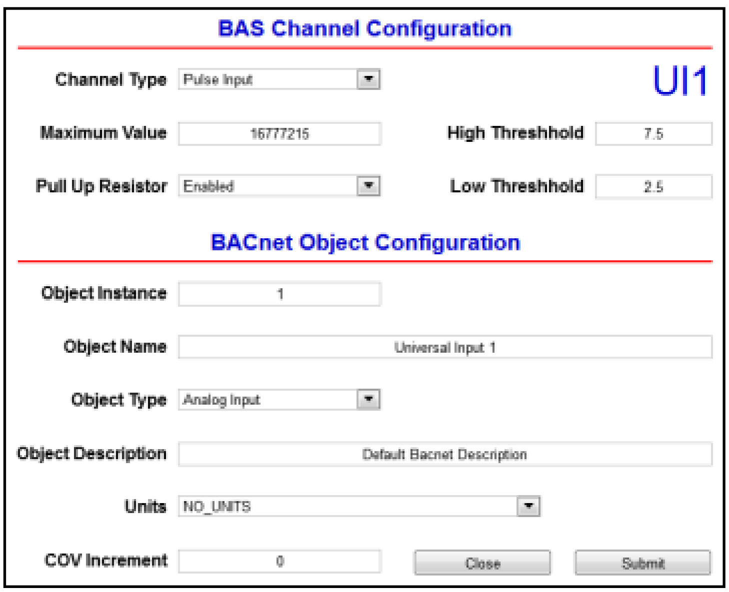


Figure 21 — Universal Input Configured as Pulse Input

5.2.4 Universal Input — Configured as Thermistor or Resistance Input (Channels UI1–UI8)

Channels UI1–UI8 can be used as Type II or Type III 10 kΩ **Thermistor** Inputs or a 20 kΩ **Thermistor** input or a **Resistance.** The channel BACnet type will be **Analog Input**.

You can accept a thermistor input with any channel UI1–UI8 as follows:

- On the Main Page (Figure 13), click a title link from among UI1–UI8.
- Under **BAS Channel Configuration** in the new page that appears (Figure 22 is an example of a Type III screen), set the ***Channel Type*** to *Therm 10kT2* or *Therm 10kT3 or Therm 20k*. Additional fields then appear ...
- The **Temperature Offset** parameter is only used as needed. If you determine that your thermistor yields an inaccurate result, enter a positive or negative offset value here to correct your thermistor reading.
- **Temperature Units** — the *Fahrenheit* default can be changed to *Celsius*. Note that the ***Units*** parameter under ***BACnet Object Configuration*** near the bottom of the screen automatically replicates your setting of the **Temperature Units** parameter.
- **Out of Bounds Value** — this is the temperature value you want assumed if an open thermistor condition occurs. A fault condition will be indicated in the universal input Sedona component.

You can accept a resistance input with any channel UI1–UI8 as follows:

- On the Main Page (Figure 13), click a title link from among UI1–UI8.
- Under **BAS Channel Configuration** in the new page that appears, set the ***Channel Type*** to *Resistance.* The ***Units*** field automatically selects *OHMS.*

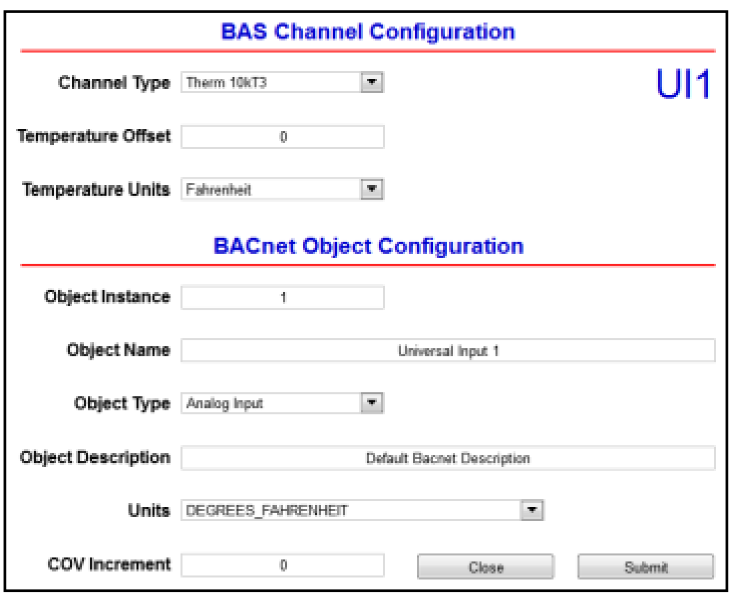


Figure 22 — Thermistor Input Configuration

5.2.5 Binary Inputs (Channels BI1–BI4)

You can accept a binary input with any channel BI1—BI4 as follows:

- On the Main Page (Figure 13), click a title link from among BI1-BI4.
- Under **BAS Channel Configuration** in the new page that appears (Figure 23), the ***Channel Type*** should be *Binary Input* by default.
- In the **BACnet Object Configuration** (lower) section of the screen, all items are as described in Section 5.2 above - but ***Units*** defaults to *NO_UNITS*.
- Attach your device to the pair of BASC20 pins for the chosen channel - so that the more positive connection is to pin ***A*** and the more negative to pin ***C***.

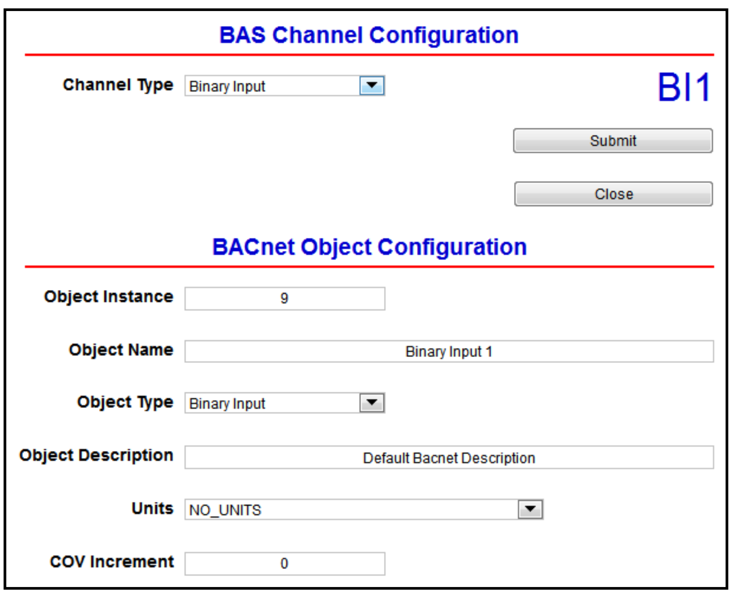


Figure 23 — Binary Input Configuration

5.2.6 Analog Outputs (Channels AO1–AO4)

Voltage in the range of 0–10 VDC (with up to 4 mA of current) can be outputted by assigning analog outputs. Configure an output using a web browser. For DC voltage, the output voltage is applied to point ***A*** with respect to ***C*** (common).

Any channel AO1–AO4 can be used to provide an analog voltage output. The BACnet type will be *Analog Output*. To configure an analog output:

- On the Main Page (Figure 13), click a title link from among AO1–AO4.
- Under **BAS Channel Configuration** (lower) section of the new screen that appears (Figure 24), the ***Channel Type*** will be *Analog Output* (read-only).
- In the **BACnet Object Configuration** (lower) section of the screen, all items are as described in Section 5.2 above — but ***Units*** defaults to *VOLTS*.
- Attach your device to the pair of BASC20 pins for the chosen channel — so that the more positive connection is to pin ***A*** and the more negative to pin ***C***.

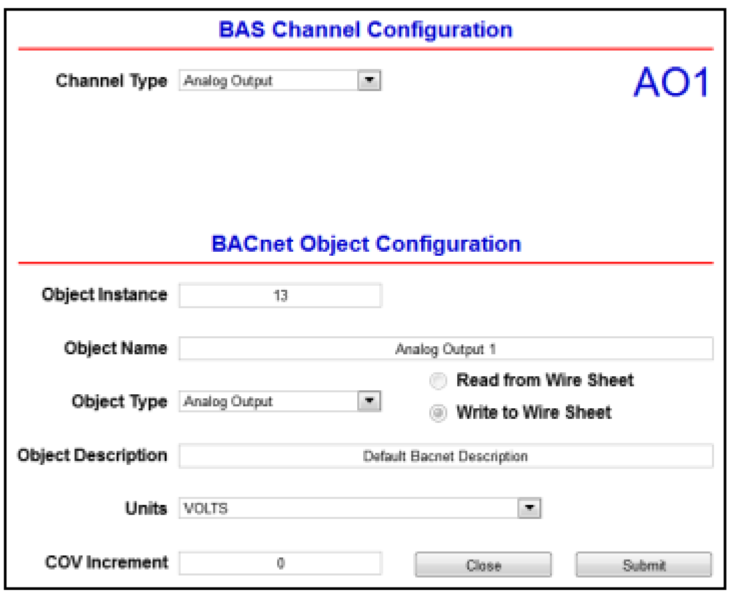


Figure 24 — Analog Output Configuration

5.2.7 Binary Outputs   (Channels BO1–BO4)

The BASC20 can provide four binary relay outputs (BASC-20R) or four triac outputs (BASC-20T). The voltage and current limits for relay units are 30 VAC/VDC and 2 A. For triac units the limits are 30 VAC and 0.5 A. Violating these limits could damage the BASC20 and void the warranty.

Relay channels can be used as contact closures for other devices, but triac channels can only be used to enable or restrict the flow of AC current. It is common for the BASC20 binary outputs to enable the coil of interposing relays which can carry larger currents and support switching higher voltages.

Any channel BO1–BO4 can be used to provide a binary output. The BACnet type will be *Analog Output*. To configure an analog output:

- On the Main Page (Figure 13), click a title link from among BO1–BO4.
- Under **BAS Channel Configuration** (lower) section of the new screen that appears (Figure 25), the ***Channel Type*** will be *Binary Output* (read-only).
- In the **BACnet Object Configuration** (lower) section of the screen, all items are as described in Section 5.2 above. ***Units*** will default to *NO_UNITS*.
- Attach your device to the pair of BASC20 pins for the chosen channel — so that the more positive connection is to pin ***A*** and the more negative to pin ***B***.

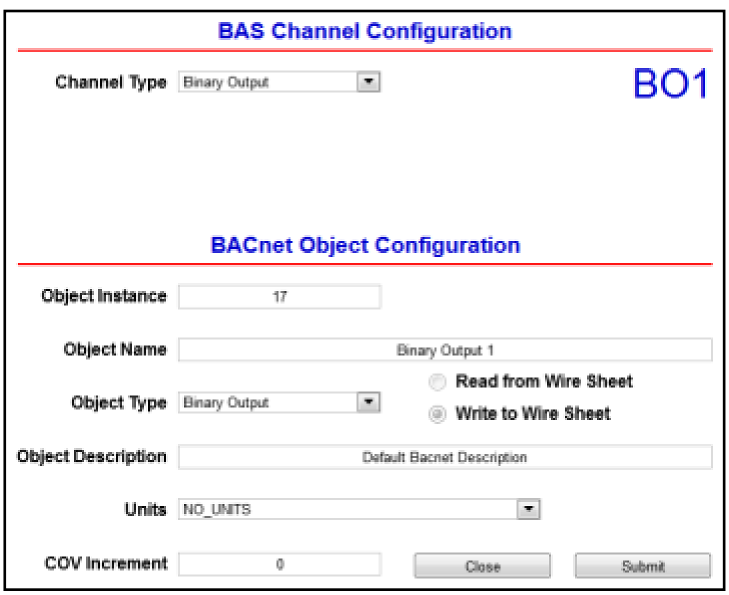


Figure 25 — Binary Output Configuration

5.2.8 Virtual Points (Channels VT01–VT24)

In the **CControls_BASC20_IO** kit are 24 virtual point components (VT01−VT24) that are used by a BACnet client to send and receive intermediate data to and from the BASC20. By intermediate data we mean that the data is neither real input data nor real output data but something in between real inputs and real outputs. It could be setpoint or reset data intended for the wire sheet or calculated or status information generated by the wire sheet. Although BACnet allows for the reading of the BASC20 real input and output points — and under certain conditions the writing of real output points — virtual points have no reading or writing restrictions. Virtual points are treated by BACnet as either a binary variable (BV) or analog variable (AV) while real points appear as binary inputs (BI), analog inputs (AI), binary outputs (BO) or analog outputs (AO). The BASC20 logic engine reads the state of its inputs (AI and BI) and outputs (AO and BO), executes logic, and then sets outputs (AO and BO) accordingly. In a similar manner, a BACnet client can "read" the BASC20's real inputs and will attempt to "write" to the BASC20's real outputs. AVs and BVs are a bit different in that they can be configured to be either an input to the BACnet client (read) or an output from the BACnet client (write). Therefore, we need to establish rules for the use of AVs and BVs.

If a BACnet client is to read intermediate data from the Sedona wire sheet, this is no different from accessing data from an input component on the wire sheet. We would call this "reading from the wire sheet" or Wire Sheet Read. The VT on the wire sheet would have a channel type (Chn Type) of "float out" or "binary out." Configuring the VT for wire sheet read or a wire sheet write requires the Workbench tool.

If a BACnet client is to write intermediate data to the Sedona wire sheet, this is no different from logic on the wire sheet writing to an output component. We would call this "writing to the wire sheet" or Wire Sheet Write. The VT on the wire sheet would have a channel type (Chn Type) of "float in" or "binary in."

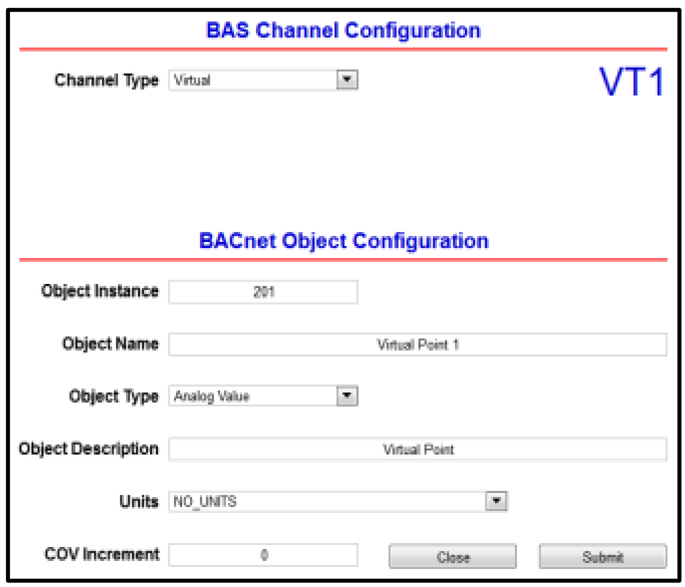


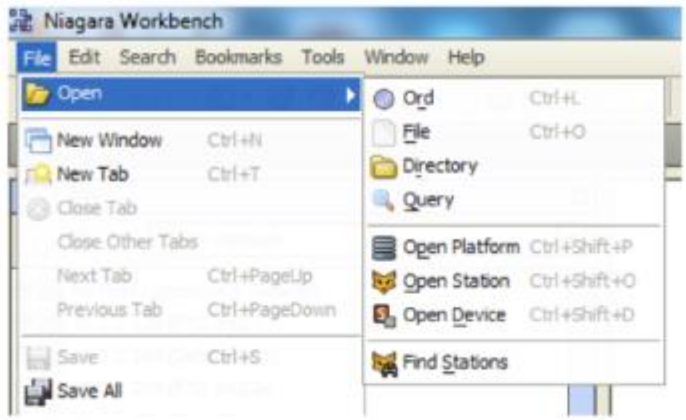
Figure 26 — Virtual Configuration Screen

Like universal inputs, virtual points are configured via a web page that is accessible from the main web page. Click on the title link of a particular virtual point to gain access to its configuration page. From the **Object Type** parameter under ***BACnet Object Configuration***, select either Analog Variable or Binary Variable. Enter a unique **Object Name** and enter an **Object Description** or change the **Units.** Notice that the radio button *Read* or *Write* from the Wire Sheet reflect that which was set by the Workbench Tool.

Upon power loss, the first eight virtual components are retentive up to seven days. This allows a BACnet client command to be retained even if power is lost to the controller. Backup is accomplished using a super-cap.

A.1 Using Workbench as a Sedona Tool

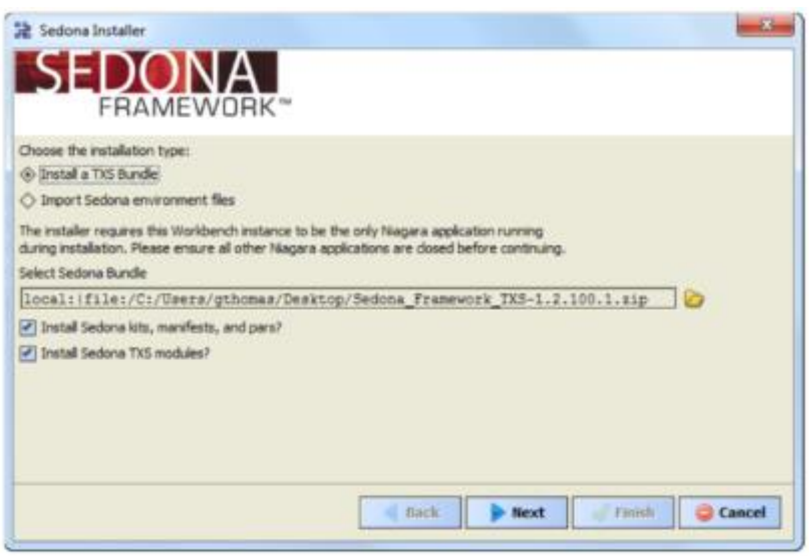
For those who have access to Niagara Workbench, this programming tool for Niagara Framework works well as a Sedona Tool when programming devices built on the Sedona Framework. Niagara Workbench is available from Tridium or from a Tridium OEM. It can be called by several different names such as Workplace or ProBuilder but we will use the generic term Workbench to mean Niagara Workbench with Sedona installed. Workbench does not come from the factory with Sedona installed but it can be easily updated for Sedona on Workbench versions 3.7.x or 3.8.x. The discussion that follows assumes a basic understanding of Niagara Workbench by the user. Keep in mind that Niagara Workbench is a complex tool because it was originally developed for Niagara Framework use. There are many features in the program that are not applicable to Sedona Framework so they will not be discussed.



Installing Sedona into Workbench
After starting Workbench, click on **File > Open** and see if you have an option called *Open Device.* If it is there, Sedona is installed and you can skip this section and go to the section on installing component bundles. If you do not see Open Device you need to install Sedona Framework into Workbench.

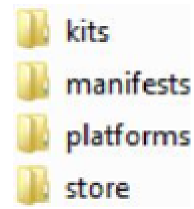
Go to the Contemporary Controls' web site and click on **Support > Product Support Materials > Sedona** and download the Sedona Framework

TXS Bundle for either Workbench 3.7 or 3.8 to match the Workbench version you have. The Workbench version is clearly marked on its welcome screen. Download the bundle and put it on your desktop for convenience but leave it zipped. Go back to Workbench and click **Tools>Sedona Installer** and you will see the following screen. Accept the default settings. Click the file icon to browse for your file. Click *Next*.



After you click *Next* you might receive a message about Module Downgrade. Just ignore the message by clicking *Yes*. Click *Finish* and Sedona will be installed.

Installing the Component Bundle



When you install Sedona in Workbench you will gain a sub folder called *sedona* within the Niagara directory. It can typically be found in the Windows' root directory at **Niagara>Niagara 3.8**. If you click on *sedona* you will see four folders – *kits, manifests, platforms and store.* The first three folders store information about the personality of Sedona devices while the fourth folder is where Sedona applications and Sedona device information is backed up. When Sedona is installed on the Workbench tool, there is a set of Sedona release 1.2 components that will populate the first three folders. Components are organized in meaningful module groups and deployed as kits. These component kits come from Tridium and are hardware independent in that they will run on any Sedona 1.2 device. For example, *And2* and *Or2* are Boolean logic components which can be found in the *Logic Kit* from Tridium.

However, Contemporary Controls has developed component kits specific to the Sedona platforms it developed and these must also be installed. These kits are designated by vendor, product name and module type such as *CControls_BASC20_IO*. In addition, Contemporary Controls has developed hardware-independent component kits that would be beneficial to the Sedona community and these should also be installed. These types of kits are identified by vendor and module type such as *CControls_Function*. Collectively, these kits are provided in a *component bundle* and labelled with a product identifier and a bundle number. As more components and kits are developed, they are added to the bundle and the revision number of the bundle is incremented. No components or kits are ever removed so that installing the latest bundle does not cause harm. For the BAScontrol series, the bundle would have a name such as *Component_Bundle_BASC_1.0.19*. The latest bundle can be found on the same web page were the TXS bundles were found. Like the TXS bundles, the Component bundles are zip files that should be left unzipped for installation. Use the same method for installing the component bundles as was done with the TXS bundles. The Sedona Installer in Workbench will then add those kits, manifests and platforms in the appropriate folders if they do not exist already. Once this is done you can access a Sedona device.

Accessing a Sedona Device

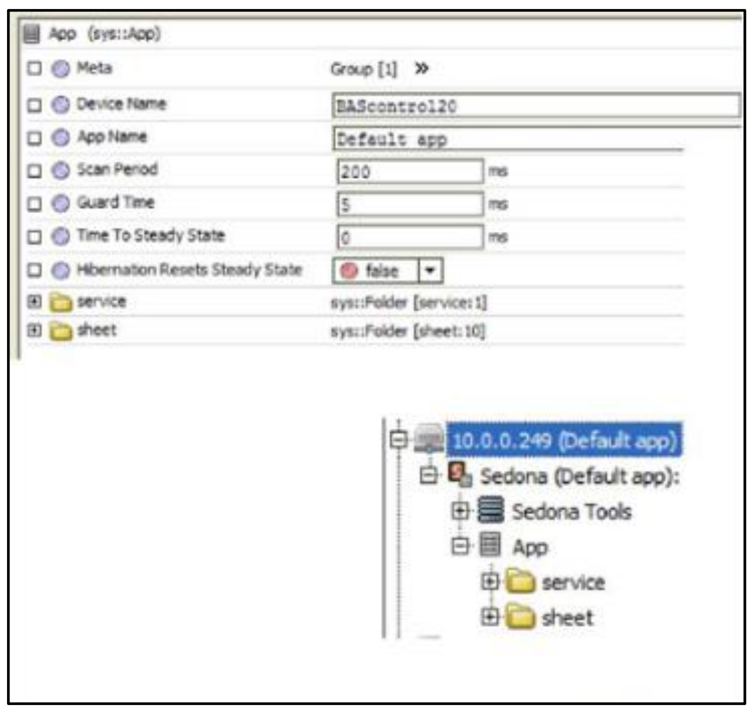
For the Sedona device we will use a controller in the BAScontrol series (BASC). Like other Sedona devices, the BASC is IP-based so we need Workbench to be on the same sub-net as the BASC. In this example we have the BASC addressed at 10.0.0.249. Using a web browser we can try to access this IP address. If we obtain an Authentication Request from this controller, we are assured we are on the right sub-net. We can enter credentials for this controller to view the main web page or we could just close our web browser and bring up Workbench.

At the Workbench home screen, click **File > Open > Open Device** and while accepting the default settings enter the IP address of the controller and then click OK. If you cannot find Open Device, Sedona is not installed.

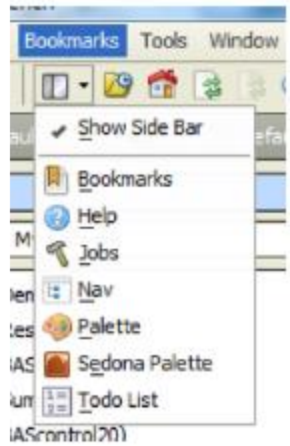
Next you will be prompted for credentials. The default credentials are *admin/admin*. You can click on *Remember these credentials* if you which. Click Ok.

If you are successful you will see a reference in the main window for Sedona Tools and App. Click on **App**.

The application property sheet should appear. The default **Device Name** would be the *product name*. This can be changed. The default **App Name** is simply *Default app* and this can be changed as well. The **Scan Period** indicates how often Sedona logic is solved. Although Sedona can execute wire sheets in less than 100 ms, time must be left for the controller to do other background tasks including updating web pages. It is best to leave this setting at 200 ms. Leave the other settings at their default value.

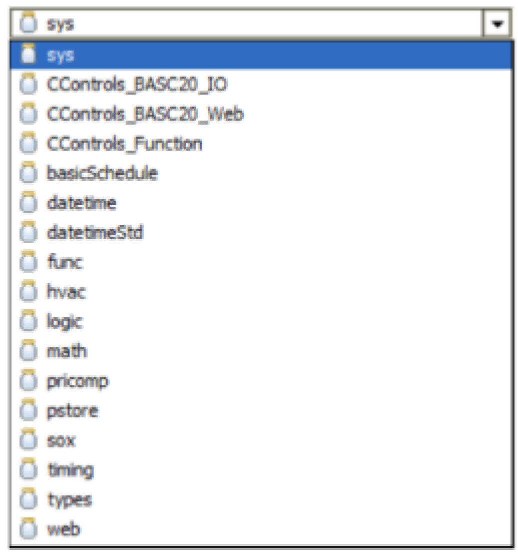


There are two ways to reach the wire sheet. The first is just to click on the **sheet** folder in the App property sheet. The second is to go to the Navigation pane and expand the navigation tree for the controller being accessed. By clicking on **sheet**, you should be able to see the main wire sheet although the default wire sheet is blank.

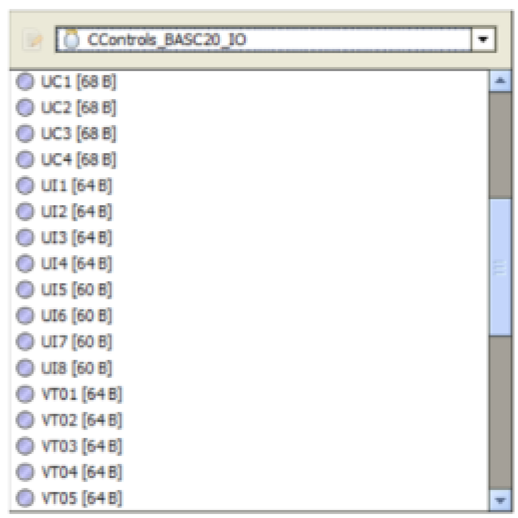


Now that you have opened up the wire sheet you should see the Sedona Palette just below the Navigation pane. If it is not there, go to the Side Bars icon just below the word Bookmarks and click on the drop-down menu. Select Sedona Palette and it will appear.

Accessing the Sedona Palette



Once the Sedona Palette is viewable you can see all the component kits that reside on the connected Sedona device. Each kit is represented by the jar icon. Click on the drop-down menu to see all the kits. The Tridium 1.2 release kits carry no vendor name while custom kits do such as the Contemporary Controls' IO, Web and Function kits. If they also carry a product designation, these kits are hardware dependent and not portable to another Sedona device.



Using the drop-down, select the CControls_BASC20_IO kit. In the case of the BAScontro20 there are 49 components to choose from – 20 real points, 24 virtual points, a scan timer and 4 retentive universal counters. All are intended for one time use and only those dragged onto the wire sheet will become part of the logic of the controller.

Using the Sedona Tools within Workbench

If you go into the navigation pane and expand the IP address of the Sedona device you can access the three Sedona Tools. The three Sedona tools along with what service they can provide are listed on the right. You will notice that at the header is the name of the Sedona application running on the attached controller. This way you can confirm that the controller is executing the application that is of interest.

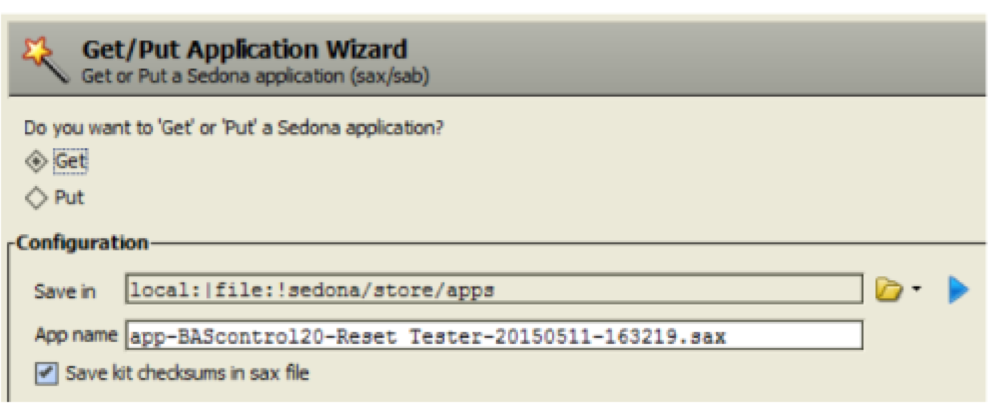
Sedona (Reset Tester):	
Name	Description
Kit Manager	Manage kits on the Sedona device
Application Manager	Get or Put a Sedona application (sax/sab)
Backup/Restore Tool	Backup or restore a Sedona device

Directly below the list of tools is a list of kits that are installed on the controller. This information comes from the *Schema* read from the installed *app.sab* file on the controller. Notice that at the top of the list is platform information that comes from the controller. A checksum accompanies each kit. Having a kit does not necessarily means that components in the kit are being used. It just means that the controller can support all of the components from that kit.

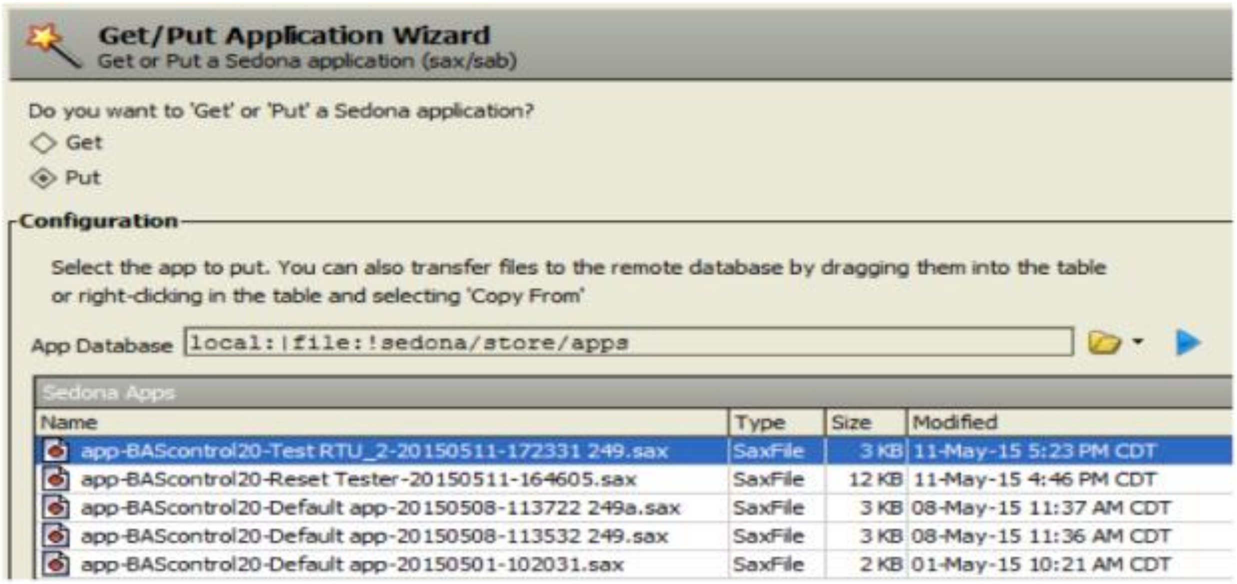
Schema (ccontrols-BASC20-Platform-1.2.28)		
Name	Version	Checksum
sys	1.2.28	d3984c51
CControls_BASC20_IO	1.2.28	71eea5c5
CControls_BASC20_Platform	1.2.28	991d038
CControls_BASC20_Web	1.2.28	6232c744
basicSchedule	1.2.28	7fdca638
datetime	1.2.28	3a280dce
datetimeStd	1.2.28	fc5628d7
func	1.2.28	821b7396
hvac	1.2.28	7264c67c
inet	1.2.28	25648ba7
logic	1.2.28	9fe95ce1
math	1.2.28	c22b255c
pricomp	1.2.28	b5cd6698
pstore	1.2.28	7ea2cb06
sox	1.2.28	397a84dd
timing	1.2.28	aeaac82a
types	1.2.28	10936551
web	1.2.28	462d43e

Using the Application Manager

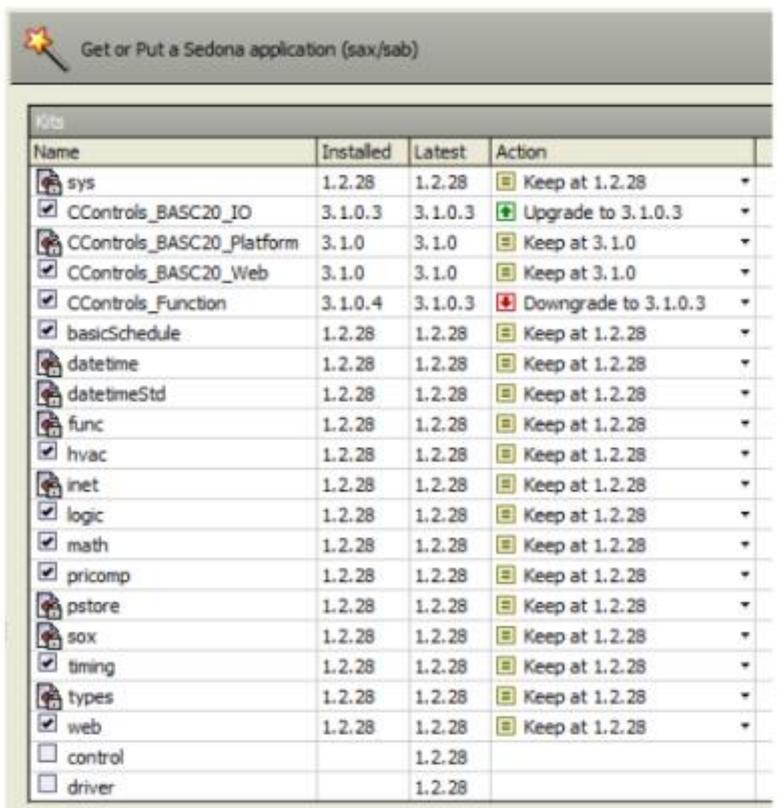
By clicking on Application Manager you can either save or restore the application which includes all the wire sheet information. A *Get* captures the app.sab from the controller, converts it to an *app.sax,* and stores it where you want it while offering you a suggested file name and location. If you want you can append the last quad of the IP address of the controller to the file name if you have several controllers running the same application. This way you can easily locate the controller you just backed up. Accept the default checkbox for saving the kit checksums, click **Next**, then **Finish** and then **Close** and you will have a copy of your application on your computer that is running Workbench. Saving the app.sab file is quick and easy but it only saves the app.sab and nothing about web page configurations and BACnet information.



To restore an app.sab file onto the controller you will need to do a Put.



You will be presented with a choice of files. You can click on the Modified column to arrange the files by date to help you search for the one you want. Once you highlight it, click Next at the bottom of the screen.

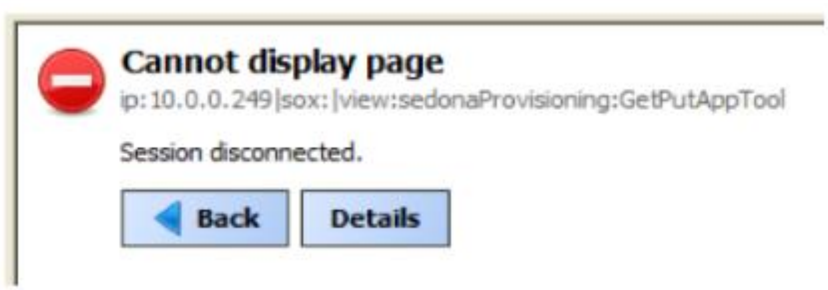


You will be presented with a list of kits along with the ability to make kit changes. The kits installed in the controller will have either a check mark or an icon indicating that it is being used in the application. The installed version is then listed. The column called Latest identifies the version number available in the Component Bundle that was installed in Workbench. If a version difference is noted, then you have the option to Keep, Upgrade or Downgrade. It is best to ignore these options and just click Next.

If a kit is not found in Workbench, or a kit is present but with a different checksum, it will be necessary to upgrade you Component Bundle. Contemporary Controls only adds kits to Component Bundles and does not remove old kits. Therefore it is safe to install the latest bundle. Complete the operation by clicking Next and then Finish. You can observe the progress of the restore operation on the subsequent screen.

Once the restore is completed you will be prompted to *Restart* the Device. Go ahead and do that.

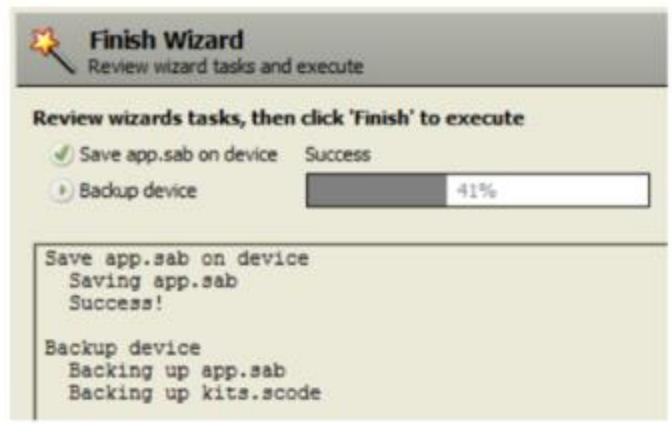
You will notice that you will lose connection to the controller. Just wait until the controller is finished restarting and then log into the controller again.



Using the Backup/Restore Tool

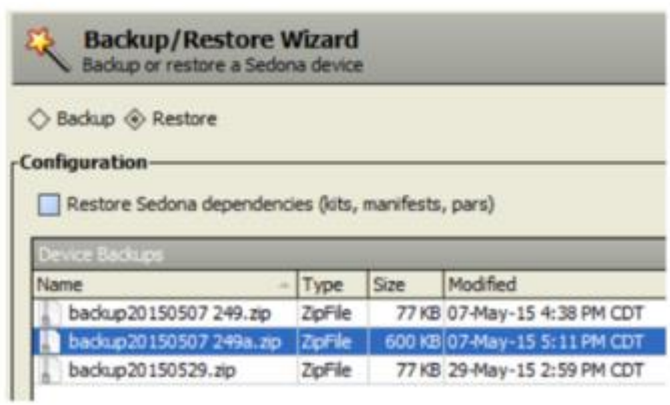
The second tool is the Backup/Restore tool which should not be confused with Contemporary Controls' Sedona Backup and Restore Utility. The former only backs up applications while the latter backs up the complete Contemporary Controls' controller project including BACnet configuration and web pages. The main difference between the Backup/Restore tool and the Application Manager tool is that the kits.scode file is also saved during the backup process and is put back during the restore process. Backing up the kits.scode file takes much more time.

Clicking on the Backup/Restore option gains you a screen asking forselections. Do not ask to have the Sedona VM backed up and it is usually not necessary to backup dependencies. However, you should leave the box checked for backing up the app.sab and the *kits.scode*. Notice that you will be saving everything in one zip file. Change the name or append the controller number if you wish. Also notice that you will not be generating an app.sax like you did when using the Application Manager. Click *Next* and then *Finish* to complete the process. You can observe the backup process with the following screen. Once the process is completed by announcing Finished, click *Close.*



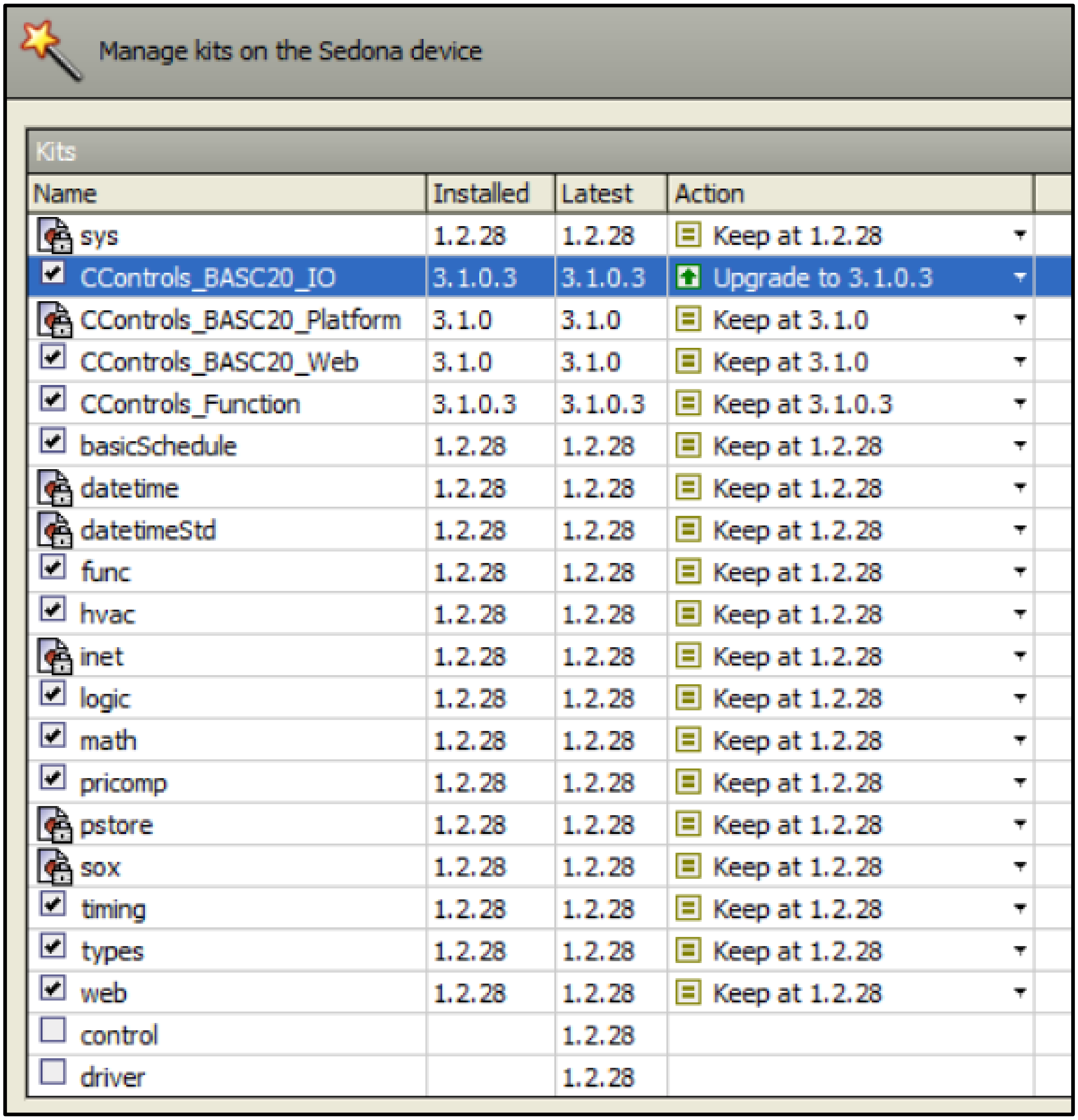
To do a restore, select the Restore option and you will be presented with a choice of files. Select the one you want and click Next and Finish. It is not necessary to restore dependencies.

Once the process is complete you will be prompted to Restart the device. Do so and then wait until the controller restarts before accessing it again with Workbench*.*



Using the Kit Manager

The final Sedona tool in Workbench is the Kit Manager. The Kit Manager allows you to generate a proper *kits.scode* file based upon the kits you select. The Kit Manager firsts compares the kits that are installed on the controller with those available on Workbench. If different versions of kits exist, then you are given the option to *Keep*, *Upgrade* or *Downgrade* that particular kit*.* Once selections are made, the Kit Manager uses the *app.sax* version of the installed *app.sab* and generates a new *kits.scode* for use in the controller. It is highly recommended to use the Kit Manager only at the direction of Contemporary Controls' technical support. Consider this tool as only necessary for invoking advanced features of Sedona Framework.



Manage kits on the Sedona device

Kits

Name	Installed	Latest	Action
sys	1.2.28	1.2.28	Keep at 1.2.28
CControls_BASC20_IO	3.1.0.3	3.1.0.3	Upgrade to 3.1.0.3
CControls_BASC20_Platform	3.1.0	3.1.0	Keep at 3.1.0
CControls_BASC20_Web	3.1.0	3.1.0	Keep at 3.1.0
CControls_Function	3.1.0.3	3.1.0.3	Keep at 3.1.0.3
basicSchedule	1.2.28	1.2.28	Keep at 1.2.28
datetime	1.2.28	1.2.28	Keep at 1.2.28
datetimeStd	1.2.28	1.2.28	Keep at 1.2.28
func	1.2.28	1.2.28	Keep at 1.2.28
hvac	1.2.28	1.2.28	Keep at 1.2.28
inet	1.2.28	1.2.28	Keep at 1.2.28
logic	1.2.28	1.2.28	Keep at 1.2.28
math	1.2.28	1.2.28	Keep at 1.2.28
pricomp	1.2.28	1.2.28	Keep at 1.2.28
pstore	1.2.28	1.2.28	Keep at 1.2.28
sox	1.2.28	1.2.28	Keep at 1.2.28
timing	1.2.28	1.2.28	Keep at 1.2.28
types	1.2.28	1.2.28	Keep at 1.2.28
web	1.2.28	1.2.28	Keep at 1.2.28
control		1.2.28	
driver		1.2.28	

B.1 Using the Sedona Project Backup and Restore Utility

Introduction

The Sedona Project Backup and Restore utility program (BASbackup) provides a convenient way of saving and restoring a BAScontrol project to a desktop or laptop computer. With the BAScontrol, a project consists of the following files:

- Sedona Framework binary application file (app.sab.target)
- Sedona Framework source application file (app.sax)
- BAScontrol configuration file (bas_cfg.xml)
- BAScontrol Scode file (kits.scode)
- BAScontrol web component file (webc.xml)

These files are sufficient to completely backup a BAScontrol project without the need of a Workbench tool. Although Sedona files can be backed up and restored using the Workbench tool using the Get and Put commands, the BAScontrol configuration and web component configurations are ignored. Therefore, the Sedona Project Backup and Restore utility is the best and simplest way to store BAScontrol projects. The BASbackup program is applicable to BAScontrol20 firmware 3.0.28 and later or BAScontrol22 firmware 3.1.0 and later.

Installation

BASbackup is a Java program (version 7_51 or later) and is intended to run on a Linux, Windows© 7 or later computer. The latest version can be downloaded as a zip file (BASbackup Utility version 1.0.x) from our website at:

http://www.ccontrols.com/support/bascontrol20.htm

Place the downloaded file anywhere on your computer — we suggest your desktop for unzipping. Extract the files from the zip file and choose a location that would be convenient for you. We suggest the root directory of your main drive (for example C:). Make a folder called BASbackup during the extracting process and extract the contents of the zip file into this folder. You will see the following files:


BASbackup


Component _ Bundle _ BASC_1.0.x

The Java utility is BASbackup and for convenience a Component Bundle for the latest version of the BAScontrol is provided. The Component Bundle provides kit information on the component types that can be used with the BAScontrol. You can always download later versions of BAScontrol bundles if necessary from the same web site page. Simply replace the current bundle with the latest if a later BAScontrol version is to be supported. Later bundle versions include support for both current and previous versions of the BAScontrol so having the latest poses no harm. Leave the Component Bundle zipped.

Double-click the BASbackup icon and the program will open up as shown below with a default IP address of a factory-set BAScontrol. A configuration file for the program along with a backups folder will be created in the same BASbackup folder. The backups folder will be the location of your backup files. The Unit Status will indicate UNKNOWN until a connection is made to a BAScontrol.

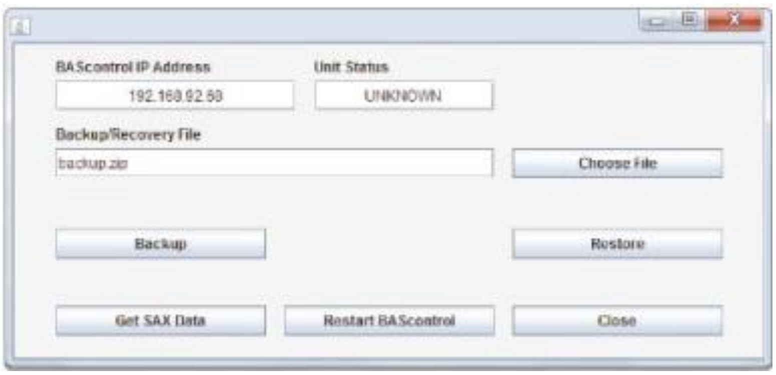


Figure 1 — The Default BASbackup Screen

Backing up a Sedona Project

Enter the correct BAScontrol IP Address of the BAScontrol that is to be backed up if it has been changed from its default address. This utility program will attempt to communicate with a device at the address entered. Make sure your computer is on the same subnet as the target controller. The target controller must be powered with Sedona enabled and accessible. The utility provides you a default backup file location. If you want something different you can edit the location on the screen. It is not necessary that the file exists. The utility will create the file you named and append the zip extension as well. If you have a previous file which you want to overwrite, you can use the *Choose File* button to find it. The utility will first look for it in the backups folder but you can use the navigation buttons if the file location is somewhere else on your computer. Once you are ready, click *Backup*.

Figure 2 — The Authentication Screen

With the correct BAScontrol IP Address entered, click the Backup button and a credentials window will appear from the targeted controller. Enter the User Name and Password of the Sedona Framework application in the BAScontrol20. The default credentials are admin/admin. The User Name and Password of the web pages are unnecessary and cannot be used.

Figure 3 — Specifying a New IP Address and Backup File Name

In our example and as shown in Figure 3, we changed the IP address and renamed the backup file. If you are successful no error messages will appear. Once the red progress messages cease on the main page you can look at the contents of your backup zip file by double-clicking it in the backups folder. You should see the five files that were identified earlier.

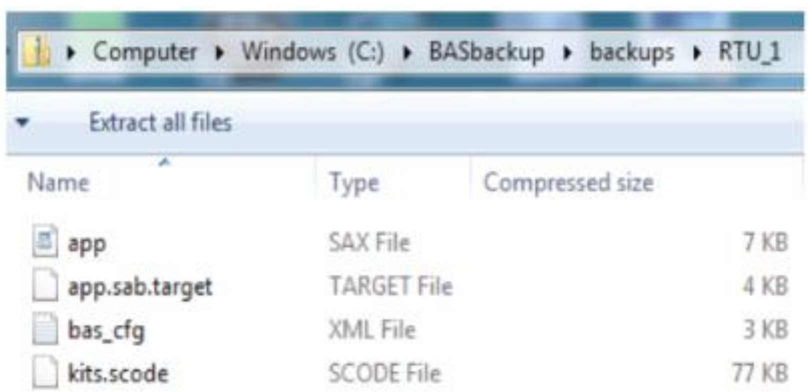


Figure 4 — Content of the RTU_1 Backup Zip File

Restoring a Sedona Project

Restoring a Sedona Project is just as easy. Before you begin the restore process, set the IP address on the main page to the location of the controller to be restored. The controller must be powered up and accessible. You will need to select the appropriate backup file by first clicking *Choose File* on the main page and selecting the file.

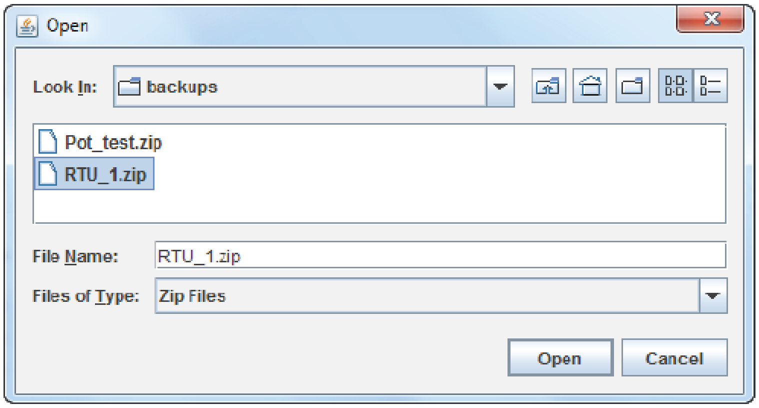


Figure 5 — Choosing a Restore File

In this example we are going to restore RTU_1.zip to controller 10.0.0.61. Once you have the correct backup file and IP address as shown in Figure 6, click the Restore button on the main page to get the screen of Figure 7.

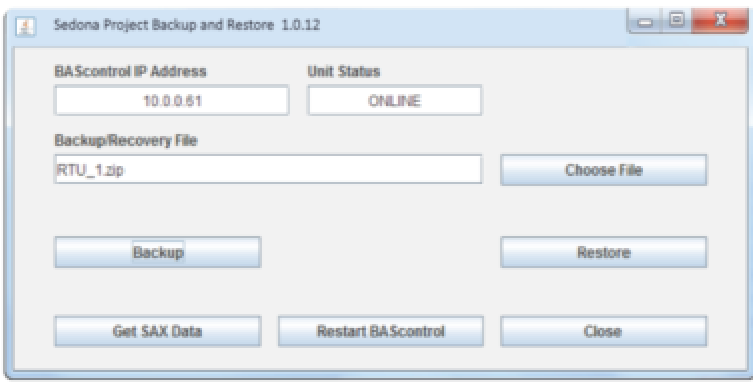


Figure 6 — Restoring to 10.0.0.61

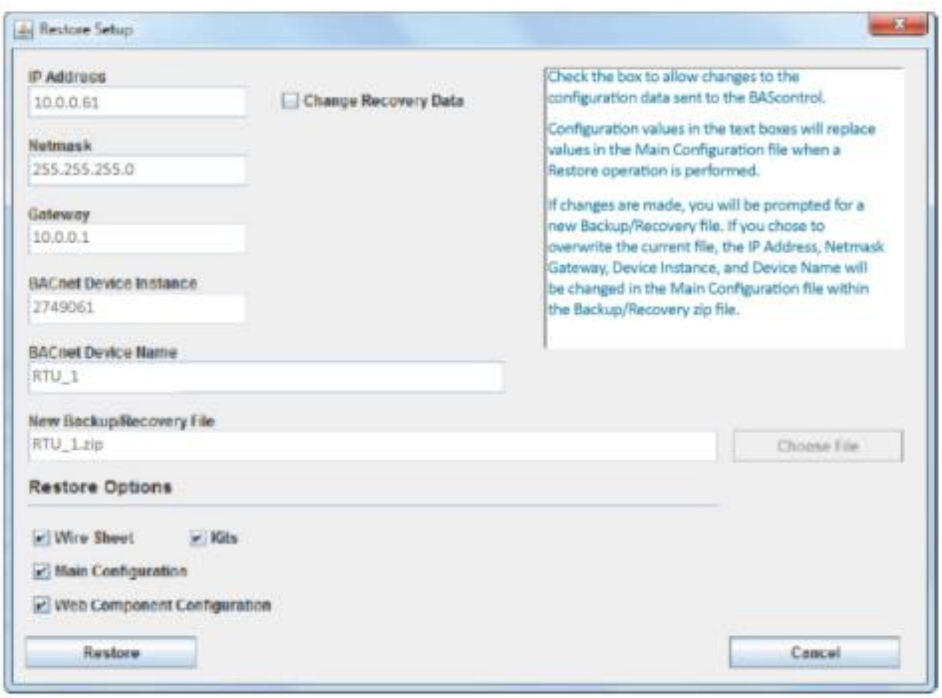


Figure 7 — Restore Setup Screen

First, notice the location of the Backup/Recovery file. Is this the file you want restored? Second, look at the top of the screen and you will see the following parameters:

- IP Address
- Netmask
- Gateway
- BACnet Device Instance
- BACnet Device Name

These five parameters will be loaded into the controller being restored. Are these five parameters that come from the saved zip file what you want? Is the IP address of the target controller the same as shown on this page? If so, study the *Restore Options*.

There are some options available before restoring a controller. Under the *Restore Options* there are four boxes:

- Wire Sheet
- Main Configuration
- Web Component Configuration
- Kits

By un-checking boxes you can control what individual files are being restored. Consider this an advanced feature because under normal conditions you want to make sure that all files you saved in the project backup file are going to be reflected in the restored controller.

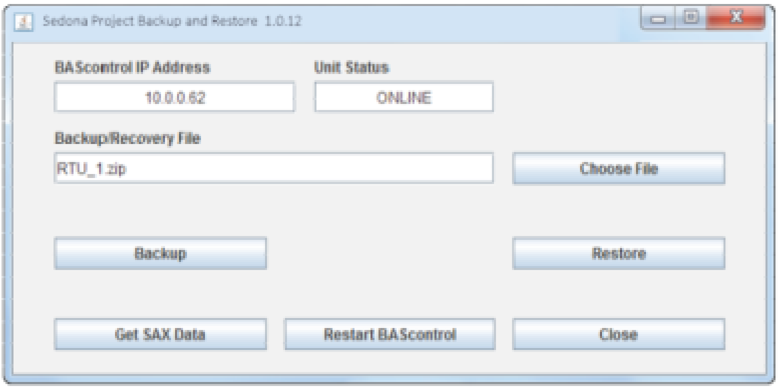
Once you are convinced the settings are correct, click the *Restore* button.

Once the Restore operation is completed, evidenced by the completion of progress messages, the application program in the target controller continues to run the old program. In order to run the newly restored program, a controller Restart is required. You will be prompted for an immediate restart. You can do it now or later.

If you want to restart the controller later, this can be accomplished by clicking the *Restart BAScontrol* button on the BASbackup main page or by restarting from the controller's web page. Regardless of what method is used, care should be exercised when restarting the program on an active controller. The controller and application should be in sight of the systems integrator initiating the restart to confirm a safely functioning restart of the application.

Cloning a Sedona Project

It is also possible to direct a saved program to a different controller that needs to run the identical saved program. However, the IP address, netmask and gateway address stored in the saved backup file need to reflect the target controller otherwise there will be an IP conflict. In addition, the BACnet Device Instance and BACnet Device Name must be unique so it must be changed from what is in the stored in the saved file.



On the BASbackup main page (Figure 8), set the IP address of the target controller and choose the backup file that is to be used for cloning. In this example, we are going to use RTU_1.zip for cloning to the 10.0.0.62 controller. Once the information is correct click, *Restore* to get you to Figure 9.

Figure 8 — Specifying the IP Address and the Backup ZIP File for Cloning

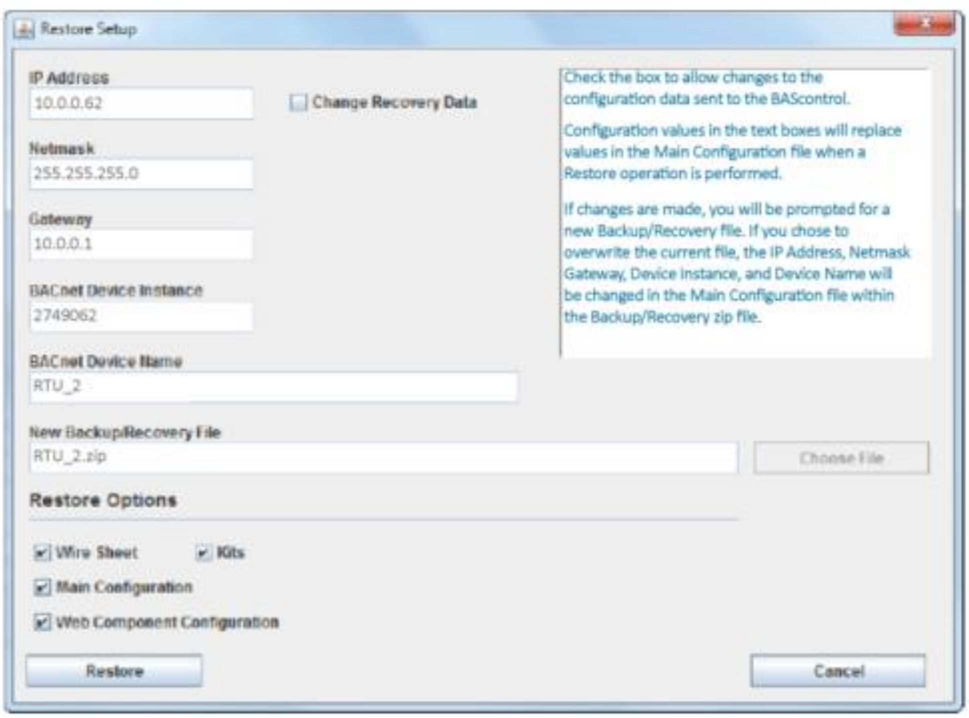


Figure 9 — Restore Setup for Cloning

On the Restore Setup page, check the box entitled *Change Recoverey Data.* Now you are able to change the settings of the five parameters to suit the target contoller. Enter the *IP* address of the target controller. The *Netmask* and *Gateway* probably do not need to change. The *BACnet Device Instance* should be changed to something unique as should the *BACnet Device Name.*

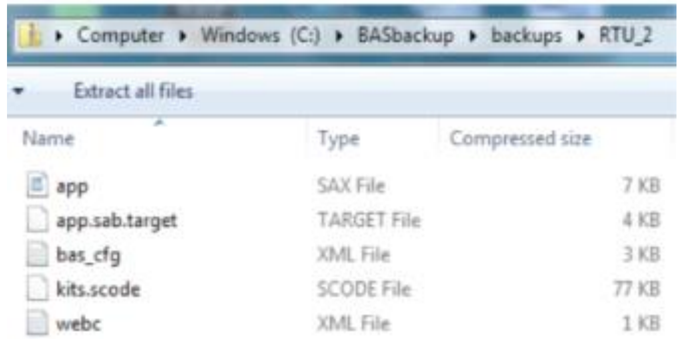


Figure 10 — Content of the RTU_2 BackupZip File

Since the target controller is for RTU_2 we will use this name while changing the *BACnet Device Instance*. It is also a good idea to save the cloned controller settings to a new zip folder for easy recovery. In this way, we will have a unique backup file for every controller on the job. In our example, the target controller is RTU_2 so that is what we are going to enter for the new backup program we are going to generate. Enter the new backup file name. Since we want all types of files saved, we will leave the four Restore Option boxes checked.

This is important to remember. The IP address on the Restore Setup screen will become the controller's IP address once the controller is restarted. So it is possible to send the Restore Setup data to a controller with a different IP address than the one indicated on the Restore Setup screen. The file is always sent to the IP address indicated on the BASbackup main page. However, once the controller is restarted it will assume the IP address indicated on the Restore Setup screen. This could be very handy when you have several BAScontrols at factory-default IP addresses that are to be commissioned to specific IP addresses in the field using a common program. Just make sure you provide unique BACnet references and IP addresses for each controller.

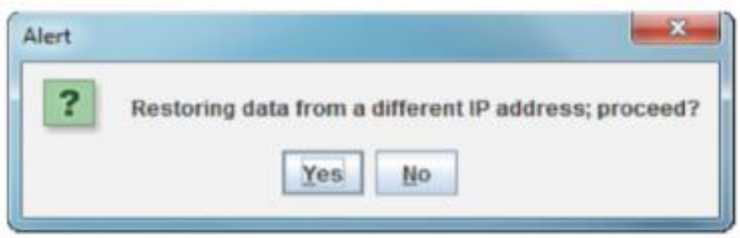


Figure 11 — Alert message when IP addresses differ in the restore operation

If you have different IP addresses on the Restore Setup screen and the BASbackup main page, you will receive an Alert message asking if you want to proceed (Figure 11). Click *Restore*.

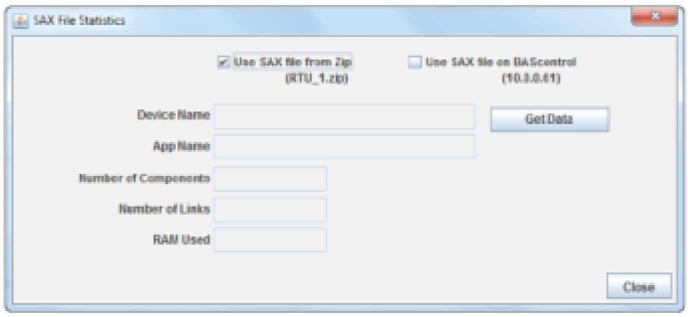
Once the Restore operation is completed, the application program in the target controller continues to run the old program. In order to run the newly restored program, a controller Restart is required. You will be prompted for an immediate restart. You can do it now or later.

If you want to restart the controller later, this can be accomplished by clicking the *Restart BAScontrol* button on the BASbackup main page or restarting from the controller's web page. Regardless of what method is used, care should be exercised when restarting the program on an active controller. The controller and application should be in sight of the systems integrator initiating the restart to confirm a safely functioning restart of the application.

When you are finished, you will have cloned a controller in the field but configured it appropriately in terms of IP address and BACnet settings. You also have created a new backup file for project storage.

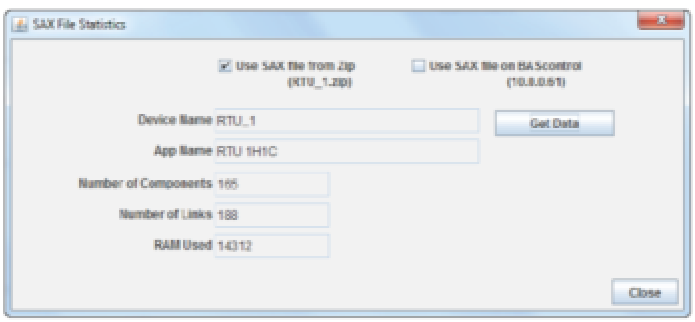
Getting SAX Data

There is a convenient feature on the BASbackup utility. By the utility recreating the Sax file from the Sab file, we can learn the amount of memory being used in the saved application or even from an active controller in the field with an unknown file. Click on the *Get SAX Data* button and you will see the following window:



The default selection gives you the data from the saved zip file indicated. If you use this option, make sure the saved location is what you want otherwise browse from the main page for the correct location. Click on Get Data to retrieve the data.

Figure 12 — Sax File Statistics Screen



If instead you check the box that provides the Sax file from a controller in the field, make sure the IP address indicated is the desired controller otherwise change the IP address on the main page.

Figure 13 — Sax File Statistics Screen

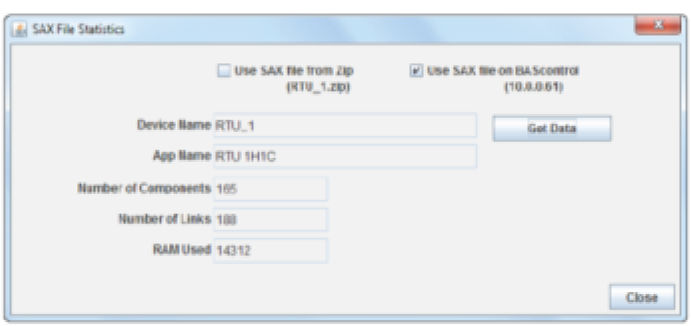


Figure 14 — Sax File Statistics Screen

C.1 Sedona 1.2 Component Descriptions

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 system integrator's role is to create an application by assembling components onto a wire sheet and connecting and configuring these components using a graphical programming tool such as Niagara Workbench or a third-party Sedona Tool. Applications can be developed live on a target device such as Contemporary Controls' BASremote or BAScontrol20/22, 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 from Tridium and Contemporary Controls that will be of the most use to system integrators when developing control applications. These components reside in kits which can be found in the Sedona Palette within Niagara Workbench (3.7.x or higher) or a Sedona Tool. Only those kits of the most interest are discussed.

Components Are Found in Kits

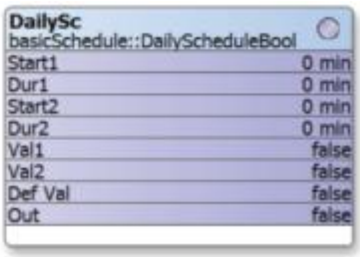
The following components are organized by kits. Boolean variables are assumed if there is a false/true state. Integers (32-bit signed integers) are shown as whole numbers while floats (32-bit floating point) are shown with a decimal point. Many of the following components may have been expanded in order to show slots for internally configurable parameters. The default view of a component may not show the same level of detail.

Index of Kits

Basic Schedule Kit (basicSchedule)

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.

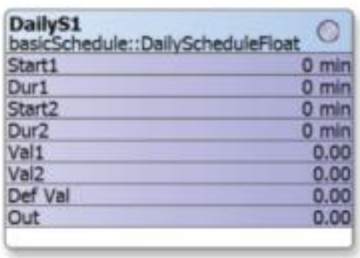
Duration periods — *Dur1* and *Dur2* — are configured in minutes from zero to 1439 minutes.



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



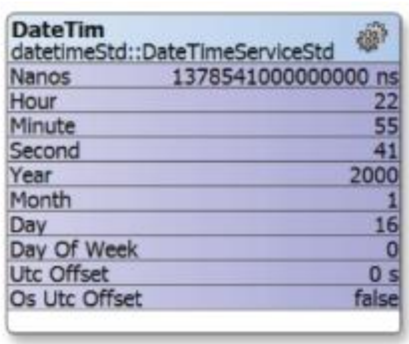
Daily Schedule Float — two-period float scheduler.

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

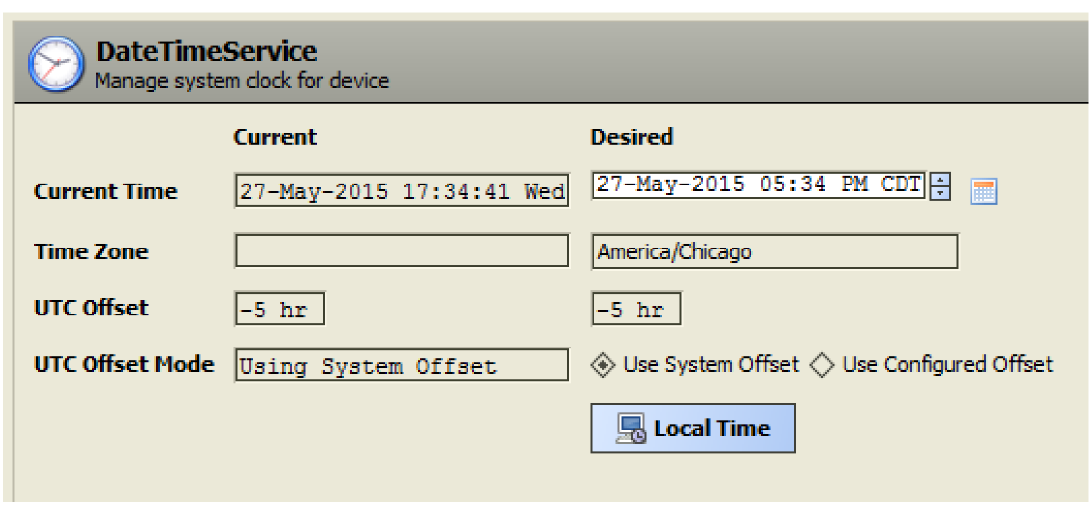
Date Time STD Kit (datetimeStd)

The DateTim component is the only component in the Date Time STD Kit. This component relies upon a properly functioning real-time clock implemented in hardware. Once date and time are configured, this component can be dragged onto a worksheet allowing individual integer outputs to be wired to logic if so desired. However, it is not necessary to have the component on the wiresheet at all. If the DailySchedule components are to be used, they will function properly without the presence of the DateTim component. The start and stop times in the DailySchedule key on the daily time generated by the DatTime component regardless if this component is on the wiresheet.

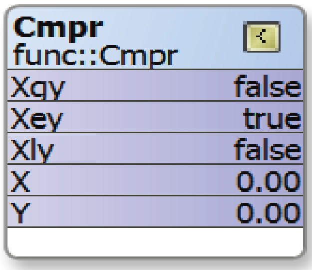


Please Note

By double clicking the DateTim component, you will see the setup screen below. When using Contemporary Controls' controllers, make sure that the Use System Offset option is selected as shown. To avoid confusing time settings, do not set the time with this component. Set the time using the Set Time web page on the controller which provides more flexibility and is less confusing. You can set time zone, daylight saving time and in some instances Network Time Protocol support using just the web page. These settings will then set this Sedona component properly.



Function Kit (func)

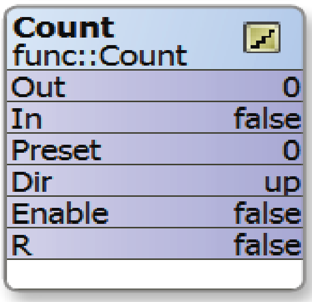


Comparison math — comparison (<=>) of two floats.

If X > Y then Xgy is true

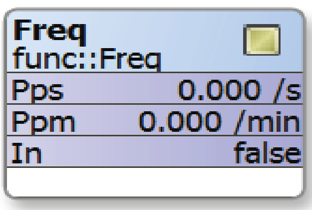
If X = Y then Xey is true

If X < Y then Xly is true



Integer counter — up/down counter with integer output.

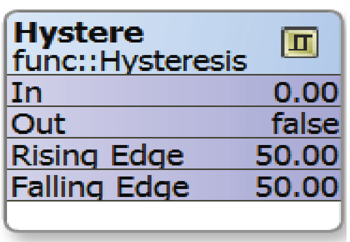
Counts on the false to true transition of *In*. If *Dir = true* the counter counts up to the maximum value of the integer. If *Dir = false* the counter counts down but not below zero. For counting to occur, *Enable* must be *true*. The counter can be preset. If *R = true* and *Enable = true*, then Out equals the preset value and will not count.



Pulse frequency — calculates the input pulse frequency.

Pps = number of pulses per second of In

Ppm = number of pulses per minute of In

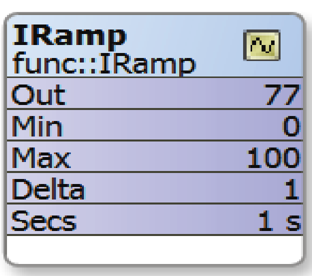


Hysteresis — setting on/off trip points to an input variable.

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.

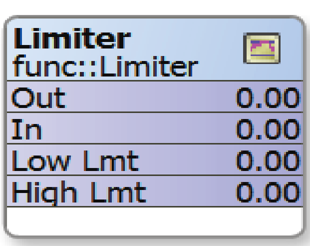
If In > Rising Edge then Out = true and will remain in that state until In < Falling Edge

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



IRamp — generates a repeating triangular wave with an integer output.

There are four configurable float parameters — *Min*, *Max, Delta and Secs*. For every scan cycle, the output increments by *Delta* units 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 *Secs* units.



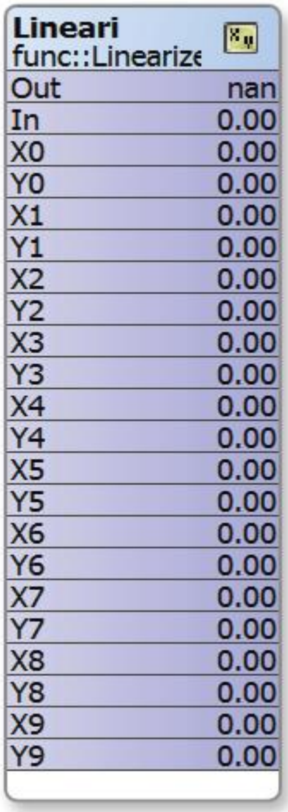
Limiter — Restricts output within upper and lower bounds.

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

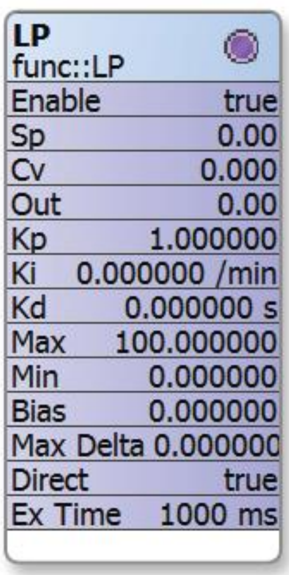


Linearize — piecewise linearization of a float.

For piecewise linearization of a nonlinear 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:

$$Out = y = y_0 + (y_1 - y_0)\frac{x - x_0}{x_1 - x_0}$$

where y is the value for input value x between coordinates x_0, y_0 and x_1, y_1



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. Setting *Kp* to zero would completely disable the controller. Typical controller operation is either:

Proportional-only (P)

Proportional plus reset (integral) (PI)

Proportional plus reset plus rate (PID)

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. However, if *Ki* or *Kd* are non-zero, internal calculations will continue.

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 greater 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 with a default of 1000. 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 –50% to +50%. When the output ranges from 0 to 50%, a proportional cooling valve is modulated. When the output ranges from 0 to –50%, a proportional heating valve is modulated. At 0% output no valve is open. This is called a split range control system. *Max* and *Min* are set to –50 and +50 respectively. When we force the controller output from maximum heat to maximum cooling (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   *where e = Cv – Sp and Bias equals zero*

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*, but in this case the bias is zero.

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

$Out = Kp(e + Ki\int e\,dt)$ (Bias is disabled when Ki is non-zero.)

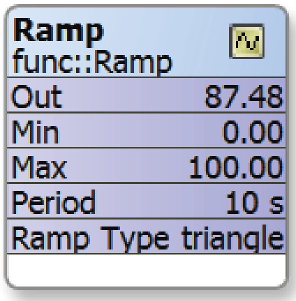
If there remains an error signal ($e \neq 0$), then the integral of the error over time will continue to drive the output until the error is driven to zero. The amount of action is determined by the *Ki* term. Notice that the integral term in the equation is also multiplied by the proportional gain before being applied to the output. The *Ki* 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 *Kp* and *Ki* is done in the field based upon system response.

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

$Out = Kp(e + Ki\int e\,dt + Kd\,de/dt)$

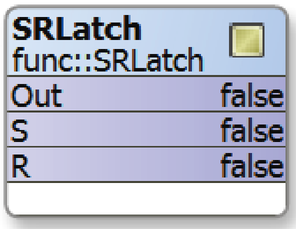
For processes with extremely long reaction times, derivative control could be helpful in reducing overshoot. *Kd* is entered in seconds. As mentioned before, it is seldom used because tuning a control loop with three parameters can be challenging.

There is one more parameter called *Max Delta*. This value limits the output slew rate by restricting the output change each time the control loop is recalculated by the amount entered. This parameter will dramatically reduce the response time of the control loop.



Ramp — generates a repeating triangular or sawtooth wave with a float output.

There are four configurable float parameters — *Min*, *Max, Period* and *Ramp Type*. 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* if *Ramp Type* is set for triangle. If *Ramp Type* is set for sawtooth, then the output will immediately drop to *Min* when *Max* is reached. The *Period* of the ramp is adjustable.



Set/Reset Latch — single-bit edge-triggered data storage.

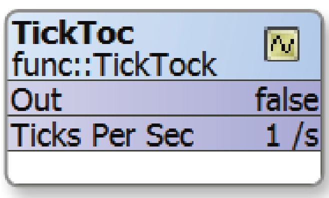
The following logic applies on the false-to-true transition of S or R:

If S goes true and R does not change, then Out = true and remains true.

If R goes true and S does not change, then Out = false and remains false.

If both S and R go true on the same scan, then Out = false and remains false.

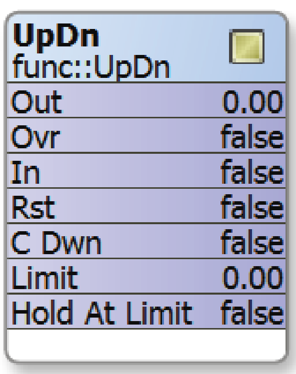
Ticking clock — an astable oscillator used as a time base.



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 periodic wave between 1 and 10 Hz

Float counter — up/down counter with float output.

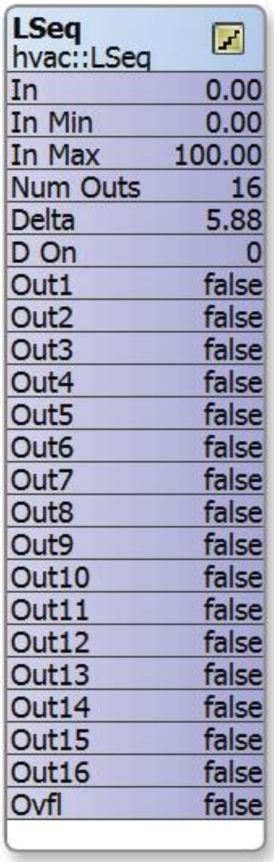


The counter range is between zero and a value that can be set with configurable parameter *Limit*. To cease counting at the limit set the configurable parameter *Hold at Limit* to true. To count down instead of up, set *C Dwn* to true. To reset the counter to zero set *Rst* to true. *Ovr* is the overflow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

Out = the current count

If Out ≥ Limit then Ovr is true

HVAC Kit (hvac)



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 one more than 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.

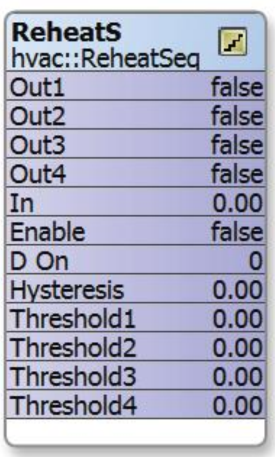
For example: *In Min* is set to 0, *In Max* to 100, and *Num Outs* is set to 9. This would give a *Delta* of 10. The following is true for increasing values of the input:

If In = 9 then Out1–16 are false and D On is zero.

If In = 70 then Out1–7 are true and Out8–16 are false. D On is 7.

If In = 101 then Out1–9 are true and Out10–16 are false. D On is 9 and Ovfl is true.

Note that for decreasing values of the input, the outputs will change by a value of Delta/2 below the input values stated above.



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:

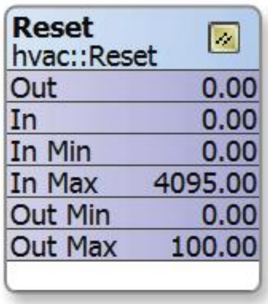
Out1 = true when In ≥ Threshold1

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.



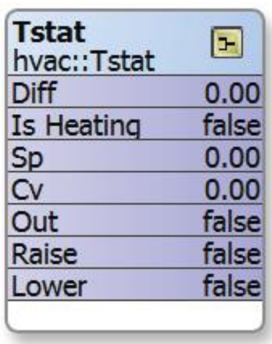
Reset — output scales an input range between two limits.

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.*



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 a nd *Cv* is the *controlled variable* input. *Raise* and *lower* are outputs.

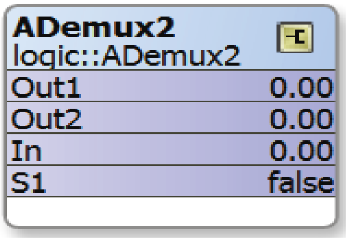
If *Cv* > *(Sp + Diff/2)* then *Lower* is true and will remain true until *Cv* < *Sp*

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*

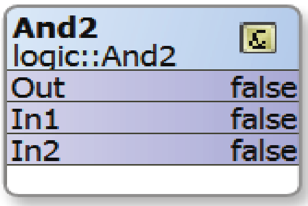
Logic Kit (logic)



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

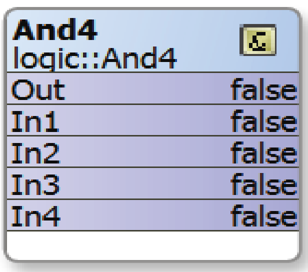
If S1 is false then Out1 = In while Out2 = the last value of In just before S1 changed.

If S1 is true then Out2 = In while Out1 = the last value of In just before S1 changed.



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

Out = In1 • In2



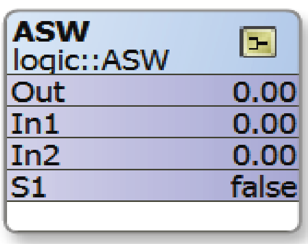
Four-input Boolean product — four-input AND gate.

Out = In1 • In2 • In3 • In4

Analog switch — selection between two float variables.

If S1 is false then Out = In1

If S1 is true then Out = In2



Analog switch — selection between four floats.

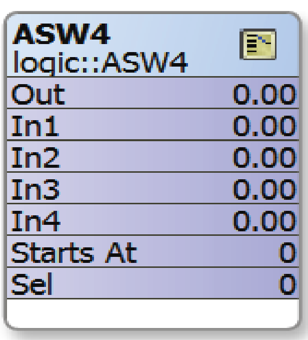
Configurable integer parameter *Starts At* sets the base selection.
If integer Sel <= Starts At then Out = In1

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

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

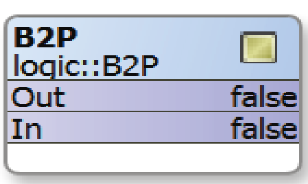
If integer Sel = Starts At + 3 then Out = In4

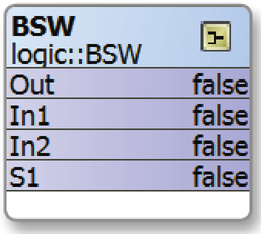
For all values of Sel that are 4 greater than Starts At then Out = In4



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

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

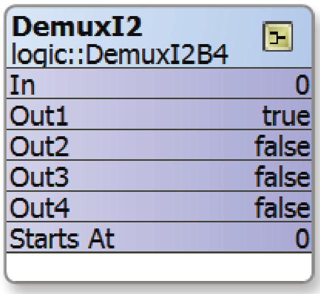




Boolean Switch — selection between two Boolean variables.

If S1 is false then Out = In1

If S1 is true then Out = In2



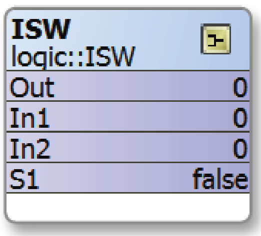
Four-output Demux — integer to Boolean de-multiplexer.

If In = StartAt + 0 then Out1 is true, else false

If In = StartAt + 1 then Out2 is true, else false

If In = StartAt + 2 then Out3 is true, else false

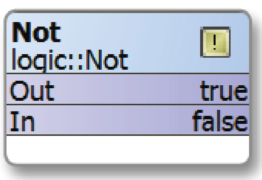
If In = StartAt + 3 then Out4 is true, else false



Integer switch — selection between two integer variables.

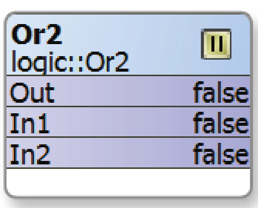
If S1 is false then Out = In1

If S1 is true then Out = In2



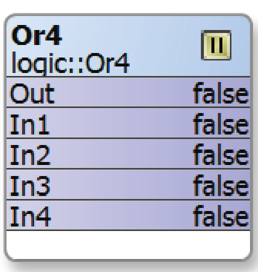
Not — inverts the state of a Boolean.

Out = In



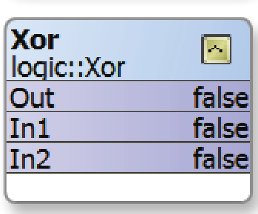
Two-input Boolean sum — two-input OR gate.

Out = In1 | In2



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

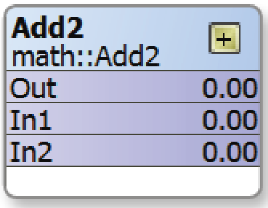
Out = In1 | In2 | In3 | In4



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

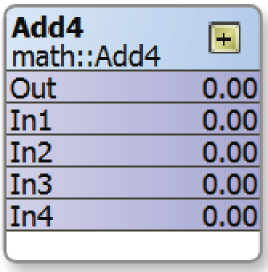
Out = In1 + In2 = In1 • In2 | In1 • In2

Math Kit (math)



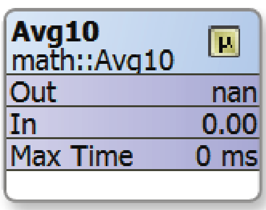
Two-input addition — results in the addition of two floats.

Out = In1 + In2



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

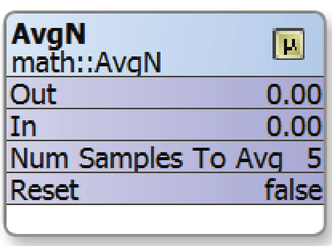
Out = In1 + In2 + In3 + In4



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

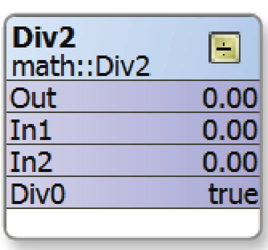
The float input *In* is sampled once every scan and stored. If the input does not change in value on the next scan, it is not sampled again — unless sufficient time passes that exceeds the internal integer *Max Time* with units of milliseconds. In this instance the input is sampled and treated as another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



Average of N — sums the last N floats while dividing by N thereby providing a running average.

Out = (sum of the last N values)/N

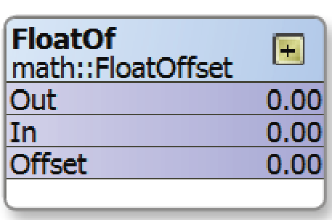
The float input *In* is sampled once every scan and stored regardless whether or not the value changes. Once *Num Samples to Avg* readings occur, the average reading is outputted.



Divide two — results in the division of two floats.

Out = In1/In2

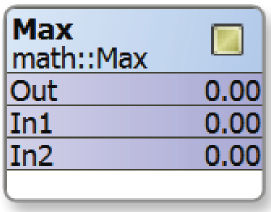
Div0 = true if In2 is equal to zero



Float offset — float shifted by a fixed amount.

Out = In + Offset

Offset is a configurable float.

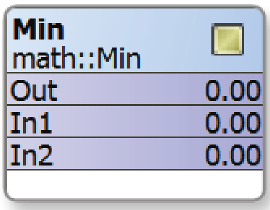


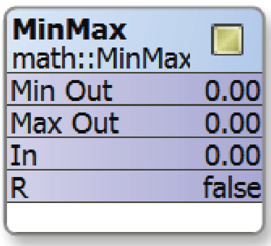
Maximum selector — selects the greater of two inputs.

Out = Max [In1, In2] where Out, In1 and In2 are floats

Minimum selector — selects the lesser of two inputs.

Out = Min [In1, In2] where Out, In1 and In2 are floats





Min/Max detector — records both the maximum and minimum values of a float.

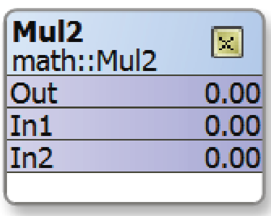
Min Out = Max [In] if R is false

Max Out = Min [In] if R is false

If R is true then Min Out and Max Out = In

Both *Min Out* and *Max Out* are floats — as is *In.*

It may be necessary to reset the component after connecting links to the component.

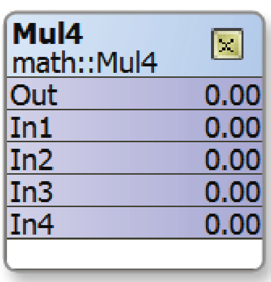


Multiply two — results in the multiplication of two floats.

*Out = In1 * In2*

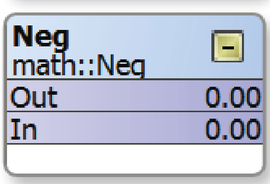
Multiply four — results in the multiplication of four floats.

*Out = In1 * In2 * In3 * In4*



Negate — changes the sign of a float.

Out = – In



Round — rounds a float to the nearest N places.

For N = -1, Out = In rounded to the nearest tens

For N = 0, Out = In rounded to the nearest units

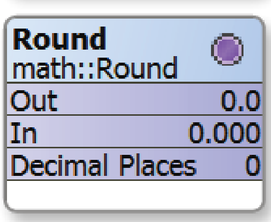
For N = 1, Out = In rounded to the nearest tenth's

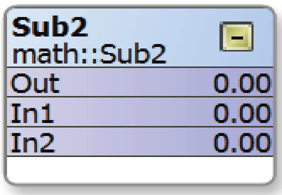
For N = 2, Out = In rounded to the nearest hundredths

For N = 3, Out = In rounded to the nearest thousandths

For positive input values, the output will round up (more positive).

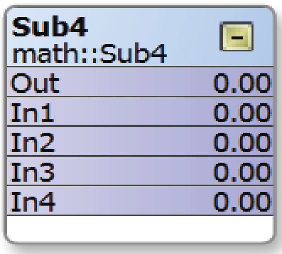
For negative input values, the output will round down (more negative).





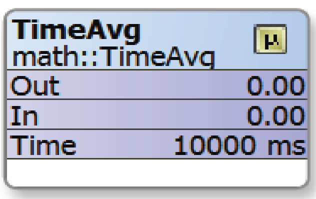
Subtract two — results in the subtraction of two floats.

Out = In1 – In2



Subtract four — results in the subtraction of four floats.

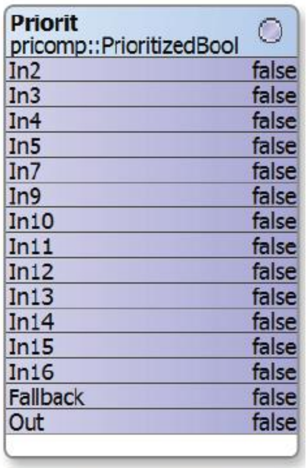
Out = In1 – In2 – In3 – In4

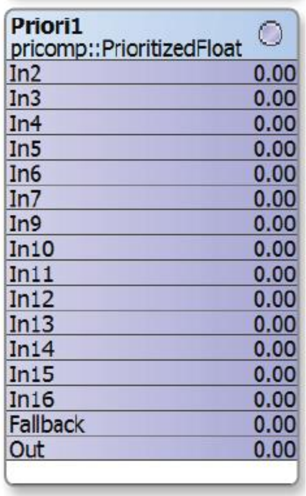


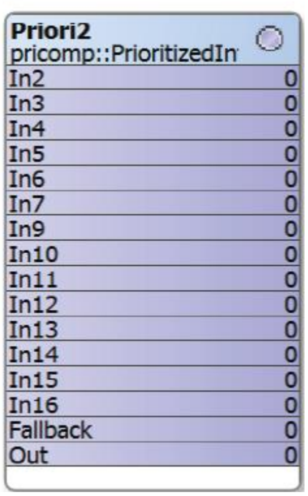
Time Average — the average of a float over a determined time.

Out = Avg[In] over the integer time in milliseconds.

Priority Kit (pricomp)







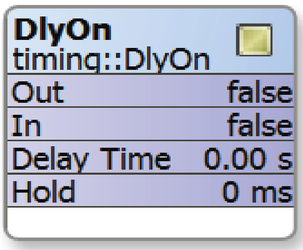
Priority array (Priorit) components exist for Boolean, float and integer variables. Up to 16 levels of priority from In1 to In16 can be assigned. In1 has the highest priority and In16 the lowest. With few exceptions, all can be pinned out. If a priority level is not assigned, it is marked as a Null and therefore ignored. If a Null is inputted to the priority array, the priority array will ignore it and choose the next input in line. The Boolean version of the array has two timer settings — one for minimum active time and one for minimum inactive time. If the highest priority device changes from false to true and then back to false, the priority component will maintain the event for the configured times.

There is a Fallback setting in each array that can be specified. If no valid priority input exists, the Fallback value is transferred to the output.

Timing Kit (timing)

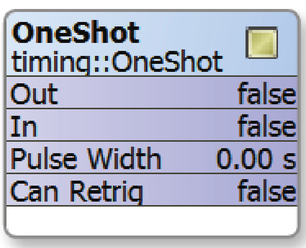
Off delay timer — time delay from a true to false transition of the input.

For input transitions from false to true, Out = true.
For input transitions from true to false that exceed the Delay Time, Out = false after the delay time.
Hold is a read-only integer that counts down the time. *Delay time* is in seconds.



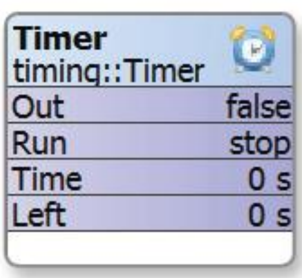
On delay timer — time delay from a false to true transition of the input.

For input transitions from true to false, Out = false.
For input transitions from false to true that exceed the Delay Time, Out = true after the delay time.
Hold is a read-only integer that counts down the time. *Delay Time* is in seconds.



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.

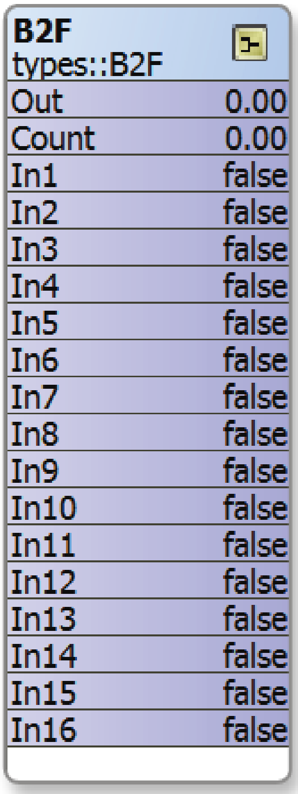


Timed pulse — predefined pulse output.

Out becomes true for a predetermined time when Run transitions from false to true. If Run returns to false, then Out becomes false.

Time determines the amount of time the output will be on in seconds.

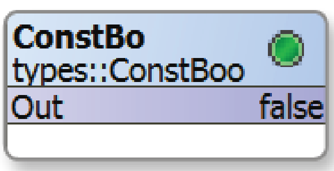
Types Kit (types)



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

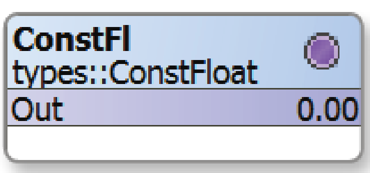
Out = encoded value of binary input with In16 being the MSB and In1 being the LSB

Count = sum of the number of active inputs



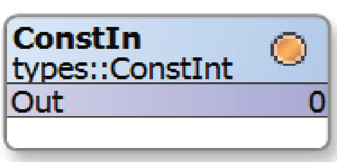
Boolean Constant — a predefined Boolean value.

Out = a Boolean value that is internally configurable



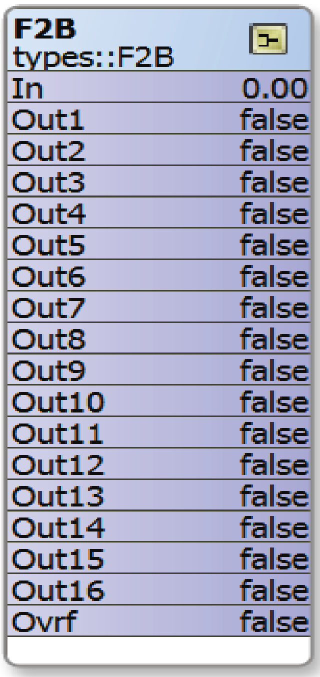
Float Constant — a predefined float value.

Out = a float value that is internally configurable



Integer Constant — a predefined integer value.

Out = an integer value that is internally configurable

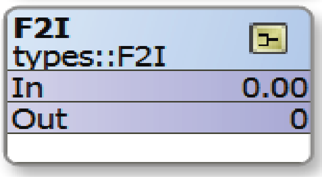


Float to binary decoder — float to 16-bit binary conversion.

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

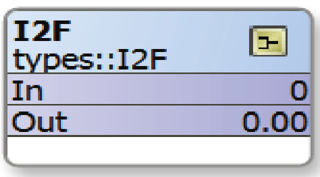
Although the input requires a float, fractional amounts are ignored during the conversion.



Float to integer — float to integer conversion.

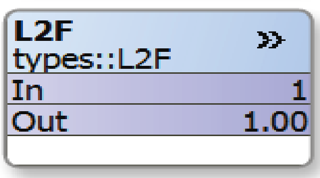
Out = In except that the output will be a whole number

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



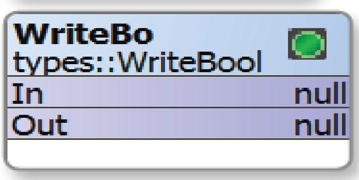
Integer to Float — integer to float conversion.

Out = In except that the output will become a float



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

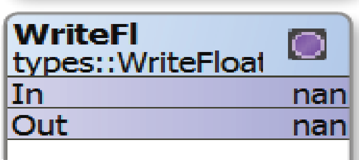
Out = In except that the output will become a float from a 64-bit signed integer



Write Boolean — setting a writable Boolean value.

Out = In

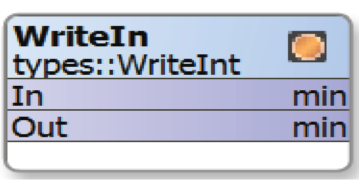
Unlike *ConstBo*, this component has an input. Could be helpful when transferring a variable between two wire sheets.



Write Float — setting a writable float value.

Out = In

Unlike *ConstFl*, this component has an input. Could be helpful when transferring a variable between two wire sheets.



Write Integer — setting an integer value.

Out = In

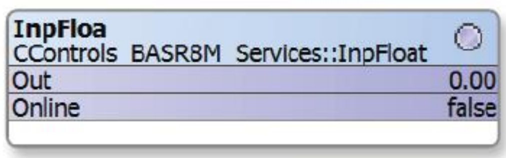
Unlike *ConstIn*, this component has an input. Could be helpful when transferring a variable between two wire sheets.

BASremote Service Kit (CControls_BASR8M_Services)

The BASremote service kit allows Sedona application to tie into real world inputs and outputs after object instance configuration. For the BASremote master, object instance assignments must match the I/O channel assignment. For configuring expansion module and virtual points, consult the BASremote User Manual for details. For the online status to revert to true, the point must be properly configured, must be actively scanned by the hardware and not be in a forced state.

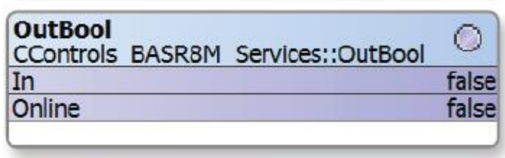
Input Boolean — BASremote binary input.

Out = value of the real world binary input



Input Float — BASremote analog input or value.

Out = value of the real world analog input

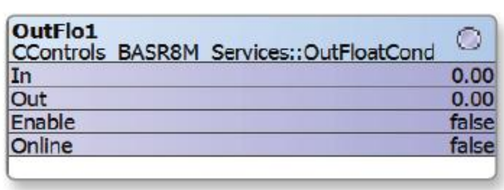


Output Boolean — BASremote binary output.

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

Output Float — BASremote analog output.

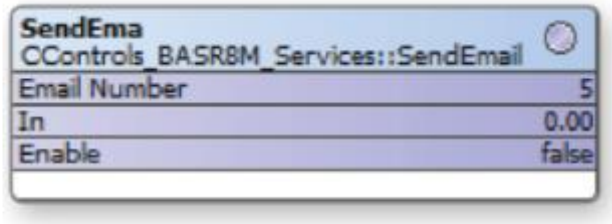
In = Float variable to be written to a real world output



Output Float Conditional — BASremote conditional analog output.

In = Float variable to be written to a real world output.
Out = Float value currently written to real world output.
Enable = Boolean which indicates whether a write should occur. True will allow the write to occur and False will inhibit any writes.

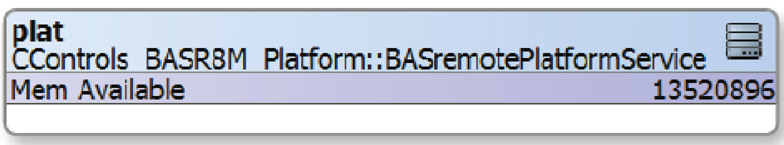
Sedona will, normally, write the outputs from its logic every cycle. This can be an issue for some Modbus registers controlled by the BASremote. The writes to these registers can be controlled via the enable signal. If enable is false the Modbus register associated with this component will not be written.



Send Email — BASremote email alert.
In = Float value to be included in email.
Enable = Boolean used to indicate when to send an email.
Email Number = which email to send (it must match the web configuration).

The BASremote can send an email using this component when the *Enable* signal is true. The email must be configured in the configuration webpage of the BASremote. When the email is sent, the text of the email will contain the current input float value. One Email will be sent on the false-to-true transition of the *Enable* signal.

BASremote Platform Kit (CControls_BASR8M_Platform)



The BASremote platform kit has one component that advises the programmer how much usable memory is available for application programming. With a Linux platform, memory is seldom an issue.

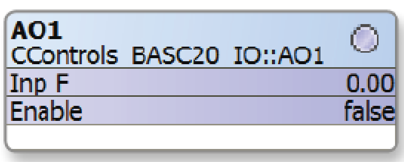
The platform kit is found in the service folder.

BAScontrol20/22 I/O Kit

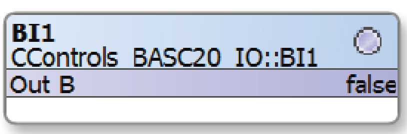
(CControls_BASC20_IO) (CControls_BASC22_IO)

The BAScontrol20 IO kit provides several components necessary to interface Sedona logic to real world inputs and outputs on the BAScontrol20. In addition to 20 real I/O points, the BAScontrol20 accommodates 24 virtual points that can be treated as either inputs or outputs. Universal inputs and virtual points require configuration via a web browser. Other components are included in this kit that are BAScontrol20 hardware dependent.

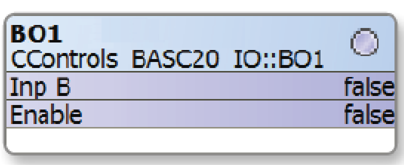
AO1 – AO4	Analog Output	analog voltage output points
BI1 – BI4	Binary Input	binary input points
BO1 – BO6	Binary Output	binary output points (B01-B04 with the CControls_BASC20_IO)
UI1 – UI4	Universal Input	binary, analog voltage, thermistor, resistance or accumulator
UI5 – UI8	Universal Input	binary, analog voltage, thermistor or resistance
UC1 – UC4	Retentive Counters	up/down retentive universal counters
VT01 – VT24	Virtual Points	share data with BACnet/IP clients - first eight components are retentive
ScanTim	Scan Timer	monitors the time to execute Sedona logic



AO1 – AO4 Analog Output — analog voltage output point.
Inp F = float value from 0–10 of respective point which translates to 0–10VDC output if Enable is true. If Enable is false, then output is controlled by a BACnet client.

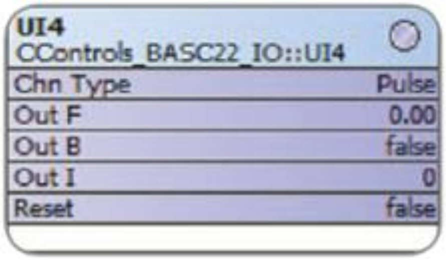


BI1 – BI4 Binary Input — binary input point.
Out B is true if input point is asserted to common; otherwise Out B is false.



BO1 – BO6 Binary Output — binary output point. (BO1-BO4 on BASC20)
Inp B = Boolean value of respective point which will translate to either a contact closure or triac output (on triac models).

If Inp B and Enable are true, the contact closure is made or the triac is turned on. If Enable is false, then output is controlled by a BACnet client.



UI1 – UI8 Universal Input — binary, analog voltage, thermistor, resistance or accumulator point (UI1-UI4 can be accumulators).

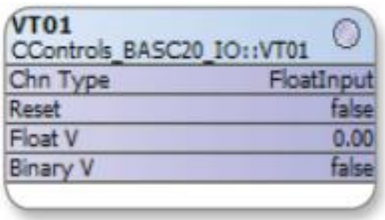
Out F = float value of respective point if configured for analog input, thermistor, resistance or pulse accumulator.

If point is configured as a thermistor, or resistance, and an out-of-range condition is detected, Out F = the configured Out of Bounds value and Out B = true (thermistor or resistance fault)

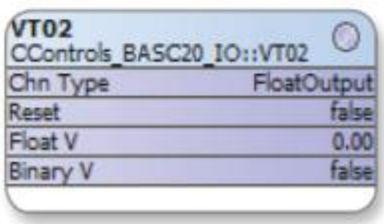
Out B = Boolean value if configured for binary input.
Out B is true if input point is asserted to common; otherwise Out B is false.
If in Pulse mode and Reset =true, then Out F = 0.
Out I = the integer representation of the float value.

VT01 – VT24 Virtual Points — wire sheet read or wire sheet write

Virtual points are used to share wire sheet data with a BACnet/IP client. A BACnet/IP client can “read” wire sheet data such as a calculated value or it can “write” to the wire sheet with a set-point or command. Virtual points are first configured from a web page to be a BACnet binary value (BV) or BACnet analog value (AV). The BACnet description field and units of measure can be set as well as the BACnet name which must be unique within the device. Next go to Workbench to change the wire sheet Read or Write directions. The title of the virtual point on the web page will change to Wire Sheet Write or Wire Sheet Read accordingly. The four possibilities are shown on the left labelled as VT01 through VT04.

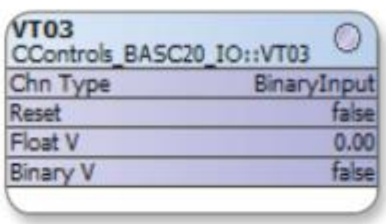


VT01 is configured as analog variable, wire sheet write, which results in the component being a *FloatInput*.

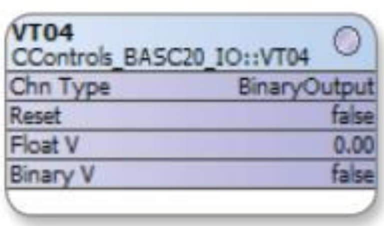


VT02 is configured as analog variable, wire sheet read, which results in the component being a *FloatOutput*.

VT03 is configured as binary variable, wire sheet write, which results in the component being a *BinaryInput*.



VT04 is configured as binary variable, wire sheet read, which results in the component being a *BinaryOutput*.

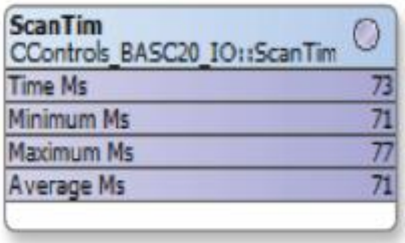


If configured as a *FloatInput,* then *Float V* represents the value written by the BACnet/IP client which can be used by other wire sheet components

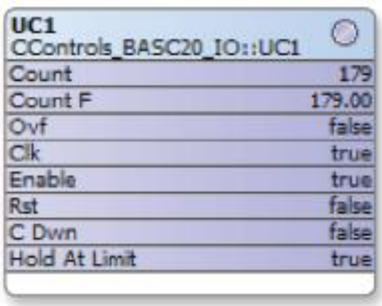
If configured as a *FloatOutput,* then *Float V r*epresents a value from a wire sheet component that can be read by the BACnet/IP client

If configured as a *BinaryInput,* then *Binary V* represents the value written by the BACnet/IP client which can be used by other wire sheet components

If configured as a *BinaryOut,* then *Binary V* represents a value from a wire sheet component that can be read by the BACnet/IP client
Asserting *Reset* will clear the component. It is usually kept in the *False* state.



ScanTimer – monitors the execution time of Sedona logic. The scan timer monitors the current, minimum, maximum and average time it takes to execute a single scan of Sedona logic. All outputs are integers. The average time is based upon the last ten samples. The result of which becomes the first value in the next ten samples. The component can be reset by right-clicking the component and invoking an Action.

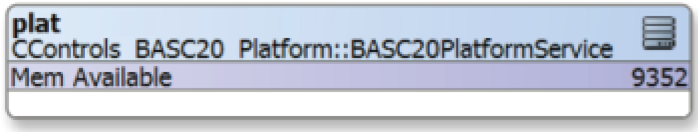


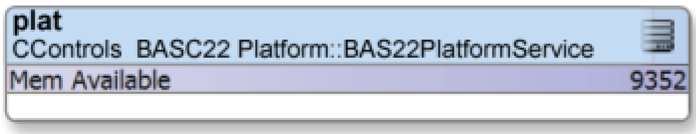
UC1 – UC4 — retentive up/down universal counters.

Counts on the false to true transition of *Clk* if *Enable* is *true*. If *C Dwn* is *true*, counting is down until zero is reached. If *C Dwn is false*, counting is up to the limit of the counter (2147483647) before it rolls over to zero. If *Hold At Limit* is set to true, the counter will stop counting at the value set in the *Limit* slot on the property page. The *Ovf* flag is set *true* when the value of status equals or exceeds the limit value. The output *count* value can be displayed as an integer (*Count*) or a float (*Count F*). *Rst* set to *true* clears the counter and prevents further counting.

BAScontrol20/22 Platform Kit

(CControls_BASC20_Platform) (CControls_BASC22_Platform)

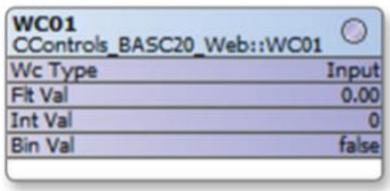




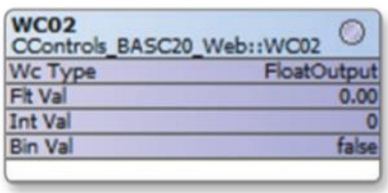
The BAScontrol20/22 platform kit has only one component that advises the programmer how much usable memory is available for application programming. It is recommended that the usable memory not fall below 8,192 bytes. It can be found in the services folder and can be copied onto the wire sheet. The output type of this component is a *Long*.

BAScontrol20 Web Kit (CControls_BASC20_Web)

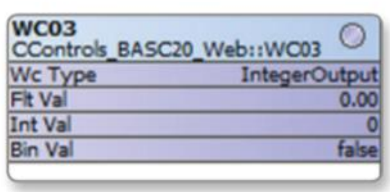
WC01 – WC48 Web Components — share data with BAScontrol20 web pages.
Web components provide a convenient method of sharing data between web pages and the wire sheet without the need of the Workbench tool. In this kit there are 48 web components that must be first configured via web pages. Web components can be configured to read wire sheet data or can write wire sheet data. The four possibilities are shown on the left labeled as WC01 through WC04.



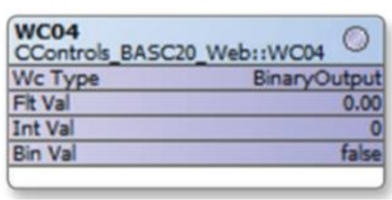
WC01 is configured as an input which results in the component being an *Input*.



WC02 is configured as an output float which results in the component being a *FloatOutput*.



WC03 is configured as output integer which results in the component being an *IntegerOutput*.



WC04 is configured as an output binary which results in the component being a *BinaryOutput*.

If configured as an Input then Flt Val, Int Val, and BinVal represents the value written by a web page which can be used by other wire sheet components

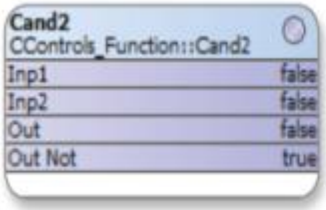
If configured as a FloatOutput, then Flt Val represents a value from a wire sheet component that can be read by a web page

If configured as an IntegerOutput, then Int Val represents a value from a wire sheet component that can be read by a web page

If configured as a BinaryOutput, then Bin Val represents a value from a wire sheet component that can be read by a web page.

Contemporary Controls Function Kit (CControls_Function)

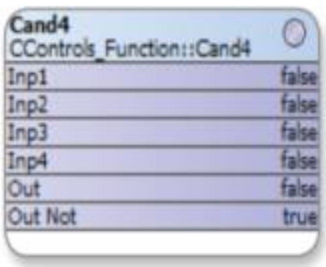
These components apply to any Sedona Virtual Machine (SVM).



Two-input Boolean product – two-input AND/NAND gate.

Out = In1 • In2

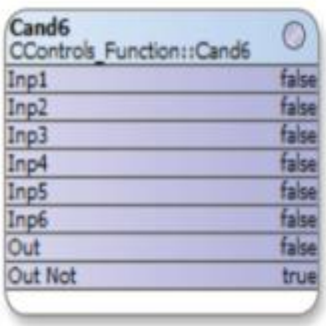
Out Not = Out



Four-input Boolean product – four-input AND/NAND gate.

Out = In1 • In2 • In3 • In4

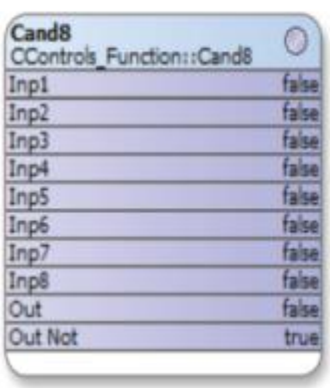
Out Not = Out



Six-input Boolean product – six-input AND/NAND gate.

Out = In1 • In2 • In3 • In4 • In5 • In6

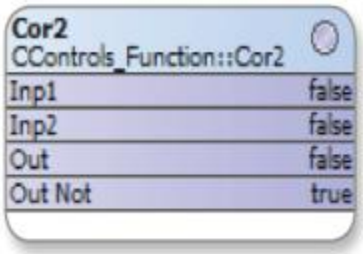
Out Not = Out



Eight-input Boolean product – eight-input AND/NAND gate.

Out = In1 • In2 • In3 • In4 • In5 • In6 • In7 • In8

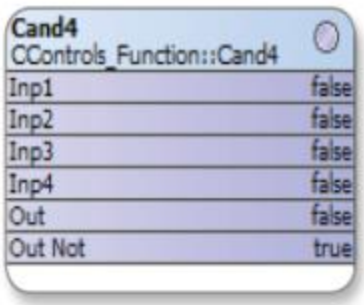
Out Not = Out



Two-input Boolean sum – two-input OR/NOR gate

Out = In1 | In2

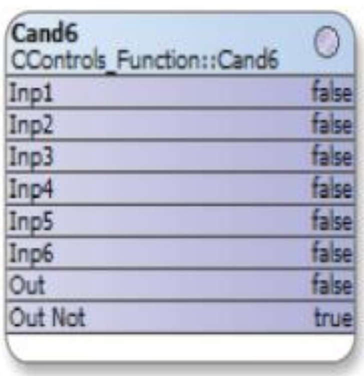
Out Not = Out



Four-input Boolean sum – four-input OR/NOR gate

Out = In1 | In2 | In3 | In4

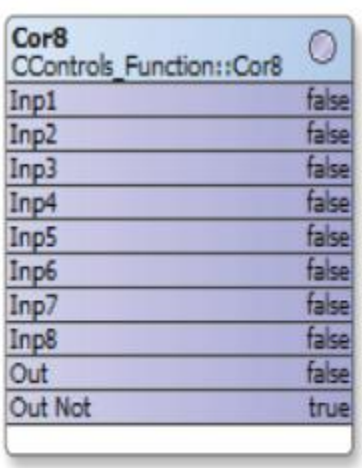
Out Not = Out



Six-input Boolean sum – six-input OR/NOR gate

Out = In1 | In2 | In3 | In4 | In5 | In6

Out Not = Out



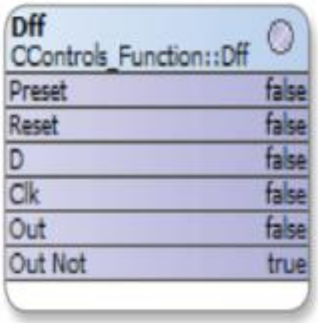
Eight-input Boolean sum – eight-input OR/NOR gate

Out = In1 | In2 | In3 | In4 | In5 | In6 | In7 | In8

Out Not = Out

Comment

A comment field from 1-64 characters used for documentation purposes.



"D" Flip-Flop – D-style Edge-triggered Single-bit Storage

If Preset = True and Reset = False then Out = True

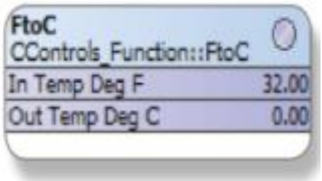
If Reset = True then Out = False regardless of all other inputs

On the rising edge of Clk with Preset = False and Reset = False;

If D = false then Out = false

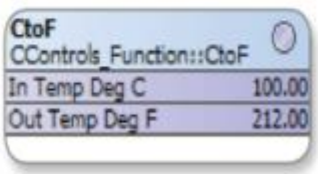
If D= true then Out = true

Out Not = Out



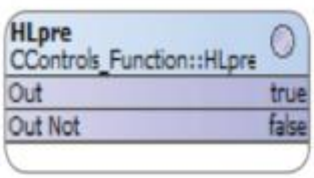
°F to °C – Fahrenheit to Celsius Temperature Conversion

*Out = 9/5 * In + 32*



°C to °F – Celsius to Fahrenheit Temperature Conversion

*Out = 5/9 * (In – 32)*



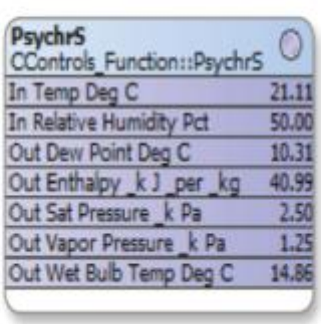
High – Low Preset – defined logical true and false states

Out = true

Out Not = false

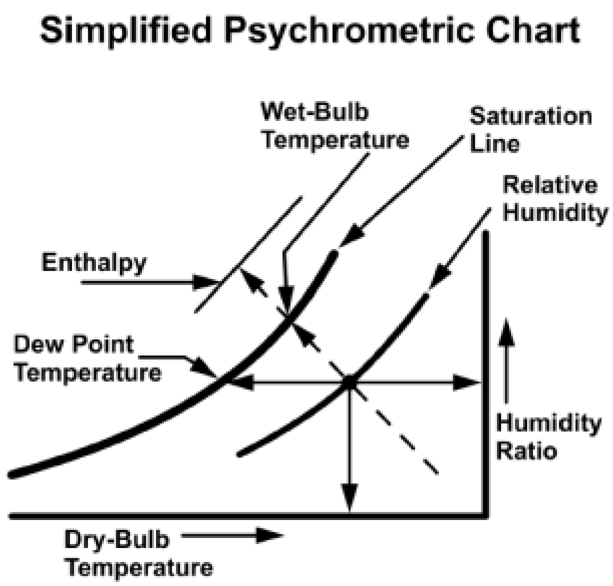
Psychrometric Calculator – English Units

Inputs are Dry-bulb temperature (°F) and Relative Humidity (%) Outputs are Dew Point (°F), Enthalpy (Btu/lb), Saturation Pressure (psi), Vapor Pressure (psi) and Wet-bulb temperature (°F) *Input temperature range 32-120°F; Input relative humidity range* 10-100%



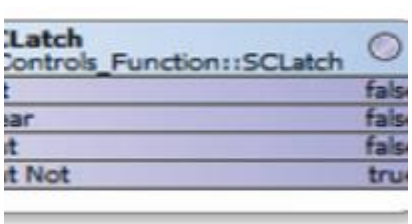
Psychrometric Calculator – SI Units

Inputs are Dry-bulb temperature (°C) and Relative Humidity (%) Outputs are Dew Point (°C), Enthalpy (kJ/kg), Saturation Pressure (kPa), Vapor Pressure (kPa) and Wet-bulb temperature (°C) *Input temperature range 0-48.8 °C; Input relative humidity range 10 100%*



A simplified psychrometric chart greatly removes the detail of a professional chart. On the X-axis is the dry-bulb temperature with a typical range from 32°F to 120°F. This is the same temperature you measure with a thermometer or wall-mounted thermostat. Lines of constant dry-bulb temperature are for all practical purposes vertical. On the Y-axis is the humidity ratio (lbw/lba) in lbs-water vapor to lbs-air ranging from zero to over 0.028. Lines of constant humidity ratio are horizontal. The left curved heavy line is called the saturation line indicating 100% saturation of water vapor or 100% relative humidity. Curves of lesser relative humidity would exist to the right of the saturation line. Along the saturation line you can determine both dew point temperature and wet-bulb temperature although their lines of constant temperature are different. For dew point, the lines are horizontal while the lines of constant wet-bulb are diagonal and almost parallel with lines of constant enthalpy.

Looking at the PsychrE component and the simplified chart we can study one example. Notice in the component that the two inputs are 70°F dry-bulb and 50% relative humidity. With these two values a single point on the psychrometic chart can be located. If you follow the horizontal line to the left you can determine the dew point temperature and to the right the humidity ratio. If you follow the diagonal line to the upper-left you can learn the wet-bulb and enthalpy values. We still have not determined the saturation pressure or the vapor pressure but these values can be derived with help from the humidity ratio. The PsychrE can make the calculations in the English system and the PsychrS can make the calculations in the SI system. Although simple conversions can be made between the two systems or to reflect the output values in different units of measure, be careful when working with enthalpy. With the English system, the change in enthalpy is referenced to a 0°F while in the SI system the reference is 0°C so a straight forward conversion between the two systems is not possible. Also note the limited range of the two psychrometric components. Both components are limited to an equivalent input range of 0-120°F dry-bulb and 10-100% relative humidity.



Set/Clear Latch – single-bit level-triggered single-bit data storage

The following logic applies to the state of Set or Clear:

If Set is true and Clear is false, then Out = true

If Clear is true, then Out = false regardless of the state of Set
Out Not = Out

Component–Kit Association

Component	Sedona Palette Folder
Add2	math
Add4	math
ADemux2	logic
And2	logic
And4	logic
AO1–AO4	CControls_BASC20_IO, CControls_BASC22_IO
ASW	logic
ASW4	logic
Avg10	math
AvgN	math
B2F	types
B2P	logic
BI1–BI4	CControls_BASC20_IO, CControls_BASC22_IO
BO1–BO6 (BO1-BO4 on BASC20)	CControls_BASC20_IO, CControls_BASC22_IO
BASC20PlatformService	CControls_BASC20_Platform
BASC22PlatformService	CControls_BASC22_Platform
BASremotePlatformService	CControls_BASR8M_Platform
BSW	logic
Cand2	CControls_Function
Cand4	CControls_Function
Cand6	CControls_Function
Cand8	CControls_Function
Cmpr	func
Cmt	CControls_Function
ConstBool	types
ConstFloa	types
Cor2	CControls_Function
Cor8	CControls_Function
Count	func
CtoF	CControls_Function
DailyScheduleBool	basicSchedule
DailyScheduleFloat	basicSchedule
DateTimeService	datetime
DemuxI2B4	logic
Dff	CControls_Function
Div2	math
DlyOff	timing
DlyOn	timing
F2B	types
F2I	types
FloatOffset	math
Freq	func
FtoC	CControls_Function
HLpre	CControls_Function
Hysteresis	func
I2F	types
InpBool	CControls_BASR8M_Services
InpFloat	CControls_BASR8M_Services

Component–Kit Association

Component	Sedona Palette Folder
ISW	logic
IRamp	func
L2F	types
Limiter	func
Linearize	func
LP	func
LSeq	hvac
Max	math
Min	math
MinMax	math
Mul2	math
Mul4	math
Neg	math
Not	logic
OneShot	timing
Or2	logic
Or4	logic
OutBool	CControls_BASR8M_Services
OutFloat	CControls_BASR8M_Services
OutFloatCond	CControls_BASR8M_Services
PrioritizedBool	pricomp
PrioritizedFloat	pricomp
PrioritizedInt	pricomp
PsychrE	CControls_Function
PsychrS	CControls_Function
Ramp	func
ReheatSeq	hvac
Reset	hvac
Round	math
ScanTim	CControls_BASC20_IO, CControls_BASC22_IO
SCLatch	CControls_Function
SendEmail	CControls_BASR8M_Services
SRLatch	func
Sub2	math
Sub4	math
TickTock	func
TimeAvg	math
Timer	timing
Tstat	hvac
UC1–UC4	CControls_BASC20_IO, CControls_BASC22_IO
UI1–UI8	CControls_BASC20_IO, CControls_BASC22_IO
UpDn	func
VT0–VT24	CControls_BASC20_IO, CControls_BASC22_IO
WC01–WC48	CControls_BASC20_Web, CControls_BASC22_Web
WriteBool	types
WriteFloat	types
WriteInt	types
Xor	logic

D.1 Using Sedona 1.2 Components from Tridium's Kits

Introduction

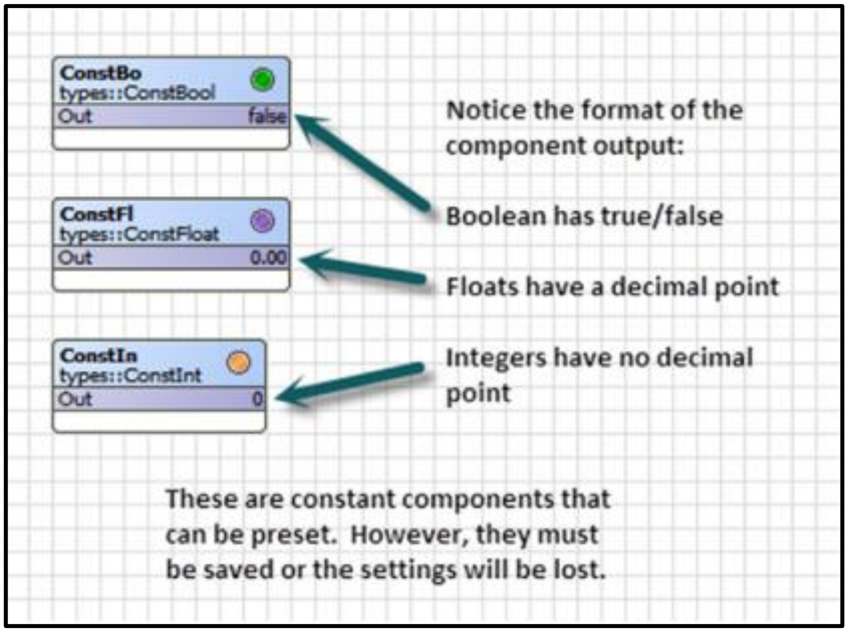
The following assists in the understanding of the Sedona components provided in Tridium's Sedona-1.2.28 release. Some of the Sedona components were changed or added since the previous release. New with the 1.2 release is that the Sedona components, previously concentrated in one Control kit, are now organized in smaller kits under a functional name. Components discussed in this document can be found in the following kits:

- basicSchedule
- datetimeSTD
- func
- hvac
- logic
- math
- pricomp
- timing
- types

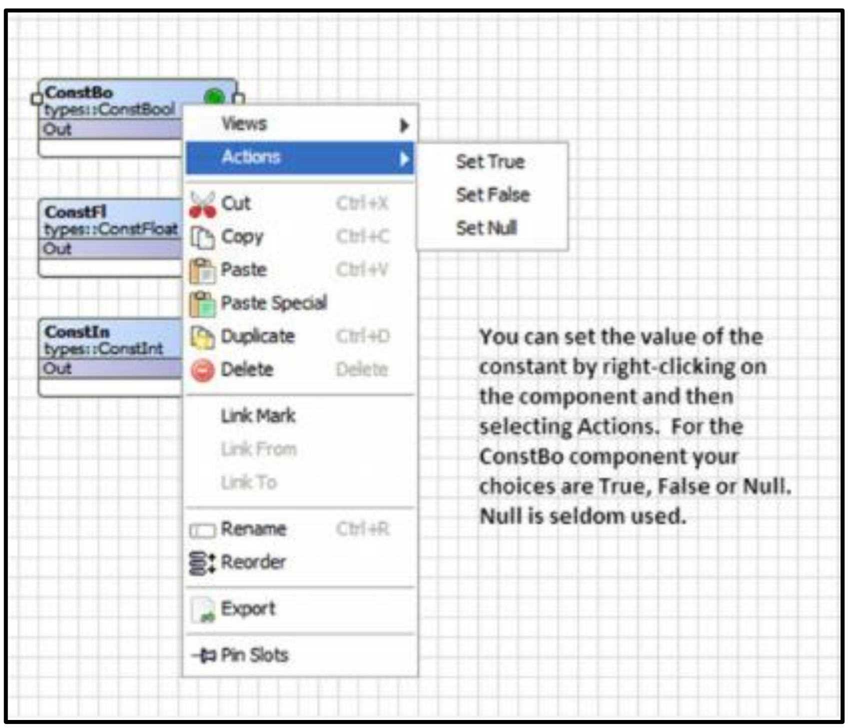
The intent of this document is to explain the potential use of those components supplied by Tridium in their Sedona 1.2 release. All are included in Contemporary Controls' BASremote and BAScontrol product families. They have not been modified for use in these products. Contemporary Controls has product specific Sedona kits that address the uniqueness of the IO structure in the BASremote and BAScontrol products. These kits are not mentioned in this document. It is Contemporary Controls' policy to provide all Sedona kits to the Sedona Framework community without charge or license. This includes kits obtained from Tridium, kits with modified Tridium components, kits developed solely by Contemporary Controls to improve the control options available to systems integrators, and kits specific to Contemporary Controls' hardware.

Variable Types

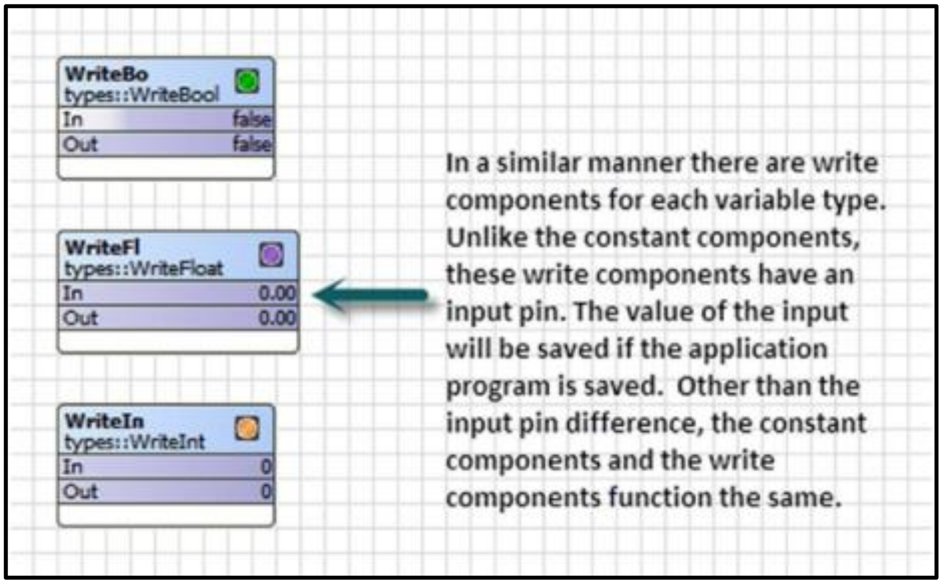
Although there are several variable types used by Sedona, three are the most interesting — Boolean, Float and Integer. You can define constants for each type and use converting components to change the data representation to a different type.



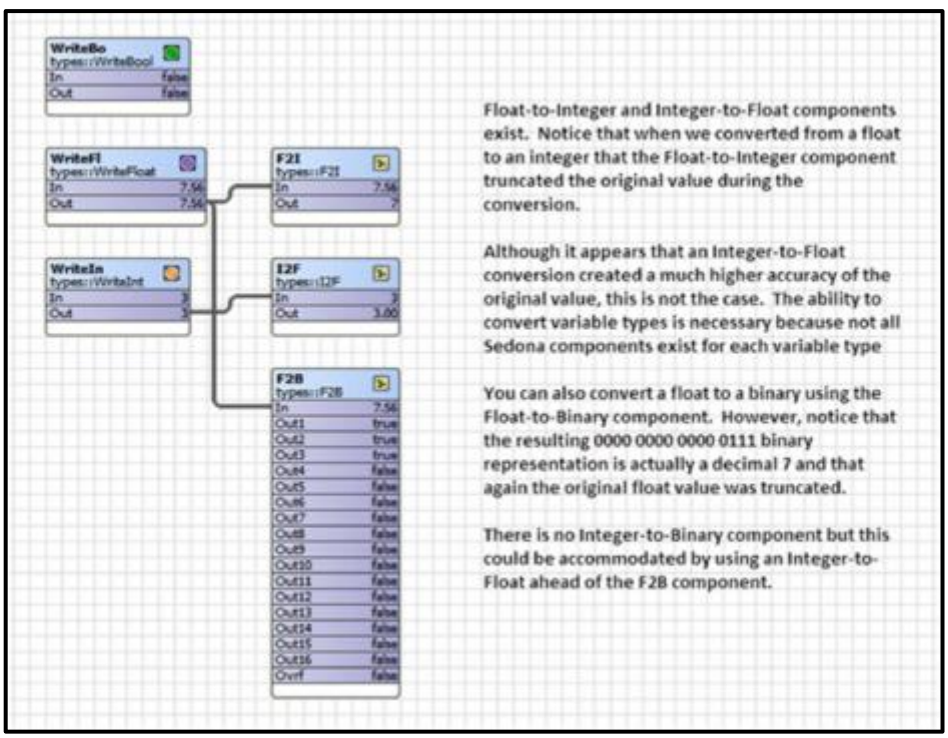
Configuring Constants



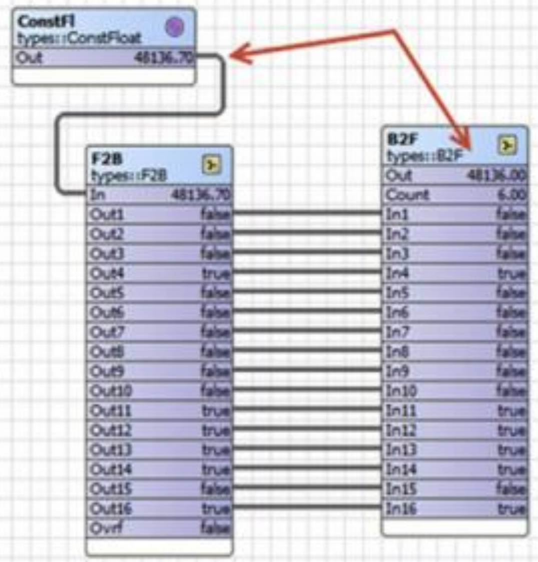
Using Write Components



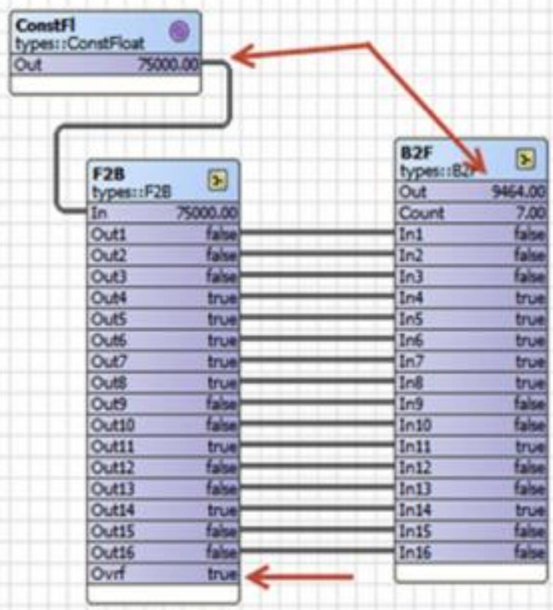
Converting Between Component Types



Float-to-Boolean and Boolean-to-Float Conversion

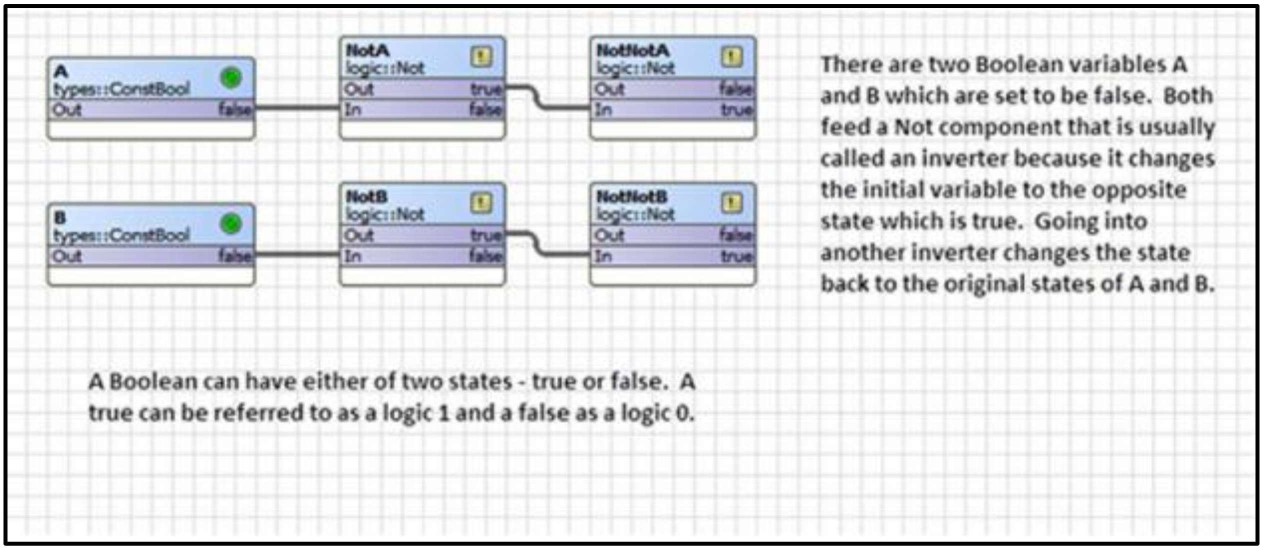


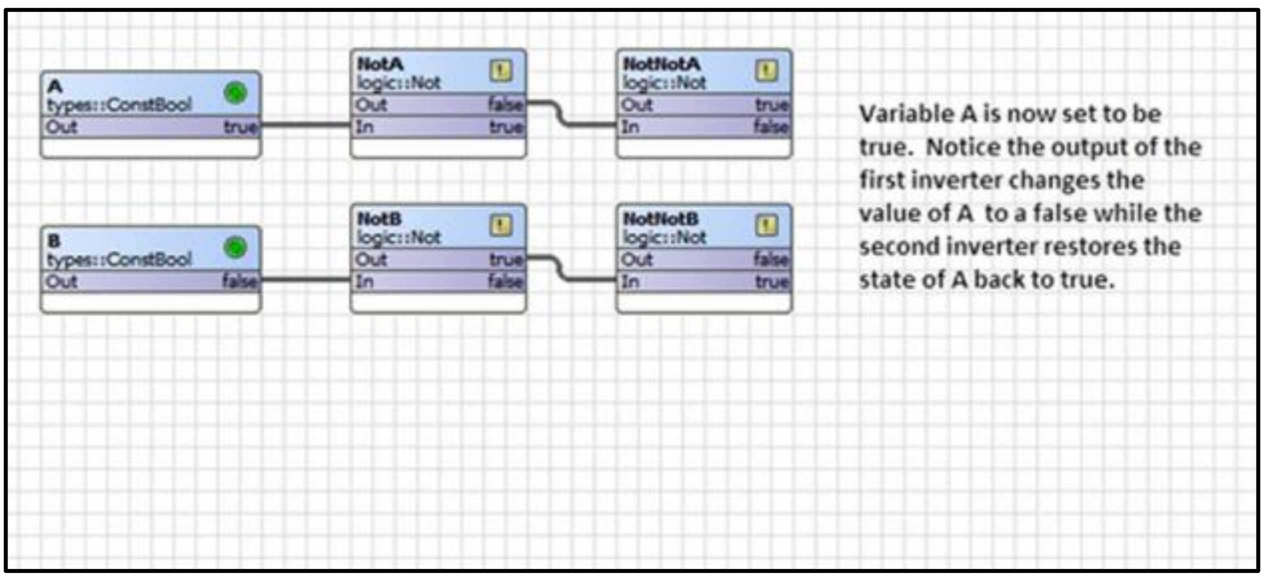
In this example we will begin with a float with a value of 48136.70 and convert it to a binary using a Float-to-Binary component and then immediately convert it back into a float using a Binary-to-Float component. Notice the exact recovered float value except for the truncation.



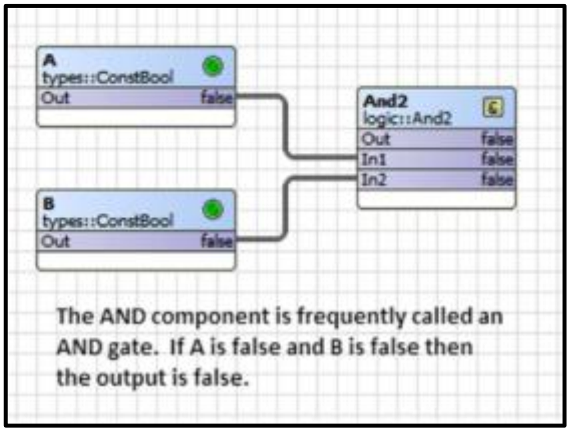
We increased the float value to 75000 but this time we have different answers. Because we are only doing a 16-bit conversion, we can only count up to 65535. Notice that the Ovrf pin is true meaning there was a counter overflow. The Ovrf pin means that a value of 65536 or higher was detected. The remaining counter value called the residue represents a Modulo-65536 result of 9464. If you subtract 65536 from 75000 you would get 9464. It is important to monitor the Ovrf flag when doing float to binary conversion.

Negating a Boolean Variable — Inverting Your Logic

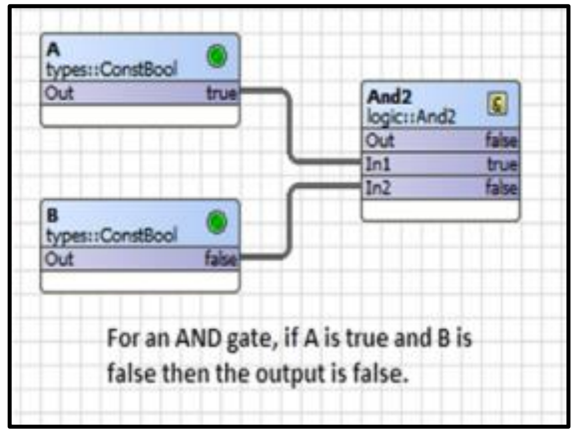




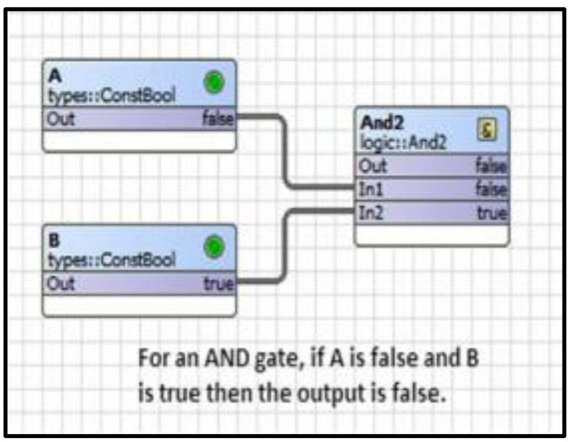
Boolean Product — "ANDing" Boolean Variables



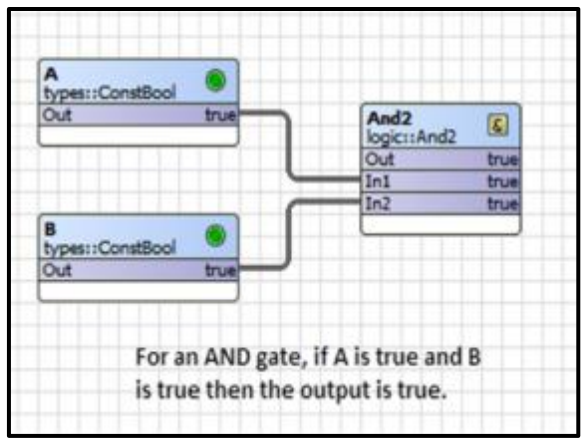
The AND component is frequently called an AND gate. If A is false and B is false then the output is false.



For an AND gate, if A is true and B is false then the output is false.

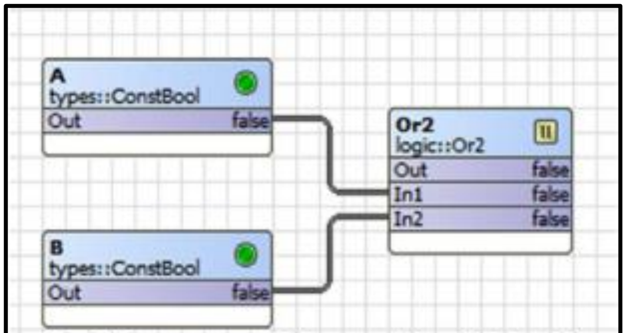


For an AND gate, if A is false and B is true then the output is false.

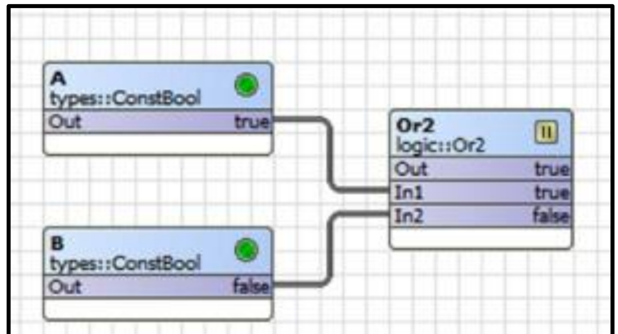


For an AND gate, if A is true and B is true then the output is true.

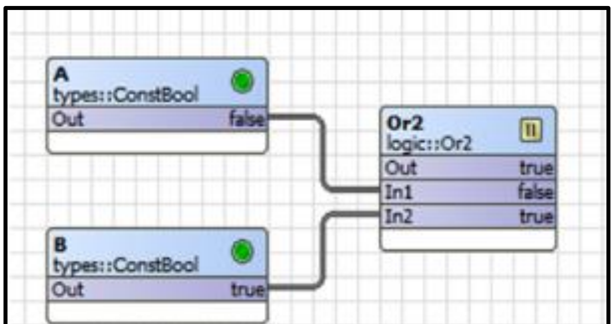
Boolean Sum — "Oring" Boolean Variables



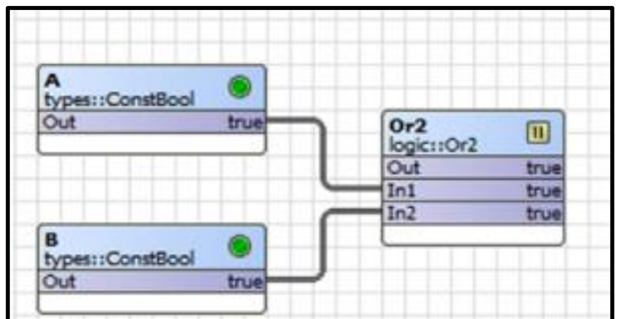
The OR component is frequently called an OR gate. If A is false and B is false then the output is false.



For an OR gate, if A is true and B is false then the output is true.

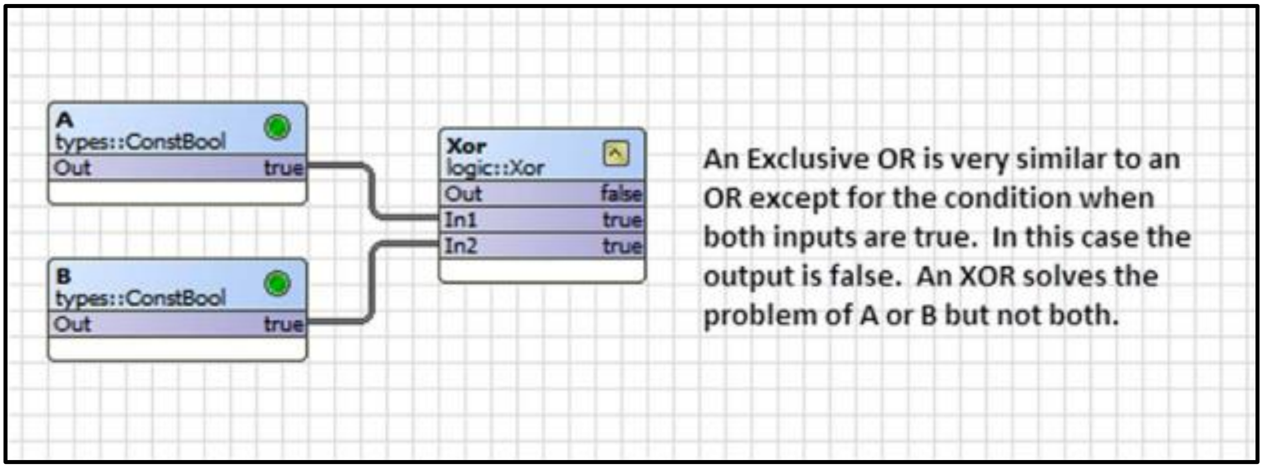


For an OR gate, if A is false and B is true then the output is true.

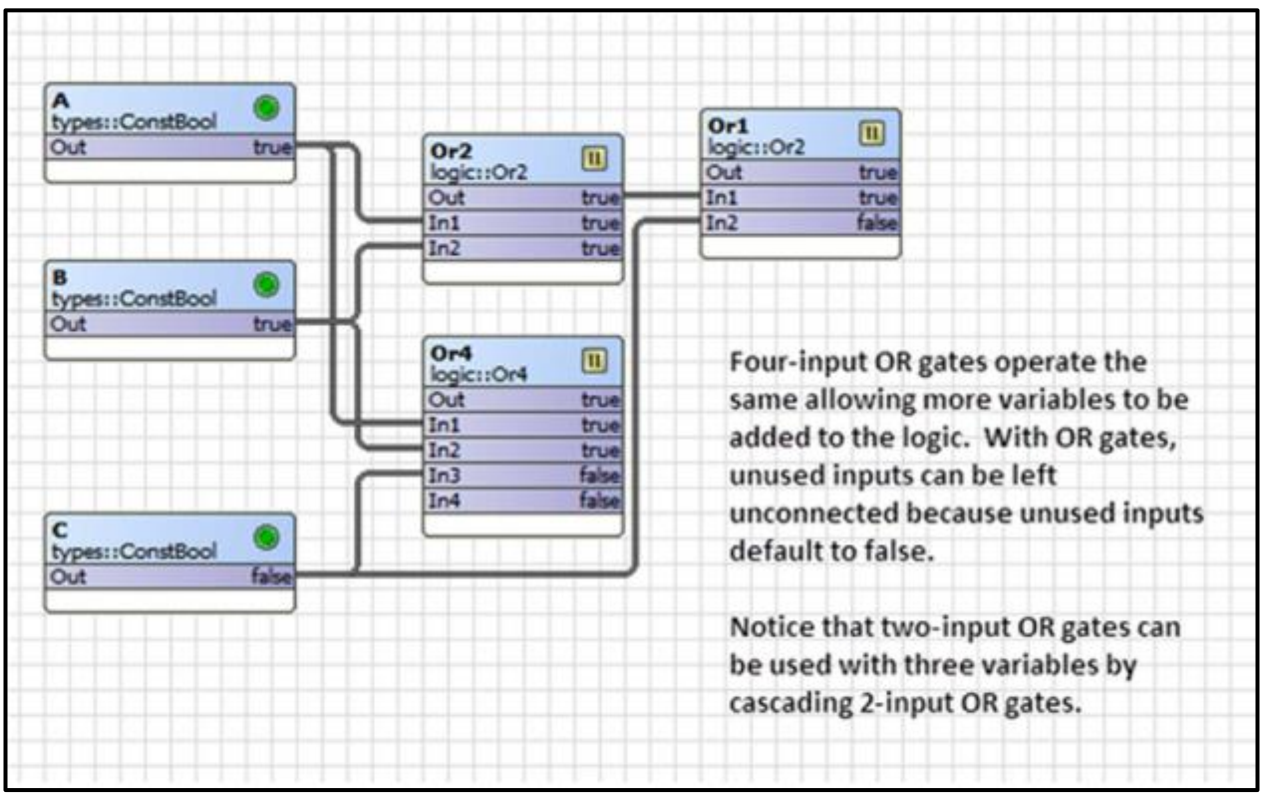


For an OR gate, if A is true and B is true then the output is true.

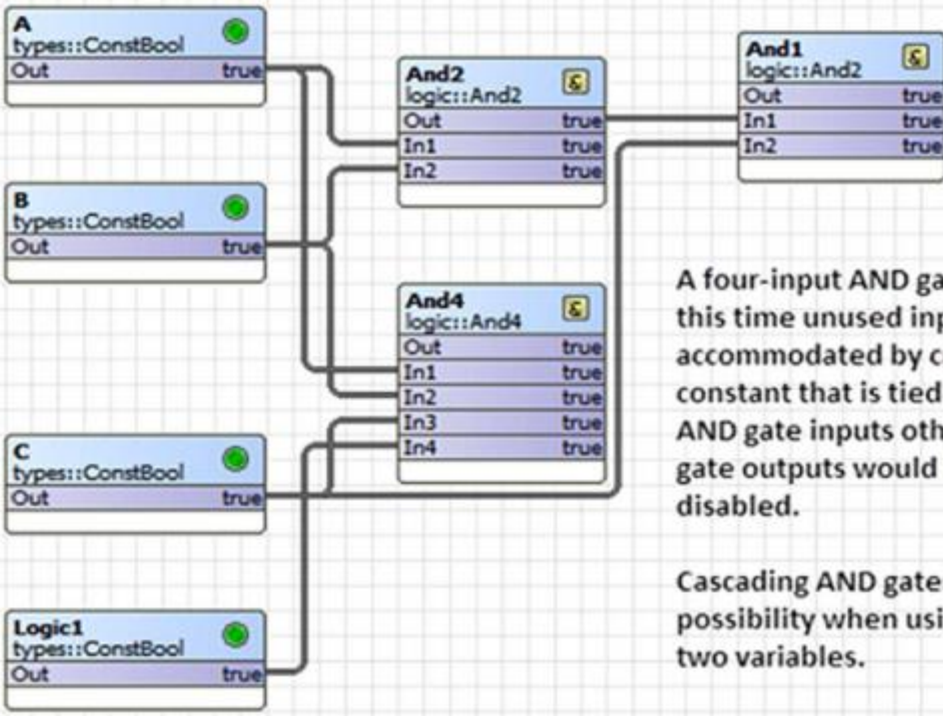
Exclusive OR — A OR B but Not Both



Cascading Logic Blocks and Unused Inputs

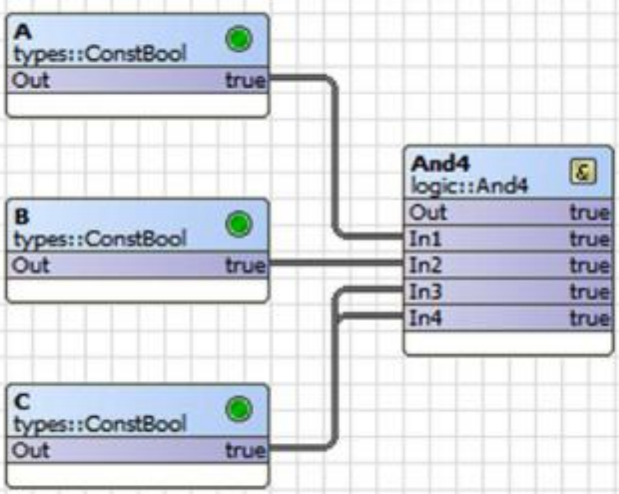


Cascading Logic Blocks and Unused Inputs (continued)



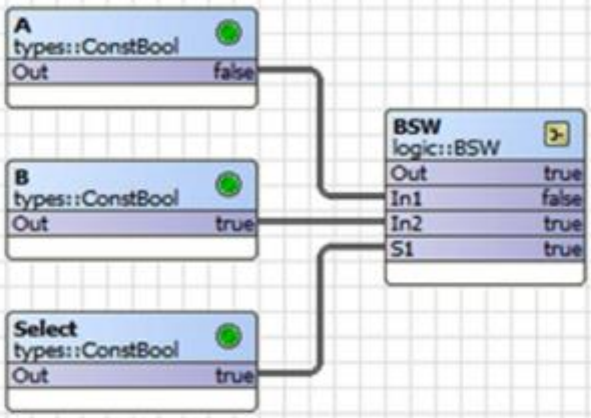
A four-input AND gate exists but this time unused inputs must be accommodated by creating a Logic1 constant that is tied to all unused AND gate inputs otherwise the AND gate outputs would be permanently disabled.

Cascading AND gates is also a possibility when using more than two variables.

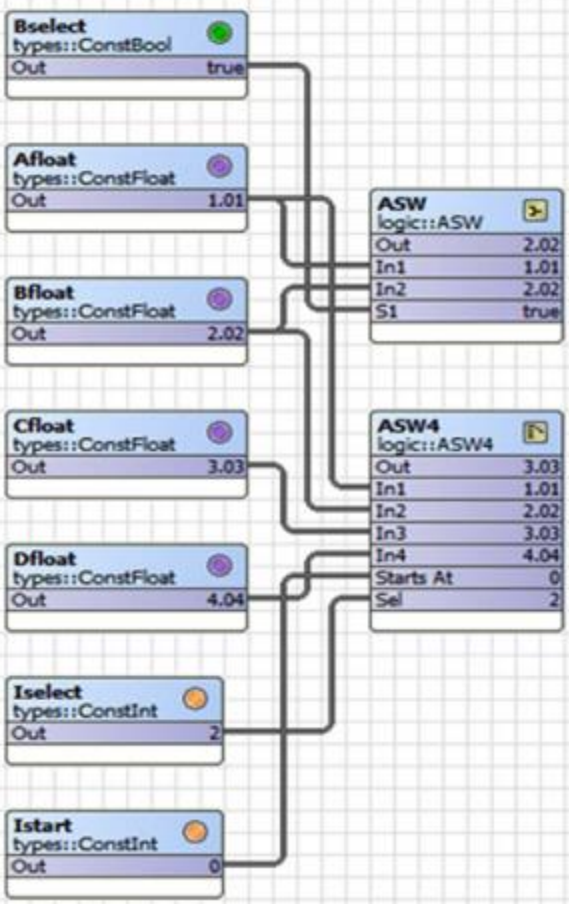


Here is another way of handling an unused input when three variables are connected to a four-input AND gate. Attach the unused input to one of the variables. It does not matter which one is used.

Boolean, Float or Integer Selection



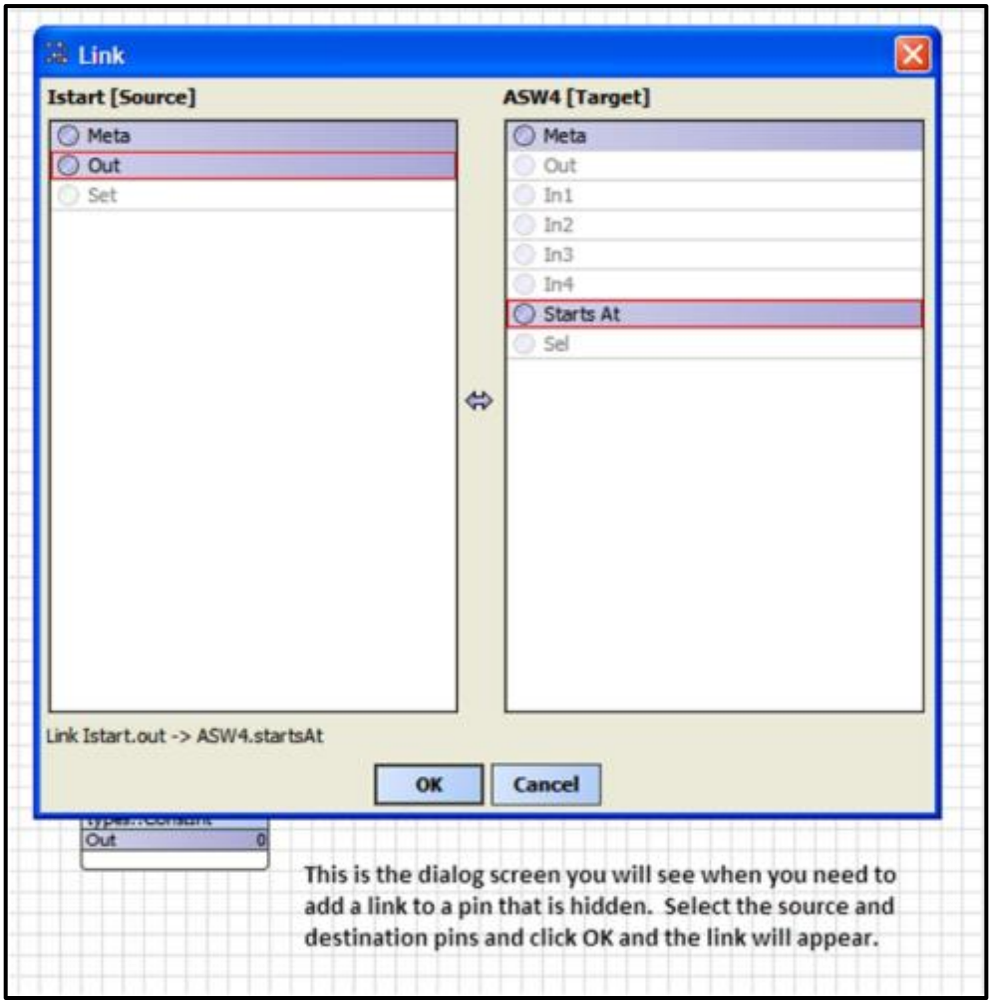
A two-input binary selector switch is used to enable one variable over another. If the S1 pin is true, then the value at In2 is passed to the output of the switch. If S1 is false, then In1 is passed to the output.



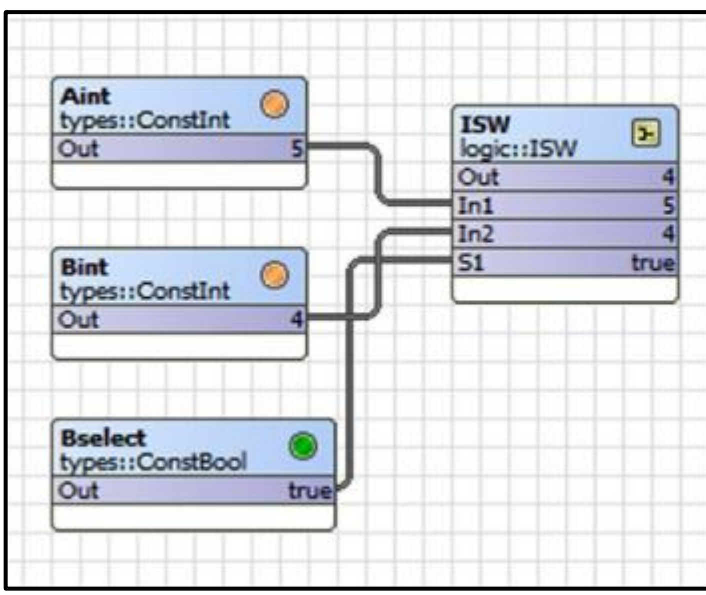
Similar in operation to the binary switch (BSW) is the analog switch (ASW). Instead of Boolean variables, the inputs are floats but the selector (S1) is a Boolean variable. With S1 set to true, the ASW output selects input I2 otherwise I1 is selected. Therefore the ASW selects one of two input float options.

The four-input analog switch (ASW4) is a bit different. Granted there are four float inputs instead of two with the ASW, there are other differences. Notice that the selector pin (Sel) is actually an integer and not a Boolean thereby allowing the selection of up to four float inputs. Also notice that the selection process begins at a particular integer value (Starts At). In this example the Starts At pin has a value of 0 meaning that input 1 (In1) is selected if the selection pin value (Sel) is 0. Since Sel is 2, the third input (In3) is selected. Normally the Start At pin is not viewable but if you drag a link between a component to the white area below the bottom slot of ASW4 and un-click, a selection box appears. After selecting Start As as the destination pin it will appear with the link connected.

Boolean, Float or Integer Selection (continued)

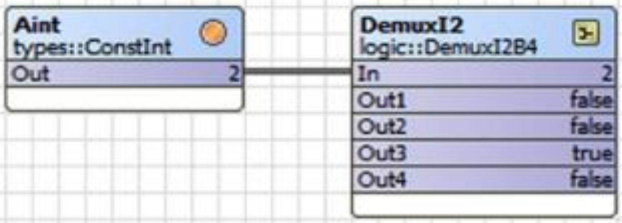


This is the dialog screen you will see when you need to add a link to a pin that is hidden. Select the source and destination pins and click OK and the link will appear.

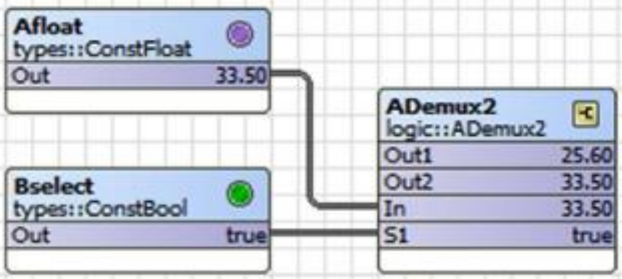


The integer switch (ISW) is very much like the BSW and the ASW. The selector is a Boolean but the inputs are integers. The output remains an integer. The logic is the same. If the selector (S1) is true, then the output follows In2 otherwise I1.

De-Multiplexing

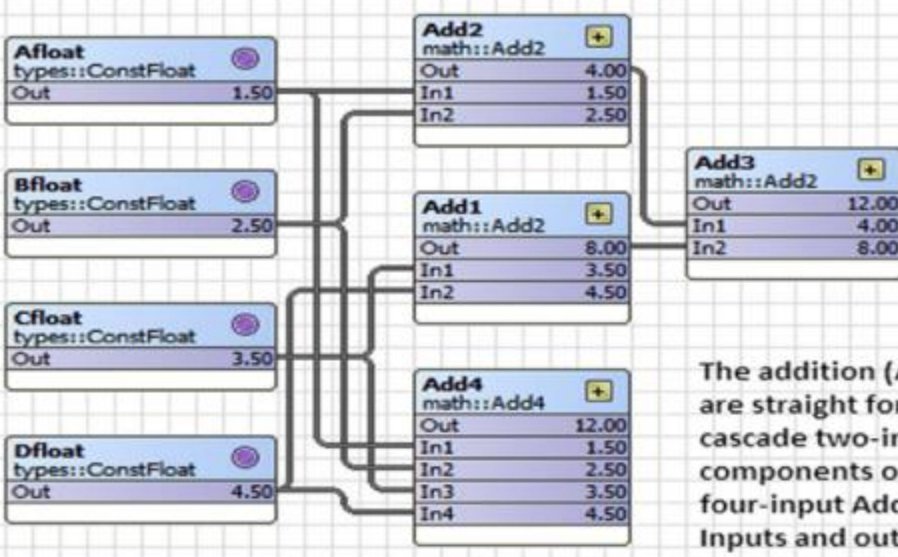


The de-multiplexer (DemuxI2) operates on integer values and provides a linear selection of the outputs based upon the value of the input. If the input is 0, the first output (Out1) is set to true. If the input is 2, the third output (Out3) is set to true.



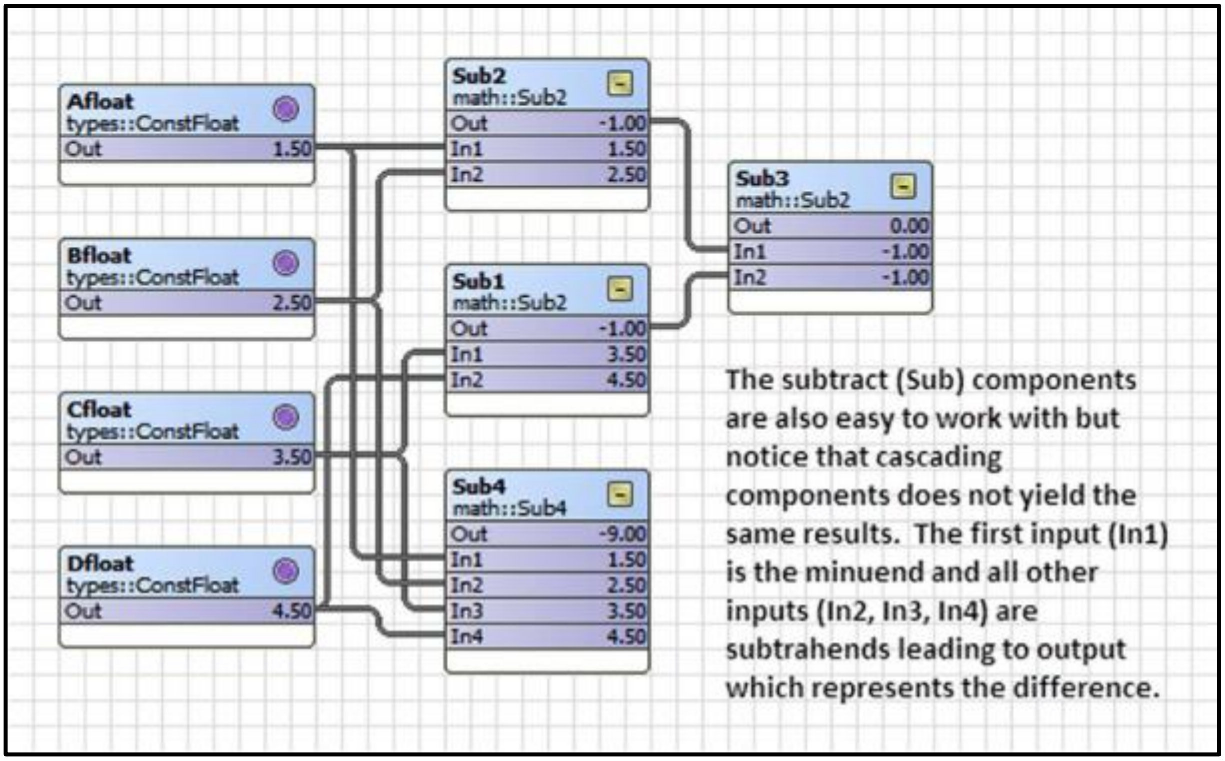
The analog de-multiplexer has only one input (In) and one selector (S1) but two outputs. When the selector is false, Out1 reflects the input. When the selector is true, Out 1 will reflect the input just before the selector changed state and Out2 will reflect the instantaneous value of the input. This component can be treated as a sample-and-hold detector.

Float Addition

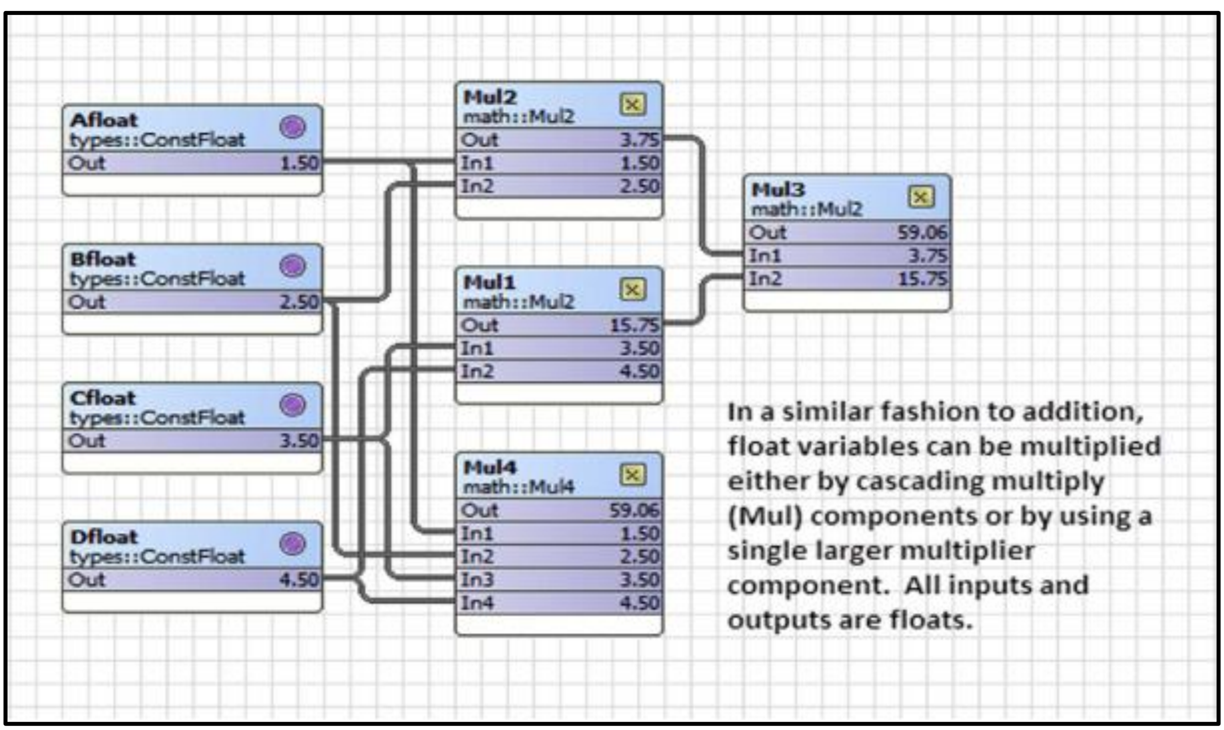


The addition (Add) components are straight forward. You can cascade two-input Add components or you can use a four-input Add component. Inputs and outputs are all floats.

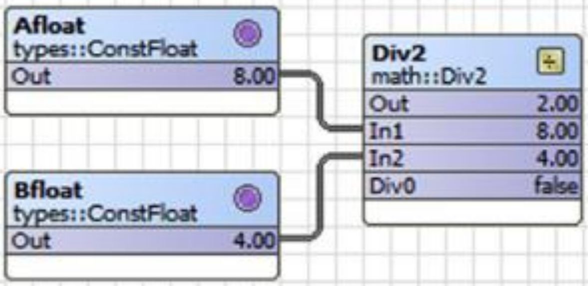
Float Subtraction



Float Multiplication

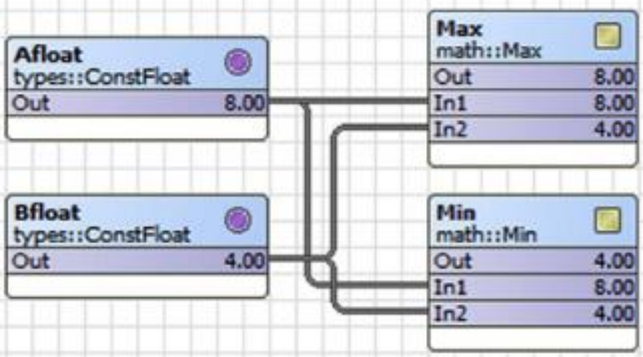


Float Division

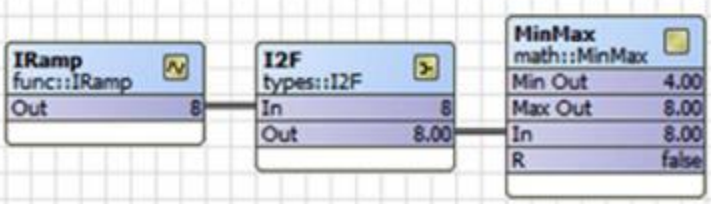


Division is also straight forward. Input 1 (In1) is the dividend, input 2 (In2) is the divisor and the output (Out) is the quotient. Dividing by zero will result in the pin Div0 being set to true.

Finding Minimums and Maximums



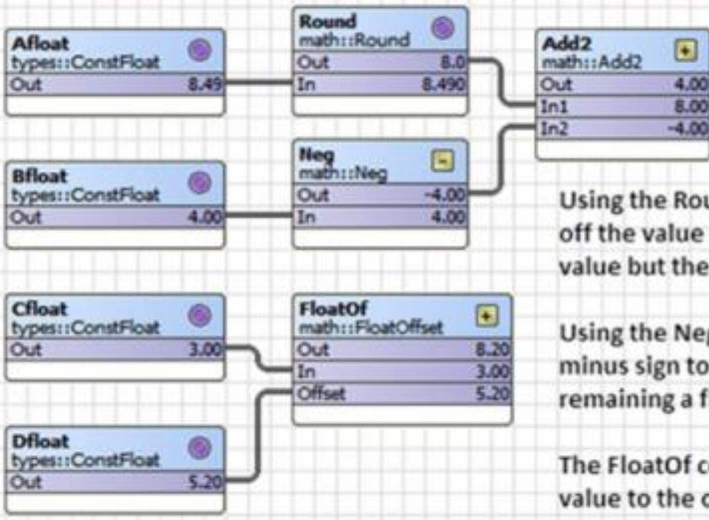
The Max component output (Out) reflects the maximum value of the two input floats (In1, In2) while the Min component reflects the minimum value of the two inputs.



To demonstrate this operation, an IRamp was configured to generate a triangle wave with a minimum value of 4 and a maximum of 8. The MinMax component captured the limits. Notice the need of an Integer-to-Float converter.

The MinMax component is a bit more complex. There is only one input and two outputs. If R is held in the true state, the two outputs simply reflect the input state. If R is false, the Min Out captures the lowest value of the input while Max Out captures the maximum of the input. When connecting up the component for the first time you should reset the component.

Rounding Off Floats

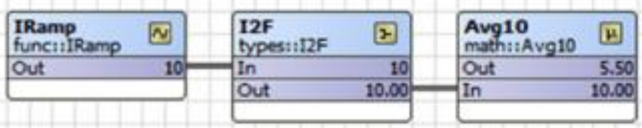


Using the Round component you can round-off the value of a float to the closest integer value but the output will remain a float.

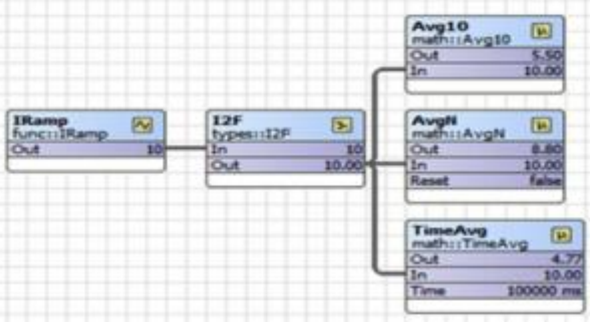
Using the Neg component you can append a minus sign to a float with the output remaining a float.

The FloatOf component appends an offset value to the output. It is not necessary to use a constant component for establishing the offset amount. The offset amount can be configured within the FloatOf component.

Averaging Successive Readings

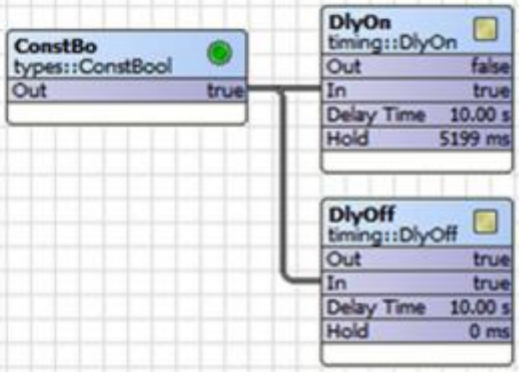


The Avg10 component averages the last ten input values and sends the result to the output. To demonstrate this, an IRamp is configured to create a triangle wave with a minimum value of 0 and a maximum value of 10. Increments are set to 1. When the IRamp reaches 10, the Avg10 component would have averaged 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 for a value of 5.5 which appears in the output.



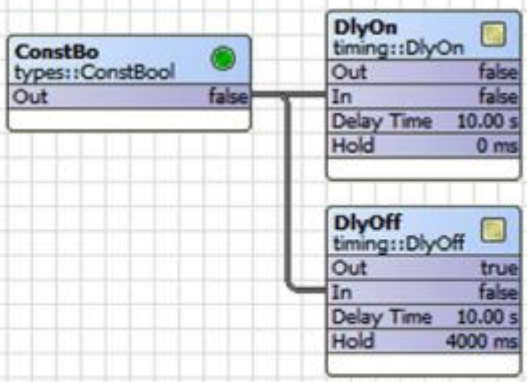
In this example three averaging components are compared. The Avg10 averages over 10 samples but the data must change to trigger a new sample. The AvgN component can be configured for the the number of samples but it samples every scan and not on just a change in value. The TimeAvg averages over a fixed period of time which is configurable. The output does not change until all samples are obtained.

On-Delays and Off-Delays



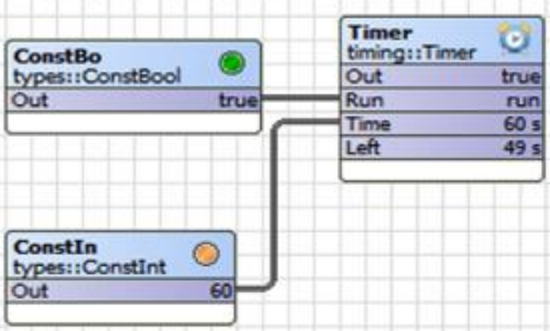
The DlyOn component is an on-delay timer which begins timing on the false to true transition of the input. Once the time (as shown in the Hold slot) goes to 0 the output will become true. The delay time is configurable. In this example the delay timer is still timing after the input went true 4.8 seconds ago.

The DlyOff component operates the same except it is triggered by a true to false transition of the input.



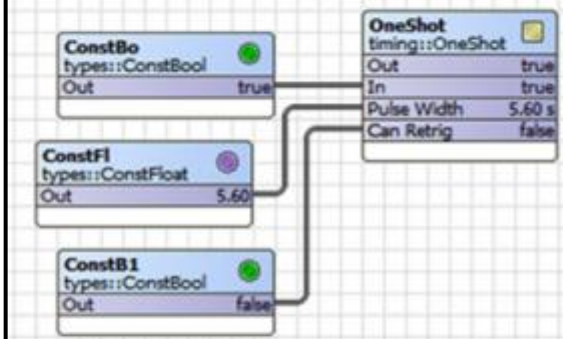
The input to the two timers made a true to false transition 6 seconds ago. The DlyOn component had immediately transitioned from true to false with the input but the DlyOff timer is still timing. In another 4 seconds its output will become false.

Using the Timer

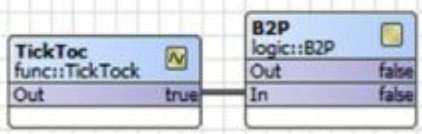


The Timer component will count down from a predetermined amount when the Run input is true. A constant integer component was used to set the time although the Timer component can be internally configured. The output will remain true during timing and transition false upon completion or if the Run input goes false. To begin a new timing period, the Run input must be cycled.

Using One-Shots — Mono-Stable Multivibrators

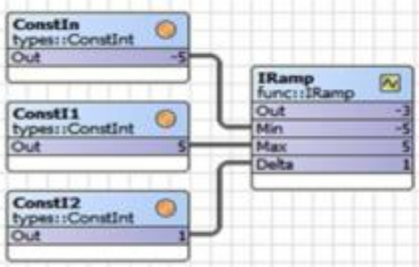


The OneShot provides a single pulse of determined value upon the false to true transition of the input signal. The output immediately goes true on its input's false to true transition and then returns to false when timing is complete. The Pulse Width can be configured or externally programmed as shown. A re-triggerable one-shot will renew timing if the input transitions from true to false to true during the timing period. A non-retriggerable will not. Notice that the Pulse Width is a float.

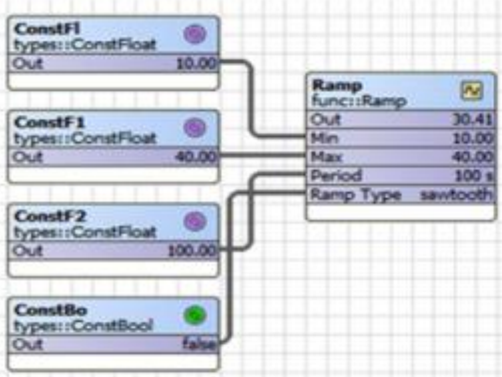


The Boolean-to-Pulse (B2P) converter is actually a very simple single-shot in that it outputs a true for only one scan time when its input goes from false to true. There are no time settings. It is used when a pulse is required after detection of an event instead of a logic level.

Creating Ramps — A-Stable Multivibrators

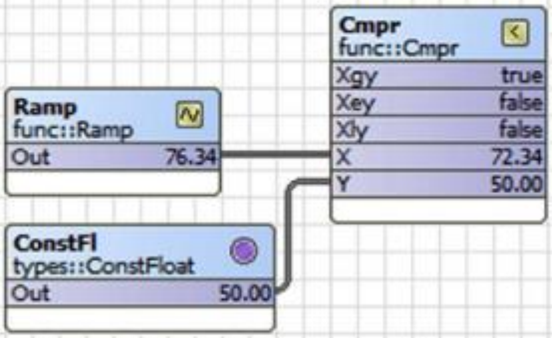


The IRamp provides a triangular output ranging from Min to Max with increments of Delta. These parameters can be configured or programmed as shown. The time increment must be configured having the units of seconds. Notice that all the configurable settings are integers as is the output.



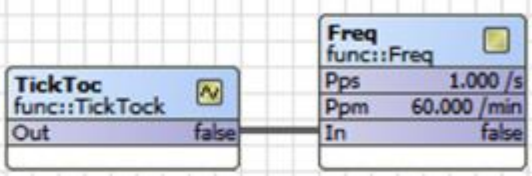
The Ramp is similar to the IRamp but the Ramp has mostly float settings and a float output. The output of the Ramp can be configured or programmed for either a sawtooth or triangle wave. Increasing the period slows down the speed of the ramp within the limits of Min and Max. Notice that although the Period is a float, the input to Period will be rounded to the nearest integer.

Comparing Two Floats



The comparator component (Cmpr) compares the the X input to that of the Y input. If X is less than Y, then the Xly output is true. If X equals Y then Xey is true. If X is greater than Y then Xgy is true. Both inputs are floats and the outputs are Booleans. In this example the output of the Ramp is compared to that of a constant. Using the default values of the Ramp, the input X varies as a triangle between 0 and 100 every 10 seconds. You can watch how the comparator outputs change over this range.

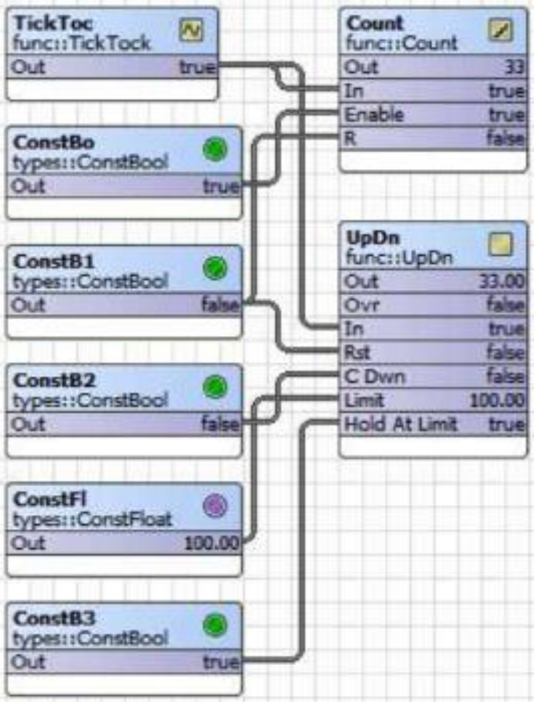
A Simple Clock — the TickToc



The TickToc component provides a convenient clock from 1 to 10 pulses per second. However, because of the controller scan time and other processing overhead it is recommended to use its default value of one second. More accurate timing is available from a real-time clock.

The Freq component can provide output values in pulses-per-second (Pps) or pulses-per-minute (Ppm). Because of the low-speed nature of these two components, the Ppm calculation will probably be the most useful.

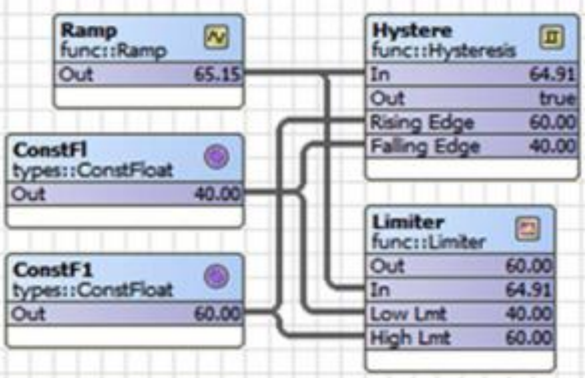
Introducing Counters



There are two counters. Count is an up/down counter with an integer output. It must be enabled to count. It can be reset to zero or preset to a postive value as long as the Enable pin is true. The direction pin (Dir) can be connected for programmable up/down counting.

UpDn is an up/down counter with a programmable direction input (C Dwn) which can also be configured. Although counters are inherently integer devices, the output of this component is a float. In this example a limit of 100 has been programmed. Once the limit is hit the overflow bit (Ovr) will be set. If Hold At Limit is true, the counter will not go pass 100. If it is false, the counter will continue to count past the limit but the overflow bit will remain set. Resetting the counter returns the component to the start position while clearing the counter and overflow bit.

Operating on Real-World Signals — Hysteresis and Limiting



The hysteresis component (Hystere) has separate rising-edge and falling-edge trip points when setting a trigger on a float variable. It is ideal for creating a digital event from a real-world analog input. Its output is Boolean.

The Limiter component restricts the range of a float variable by outputting a float that does not exceed the configurable low-limit (Low Lmt) or high-limit (High Lmt). The Limiter only limits the range of its output and does not scale the input float.

Handling Non-Linear Signals

The linearize component (Lineari) operates on a float input and creates a piece-wise linear representation of a non-linear input (such as a thermistor) or it can create a non-linear piece-wise representation of a linear input. There is complete flexibility in defining the ten X,Y coordinates along the output curve. The component determines the approximate output between the ten coordinates using linear interpolation.

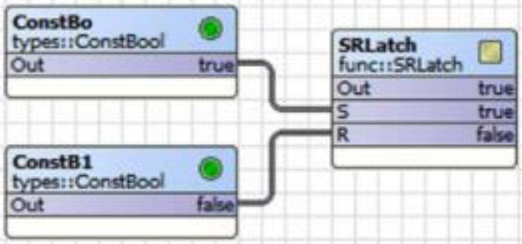
Handling Non-Linear Signals (cont)

Meta	Group [1] »
Out	56.50
In	7.50
X0	0.00
Y0	0.00
X1	1.00
Y1	1.00
X2	2.00
Y2	4.00
X3	3.00
Y3	9.00
X4	4.00
Y4	16.00
X5	5.00
Y5	25.00
X6	6.00
Y6	36.00
X7	7.00
Y7	49.00
X8	8.00
Y8	64.00
X9	9.00
Y9	81.00

In this example we will do the reverse of what is commonly done. We will use a linear input and create a non-linear output that approximates the equation Y=X*X over the range of X values from 0 to 9. We need to input corresponding values of Y that obey the desired equation. To make it easy we will use integer values but this is not a restriction. For example, the square of 4 is 16 and the square of 5 is 25. We enter the X values as an independent variable and then the Y value as the dependent variable. We need to be careful that the input does not exceed 9 in this example because we do not define a corresponding value for Y above 9.

You can test the interpolation by entering a value for X in the In slot assuming no link is connected to the linearize component. This is done here. Notice that the result is 56.50 for an input value of 7.5. The correct value would have been 56.25 which is very close.

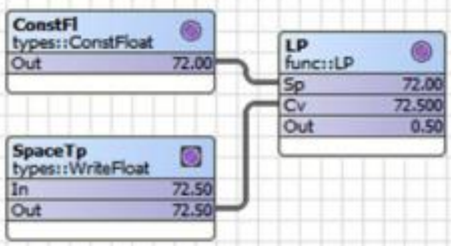
Simple Set-Re set Flip Flop — Bi-Stable Multivibrator



The SRLatch appears to be straight-forward logic block. The output would become true if the set (S) pin is high and would go low if the reset (R) pin goes high. However, both the S and R pins are positive leading-edge sensitive. Regardless of their steady-state condition, the output (Out) will only change on the false-to-true transition of either input. If this occurs on the S pin the output goes high and will remain high until the R pin does its transition.

On the rare condition that both S and R transition from false-to-true during the same logic scan, R will take precedence because its state is tested last in the logic and therefor the output will be false.

The Loop Component — Basic PID Controller



The LP or loop component is one of the most complex components. It can provide three modes of control P-proportional, I-integral, and D-derivative. In this example we will assume a temperature loop with a setpoint (Sp) of 72 degrees and a controlled variable (Cv) currently at 72.5 degrees which is the space temperature which we want to control.

Enable must be configured true otherwise there is no control.

Kp is the proportional gain which defaults to 1. Notice that the error signal is Cv-Sp or 0.5. The error multiplied by the proportional gain of 1 yields an output of 0.50. If the Ki and Kd factors are used, their contributions are also multiplied by the proportional gain factor. Ki is the integral gain in units of resets per minute. It is multiplied by the error signal. Kd is the derivative gain in seconds and it is also multiplied by the error signal.

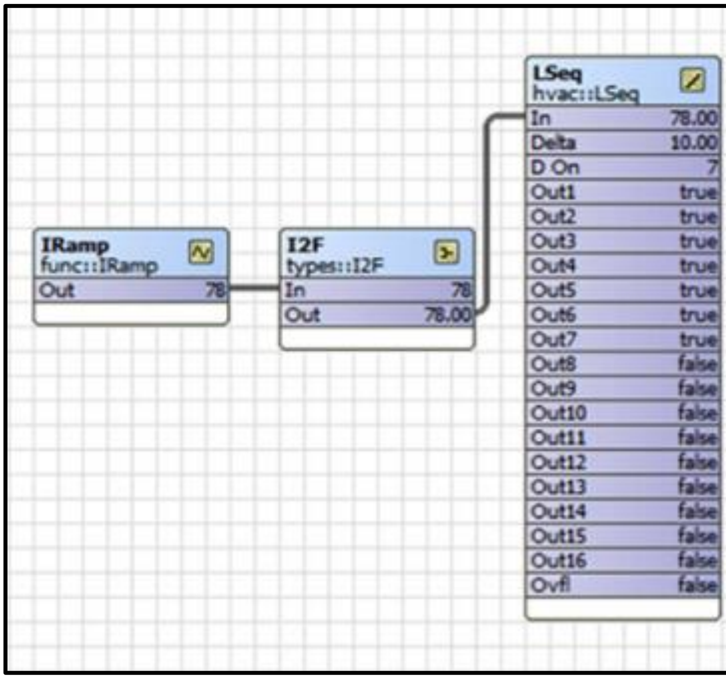
Min and Max are the limits of the output signal. They can be set to any value. Bias can offset the output regardless of the error. Max Delta sets the rate of change of the output within the output limits. This will slow the output swing.

Bias only applies to proportional-only (P) control. When using a PI controller, reset-windup can be minimized by limiting the output range.

For a cooling application set Direct to true. For heating set it for false.

The loop equation is solved each execute time (Ex Time) in milliseconds.

Linear Sequencer — Bar-Graph Representation of a Float

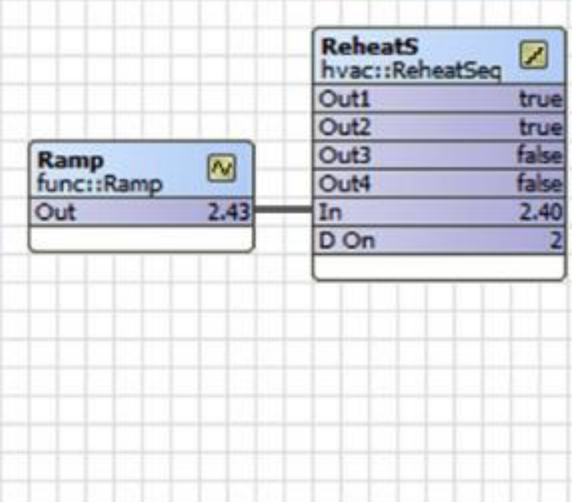


The linear sequencer (LSeq) provides a digital representation of an input float similar in operation to a bar graph on audio equipment. It is easier to understand its operation using an integer input. There are 16 possible Boolean outputs plus one overflow (Ovfl) flag. The input ramp provides a triangle wave from 0 to 100. The sequencer was configured for a 0 minimum input and a 100 maximum input. The maximum number of outputs was configured for 9 yielding a Delta of 10.

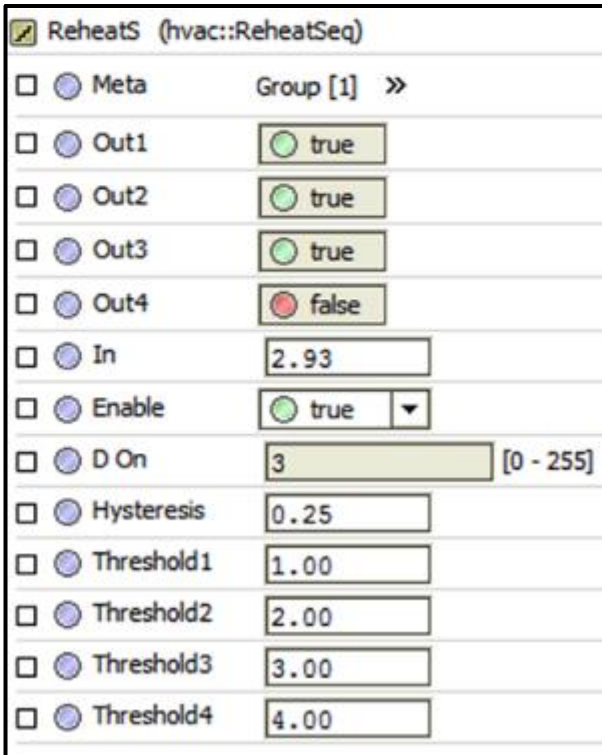
The range of the linear sequencer is configured using In Min at the low-end and In Max at the high-end. Selecting the number of outputs (Num Outs) determines the difference (Delta) between successive outputs turning on. In this case the range is 100 and the number of desired outputs is 9. Divide 100 by Num Outs + 1 and you will get a Delta of 10.

You will notice that the input (In) is at 60 and D On is indicating that six outputs are on. With an input between 0-9, there are no outputs on but once you hit a decade such as 10, 20 on up to 90, successive outputs will come on. At the maximum of 100, 9 lights will be on. If the input exceeds the maximum intended, the overflow flag will set but the number of outputs will remain as specified by Num Outs.

Reheat Sequencer — Four Staged Outputs from a Float Input



The reheat sequencer (ReheatS) provides a linear sequence of up to four outputs based upon the input float (In). The threshold for the four outputs can be configured for increasing values of the input. As the input increases to each threshold, the corresponding output will go on. As the input decreases below the threshold, the corresponding output will remain on until the Hysteresis value is exceeded.

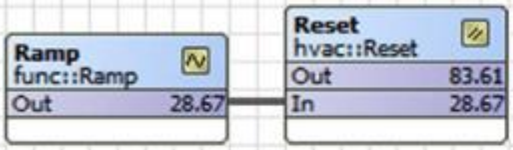


Enable must to true otherwise the outputs to be false.

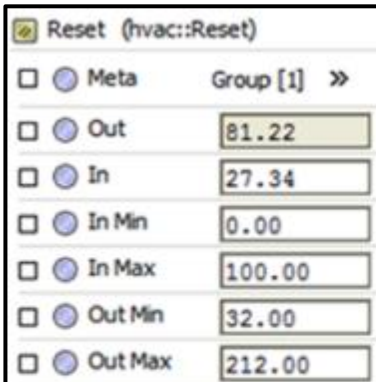
There are four possible threshold settings corresponding to four outputs. As the input signal increases to each threshold its corresponding output goes on and stays on until the input drops below the threshold plus the value of the hysteresis.

The input signal is decreasing but it has not exceeded the amount of the threshold so output 3 (Out3) remains set. Once the signal is below 2.75, output 3 will go off.

Reset — Scaling a Float Input between Two Limits

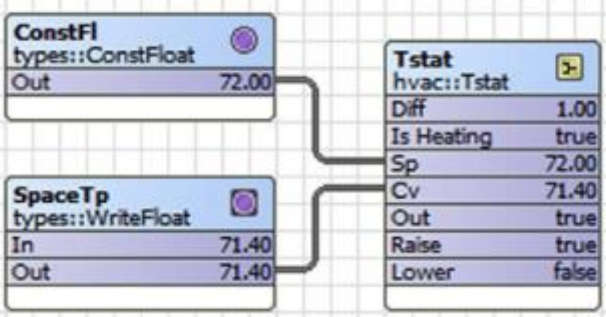


The reset component (Reset) will scale the output linearly between two limits. The input range must be configured by setting In Min and In Max. The corresponding output for those two points must be configured as Out Min and Out Max. If the input signal exceeds the defined input range, the output will be clamped to one of the two output limits.



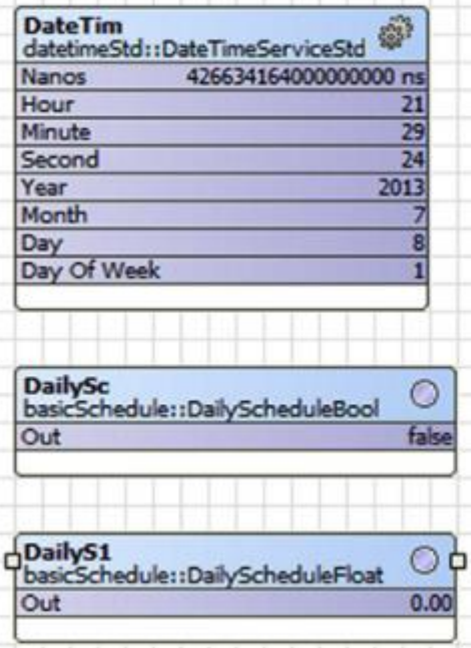
In this example we are converting degrees Celsius to degrees Fahrenheit within the 0-100 degree Celsius range. Therefore we set Out Min and Out Max to the corresponding Fahrenheit values. All Celsius input values between these two limits will be interpolated thereby providing the correct Fahrenheit values.

Tstat — Basic On/Off Temperature Controller



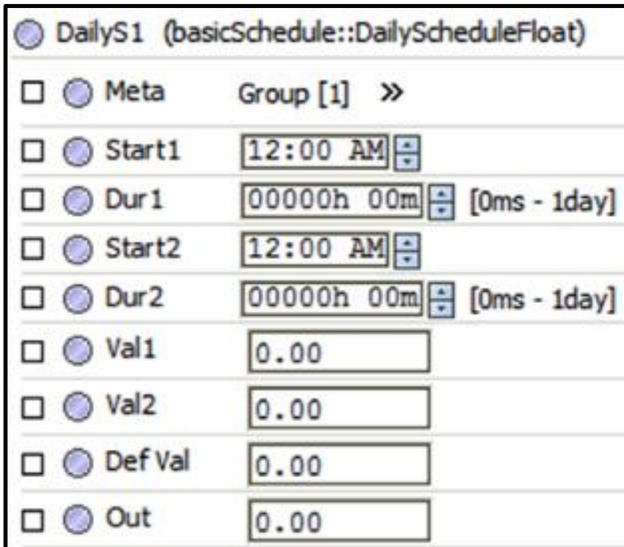
The Tstat is an on/off temperature controller for either heating or cooling. For heating configure the Is Heating bit to true. The deadband can be set by the Diff value. If the controlled variable (Cv) deviates from the setpoint (Sp) by half the Diff value, the output (Out) will become true and stay set until Cv deviates from the setpoint by a like amount in the other direction. In this way Diff also provides hysteresis. The Raise and Lower outputs are a function of the Is Heating setting. If Is Heating is true, Out=Lower, otherwise Out= Raise.

Real-Time Clock and Scheduling



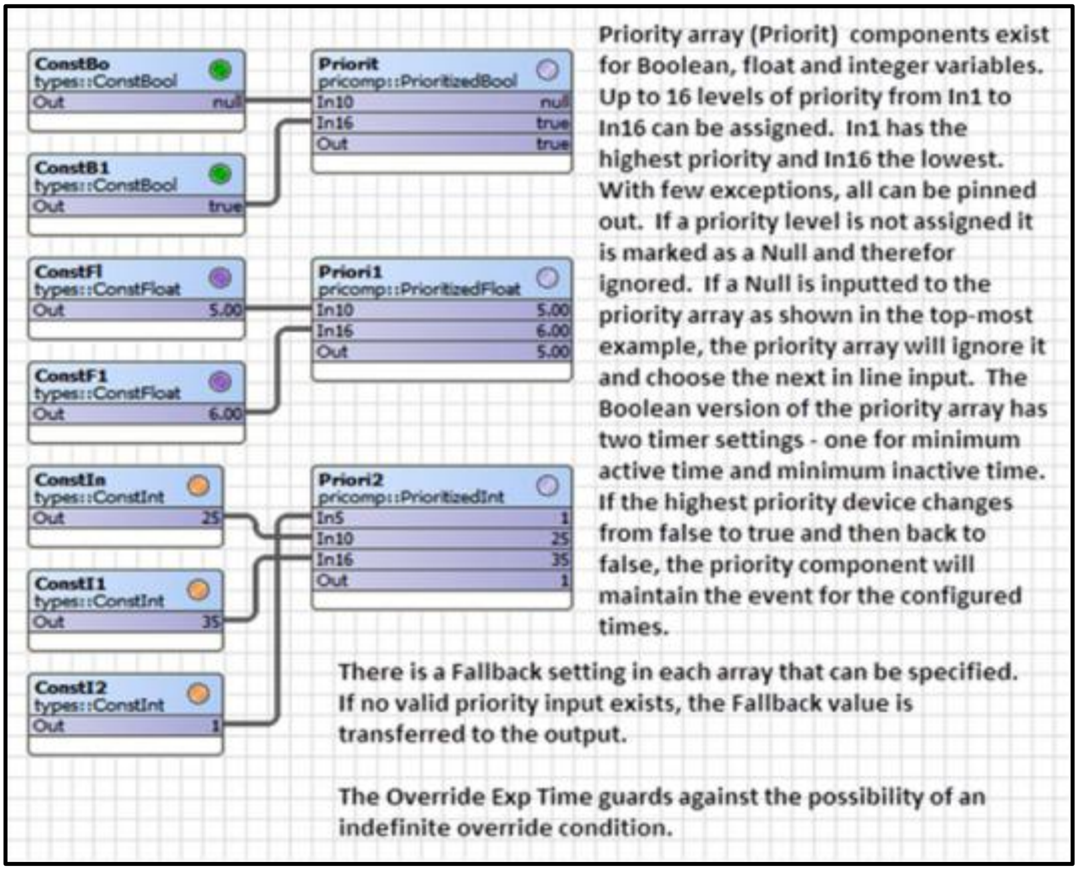
The DateTim component provides real-time information. There is no need to place it on the wiresheet. However, if you need specific information from the component for driving logic, you can connect to the various integer outputs such as Hour, Minute and Second.

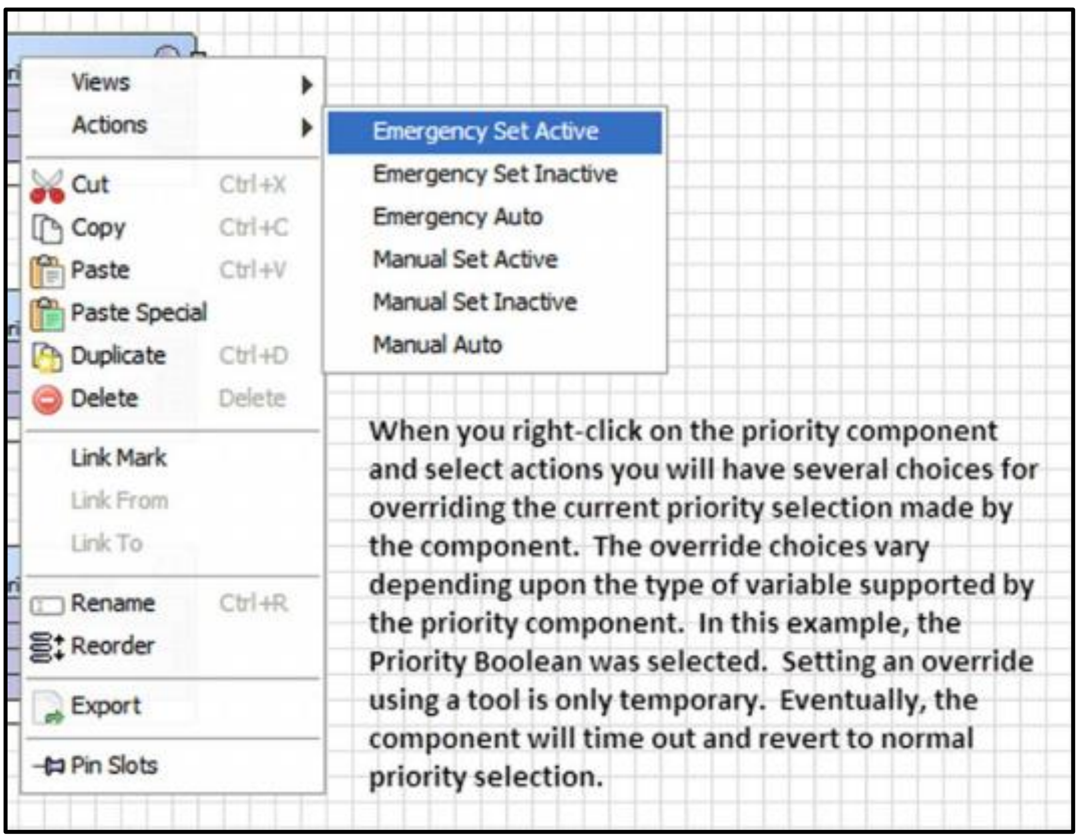
There are two schedule components which have different output types. One is for Boolean and the other for float. There is no need to connect the DateTim component to either of the schedulers. Each scheduler can handle two events over the 24 hour period by configuring the time and duration of each event. The output of each schedule will change with each event. If more events or more outputs are needed, multiple schedulers can be placed on the wiresheet.



Configuration of the two scheduler components is similar. For the float version, Val1 and Val2 need to be specified along with the start times (Start1 and Start2) and the durations (Dur1 and Dur2). The output (Out) will assert either Val1 or Val2 during the scheduled times. If neither are programmed, the Def Val should be configured.

Priority Arrays





E.1 BAScontrol20 Firmware Release 3.1

The BAScontrol20 is a 20-point BACnet/IP Sedona Field Controller ideal for unitary control applications. It is considered an "open controller" in that it supports both BACnet/IP and Sedona Framework (SOX) protocols. It complies with the BACnet B-ASC device profile having eight universal inputs, four binary inputs, four analog outputs and four binary outputs (relay or triac). No licensing is required to purchase or use the product. It is a freely-programmable controller executing Sedona's drag-and-drop methodology of assembling components onto a wire sheet to create applications. It can be programmed using Niagara Workbench or a third-party Sedona programming tool or configured for BACnet/IP remote I/O applications using a common web browser. Release 3.1 will be shipped with new BAScontrol20 orders with no change in product pricing.

To complement the standard Tridium-developed Sedona 1.2 components that reside in the unit, Contemporary Controls has developed more than 100 custom Sedona components. Unique to the BAScontrol20 are 48 Web Components that allow wire sheet data to be read and written from a common web browser. Besides the 20 real I/O points, 24 virtual points on the wire sheet can be read or written by a BACnet client. A new hardware-independent CControls Function Kit provides additional logic elements for expanded functionality along with sophisticated Psychrometric components.

Contemporary Controls has developed a free Sedona Backup and Restore utility called BASbackup that allows the system integrator the ability to completely backup a Sedona project including wire sheet, web configuration, BACnet configuration, and kits without the need of the workbench tool.

The current version firmware on the BAScontrol20 is 3.0 and the new version is 3.1. With this release are new kits that can be easily installed on Workbench and BASbackup as a single bundle. The new kits support both 3.0 and 3.1 controllers. Some minor issues may exist for moving 3.0 programs over to 3.1 controllers but they can easily be resolved. There are no hardware changes on the BAScontrol20 and it is possible to re-flash existing controllers in the field by first contacting Contemporary Controls' technical support. What follows is a list of new features in version 3.1.

1. Virtual points increased from 8 to 24 points

Virtual points are wire sheet components that function as network variables in that they can be read by or written to from a BACnet client as a binary variable (BV) or an analog variable (AV). Since they are wire sheet components, they should be configured as wire sheet inputs or wire sheet outputs by the Workbench tool and not by web pages. However, BACnet configuration continues to be accomplished with web pages. Virtual points are now tagged VT01-VT24 and they have their own web page where the status of these points can be viewed and forced without the need of a Workbench tool.

2. Universal Input options expanded

Currently, the BAScontrol20's universal inputs support analog inputs, contact closure inputs, type II and type III 10kΩ thermistors and pulse inputs. More flexibility has been achieved to universal inputs with the addition of 20kΩ thermistor range and the ability to read resistance.

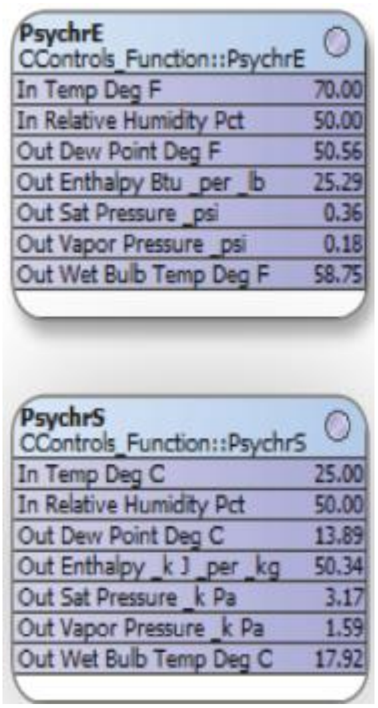
It is now possible to read 2-wire potentiometers from set point stations. An input choice called "resistance" has been added that can read a passive resistance in the 1-100 kΩ range. To accommodate set point face plates with linear graduations, it is recommended that the Sedona Linearize component is used to convert the non-linear resistance measurement to match the face plate readings.

Another change made was detection of an open thermistor, which may produce an indeterminate state from the universal input component. The systems integrator is able to assign a default output to the universal input if an out-of-bounds situation occurs. In addition, a flag is set using the binary output of the universal input to provide an indication of this fault condition.

3. CControls Function Kit added

A new Function Kit expands the choice of AND, OR, NAND and NOR logic along with providing improved latching registers. The use of complementary outputs within the component ensures predictable logic execution.

Using dry-bulb and relative humidity (RH) as inputs, the Psychrometric components will output saturation pressure, vapor pressure, enthalpy, dew-point and wet-bulb temperatures. Two components exist – one for English and one for SI units.



4. Additional Change-of-Value (COV) subscriptions

With version 3.0, the BAScontrol20 supported two binary and two analog subscriptions. Looking at recommended ASHRAE air-handler sequences it was decided to increase the number of binary subscriptions to 14 giving the 3.1 controller a total of two analog and fourteen binary subscriptions.

5. Increased performance and larger application memory space

With single-chip microcontrollers, there is always concern for sufficient RAM and ROM space. The BAScontrol20 is BACnet/IP compliant with a B-ASC profile. It has a resident Sedona Virtual Machine (SVM) with an application program (app.sab) stored in flash memory but executes the application out of RAM. It is the RAM space that is critical so every buffer memory space was studied to free up as much RAM as possible. More RAM had to be provided for increased COVs and virtual points but gains were made in other places thus providing a net 6 kB gain in memory space. This allows for at least a 200 Sedona component wire sheet.

6. More informative web pages

Much of the configuration of the BAScontrol20 is via web pages. With the addition of 16 more virtual points, it was decided to move all 24 virtual points to a separate page. Configured virtual points will now show the BACnet name up to the limit of the display along above the value of the point. The virtual point tag just to the left of the point value will remain unchanged. However, by hovering over the tag it can be learned if the point is configured as a “Read from Wire Sheet” or a “Write to Wire Sheet.” The points that are placed on the wire sheet will have their tags shown with the color green indicating that they are active and are available for communicating to a BACnet client. The VT01-08 points are stored in persistent memory and will be saved during power outages less than seven days. The VT09-24 points are not in persistent memory.

All 20 physical input/output points appear on the main web page. If the I/O component has been placed on the wire sheet, the point tag will turn green. Hovering over the point tag will verify the type of configuration and hovering above the point value will show the BACnet name truncated to fit the space. This is especially helpful in understanding the configuration of universal inputs.

Web components are unique to the BAScontrol20 providing a means to set parameters on a wire sheet or for reading parameters from a wire sheet using a common web browser.

A total of 48 web components exist and limits can be placed on those components that are configured as inputs to the wire sheet. These minimum and maximum values are set in the web component but the values can be viewed on the web components web page. Limit values associated with web components configured as outputs are ignored.

7. Network Time Server can be found by domain name

If Internet access is possible, the BAScontrol20 will have it time set from a pool of NTP servers instead of relying upon one fixed IP address. With version 3.1, domain naming services (DNS) is supported with the opportunity to make two DNS entries. It is recommended to use the domain name pool.ntp.org as the time server assuring a server will be found. Daylight Savings Time (DST) continues to be supported and if time is to be maintained in the absence of an Internet connection, time can be set manually and it will be backed up for up to seven days upon a loss of power.

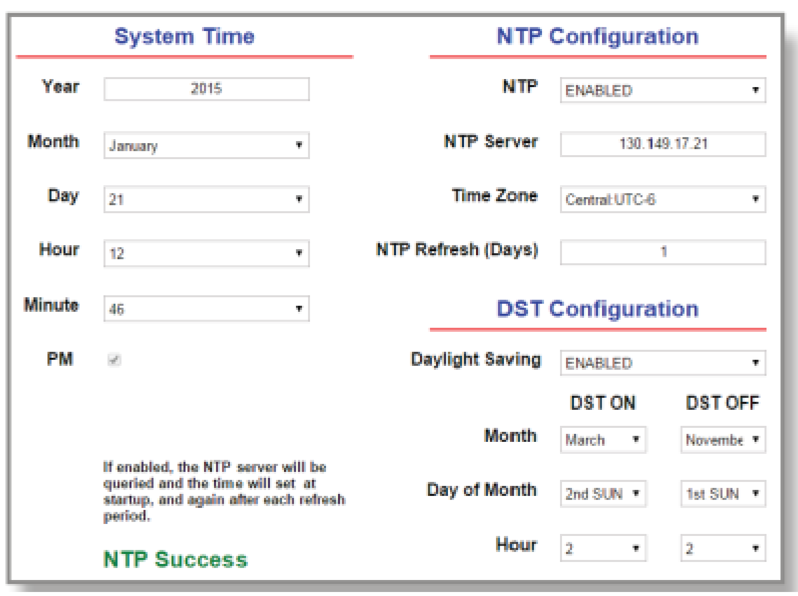
8. Improved Universal Counter (UC) component

The UC component differs from the two other Sedona counters in that its count output is retained up to seven days in persistent memory which is ideal for run-time calculations. The UC component has been designed to meet or exceed the capabilities of the volatile Sedona counters.

9. The BASbackup utility is easier to use and not dependent upon the Workbench tool

BASbackup is a Java program that allows the system integrator to completely backup and restore a Sedona project including wire sheet, web configuration, BACnet configuration, and kits into one zip file without the need of the Workbench tool. The program is free and it no longer needs access to the Component Bundle residing in Workbench.

Although the BAScontrol20 is a freely-programmable controller, it can be used as a configurable controller by loading in a Sedona application and configuring the application using just configuration web pages. The use of the 48 Web Components makes this possible greatly increasing the flexibility of the controller to adapt to either a freely-programmable or configuration-only environment. Contemporary Controls continues to develop wire sheet applications and components for its customers.



United States
Contemporary Control Systems, Inc.
2431 Curtiss Street
Downers Grove, IL 60515
USA

Tel: +1 630 963 7070
Fax:+1 630 963 0109

info@ccontrols.com
www.ccontrols.com

China
Contemporary Controls (Suzhou) Co. Ltd
11 Huoju Road
Science & Technology Industrial Park
New District, Suzhou
PR China 215009

Tel: +86 512 68095866
Fax: +86 512 68093760

info@ccontrols.com.cn
www.ccontrols.asia

United Kingdom
Contemporary Controls Ltd
14 Bow Court
Fletchworth Gate
Coventry CV5 6SP
United Kingdom

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

info@ccontrols.co.uk
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
www.ccontrols.eu

TD100700-0MB – June 1, 2015