Honeywell

# CIPer Model 30 Controller

## SYSTEM ENGINEERING USER GUIDE

JUNE 2021

# Disclaimer

The material in this document is for information purposes only. The content and the product described are subject to change without notice. Honeywell makes no representations or warranties with respect to this document. In no event shall Honeywell be liable for technical or editorial omissions or mistakes in this document, nor shall it be liable for any damages, direct or incidental, arising out of or related to the use of this document. No part of this document may be reproduced in any form or by any means without prior written permission from Honeywell.



Niagara Framework® is a registered trademark of Tridium Inc.

# Trademark Notice

BACnet and ASHRAE are registered trademarks of American Society of Heating, Refrigerating and Air-Conditioning Engineers. Microsoft, Excel, Internet Explorer, Windows, Windows Vista, Windows Server, and SQL Server are registered trademarks of Microsoft Corporation. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Mozilla and Firefox are trademarks of the Mozilla Foundation. Echelon, LON, LonMark, LonTalk, and LonWorks are registered trademarks of Echelon Corporation.

Niagara Framework ® is a registered trademark of Tridium Inc.

WEBs-AX, WEBs-N4 and WEBStation are registered trademarks of HONEYWELL International, Inc.

# Document History

| Version | Revision | Published | Comments |
|---|---|---|---|
| 31-00237 - 03 | 2021-03-29 | March 2021 | ▪ RSTP Configuration<br>▪ Switch Port Configuration |
| 31-00237 - 03 | 2021-05-12 | May 2021 | ▪ USB Support |

# Table of Contents

# About This Guide

This document serves as a guide to configure and use the Honeywell CIPer Model 30 programming model. Released versions of the tool include a complete collection of technical information that is provided in both online help and PDF format. The information in this document includes basic descriptions and concepts as well as reference information, to help Systems Integrators and Engineers use the CIPer Model 30 programming model. To make the most of the information in this guide, readers should have some training or previous experience working with Honeywell WEBs controllers, as well as Niagara 4 or Niagara AX software.

## Other Related Documents

- CIPer 30 Expansion IO Installation Instruction - 31-00319
- CIPer 30 Expansion IO Product Data - 31-00239
- CIPer Model 30 Installation and Operation Guide - 31-00206
- CIPer Model 30 Installation Instruction - 31-00183
- CIPer Model 30 Product Data - 31-00236EFS
- CIPer Model 30 Hardening Guide - 31-00207
- CIPer Model 30 Quick Setup Guide
- Software Release Bulletins
- Niagara 4 Installation Guide

For more details about CIPer Model 30 controller, refer to The Honeywell Buildings Forum.

## Abbreviations

| Abbreviation | Full Form |
|---|---|
| CIPer | Internet Protocol Controller |
| VAV | Variable Air Volume |
| I/O or IO | Input / Output |
| UI/AO | Universal Input Output |
| AO | Analog Output |
| DO | Digital Output |
| AI | Analog Input |
| DI | Digital Input |
| HTTP | Hypertext Transfer Protocol |
| LAN | Local Area Network |
| DHCP | Dynamic Host Configuration Protocol |
| RSTP | Rapid Spanning Tree Protocol |
| STP | Spanning Tree Protocol |
| DNS | Domain Name System |
| UTP | Unshielded Twisted Pair |
| HOA | Hand Off Auto |
| AIA | Adaptive Integral Action controller |

| Abbreviation | Full Form |
|---|---|
| PID | Proportional Integral Derivative |
| tr | throttling range |
| CPH | Cycles per Hour |
| A | Ampere-Current Unit |
| Hz | Hertz-Frequency Unit |
| GB | Gigabytes |
| TUNCOS | Time Until Next Change of State |

## About CIPer Model 30 Controller

CIPer Model 30 is a unitary IP edge controller (interchangeably called IPC or ipc) designed by Honeywell.

The controller device series is WEB-C3036. It is designed for Variable Air Volume (VAV), Unitary and plant applications. You can use the CIPer Model 30 controller for aggregating the real-time information—alarms, trends, and history. In future, the controller will also be able to further integrate the aggregated information to the Sentience Cloud for value-added data analytics.

## Features

- Native Niagara N4 for faster programming, installation, and commissioning
- No tools needed for installation of CIPer Model 30 controller mounting
- Spyder to CIPer (Model 30) Migrator Tool
- Integrate a a variety of IP devices (cameras, access control, etc.)
- Scalable can expand I/O count to 15 additional expansion modules
- 4-port IP unmanaged switch for maximum flexibility
- Utilizes lower cost CAT5 or CAT6 IP cabling
- Faster installation using pre-terminated CAT5 or CAT6 cables
- On-board H-O-A switches for easy commissioning
- VAV model includes an on-board differential pressure sensor
- Digital outputs can drive 1.5A Continuous current, 3.5A inrush current for 100 mS
- Niagara N4 License included, good for 150 external points and 3 devices. Expandable
- 1 Gigabit per second (Gbps) IP switch supports demanding IP peripherals such as color cameras
- On-board programming platform
- Web-serving capability
- Integrated Control: The web server and controller have a combined package in the CIPer Model 30 controller
- Data logging
- Alarming
- Trending
- Schedule management

# CIPer Model 30 Controller Details

There are four variants in CIPer Model 30 controller.

- **WEB-C3036EPUBNH** (UNITARY)
- **WEB-C3036EPVBNH** (VAV)

The following table describes the CIPer Model 30 controller pin slots and its meaning.

Table 1: WEB-C3036 Series Pin Slots

| Attribute | Variants | |
|---|---|---|
| | WEB-C3036EPUBNH | WEB-C3036EPVBNH |
| Brand Identifier | WEB | WEB |
| Controller | C | C |
| Analog Inputs | 3 | 3 |
| Digital Inputs | 0 | 0 |
| Analog Outputs | 3 | 3 |
| Digital Outputs | 6 | 6 |
| Ethernet | E | E |
| Programmable | P | P |
| Unitary vs VAV | U | V |
| BACnet | B | B |
| No Actuator | N | N |
| Hand Off Auto | H | H |

# Expansion I/O Model Number Details

The CIPer Model 30 controller is compatible with two expansion or external I/O models.

- **WEB-O9056H** (large expansion)
- **WEB-O3022H** (small expansion)

Table 2: I/O Models in CIPer Model 30

| Attribute | Model | |
|---|---|---|
| | WEB-O9056H | WEB-O3022H |
| Brand identifier | WEB | WEB |
| Expansion I/O Module | O | O |
| Analog Inputs | 9 | 3 |
| Digital Inputs | 0 | 0 |
| Analog Outputs | 5 | 2 |
| Digital Outputs | 6 | 2 |
| Hand Off Auto | H | H |

# License Limits

The Honeywell CIPer Model 30 controller comes with features that are license-controlled.

- CIPer Model 30 controller (UNITARY/ VAV) has 150 points Niagara N4 license. You cannot use the CIPer Model 30 programming model with other devices other than CIPer Model 30 controller. 150 points also include data sharing points from third party BACnet devices using BACnet devices.

- Expansion I/O device (WEB-O9056H and WEB-O3022H) supports 15 devices and additional license for 50 I/O points.
- Sylk device limitation is 14. See License Requirements and Behaviors
- Function Block maximum limitation is 5000. You can add more than 5000 blocks till there is memory in CIPer device, but the controller may take more than 1 second execution time when more than 5000 blocks are added in the Sequenced Control Program.

| Note: |
|---|
| To validate the license that you have, see Managing License section under Managing Software and Licenses. |

## Hardware Input Specifications

Universal Inputs (configurable): 3 UI

3 UI/AOs configurable as UIs

Differential pressure sensor range (VAV model): 0-2" WC (0 to 374 Pa) 32 to 122F (0 to 55C)

Pulse Inputs: 100Hz max, minimum duty cycle: 5 mS ON / 5 mS OFF

Flexible UI's to connect external sensors like 20KNTC, PT1000, and other resistive sensors.

Range (voltage/current):

- Rated voltage: 20-30 VAC; 50/60Hz
- WEB-C3036EPUBNH /WEB-C3036EPVBNH controller power consumption (AC):
  - 50 VA maximum for controller only load
  - 100VA maximum for controller and all connected Loads
- WEB-O9056H: 35VA
- WEB-O3022H: 15VA

For example, you can configure temperature sensor C7041N2020/U as 20 KNTC in UI/UI/AO port.

## Hardware Output Specifications

- **Analog Outputs (configurable):** 3 UI/AOs configurable as AO
- **Digital Output voltage rating**: 20 to 30 VAC @ 50/60 Hz
- **Digital Output current rating:** Solid-State Relay, 1.5A Continuous, 3.5A Inrush for 100 mS.

For example, you can configure damper actuator AFB24-MFT N4H as 20 KNTC in UI/UI/AO port.

# Available Features

**NOTE**

To get better performance with the below listed features, make sure the CIPer Model 30 controller is up-dated to the latest compatible tool and software version.

| Number | Feature | Feature Details/Compatibility |
|---|---|---|
| 1 | IOs | • FloatingOutput<br>• PhysicalPoints<br>• ModulatingInput<br>• ModulatingOutput<br>• BinaryInput<br>• BinaryOutput-Direct<br>• BinaryOutput-SlowPWM<br>• BinaryOutput-Floating motor |
| 2 | Analog Functional Blocks | • Analog Latch<br>• Average<br>• Compare<br>• Decision Box<br>• Edge<br>• Encode<br>• Hysteretic Relay<br>• Maximum<br>• Minimum<br>• Priority Select<br>• Select<br>• Switch |
| 3 | Control Functional Blocks | • AIA<br>• Cycler<br>• Flow Control<br>• PID<br>• Rate Limit<br>• Stager<br>• Stage Driver |
| 4 | Logic Functional Blocks | • AND<br>• One Shot<br>• OR<br>• XOR |
| 5 | Math Functional Blocks | • Add<br>• Digital Filter<br>• Divide<br>• Enthalpy<br>• Exponential<br>• Flow Velocity<br>• Limit<br>• Logarithm<br>• Multiply<br>• Ratio<br>• Reset<br>• Square Root<br>• Subtract |
| 6 | Data Function Functional Blocks | • Counter<br>• Override<br>• Runtime Accumulate |
| 7 | Zone Control Functional Blocks | • General Setpoint Calculator<br>• Occupancy Arbitrator<br>• Set Temperature Mode<br>• Temperature Setpoint Calculator |
| 8 | BuiltIn | • ConventionalWallModule |
| 9 | Utils Functional Blocks | • Pass Thru<br>• Text Block<br>• SystemTime<br>• Tuncos |
| 10 | Sylk Parameters | • BypassTime<br>• CO2<br>• ControllerValue<br>• FanCommand<br>• Humidity<br>• Roomtemp<br>• SensorOffset<br>• SylkSchedule<br>• SylkTime<br>• SystemCommand |

| Number | Feature | Feature Details/Compatibility |
|---|---|---|
| | | • HomeScreen<br>• NetworkSetpoint<br>• OccupancyOverrideCommand<br>• OccupancyStatus<br>• SystemStatus<br>• TimeField<br>• TimeOfDay<br>• ValueFromWallModule<br>• SylkActuatorInputParam<br>• SylkActuatorOutputParam |
| 12 | Override Function Blocks | To view the list of function blocks that are overridden. |
| 13 | I P C Point Manager View | This view provides details information about the Physical Points associated with Local Device (CIPer Model 30 controller) and expansion IO Modules. |
| 14 | Wire Sheet View | To write the application logic. |
| 15 | Commission Sylk Devices View | This view provides information about the errors in the configuration of Sylk parameters. |
| 16 | RSTP Configuration via Workbench | Rapid Spanning Tree Protocol which facilitates with a loop-free topology for Ethernet/IP networks. |
| 17 | Network firewall configuration via workbench | Network security to prevent unauthorized access to or from private networks. |
| 18 | Niagara Provisioning Service | Provisioning service is used to run batch process for multiple CIPer Model 30 controllers at once. |

# Sylk Device Support

A brief view to the different wall modules supported in CIPer Model 30 programming model and the parameters that these modules support is as follows:

Table 3: Sylk TR120X Modules and Parameters

| Parameter | Module-TR120X | |
|---|---|---|
| | TR120BusWallModule | TR120HBusWallModule |
| ROOMTEMP | Y | Y |
| HUMIDITY | N | Y |
| OccupancyOverrideCommand | Y | Y |
| ValueFromWallModule | Y | Y |
| TimeOfDay | Y | Y |
| SystemStatus | Y | Y |
| OccupancyStatus | Y | Y |
| ValueFromController | Y | Y |
| SystemCommand | Y | Y |
| TimeField | Y | Y |
| BypassTime | Y | Y |
| SensorOffset | Y | Y |
| HomeScreen | Y | Y |
| NetworkSetpoint | Y | Y |
| SylkTime | Y | Y |
| FanCommand | Y | Y |
| SylkSchedule | Y | Y |

Table 4: Sylk TR7X Modules and Parameters

| Parameter | Module-TR7X | | | |
|---|---|---|---|---|
| | TR75HSBus-WallModule | TR75SBusWall-Module | TR71HSBusWall-Module | TR71SBusWall-Module |
| ROOMTEMP | Y | Y | Y | Y |
| HUMIDITY | Y | N | Y | N |
| OccupancyOverrideCommand | Y | Y | Y | Y |
| ValueFromWallModule | Y | Y | Y | Y |
| TimeOfDay | Y | Y | Y | Y |
| SystemStatus | Y | Y | Y | Y |
| OccupancyStatus | Y | Y | Y | Y |
| ValueFromController | Y | Y | Y | Y |
| SystemCommand | Y | Y | Y | Y |
| TimeField | Y | Y | Y | Y |
| BypassTime | Y | Y | Y | Y |
| SensorOffset | Y | Y | Y | Y |
| HomeScreen | Y | Y | Y | Y |
| NetworkSetpoint | Y | Y | Y | Y |
| SylkTime | Y | Y | Y | Y |
| FanCommand | Y | Y | Y | Y |
| SylkSchedule | Y | Y | N | N |

**Table 5: Sylk TR42X Modules and Parameters**

| Parameter | Module-TR42X | | | |
|---|---|---|---|---|
| | TR42HCO2SBus-WallModule | TR42HSBusWall-Module | TR42CO2SBus-WallModule | TR42SBusWall-Module |
| ROOMTEMP | Y | Y | Y | Y |
| HUMIDITY | Y | Y | N | N |
| CO2 | Y | N | Y | N |
| OccupancyOverrideCommand | Y | Y | Y | Y |
| OccupancyStatus | Y | Y | Y | Y |
| BypassTime | Y | Y | Y | Y |
| NetworkSetpoint | Y | Y | Y | Y |
| FanCommand | Y | Y | Y | Y |

**Table 6:Sylk TR40X Modules and Parameters**

| Parameter | Module-TR40X | | | |
|---|---|---|---|---|
| | TR40HCO2SBus-WallModule | TR40HSBusWall-Module | TR40CO2SBus-WallModule | TR40SBusWall-Module |
| ROOMTEMP | Y | Y | Y | Y |
| HUMIDITY | Y | Y | N | N |
| CO2 | Y | N | Y | N |
| OccupancyOverrideCommand | N | N | N | N |
| OccupancyStatus | N | N | N | N |
| BypassTime | N | N | N | N |
| NetworkSetpoint | N | N | N | N |
| FanCommand | N | N | N | N |

Table 7: Sylk Zeleny Modules and Parameters

| Parameter | Module- C7400S (Zeleny) |
|---|---|
| ROOMTEMP | Y |
| HUMIDITY | Y |

**Table 8: Sylk Actuator Modules and Parameters**

| Parameter | Module- Sylk Actuator |
|---|---|
| SylkActuatorInputParam | Y |
| SylkActuatorOutputParam | Y |

Y: Yes N: No

# Sylk Device Limits

The CIPer Model 30 controller does not support Sylk I/O, TR70, TR70H, and Zelix devices.

## Power and Grounding Requirements

The CIPer Model 30 controller requires 20-30 VAC, 50/60Hz. Power consumption is based on the sum of the VA rating for each controller and should not exceed 100VA. If additional modules are required, they must be powered from a separate transformer. Refer VA rating for each module in the following note.

**Note:**

Transformer VA load for module power only (no BO loads)

- WEB-C3036EPUBNH/WEB-C3036EPVBNH: 50VA
- WEB-O9056H: 35VA
- WEB-O3022H: 15VA

For more details on power and grounding, see CIPer Software Tool Installation and Operations Guide.

## Power Slaves

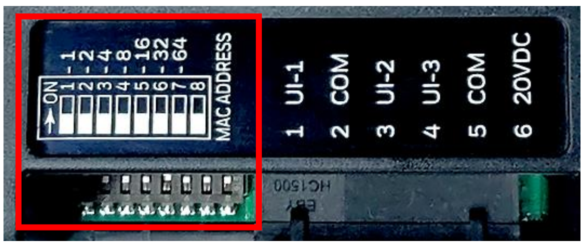
The following table lists all of the current Sylk Power Slave devices.

| Device | Base OS Numbers | Addresses Available | Device Max Power or Avg/Peak Current **Req't** | Total Bus Power or Avg/Peak Current | Normalized Value (Total=1.000) |
|---|---|---|---|---|---|
| Zio Plus/Enhanced | TR71<br>TR75<br>TR71-H<br>TR75-H | 1-10 | 230 mW | 950 mW | 0.242 |
| Zio Lite | TR40<br>TR40-H | 1-15 | 6.5 mA (avg cur) | 72 mA | 0.090 |
| Zio Lite with CO2 | TR40-CO2<br>TR40-H-CO2 | 1-15 | 17.5 mA (peak cur) | 96 mA | 0.182 |
| Zio Lite with Small Display | TR42<br>TR42-H | 1-15 | 122 mW | 950 mW | 0.128 |
| Zio Lite with Small Display with CO2 | TR42-CO2<br>TR42-H-CO2 | 1-15 | 20 mA (peak cur) | 96 mA | 0.208 |
| Jade Enthalpy Sensor | C7400S1000 | 0-7 or 8-15 (depending on setting by FFT) | 6.5 mA (avg cur) | 72 mA | 0.090 |
| TR120 | TR120<br>TR120-H | 1-15 | 523 mW | 950 mW | 0.55 |

# Networking Requirements

The CIPer Model 30 controllers are shipped from the factory with a default platform and station and all necessary items to run the station, along with a Tridium certificate. To start using the platform and the station inside it, you must change the default credentials for platform and station using commissioning process provided in the Connecting to Default Platform and Station section of the CIPer Software Tool Installation and Operations Guide. Once the default credentials are changed, you can use the device.

> **Note:**
>
> The CIPer Model 30 controllers comes within built DIP Switch feature, before configuring IP setting make sure all DIP switches are in off position.




## Default IP Address

The factory-shipped state of a controller has the following default settings for IP address, HTTP port, and platform credentials.

### IP Address

When shipped, a new CIPer Model 30 controller is pre-configured with an IPv4 address in the range: 192.168.1.160.

Default subnet mask: 255.255.255.0

You change these IPv4 network settings while starting up the commissioning of the CIPer Model 30 controller.

### HTTP Port for Platform Access

When shipped, the platform daemon of CIPer Model 30 is configured to listen on HTTPS port 5011. Often, this is left at default. However, if a different port is needed for a platform connection (perhaps for firewall reasons), you can change this during the commissioning of the CIPer Model 30.

### Platform Daemon Credentials

Any CIPer Model 30 controller is shipped with default platform daemon (administrator) credentials.

<u>Default platform credentials:</u>

- Username: honeywell
- Password: webs

<u>Default station credentials:</u>

- Username: admin

- Password: Admin12345
- Passphrase: Honeywell1

Initially, you need to use the default credentials (User ID and Password) to open (login to) a platform connection to the CIPer Model 30. During the startup commissioning, you must replace this platform administrator account with at least one different platform administrator user. Make sure to guard the credentials for such platform users closely.

| | Note: |
| --- | --- |
| The Niagara 4 Commissioning Wizard does not allow you to commission and startup a controller while retaining the factory platform user. | |

# Protocols

The communication protocols supported in the CIPer Model 30 are separated into two categories

- Network-Level
- Controller-Level

## Controller-Level

### Fox/Foxs (Fox-secure)

Tridium's proprietary TCP/IP protocol used for station-to-station and Workbench-to-station communication. The Fox Service in each station defines the port to use and manages the access.

### HTTP/HTTPS (HTTP-secure)

Standard protocol used by web browsers to access station web pages. The Web Service facilitates communication over HTTP.

### Niagarad/platformtls (secure niagarad)

Tridium's proprietary protocol used for Workbench-to-daemon communication. In the Supervisor station, Niagarad also communicates with the Provisioning Service. This service automates the performing of tasks on remote controllers.

### Field Bus protocols:

- BACnet/IP: To connect with other BACnet devices over IP network
- Panel Bus: To connect with panel bus I/O devices
- Sylk: To connect with Sylk devices

## Network- or Ethernet-Level

Cable Type and Length: Use an approved Category 5e or better Ethernet drop cable with RJ-45 plugs. Use professionally manufactured cables of no more than 328 feet (100 meters).

## Rapid Spanning Tree Protocol (RSTP)

It is a network protocol which facilitates with a loop-free topology for Ethernet networks. RSTP is faster than STP in terms of convergence when topology changes occur. The loop-free topology ensures that there is no broadcast storms and duplicate frame transmission.

## Spanning Tree Protocol (STP)

It is a network protocol which facilitates with a loop-free topology for Ethernet networks. STP is slower in comparison with RSTP.

## Dynamic Host Configuration Protocol (DHCP)

It is network management protocol, where the DHCP server dynamically allocates an IP address to the network devices or systems, so that all the device in the network communicate with each other.

## Domain Name System (DNS)

The DNS translates the domain name of a network system or device into numeric IP address, which is required for identifying a specific device in the network.

## TCP/IP Configuration

The TCP/IP configuration step enables you to review and adjust the TCP/IP settings for a platform.

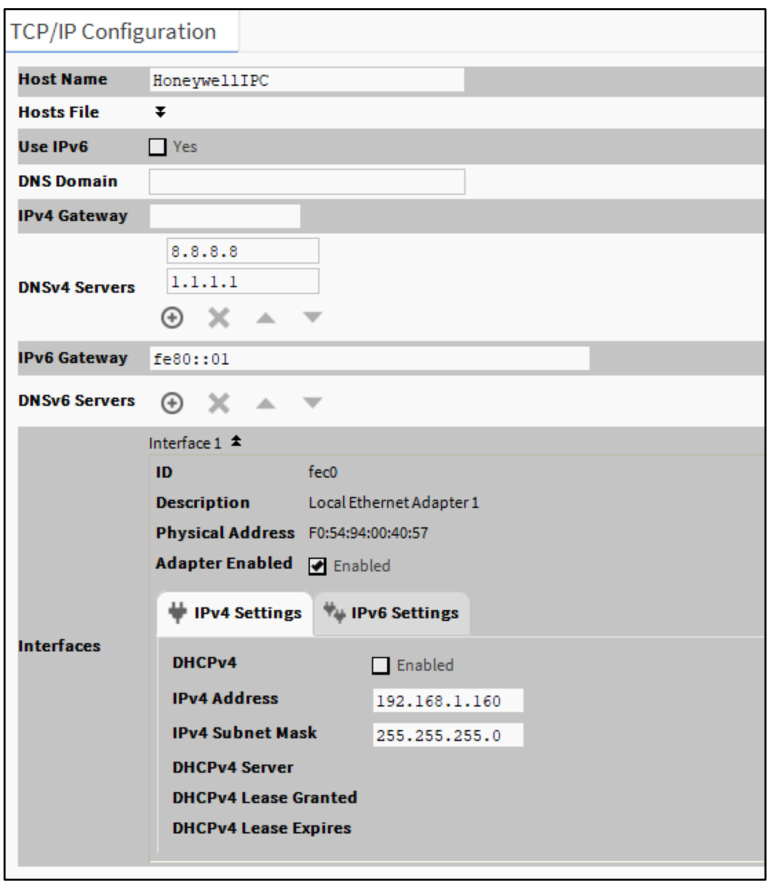



Figure 1: TCP/IP Configuration Wizard

| Note: |
|---|
| IPv6 support is available; however, this document focuses on IPv4 configuration. |

## Configuring TCP/IP Settings

While configuring the TCP/IP properties, do the following:

1. Review the Interface 1 settings on the IPv4 Settings tab, which includes the temporary factory-shipped IP address.
2. Enter a unique IPv4 address for the network. No other device on this network should use this IP address.
3. Enter the appropriate subnet mask used by the network.

   Alternatively, if the network supports Dynamic Host Configuration Protocol (DHCP), you can enable it by selecting the DHCPv4 option. In this case, the IPv4 Address and IPv4 Subnet Mask fields become read-only.

   **Note:**

   - Generally, static IP addressing is recommended over DHCP for stability. If DHCP is preferred, an IP address Reservation should be entered for the CIPer Model 30 in the DHCP server and the CIPer Model 30 IP address should not change.
   - Do not enable DHCP unless you are sure that the network has the DHCP servers. Otherwise, the CIPer Model 30 controller may become unreachable over the network.
   - In case you forget the IP address of the CIPer Model 30, you can connect to the device using serial mode of communication to know the IP address of the controller.

4. Review, and if needed adjust other TCP/IP settings, which (in the order of importance) include:

   - IPv4 Gateway – The IP address for the device that forwards packets to other networks or subnets.

     **Note:**

     The CIPer Model 30 controller supports only one gateway for all adapters. This includes the CIPer Model 30 Wi Fi Adapter in Client mode.

   - DNS Domain Name – The name of the network domain. If it is not applicable, leave blank.
   - DNSv4 Servers – The IPv4 address of one or more DNS servers.
   - Hostname – Default hostname may be localhost or enter another name you want to use for this host. If the hostname is entered, typically the name is unique for the domain.

     **Note:**

     In some installations, changing the hostname may result in unintended impacts on the network, depending on how the DHCP or DNS servers are configured. If in doubt, leave hostname at default.

   - Hosts File – Click control to expand and modify field. Format is a standard TCP/IP host file, where each line associates an IP address with a known host name. Each entry should be on an individual line. The IP address should be placed in the first column, followed by the corresponding host name. The IP address and hostname should be separated by at least one space.

a. To add a line, click at the end of the last line and press Enter key on the keyboard.

b. Enter the required data on the new line.

c. To return to see all TCP/IP settings, click the control to collapse the edit field when done.

5. Click Next to go to the next step.

Note:

- The CIPer Model 30 controllers have four Ethernet ports with 1Gbps speed support, where you can configure IPv4 or IPv6 for interface-1 using TCP/IP Configuration section under Platform. All four Ethernet ports also work as Ethernet switch.
- You can connect to CIPer Model 30 platform using any of the four Ethernet ports and remaining ports can be connected to IP based devices if needed. For example, IP Cameras, IP Thermostats). You can also perform IP daisy chain by looping the Ethernet cable from one CIPer Model 30 programming model to another CIPer Model 30 programming model, because Ethernet ports work as switch.
- You can enable DHCPv4 for interface 1 using TCP/IP Configuration section under Platform, if you want DHCP server to assign IPv4 automatically.

# Network Topologies

This section describes the network topologies that are used in CIPer Model 30 controller to communicate with other devices in the network.

## Supported Cables

The CIPer Model 30 controller supports four Ethernet ports. The Ethernet switch shall support:

- IEEE 802.3 standard with category (CAT) 3,4,5,6 Unshielded Twisted Pair (UTP) wiring.
- Segment length up to 80 percent of the maximum length allowed. IEEE 802.3ab supports maximum length up to 100 meters.
- Up to 2 sequential CIPer Model 30 controller connections through IP wiring
- Rapid Spanning Tree protocol (IEEE 802.1w) supports over 200 controllers on a daisy-chain bus with fewer home runs for faster and lower cost wiring.
  - Up to 40 controllers in a redundant ring configuration
- for enhanced fault tolerance.
- LAN star configuration.
- Continuous communications bandwidth of up to 50 percent of maximum bps capacity of Gigabit Ethernet.

## Non-Failover (Daisy Chain)

In non-failover, that is daisy chain connection type, if any of the device in the network fails, the devices next to the failed device also fail.

For example, there are 10 devices in a network and device number 1 is the master device, which connected to device 2, and device 2 is connected to 3, and so on. If device 5 fails to function, the device after 5, that is 6, 7, 8, 9, and 10 also fail to communicate with master device.

## Failover (Spanning Tree)

In the failover or spanning tree connection type, the devices connected in the ring, communicate with each other. If one of the devices in the network fails or stops working, the rest of the devices continue to work and failure of one device does not affect the working of other devices.

For example, there are 10 devices in a network and device number 1 is the master device, which connected to device 2, and device 2 is connected to 3, and so on. If device 5 fails to function, all the devices in the network except 5 continues functioning and communicating with master device.

# Getting Started

This section gives the information about the software tools that you need to download and install to start using CIPer Model 30 programming model.

## Digital Signature

IMPORTANT:

The Honeywell CIPer Model 30 software tool is signed. You can verify the signature using any OpenSSL tool. Following are the prerequisites and steps to verify the digital signature using OpenSSL community distribution.

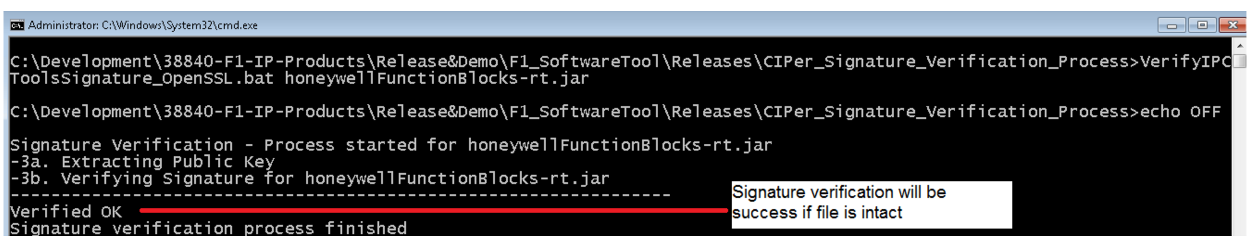
Prerequisites:

1. Download the Honeywell public key "Honeywell_IP_Controller.crt" from The Honeywell Buildings Forum.
2. Download the batch file "VerifyIPCToolsSignature_OpenSSL.bat" from The Honeywell Buildings Forum. This file has the commands to verify the module signature using the public key specified in Step 1.
3. Download OPENSSL from the link - https://www.openssl.org/source/openssl-1.0.2o.tar.gz.
4. Extract the file using any ZIP utility to get the folder-openssl-1.0.2o.
5. In the extracted folder find the file "openssl.cnf".
6. Set Windows environment variable OPENSSL_CONF=<Path to openssl.cnf>, for example OPENSSL_CONF=C:\openssl-1.0.2o\apps\openssl.cnf

To verify the signature:

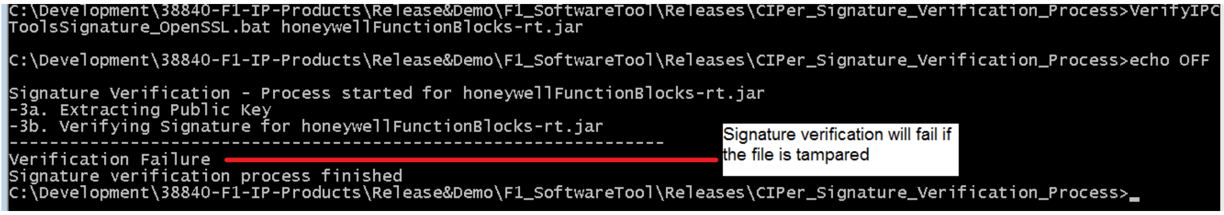
1. Place the files "Honeywell_IP_Controller.crt", "VerifyIPCToolsSignature_OpenSSL.bat", Honeywell CIPer Model 30 software tool distribution/modules and signature file together at the same location. For example, following files are in one place.
   - Honeywell_IP_Controller.crt
   - VerifyIPCToolsSignature_OpenSSL.bat
   - honeywellFunctionBlocks-rt.jar
   - honeywellFunctionBlocks-rt.jar.sig
2. Open the command prompt and navigate to the location where you saved the above files.
3. Verify all the modules released to confirm their authenticity by executing the batch file.

   For example, verify "VerifyIPCToolsSignature_OpenSSL.bat" against a module, C:\Development\38840-F1-IP-Products\Release&Demo\F1_SoftwareTool\Releases\CIPer_Signature_Verification_Process>VerifyIPCToolsSignature_OpenSSL.bat honeywellFunctionBlocks-rt.jar.
4. OpenSSL verifies the module's signature and printout the below verification details:



Caution! You must trust the module authenticity only when you get the confirmation "Verified OK".

If the Niagara module is compromised, you get the following log, where verification has failed:




# Connecting to Platform and Station with Default Credentials

To know more see CIPer Model 30 Installation and Operations Guide.

## Connecting to Platform

To open and connect to a platform:

1. Navigate to the Nav tree and right-click My Host<host_id> and click Open Platform. The Connect window is displayed.

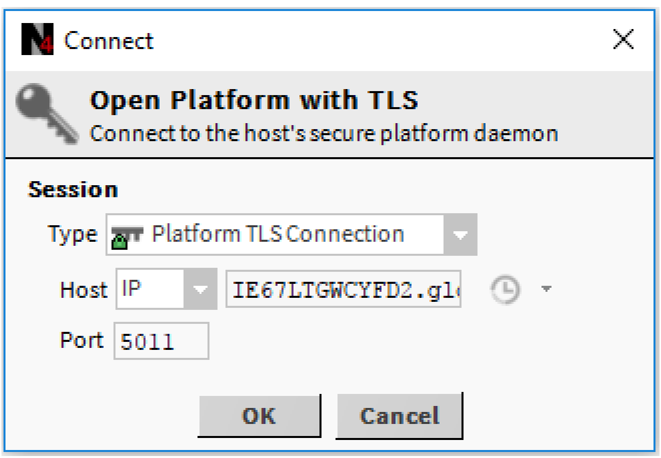



Figure 2: Connect Window to Open Platform

2. Select the session type either Platform TLS Connection (secured) or PlatformConnection (unsecured / standard) in the Type drop-down menu.
3. Select the Host as IP in the Host drop-down menu.
4. Enter the host Id in the input field next to Host. By default, the application takes the host Id of your system. If you select the secured platform type, the default port number is 5011 and if you select the unsecured planform type then the port is 3011.

| Note: |
|---|
| • The (History) icon next to the host Id displays the list of host Ids used before. You can also select the host Id from the History drop-down menu.<br>• Honeywell recommends use of TLS type connection for secure connection. |

5. Click Ok. The Niagara Identity Verification dialog box is displayed for the TLS connection.

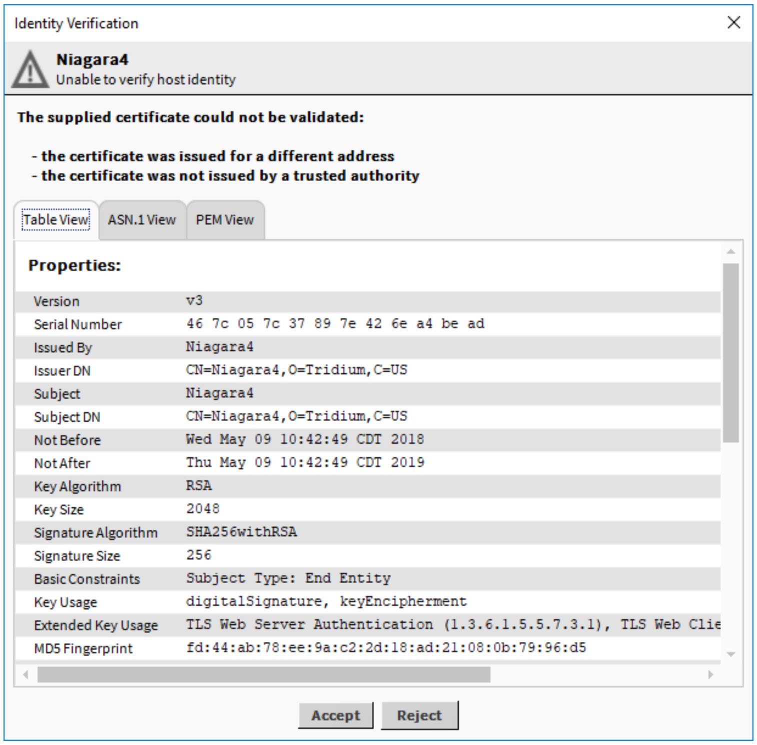



Figure 3: Identity Verification Dialog Box

6. Click Accept and the Authentication window is displayed.

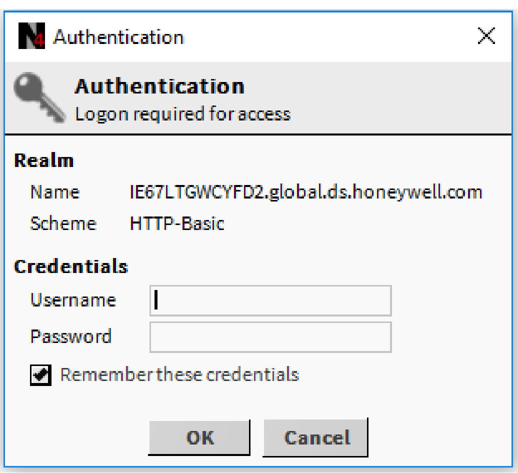



Figure 4: Authentication Window

7. Enter the credentials and click Ok. You must login with your platform credentials.

## Connecting to Factory Installed Station

The default station will be empty with IPC Network added to it. You can connect to the station and start using the device. The default credentials for station are:

Username: honeywell

Password: webs

# Creating and Connecting to New Station

To create and connect to a new station:

1. Perform the steps to open a platform.
2. Navigate to Tools drop-down menu and click New Station. The New Station Wizard is displayed.

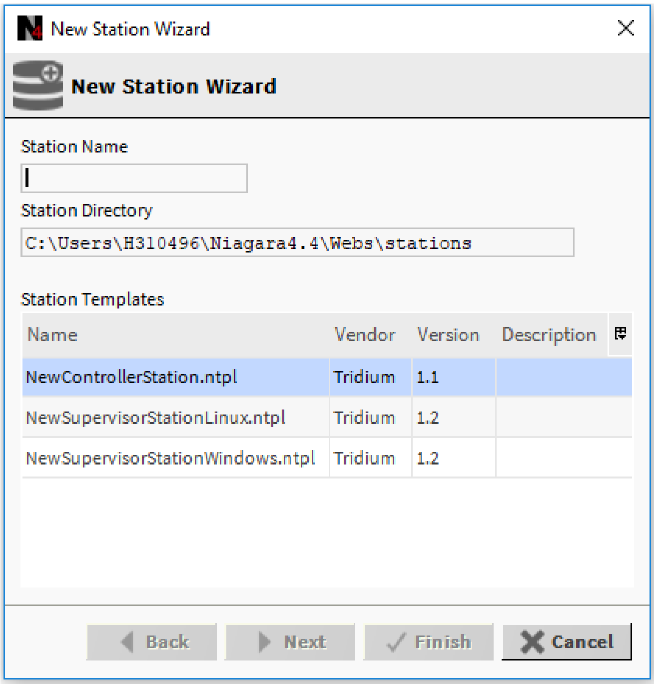



Figure 5: New Station Wizard

3. Enter the name of the station. The Station Directory field, which is non-editable field, displays the location of the station.
4. Select the NewControllerStation.ntpl template from the Station Templates and click Next. The next screen of the wizard is displayed.

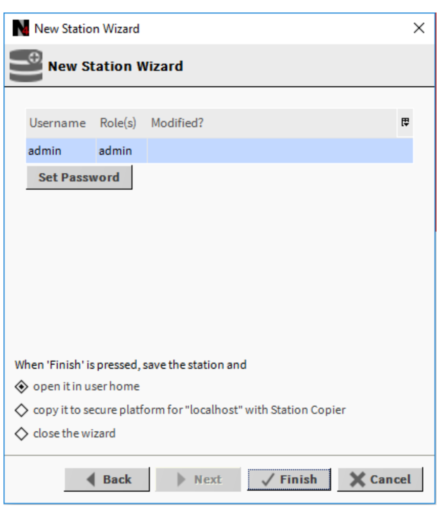



Figure 6: New Station Wizard

5. Click Set Password and Set Password window is displayed.

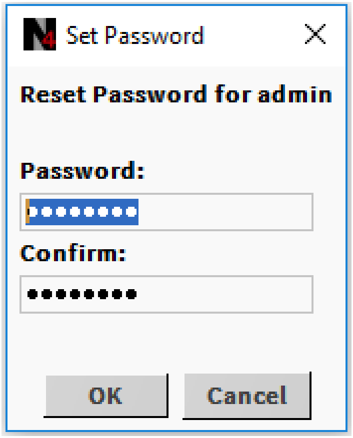



Figure 7: Set Password Window

6. Enter the password and confirm it by re-entering the same password in the Confirm field.
7. Click Ok.
8. Select the **copy it to secure platform for “localhost” with** Station Copier action to perform after completing the process for opening a station.

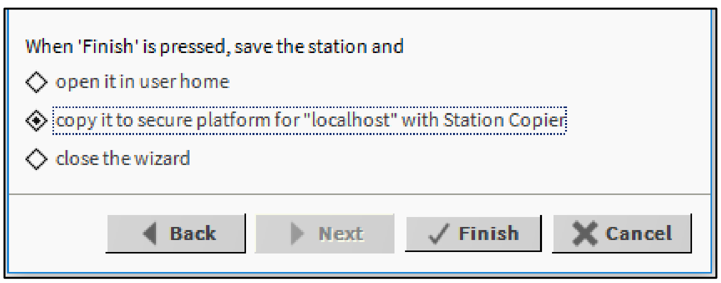



Figure 8: New Station Wizard

9. Click Finish to complete the process of opening a station. The application shows a status notification at the lower-right side of the screen.

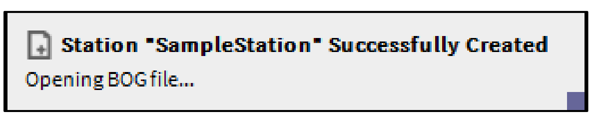



Figure 9: Status Notification After Opening Station

The Station Copier screen is displayed, and then the Station Transfer Wizard is displayed as shown in the figure Station Transfer Wizard.

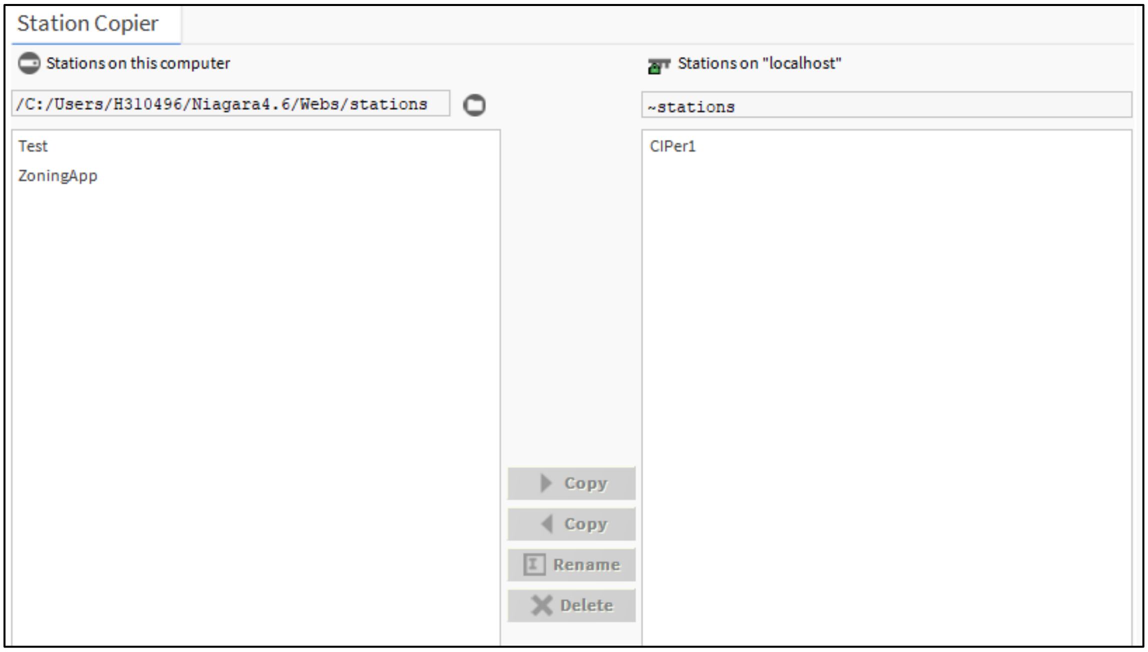



Figure 10: Station Copier Screen

10. Select the options—START AFTER INSTALL and AUTO-START as required and click Next.

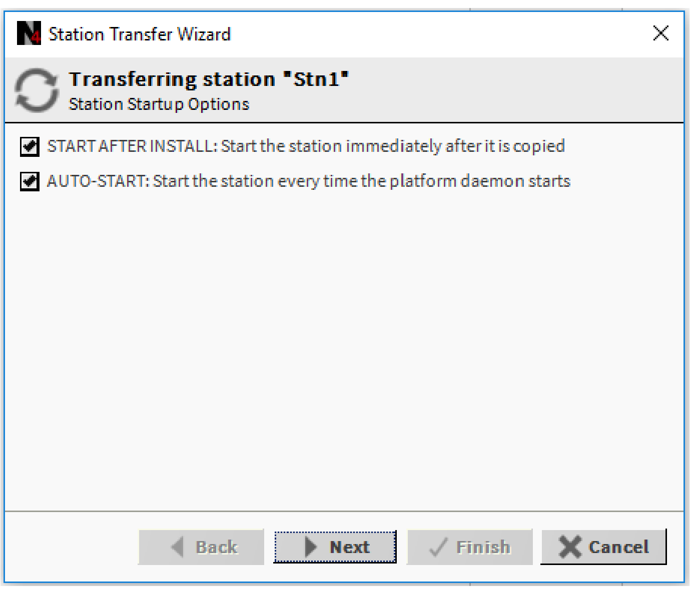



Figure 11: Station Transfer Wizard

| Note: |
| --- |
| The AUTO-START option is disabled by default due to security reasons. You must enable it if required. |

11. Click Finish to complete the process of transferring the station from local device to localhost. The Open Application Director dialog box is displayed.

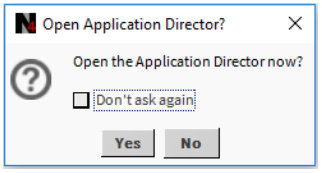



Figure 12: Open Application Director Dialog Box

12. Click Yes and the application director is displayed with the station that you created in the list of stations along with station details like name, type, status, and so on.

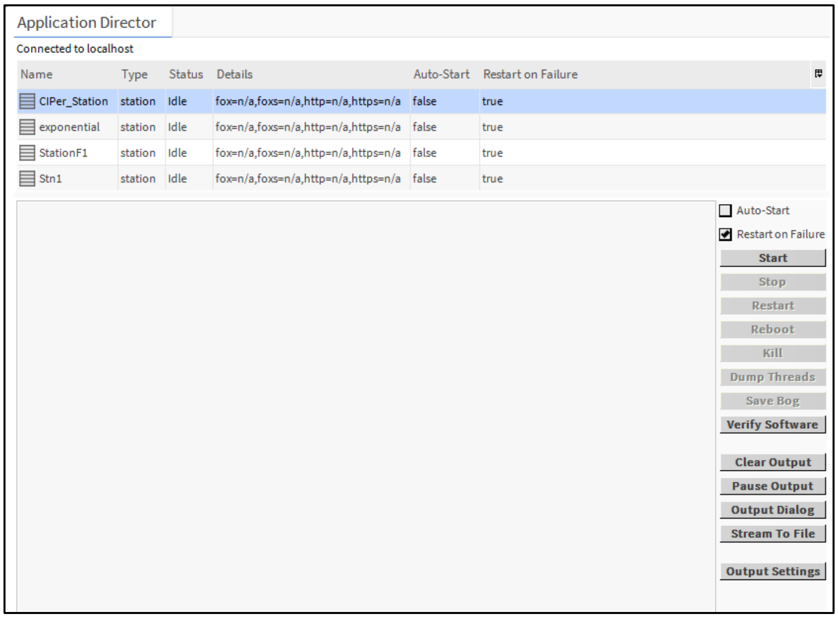



Figure 13: Application Director

13. Select the station to start.
14. Clear or select the Auto-Start and Restart on Failure check boxes as required.
15. Click Start to start the station.

## Signing into Station

To sign in to and set up a station:

1. Navigate to the Nav tree and right-click <IP address of CIPer> and click Open Station. The Connect window is displayed.

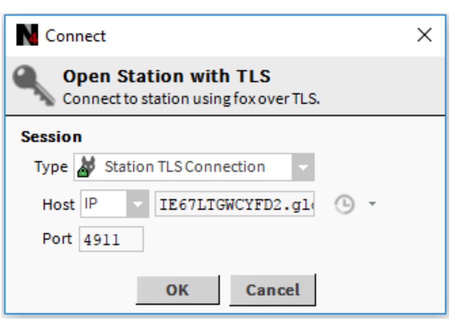



Figure 14: Connect Window

2. Select the station type as Platform TLS Connection (secured) or Platform Connection (unsecured / standard) in the Type drop-down menu.

3. Select the host as IP in the Host drop-down menu.

4. Enter the host Id in the input field next to Host. By default, the application takes the host Id of your system. If you select the secured platform type the default port number is 5011 and if you select the unsecured planform type, the port is 3011.

> **Note:**
>
> The  (History) icon next to the host Id displays the list of host Ids used before. You can also select the host Id from the History drop-down menu.

5. Click Ok. The Authentication window is displayed.

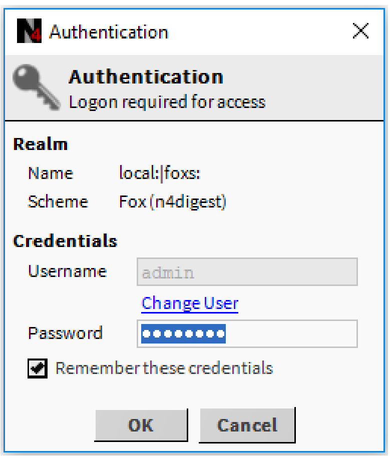



Figure 15: Authentication Window

6. Enter the credentials and click Ok. You must login with station credentials.

7. Expand the Station and navigate to Config > Drivers.

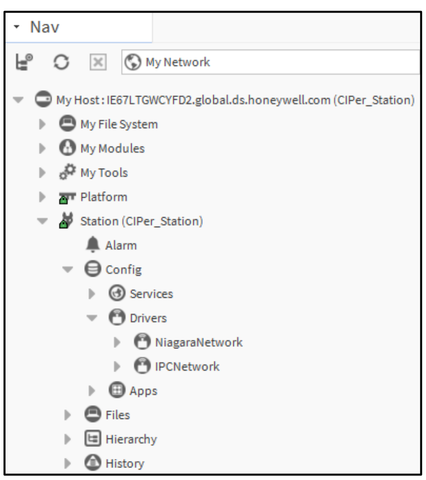



Figure 16: Nav Tree View

You can do the programming as required in the Sequenced Control Program and Event Control Program folders under IPCNetwork folder.

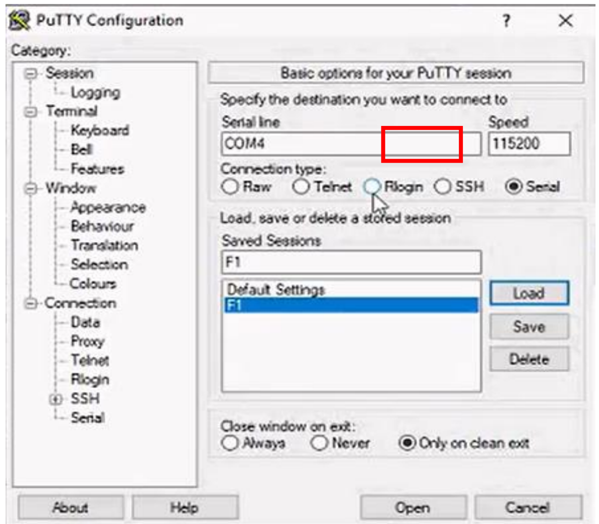
## Resetting CIPer 30 Controller to Factory Defaults Settings

In case you forget the station credentials, you can reset to the factory defaults. Follow the below steps to reset CIPer 30 controller to default factory setting.

NOTE: Since this operation performed in bootloader, the controller must be running in bootloader.

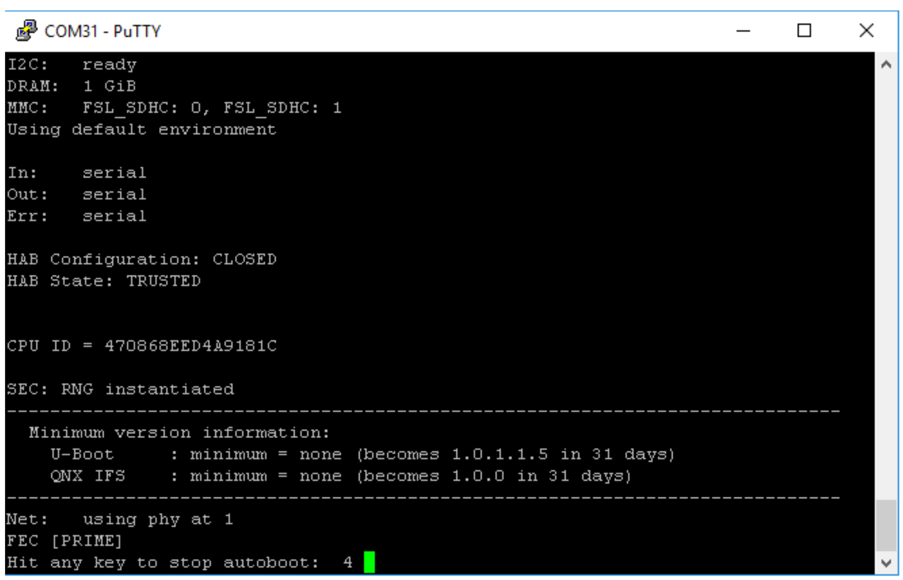
1. Connect the CIPer 30 controller with serial console using terminal emulators tool (for example Putty).

   NOTE: The supported baudrate on CIPer 30 devices is 115200 bps.

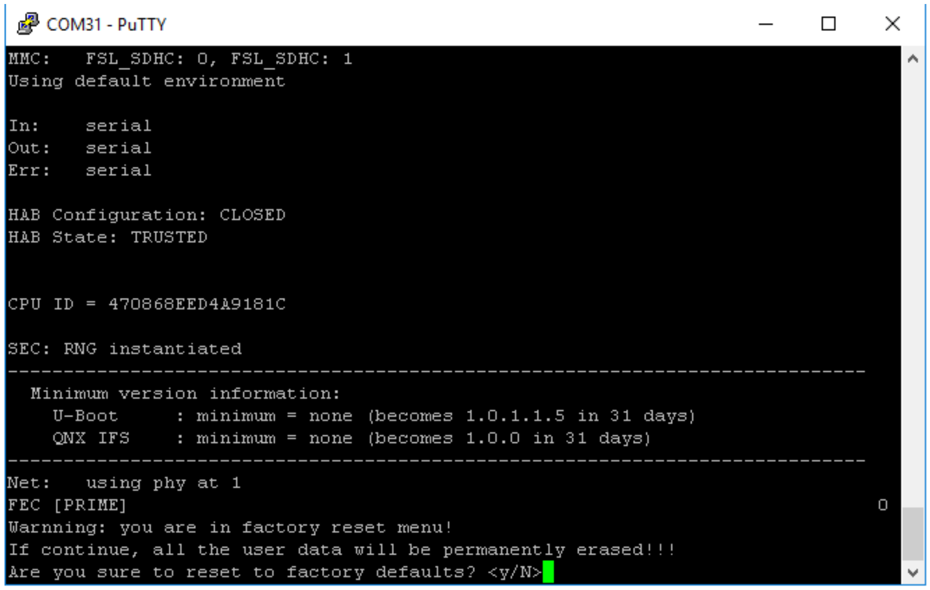



2. After configuring CIPer 30 controller with serial console, power cycle the device.
3. When system displays Hit any key to stop autoboot, type passphrase enter and press Enter key on the keyboard.

   NOTE: Complete the step (3) within 5 seconds. If fail to enter, system will continue autoboot.




This action redirects you to factory reset menu and conformation message is displayed "Are you sure to reset to factory defaults? <Y/N>"




4. Type Y and press Enter key on the keyboard, this action reset controller to default factory settings.

   Wait until the LED shows the device is working properly. Do not power off the controller during the reset process or the controller will be damaged permanently.

   This completes CIPer Model 30 Controller factory reset.

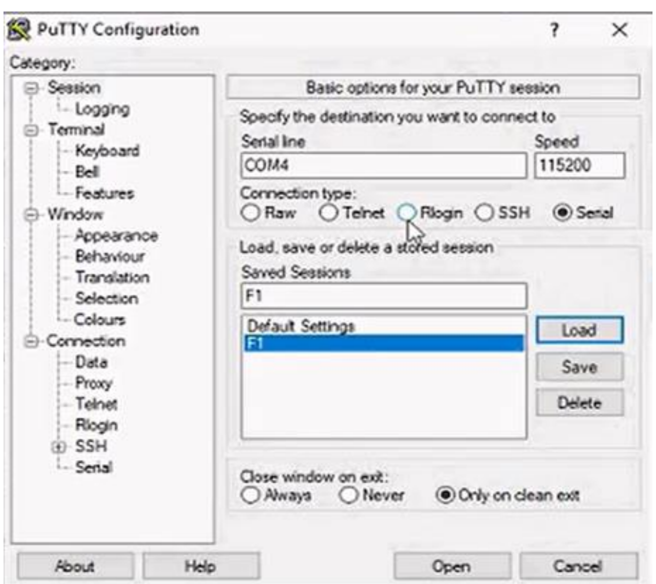
## Serial Shell option available in CIPer Model 30 controller

Follow the below steps to login to the IPC System Shell of the CIPer Model 30 controller using Serial Shell.

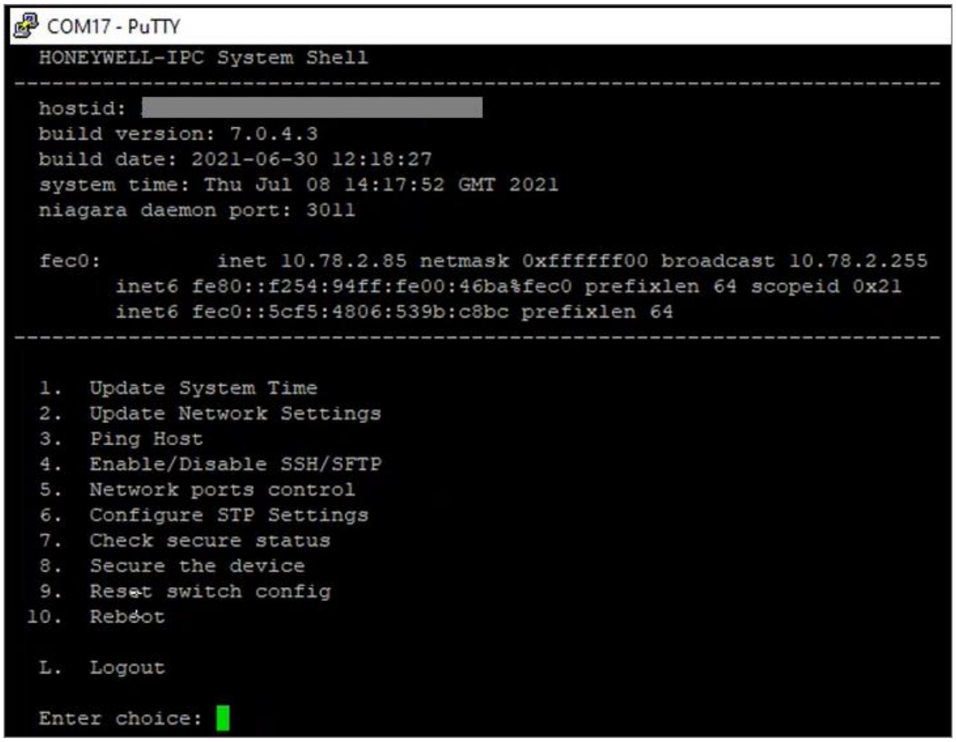
**NOTE**: Since this operation is performed in the bootloader, the controller must be running in the bootloader.

1. Connect the CIPer Model 30 controller with the serial console using the terminal emulators tool (for example Putty).

   **NOTE:** The default baud rate of the CIPer **Model 30** controller is 115200bps.




   After configuring CIPer 30 controller with a serial console, power cycle the device. This action displays IPC System Shell.



```
COM17 - PuTTY
  HONEYWELL-IPC System Shell
---------------------------------------------------------------------------
  hostid: [redacted]
  build version: 7.0.4.3
  build date: 2021-06-30 12:18:27
  system time: Thu Jul 08 14:17:52 GMT 2021
  niagara daemon port: 3011

  fec0:         inet 10.78.2.85 netmask 0xffffff00 broadcast 10.78.2.255
        inet6 fe80::f254:94ff:fe00:46ba%fec0 prefixlen 64 scopeid 0x21
        inet6 fec0::5cf5:4806:539b:c8bc prefixlen 64
---------------------------------------------------------------------------

  1.  Update System Time
  2.  Update Network Settings
  3.  Ping Host
  4.  Enable/Disable SSH/SFTP
  5.  Network ports control
  6.  Configure STP Settings
  7.  Check secure status
  8.  Secure the device
  9.  Reset switch config
 10.  Reboot

  L.  Logout

  Enter choice:
```

Following are the options available in the Serial Shell.

1 = Update System Time

2 = Update Network Settings

3 = Ping Host

4 = Enable/Disable SSH/SFTP

5 = Network Ports control

6 = Configure STP Settings

7 = Check secure status

8 = Secure the device

9 = Reset switch config

10 = Reboot

## Opening and Using Palette

To open and use a palette:

1. Navigate to Window > Side Bars > Palette to open the palette pane. The palette pane is displayed at the lower left side of the screen.

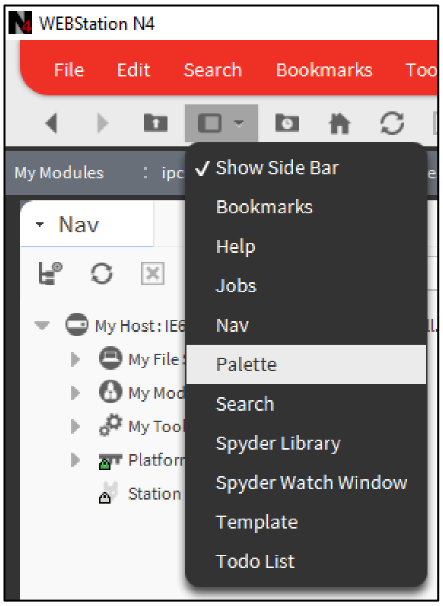



Figure 17: Palette Pane

2. Click (Open Palette). The Open Palette window is displayed.

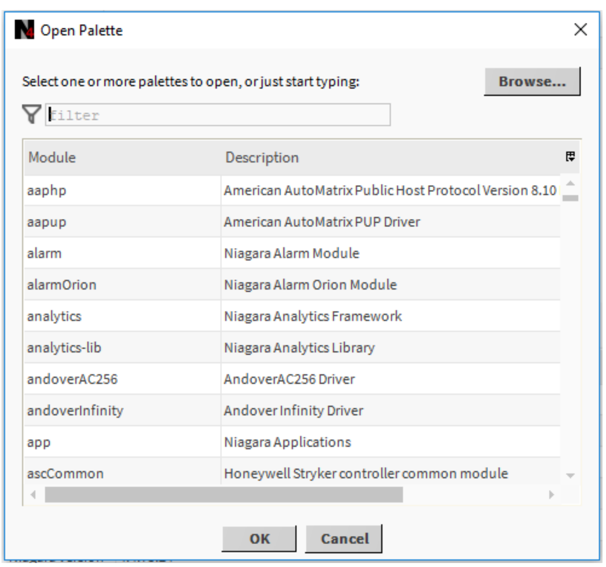



Figure 18: Open Palette Window

3. Select the module ipcProgrammingTool from the list or type the module name in the input field to open the palette, and then click Ok. To select multiple modules, hold the Ctrl key on the keyboard and select the required modules. This adds the selected module to the palette drop-down menu and the palette pane displays the selected palette.

   You can also use the Browse button on the Open Palette window to select the path of the module file from the device if you know the module location.

See To open module file using Browse button.

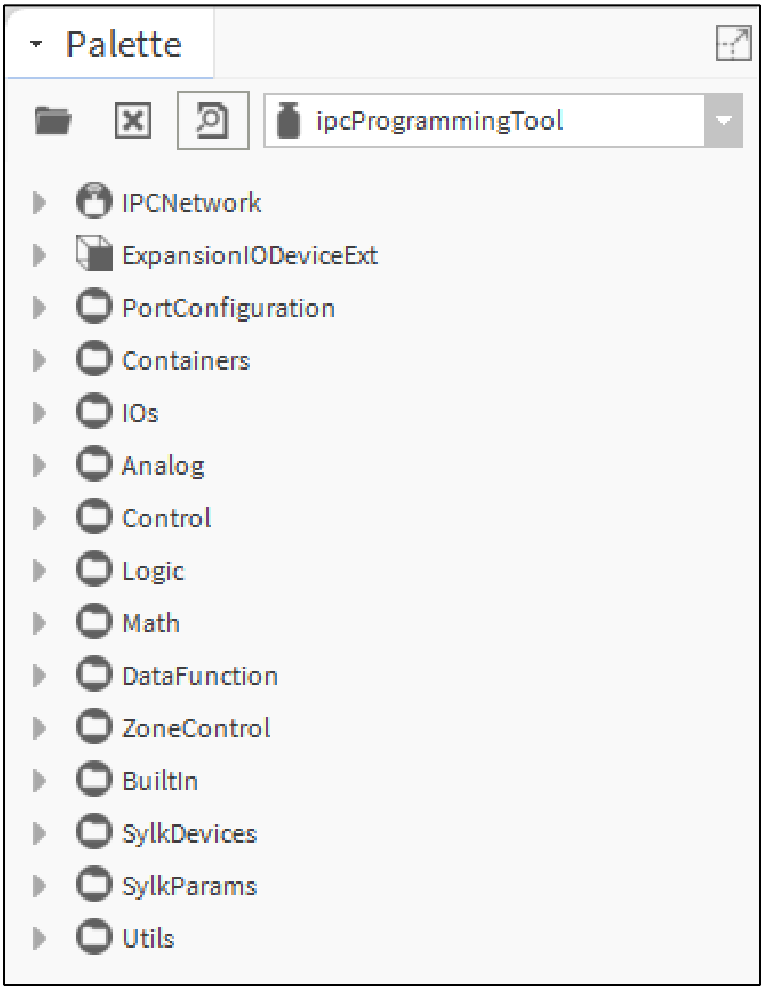



Figure 19: ipcProgrammingTool Palette Objects

**Note:**

To open another palette, click the palette drop-down menu and select the required palette if present, or open it by clicking the Open Palette icon.

To close the opened palette, click (Close Palette).

To view the preview of an item inside the object in the palette, select the item, and then click (Preview). The preview of the selected item is displayed at the lower side of the palette pane.

**Note:**

- To open another palette, click the palette drop-down menu and select the required palette if present, or open it by clicking the Open Palette icon.
- To close the opened palette, click (Close Palette).

- To view the preview of an item inside the object in the palette, select the item, and then click (Preview). The preview of the selected item is displayed at the lower side of the palette pane.

4. Navigate to the Drivers folder in the Nav tree.
5. Drag and drop the IPCNetwork object from the ipcProgrammingTool palette into the Drivers folder. The IPCNetwork folder is displayed under the Drivers folder along with NiagaraNetwork folder.
6. Expand the IPCNetwork folder and navigate to LocalDevice > Points. The Points node contains two folders
   - SequencedControlProgram - The SequencedControlProgram folder is used for Honeywell function blocks (honFunctionBlocks)
   - EventControlProgram - The EventControlProgram is used for kitControl components.

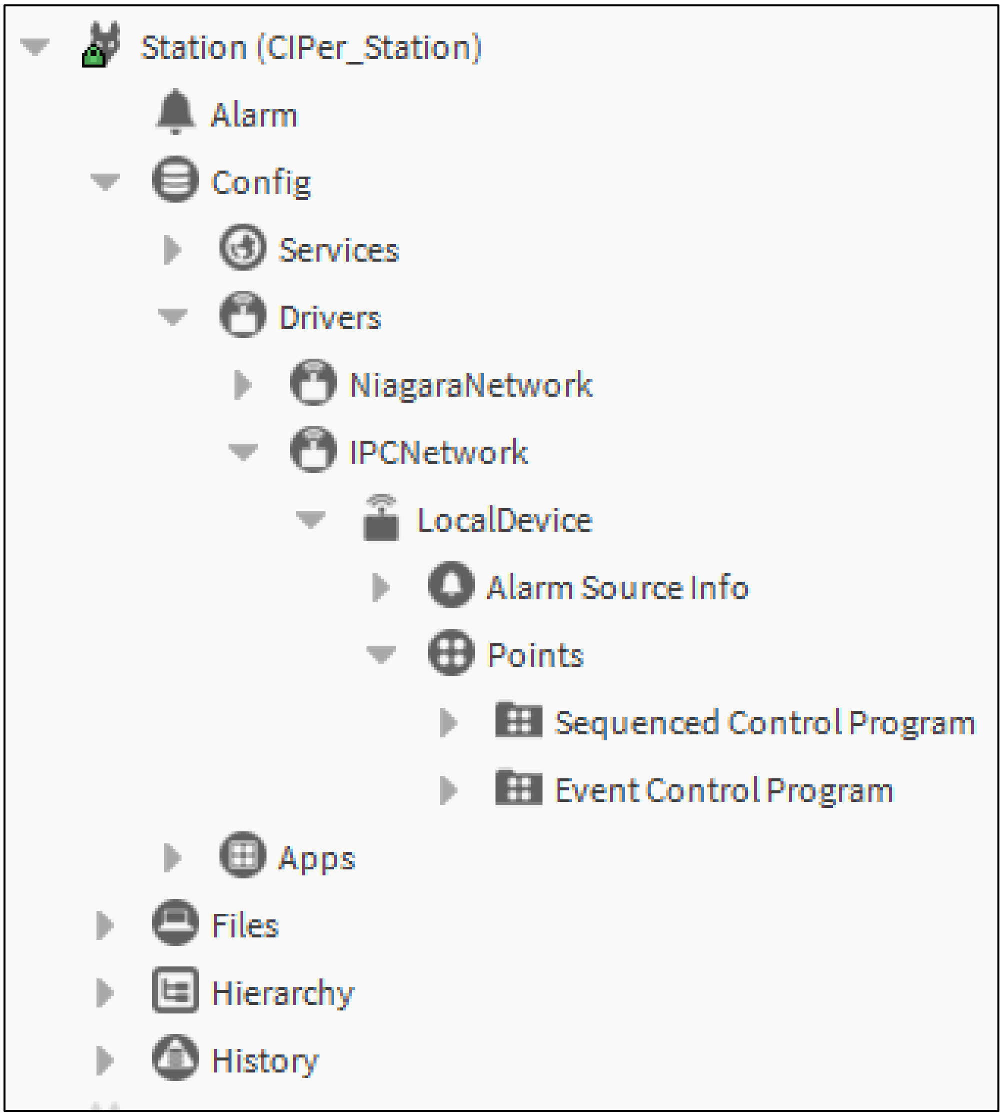



Figure 20: Nav Tree View

# Replacing Pre-Configured Station with User-Supplied Station

You can transfer the station by using station copier.

To replace the pre-configured station by user-supplied station:

1. Connect to the Platform and double-click Platform. The Platform screen is displayed.

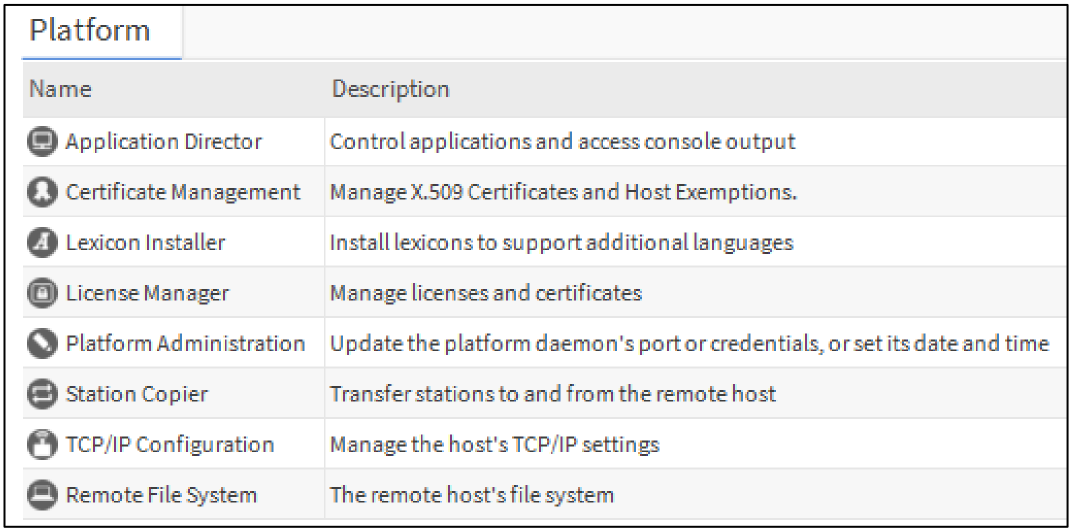



Figure 21: Platform Screen

2. Double-click Station Copier. The Station Copier screen is displayed along with stations on your computer and default station running in the CIPer Model 30 controller.


Figure 22: Station Copier Screen

3. Select the required station to copy to the localhost from the stations available on your computer.
4. Click Copy. The Station Transfer Wizard starts copying the station to the CIPer Model 30 controller.

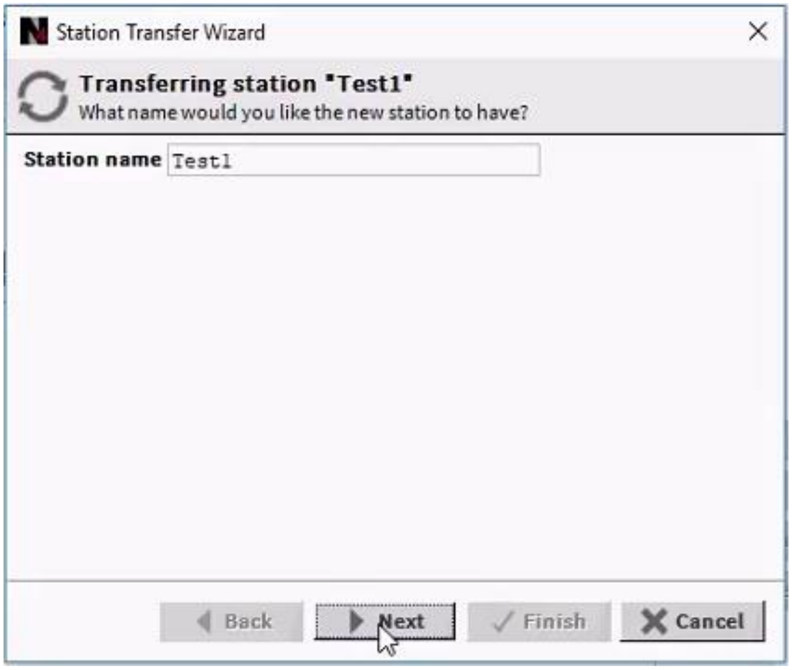
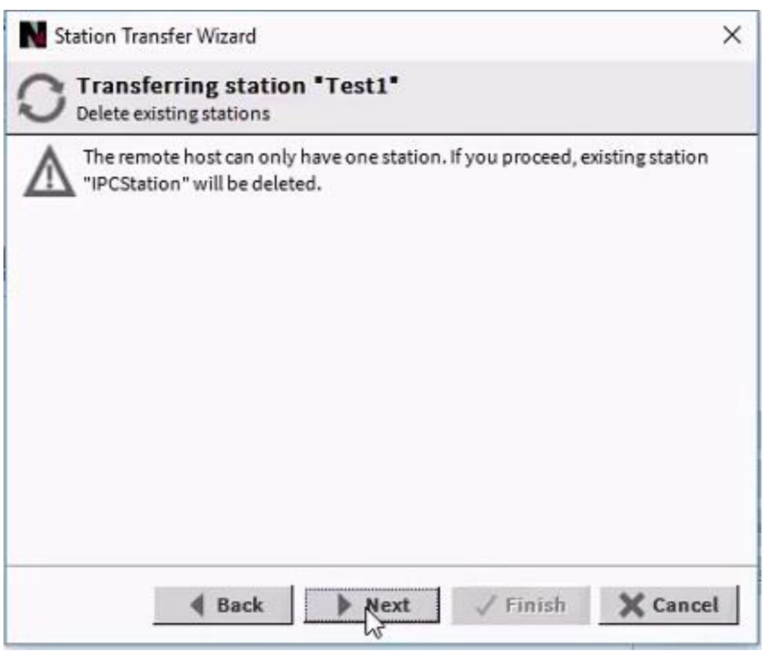


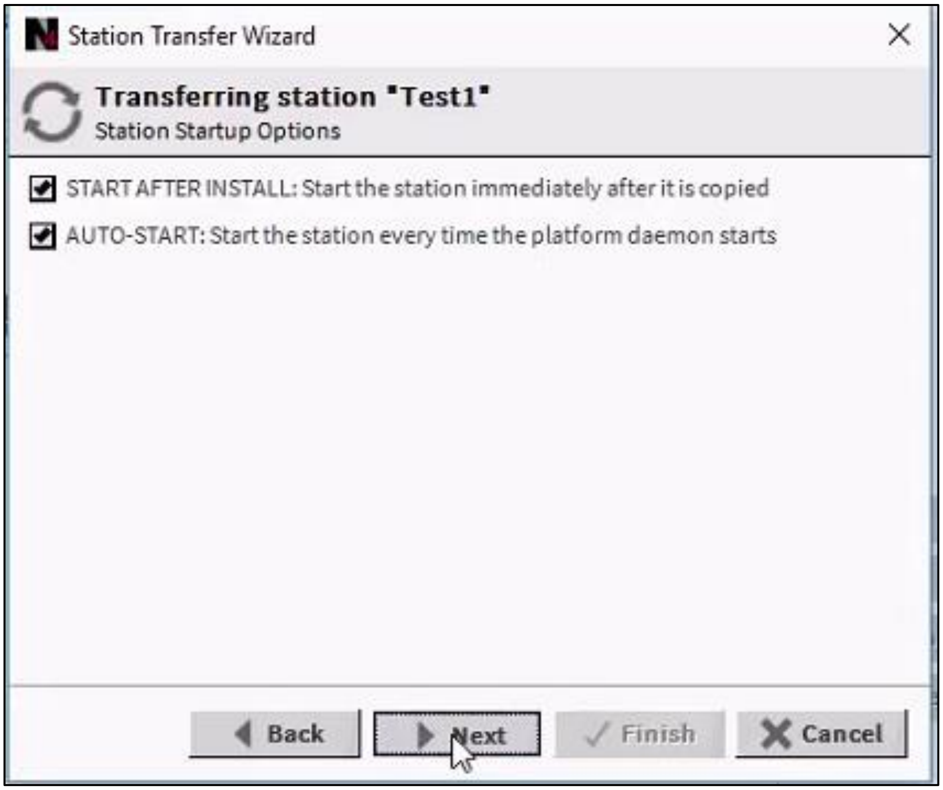

Figure 23: Station Transfer Wizard

5. Enter a name for the new station and click Next. The Station Transfer Wizard inform about the deletion of the existing the station in the remote host.

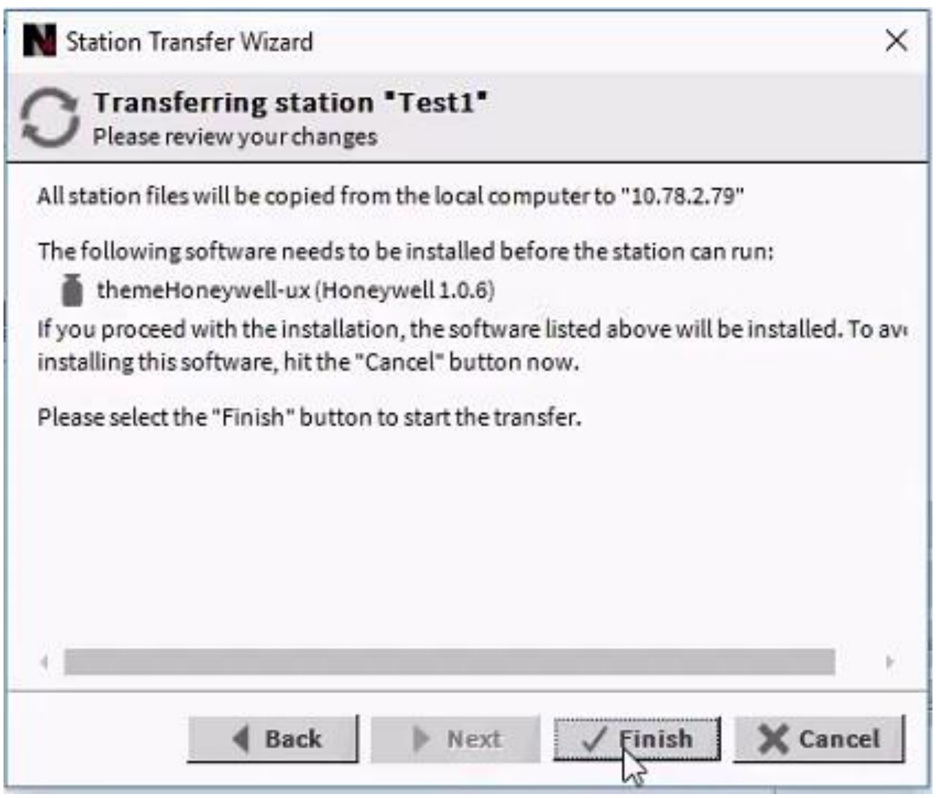



6. Click Next, if you want to delete the existing station and replace it with new station.

   You can also delete the existing station in the remote host (CIPer Model 30) before you start copying the new station by selecting the station in the remote host and clicking Delete.




7. Click Next.




8. Click Finish. The Transferring Station Wizard is displayed where all the applications are stopped, existing station is deleted, and new user-specified station is copied.

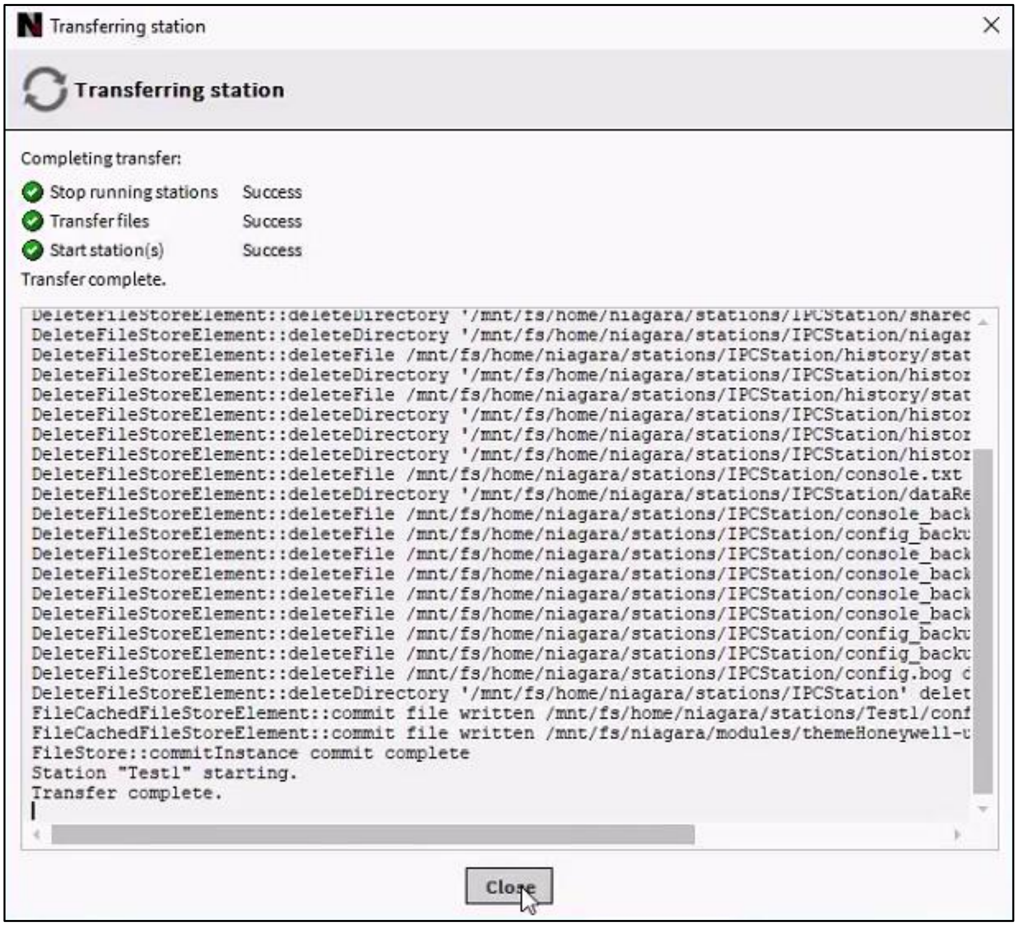



Figure 24: Successful Transferring Station

The Open Application Director window is displayed.

9. Click Yes.

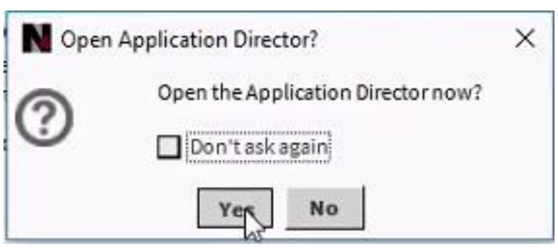



Figure 25: Open Application Director Window

The user-created station is displayed on the Application Director screen and you can start using the newly copied station.

# Connection Between Two CIPer Controllers Through Niagara Network

You can connect two different CIPer controllers in the workbench through Niagara Network.

1. Open the Niagara workbench.
2. Connect to the required platform and station.
3. Navigate to NiagaraNetwork folder under Station > Config > Drivers, double-click NiagaraNetwork.

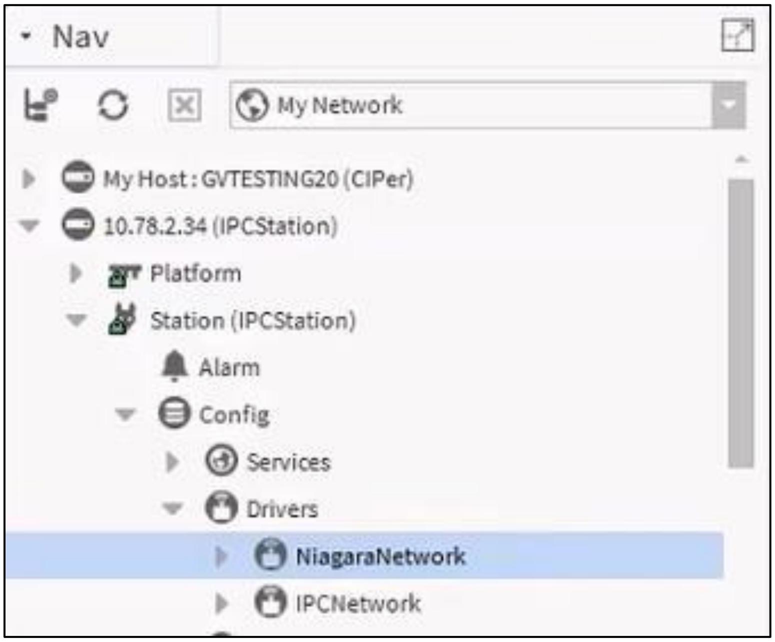



Figure 26: NiagaraNetwork Folder

4. Click Discover to discover the station in the platform of another controller.

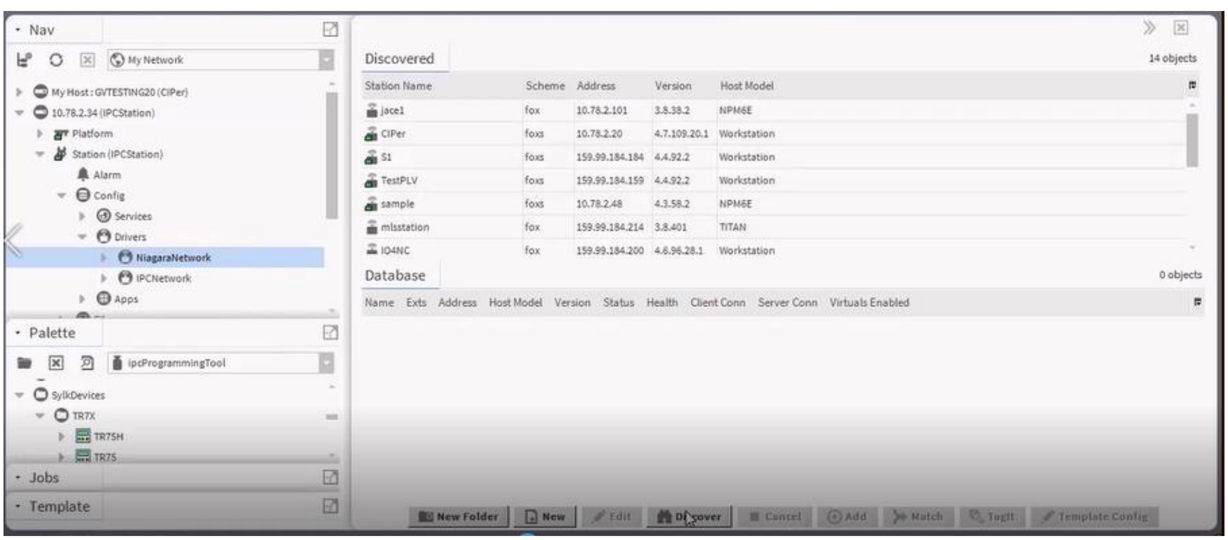



Figure 27: List of Discovered Stations

5. Locate the required station to connect and drag and drop it from the Discovered pane to the Database pane. The Add window is displayed.

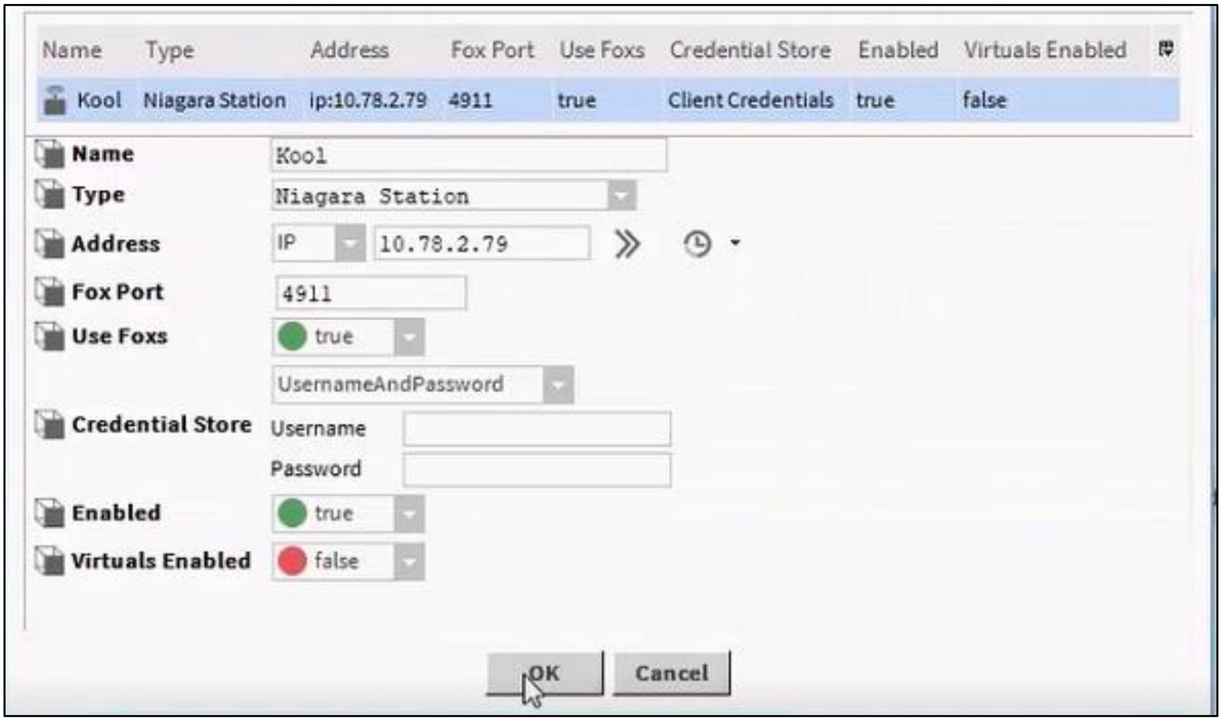



Figure 28: Add Window

6. Enter the username and password of the station in the respective fileds.
7. Click OK. The station is now added to the database.

   You can check the connection between two stations of different controllers by right-clicking the station and then selecting Actions > Ping.

# Connecting Multiple CIPer 30 Controllers using RSTP Configuration

You can connect to multiple CIPer 30 controllers using RSTP configuration. RSTP loop (i.e. RING topology) with 40 number of CIPer 30 IP controllers with maximum load in each controller. To configure and connect all CIPer 30 controllers in Workbench. For RSTP Configuration Setting refer Rapid Spanning Tree Protocol (RSTP)

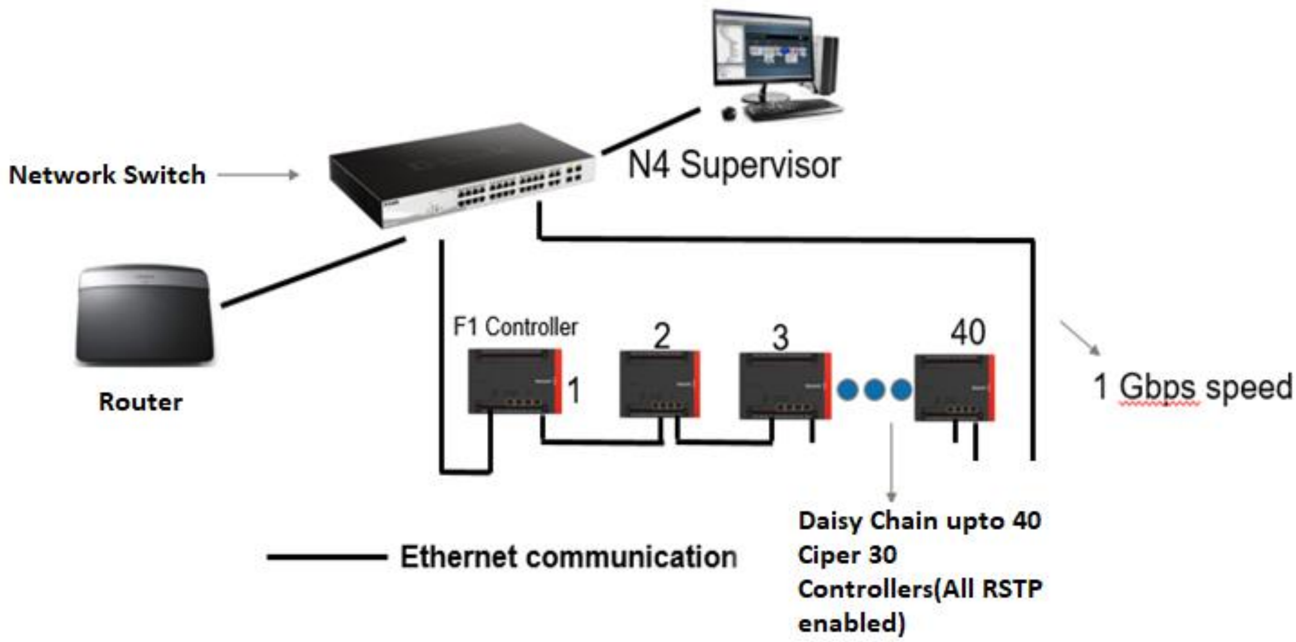



Figure 29: RSTP Configuration – 40 CIPer 30 Controllers and router connected to a single Network Switch

# Connecting Multiple CIPer 30 Controllers using Daisy chain loop

To connect the CIPer 30 Controllers in a daisy chain loop you need to add Port Configuration Property under Local Device from the IPC Programming Tool palette. For port configuration details refer to Port Configuration

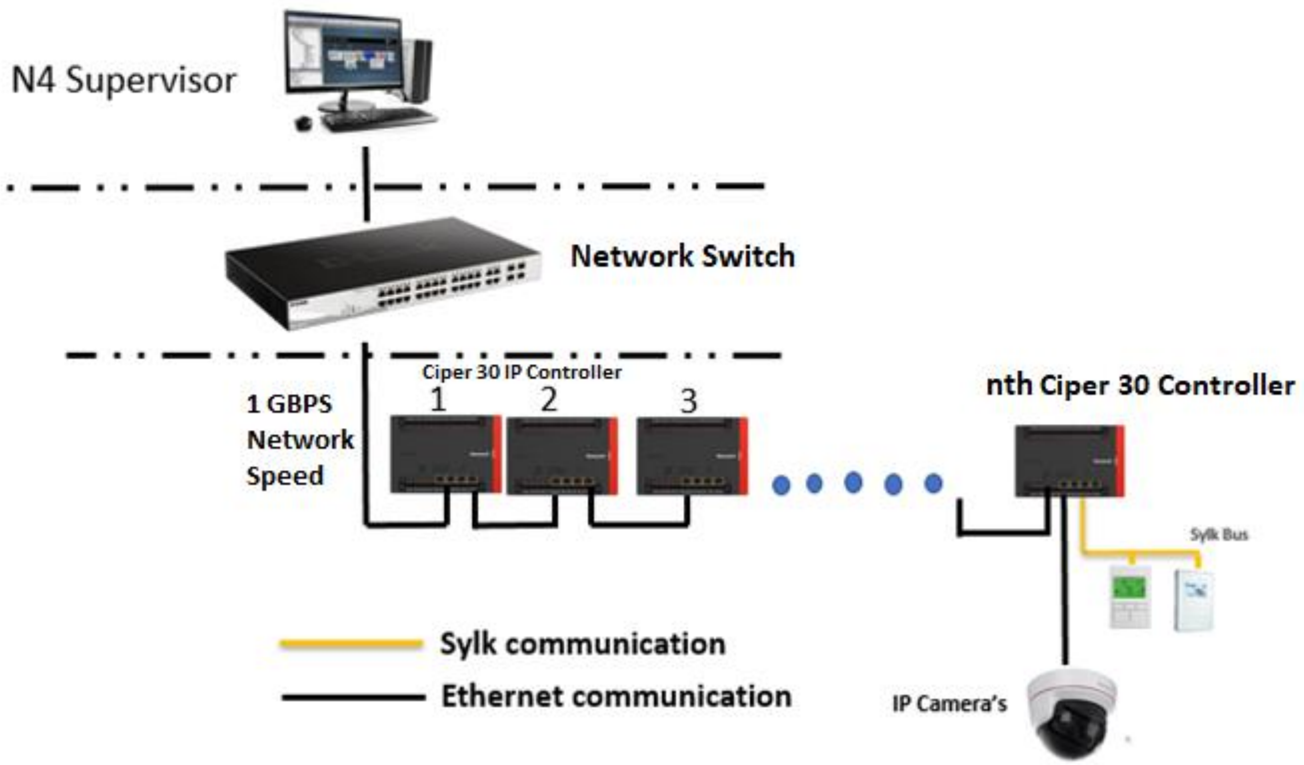



Figure 30: IP Daisy Chain Configuration – N number of CIPer 30 Controllers connected to single Network Switch (Full load)

> **Note:**
>
> You can add any IP compatible devices (ex.- Edge Devices) in the spare Ethernet port.
>
> Rapid Spanning Tree protocol (IEEE 802.1w) supports over 200 controllers on a daisy-chain bus with fewer home runs for faster and lower cost wiring.
>
> - Up to 40 controllers in a redundant ring configuration

# Managing Software And Licenses

This section gives information about how you can manage the CIPer Model 30 application licenses and certificates.

The access rights in CIPer Model 30 are set to default for the administrator users and can be modified by the user who is responsible for configuring the CIPer Model 30.

## Managing License

To see the license details or validate your license, navigate to Platform in the Nav tree and double-click License Manager. The License Manager screen is displayed. The details include the host address and ID of the system, brand ID, and list of license and certificate files. The host ID is unique for each controller.

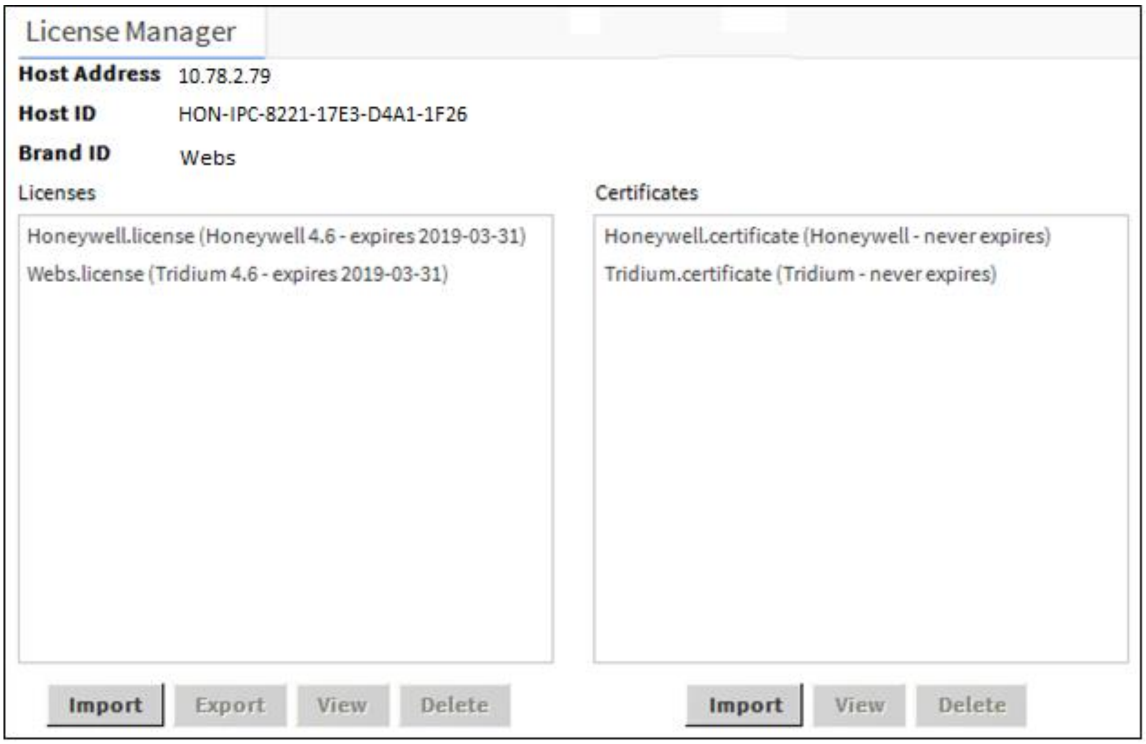



Figure 31: License Manager Screen

### License Manager

The capabilities that License Manager provides are as follow:

- Importing License
- Exporting License
- Viewing License
- Deleting License
- Importing Certificate
- Viewing Certificate
- Deleting Certificate

## Importing License

You can import license by importing one or more licenses from files, local license database, and licensing server.

To import a license:

1. Navigate to Platform in the Nav tree and double-click License Manager. The License Manager screen is displayed.
2. Under Licenses section, click Import. The Import License window is displayed.

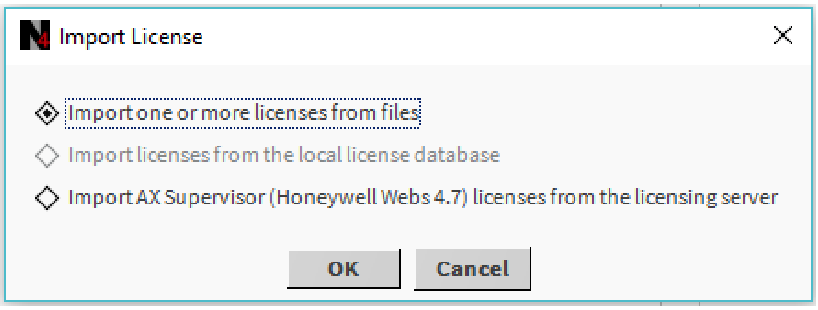



Figure 32: Import License Window

- Import one or more license from files: To import license from the local file
- Import license from the local license database: To import license from the local database. This option is enabled when there is a local license database.
- Import AX Supervisor (Honeywell Webs 4.7) licenses from the licensing server: To import license from the licensing server in case there are no license present in the local files and databases. When you select this option for importing license, the application displays following window to restart the station.

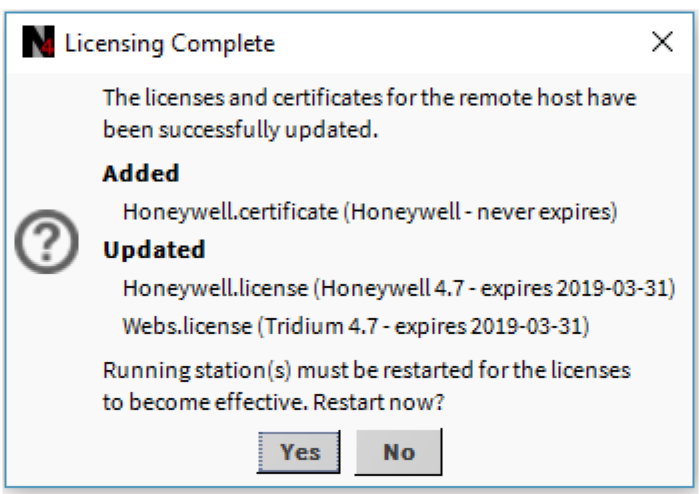



Figure 33: License Complete Dialog Box

3. Select the required option to import the license.
4. Click Ok. The Select File window is displayed.

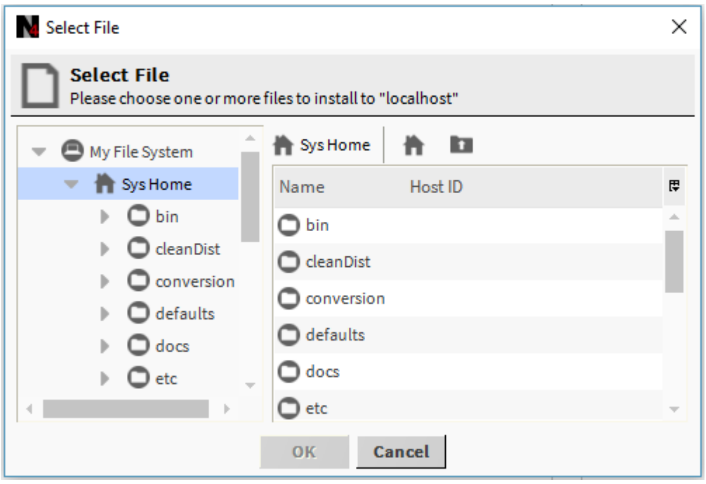



Figure 34: Select File Window

5. Navigate to the required path where the license file is present.
6. Select the required license file and click Ok. The import status is shown in the window displayed.

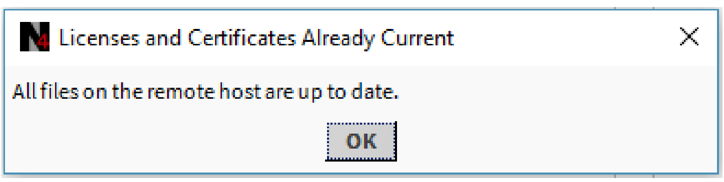



Figure 35: Licenses and Certificates Already Current Dialog Box

7. Click Ok.

## Exporting License

To export a license:

1. Navigate to Platform in the Nav tree and double-click License Manager. The License Manager screen is displayed.
2. Under Licenses section, click Export. The Save License As window is displayed.

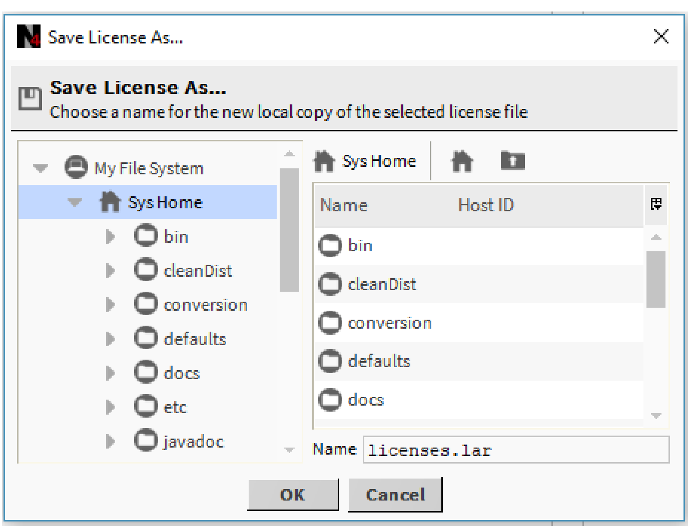



Figure 36: Save License As Window

3. Navigate to the required path where you want to save the license file and click Ok.

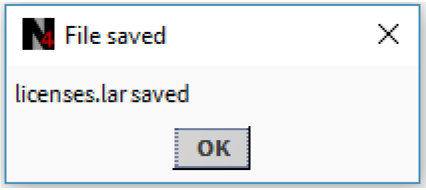



Figure 37: File saved Dialog Box

4. Click Ok.

## Viewing License

To view a license detail:

1. Navigate to Platform in the Nav tree and double-click License Manager. The License Manager screen is displayed.
2. Under Licenses section, select the required license to view, and click View. The details of the selected license are displayed.

   Or double-click the required license.

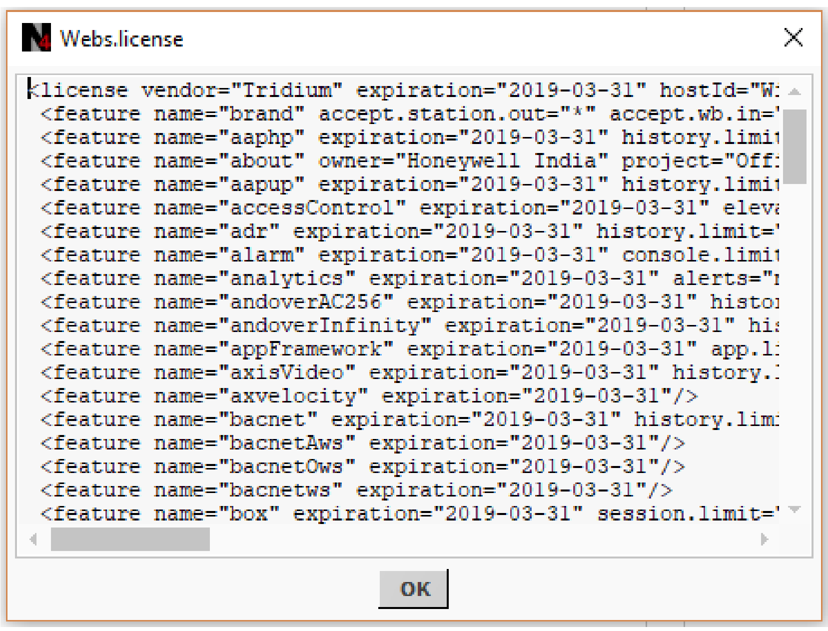



Figure 38: License Details

3. Click Ok.

## Deleting License

To delete a license:

1. Navigate to Platform in the Nav tree and double-click License Manager. The License Manager screen is displayed.
2. Under Licenses section, click the required license to delete, and click Delete.

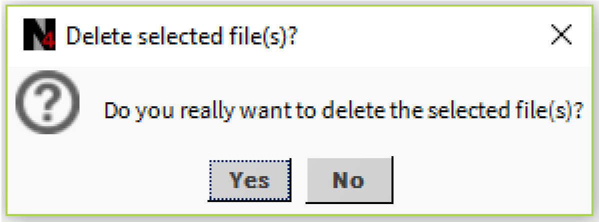



Figure 39: Delete Confirmation

3. Click Yes to confirm the license deletion.

    Or click No, if you do not want to delete the license.

## Importing Certificate

To import a certificate:

1. Navigate to Platform in the Nav tree and double-click License Manager. The License Manager screen is displayed.
2. Under Certificates section, click Import. The Select File window is displayed.

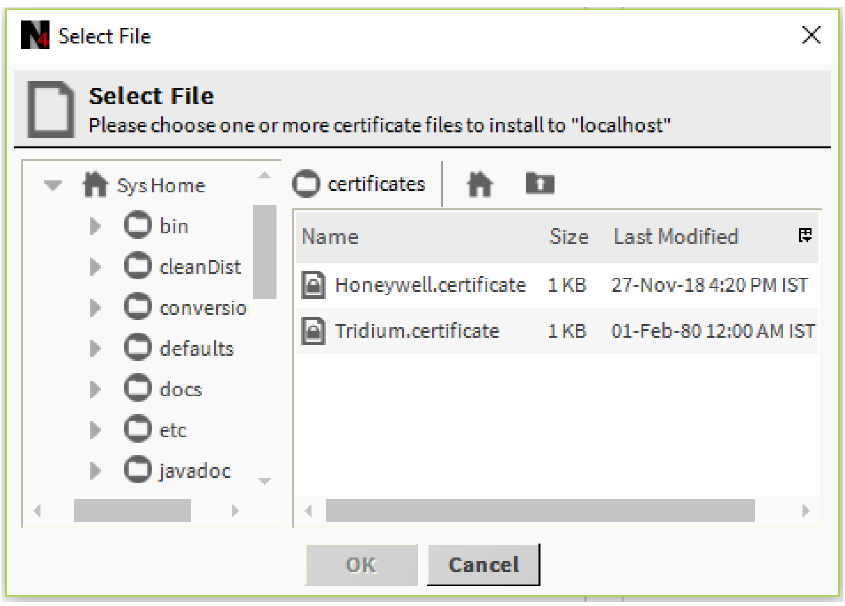



Figure 40: Select File Window

3. Navigate to the required path where the certificate file is present.
4. Select the required certificate file and click Ok. The import status is shown in the window displayed.

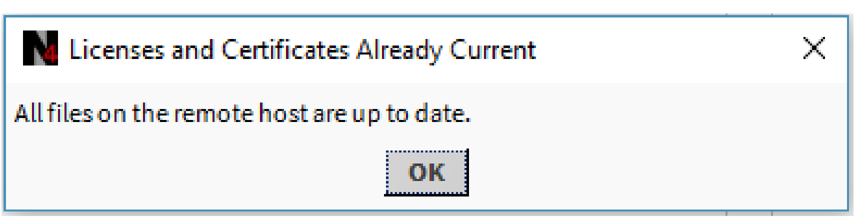



Figure 41: Licenses and Certificates Already Current Dialog Box

5. Click Ok.

## Viewing Certificate

To view a certificate detail:

1. Navigate to Platform in the Nav tree and double-click License Manager. The License Manager screen is displayed.
2. Under Certificates section, select the required certificate to view, and click View. The details of the selected certificate are displayed.

   Or double-click the required certificate.

Figure 42: Certificate Details

3. Click Ok

## Deleting Certificate

To delete a certificate:

1. Navigate to Platform in the Nav tree and double-click License Manager. The License Manager screen is displayed.
2. Under Certificates section, click the required certificate to delete, and click Delete.

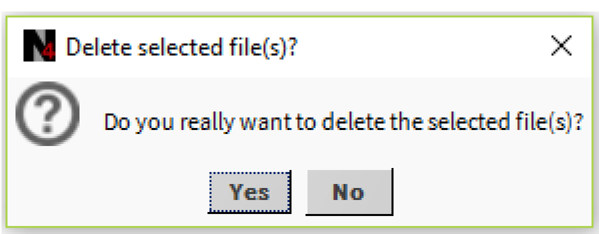



Figure 43: Delete Confirmation

3. Click Yes to confirm the certificate deletion.

   Or click No, if you do not want to delete the certificate.

## Version Compatibility

The version compatibility is similar to Webs. For more information on the operating system, browser, and other compatibility details, see CIPer Model 30 Installation and operation guide and Niagara 4 Installation Guide.

# Migrating Existing Spyder Applications

With Spyder To IPC Migrator tool, you can migrate the existing Spyder applications to CIPer Model 30-compatible applications. The file formats that you can migrate from Spyder backup using the IPC Migrator tool are:

- **Station:** To migrate Spyder controllers in the selected station to CIPer Model 30-compatible applications
- **Library:** To migrate Spyder library to CIPer Model 30-compatible application library
- **Exported Library:** To migrate exported Spyder libraries to CIPer Model 30-compatible applications
- **Custom Palette:** To migrate custom palettes containing Spyder applications to CIPer Model 30-compatible palette

Note:

- In the migration process, only the contents of ControlProgram are migrated, that is, only the control programs and its associated points present in the control program are migrated. The Niagara points that you have discovered in the Spyder application are not migrated to CIPer Model 30 application. So, if you need these points in the CIPer Model 30 application, you need to rediscover the points.
- The IPC Migrator tool does not support Niagara AX version backups. See Prerequisites for more details.
- If the time and schedule are defined in the wire sheet and TR75 also, only one time and schedule get migrated to CIPer Model 30 application. The default schedule, that is, original Schedule block is migrated.
- When a function block contains both in and out parameters, a new user needs to create the loop between that function block and the source component, so that the changes made at one place reflect in the CIPer Model 30 application. If you are existing user and migrating your existing application to CIPer, the migration process creates the linking between the function block and the source component. When a Spyder application is migrated, the migrator tool automatically creates the loop-back link to the component linking to an inout parameter. Refer following examples.
- If any Network Setpoint is connected with the SBus wall module, after migration the setpoint output from the SBus all module bock will be connected with the Fallback of the Network Input to reflect the changes done in wall module in the network Input & vice versa.

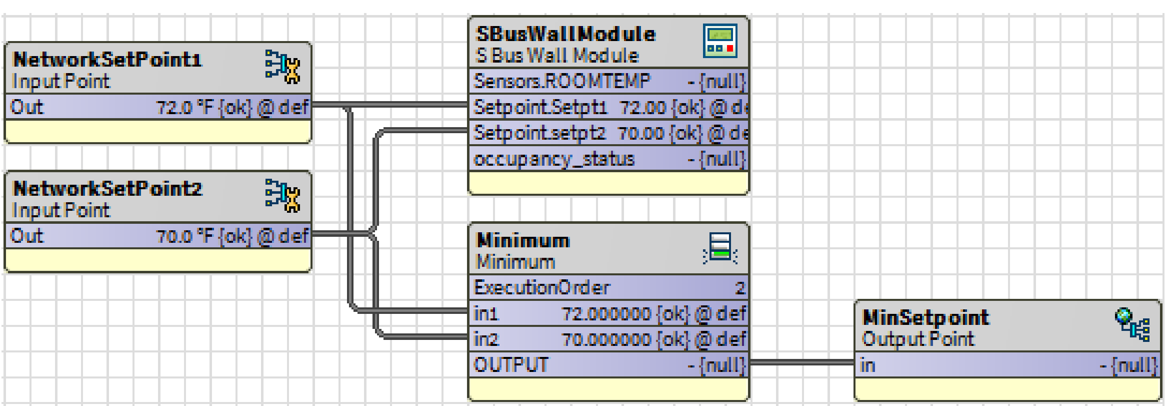



Figure 44: Network setpoints associated with SBus Wall Module

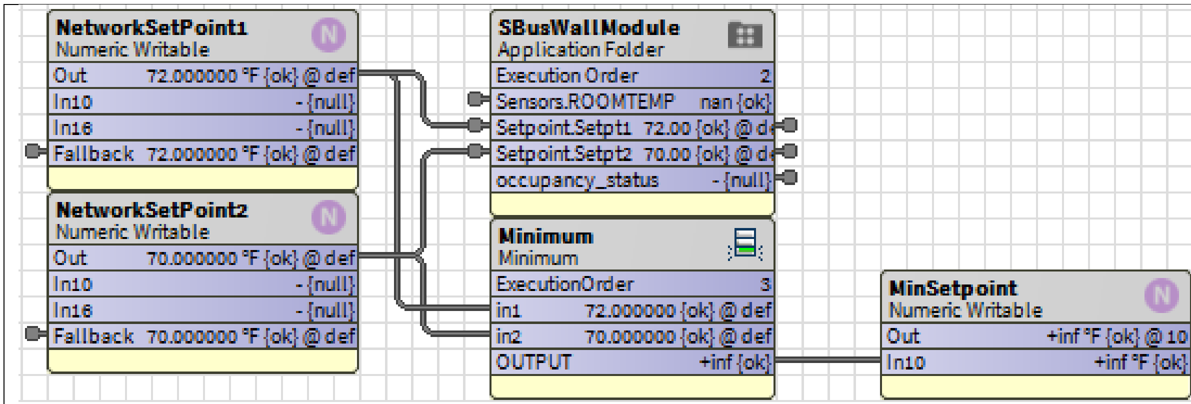



Figure 45: Network setpoints reverse connection with SBus Wall Module after migration

SBusWallModule folder:

In the migration process, the parameters present in the SBusWallModule are segregated as per their types. For example, after migration if you double-click the SBusWallModule block on the wire sheet, you can see different categories of parameters like sensors, setpoints, status, and so on. You can see all the sensor parameters available for the respective SBusWallModule by double-clicking the sensors block. If you are creating a new application, you can segregate the parameters in any required manner.

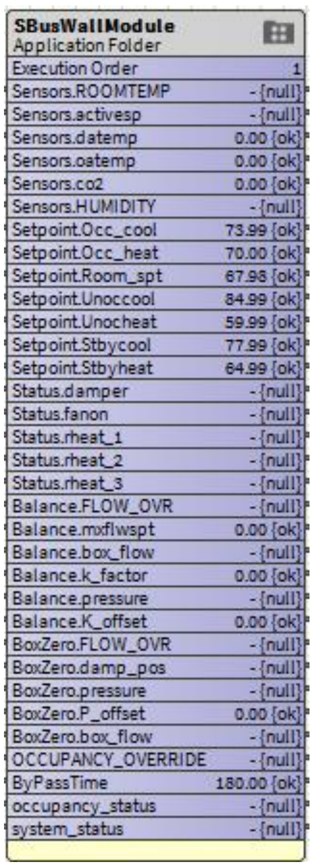



Figure 46: SBusWallModule Function Block

Parameters inside Sensors block

Parameters are created under a folder (folder name is based on the category of a Sylk parameter) and category folder contains the Sylk parameters, which are present inside a folder, SylkDevice. In this case, it is SBusWallModule.

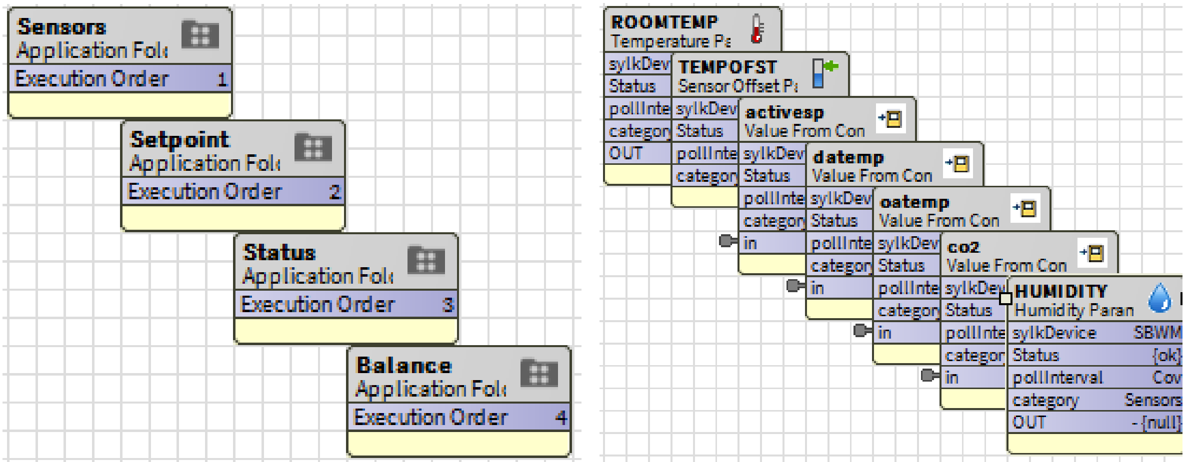



Figure 47: Parameters Inside Sensors Block

## Prerequisites

CIPer Model 30 application is supported in WEBs N4 only. If your current Spyder program is on WEBs AX platform, you must upgrade it to WEBs N4 before you start migration.

The prerequisites for the migration process are:

- AX to CIPer Model 30 Migration:

  Step 1: Migrate the final backup of AX to N4.10.

  Step 2: Migrate the final backup of N4.7 to CIPer Model 30-compatible version 4.10.

- N4 to CIPer Model 30 Migration:

  Step 1: Migrate the final backup of N4 (N4.1, N4.2, N4.3, or N4.4) to N4.10.

  Step 2: Migrate the backup from N4.10 to CIPer Model 30-compatible version 4.10.

# Spyder to IPC Migrator Tool

Follow the below steps to start migration process. Navigate to Tools > Spyder To IPC Migrator tool.

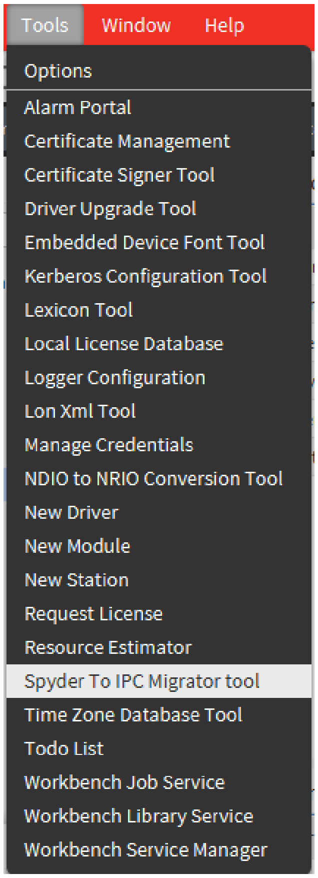



Figure 48: Spyder To IPC Migrator tool in Tools Drop-Down Menu

When you navigate to Tools > Spyder To IPC Migrator tool, default IPC Migrator tool window displayed.

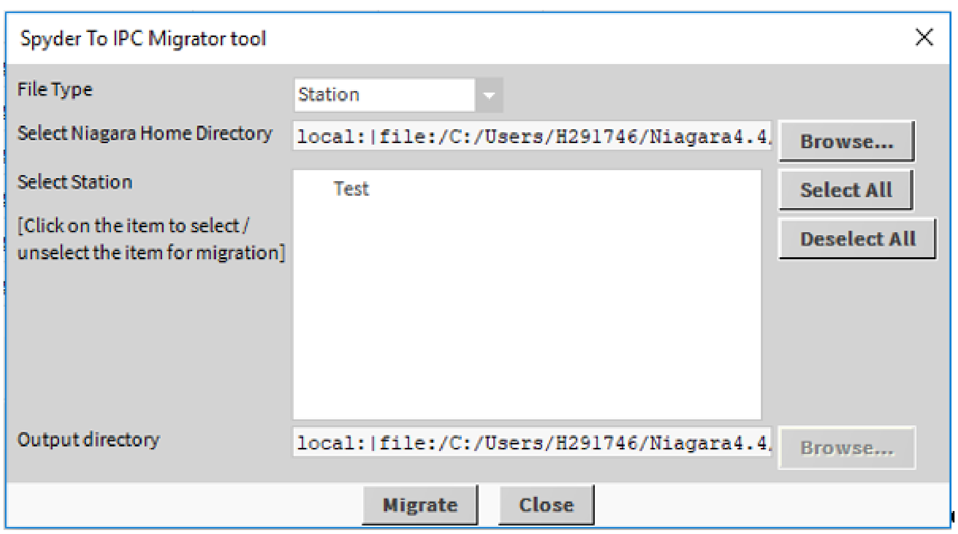



Figure 49: IPC Migrator Tool Window

Migration process from Spyder to Ciper 30 application consist of following steps:

- Migrating Station

- Migrating Library
- Migrating Exported Library
- Migrating Custom Palette
- Copying Migration Results to CIPer Model 30
- Limitations of Spyder to IPC Migrator Tool

Follow the below steps to migrate the Spyder station to a CIPer Model 30-compatible format using IPC Migrator tool.

## Migrating Station

To migrate a station:

1. Select the file type as Station in the File Type drop-down menu.
2. Click Browse next to the Select Niagara Home Directory field to select the path to the Niagara home directory, where the list of stations is available.
3. The Output directory input field displays the location, where the migrated applications are stored. By default, all the migrated applications are stored in the SequencedControlProgram folder.
4. Browse to the Output directory, where the migrated applications are stored. The migrated stations are stored in the MigratedStations folder.

Following figure shows the file type as Station and the selected station to migrate.

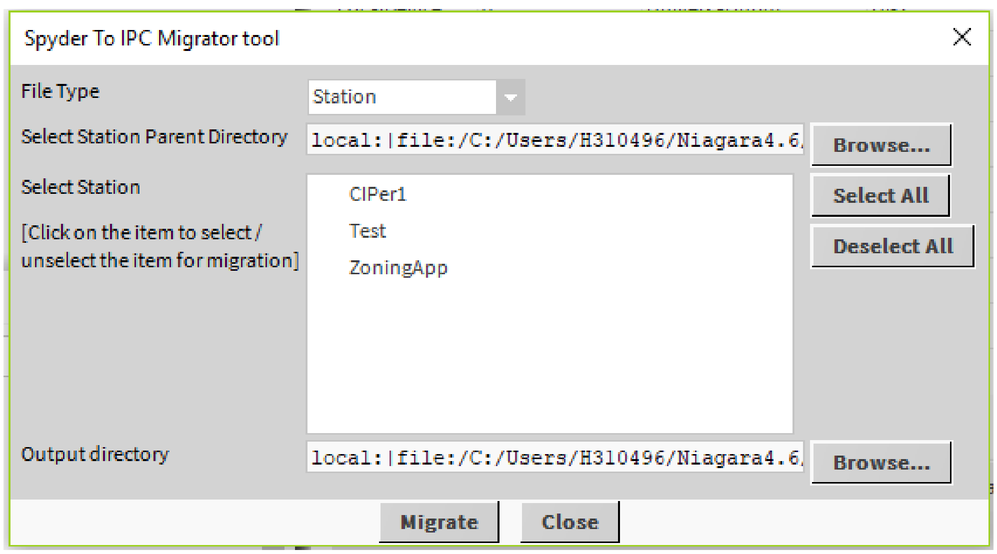



Figure 50: Station to Migrate

## Migrating Library

With the help of IPC Migrator tool, you can migrate the Spyder libraries to a CIPer Model 30-compatible format.

To migrate a library:

1. Select the file type as Library in the File Type drop-down menu.
2. Click Browse next to the Select Niagara Home Directory field to select the path, where the list of libraries is available.
3. Browse to the Output directory, where the migrated applications are stored. The migrated libraries are stored in the MigratedLibraries folder.

   Following figure shows the file type as Library.

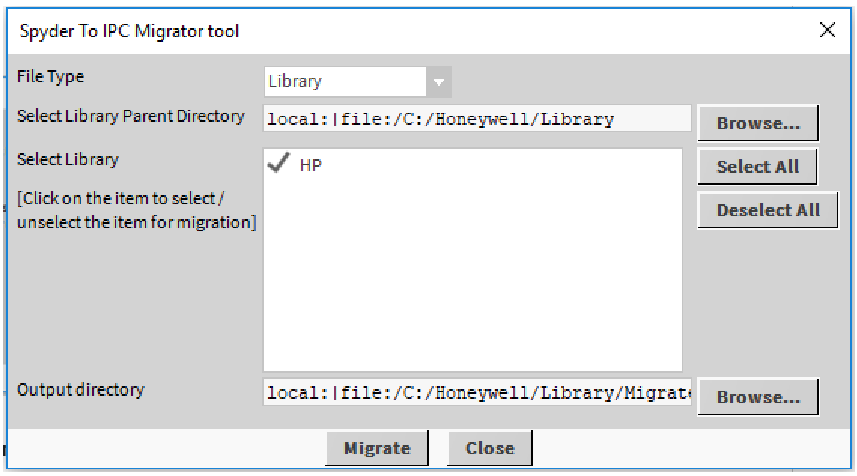



Figure 51: Library to Migrate

Using Library file type under IPC Migrator tool you can migrate the following in the selected library:

- Spyder devices
- Applications
- Macros, which contain some part of the application
- SBus wall module

## Migrating Exported Library

With Export Library option, you can convert the Spyder Export Library into CIPer Model 30-compatible Export Library of applications and on exporting the Spyder library, the application creates a folder with '.slb' extension containing migrated applications.

To migrate exported library:

1. Select the file type as Exported Library in the File Type drop-down menu.
2. Click Browse next to the Select Niagara Home Directory field to select the path, where the list of exported libraries is available.
3. Browse to the Output directory, where the migrated applications are stored.

   Following figure shows the file type as Exported Library.

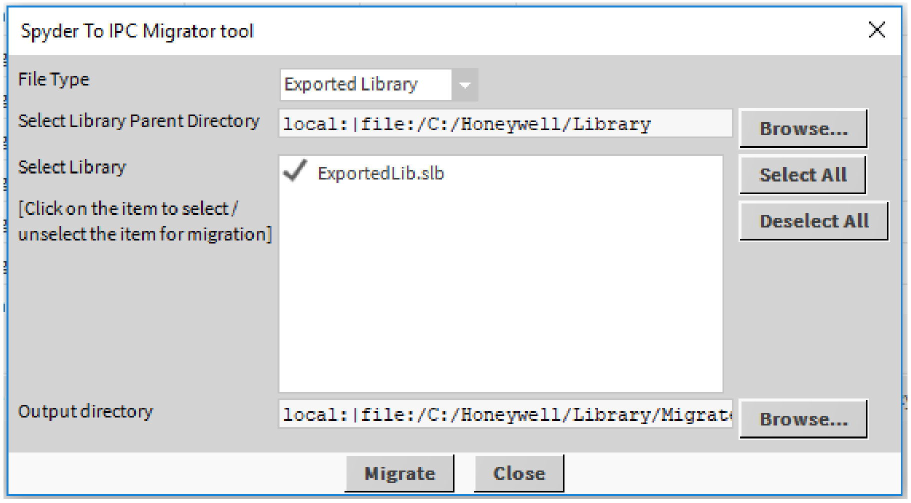



Figure 52: Exported Library to Migrate

The Export to file window enables you to select multiple items to export.

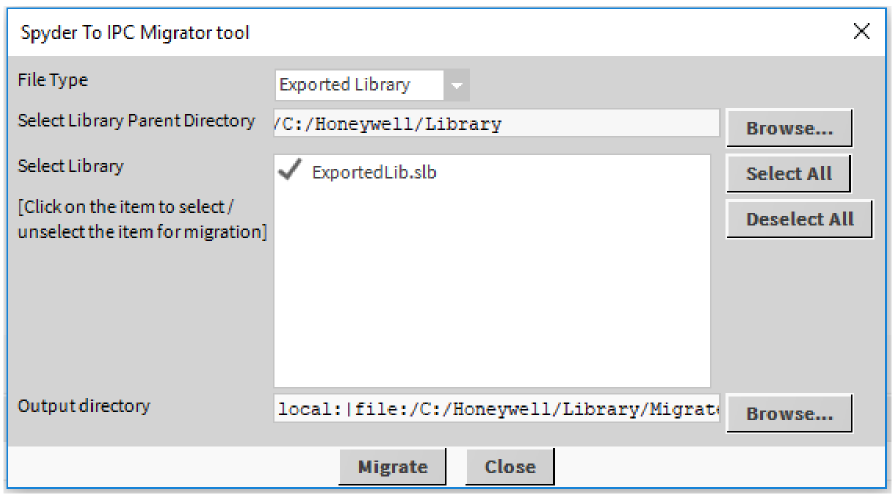



Figure 53: Export to file Window

## Migrating Custom Palette

With the help of the IPC Migrator tool, you can migrate the custom palette files containing Spyder applications. The migrated custom palettes are stored in the MigratedPalettes folder.

To migrate exported custom palette:

1. Select the file type as Custom Palette in the File Type drop-down menu.
2. Click Browse next to the Select Niagara Home Directory field to select the path, where the list of custom palettes is available.
3. Browse to the Output directory, where the migrated applications are stored.

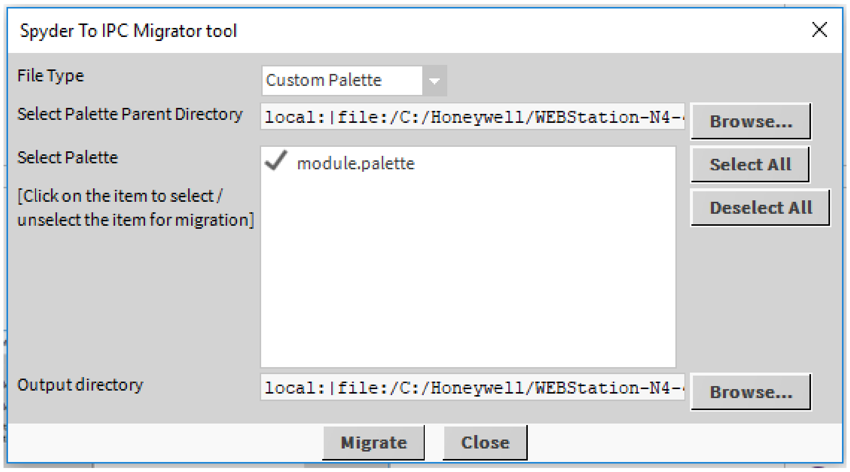



Figure 54: Custom Palette to Migrate

## Copying Migration Results to CIPer Model 30

In the migration process a .bog file is generated in the specified output directory. Once the migration process is completed, you can see multiple .bog files in the specified output directory. One .bog file represents one Spyder application. On expanding these .bog files, you can see IPCNetwork, which you can copy and paste to your new station.

| Note: |
|---|
| You need to transfer KitControl modules (KitControl-rt, KitControl-ux and KitControl-wb) to CIPer Model 30 controller before performing copy paste of migrated IPCnetwork which contains numeric constant. |

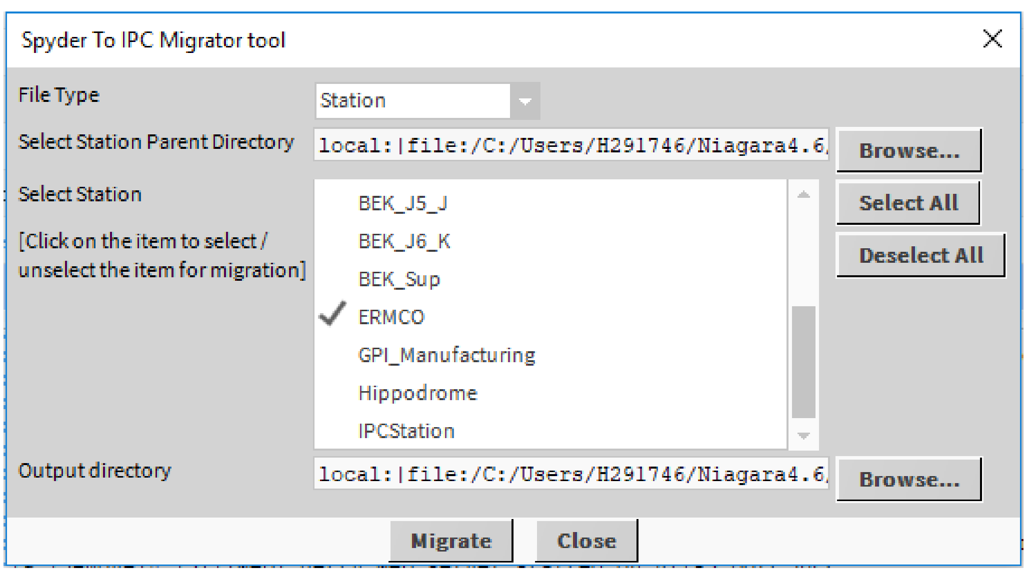



Figure 55: Migrating Station

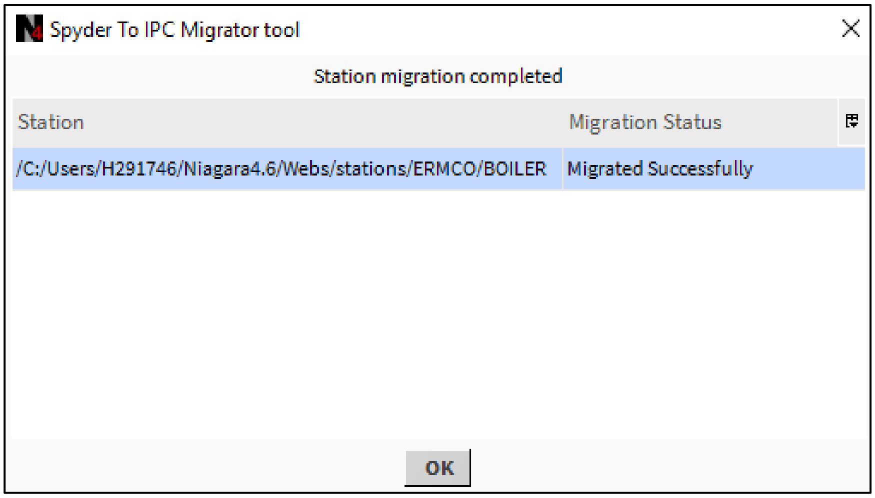



Figure 56: Successful Station Migration

Navigate to C:\Users\user account\Niagara4.7\Webs\stations\MigratedStations.

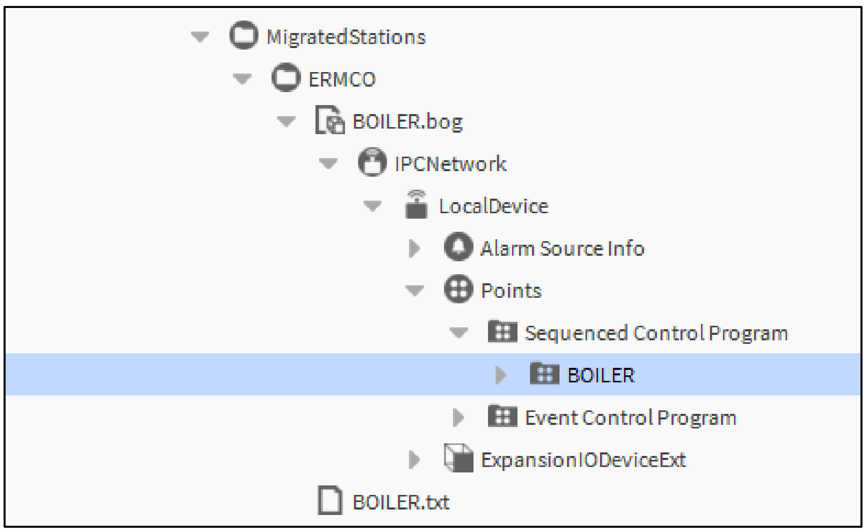



Figure 57: Tree View of bog File Generated after Migration

Copy the logic present in the Sequenced Control Program folder under Nav tree and paste it into the Sequenced Control Program of the current station in CIPer controller.

Note:

- Similarly, you can copy migration results of library, exported library, and custom palette file to CIPer.
- It is recommended that you revisit or review the custom sensor linearization data entries after migration process.
- After the migration process, the unassigned points are assigned to the valid terminals in the CIPer Model 30 software tool.
- Enum names used in Physical points/Network points are not migrated.
- Default value of enum is not migrated. First item in the enum is used as default value.
- Niagara Alarm and History Extensions added to points are migrated and you need to configure those manually.

Sylk:

- Unassigned fixed Sylk point is migrated to a placeholder component (Zelix parameter). You need to correct the component, that is, replace Zelix parameter with an appropriate component and its links accordingly.
- Zelix Sylk Device block is not supported in CIPer Model 30 programming model. Zelix is converted to a BZelixParam placeholder component. You need to correct the component, that is, replace Zelix parameter with an appropriate component and its links accordingly.
- Sylk IOs if any, are migrated as physical inputs/outputs, and next available pin is assigned. COV of Sylk IO is not migrated.
- TR70x device is migrated as TR71x.
- Links to InOut parameters are not bidirectional. Explicit link is created automatically from Out slot of InOut parameter to source component which is connected to the InOut parameter.

Function Blocks:

- Alarm function block is migrated as NumericWritable with Out-of-range alarm extension.
- PriorityOverride block is migrated as a NumericWritable component.
- If the units of the same type of parameters connected with any function block are different then after migration unit converter block will be added ib between of the parameter and the function block For example if the Temperature setpoints connected to the Temperature Setpoint Calculator function block have different units then after migration the unit will remain same but unit converter block are added to convert the parameters unit to support the default unit of the function block of IPC programming tool palette

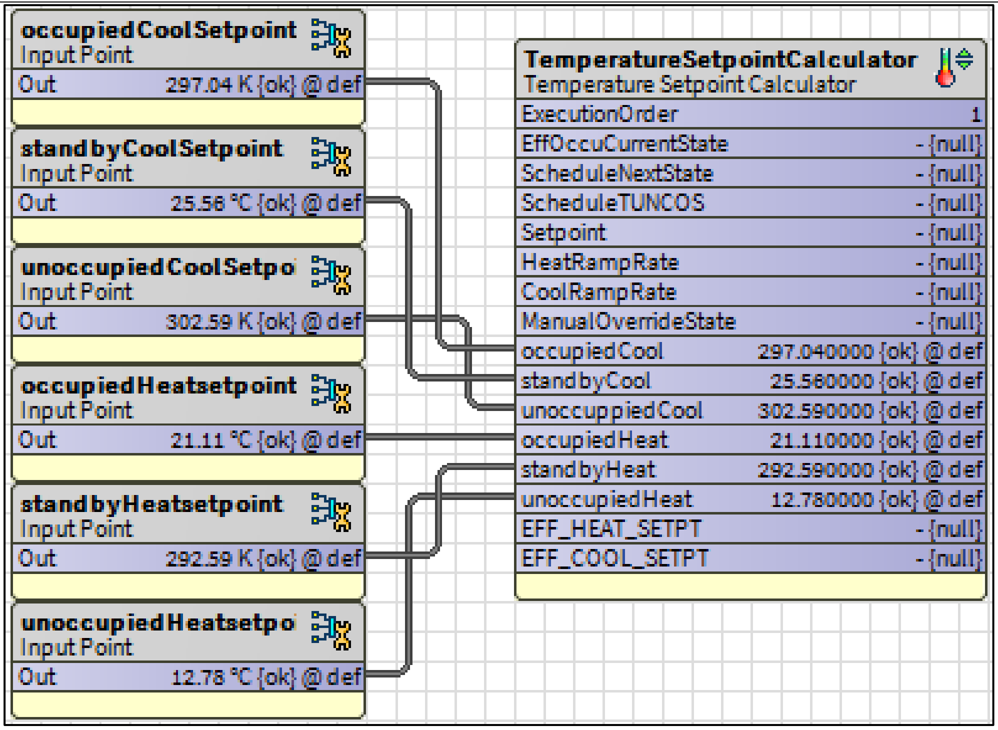



Figure 58: Temperature Setpoint Calculator Function Block

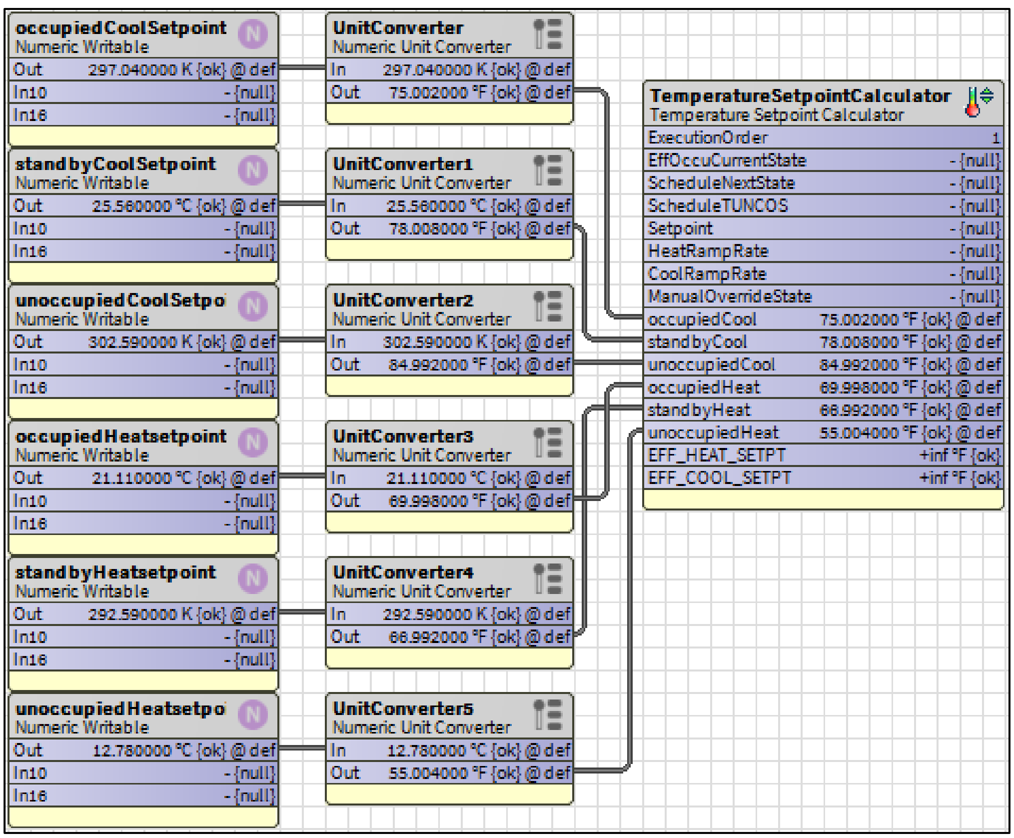



Figure 59: Unit Converter addition after migration

IOs:

- FloatingMotor output/Actuator type is migrated as periodic execution block of type BFloatingMotor.
- Pin assignment for physical inputs/outputs is not preserved during migration. After migration, next available pin is assigned.
- Modulating Inputs of type CustomResistive/CustomVoltage are migrated as CustomSensor
- Sensor Offsets for modulating inputs, if any, are picked up from the Bacnet interface during Library Application Migration.
- Physical Outputs are migrated as NumericWritables. If the migrated application is a Bacnet application, the priority is picked up from Bacnet Object Advanced Settings. Else priority 12 is used as default priority.
- Network Inputs/Outputs are migrated as NumericWritables. Any links to/from Network Inputs/Outputs are linked to/from priority10 slot of the NumericWritable.

## Limitations of Spyder to IPC Migrator Tool

Following are the limitations in the migration process:

### Schedules:

This limitation is applicable for the Weekday/Month for every year option in the Holidays tab. Here you can select a month and date of holiday and those details are migrated as they are.

If the number in the Duration field is more than one then only the first day is migrated to CIPer Model 30 application, because of the complexity in directly mapping the rest of the days following the first day in Niagara 4 version. Remaining days are not migrated, and application shows an error message that you should separately migrate the schedule for remaining days.

For example, you select January from the Select Holiday Start Month drop-down menu, LastSunday from the Select Holiday Start Day drop-down menu, and enter 3 as Duration. In the migration process, out of three last days of January, only one day is migrated, that is, Last Sunday of January. The remaining two days, that is Monday and Tuesday, are not migrated.

### Exception to the limitation:

- If you click Load US Holidays, there can be some holidays which fall in the Weekday/Month for every year category, for example, Thanksgiving and Day After. In this scenario, the limitation related to migration of the schedule is not applicable and the schedule for all days mentioned in the Duration field is migrated.

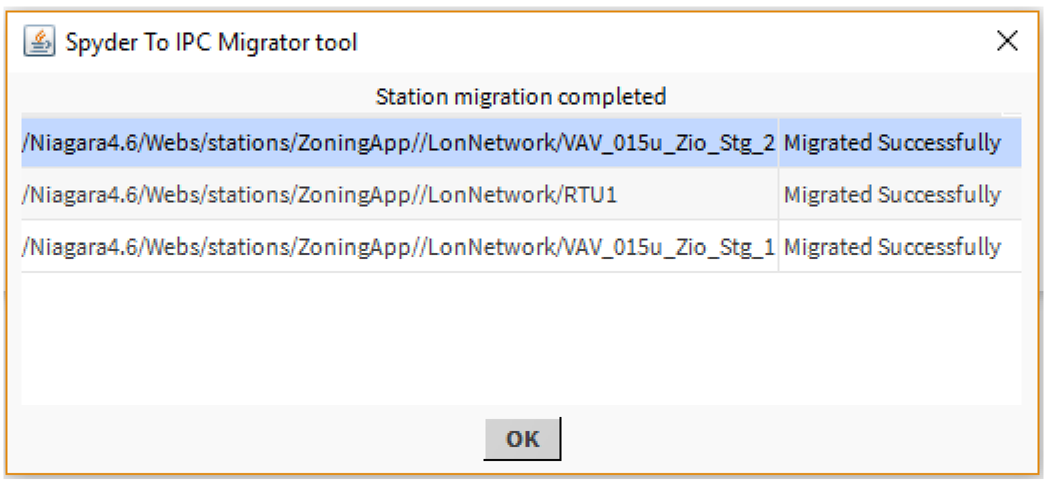



Figure 60: Spyder To IPC Migrator Tool Window

- The Spyder schedule is converted into Niagara Schedule, and the Niagara Schedule does not contain TUNCOS (Time Until Next Change of State) functionality. To overcome this problem, after migration of schedule, the Application Folder is created with the same name of the schedule. A tuncos block is created.in this folder. This block converts the Next Time of Niagara Schedule into TUNCOS of Spyder Schedule. If the block is accidentally deleted, you must use the TUNCOS block present in the Utils folder in the ipcProgrammingTool palette. To know how to add Tuncos block, see to add Tuncos function block onto wire sheet section.
- If Spyder programming does not have any Schedule block in the logic, EnumSchedule is created with default weekly schedule configuration and the component is present under LocalDevice.

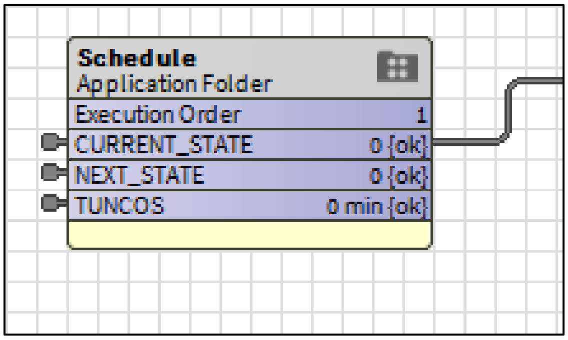



Figure 61: Schedule Function Block

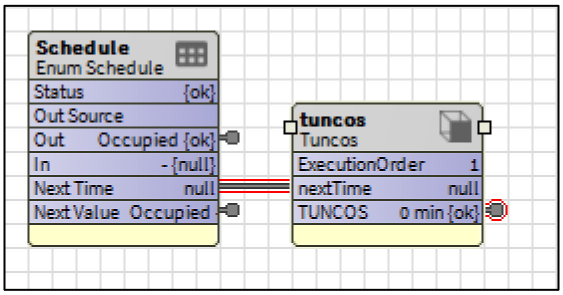



Figure 62: Linking Between Schedule and Tuncos Function Blocks

- Spyder to CIPer Migration will work fine only if the Spyder application being migrated has no validation errors.

# CIPer Model 30 Programming Models

The CIPer Model 30 software tool offers a graphical environment to program the Honeywell CIPer Model 30 programming model

Using CIPer Model 30 software tool and all its components you can add a controller on the required network and create the application as per your requirement.

Continuous Simulation mode is available for testing the application and its function offline. After compiling the program, you can perform operations such as download, upload, and online testing on live controller installed in the field. Following are the components of the CIPer Model 30 controller:

## Local Device Overview

This section provides overview of Local Device property sheet of IPC programming tool.

### Network

It is the communication channel used for CIPer Model 30 controller to communicate with CIPer devices. The IPC network corresponds directly to physical network of the device.

### Device

The local device represents the local interface to the IPC network. There is only one local device and you cannot delete or duplicate it. For example, LocalDevice in the CIPer Model 30 application. Following are the properties of the LocalDevice folder.

### Status

This property reflects the status of the local device. This field is not editable.

### Enabled

To enable or disable the local device. You can set it to true to enable and to false to disable.

### Fault Cause

You can set the fault cause message. This field is not editable.

### Health

The health of the local device contains information on the device state and time and date when that state was captured or noted, the condition of the local device-if it is in working condition or not (up or down), alarm, the last time when the local device was in Ok status, the last time when the local device was failed, and the cause of the failure. All these fields are not editable and are auto-generated.

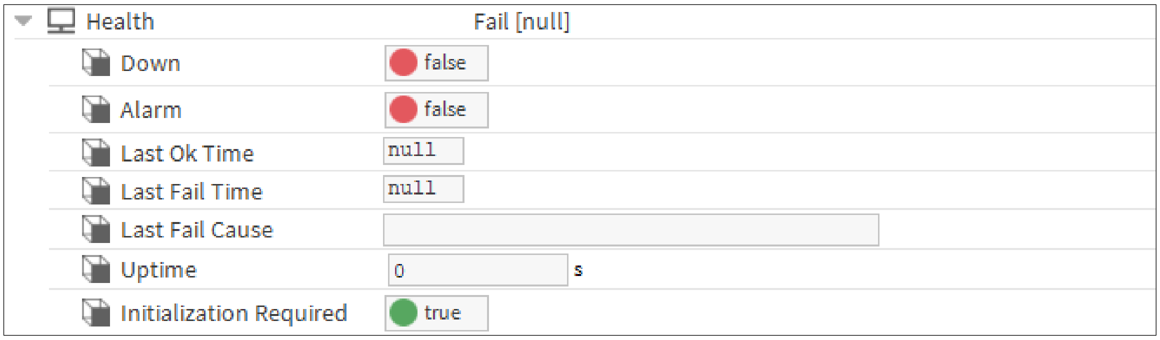



Figure 63: Health Properties

## Alarm Source Info

This section of the property sheet provides the details on the alarms.

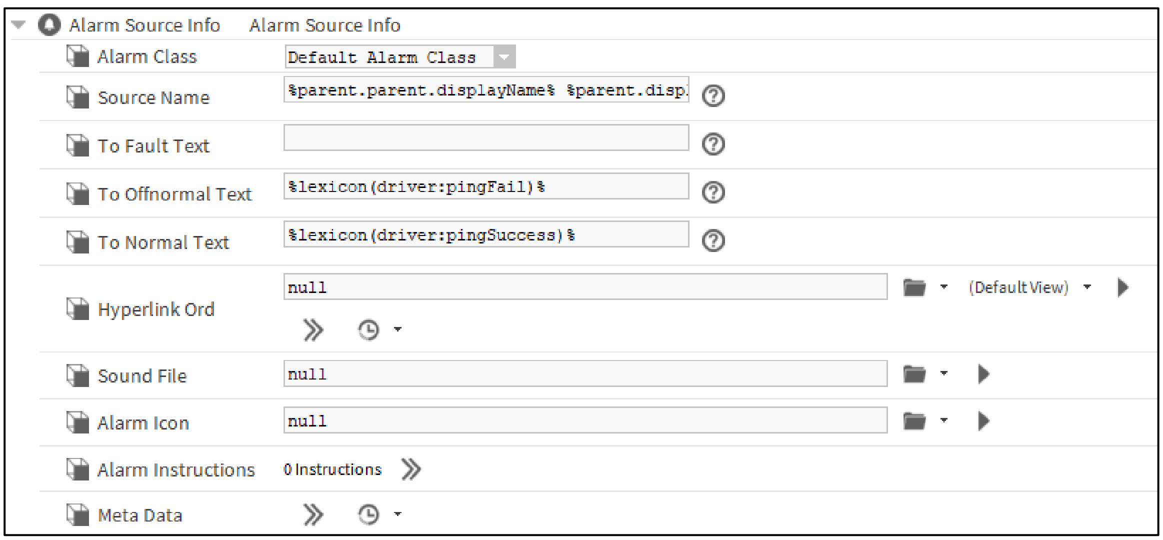



Figure 64: Alarm Source Info Properties

## Alarm Class

Specifies or returns the alarm routing option for the component.

## Source Name

Displays the name in an alarm record that identifies the source of the alarm.

## To Fault Text

The text to display when the component transitions to a Fault status. When applicable, text entered for Fault Algorithm, High Limit Text and/or Low Limit Text may override this text.

### To Offnormal Text

The text to display when the component transitions to an Offnormal (alarm) state. When applicable, text entered for Fault Algorithm, High Limit Text and/or Low Limit Text may override this text.

### To Normal Text

The text to display when the component transitions to a Normal status. When applicable, text entered for Fault Algorithm, High Limit Text and/or Low Limit Text may override this text.

### Hyperlink Ord

Associates an ord, BLQ query or path with an alarm state on the component. When an alarm is reported in the console, the Hyperlink button activates. Clicking this button links to the location you specify here.

### Sound File

The path to a sound file that plays when the current component is in an alarm state. Use the folder icon to browse to the file. Click the arrow icon to the right of the folder icon to test the path.

### Alarm Icon

Defines the path to a graphic file the system includes in the Timestamp column of the alarm table in the Console Recipient view. Use the folder icon to browse for the file. Use the right-arrow to test the location you entered.

### Alarm Instructions

Advice that accompanies the alarm notification (Alarm Record window) that provides important information for the operator. Click the right-pointing arrow to view the instructions.

### Meta Data

Allows you to enter new facets for the extension.

## FirmwareDetails

The Firmware Details of the local device contains information on the firmware minor version, major version, installed patch version, installed build number, firmware health status, details of any fault occurrence, & its updating logs.

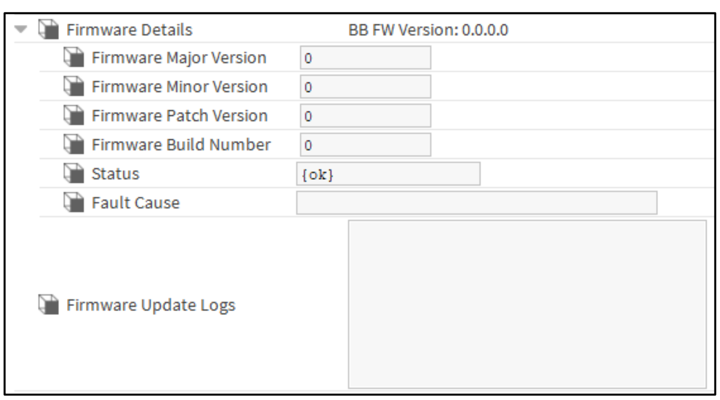



Figure 65: Firmware Details Properties

## Model

Select the Model from the Model drop-down menu.

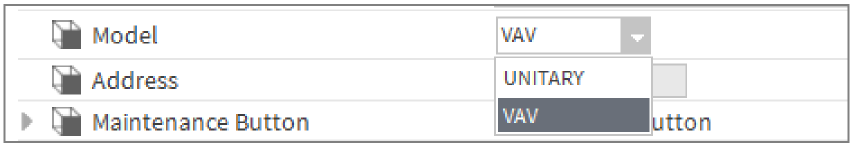



Figure 66: Model Selection

| Model | Model Number | Description |
|---|---|---|
| UNITARY | WEB-C3036EPUBNH | CIPer IP Unitary Controller, BACnet, 150 Point/4-Device Niagara 4 License, SMA |
| VAV | WEB-C3036EPVBNH | CIPer IP VAV Controller, BACnet, 150 Point/4-Device Niagara 4 License, SMA |

Modify the address as required.

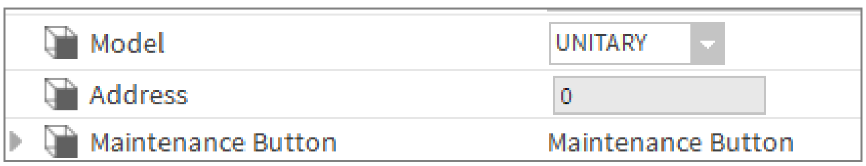
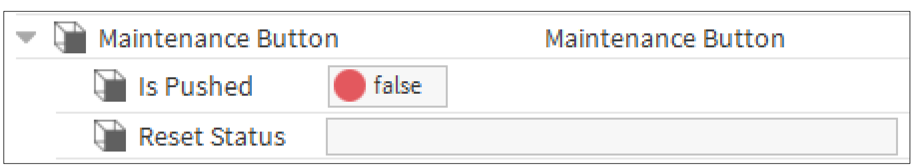



Figure 67: Device Address

## Maintenance Button

The maintenance button will turn to True of the Maintenance button is pressed on the connected controller. Once it becomes true the operator need to reset it to false by doing the mentioned steps. Select Maintenance Button-> Right click-> Action-> select Reset Maintenance Button Status.




.

Figure 68: Maintenance Button Properties

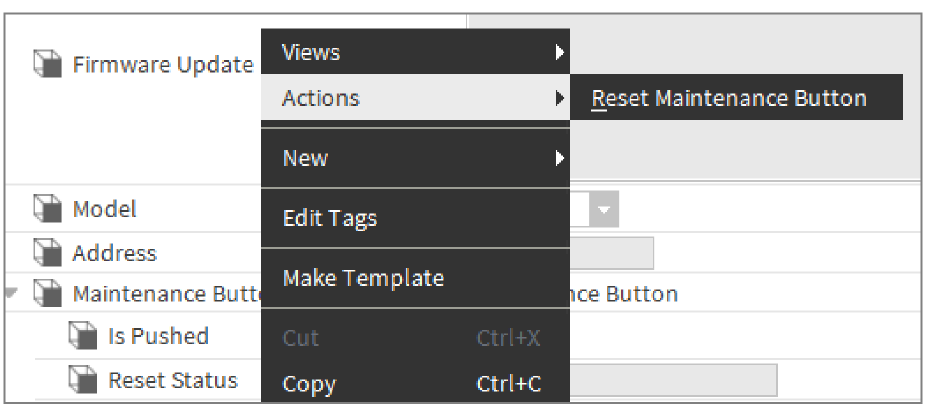



Figure 69: Maintenance Button Status Reset

## RSTP Configuration

The Rapid Spanning Tree Protocol (RSTP) configures the Port State of each Bridge Port in the Bridge Local Area Network.

RSTP is faster than STP in terms of convergence when topology changes occur. The loop-free topology ensures that there are no broadcast storms and duplicate frame transmission.

Table 9: Spanning Tree Algorithm Timer Values

| Parameter | Recommended or default value | Fixed value | Range |
|---|---|---|---|
| Bridge Hello Time | 2.0 | — | 1.0-10.0 |
| Bridge Max Age | 20.0 | — | 6.0-40.0 |
| Bridge Forward Delay | 15.0 | — | 4.0-30.0 |
| Hold Time | — | 1.0 | — |

NOTE

- It is always recommended for a user to keep Bridge Hello Time default value 2 seconds. Change the default value cause performance issue.
- Bridge Max Age recommendation
  - If user has less than 20 RSTP devices connected, keep the Bridge Max Age to default value 20 seconds.
  - If user has more than 20 RSTP devices connected, keep the Bridge Max Age to be the same as the devices' number that has RSTP enabled.

    Example: If 30 RSTP devices are connected in the environment, set the Bridge Max Age to 30 seconds. The max value is 40s.
- Bridge Forward Delay should be changed accordingly when Bridge Max Age change.

  Rule: 2 × (Bridge_Forward_Delay – 1.0 seconds) >= Bridge_Max_Age

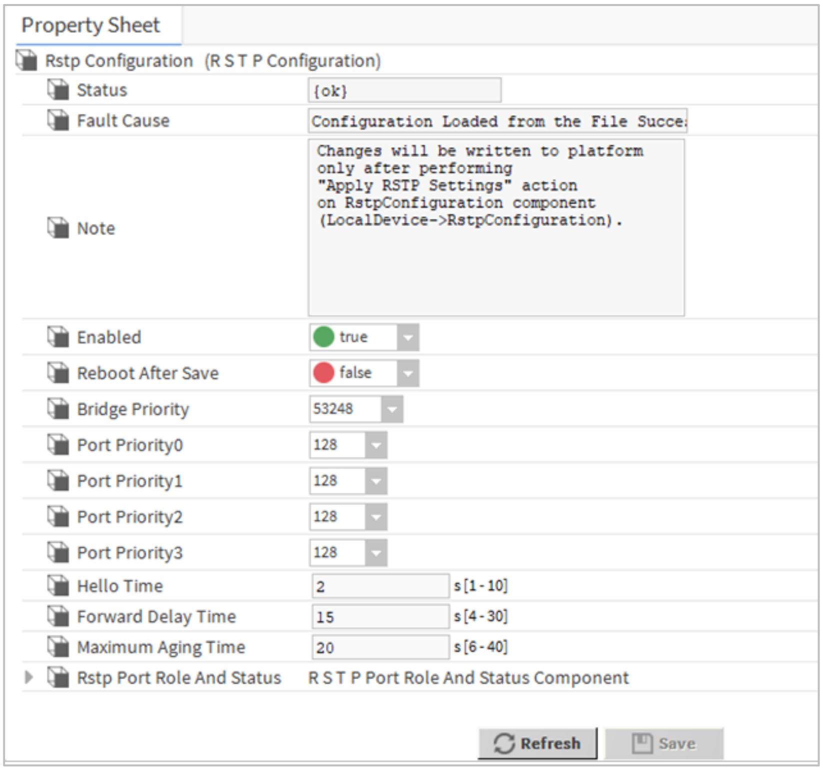
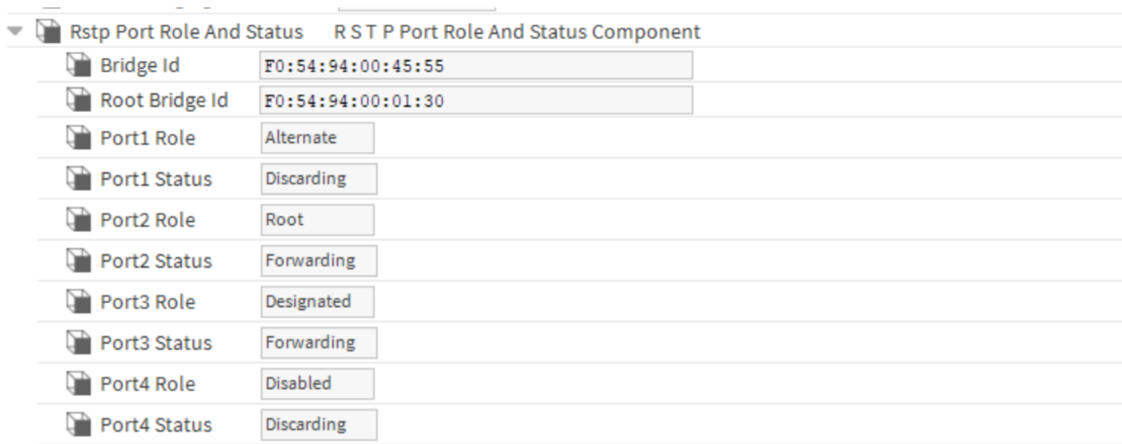



Figure 70: RSTP Configuration Properties

| Property | Description |
|---|---|
| Status | Read-only. Displays the status of the RSTP configuration(ok/fault). |
| Fault Cause | Read only. Displays the fault, that caused the RSTP configuration to go into the fault state. |
| Note | Read only. Displays the information, “changes will be written to the platform after performing Apply RSTP Settings on the RSTP configuration component”. |
| Enabled | Enables/Disables the spanning tree protocol. |
| Reboot After Save | Specifies if a controller will reboot after the configuration is saved<br>True - A reboot is required after saving the RSTP configuration change.<br>False - No reboot is required after saving the RSTP configuration change. |
| Bridge Prior-ity | The bridge priority for forwarding the packets. Range = 0 to 61440, 0 is highest priority. |
| Port Priority | Port Priority0 - The port priority for port 0. Range = 0 to 240, 0 is highest priority.<br>Port Priority1 - The port priority for port 1. Range = 0 to 240, 0 is highest priority.<br>Port Priority2 - The port priority for port 2. Range = 0 to 240, 0 is highest priority.<br>Port Priority3 - The port priority for port 3. Range = 0 to 240, 0 is highest priority. |
| Hello Time | The Hello Time interval between transmissions of configuration messages by the root device. Range = 0 to 10 seconds, default = 2 seconds. |

<table>
<tr><td>Forward Delay Time</td><td>The maximum amount of time for which the root device waits before changing the states. Range = 4 to 30 seconds, default = 15 seconds.</td></tr>
<tr><td>Maximum Aging Time</td><td>The maximum length of time old messages remains on the network. This will ensure that old messages do not circulate endlessly on the network. Range = 6 to 40 seconds, default = 20 seconds.</td></tr>
<tr><td>Rstp Port Role And Status</td><td>Read only. Displays the role and status of each port, that is connected and configured with the device.



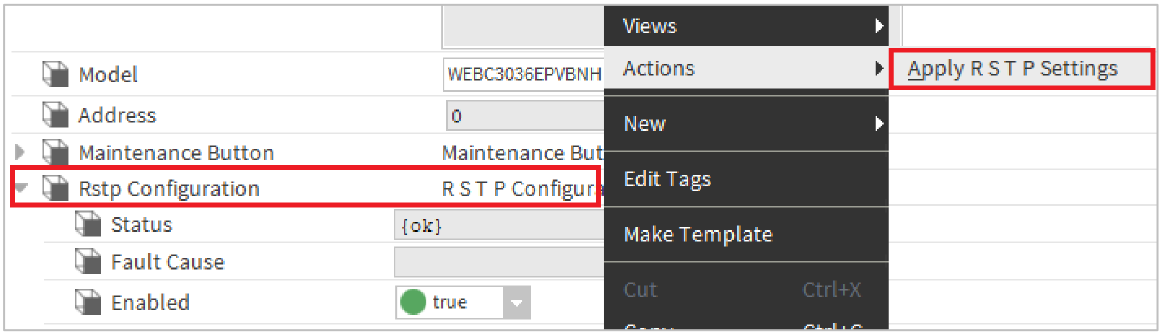

RSTP adds new bridge port roles to speed convergence following a link failure. The number of states a port can be in, has been reduced to three instead of STP’s original five.

RSTP standard Port Roles:
<ul>
<li>Root - A forward port that is the best port from non-root bridge to root bridge.</li>
<li>Designation – A forwarding port for every LAN segment</li>
<li>Alternate – An alternate path to the root bridge. This path is different from using the root port</li>
<li>Backup – A backup/redundant path to a segment where another bridge port already connects</li>
<li>Disabled – Not strictly part of STP a network administrator can manually disable a port.</li>
</ul>
RSTP switch port States:
<ul>
<li>Discarding – No user data is sent over the port</li>
<li>Learning – The port is not forwarding frames yet, but is populating its MAC-address-table</li>
<li>Forwarding – The port is fully operational</li>
</ul></td></tr>
</table>

After changing the RSTP properties, it is necessary to Apply RSTP Settings, to save the changes in the CIPer 30 controller.

Procedure

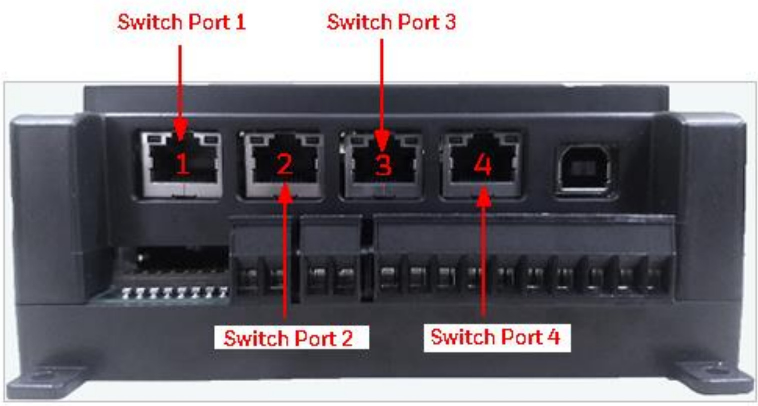
1. Select Rstp Configuration and right click Actions and select Apply RSTP Settings.




| Note: |
|---|
| If the user reboots the station without performing the Apply RSTP settings action, any RSTP property changes will not be saved in the CIPer 30 controller. |

## Switch Port Configuration

The CIPer Model 30 controller has a 4-port Ethernet IP switch. The Switch Port Configuration property sheet allows the user to configure the ports. The Switch port location on the CIPer Model 30 controller is shown below.



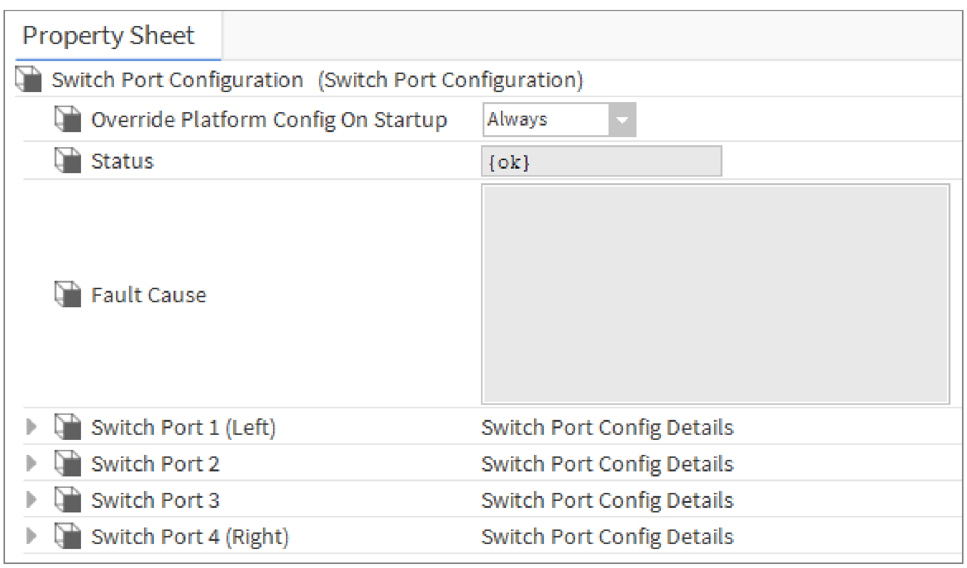

Switch Port Configuration allows the user to configure the CIPer Model 30 controller's ports using IPC programming tool.

Switch Port Configuration has the following features:

- Enable/ disable ports
- Restrict access to each port to specified MAC addresses
- Allows user to enable or disable MAC Address Filtering. For best security practices, refer CIPer Model 30 Hardening Guide - 31-00207, APPENDIX D section.
- Allows user to configure "Allowed MAC Addresses"

<u>To configure a Switch Port:</u>

1. Navigate to **IPCNetwork > LocalDevice** and click **Switch Port Configuration**. This opens the Switch Port Configuration property sheet.



2. In the Override Platform Config box specify whether the Ethernet port configuration in the station overwrites the Ethernet port configuration in the platform.

   – Always - The Ethernet port configuration from the station will be copied to the platform each time an IPCNetwork is enabled in CIPer Model 30 controller, each time an enabled IPCNetwork is added to CIPer Model 30 controller, or each time CIPer Model 30 station is started with an enabled IPCNetwork.

   – Only Once - The Ethernet port configuration from the station will be copied to the platform the first time an IPCNetwork is enabled in CIPer Model 30 controller, the first time an enabled IPCNetwork is added to CIPer Model 30 controller, or the first time CIPer Model 30 station is started with an enabled IPCNetwork.

   – Never - The Ethernet port configuration from the platform will be copied to the station each time an IPCNetwork is enabled in CIPer Model 30 controller, each time an enabled IPCNetwork is added to CIPer Model 30 controller, or each time CIPer Model 30 station is started with an enabled IPCNetwork.

**Note:**

- The Ethernet ports are controlled by the platform's Ethernet configuration therefore copying the configuration from the station to the platform (Always and Only Once) will mean the configuration defined in the station is used. Similarly copying the configuration from the platform to the station (Never) will set the station's Ethernet port configuration to be the same as the platform’s Ethernet port configuration meaning the platform's Ethernet port configuration will be used.
- When Override Platform Config is set to Always or Only Once, if there is an error in the port configuration an enabled IPCNetwork is added to CIPer Model 30 controller, or the CIPer Model 30 station is started with an enabled IPCNetwork., the switch configuration will not be written to the platform.
- When Override Platform Config is set to Never, controller station starts, or the IPC network is enabled the switch configuration will be read from the platform and will override the switch port configuration on the station.

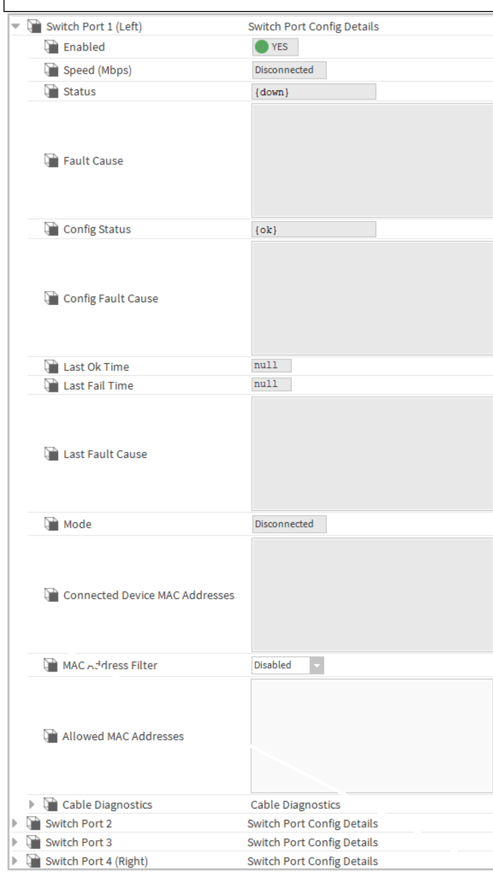
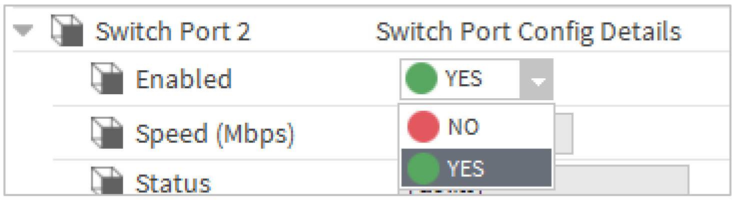
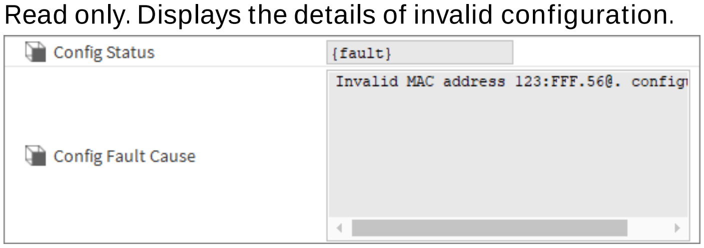
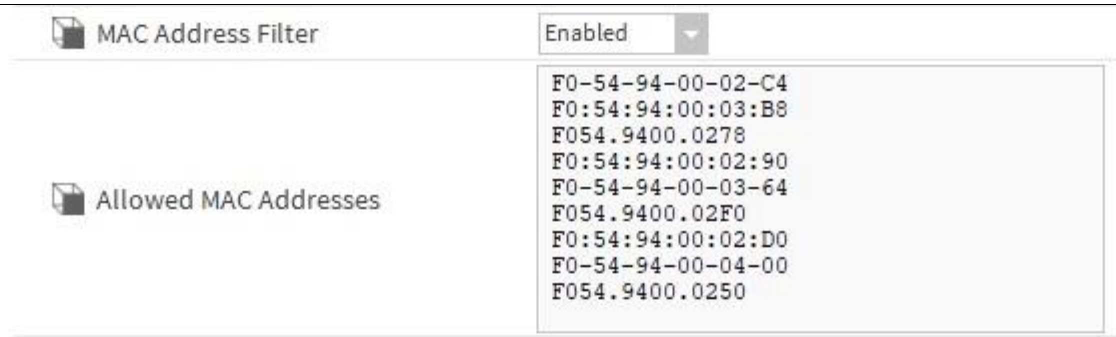
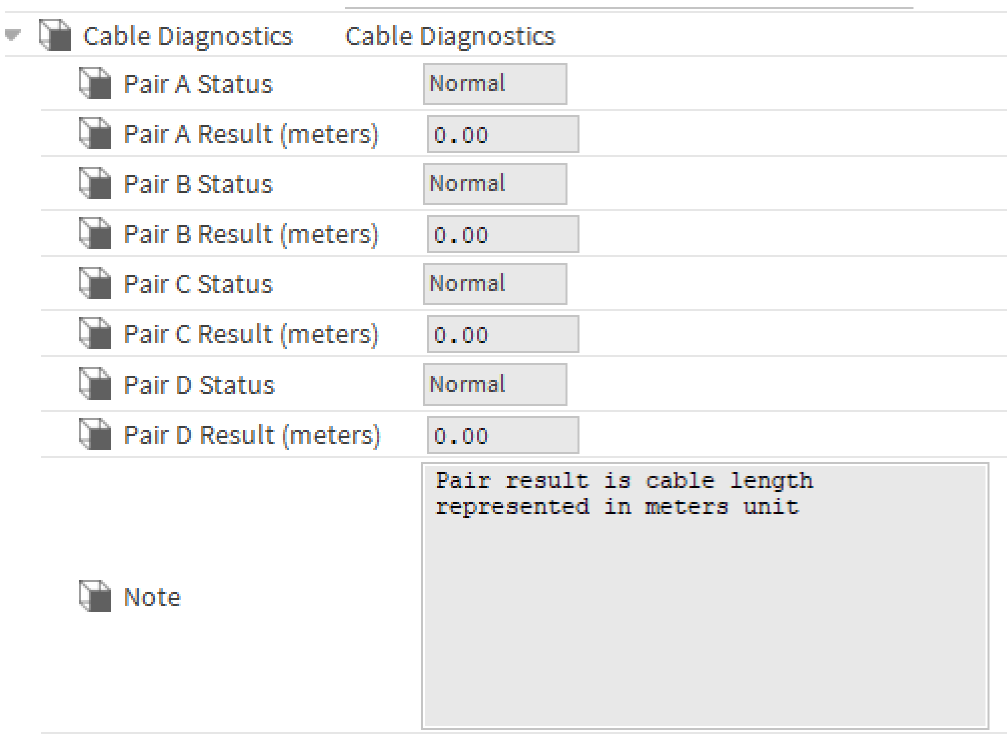



Figure 71: Switch Port Configuration Properties

| Property | Description |
|---|---|
| Enable | Switch Port 1(left) is read only and always enabled, this prevents the port from being disabled, ensure that it is not possible to be locked out of the controller.<br>**Note:**<br>Switch Port 2, Switch Port 3, and Switch Port 4(Right) are configurable and can be individually enabled/disabled.<br><br>Changes to this parameter will take effect immediately after saving. Ensure that the right users get access to enable or disable the switch ports. |
| Speed (Mbps) | Read only. Displays the connectivity speed of the connected device (10/100/1000/Disconnected). |
| Status | Read only. Displays the physical status of the port (OK/Down). |
| Fault Cause | Read only. Displays the reason port is in Down/Fault state. |
| Config Status | Read only. Displays the configured status<br>OK - There are no configuration errors on the port<br>Fault - The port configuration is invalid |
| Config Fault Cause | Read only. Displays the details of invalid configuration.<br> |
| Last Ok Time | Read only. Displays last time when the port is in the OK state. |
| Last Fail Time | Read only. Displays last time when the port in Down or Fault state |

| | |
|---|---|
| Last Fault Cause | Read only. Displays the details of the last reported invalid configuration reported, which caused controller to go into the Down or Fault state:<br>• Link Down – A device is connected to CIPer Model 30 controller port, but the controller does not detect a device connected to the port.<br>• Cable – There is issue with device cable connection.<br>Note:<br>CIPer Model 30 controller has auto sensing feature, which allows to detect the ethernet cable connection. |
| Mode | Read only. Displays the device communication mode (Full/ Half/Disconnected). |
| Connected Device MAC Addresses | Read only. Displays the MAC address(s) of devices connected to the port.<br> |
| MAC Address Filter | Enables/Disables MAC addresses filtering.<br>• Enabled- Only devices whose MAC address listed in the Allowed MAC Address property will be able to communicate with the controller.<br>• Disabled(default) – All the devices connected to the port can communicate with the controller.<br>Note:<br>Enabling or Disabling the MAC address filter will have immediate effect, no restart is required. |
| Allowed MAC Addresses | The MAC address of devices that user wants to allow to communicate with the CIPer 30 controller.<br>✓ Only the following formats allowed<br>▪ FF:FF:FF:FF:FF:FF<br>▪ FF-FF-FF-FF-FF-FF<br>▪ FFFF.FFFF.FFFF<br>✓ Maximum of 16 MAC address<br>✓ Each MAC address must be on a separate line<br>✓ Duplicate MAC addresses are not allowed.<br>**NOTE:** Changes in the **Allowed MAC Addresses** property will have immediate effect, no restart is required. |

<table>
<tr>
<td></td>
<td>


</td>
</tr>
<tr>
<td>Cable Diagnostics</td>
<td>
Read only. Displays the cable diagnostics status for all the four cable pairs.
Pair Status – The staus of the cable pair (Normal / Open / Short / TestFailed)
Pair Result – The length of the cable, which is basically the location of the issue in the cable, that caused the port to be in the Down state.
If any of the pairs display Open / Short / TestFailed status, then the port status is Down.
If all of the pair display Normal status, then the port status is OK and port result is 0.00.


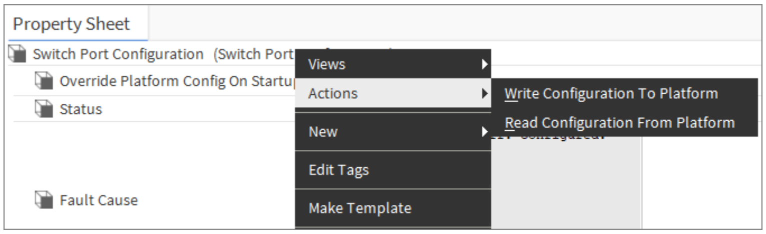
Note:
<ul>
<li>When a port is connected to a device port with a speed of 100Mbps, the Switch Port configuration may display inaccurate Cable Diagnostics. Sometimes it may display inaccurate cable data length for pairs A, C, and D or display link failure status.</li>
<li>When the pair status is reported as normal, the corresponding pair result might report inaccurate data, which can be ignored as the links will be working correctly, if the status is normal.</li>
</ul>
</td>
</tr>
</table>

Once the changes to the Switch Port Configuration are saved in the property sheet, the switch configuration will be saved in controller platform.

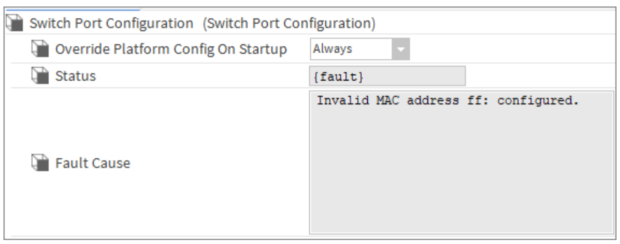
**Write Configuration To Platform** – Writes the switch port configuration from the station to platform.

**Procedure**

3. Right click on **Switch Port Configuration**, select **Action**, and click **Write Configuration To Platform**.

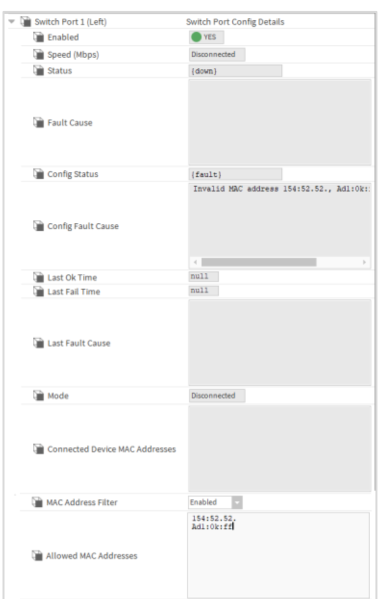



There are errors in the switch port configuration, the switch port configuration status will be in the Fault state. The changes will not be written in the controller platform until the switch port configuration errors are fixed.




- Status – Read only. Displays the outcome of configuration for all the ports.
  - Ok - All the ports have a valid configuration.
  - Fault - At least one port has an invalid configuration
- Fault Cause - Read only. Displays the details of the fault state.

**Config Status** and **Config Fault Cause** of each port will have details which will help fix the invalid configuration of the switch port.



> **Note:**
>
> - If the Config Status of switch port configuration enters the fault state because of changes made by user, the changes will not be written to the platform.
> - Once the Config Status returns to OK state i.e. configuration errors are corrected, all the changes made after the switch port configuration entered the fault state will be written to the platform.

Example: In a station, while adding a new MAC address of a device in the 4th switch port, you enter an invalid MAC address. After saving the Switch Port configuration status goes to fault state.

Without fixing the error in the Switch Port configuration, you modified the 1st and 3rd switch port configuration. When you click save, these changes will not be written to controller platform, as the switch port configuration is in the fault state.

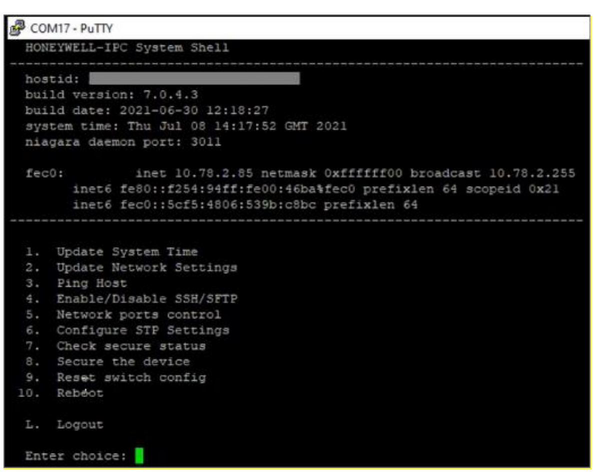
To fix the error, you need to check the Config Status and Config Fault Cause of each port to identify the issue and correct it.

Once the error is fixed and Switch port configuration returns to the OK state the properties that were changed in the 1st and 3rd port 's configuration, while the Switch Port configuration was in the fault state will be written to the controller platform.

Read Configuration From Platform – This action will read switch configuration from platform and overrides switch port configuration in station.

Procedure

1. Right click on **Switch Port Configuration**, select **Action**, and click **Read Configuration From Platform**.

> **Note:**
>
> - If the user has enabled “MAC Address Filter” on all four ports and not configured “laptop/supervisor/programming computer” in any one of the port, then the platform will allow connections only from the MAC addresses that are configured in “Allowed MAC Addresses”.
> - If the user forgets the MAC address of the configured laptop/supervisor/programming computer. Use Serial connection through USB, to reset the switch configuration to default factory settings and can get access to the controller.
>
> Steps to reset switch configuration.
>
> 1. Connect the CIPer Model 30 controller with the serial console using the terminal emulators tool. Refer to Serial Shell option available in CIPer Model 30 controller.
>
> 
> 

2. In the IPC System Shell, choose “9 Reset switch config” and type Yes <Y> or No<N> to double confirm.

After resting the switch configuration, reboot the controller.

## Connected Device Known Behavior

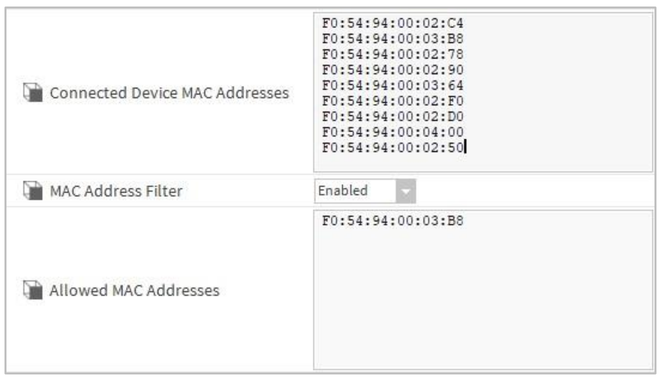
The Connected Device MAC address property displays all learned MAC address on the port, irrespective whether the address is allowed or not.

Example: In the example below, the port receives the packets from the devices with these MAC address, so the source address is learned and displayed.

Whether these devices can communicate successfully with the CIPer 30 controller depends on the configuration of MAC Address Filter and allowed MAC Addresses.

When the MAC Address Filter set to Enable, only MAC Address listed in the Allowed MAC Addresses property (F0:54:94:00:03:B8) can communicate with the CIPer 30 controller.

Similarly, when the MAC Address Filter set to Disable, all the devices connected to the controller’s port can communicate with the CIPer 30 controller.

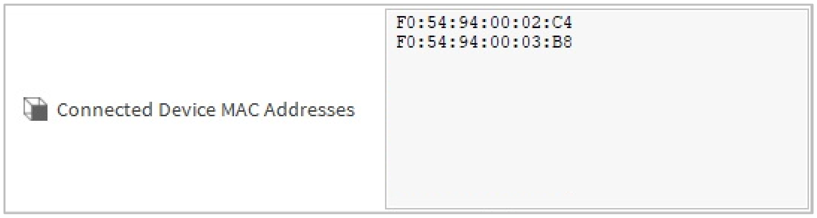



If a device connected to multiple other devices using a daisy chain connection is connected to the controller’s port, then the Connected Device MAC address property displays all the MAC address of all the devices in the daisy chain when all these devices are communicating with the CIPer 30 controller.

Because the MAC address is the list has aging time (default is 300s), if a device stops communication with the CIPer 30 controller, its MAC address will disappear from the Connected Device MAC Address box after the aging time i.e. 300s ~ 360s.

Example: In the image below, the device F0:54:94:00:03:B8 is connected to multiple other devices using the daisy chain connection type. The Connected Device MAC Address displays, all the connected devices in the daisy chain connection type till they are communicating.

After the aging time (300s ~ 360s), if there is no communication between the devices and the CIPer 30 controller. The devices MAC address in the daisy chain connection will disappear from the Connected Device MAC Address list.




When a device connected to the controller's port using a daisy chain connection type, the following behavior is expected

– If the RSTP property is Enabled, the expected MAC address is not displayed in the Connected MAC Address box for a few seconds.
– If the RSTP property is Disabled, the MAC address is displayed in Connected MAC Address box as expected.

## Network Firewall Configuration

The Network Firewall configuration will show the status of the Network Firewall.

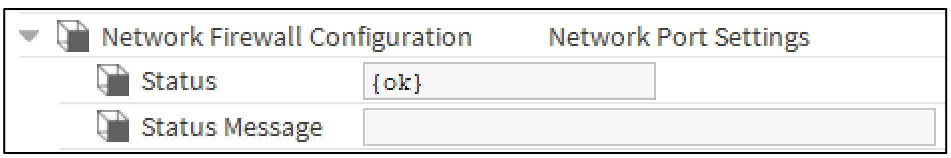



Figure 72: Network Firewall Configuration Offline mode Properties

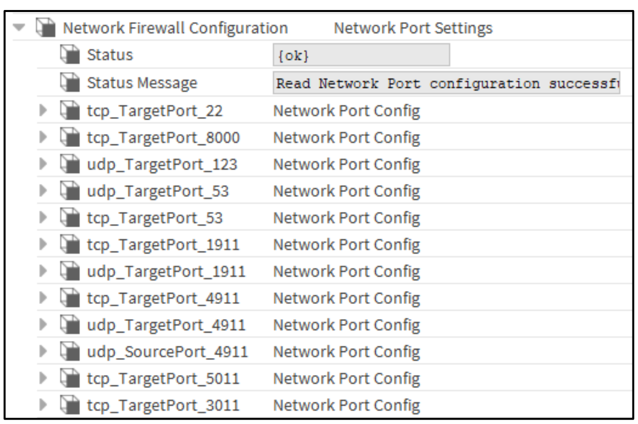



Figure 73: Network Firewall Configuration Online Mode Properties

## Port Configuration

For CIPer30 controller, outgoing traffic has no restriction, but all incoming IP traffics are restricted except that allowed by the access rule. To define the access rule user need to change the Port Number, Port access rule, protocol via Port configuration option. To edit the port configuration, user need to drag NetworkPortConfig from Port Configuration folder of IPC Programming Tool palette and drop the same under the Network Firewall Configuration of the Local device

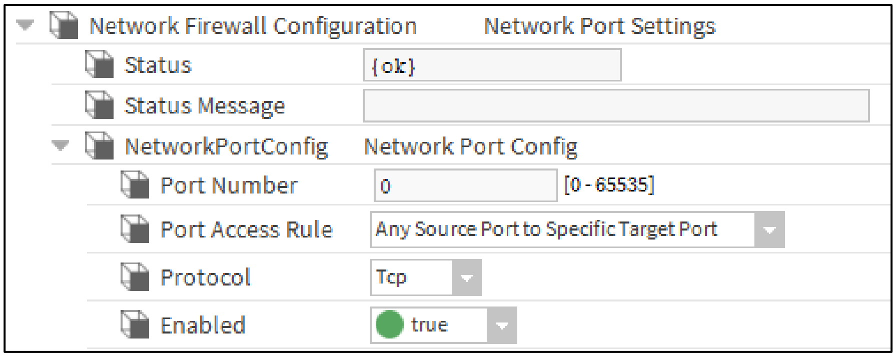



Figure 74: Port Configuration Properties

Different services use different port of TCP/UDP. And for some incoming service, source port is fixed, while for others destination port is fixed. Port access rule include two choices **'Any Source Port to Specific Target port'** and '**Specific Source Port to Any Target port'**. This configuration depends on the service.

E.g. For SSH service, user start a SSH connection from PC using the protocol TCP and destination port 22; so the rule is to allow Any Source Port to Specific Target port. CIPer 30 controller will listen on TCP port 22 and accept the incoming connection.

While for DNS service, CIPer30 want to receive traffic from remote DNS server which has fixed source port udp/tcp 53. Now the rule is to allow Specific Source Port to Any Target port. And DNS service uses both TCP and UDP protocol, so two rules for TCP and UDP need to be added separately.

For incoming traffic, there is default rule for each CIPer 30 controller. By default, ICMP/ IPv6/ SSH/ NTP/ DNS/ DHCP/ Niagara platform/ Fox/ Foxs/ BACnet service related ports are allowed. Controller can receive these traffics from other devices. When you want to accept new service e.g. TFTP, then user need to manually add a new rule to accept the TFTP related port: UDP source port 69. You can drag Port Configuration to Network Firewall Configuration and set port number->69, rule->Specific Source Port to Any Target port, protocol->UDP and enable -> true. After that, right click the Network Firewall Configuration and apply port setting.

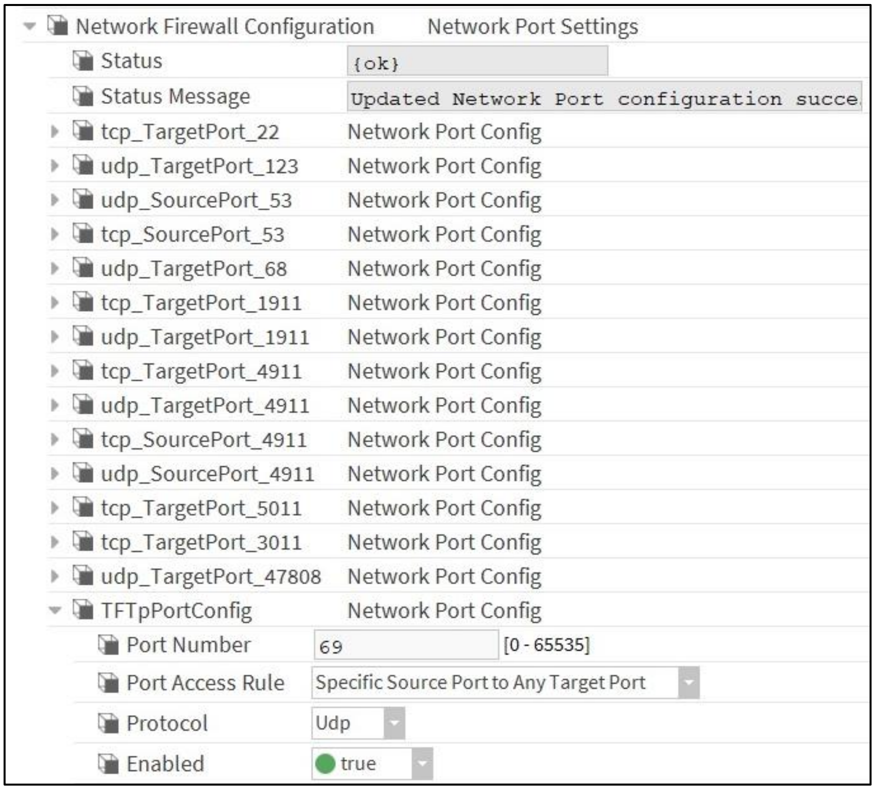



Figure 75: Port Configuration Properties

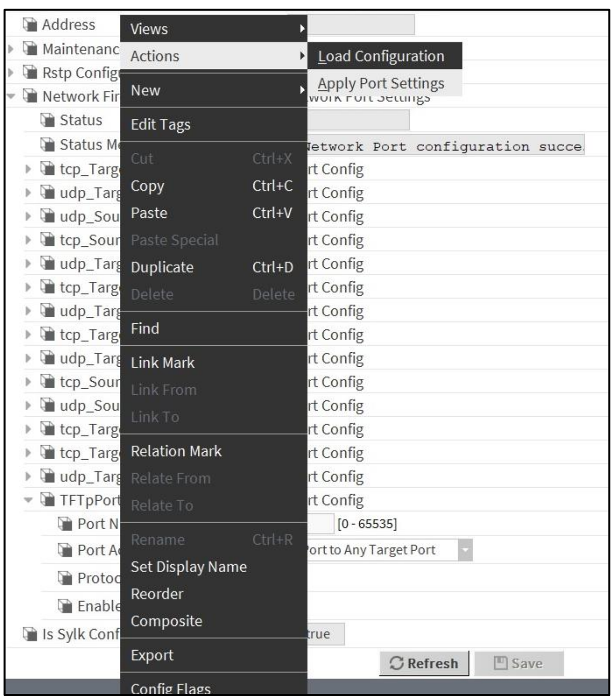



Figure 76: Applying Port Configuration Properties

If user modify the default port for some service e.g. BACnet 47808 to other ports, then user also need to modify the rule accordingly to make sure service can pass.

| | |
|---|---|
| | Note: |

1. By default, BACnet port is disabled for security reason, to enable the BACnet port you need to man-ually enable BACnet UDP port and enter port number in hexadecimal format (example: For Port BAC 0 hex-adecimal port number is 47808)
2. You can add multiple Port Configuration under one local device to make the specific device as Source and Target both
3. By default, the CIPer 30 controller have Secure Fox connection. If you want to connect any device with standard/ non-secured Fox connection, then you need to disable the secured Fox connection from Fox services and the need to add access rules to the Network Firewall Configuration.
4. You cannot edit or delete configured port directly from Network Firewall Configuration. If you accidently delete or try to edit any configured port, system will restrict the operation, generating error message.

## Sylk Configuration Download Status and Total Power consumption

After adding the sylk device user can check the power consumption of the sylk device in the property sheet of the Local Device. Also, user can check if the sylk configuration is downloading status.

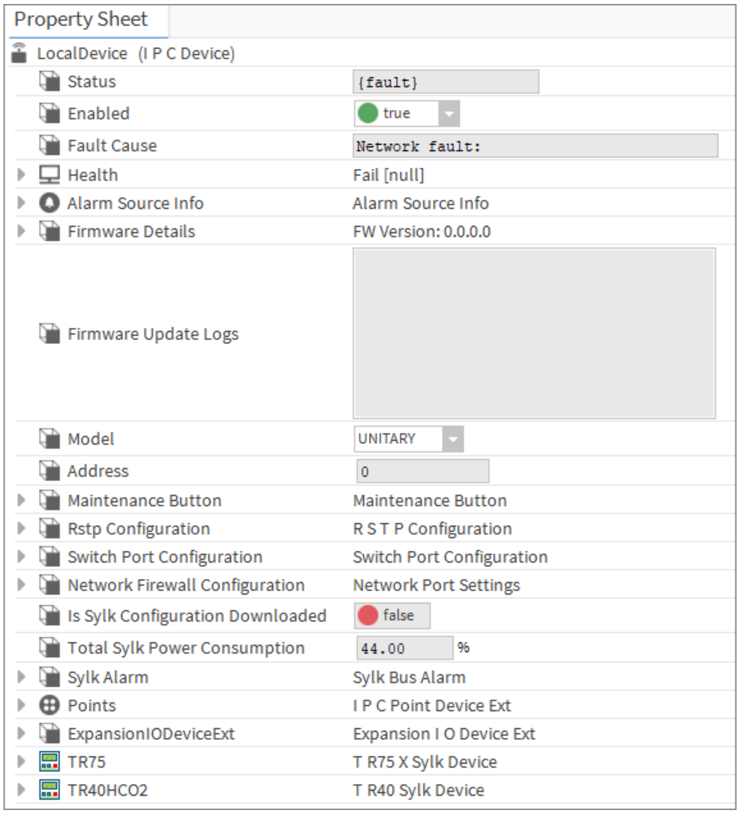



Figure 77: Total Sylk Power Consumption data checking

## Sylk Alarm

This will show the status of the Sylk bus whether any discrepancy are there in sylk device configuration .

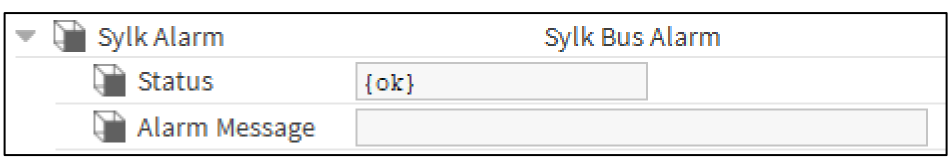



Figure 78: Sylk Alarm Properties

## Points

The Points node contains the Sequenced Control Program and Event Control Program. This option enables you to discover and configure the physical points in the CIPer Model 30 application. To add the devices to the data-base, you can use the Points folder.

### Sequenced Control Program Container

In this wire sheet, the value of the output changes continuously based on the execution of the function block in ascending order. The execution order of the function block can be changed anytime. See Order of Execution section.

| Note: |
| --- |
| You must use ipcProgrammingTool from palette to write program in this wire sheet. To know more, see SEQUENTIAL PROGRAMMING. |

### Event Control Program Container

In this wire sheet, the value of the output changes only if there is a change in the input values. The function block gets executed based on the order of the changes in the input values.

| Note: |
| --- |
| You must use Kit Control from palette to write program in this wire sheet. To know more, see EVENT-BASED PROGRAMMING. |

## Views

You can visualize the components in the system through views. Along with the Niagara-provided views, the CIPer Model 30 programming model provides the following views:

- I P C Point Manager View: This view provides details information about the Physical Points associated with Local Device (CIPer Model 30 controller) and expansion IO Modules.
- Wire Sheet View: You can write the logic in the wire sheet by using the CIPer Model 30 programming model.
- Sylk Parameter Summary View: This view provides detailed information about the Sylk parameters associated with the Sylk devices.
- Commission Sylk Devices View: This view provides information about the errors in the configuration of Sylk parameters.

### I P C Point Manager View

This view provides details information about the Physical Points associated with Local Device (CIPer Model 30 controller) and expansion IO Modules. It displays details such as the associated Device Name, Device address, Pin Type, Terminal Number, Terminal Name & Description for Physical Points. You must convert the Physical Points to Niagara Points before using it in Control Program Logic.

To access the IPC Point Manager View:

1. In the Nav tree, browse to Station > Config > Drivers > IPCNetwork > LocalDevice > Points > right click Views > IPC Point Manager.

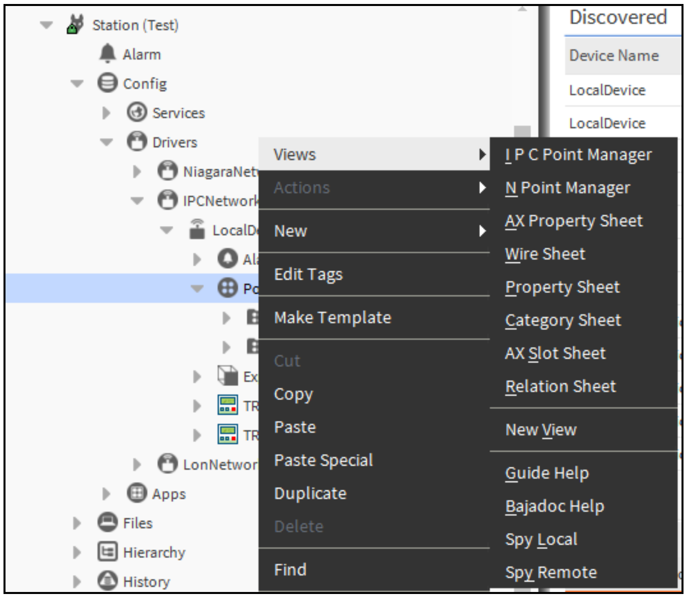
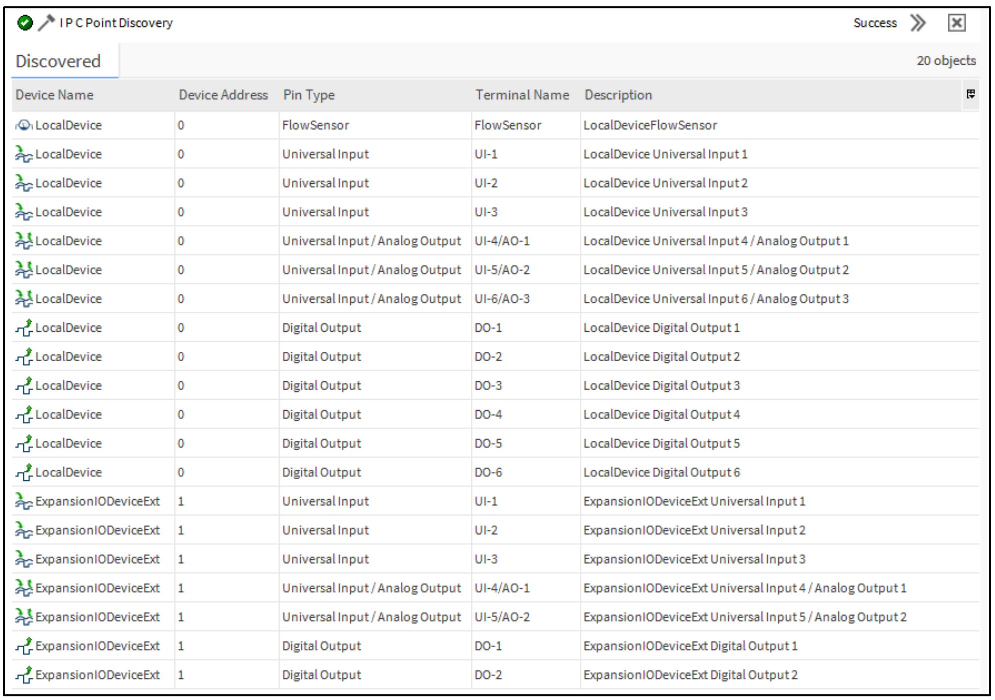



Figure 79: I P C Point Manager Option in Views Menu

2. The Discovered and Database tabs are displayed as shown in the following figure.




| Device Name | Device Address | Pin Type | Terminal Name | Description |
|---|---|---|---|---|
| LocalDevice | 0 | FlowSensor | FlowSensor | LocalDeviceFlowSensor |
| LocalDevice | 0 | Universal Input | UI-1 | LocalDevice Universal Input 1 |
| LocalDevice | 0 | Universal Input | UI-2 | LocalDevice Universal Input 2 |
| LocalDevice | 0 | Universal Input | UI-3 | LocalDevice Universal Input 3 |
| LocalDevice | 0 | Universal Input / Analog Output | UI-4/AO-1 | LocalDevice Universal Input 4 / Analog Output 1 |
| LocalDevice | 0 | Universal Input / Analog Output | UI-5/AO-2 | LocalDevice Universal Input 5 / Analog Output 2 |
| LocalDevice | 0 | Universal Input / Analog Output | UI-6/AO-3 | LocalDevice Universal Input 6 / Analog Output 3 |
| LocalDevice | 0 | Digital Output | DO-1 | LocalDevice Digital Output 1 |
| LocalDevice | 0 | Digital Output | DO-2 | LocalDevice Digital Output 2 |
| LocalDevice | 0 | Digital Output | DO-3 | LocalDevice Digital Output 3 |
| LocalDevice | 0 | Digital Output | DO-4 | LocalDevice Digital Output 4 |
| LocalDevice | 0 | Digital Output | DO-5 | LocalDevice Digital Output 5 |
| LocalDevice | 0 | Digital Output | DO-6 | LocalDevice Digital Output 6 |
| ExpansionIODeviceExt | 1 | Universal Input | UI-1 | ExpansionIODeviceExt Universal Input 1 |
| ExpansionIODeviceExt | 1 | Universal Input | UI-2 | ExpansionIODeviceExt Universal Input 2 |
| ExpansionIODeviceExt | 1 | Universal Input | UI-3 | ExpansionIODeviceExt Universal Input 3 |
| ExpansionIODeviceExt | 1 | Universal Input / Analog Output | UI-4/AO-1 | ExpansionIODeviceExt Universal Input 4 / Analog Output 1 |
| ExpansionIODeviceExt | 1 | Universal Input / Analog Output | UI-5/AO-2 | ExpansionIODeviceExt Universal Input 5 / Analog Output 2 |
| ExpansionIODeviceExt | 1 | Digital Output | DO-1 | ExpansionIODeviceExt Digital Output 1 |
| ExpansionIODeviceExt | 1 | Digital Output | DO-2 | ExpansionIODeviceExt Digital Output 2 |

Figure 80: I P C Point Manager View

- New Folder: To add EventControlProgram folder
- Edit: To modify the Niagara point added in the database
- Discover: To discover the physical points present in the local device (CIPer Model 30 controller) and in associated expansion I/O devices
- Add: To add the selected physical point to database
- Tagit: To tag the Niagara Points as per requirement for quick searching.

## Wire Sheet View

You can write the logic in the wire sheet by using the CIPer programming model.

Following are the steps to do this:

1. Expand IPCNetwork in the Nav tree.
2. Navigate to LocalDevice > Points.
3. Select SequencedControlProgram or EventControlProgram as per requirement.
4. Double-click the folder and the wire sheet is displayed.

   Or

   Right-click the folder > Views > Wire Sheet. The wire sheet is displayed on the right-side of the screen.

## Sylk Parameter Summary View

This view provides detailed information about the Sylk parameters associated with the Sylk devices. It displays the details like associated Sylk Device Name, Category, Parameter Name, Slot Path, and the Type of the parameter.

To access the Sylk Parameter Summary View:

- In the Nav tree, navigate to Station > Config > Drivers > IPCNetwork > LocalDevice.
- Right-click LocalDevice, click Views, and then select Sylk Parameter Summary.

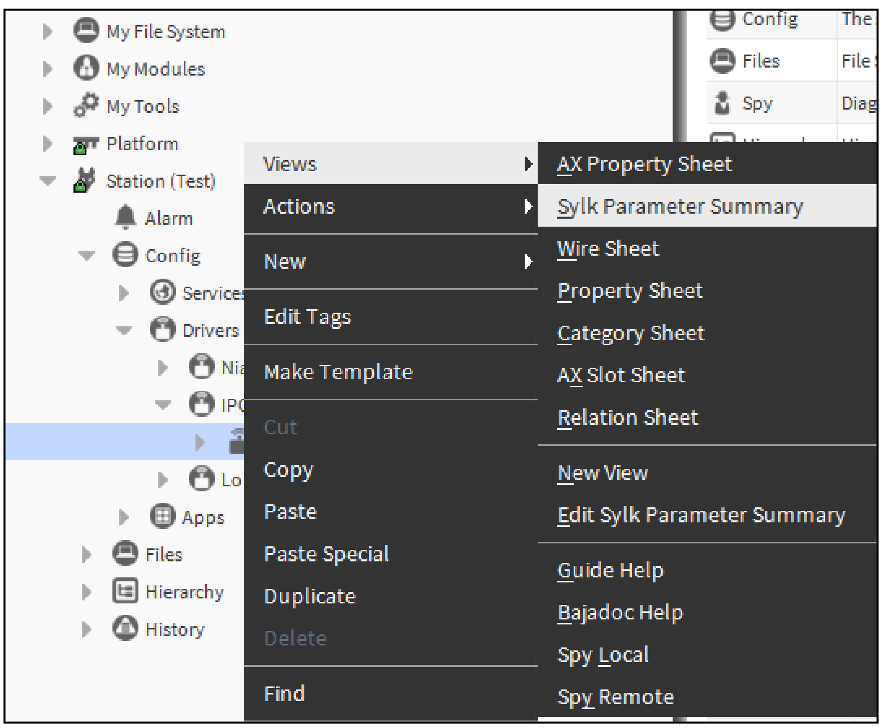



Figure 81: Sylk Parameter Summary Option Views Menu

The following details are displayed in the right-side of the screen.

Station (Test) | LocalDevice | Sequenced Control Program

| Sylk Device Name | Category | Param Name | Slot Path | Type |
|---|---|---|---|---|
| TR42HCO2SBusWallModule | Temp Sensor | ROOMTEMP | slot:/Drivers/IPCNetwork/localDevice/points/SequencedControlProgram/ROOMTEMP | honeywellSylkDevice:TemperatureParam |
| TR42HCO2SBusWallModule | Humidity Sensor | HUMIDITY | slot:/Drivers/IPCNetwork/localDevice/points/SequencedControlProgram/HUMIDITY | honeywellSylkDevice:HumidityParam |
| TR42HCO2SBusWallModule | CO2 Sensor | CO2 | slot:/Drivers/IPCNetwork/localDevice/points/SequencedControlProgram/CO2 | honeywellSylkDevice:CO2Param |
| TR75SBusWallModule | | OccupancyStatus | slot:/Drivers/IPCNetwork/localDevice/points/SequencedControlProgram/OccupancyStatus | honeywellSylkDevice:OccupancyStatus |
| TR42HCO2SBusWallModule | Delays | BypassTime | slot:/Drivers/IPCNetwork/localDevice/points/SequencedControlProgram/BypassTime | honeywellSylkDevice:BypassTimeParam |
| TR42HCO2SBusWallModule | Setpoints | NetworkSetpoint | slot:/Drivers/IPCNetwork/localDevice/points/SequencedControlProgram/NetworkSetpoint | honeywellSylkDevice:NetworkSetpointParam |
| TR75SBusWallModule | Temp Sensor | ROOMTEMP1 | slot:/Drivers/IPCNetwork/localDevice/points/SequencedControlProgram/ROOMTEMP1 | honeywellSylkDevice:TemperatureParam |
| TR75SBusWallModule | Setpoints | NetworkSetpoint1 | slot:/Drivers/IPCNetwork/localDevice/points/SequencedControlProgram/NetworkSetpoint1 | honeywellSylkDevice:NetworkSetpointParam |

- Sylk Device Name: The associated Sylk device
- Category: The category defined while configuring Sylk parameters. By default, it shows Category as category.
- Parameter Name: The name configured while adding the parameter onto the wire sheet.
- Slot Path: The path of the Sylk Parameter
- Type: The type of the Sylk Parameter

## Commission Sylk Devices View

This view provides information about the errors in the configuration of Sylk parameters. It displays the error message including the Sylk Parameter Name, Parameter ORD number, and the Type of the error.

To access the Commission Sylk Devices View:

1. In the Nav tree, navigate to Station > Config > Drivers > IPCNetwork > LocalDevice.
2. Right-click LocalDevice, click Actions, and then select Commission Sylk Devices.

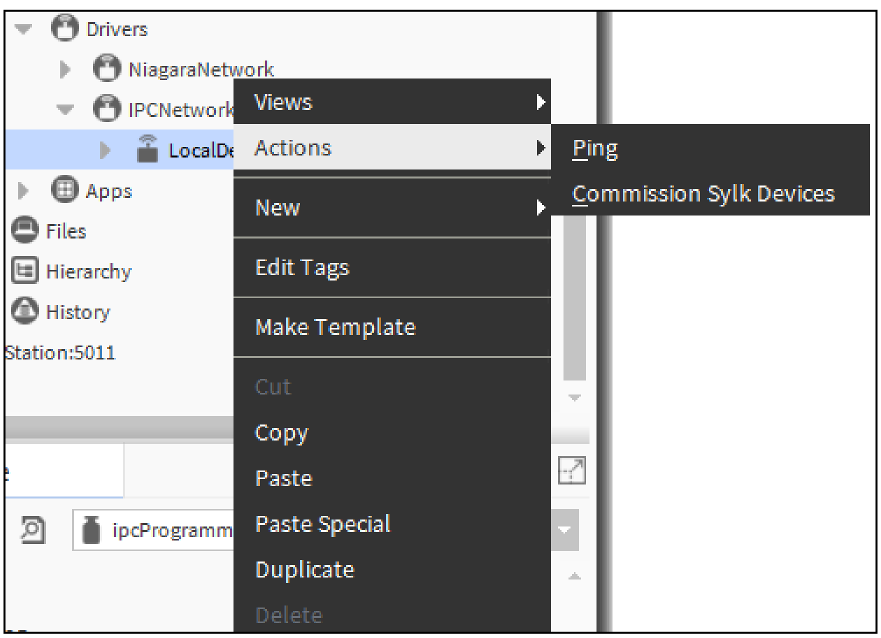



Figure 82: Commission Sylk Devices Option in Actions Menu

The status notification of the commission Sylk devices action is displayed in the lower right side corner of the screen.


Figure 83: Notification-Commission Sylk Devices

3. Navigate to Window > Side Bars and select Jobs.

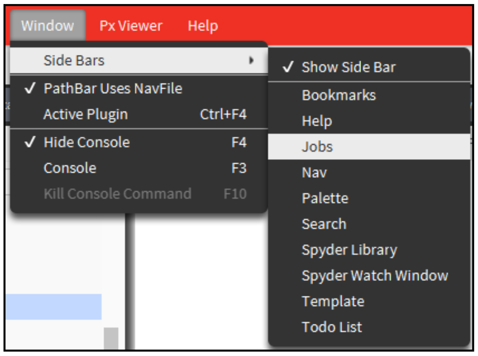



Figure 84: Jobs Option in Window > Side Bars Menu

4. Navigate to Jobs Palette and click  next to the required job.

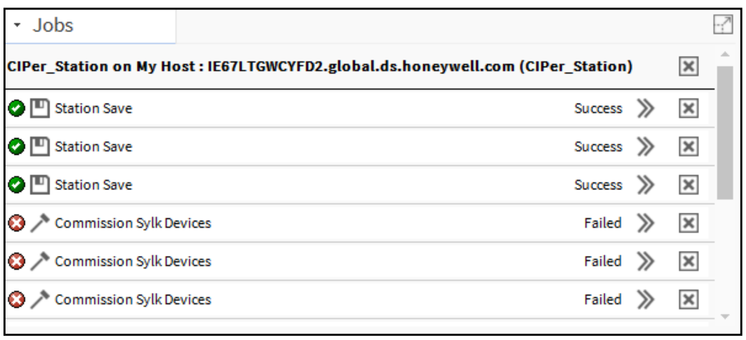



Figure 85: Jobs Palette

The Job Log window is displayed. The job log contains details like status of the job, timestamp, and message.

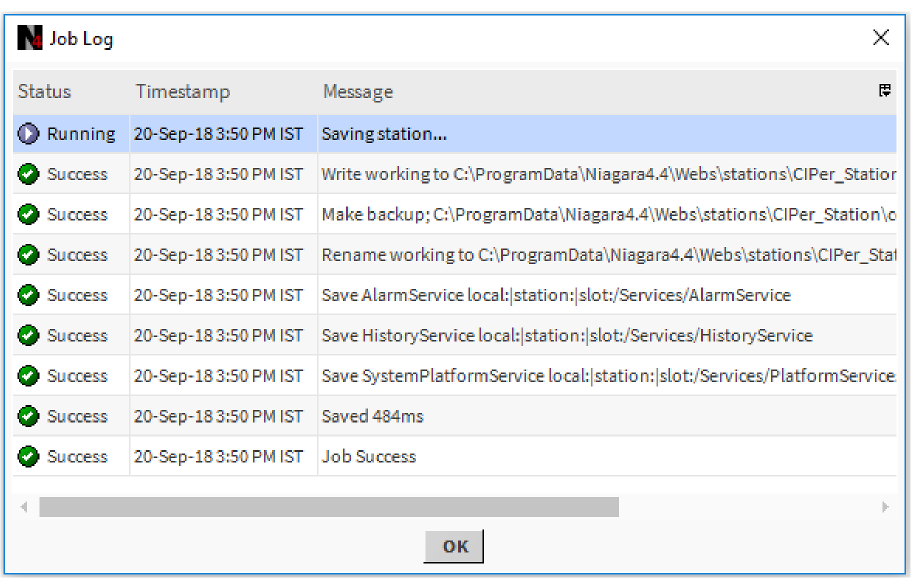


Figure 86: Job Log

The display provides the data regarding the Sylk Parameter Name, Parameter ORD number & the Type of the error. You can use the ORD number to locate the Point.

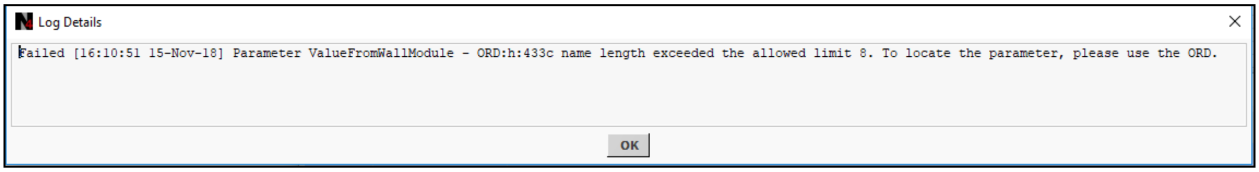



Figure 87: Log Details Window

**Note:**

The job log generated after commissioning the Sylk device, displays all the validation errors, but the device level fault cause displays the latest fault cause only.

## USB Support

The CIPer Model 30 controller has a single Type-A USB socket that supports RS-485 communication via RS-485 converter. The RS-485 communication is limited to three RS-485 devices.

The RS-485 allows multiple devices to communicate at half-duplex on a single pair of wires, plus a ground wire, with distances up to 1200 meters (4000 feet) and supports MODBUS and BACnet MSTP protocols. The supported chipset used in USB to RS485 converter are FTDI (FT232 series) or prolific chipset (PL2303).

- The MSTP channel allows up to three generic BACnet controllers to be configured.
- ModbusAsync serial channel allows up to three MODBUS controllers to be configured.

All the baud rates up to 115200 bps are supported.

- Baud_9600
- Baud_19200
- Baud_38400
- Baud_57600
- Baud_115200

**Basic Setup Steps**

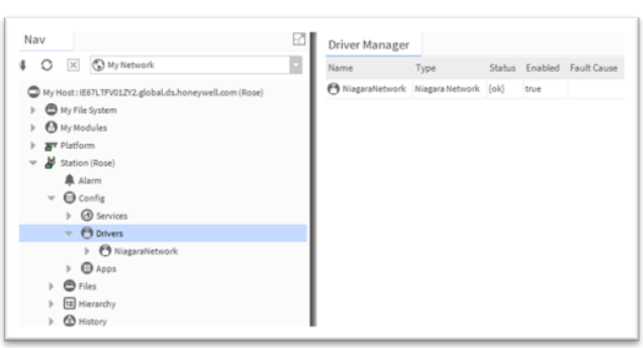
1. Before powering up the CIPer Model 30 controller, connect the 'USB to RS-485' converter to the type-A USB socket.
2. After powering-up the CIPer Model 30 controller, configure the protocol (Modbus or BACnet) and baud rate using connected Niagara workbench. Refer Configuring BACnet Network as an MS/TP Router and Configuring ModbusAsyncNetwork.
3. Now, connect the Modbus/BACnet controllers to the RS485 port of CIPer Model 30 controller.

> **NOTE**
>
> Every time the 'USB to RS-485' connector is disconnected and reconnected, or changed to a different converter, the CIPer Model 30 controller must be power-cycled.

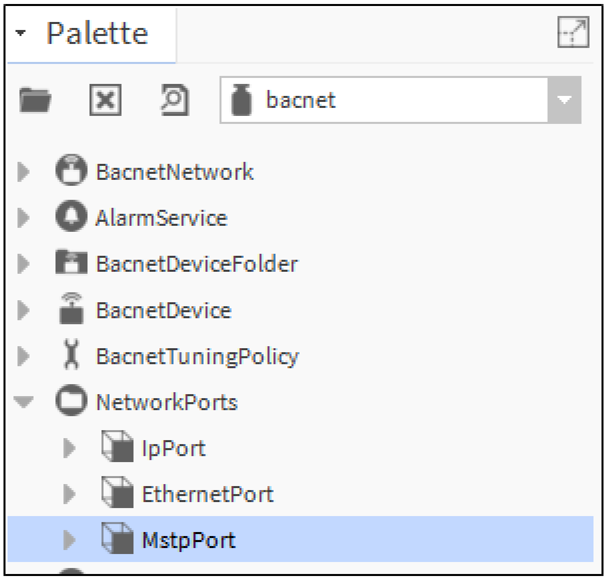
### Configuring BACnet Network as an MS/TP Router

To configure the BACnet network as an MS/TP Router:

1. Connect to the controller's station using Niagara Workbench.
2. Open the bacnet **Palette.**
3. In the NAV tree open the **Station > Config** folder**.**
4. Double click **Drivers**. The Driver Manager is displayed.

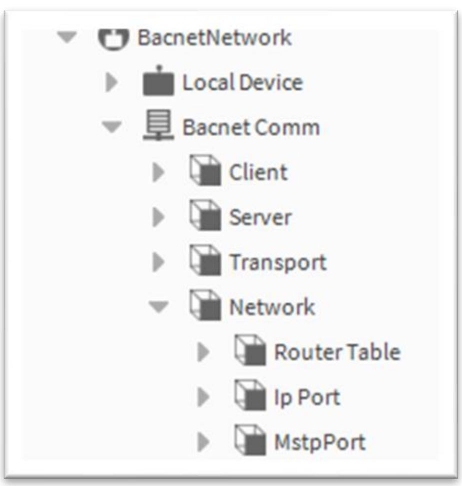


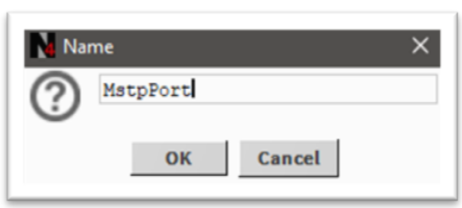
5. In the bacnet palette, open the **NetworkPorts** folder.




> **NOTE**
> Make sure to add **BacnetNetwork** under **Drivers** in the controller's station.

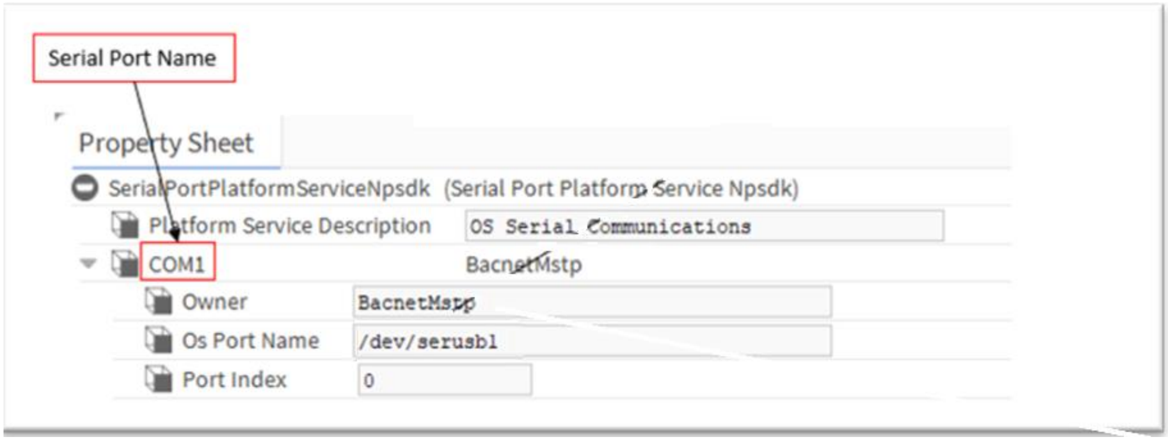
6. Select **MstpPort**, drag and drop the **MstpPort** to **Drivers > BacnetNetwork > Bacnet Comm >Network** in the **NAV** tree.




7. In the NAV tree double-click on the **MstpPort.** The MstpPort (Network Port) property sheet is displayed.
8. In the **Network Number** field, enter the network number of the BACnet network segment to which the controller is being connected.
9. In the **Port Name** field, enter the Serial Port Name of the device connected to the controller port (ideally it should be COM1).
10. To verify the controller port name, check the platSerialNPSDK property sheet.

    10.1 Navigate to **Station> Config > Service> Platform service** and double-click **platSerialNPSDK** to display its property sheet. Check the Serial Port Name (COM1)..




> **NOTE**
>
> - If there is no USB device connected to the controller, COM1 is not recognized.
> - If the COM port does not appear under the npsdk service, it means the CIPer Model 30 does not recognize the connected 'USB to RS485' converter. Ensure the converter is connected and is using one of the supported chipsets (FTDI or Prolific).

11. In the **Baud Rate** field, set the baud rate used for BACnet MS/TP communication.
12. In the **Enabled** field select **true.**

Configure the other parameters as required.

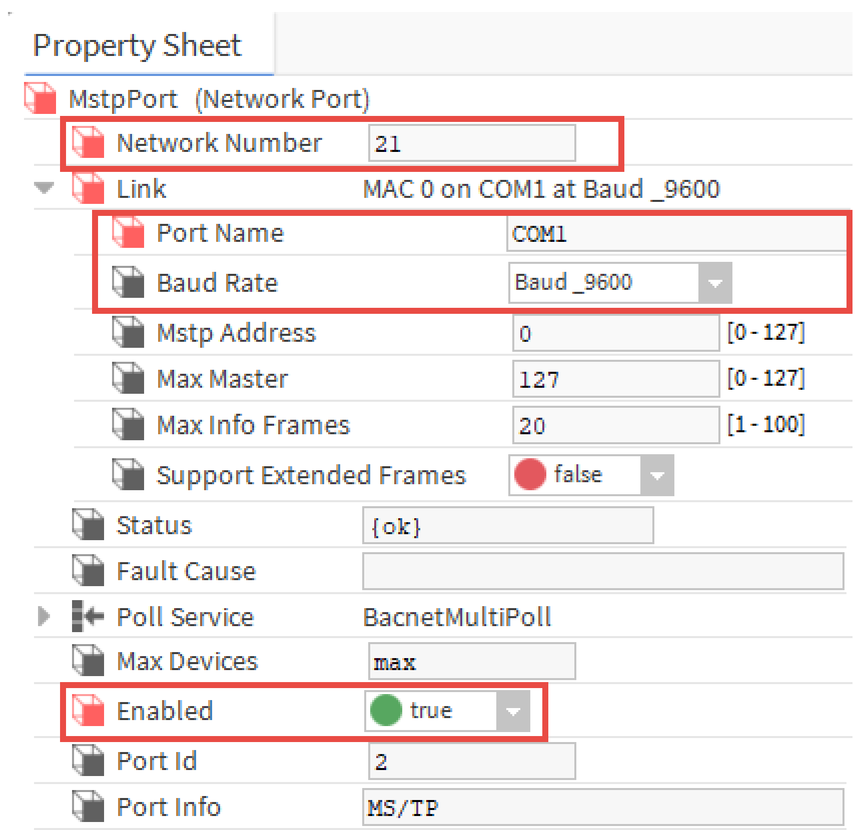


**Table 10: BACnet Network Properties**

| Property | Description |
|---|---|
| Network Number | The BACnet network number of the network segment to which you are connecting.<br>• If connecting to an existing BACnet installation, use the network number for that installation.<br>• For a new BACnet installation, set to the required value. |
| Port Name | The port should be the same as the name of the connected USB to RS485 converter. Typically, the name is COM1. |
| Baud Rate | The Baud Rate of device on the BACnet MSTP bus. The baud rate of each controller is automatically set to the baud rate of the BACnet IP – MS/TP Router (this device) when the CIPer Model 30 controller is powered up and connected to the BACnet MS/TP bus.<br>**NOTE:** If the baud rate of the BACnet MSTP bus of a running system is changed, then you need disable and re-enabled BACnet network of all the CIPer 30 Controllers connected to the bus.<br>For other controllers (non CIPer 30 controllers) respective manual has to bereferred. |
| Mstp Address | The address of the controller on the Mstp bus. Range = 0 to 127 (default value is 0)<br>Each BACnet controller on the MS/TP network segment must have a unique MAC address (Mstp Address). Leave the Mstp Address at 0 (the default) and verify that no other MS/TP controller has a same address. |
| Max Master | Set max master value to the highest known master device on the network, with possible room for expansion. |
| Max Info Frames | Controls how many messages are sent before passing the token and may be increased up to 50 to increase performance in some cases.<br>Range = 0 to 100. |
| Support Extended Frames | Enables/ disables the use of larger frames, which, if supported by other devices connected to the USB, may improve performance. |
| Status | Read only. Indicates whether the MstpPort configuration is enabled/disabled. |
| Fault Cause | Read only. Displays the details of the invalid configuration. |
| Poll Services | Allows network polling properties to be configured. |
| Max Devices | Read only. Indicates the maximum number of devices available on the network. |
| Enable | Enables/Disables the MstpPort configuration. |
| Port Id | Read only. The number of the Ethernet port being configured. |
| Port Info | Read only. Reports the type of port (Ethernet, MS/TP, etc.). |

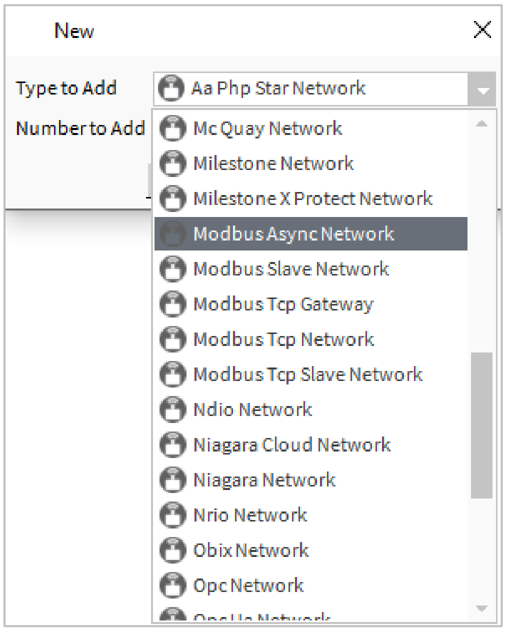
13. Click **Save**.

> **NOTE**
>
> Every time the 'USB to RS485' connector is disconnected and reconnected, or changed to a different converter, the CIPer Model 30 controller must be power-cycled.

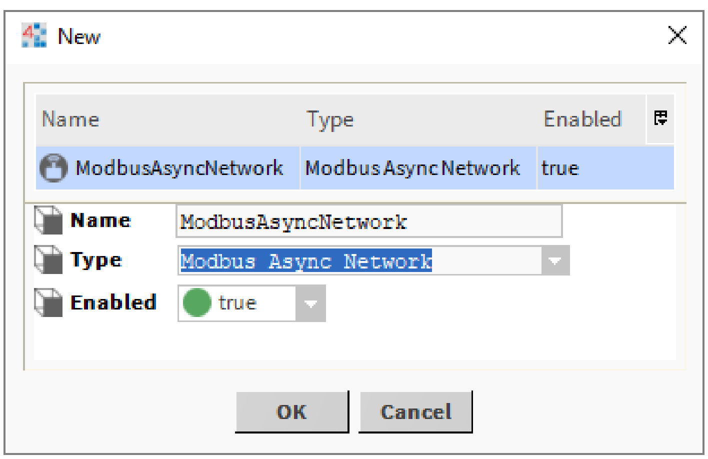
## Configuring ModbusAsyncNetwork

To configure the ModbusAsynchNetwork:

1. Connect to the controller's station using Niagara Workbench.
2. In the NAV tree open the **Station> Config folder.**
3. Double-click **Drivers**. The Driver Manager is displayed.
4. Click **New** and select the **ModbusAsynch Network** from drop-down list.




5. Click **OK**. This opens ModbusAsynchNetwork dialogue box.




6. Click **OK.** This adds ModbusAsynchNetwork.
7. In the NAV tree open the **Station > Config > Drivers** folder.
8. Right-click on **ModbusAsyncNetwork**, select **View>AX property sheet.** The property sheet for the ModbusAsyncNetwork is displayed.

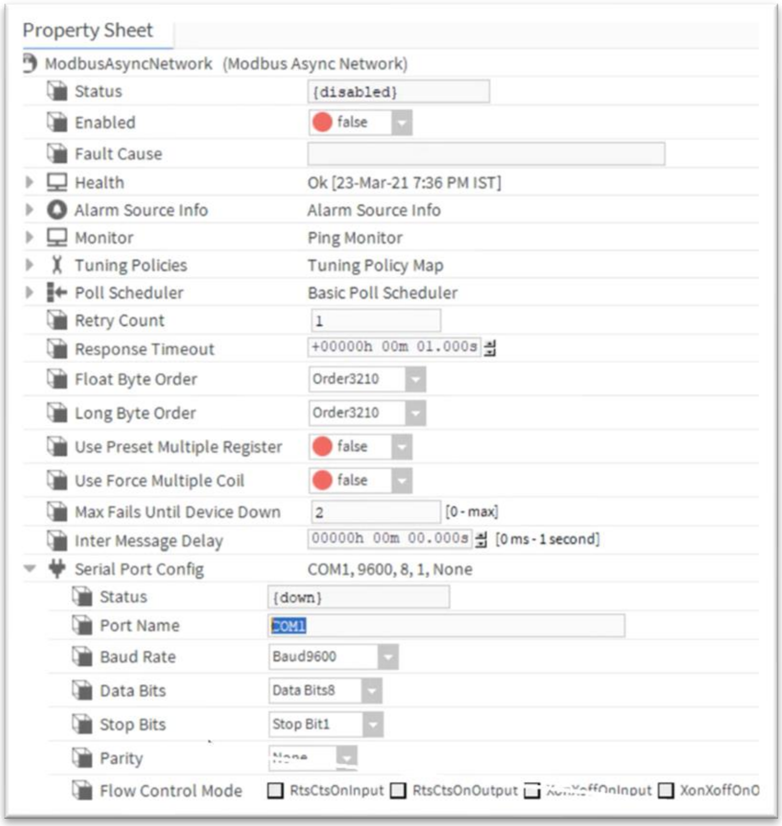


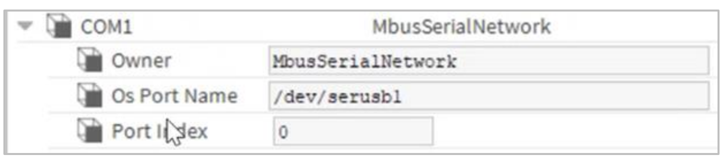

**Table 11: ModbusAsyncNetwork Properties**

| Property | Description |
|---|---|
| **Status** | Read only. Indicates the condition of the ModbusAsyncNetwork configuration at the last check.<br>**Ok** - The component is licensed and polling successfully.<br>**Down** - The last check was unsuccessful, e.g. because of an incorrect property setting, or loss of network connection.<br>**Disabled** - The network is disabled (the Enable property is set to false).<br>**Fault** - There is another problem. Refer to Fault Cause for more information. |
| **Port Name** | The port should be the same as the name of the connected USB to RS485 converter. Typically, the name is COM1. |
| **Baud Rate** | The communication speed in bits per second. (defaults = 9600) |
| **Data Bits** | Specifies how many bits form a character (byte). (default = 8) |
| **Stop Bits** | The number of stop bits. (default = 1) |
| **Parity** | The parity used to confirm that the system communicated each character success-fully. (default = Even) |

9. In the **Enabled** field, select **True.**
10. Expand the **Serial Port Config** property.

11. Configure the Serial port as required.
12. Click **Save**.

## Known Behaviors

| | |
|---|---|
| If the 'USB to RS485' converter is connected to the CIPer Model 30 controller's USB port while running station, then no COM port will be detected by the station. | To detect the COM port, power-cycle the CIPer Model 30 controller. |
| If the 'USB to RS485' converter is disconnected from the CIPer Model 30 controller's port while the station is running, then the serial communication will stop working. | To resume the serial communication.<br>1. Plug-in 'USB to RS485 converter or check the connections.<br>2. Power-cycle the CIPer Model 30 controller. |
| If the connected 'USB to RS485' converter FTDI Cable is replaced with a new 'USB to RS485' converter (e.g. from a different vendor). Then the name of the COM port might change.<br>If the station is running while this change is done, the serial communication will stop working.<br><br>This will break the COM port mapping and the Serial Communication will fail to start. | To resume the serial communication.<br>1. Verify the new COM port name<br>2. Resume the serial communication power-cycle the CIPer model 30 controllers. |

# I/O PROGRAMMING

The CIPer Model 30 controller provides various input and output points for the configuration. Some points are on-board points, which are available on the controller itself, some points are expansion points, which as per requirement can be configured and used.

- IPCNetwork: The IPCNetwork is the communication channel used for CIPer Model 30 controller. The IPC network corresponds directly to physical network of the device.
- ExpansionIODeviceExt: To add expansion IO devices

| Note: |
|---|
| • Expansion I/O devices are added to LocalDevice folder only.<br>• This device cannot be dragged onto the wire sheet. |

- Containers: This component provides folders, ApplicationFolder and EventControlProgram, to group or segregate the logic on the wire sheet for better understanding.

| Note: |
|---|
| • Application Folder is for Sequenced Control Program only. You must not add the same into the EventControlProgram folder.<br>• EventControlProgram folder is for Event Control Program only. You must not add the same into the SequencedControlProgram folder. |

- IOs: To configure the floating motor output
- Function Blocks: To create the programming logic

# IPC Network Component

The IPCNetwork block in the CIPer Model 30 programming model provides following features:

- Adding Network
- Viewing or Modifying Controller Summary Details

## Adding Network

To add an IPCNetwork:

1. Click Window > Side Bars > Palette to add the Palette, if it is not visible on the screen.
2. Drag and drop the IPCNetwork object from the ipcProgrammingTool palette to the Drivers folder in the Nav tree. The IPCNetwork folder appears in the Station > Config > Drivers folder in the tree. The Name window is displayed.

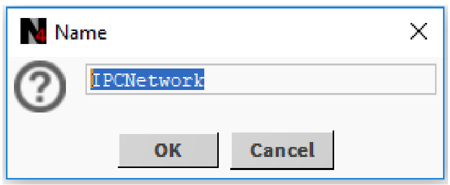



Figure 88: IPCNetwork Name Window

3. Enter a name for the IPCNetwork in the Name window or use the default name set by the application and click Ok. The IPCNetwork is added to the Drivers folder.

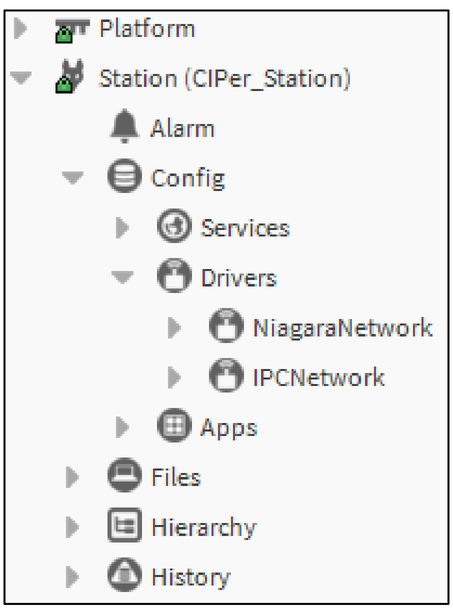



Figure 89: IPCNetwork Added to Drivers

**Note:**

While running I/O programming, set IO Heart Beat value to 0 sec. This will disable IO Heart Beat. Once I/O programming completed and when the controller restart, you can set IOHeartbeat value to 8sec.

## Viewing or Modifying IPCNetwork Components

To view or modify IPCNetwork components:

1. Double-click the IPCNetwork folder in the Nav tree. The Database tab is displayed with the device name and details.

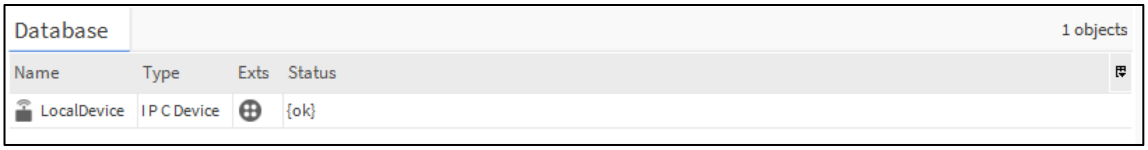



Figure 90: Local Device Name and Details in Database Tab

2. Double-click the LocalDevice folder to modify the device name and status. The Edit window is displayed.

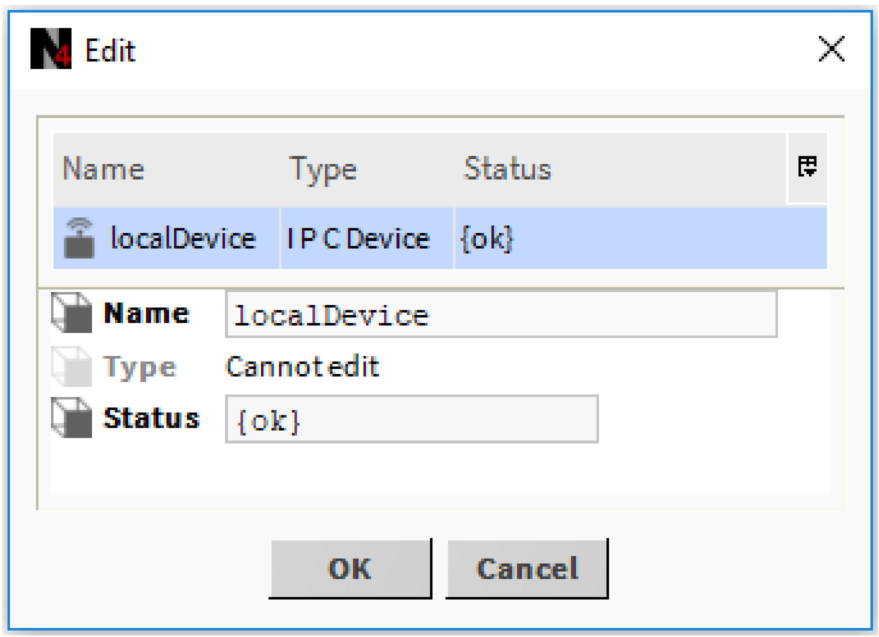



Figure 91: Edit Window

3. Modify the required details and click Ok to save the changes or click Cancel if you do not want to save the changes.
4. Double-click the LocalDevice folder in the Nav tree to configure the LocalDevice properties.

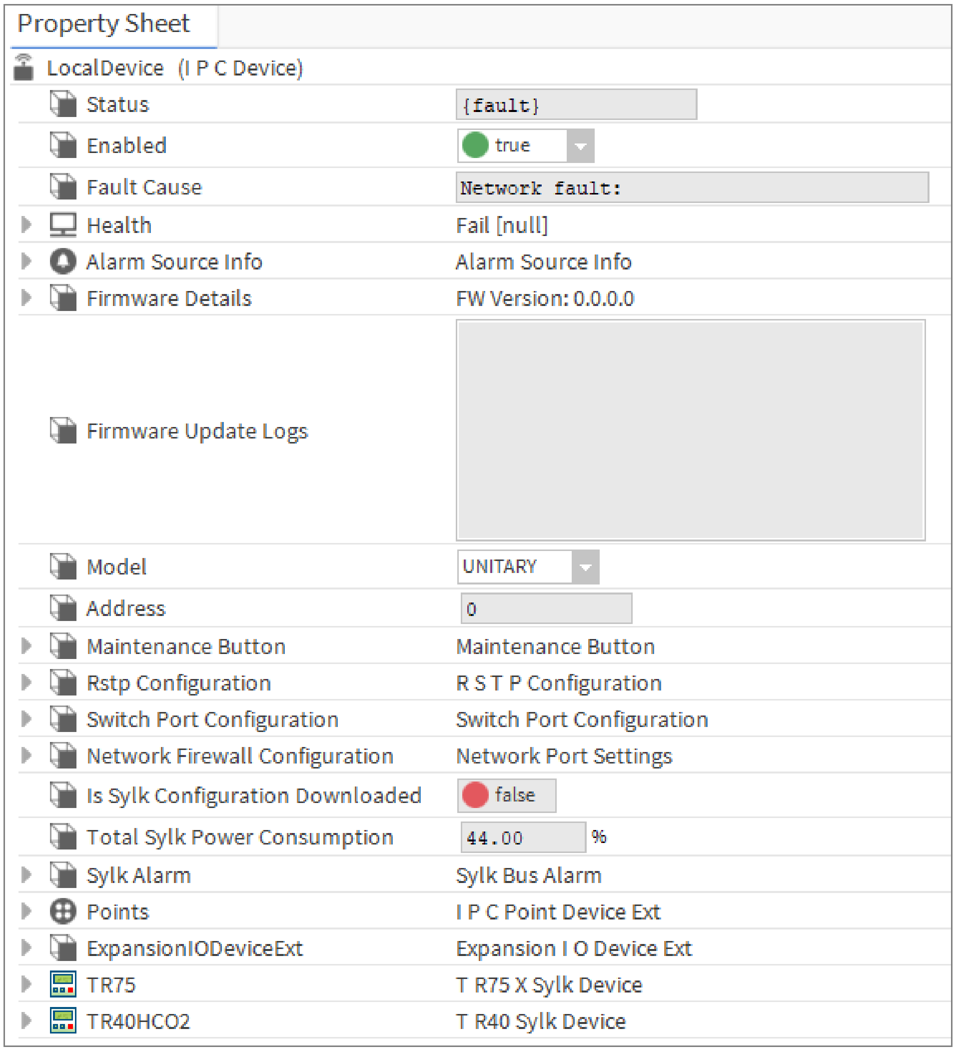



Figure 92: Property Sheet of LocalDevice

5. Modify the properties as required. See Overview of Major Components.
6. Click Save to save the changes.

   Or Click Refresh, and then No, if you do not want to save the changes.

| Note: |
| --- |
| You can configure UI/AO as UI and UO. A UI can be configured as AI and BI, whereas, UO can be configured as AO and BO. |

# Local Device

The I/O points which are present on the CIPer Model 30 controller are called on-board or local device I/O points. The on-board I/O points are present in the application by default. You need to add and configure those I/O points into the database as per requirement.


Figure 93: CIPer Model 30 Controller

Following are the details of on-board I/O points available in the CIPer Model 30 controller:

| Variant | Model Number | Description |
| --- | --- | --- |
| UNITARY | WEBC3036EPUBNH | CIPer IP Unitary Controller, BACnet, 150 Point/4-Device Niagara 4 License, SMA |
| VAV | WEBC3036EPVBNH | CIPer IP VAV Controller, BACnet, 150 Point/4-Device Niagara 4 License, SMA |

## Configuring LocalDevice

You can configure the on-board I/O point by changing the CIPer Model 30 controller model.

To change the CIPer Model 30 controller model:

1. Navigate to <Required Station> > Config > Drivers > IPCNetwork > LocalDevice.
2. Double-click LocalDevice. The application displays the Property Sheet of the local device.
3. Modify the details as required.
    - Select the Model from the Model drop-down menu.
    - Modify the address as required.

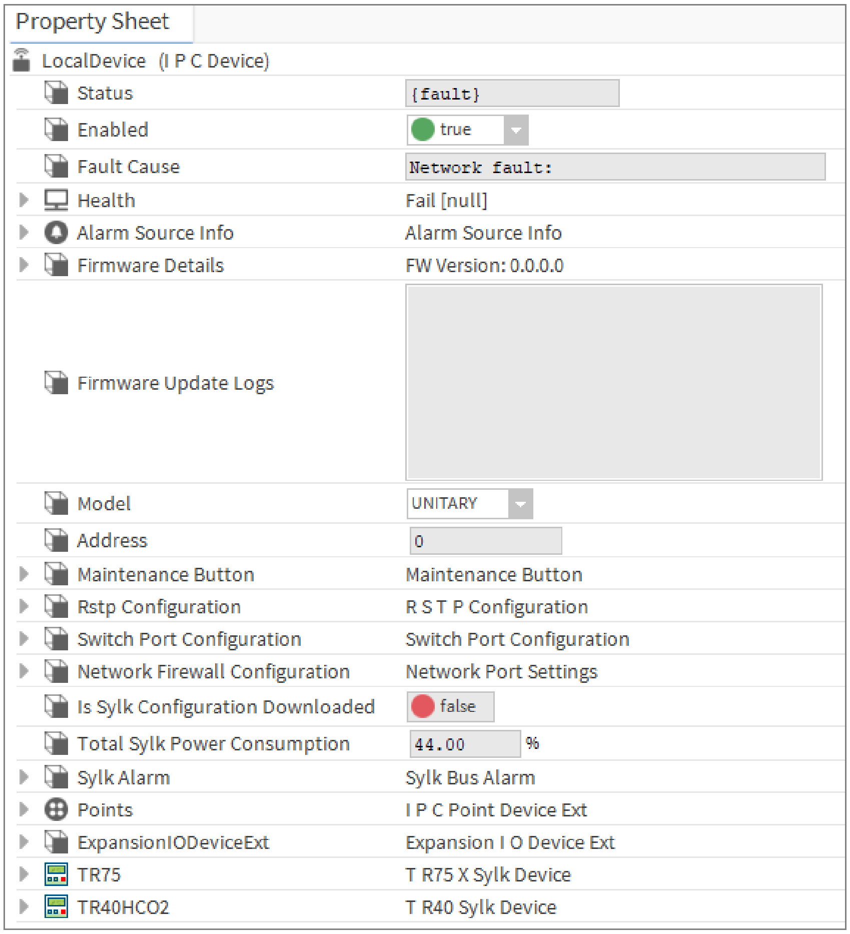



Figure 94: Property Sheet of Local Device

4. Click Save.

**Note:**

To assign an unassigned Sylk parameter to a Sylk device, right-click the Sylk device, select Actions, and click Attach Unassigned Params.

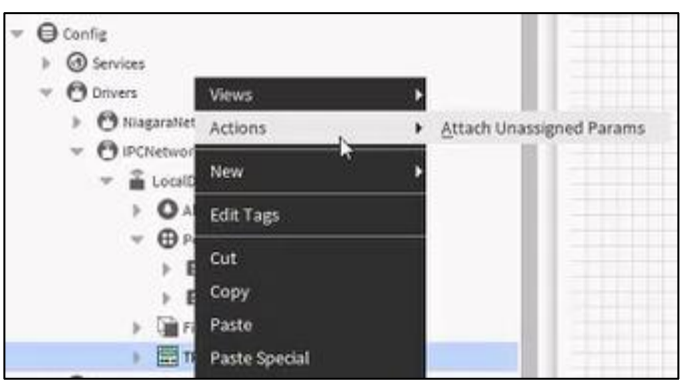



Figure 95: Attach Unassigned Params Action for Sylk Devices

For example, you have two Sylk devices, TR75 and TR75H, added to the LocalDevice node. If you drag and drop the ROOMTEMP parameter to the wire sheet, it shows None in front of the sylkDevice field. Now you can assign the ROOMTEMP parameter to the TR75 or TR75H Sylk device by either following the above-mentioned method or double-clicking the parameter on the wire sheet and assigning a Sylk device from the available list of applicable devices. Assume you have assigned the ROOMTEMP parameter to the TR75 Sylk device, and you delete the TR75 from the LocalDevice node, the ROOMTEMP parameter shows None in front of sylkDevice field.

Future Implementation (not available for current Beta release): once the user has connected the controller with the station then the slot for controller selection will be disabled and user will not be able to change the model type. If user wants to change the model type then user must disconnect the physical connection between controller and station, then change the mode type in offline mode.

## Expansion Devices

The I/O points which you can add externally as an expansion to existing on-board points are called as expansion I/O points. The CIPer Model 30 programming model allows you to connect extra input and output connections along with the on-board I/O connections. The ExpansionIODeviceExt function block enables you to add the extra input/output (I/O) connections to the device.


Figure 96: Expansion Device WEB-O3022H

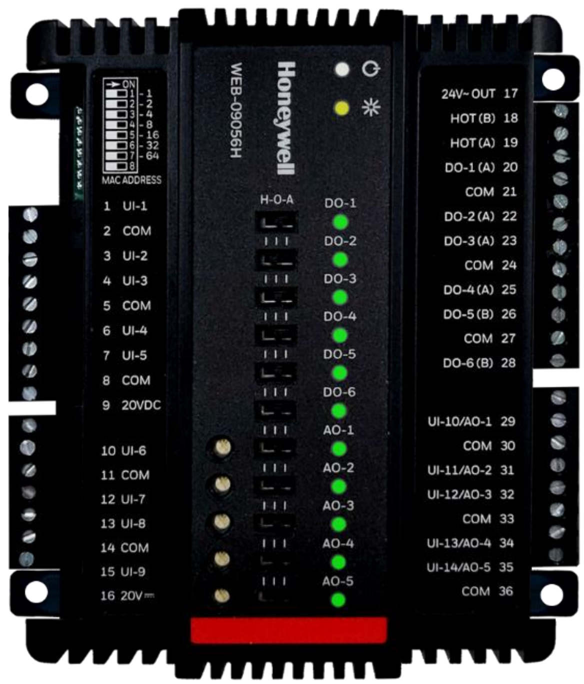



Figure 97: Expansion Device WEB-O9056H

### Adding Expansion I/O Device

You can add expansion I/O devices to the LocalDevice folder.

To add expansion I/O:

1. Navigate to the ExpansionIODeviceExt object in the ipcProgrammingTool palette.
2. Drag the ExpansionIODeviceExt object and drop it under LocalDevice.
3. Enter a name for the expansion I/O device and click Ok.

Following are the details of I/O points available in Expansion I/O modules:

- WEB-O3022H: 3 universal inputs, 2 universal I/O, 2 binary outputs, and HOA switches.

- WEB-O9056H: 9 universal inputs, 5 Universal I/O, 6 Binary Outputs, and HOA switches.

**Note:**

- The CIPer Model 30 controller supports maximum 15 expansion I/O modules.
- When you add more than one I/O device to the LocalDevice, the Address in the Property Sheet automatically increases by one.
- You cannot directly swap the expansion I/O address. Enter a different address to one device, and then swap the address.

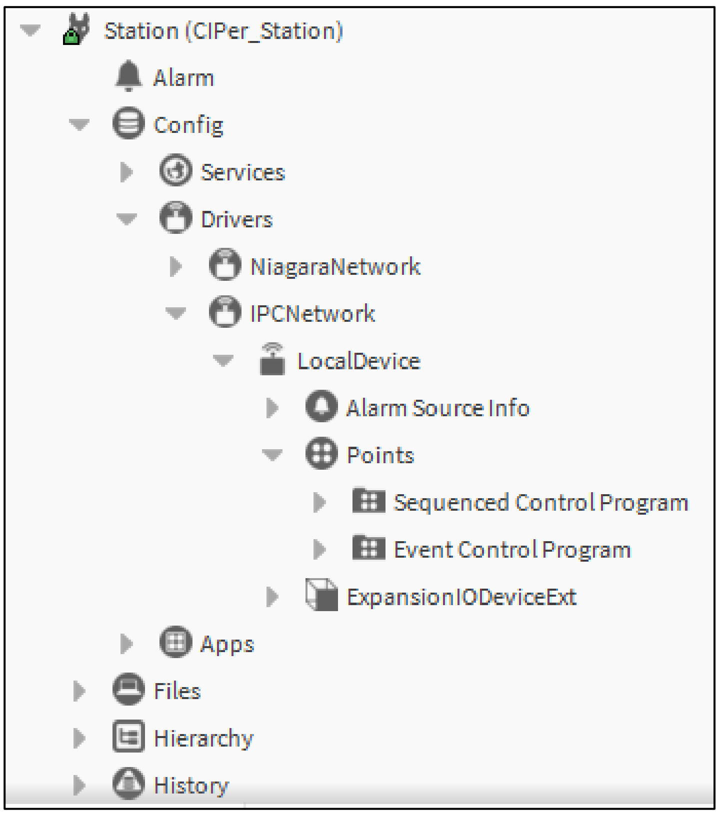



Figure 98: Expansion I/O Device Added to LocalDevice

## Configuring Expansion I/O Points

You can configure the properties of an expansion I/O device to customize the I/O devices.

To configure properties of expansion I/O:

1. Navigate to <Required Station> Config > Drivers > IPCNetwork > LocalDevice.
2. Double-click the required I/O device. The application displays the Property Sheet of the I/O device.
3. Modify the details as required.
    - Set the Enabled to true or false.
    - Select the Model from the Model drop-down menu.
    - Modify the address as required.

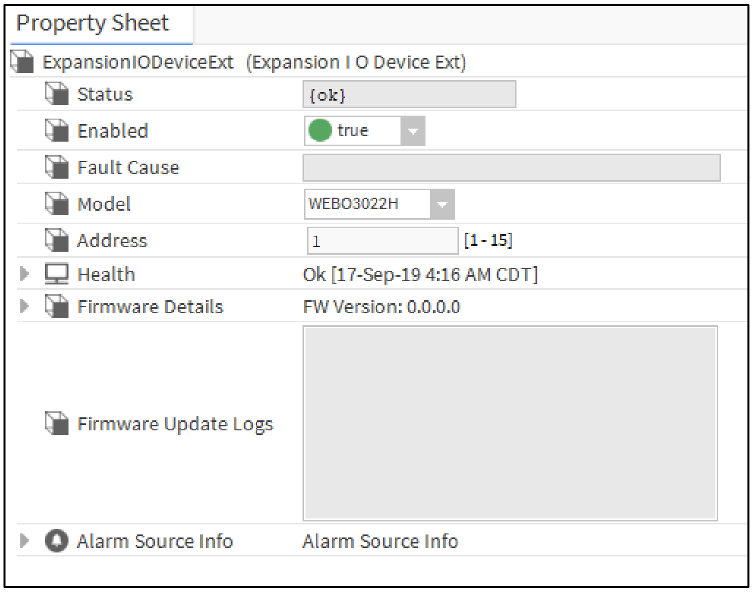


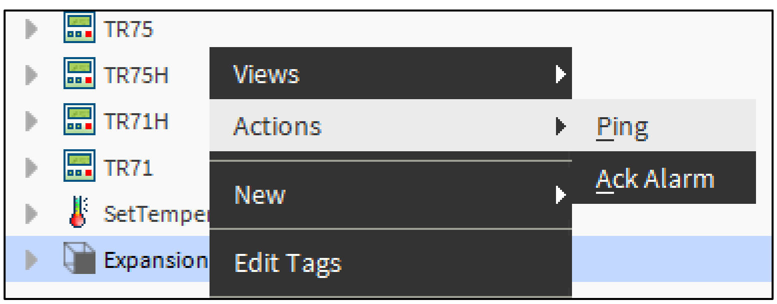

Figure 99: Property Sheet of Expansion I/O Device

4. Click Save.

## ExpIO Device Ping

ExpIO Ping allows user to discover externally connected I/O points connected to CIPer 30. Also, it verifies the configured of the Sylk device.

1. To run ExpIO Ping, Navigate to the ExpansionIODeviceExt in Nav window > **Actions** > **Ping**.




- If no device discovered **Status** displays **Down** and **Fault Cause** shows **No device found** and ExpIO Device stops read write operation.
- If unconfigured or defected device discovered **Status** displays **Down** and **Fault Cause** shows **Incompatible device found** and ExpIO Device stops read write operation.
- If the discovered device is compatible **Status** displays **Ok** and ExpIO Device performs read write operation.

# Using On-Board and Expansion I/Os

Following are the functions provided in I/O configuration of local and expansion devices:

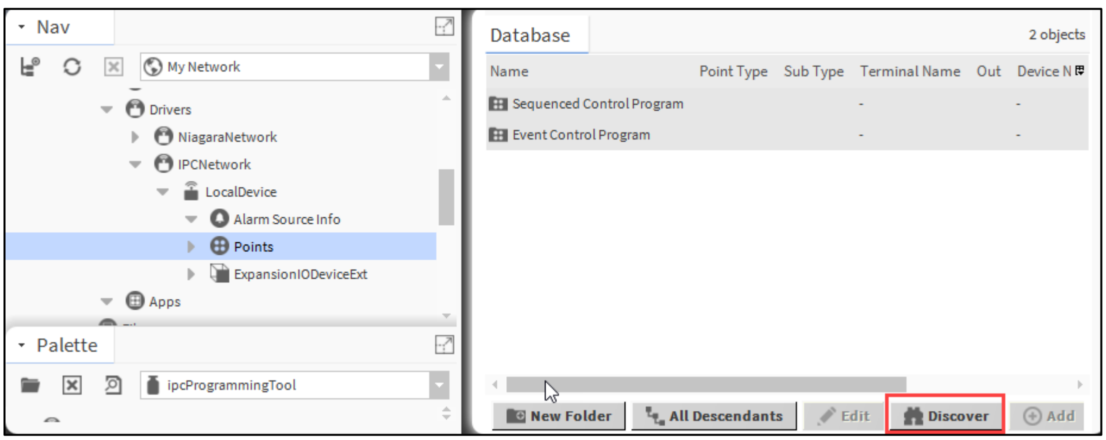
- Discovering on-board or expansion points in IPC
- Adding on-board or expansion I/O points to database

## Discovering On-Board or Expansion Points in IPC

You can discover the points as follows.

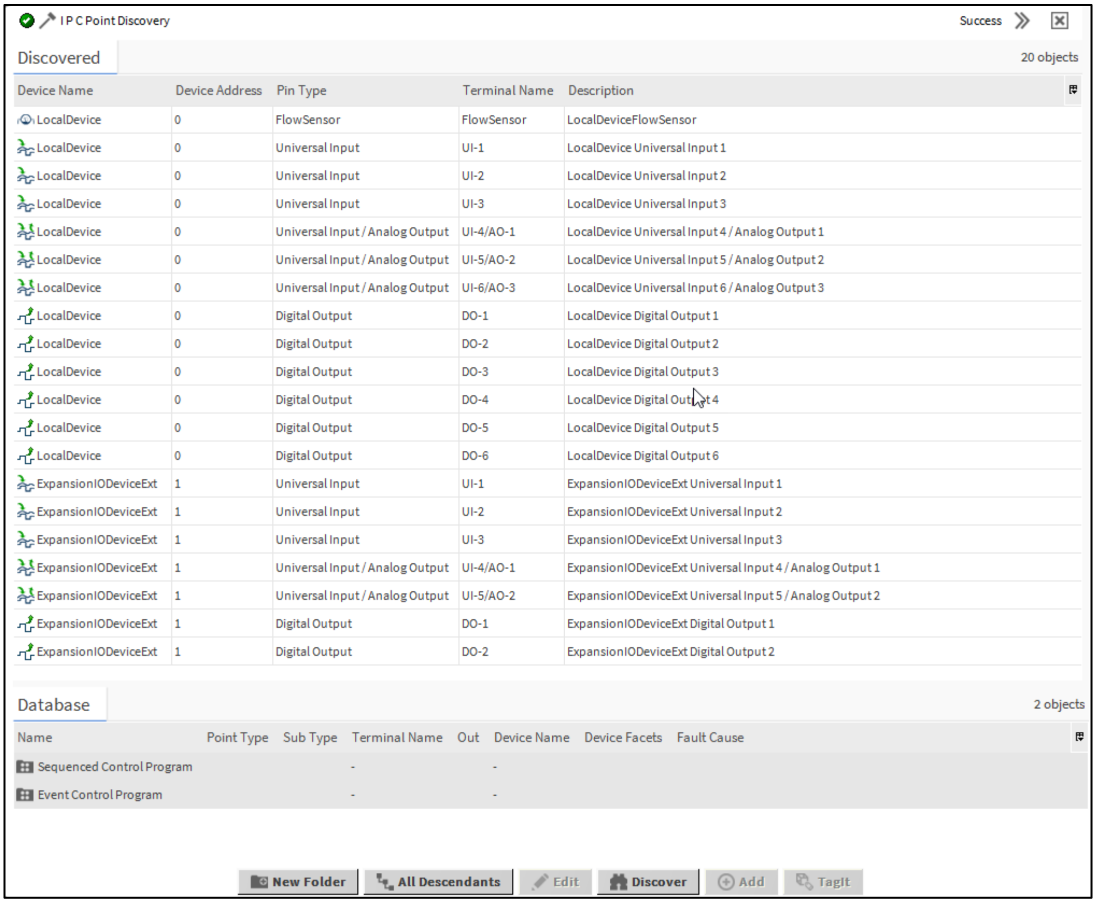
**To discover points in CIPer Model 30:**

1. Navigate to <Required Station> > Config > Drivers > IPCNetwork > LocalDevice.
2. Double-click the Points folder.
3. Click Discover. The application displays all the points present in the local device (CIPer Model 30 controller) and expansion I/Os associated with the controller. The point names are displayed along with address, pin type, terminal number, terminal name, and description. The application also shows a pop-up notification with the status of the IPC Point Discovery at the lower-right side of the screen.




**Note:**

- If you change the device model after discovering the point in Niagara database, the application displays the Fault Cause-Invalid Terminal Number and error message-Error: NoSuchTerminal. Rediscover those points and drag and drop the points manually for the changed device.
- The order in which the list of devices is displayed is first the on-board I/O local devices are displayed followed by the expansion I/O devices in the order same as displayed in the Nav tree.
- You can configure the I/O devices in the CIPer Model 30 programming model without connecting CIPer Model 30 device and the hardware points, that is, you can configure the I/O devices offline also.



I P C Point Discovery — Success

Discovered — 20 objects

| Device Name | Device Address | Pin Type | Terminal Name | Description |
|---|---|---|---|---|
| LocalDevice | 0 | FlowSensor | FlowSensor | LocalDeviceFlowSensor |
| LocalDevice | 0 | Universal Input | UI-1 | LocalDevice Universal Input 1 |
| LocalDevice | 0 | Universal Input | UI-2 | LocalDevice Universal Input 2 |
| LocalDevice | 0 | Universal Input | UI-3 | LocalDevice Universal Input 3 |
| LocalDevice | 0 | Universal Input / Analog Output | UI-4/AO-1 | LocalDevice Universal Input 4 / Analog Output 1 |
| LocalDevice | 0 | Universal Input / Analog Output | UI-5/AO-2 | LocalDevice Universal Input 5 / Analog Output 2 |
| LocalDevice | 0 | Universal Input / Analog Output | UI-6/AO-3 | LocalDevice Universal Input 6 / Analog Output 3 |
| LocalDevice | 0 | Digital Output | DO-1 | LocalDevice Digital Output 1 |
| LocalDevice | 0 | Digital Output | DO-2 | LocalDevice Digital Output 2 |
| LocalDevice | 0 | Digital Output | DO-3 | LocalDevice Digital Output 3 |
| LocalDevice | 0 | Digital Output | DO-4 | LocalDevice Digital Output 4 |
| LocalDevice | 0 | Digital Output | DO-5 | LocalDevice Digital Output 5 |
| LocalDevice | 0 | Digital Output | DO-6 | LocalDevice Digital Output 6 |
| ExpansionIODeviceExt | 1 | Universal Input | UI-1 | ExpansionIODeviceExt Universal Input 1 |
| ExpansionIODeviceExt | 1 | Universal Input | UI-2 | ExpansionIODeviceExt Universal Input 2 |
| ExpansionIODeviceExt | 1 | Universal Input | UI-3 | ExpansionIODeviceExt Universal Input 3 |
| ExpansionIODeviceExt | 1 | Universal Input / Analog Output | UI-4/AO-1 | ExpansionIODeviceExt Universal Input 4 / Analog Output 1 |
| ExpansionIODeviceExt | 1 | Universal Input / Analog Output | UI-5/AO-2 | ExpansionIODeviceExt Universal Input 5 / Analog Output 2 |
| ExpansionIODeviceExt | 1 | Digital Output | DO-1 | ExpansionIODeviceExt Digital Output 1 |
| ExpansionIODeviceExt | 1 | Digital Output | DO-2 | ExpansionIODeviceExt Digital Output 2 |

Database — 2 objects

| Name | Point Type | Sub Type | Terminal Name | Out | Device Name | Device Facets | Fault Cause |
|---|---|---|---|---|---|---|---|
| Sequenced Control Program | | | - | | - | | |
| Event Control Program | | | - | | - | | |

New Folder | All Descendants | Edit | Discover | Add | TagIt

Figure 100: IPC Point Discovery and Its Status

## Adding On-Board or Expansion I/O Points to Database

After adding expansion, I/O devices to the LocalDevice folder, it is necessary that you add the on-board and expansion I/O points to the database, so that the discovered points become the Niagara-compatible physical points. You cannot directly use the physical points onto the wire sheet until you add the points to the database.

You can configure Universal Input/Output (UI/AO) point as input or output point as per requirement. Once the point is added to the database, you cannot change its type. Delete and add the same point again with the required configuration.

To add expansion, I/O points to database:

1. Discover the IPC I/O points.
2. Double-click the required point. The application displays a window to add the device to the database as shown in the following figure.

   Or

   Drag the required device and drop it into the Database tab.

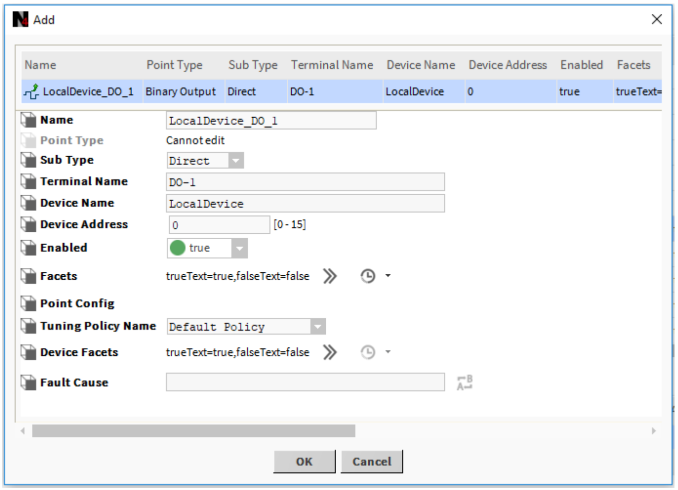



Figure 101: Adding Local Device Point to Database

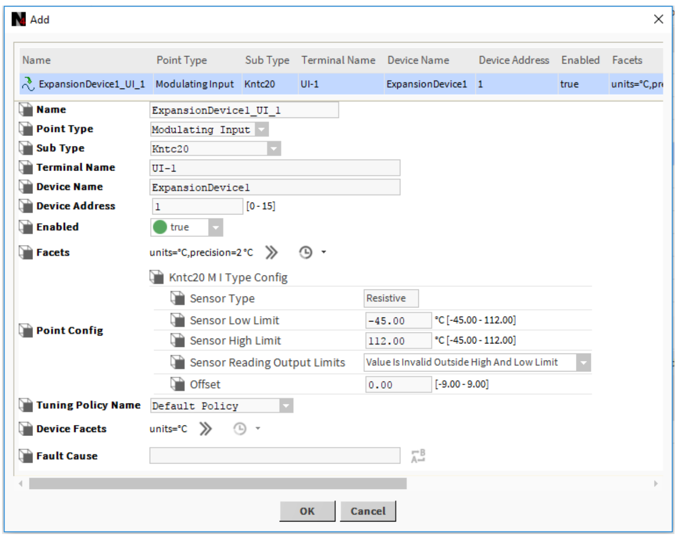
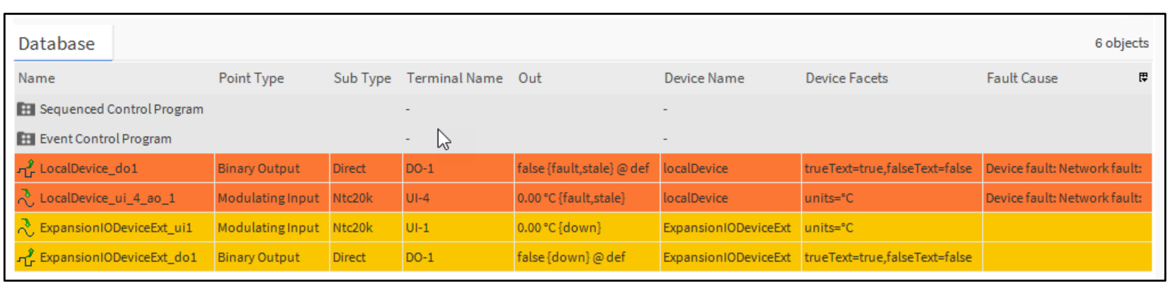



Figure 102: Adding Expansion I/O Point to Database

3. Modify the editable properties of the device as required.
4. Click Ok.

The action adds points into the database, you can view the added points in the Database section.



Database — 6 objects

| Name | Point Type | Sub Type | Terminal Name | Out | Device Name | Device Facets | Fault Cause |
|---|---|---|---|---|---|---|---|
| Sequenced Control Program | | | - | | - | | |
| Event Control Program | | | - | | - | | |
| LocalDevice_do1 | Binary Output | Direct | DO-1 | false {fault,stale} @ def | localDevice | trueText=true,falseText=false | Device fault: Network fault: |
| LocalDevice_ui_4_ao_1 | Modulating Input | Ntc20k | UI-4 | 0.00 °C {fault,stale} | localDevice | units=°C | Device fault: Network fault: |
| ExpansionIODeviceExt_ui1 | Modulating Input | Ntc20k | UI-1 | 0.00 °C {down} | ExpansionIODeviceExt | units=°C | |
| ExpansionIODeviceExt_do1 | Binary Output | Direct | DO-1 | false {down} @ def | ExpansionIODeviceExt | trueText=true,falseText=false | |

Figure 103: Device Added to Database

You can add multiple discovered I/O points to the database, by hold down Ctrl while clicking the right mouse button on the required points.

Also, while adding the multiple I/O points, you can edit the properties of all the selected I/O points, if the device properties like Point Type, Sub Type, Terminal Name are same for the multiple I/O points, If device properties like Point Type, Sub Type, Terminal Name are different for selected I/O points, then some of the properties are non-editable.

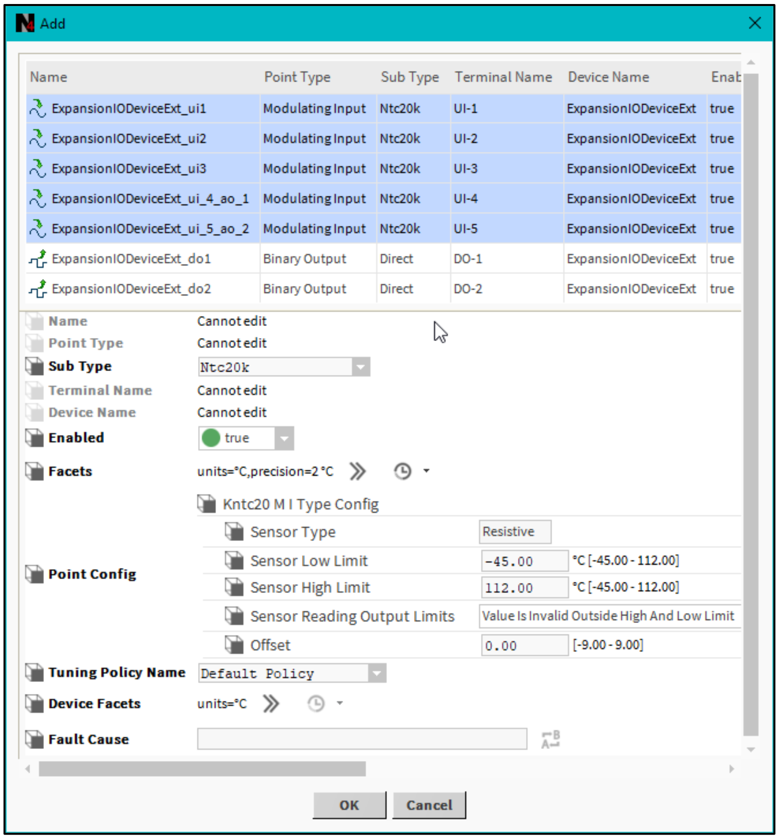


Figure 104: Setting Value for Multiple Points in Add Window

The following table describes the input fields available while adding a point to database.

Table 12: Properties and Description to Add Point to Database

| Property | Description |
|---|---|
| Name | Name of the point. |
| Point Type | Type of Input i.e. modulating or Binary |
| Sub Type | Sub type of the point i.e. sensor or signal details |
| Terminal Name | Details of the terminal port. |
| Device Name | Name of the associated device. You can change the device name in the Nav tree only. In the Add window it is not editable. |
| Device Address | Physical address of the associated controller. If you change the value and add the point into the database, the application displays the error message-Error: NoSuchTerminal. |
| Enabled | Status of the device. If Enabled is set to true, the point is enabled. If Enabled is set to false, the point is disabled. |
| Facets | Primarily, facets determine how the point’s value displays in the station. Examples include engineering units and decimal precision for numeric types, and descriptive value (state) text for boolean and enum types |

| Property | Description |
|---|---|
| Point Config | You can set specific property values for configuration of the point. The point configuration properties differ for each point type and sub type. |
| Tuning Policy Name | Name of the Tuning policy |
| Device Facets | Default unit of the respective point. This property is not editable. |
| Fault Cause | Indicates the reason why a network, component, or extension is in fault. This field is empty unless a fault exists. |

The following table describes different scenarios of operations that you can perform and their results.

Table 13: Different Operations and Their Results

| Operation | Result | Description | Solution |
|---|---|---|---|
| Changing the device name in the Nav tree | | Automatically reflects in the database | |
| Changing the device address in the Property Sheet | | Automatically reflects in the database | |
| Adding a point with the same Terminal Number to the database more than once | Error | Fault Cause-Terminal Number is Duplicate on this device address. | Deleting any one point from the database removes the error. |
| Deleting any expansion I/O module after adding its points to database | Error | Error-NoSuchDevice | |
| Adding a point from a disabled device to the database | Error | Point row becomes gray colored | Enable the expansion I/O device from its Property Sheet. |
| Disabling the LocalDevice | Error | Error-disabled.<br>All I/Os are disabled under Points folder, because Points folder is inside LocalDevice folder. See the exception to this case in the following Note. | Enable the LocalDevice from the Property Sheet. |

Note:

- Exception: If you want to use expansion I/O devices, but not the local devices, and disable the LocalDevice, you cannot do so, because disabling the LocalDevice disables all the devices including local and expansion I/O devices.
- You can replicate the same device with different Device Address to the database more than once.
- If folder is removed or copy pasted (with I/O) then perform this action, right click on Pointsfolder > Action > Reconfigure points.

# Remotely Mounted Expansion Module

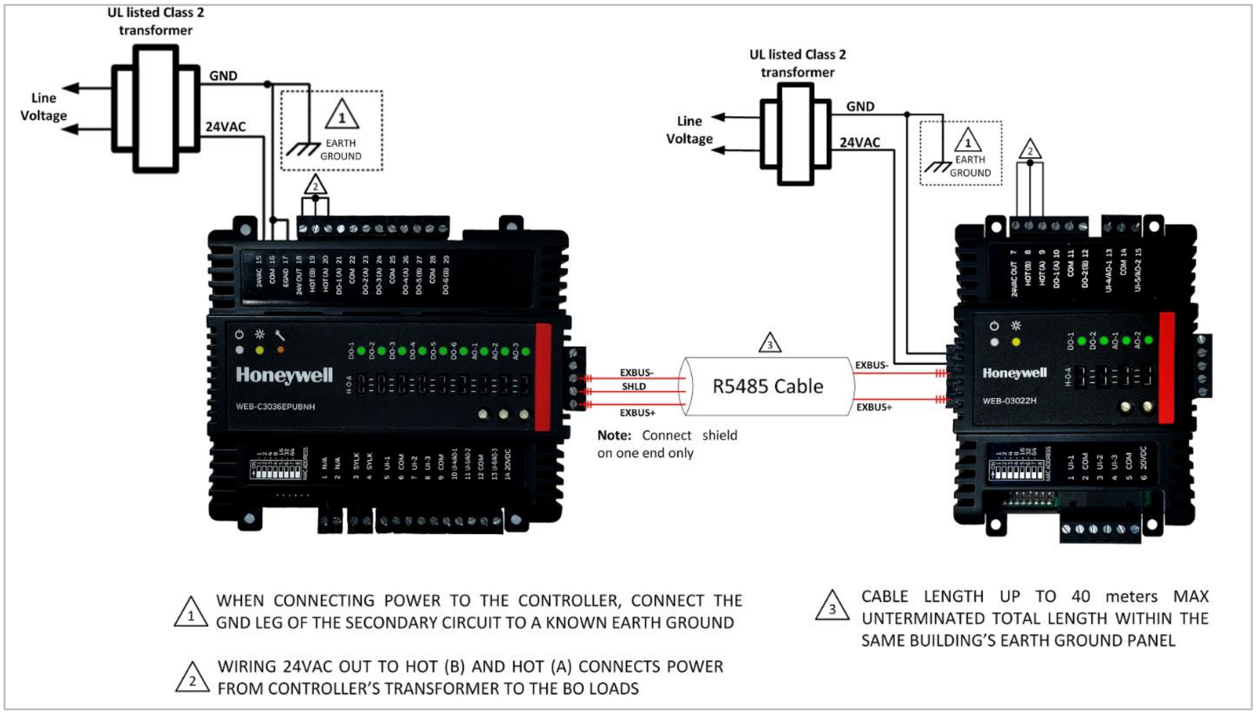



Figure 105: Remotely Mounted Expansion Module

The cable length is up to 40 meters unterminated total length within the same building's earth ground plane.

## Hand Off Auto (HOA) Switch

HOA switches (physical) are provided for each output—Analog and Binary—present on the CIPer Model 30 controller or expansion I/O. You can configure the HOA modes via physical switches present on the CIPer Model 30 controller and expansion I/O modules only. The H-O-A switch overrides all the software commands when set to HAND or OFF.

### Digital Outputs (DOs)

- If you configure the HOA switches to Off Mode, the Physical Digital Output is disabled irrespective of the Control program logic output.
- If you configure the HOA switches to Auto Mode, the Physical Output responses as per the value provided by the Control program logic output.
- If you configure the HOA switches to Hand Mode, the Physical Output is enabled irrespective of the Control program logic output.

### Analog Outputs (AOs)

When the H-O-A switch is in HAND position, the H-O-A trimpot drives the output from 0–100% (0–20 mA or 0–10 VDC as appropriate) and it ignores the Control program logic output.

When the switch is in OFF position, the output is at 0%. The trimpot value covers the full range of the output regardless of the Control program logic output.

If you configure the HOA switches to Auto Mode, the Physical Output responses as per the value provided by the Control program logic output.

| Note: |
|---|
| When you change the HOA switch to Off, Auto, and Hand mode in the physical device, there is a delay of one second. This is to avoid the switch mode to change when it is mistakenly changed. So, if you want to see the switch mode to change, you need to keep the switch in the required mode at least for one second, otherwise the device ignores the change made by switch. |

# Actions

Use the Actions option to quickly force values to Network Input points. These options can be used to set values based on the priority:

Emergency Override > Override > Set.

Right-click the point on the wire sheet and select Actions.

The actions allowed in the Online and Offline mode are:

## Input Actions

The following table lists the actions available for numeric and Boolean input points.

Table 14: Input Actions

| Action | Description | Point Type | |
|---|---|---|---|
| | | Numeric Input Points | Boolean Input Points |
| Override | This option allows you to override the actual value of the input received from sensor as per requirement. | Available | Not available |
| Auto | The Auto option removes the override value from the input and input shows the actual value. | Available | Available |
| Active | If Action is set to Active, the Input point becomes True irrespective of the actual value sensed by sensor. | Not available | Available |
| Inactive | If Action is set to Inactive, the Input point becomes False irrespective of the actual value sensed by sensor. | Not available | Available |

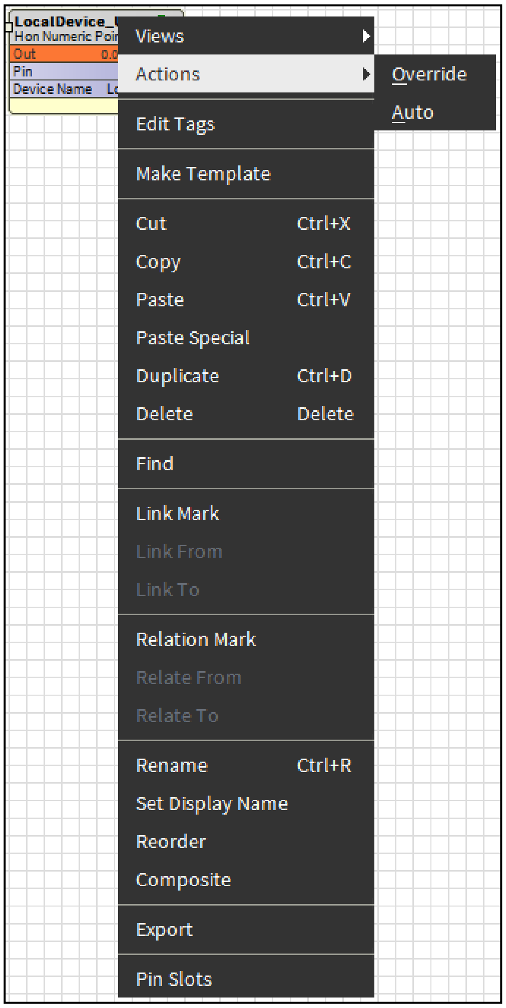



Figure 106: Input Actions for Analog Input Points

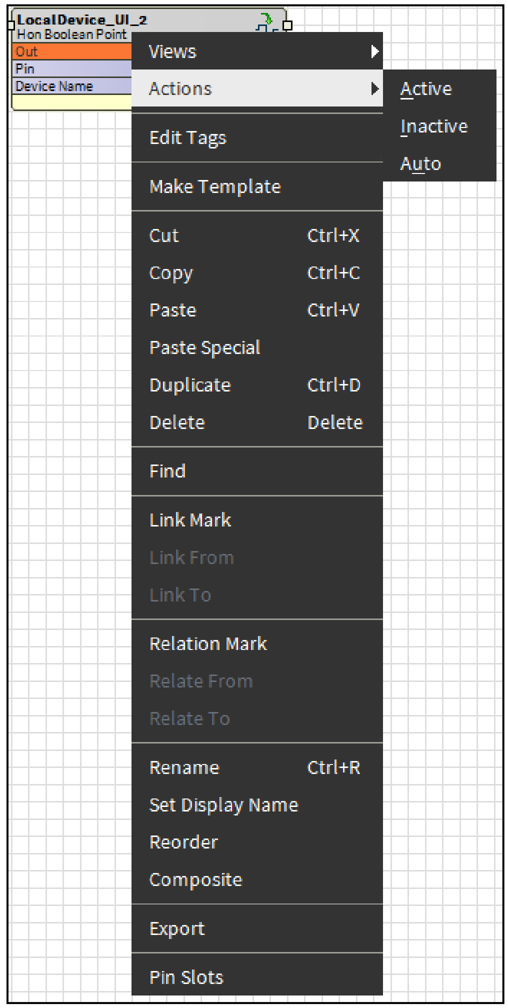



Figure 107: Input Actions for Binary Input Points

To override input:

1. Right-click the input function block on the wire sheet or in the Nav tree.
2. Navigate to Actions and Override. The Override dialog box to enter the new value is displayed.

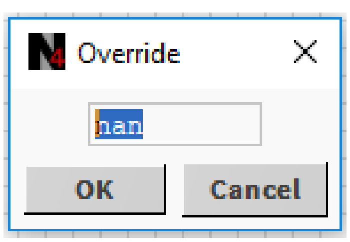



Figure 108: Override Dialog Box

3. Enter the new value in the input field of the Override dialog and click Ok.

Similarly, you can select the Auto option and the application removes the overridden value from the input and input shows the actual value.

## Output Actions

The following table lists the actions available for numeric writable and Boolean writable output points.

| Action | Description | Point Type | |
|---|---|---|---|
| | | Analog Output Point | Binary Output Point |
| Emergency Override | Among other input actions Emergency Override has the highest priority. When you select the Emergency Override option, the value set for the Emergency Override is passed to the input point. | Available | Not available |
| Emergency Auto | This option overrides the Emergency Override value and the output point takes the value set by Override. Auto clears off the Override state of the point and the point is assigned the Sine/Cosine/Range value, if it is set. | Available | Available |
| Emergency Active | This has the highest priority. If Action is set to Emergency Active, the Output point becomes True irrespective of the actual value determined by the Control Program. | Not available | Available |
| Emergency Inactive | This has the highest priority. If Action is set to Emergency Inactive, the Output point becomes False irrespective of the actual value determined by the Control Program. | Not available | Available |
| Override | The Override has second-highest priority. The value set by Override is taken when the Emergency Auto option is selected, and the Override value is set. | Available | Not available |
| Active | This has the second highest priority. If Emergency Auto is selected (that is Emergency Active and Emergency Inactive are removed) and the Action is set to Active the Output point becomes True irrespective of the actual value determined by the Control Program. | Not available | Available |
| Inactive | This has the second highest priority. If Emergency Auto is selected (that is Emergency Active and Emergency Inactive are removed) and the Action is set to Inactive, then the Output point becomes False irrespective of the actual value determined by the Control Program. | Not available | Available |
| Auto | The Auto option overrides the Override option, that is selecting Auto removes the value set by Override. | Available | Available |
| Set | The Set option has the least priority among other actions. The value of Set is assigned to a point when Clear Sine/Cosine/Range option is selected, and the Set value is already defined. | Available | Available |

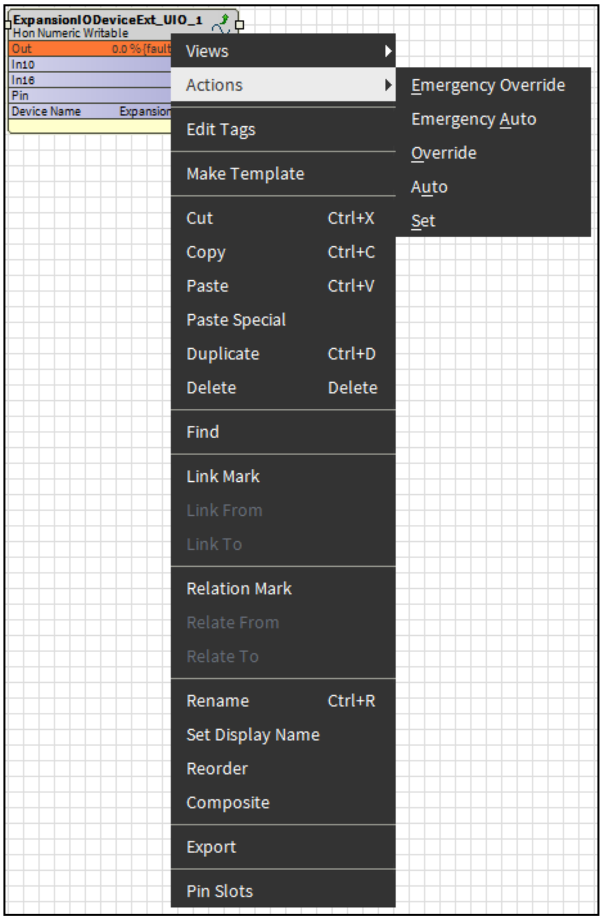



Figure 109: Output Actions for Analog Output Points

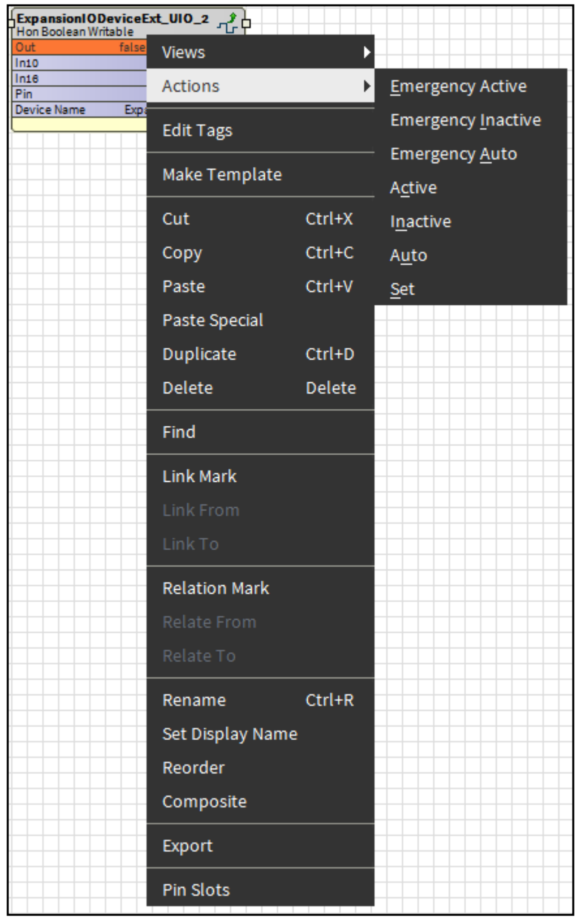



Figure 110: Output Actions for Binary Output Points

# Order of Execution

The order of execution defines the sequence in which function blocks are executed by the controller. When the function blocks are dragged onto the wire sheet to build the application logic, by default, the tool sets the execution order of the function blocks in the order they are dropped onto the wire sheet. However, you can alter the order in which the controller executes the function blocks by reordering the blocks. In the Simulation Mode, the order of execution that you set is followed.

Note:

- The execution of function blocks can be reordered only. Although Software and Physical points are shown in the Reorder screen, their order of execution cannot be reordered.
- When a block is removed, the order of execution gets affected.
- The order of execution cannot be changed for Built In function blocks.

To change the order of execution:

1. Drag the function blocks onto the Sequenced Control Program wire sheet from the ipcProgrammingTool palette. The order in which the function blocks are dragged determines the execution order. The execution order is displayed on the container of each function block on the wire sheet.
2. Right-click the required container or Sequenced Control Program in the Nav tree and select Reorder. The Reorder window is displayed.

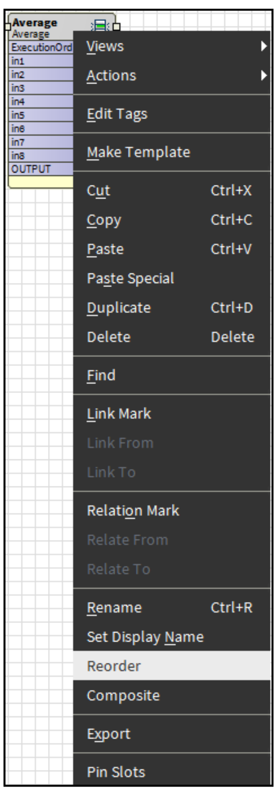



Figure 111: Reorder option

3. Select the required application and click Move Up or Move Down to change the order of execution.

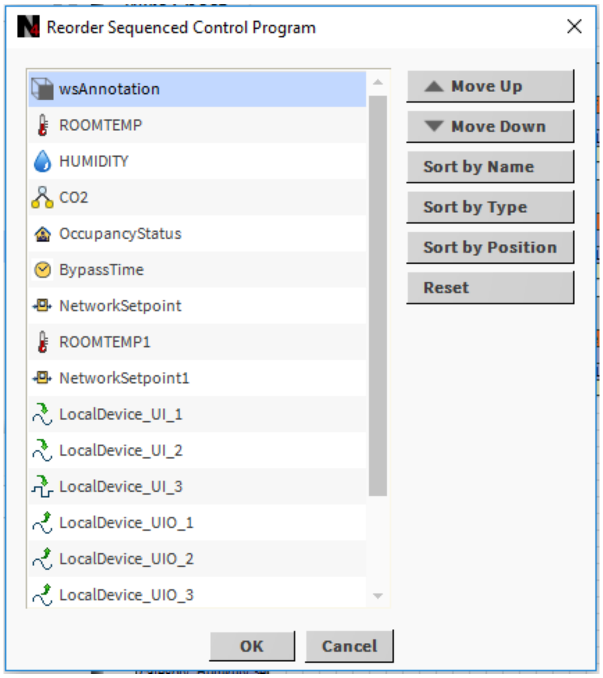



Figure 112: Reorder ControlProgramScreen

4. Click OK to close the dialog box. The value, in the Online mode, is directly written to the controller.

   Or

   Click Cancel, if you do not want to save the changes.

# Physical Points

Physical points are logical objects that are used in building application logic. Depending on the model selected, default (Fixed) physical points are available.

In CIPer Model 30 controller, three types of physical points are available to configure in the required application:

- Universal Input can be configured as:
  - Binary Inputs
  - Modulating Inputs
- Universal Input / Outputs can be configured as:
  - Binary Inputs
  - Modulating Inputs
  - Binary Outputs
  - Modulating Output
- Digital Outputs

Physical points are logical objects that are used in building application logic. Depending on the model selected, default (Fixed) Physical points are made available.

The CIPer Model 30 programming model automatically validates the rules based on the model selected. Five types of physical points are available to configure in the required application:

- ModulatingInput
- ModulatingOutput
- BinaryInput
- BinaryOutput-Direct
- BinaryOutput-SlowPWM

To know how to add, configure, override, clear override, and delete a physical point block, and remove non-required pin slots of a physical point block, see Function Block Details section under Function Block Library.

## ModulatingInput

A ModulatingInput is a physical input. You can use the ModulatingInput function block while creating the application logic.

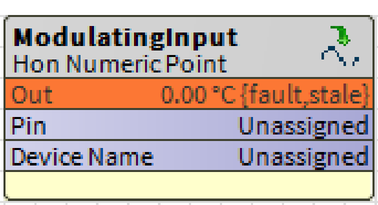



Figure 113: ModulatingInput Function Block

## ModulatingOutput

A ModulatingOutput is a physical output. It is used to create ModulatingOutput in the application logic.

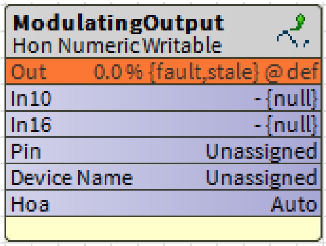



Figure 114: ModulatingOutput Function Block

### BinaryInput

A BinaryInput is a physical input. You can use the BinaryInput function block while creating the application logic.

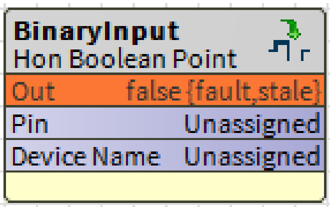



Figure 115: BinaryInput Function Block

### BinaryOutput-Direct

A BinaryOutput is a physical output. You can use the BinaryOutput function block while creating the application logic.

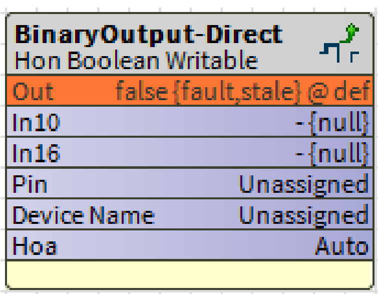



Figure 116: BinaryOutput-Direct Function Block

### BinaryOutput-SlowPWM

A BinaryOutput-SlowPWM is a physical output. You can use the BinaryOutput-SlowPWM function block while creating the application logic.

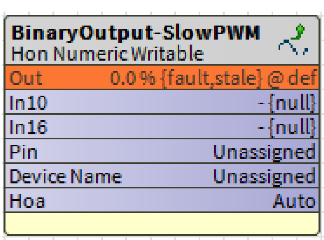



Figure 117: BinaryOutput-SlowPWM

PWM Configuration: This is a sub type in the property sheet of the function block. When the sub type is Slow Pwm, enter the values for the following input fields:

Period: The range is from 1 to 3276.7 seconds in tenths of seconds. It is the time of a one cycle of the pulse width modulation.

Zero Time: When 0% command is given then the pulse width is equal to the value specified in this parameter.

Full Time: When 100% command is given then the pulse width is equal to the value specified in this parameter.

To know about the properties of above function blocks, see Using On-board and Expansion I/Os section under I/O Programming.

### Totalizer

Totalizer sums up the pulses per second.

### Counter

Counter counts the value per second.

## Point Status Behaviors

The status is indicated by text on a colored background.

The following table lists the status types, the default colors, and their meaning.

Table 15: Point Status Behaviors

| Type | Default Color Example | Meaning |
|---|---|---|
| alarm | white text, red background<br>65.0°F | Point currently has a value in an alarm range, as defined by property in its alarm extension. |
| fault | black text, orange background<br>65.0°F | Originates from a proxy point only. Typically indicates a configuration or licensing error. If it occurs after normal operation, it may indicate a native fault in device, or the point's parent device has a fault status. |
| overridden | black text, magenta background<br>65.0°F | Current point control is from an action, meaning a user-invoked command at either priority level 8 (override) or priority 1 (emergency). |
| disabled | gray text, light gray background<br>65.0°F | Originates from a proxy point only. Point (or its parent device or network) has been manually disabled (property enabled = false). |
| down | black text, yellow background<br>65.0°F | Originates from a proxy point only. Driver communications to the parent device are currently lost, based upon the device status (Monitor) configuration for that network. |
| stale | black text, tan background<br>65.0°F | Originates from a proxy point only. Driver communications have not received a requested response for this data item within the configured times (Tuning period). |
| null | (No color indication) | Current point control has entered a null state, vs. a specific value and priority level. Typical to fallback operation for a writable point. |

| Type | Default Color Example | Meaning |
|---|---|---|
| | | Note: If linking a null status Out to a simple data slot, the *point's null value is processed.* |
| unackedAlarm | (No color indication) | Last point alarm event has not yet received user acknowledgment. Point's alarm extension uses alarm class requiring acknowledgment. |

## License Requirements and Behaviors

Following is the list of licensing requirements for CIPer Model 30 programming tool.

- If the CIPer Model 30 programming tool is not licensed, any of the components in the controller does not function.
- The license installed for CIPer Model 30 controller should be valid till the life of the CIPer Model 30 programming tool, so that you need not renew and install the license every year.
- You do not need any separate license to use the on-board I/Os and can use the on-board I/Os with the basic CIPer Model 30 license.

## Configuring UI or UI/AO as Modulating Inputs

You can configure the Modulating Input blocks and use them while adding the discovered physical UI/AO points into the database.

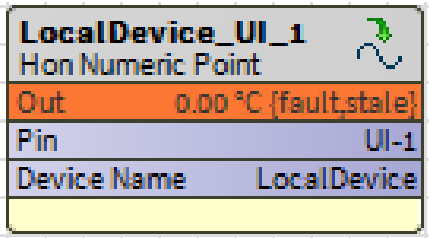



Figure 118: UI configuration as Modulating Input

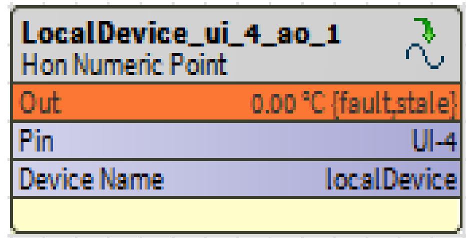



Figure 119: UI/AO configuration as Modulating Input

To add and configure the modulating input block:

1. Discover the CIPer Model 30 I/O points.
2. Double-click the required UI / UI/AO point. The application displays a window to add the device to the database as shown in the following figure.

   Or

   Drag the required UI / UI/AO point and drop it into the Database tab.
3. Select Point Type as Modulating Input from the Point Type drop-down menu.

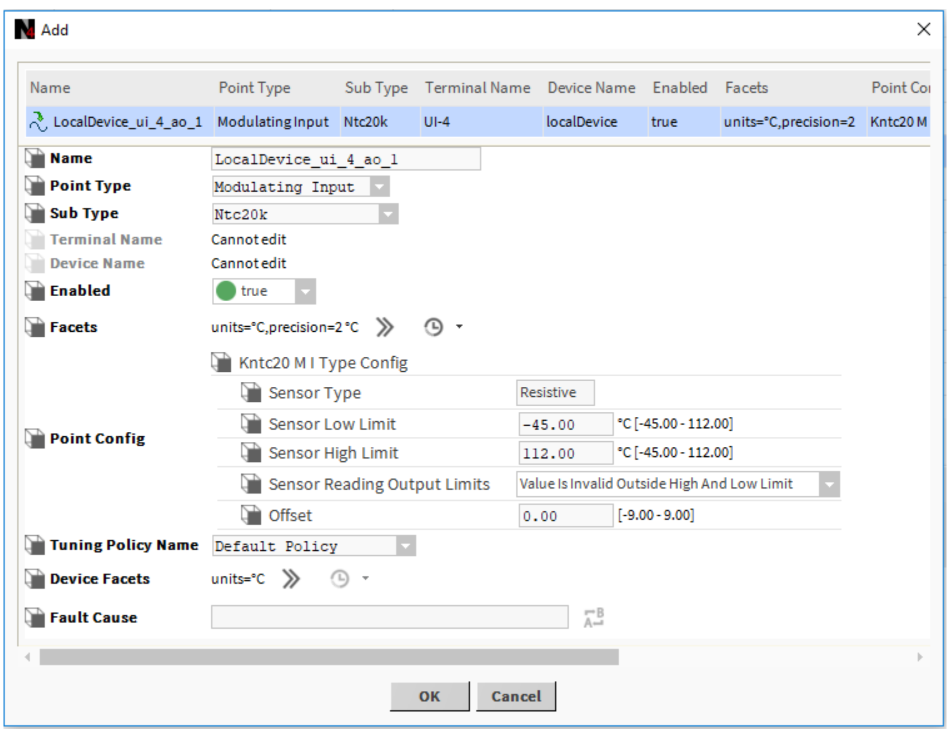
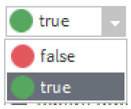



Figure 120: Adding Modulating Input to Database

The following table describes the configuration properties and their definitions.

Table 16: Configuration of Modulating Input Properties

| Name | Definition |
|---|---|
| Point Name | Enter the name of the function block or use the default names given by the application. |
| Point Type | Select Modulating Input from the Point Type drop-down menu. |
| Sub Type | Select a sensor type from the Sub Type drop-down menu. |
| Enable | Select True to enable the selected point.<br><br> |
| Facets | Click  to view the details of the facets for the network/internal data type.<br>The following information can be configured:<br>• Minimum: The minimum limit for selected unit<br>• Maximum: The maximum limit for selected unit<br>Or<br>• Range: Indicates the possible enumeration with its ordinal for a selected unit<br>• Units: Indicates the unit symbol for the selected unit (if it shows null, it means the unit symbol is not applicable there.)<br>• Resolution: Indicates scaling factor for the selected unit. When a value is written to the controller, the value is divided by the value specified in the Resolution field and when it is read from the controller, it is multiplied by the resolution value before it is displayed in Niagara.<br>• Precision: Precision for the selected unit |
| Point Configuration | Displays the unit of measurement for the selected point type. This is enabled when Resistive or Voltage is selected in the Sensor Type field.<br>Enter the values for:<br>• Sensor High Limit: Enter an upper limit in the High Limit field.<br>• Sensor Low Limit: Enter a lower limit in the Low Limit field.<br>• Sensor Reading Output Limits:<br>○ If Value is Invalid Outside High And Low Limit option is selected, and when output crosses the limit then output becomes invalid.<br>○ If Clamp Value To High and Low Limits option is selected and if output crosses the High Limit or Low Limit then output is clamped to Low or High Limit, and it doesn't become invalid. |

4. Click Ok, to save the information updated.

   Or

   Click Cancel, if you do not want to save the changes.

## Configuring UI or UI/AO as Pulse Meter or Counter

In CIPer Model 30 controller model Unitary and VAV, all UI & UI/AO points can be configured as Pulse Meter or Counter type sensor.

When the modulating input is configured to type Pulse_Meter in the CIPer Model 30 Controller, it reads the number of pulses per hour. The algorithm averages the readings depending on the rate at which the pulses come in.

- For fast pulses (< 20 seconds apart), the average of the last 4 readings is taken.
- For medium pulses (< 40 seconds apart), the average of the last 2 readings is taken.
- For slow pulses (>40 seconds apart), the last reading is taken.

When the pulses stop coming in, the power gradually decreases and goes to 0 in approximately 11 minutes. The maximum measured rate is 54000 pulses per hour. The calculated output of a pulse meter input is in pulses per hour. This can be connected to the function block logic (multiply by scale factor) to the computer power. For example, 1 pulse = 1 W and 10 is added as a scale factor to the pulse. If pulse meter receives 3600 pulses per hour, actual power consumption is equal to:

3600 * 10 = 36 KW

You can connect this to the function block logic and accumulate counts.

To add and configure the pulse input block:

1. Discover the CIPer Model 30 I/O points.
2. Double-click the required UI / UI/AO point. The application displays a window to add the device to the database as shown in the following figure.

   Or

   Drag the required UI / UI/AO point and drop it into the Database tab.
3. Select Point Type as Modulating Input from the Point Type drop-down menu.
4. Select Sub Type as Pulse Meter or Counter from the Sub Type drop-down menu.

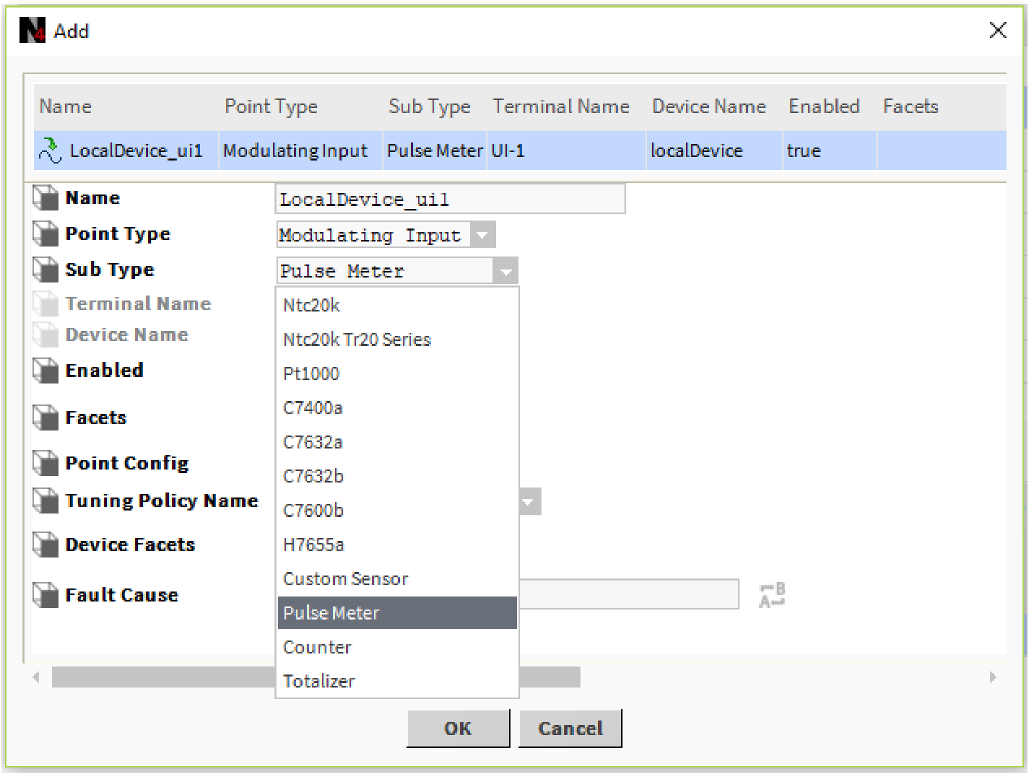



Figure 121: Adding Modulating Input to Database

5. Click Ok to save the information updated.

   Or

   Click Cancel, if you do not want to save the changes.

   You can change the poll frequency of the Pulse Meter and Counter to Slow (), Fast (every second), and Normal (once in 5 seconds) as per requirement in the property sheet.

## Configuring UI or UI/AO as Custom Sensor

If the sensor does not meet any standard characteristics of the available sensors, you can select the Type as Custom Sensor and set its characteristics. There are three type of custom sensor.

- Resistive Input: A Resistive Input Point is an Analog Point which configures a UI / UI/AO to read a resistance from 0 to 300000 ohms. Configuration provides for default (ohms).
- Voltage Input: A Voltage Input Point is an Analog Point which configures a UI / UI/AO to read a Voltage signal from 0 volt to 10.3 Volt. Configuration provides for default (Volts).
- Current Input: A Current Input Point is an Analog Point which configures a UI / UI/AO to read a resistance from 0 to 20mA. Configuration provides for default (mA).

### Resistive Sensor

A Restive Input Point is an analog point which configures a UI / UI/AO to read a resistance signal range from 0 to 300000 ohms.

To define custom sensor:

1. Discover the CIPer Model 30 I/O points.
2. Double-click the required UI point. The application displays a window to add the device to the database as shown in the following figure.

   Or

   Drag the required UI point and drop it into the Database tab.
3. Select Point Type as Modulating Input.
4. Select the Sub Type as Custom Sensor. If the Sub Type is selected as Custom Sensor, you must set the Linearization, otherwise the custom sensor does not function properly, and shows a fault cause.

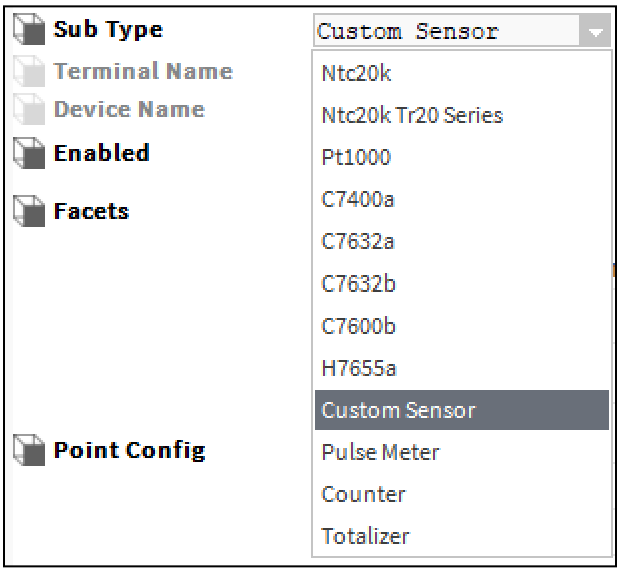



Figure 122: Selecting Custom Sensor in the Sub Type Drop-Down Menu

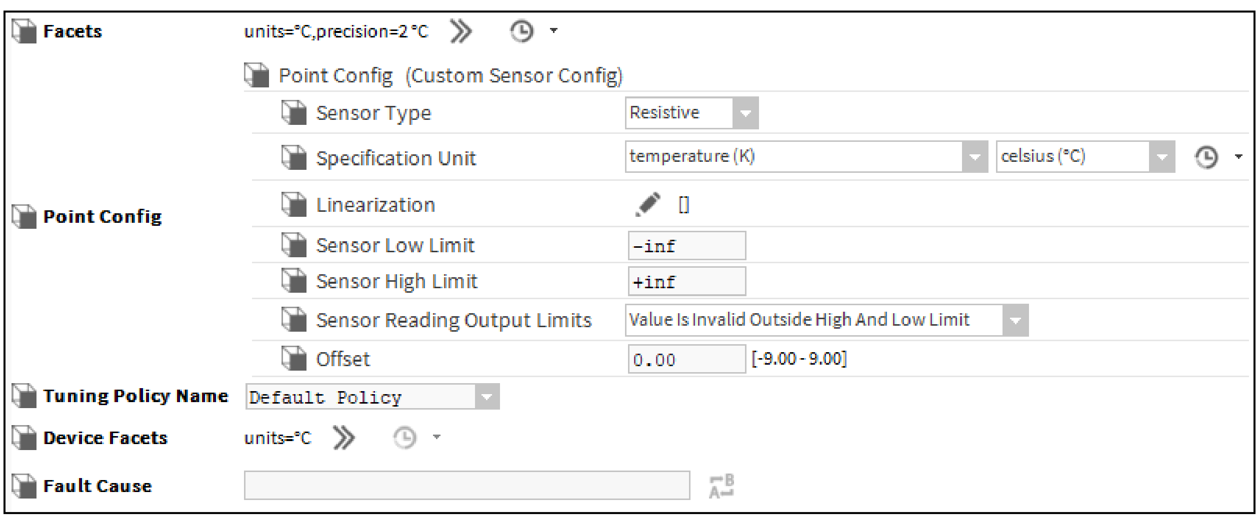



Figure 123: Custom Sensor Configuration

5. Click  to configure the facets.
6. Select Sensor Type as Resistive.

   Specification Unit option in the drop-down menu depends upon the Data Type selected while configuring the details on the Add window.

7. Click next to the Linearization field, where the linearization points are provided to define the characteristics of the sensor. The Tabular Conversion Dialog window is displayed.

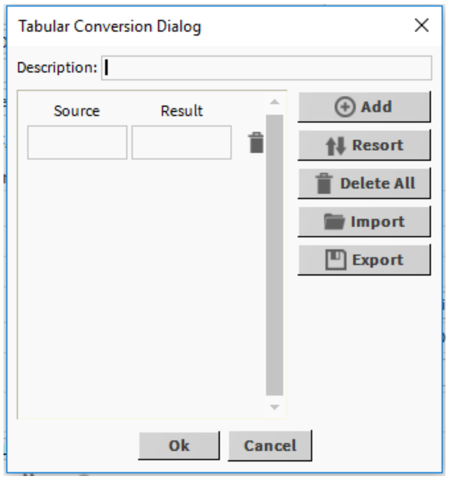



Figure 124: Tabular Conversion Dialog Window

8. Click Add.
9. Enter the sensor resistance value for the electrical signal in the Source field and enter the corresponding value of the parameter which is being sensed in the Result field.

   Enter the low and high limit sensor values in the Sensor Low Limit (0 ohm) and Sensor High Limit (300000 ohm) fields respectively as per Sensor Data Sheet.
10. Select the value for Sensor Reading Output Limits from the respective drop-down menus.
    - If Value is INVALID outside High Limit option is selected, and when output crosses the limit, the output becomes invalid.
    - If Clamp Value as High and Low Limits option is selected, and if output crosses the high limit or low limit, the output is clamped to Low or High Limit and it doesn't become invalid.
11. Click Save after entering the required details.

## Voltage Input

A Voltage input point is an analog point which configures a UI / UI/AO to read a Vdc signal range from 0 to 10.3V.

To define voltage input:

1. Discover the CIPer Model 30 I/O points.
2. Double-click the required UI point. The application displays a window to add the device to the database as shown in the following figure.

   Or

   Drag the required UI point and drop it into the Database tab.
3. Select Point Type as Modulating Input.
4. Select the Sub Type as Custom Sensor.

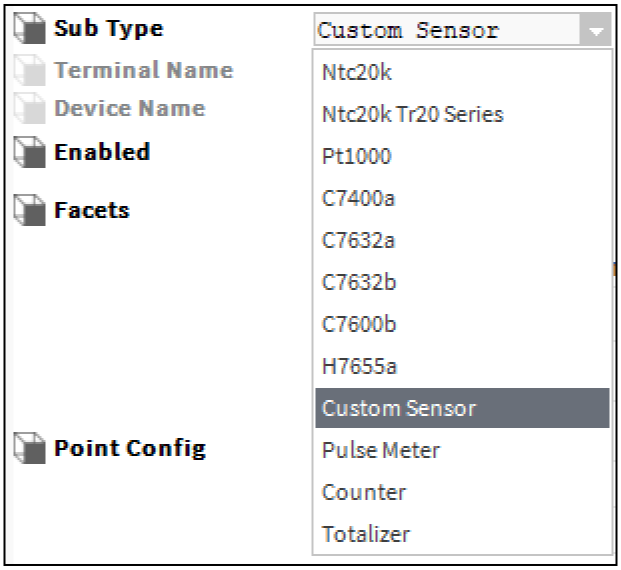



Figure 125: Selecting Custom Sensor in Sub Type Drop-Down Menu

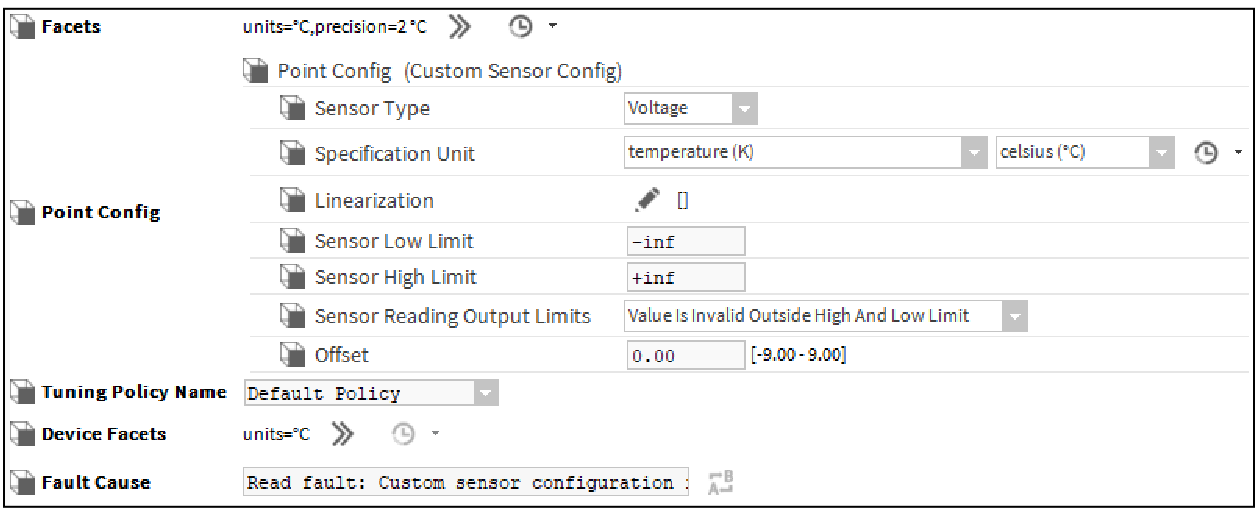



Figure 126: Custom Sensor Configuration

5. Click to configure the facets.
6. Select Sensor Type as Voltage as per requirement.
7. Specification Unit option in the drop-down menu depends upon the Data Type selected while configuring the details on the Add window.
8. Click next to the Linearization field, where the linearization points are provided to define the characteristics of the sensor. The Tabular Conversion Dialog window is displayed.

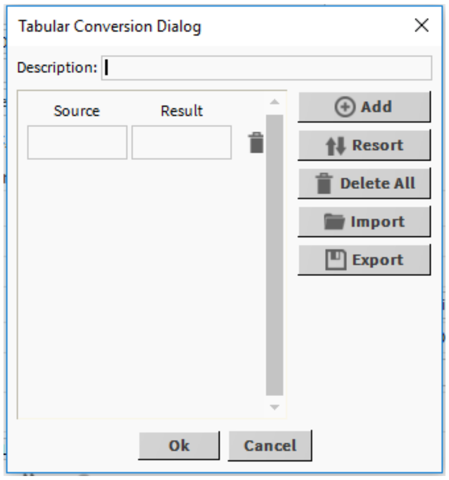



Figure 127: Tabular Conversion Dialog Window

9. Click Add.
10. Enter the sensor voltage value for the electrical signal in the Source field and enter the corresponding value of the parameter which is being sensed in the Result field.

    Enter the low and high limit sensor values in the Sensor Low Limit (Sensor value at 0 Volt) and Sensor High Limit (Sensor value at 10.3 Volt) fields respectively as per Sensor Data Sheet.
11. Select the value for Sensor Reading Output Limits from the respective drop-down menus.
    - If Value is INVALID outside High Limit option is selected, and when output crosses the limit, the output becomes invalid.
    - If Clamp Value as High and Low Limit option is selected, and if output crosses the high limit or low limit, the output is clamped to Low or High Limit and it doesn't become invalid.
12. Click Save after entering the required details.

## Current Input

A Current input point is an analog point which configures a UI / UI/AO to read a current signal range from 0 to 20mA.

To define Current input:

1. Discover the CIPer Model 30 I/O points.
2. Double-click the required UI point. The application displays a window to add the device to the database as shown in the following figure.

   Or

   Drag the required UI point and drop it into the Database tab.
3. Select Point Type as Modulating Input.
4. Select the Sub Type as Custom Sensor.

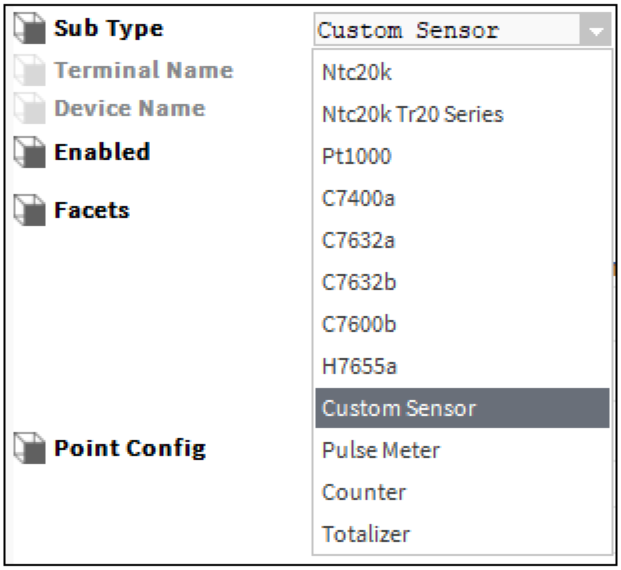



Figure 128: Selecting Custom Sensor in Sub Type Drop-Down Menu

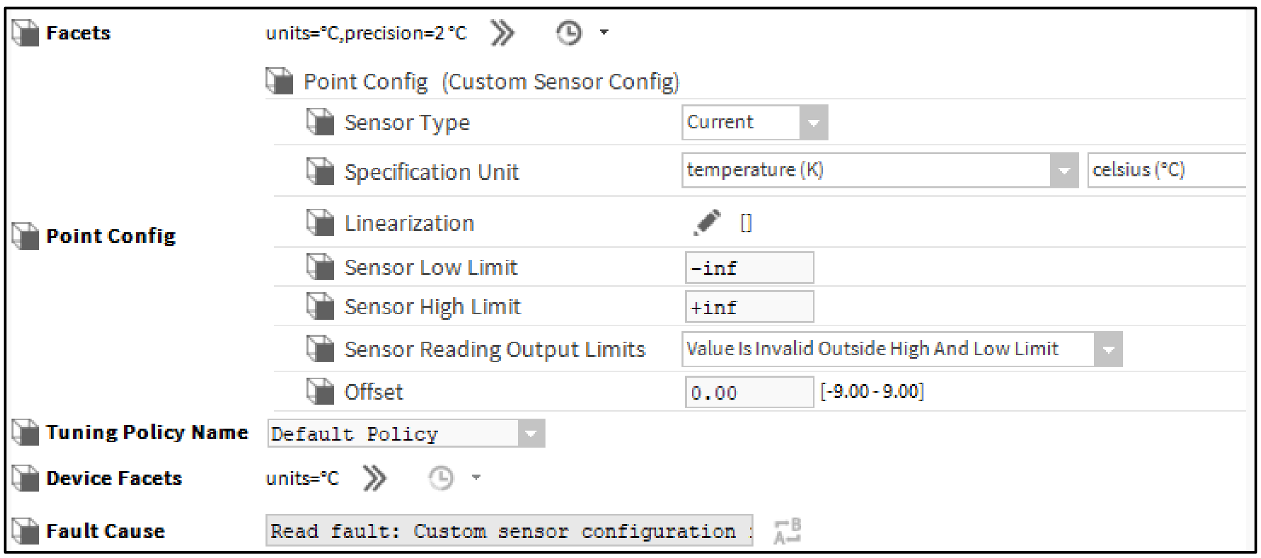



Figure 129: Custom Sensor Configuration

5. Click to configure the facets.
6. Select Sensor Type as Current as per requirement.
7. Specification Unit option in the drop-down menu depends upon the Data Type selected while configuring the details on the Add window.
8. Click next to the Linearization field, where the linearization points are provided to define the characteristics of the sensor. The Tabular Conversion Dialog window is displayed.

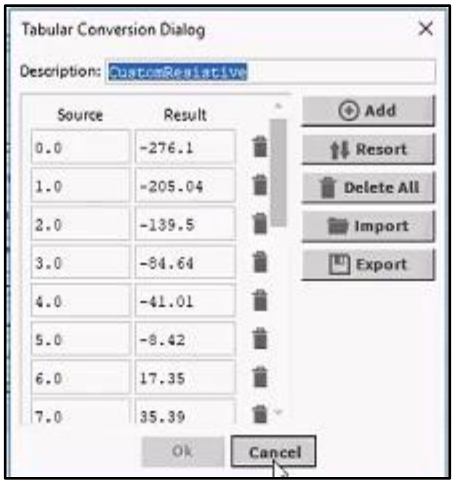



Figure 130: Tabular Conversion Dialog Window

9. Click Add.
10. Enter the sensor current value for the electrical signal in the Source field and enter the corresponding value of the parameter which is being sensed in the Result field.

    Enter the low and high limit sensor values in the Sensor Low Limit (Sensor value at 0 mA) and Sensor High Limit (Sensor value at 20mA) fields respectively as per Sensor Data Sheet.
11. Select the value for Sensor Reading Output Limits from the respective drop-down menus.
    - If Value is INVALID outside High Limit option is selected, and when output crosses the limit, the output becomes invalid.
    - If Clamp Value as High and Low Limit option is selected, and if output crosses the high limit or low limit, the output is clamped to Low or High Limit and it doesn't become invalid.
12. Click Save after entering the required details.

## Configuring UI or UI/AO as Ntc20k

Negative Temperature Coefficient (NTC) are thermistors, which decrease in resistance when the temperature increases.

To define Ntc20k:

1. Discover the CIPer Model 30 I/O points.
2. Double-click the required UI or UI/AO point. The application displays a window to add the device to the database as shown in the following figure.

   Or

   Drag the required UI or UI/AO point and drop it into the Database tab.

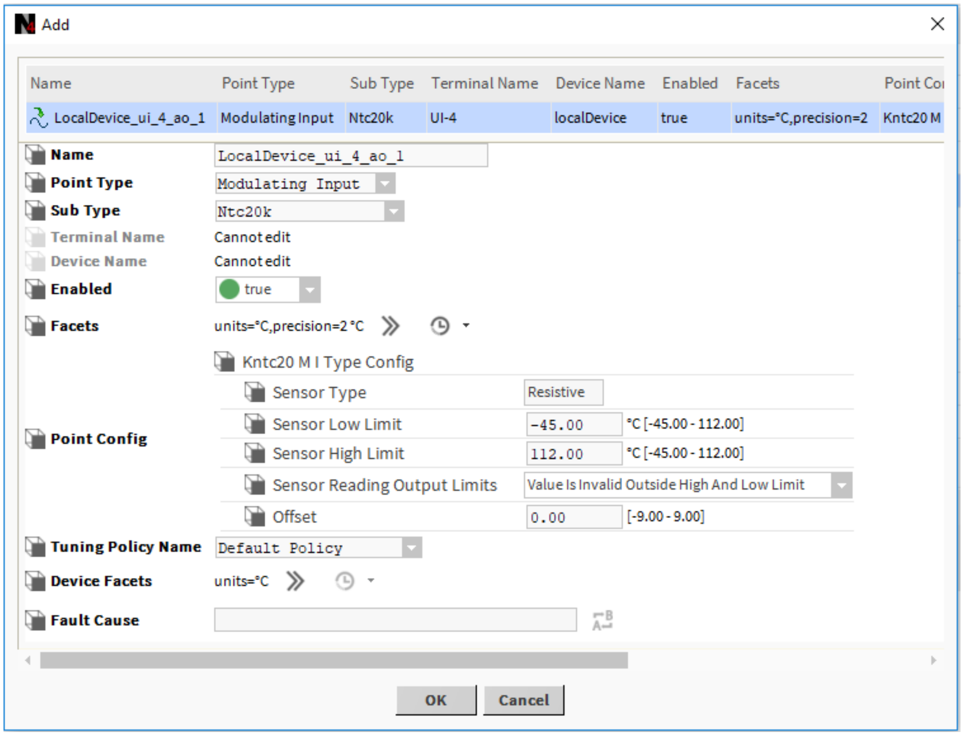



Figure 131: Add Window to Add Device to Database

3. Select Point Type as Modulating Input.
4. Select the Sub Type as Ntc20k.

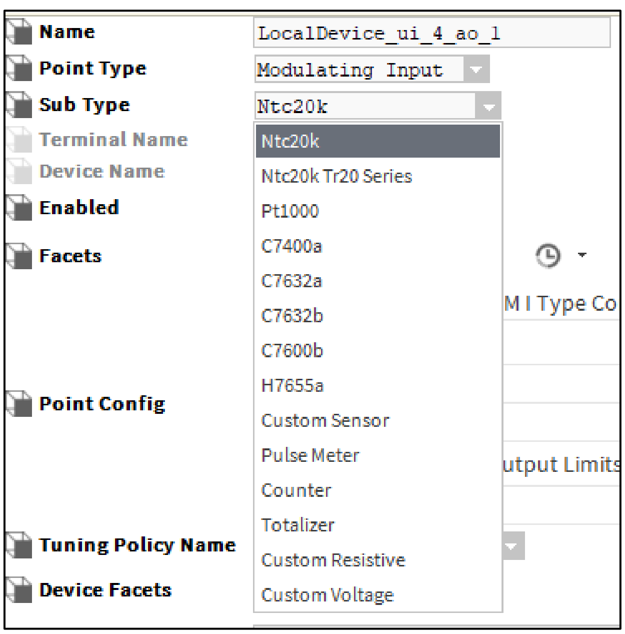



Figure 132: Selecting Ntc20k from Sub Type Drop-Down Menu

5. Configure the Point Config, Tuning Policy Name, Device Facets, and Fault Cause properties as required.
6. Click Ok.

## Configuring UI or UI/AO as Pt1000

Pt1000 is a temperature sensor of platinum resistance thermometer type.

To define Pt1000:

1. Discover the CIPer Model 30 I/O points.
2. Double-click the required UI or UI/AO point. The application displays a window to add the device to the database as shown in the following figure.

   Or

   Drag the required UI or UI/AO point and drop it into the Database tab.

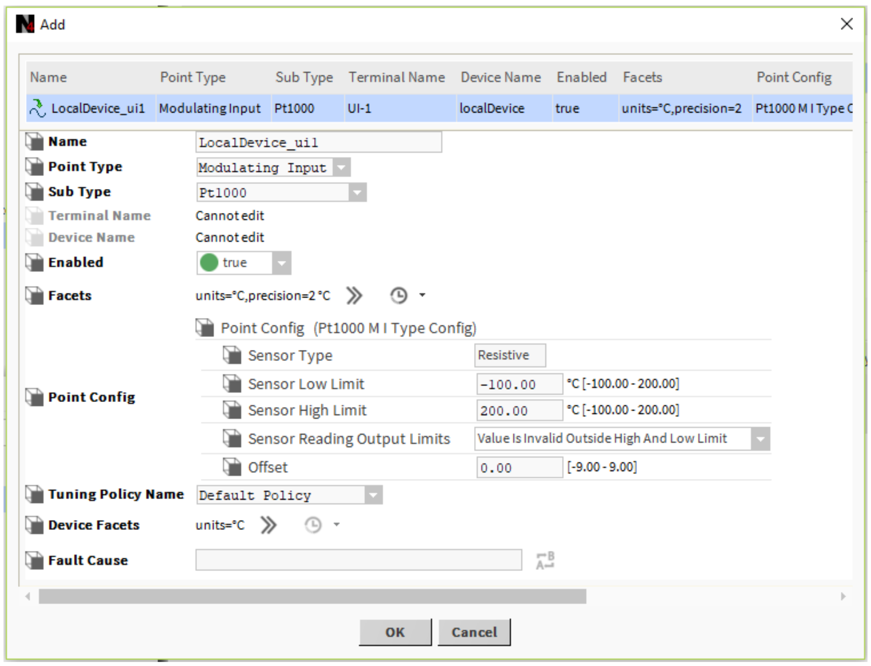



Figure 133: Add Window to Add Device to Database

3. Select Point Type as Modulating Input.
4. Select the Sub Type as Pt1000.

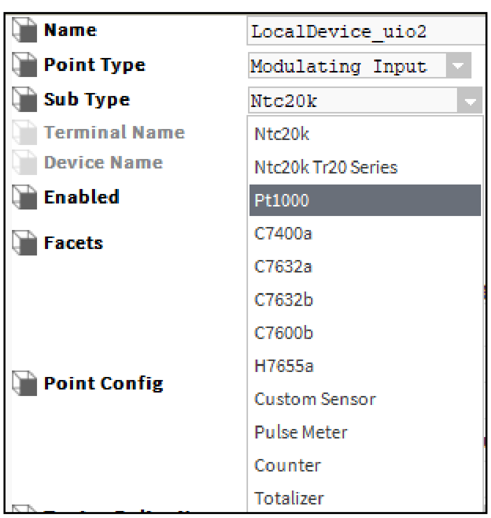



Figure 134: Selecting Pt1000 from Sub Type Drop-Down Menu

5. Configure the Point Config, Tuning Policy Name, Device Facets, and Fault Cause properties as required.
6. Click Ok.

## Configuring UI or UI/AO as Custom Resistive

A resistive sensor is a transducer or electromechanical device that converts a mechanical change such as displacement into an electrical signal that can be monitored after conditioning.

To define Resistive sensor:

1. Discover the CIPer Model 30 I/O points.
2. Double-click the required UI or UI/AO point. The application displays a window to add the device to the database as shown in the following figure.

   Or

   Drag the required UI or UI/AO point and drop it into the Database tab.

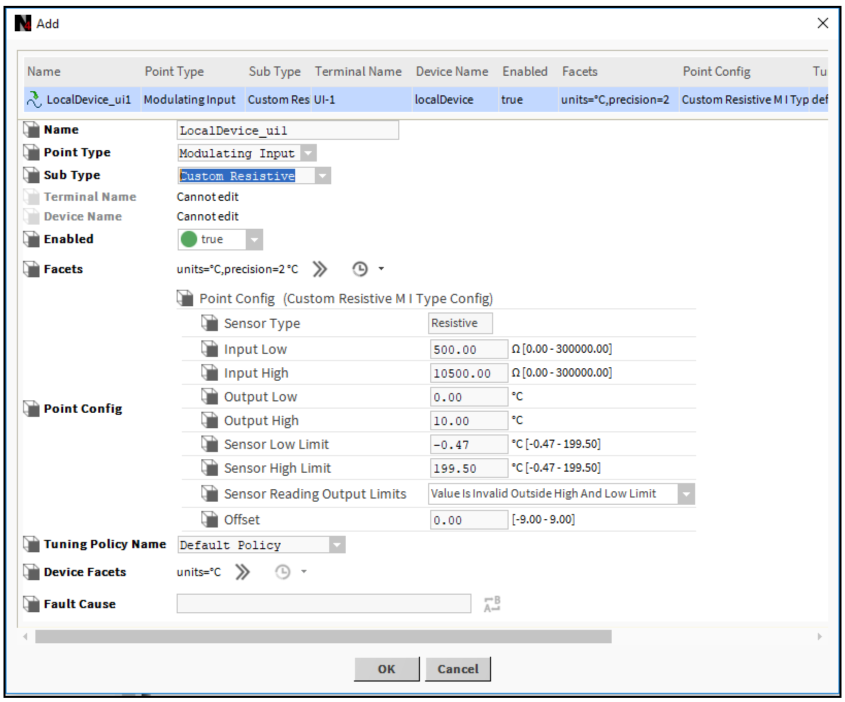



Figure 135: Add Window to Add Device to Database

3. Select Point Type as Modulating Input.
4. Select the Sub Type as Custom Resistive.

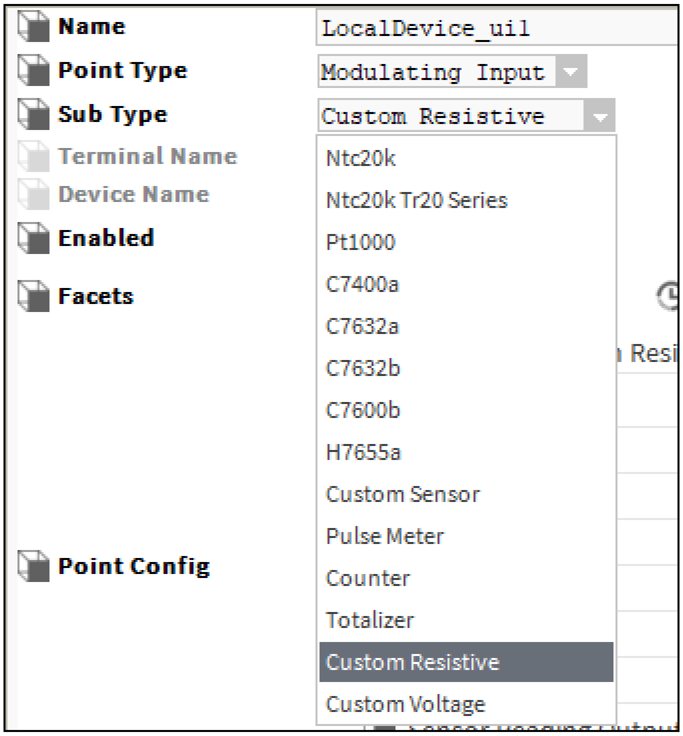



Figure 136: Selecting Custom Resistive from Sub Type Drop-Down Menu

5. Enter the value for Input Low, Input High, Output Low, Output High, Sensor Low Limit, Sensor High Limit, Sensor Reading Output Limits & Offset as required.
6. Configure the Point Config, Tuning Policy Name, Device Facets, and Fault Cause properties as required.
7. Click Ok.

## Configuring Built-In Flow Sensor

The built-in flow sensor is always assigned to the Universal Input 0 of the VAV controller. Built-in flow sensor is only available for the models which have pressure tubes to measure velocity pressure. These models are mostly used for VAV applications.

To add and configure a built-in flow sensor:

1. Discover the CIPer Model 30 I/O points.
2. Double-click the Flow Sensor UI Point. The application displays a window to add the device to the database.
3. Enter the sensor low and high limit values in the Sensor Low Limit and Sensor High Limit respectively as per Sensor Data Sheet.
4. Select the value for Sensor Reading Output Limits from the respective drop-down menus. Sensor Readings Outside Limit,
   - If the Value is INVALID outside High Limit option is selected and when output crosses the limit, the output becomes invalid.
   - If the Clamp Value as High and Low Limits option is selected and if output crosses the high limit or low limit, the output is clamped to Low or High Limit and it doesn't become invalid
5. Click OK to complete adding a built-in flow sensor.

## Configuring UI or UI/AO as Binary Inputs

You can configure the Binary Input blocks and use them while adding the discovered physical UI (universal input) or UI/AO (Universal Input Output) points into the database.

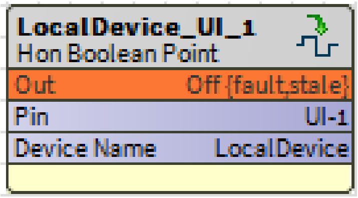



Figure 137: Binary Input Function Block

To add and configure a binary input block:

1. Discover the CIPer Model 30 I/O points.
2. Double-click the required UI / UI/AO point. The application displays a window to add the device to the data-base as shown in the following figure

   Or

   Drag the required UI / UI/AO point and drop it into the Database tab.
3. Select Point Type as Binary Input.

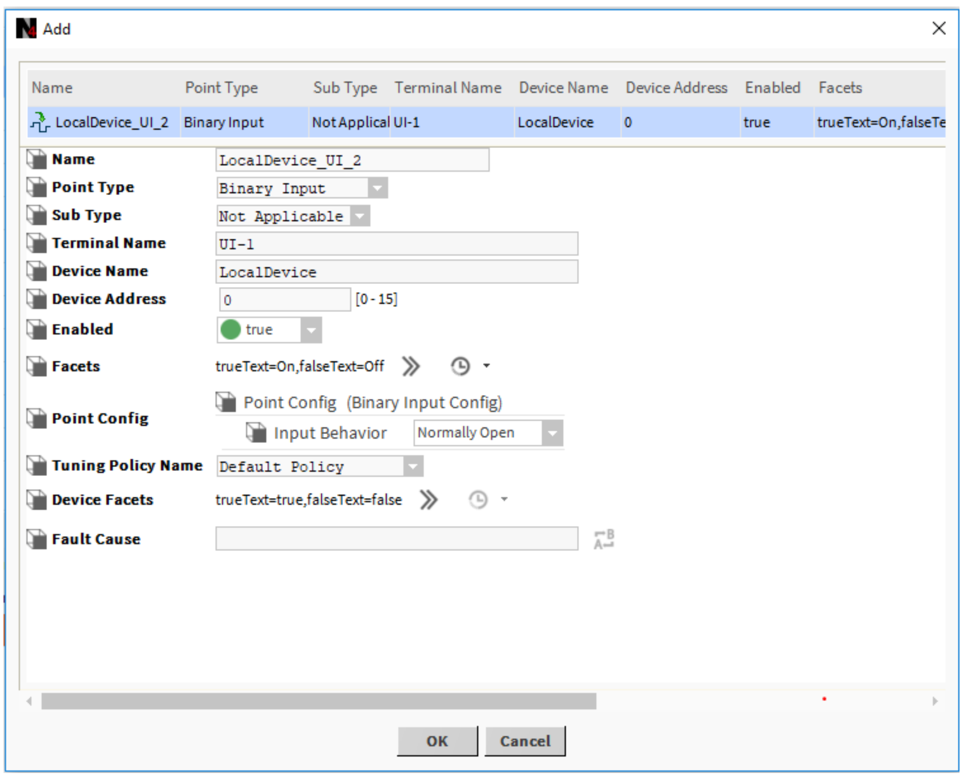
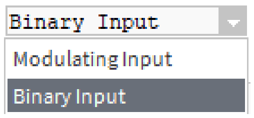
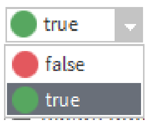
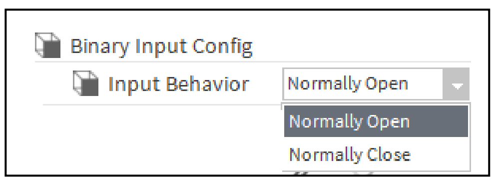



Figure 138: Adding Binary Input to Database

The following table describes the configuration details for a binary input.

Table 17: Configuration of Binary Input Properties

| Name | Definition |
|---|---|
| Name | Enter the name of the physical point or use the default names given by the application. |
| Point Type | Select Binary Input from Point Type drop-down menu.<br><br> |
| Sub Type | Not applicable |
| Enable | Select True to enable the selected point.<br><br> |
| Facets | Click . Change the values for trueText & falseText to match the requirement. For example, for Supply Fan status, the trueText can be On and falseText can be Off. |
| Point Configuration | • Normally Open<br>• Normally Close<br><br> |

4. Click Ok to save the information updated.

   Or

   Click Cancel, if you do not want to save the changes.

## Configuring UI/AO as Modulating Outputs

You can configure the ModulatingOutput blocks and use them while adding the discovered physical UI / UI/AO points into database.

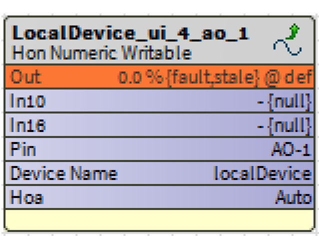



Figure 139: UI/AO configuration as Modulating Output

To add and configure a modulating output block:

1. Discover the CIPer Model 30 I/O points.

2. Double-click the required UI/AO point. The application displays a window to add the device to the database as shown in the following figure.

   Or

   Drag the required UI/AO point and drop it into the Database tab.

3. Select Point Type as Modulating Output from the Point Type drop-down menu.

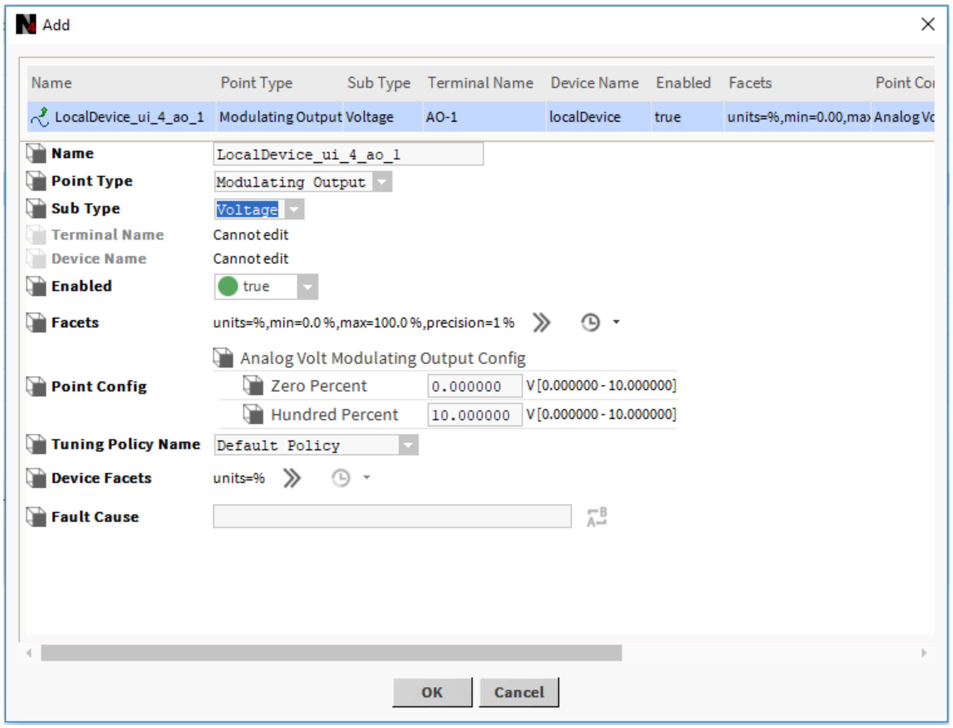
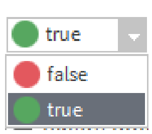



Figure 140: Adding Modulating Output to Database

The following table describes the configuration properties and their definitions

Table 18: Configuration of Modulating Output Properties

| Name | Definition |
|---|---|
| Point Name | Enter the name of the function block or use the default names given by the application. |
| Point Type | Select Modulating Output from Point Type drop-down menu. |
| Sub Type | Default value is Voltage. You can change it to current. Based on the sub type selection the units in the Point Config section changes. |
| Enable | Select True to enable the selected point.   |
| Facets | Click  to view the details of the facets for the network/internal data type.<br>The following information can be configured: |

| Name | Definition |
|---|---|
| | • Minimum: The minimum limit for selected unit<br>• Maximum: The maximum limit for selected unit<br>Or<br>• Range: Indicates the possible enumeration with its ordinal for a selected unit<br>• Units: Indicates the unit symbol for the selected unit (if it shows null, it means the unit symbol is not applicable there.)<br>• Resolution: Indicates scaling factor for the selected unit. When a value is written to the controller, the value is divided by the value specified in the Resolution field and when it is read from the controller, it is multiplied by the resolution value before it is displayed in Niagara.<br>Precision: Precision for the selected unit |
| Point Config | As per requirement, select one of following:<br>• Volts: The range is 0 VDC - 10 VDC<br>• Milliamps: The range is 0 mA - 20 mA.<br>Enter the value for Zero Percent and Hundred Percent.<br>Note: Each ModulatingOutput can be configured for the output voltage/current at 0% and at 100%. Each modulating output circuit operates in current mode for loads up to 600 $\Omega$. For loads of 600 $\Omega$ to 1000 $\Omega$, the output transitions to voltage mode. For loads above 1000 $\Omega$, the output operates in voltage.<br>When full percent is less than zero percent, the motor runs in the reverse direction. |
| OK | Saves the entered information and exits the dialog box. |
| Cancel | Exits the dialog box. Any information entered is lost. |

**Note:**

When a ModulatingOutput is copied and pasted on wire sheet, then the same configuration is retained. The error message-Terminal Number is duplicate on the device address-is displayed on the IPC Point Manager view of the Points folder. You need to delete the duplicate point manually.

## Configuring DO as Binary Output

You can configure the BinaryOutput blocks and use them while adding the discovered physical DO or UI/AO points into database.

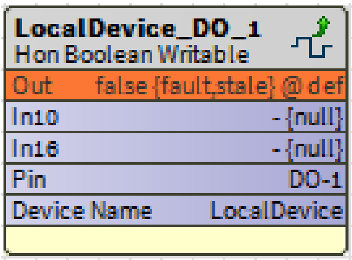



Figure 141: DO Configuration as Binary Output

To add and configure a binary output block:

1. Discover the CIPer Model 30 I/O points.
2. Double-click the required DO point. The application displays a window to add the device to the database as shown in the following figure.

   Or

   Drag the required DO point and drop it into the Database tab.
3. Select Point Type as Binary Output from the Point Type drop-down menu.

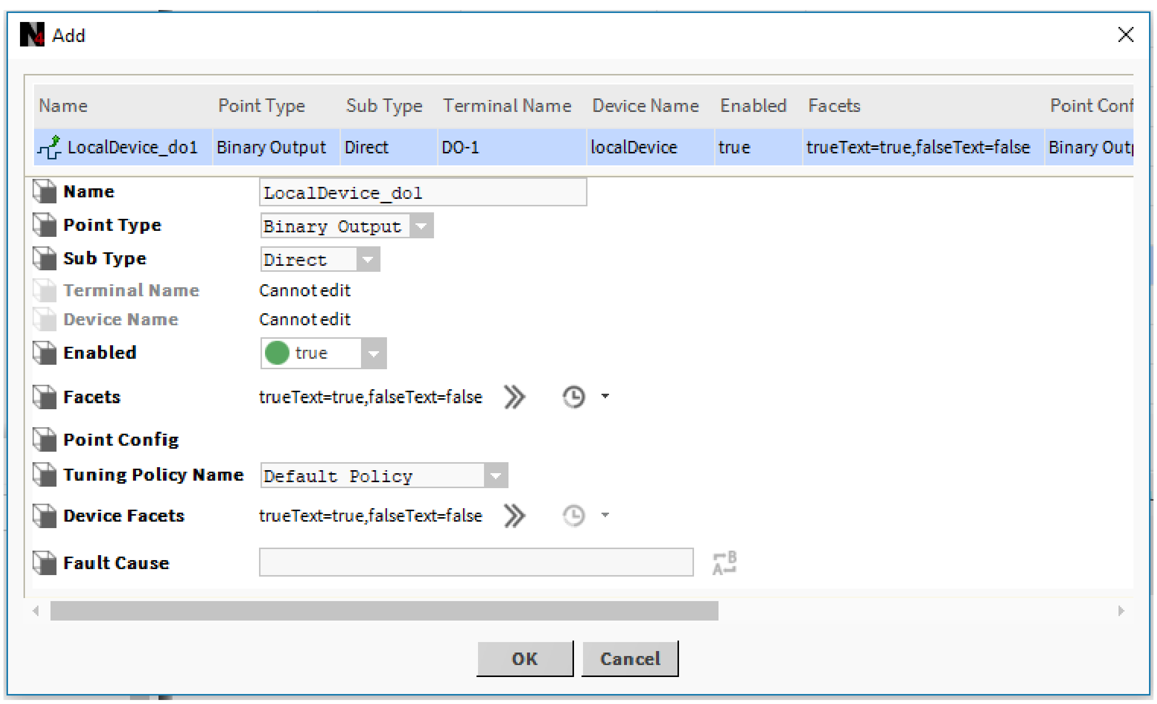



Figure 142: Adding Binary Output to Database

The following table describes the configuration properties and their definitions.

Table 19: Configuration of Binary Output Properties

| Name | Definition |
|---|---|
| Point Name | Enter the name of the function block or use the default names given by the application. |
| Point Type | Select Binary Output from Point Type drop-down menu. |
| Sub Type | Select a Signal type from the Sub Type drop-down menu.<br>• Direct: Select this option if the final control element is accepting the signal in the Constant Width Pulse signal form.<br>• Slow Pwm: Select this option if the final control element is accepting the signal in the Pulse Width Modulation form. |
| Enable | Select True to enable the selected point. |
| Facets | Click .<br>If you select Direct as Sub Type,<br>Change the values for trueText and falseText to match the requirement. For example, for Supply Fan status, the trueText can be On and falseText can be Off.<br>If you select Slow Pwm as Sub Type, the following information can be configured.<br>• Minimum: The minimum limit for selected unit<br>• Maximum: The maximum limit for selected unit<br>Or<br>• Range: Indicates the possible enumeration with its ordinal for a selected unit<br>• Units: Indicates the unit symbol for the selected unit (if it shows null, it means the unit symbol is not applicable there.)<br>• Resolution: Indicates scaling factor for the selected unit. When a value is written to the controller, the value is divided by the value specified in the Resolution field and when it is read from the controller, it is multiplied by the resolution value before it is displayed in Niagara.<br>• Precision: Precision for the selected unit |
| Point Configuration | This is enabled when Slow Pwm is selected in the Type field. Enter the values for the following:<br>• Period: The range is 1 - 3276.7 sec in tenths of seconds. It is the time of a one cycle of the pulse width modulation.<br>• Zero time: When 0% command is given, the pulse width is equal to the value specified in this parameter,<br>• Full time: When 100% command is given, the pulse width is equal to the value specified in this parameter. |

| | |
|---|---|
| OK | Saves the entered information and exits the dialog box. |
| Cancel | Exits the dialog box. Any information entered is lost. |

4. Click Ok to save the information updated.

   Or

   Click Cancel, if you do not want to save the changes.

## Configuring UI/AO as Binary Output

You can configure the BinaryOutput blocks and use them while adding the discovered physical DO or UI/AO points into database.

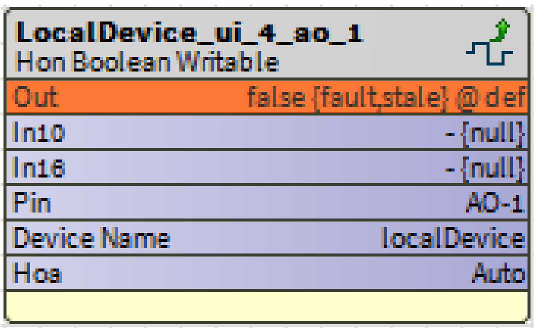



Figure 143: UI/AO Configuration as Binary Output

1. To add and configure a binary output block:
2. Discover the CIPer Model 30 I/O points.
3. Double-click the required UI/AO point. The application displays a window to add the device to the database as shown in the following figure.

   Or

   Drag the required UI/AO point and drop it into the Database tab.
4. Select Point Type as Binary Output from the Point Type drop-down menu.

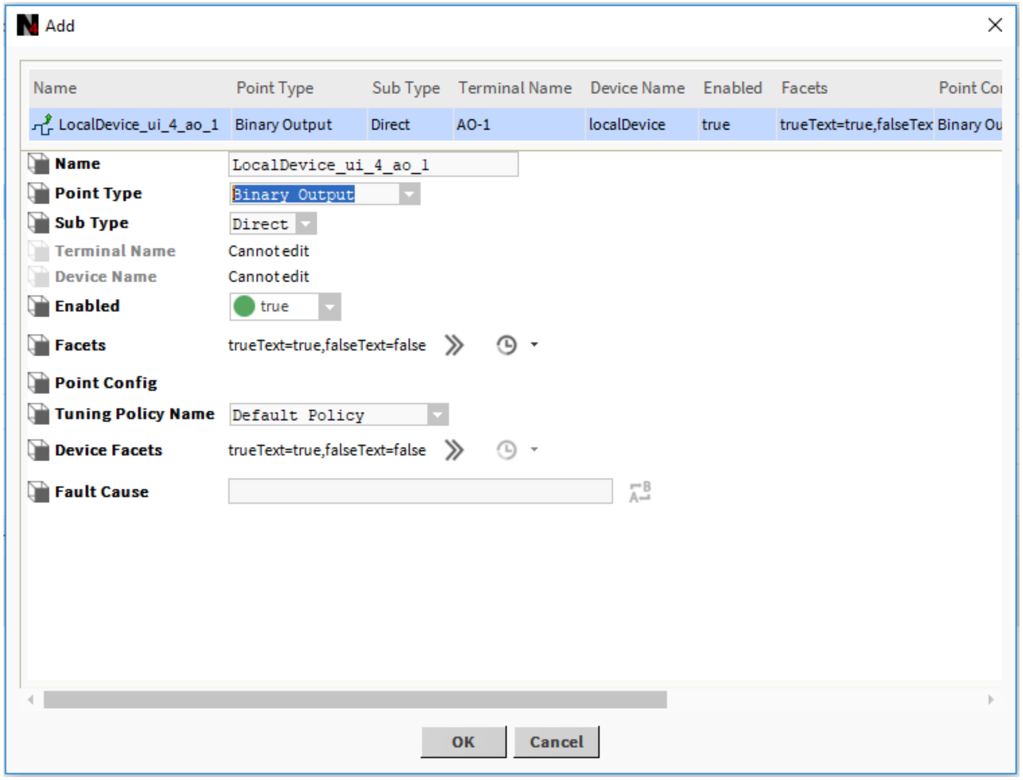



Figure 144: Adding Binary Output to Database

The following table describes the configuration properties and their definitions.

Table 20: Configuration of Binary Output Properties

| Name | Definition |
|---|---|
| Point Name | Enter the name of the function block or use the default names given by the application. |
| Point Type | Select Binary Output from Point Type drop-down menu. |
| Sub Type | Select a Signal type from the Sub Type drop-down menu.<br>• Direct: Select this option if the final control element is accepting the signal in the Constant Width Pulse signal form. |
| Enable | Select True to enable the selected point. |
| Facets | Click .<br>If you select Direct as Sub Type,<br>Change the values for trueText and falseText to match the requirement. For example, for Supply Fan status, the trueText can be On and falseText can be Off. |
| Point Configuration | Not applicable |
| OK | Saves the entered information and exits the dialog box. |
| Cancel | Exits the dialog box. Any information entered is lost. |

5. Click Ok to save the information updated.

   Or

   Click Cancel, if you do not want to save the changes.

## Configuring DO and UI/AO as Floating Output

You can configure the digital output and universal input output as floating output.

### FloatingOutput

Floating: You can use this object when an actuator or a final control element is of Floating type. The CIPer Model 30 controller uses following two BinaryOutputs for floating-type signal:

- One digital output moves the final control element in the clockwise direction
- Other digital output moves the final control element in the anticlockwise direction

The time for which the digital outputs are held on depends upon the stroke time of the final control element.

For example, if the stroke time of the actuator is 90 seconds and the actuator is at full close position, and if 50% command is given to the actuator then the clockwise binary output turns on for 45 seconds. At this position, if again 0% command is given to that actuator then the anticlockwise BinaryOutput turns on for 45 seconds.

In CIPer Model 30 Relay models the resolution of the floating control is limited to 1 second. The SyncEdgeTrigger option is available for CIPer Model 30.

**Note:**

- FloatingOutput block cannot be connected to two different devices.

  For example, FloatingOutput block cannot be used as one in local device and expansion I/O.
- If the FloatingOutput block is used with two different devices, a fault status is shown in the block present on wire sheet.

You can configure the FloatingOutput by adding IO Function Block present in the ipcProgrammingTool palette.

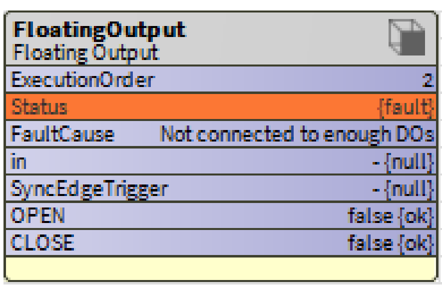



Figure 145: Configuration of Floating Output

To add and configure a floating output block:

1. Click Window > Side Bars > Palette to add the Palette, if it is not visible on the screen.
2. Drag and drop the IOs > FloatingOutput object from the ipcProgrammingTool palette to the wire sheet.

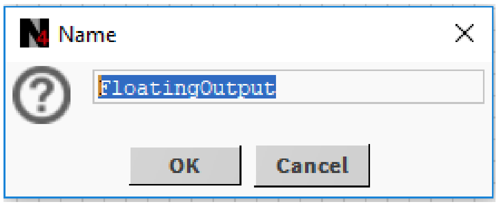



Figure 146: FloatingOutput Name Window

3. Enter the name for the FloatingOutput in the Name window or use the default name set by the application, and then click Ok. The FloatingOutput is added to the Drivers folder.

The following table describes the configuration properties and their definitions.

Table 21: Floating Output Configuration

| Property Name | Definition |
|---|---|
| SyncEdgeTrigger | If SyncEdgeTrigger as the output is selected, it can be triggered the motor point at any point in time using the following values:<br>• <=0: No effect<br>• 1: Sync closed<br>• 2: Sync open<br>• >=3: No effect |
| TravelTime | It is the maximum specified time for which clockwise or counterclockwise output remains ON when 100% command is given. |
| AutoSyncType | Following options are available for the AutoSyncType property:<br>• None: When this option is selected, the CIPer Model 30 controller assumes that the motor is fully closed.<br>• Sync Open: The motor is driven to fully open after the completion of AutoSyncInterval.<br>• Sync Closed: The motor is driven to fully closed open after the completion of AutoSyncInterval. |
| AutoSyncInterval | The auto-synchronization interval is configured from 0 hour to 255 hours in one-hour increments. The timer is loaded and starts counting down after power up reset and power up delay. When the timer expires, the motor is synchronized. This is applicable only if you configure auto-synchronization to Sync Open or Sync Close. |
| PowerupSyncType | Select one of the following values:<br>• None: The CIPer Model 30 controller assumes that the motor is fully closed.<br>• Sync Open: The motor is driven to fully open.<br>• Sync Closed: The motor is driven to fully closed. |
| PowerupDelay | The power-up delay is configured from 0 - 3276.7 seconds in tenths of seconds. Zero (0) means no delay. |

| Property Name | Definition |
|---|---|
| MotorAction | This is enabled only when Floating is selected in the Type field. Select one of the following values:<br>• Direct<br>• Reverse<br>Reverse Action is configured for:<br>• True - 100% = full close, 0% = full open<br>• False is opposite. 0% = full Open, 100% = full close. |

4. Click Save to save the information updated.

   Or

   Click Cancel, if you do not want to save the changes.

### Configuring DO as Slow PWM

You can configure a digital output as Slow Pwm (pulse width modulation).

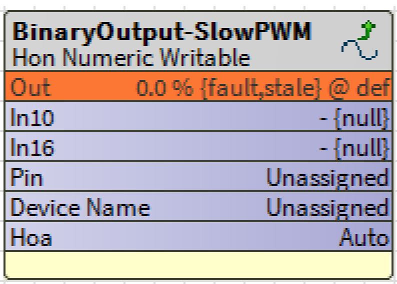



Figure 147: Function Block of BinaryOutput-SlowPWM

To add and configure DO as Slow Pwm:

1. Discover the CIPer Model 30 I/O points.
2. Double-click the required DO point. The application displays a window to add the device to the database as shown in the following figure.

   Or

   Drag the required DO point and drop it into the Database tab.
3. Select Point Type as Binary Output from the Point Type drop-down menu.

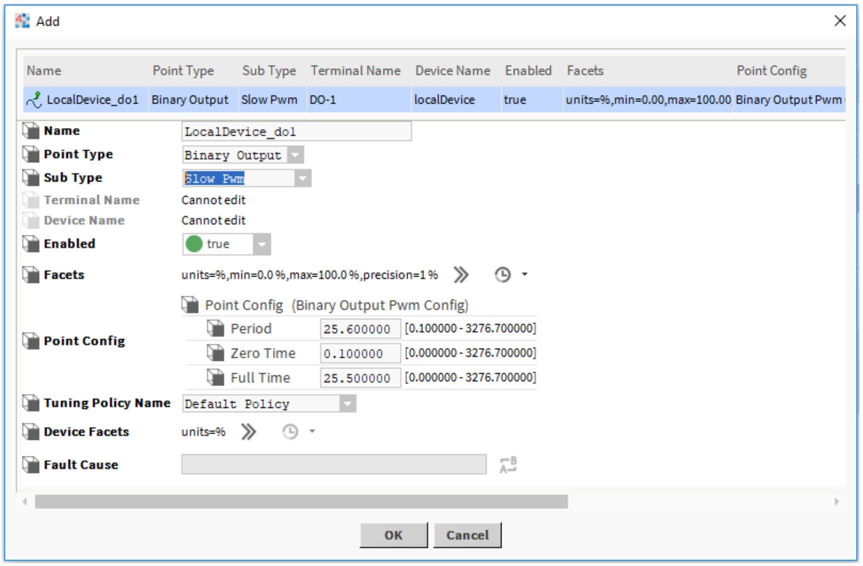



Figure 148: Adding Binary Output (Slow Pwm) to Database

The following table describes the configuration properties and their definitions.

Table 22: Configuration of Binary Output Properties

| Name | Definition |
| --- | --- |
| Point Name | Enter the name of the function block or use the default names given by the application. |
| Point Type | Select Binary Output from Point Type drop-down menu. |
| Sub Type | Select a Signal type from the Sub Type drop-down menu.<br>• Slow Pwm: Select this option if the final control element is accepting the signal in the Pulse Width Modulation form. |
| Enable | Select True to enable the selected point. |
| Facets | Click .<br>If you select Slow Pwm as Sub Type, the following information can be configured.<br>• Units: Indicates the unit symbol for the selected unit (if it shows null, it means the unit symbol is not applicable there.)<br>• Minimum: The minimum limit for selected unit<br>• Maximum: The maximum limit for selected unit<br>• Precision: Precision for the selected unit |
| Point Configuration | This is enabled when Slow Pwm is selected in the Type field. Enter the values for the following:<br>• Period: The range is 1 - 3276.7 sec in tenths of seconds. It is the time of a one cycle of the pulse width modulation.<br>• Zero time: When 0% command is given, the pulse width is equal to the value specified in this parameter,<br>• Full time: When 100% command is given, the pulse width is equal to the value specified in this parameter. |
| OK | Saves the entered information and exits the dialog box. |
| Cancel | Exits the dialog box. Any information entered is lost. |

4. Click Ok to save the information updated.

   Or

   Click Cancel, if you do not want to save the changes.

## Modifying Terminal Assignment Using Property Sheet

You can modify the terminal assignments from the property sheet and remove the assignment for a device and terminal. While modifying the terminal assignment, you can swap across different devices, and within a device you can swap across different terminals.

To modify a terminal:

1. In the Nav tree, browse to Station > Config > Drivers > IPCNetwork > LocalDevice > Points > right click Views > IPC Point Manager.

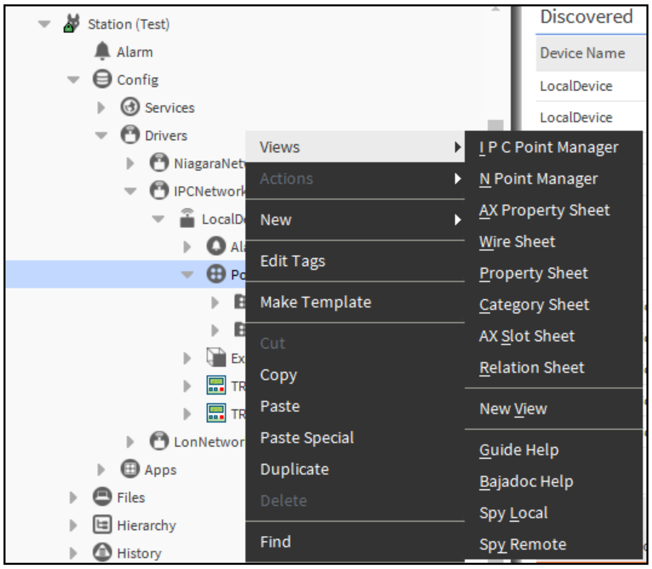
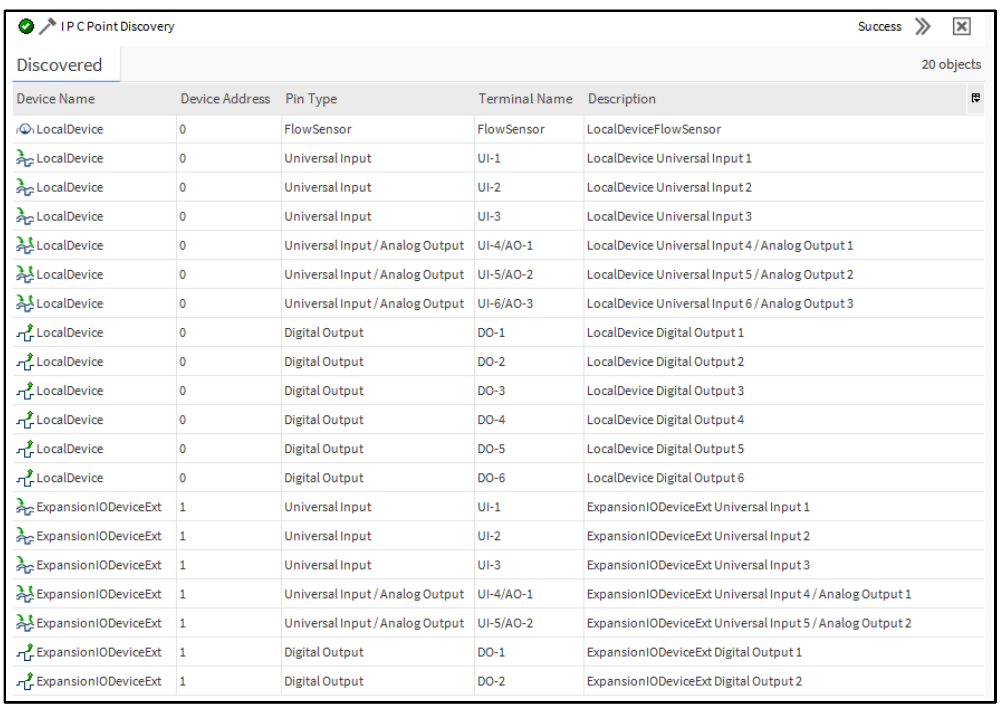



Figure 149: I P C Point Manager Option in Views Menu

The Discovered and Database tabs are displayed as shown in the following figure.




| Device Name | Device Address | Pin Type | Terminal Name | Description |
|---|---|---|---|---|
| LocalDevice | 0 | FlowSensor | FlowSensor | LocalDeviceFlowSensor |
| LocalDevice | 0 | Universal Input | UI-1 | LocalDevice Universal Input 1 |
| LocalDevice | 0 | Universal Input | UI-2 | LocalDevice Universal Input 2 |
| LocalDevice | 0 | Universal Input | UI-3 | LocalDevice Universal Input 3 |
| LocalDevice | 0 | Universal Input / Analog Output | UI-4/AO-1 | LocalDevice Universal Input 4 / Analog Output 1 |
| LocalDevice | 0 | Universal Input / Analog Output | UI-5/AO-2 | LocalDevice Universal Input 5 / Analog Output 2 |
| LocalDevice | 0 | Universal Input / Analog Output | UI-6/AO-3 | LocalDevice Universal Input 6 / Analog Output 3 |
| LocalDevice | 0 | Digital Output | DO-1 | LocalDevice Digital Output 1 |
| LocalDevice | 0 | Digital Output | DO-2 | LocalDevice Digital Output 2 |
| LocalDevice | 0 | Digital Output | DO-3 | LocalDevice Digital Output 3 |
| LocalDevice | 0 | Digital Output | DO-4 | LocalDevice Digital Output 4 |
| LocalDevice | 0 | Digital Output | DO-5 | LocalDevice Digital Output 5 |
| LocalDevice | 0 | Digital Output | DO-6 | LocalDevice Digital Output 6 |
| ExpansionIODeviceExt | 1 | Universal Input | UI-1 | ExpansionIODeviceExt Universal Input 1 |
| ExpansionIODeviceExt | 1 | Universal Input | UI-2 | ExpansionIODeviceExt Universal Input 2 |
| ExpansionIODeviceExt | 1 | Universal Input | UI-3 | ExpansionIODeviceExt Universal Input 3 |
| ExpansionIODeviceExt | 1 | Universal Input / Analog Output | UI-4/AO-1 | ExpansionIODeviceExt Universal Input 4 / Analog Output 1 |
| ExpansionIODeviceExt | 1 | Universal Input / Analog Output | UI-5/AO-2 | ExpansionIODeviceExt Universal Input 5 / Analog Output 2 |
| ExpansionIODeviceExt | 1 | Digital Output | DO-1 | ExpansionIODeviceExt Digital Output 1 |
| ExpansionIODeviceExt | 1 | Digital Output | DO-2 | ExpansionIODeviceExt Digital Output 2 |

Figure 150: I P C Point Manager View

2. Click Discover to discover the available points.
3. Add a device to the database by double-clicking the device, or by selecting a device and then clicking Add.
4. Update the Terminal Name property by selecting the required option in the Terminal Name drop-down menu.

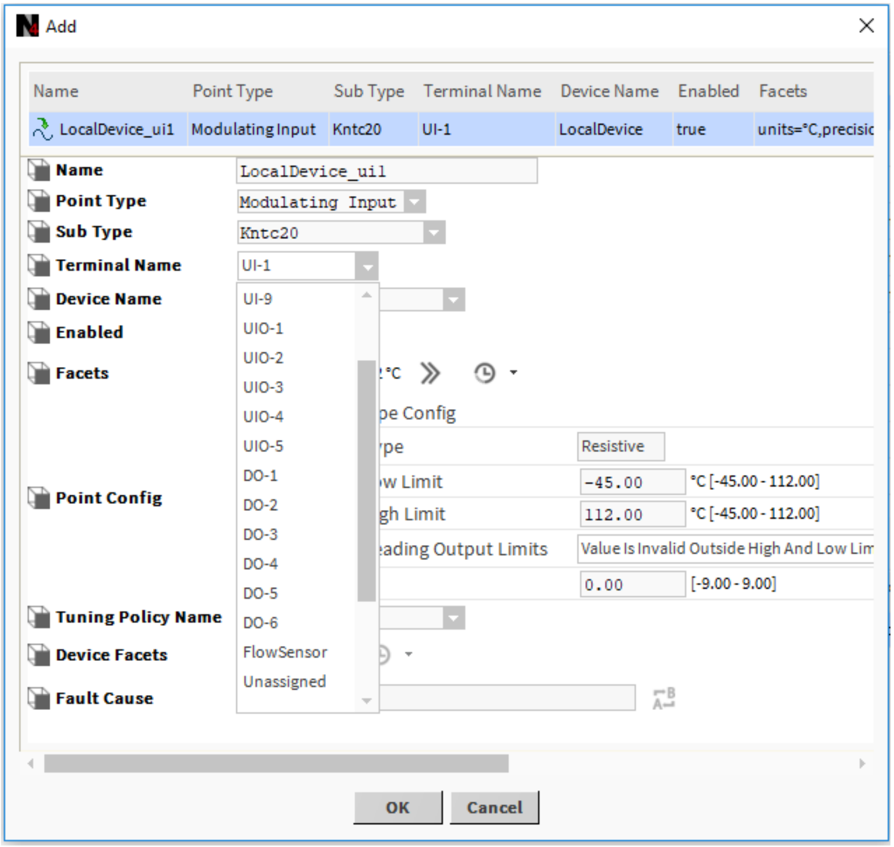



Figure 151: Modifying Terminal Assignment

5. Click Ok to save the changes.

   Or

   Click Cancel, if you do not want to save the changes.

| Note: |
|---|

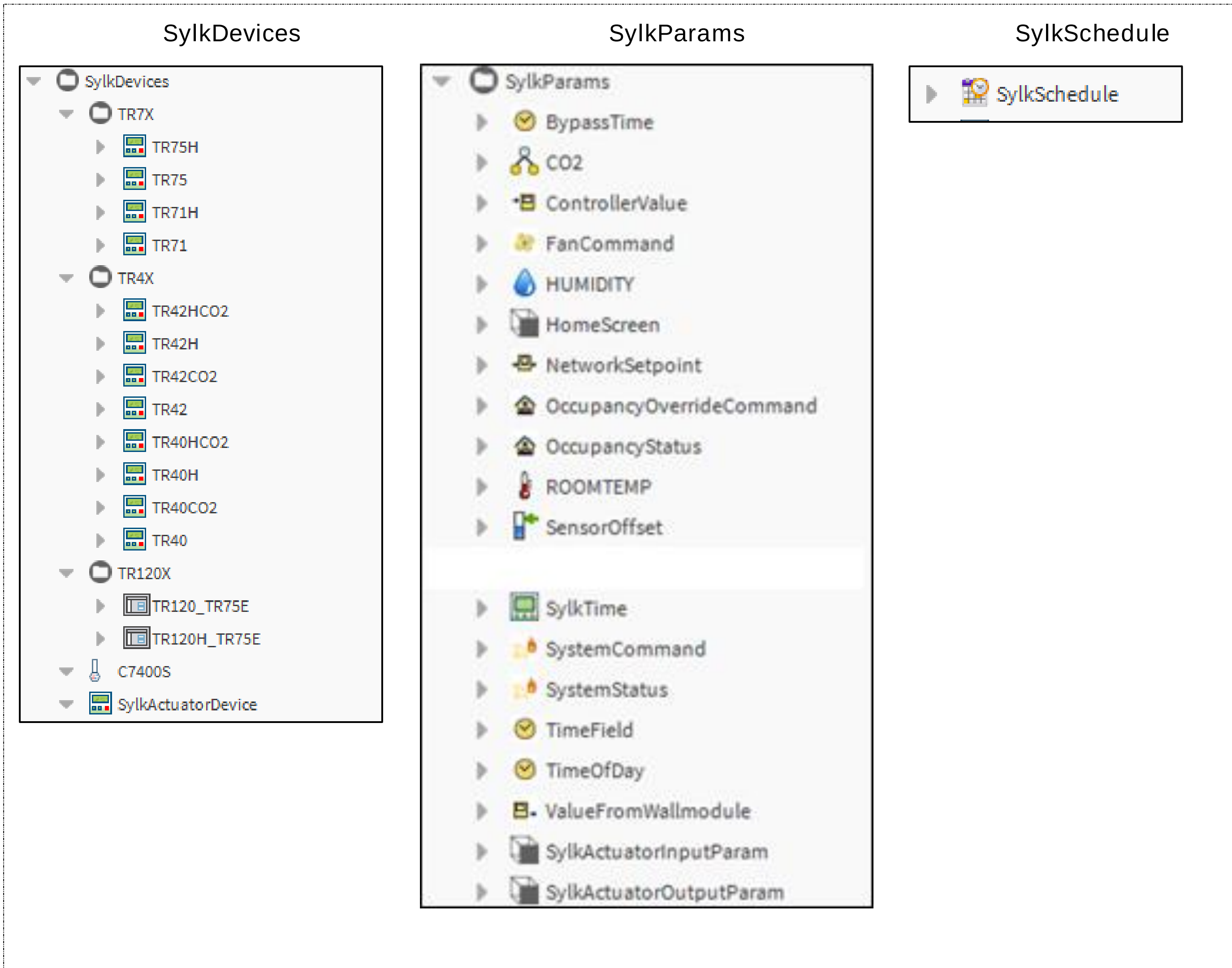
- You can also modify the terminal assignment by:
  - Clicking the Edit button after selecting the required device in the database
  - Double-clicking the required point on the wire sheet and then modifying the Pin property in the property sheet
- To remove the assignment of the terminals and devices, select the Unassigned option in the Terminal Name and Device Name drop-down menus.

# Sylk Device Programming

The CIPer controller family supports Sylk Devices. Using Sylk-enabled sensors saves I/O on the controller and is faster and cheaper to install since only two wires are needed and the bus is polarity insensitive.

CIPer controllers supports Sylk Bus. Sylk is a two wire, polarity insensitive bus that provides both 18 VDC power and communications between a Sylk-enabled sensor and a Sylk-enabled controller. Using Sylk-enabled sensors saves I/O on the controller and is faster and cheaper to install since only two wires are needed and the bus is polarity insensitive. Sylk sensors are configured using the latest release of the of the CIPer Tool for Web-N4™ and WEBStation®

The ipcProgrammingTool palette comes with inbuilt SylkDevices and SylkParams. IPCProgrammingTool palette device contain following Sylk modules:



# Sylk Component Status Behaviors

The status is indicated by text on a colored background. The following table lists the status types, the default colors, and their meaning.

Table 23: Sylk Status Type, Default Colors, and Meaning

| Type | Default Colors, Example | Meaning |
|---|---|---|
| alarm | white text, red background<br>65.0°F | Point currently has a value in an alarm range, as defined by property in its alarm extension. |
| fault | black text, orange background<br>65.0°F | Originates from a proxy point only. Typically indicates a configuration or licensing error. If it occurs after normal operation, it may indicate a "native fault" in device, or the point's parent device has a fault status. |
| overridden | black text, magenta background<br>65.0°F | Current point control is from an action, meaning a user-invoked command at either priority level 8 (override) or priority 1 (emergency). |
| disabled | gray text, light gray background<br>65.0°F | Originates from a proxy point only. Point (or its parent device or network) has been manually disabled (property enabled = false). |
| down | black text, yellow background<br>65.0°F | Originates from a proxy point only. Driver communications to the parent device are currently lost, based upon the device status (Monitor) configuration for that network. |
| stale | black text, tan background<br>65.0°F | Originates from a proxy point only. Driver communications have not received a requested response for this data item within the configured times (Tuning period). |
| null | (No color indication) | Current point control has entered a null state, vs. a specific value and priority level. Typical to fallback operation for a writable point.<br>Note: If linking a null status Out to a simple data slot, the *point's null value is processed.* |
| unackedAlarm | (No color indication) | Last point alarm event has not yet received user acknowledgment. Point's alarm extension uses alarm class requiring acknowledgment. |

## License Requirements and Behaviors

The limitation for Sylk devices is as follows:

| Property | Limit | Remarks |
|---|---|---|
| Sylk Device Limitation | Device limit = 14 | If you try to add more than 14 devices, the application shows the device address property as unassigned and status as down. The Commission Sylk Device option when clicked also detects and shows an error, if you try to add more than 14 Sylk devices. |

Following are the behaviors of a Sylk device:

- Sylk device aid you to perform offline configuration.
- You can perform the compilation of the Sylk device.
- You need license to download Sylk device.

# SylkDevices

IPCProgrammingTool helps you to add and configure the Sylk devices. CIPer 30 controller connection limited to 14 Sylk device. If you try to add more than 14 devices, the added device address property sheet shows Address unassigned and Status down.

Also, when you try to perform Commission Sylk Device with more than 14 Sylk device add to controller, it shows an error message.

| | Note: |
|---|---|
| Whenever there is any change in the schedule component (SylkSchedule or EnumSchedule), you need to re-commission the Sylk device. Recommission of the Sylk device enables the tool to send modified value to wall-module. | |

## Adding Sylk Device

To perform action on Sylk device, you need to add the Sylk device to the workstation.

To add Sylk device:

1. Navigate to Station > Config > Drivers > IPCNetwork > LocalDevice and open Wire Sheet view.
2. Navigate to the pallet section select ipcProgrammingTool > SylkDevices.
3. Select the sylk device from the SylkDevices list and drop it to Wire Sheet or drop it under LocalDevices.
4. If you want to add multiple sylk device, hold down Ctrl while clicking right mouse button select multiple sylk device from the SylkDevices list and drop it to Wire Sheet or drop it under LocalDevices.
5. After adding the sylk device to Wire Sheet or under Localdevice. Double-click on the added devices, this action opens property sheet, you can check the power consumption and configure the sylk device.

   Also, user can check if the sylk configuration is downloading status or not.

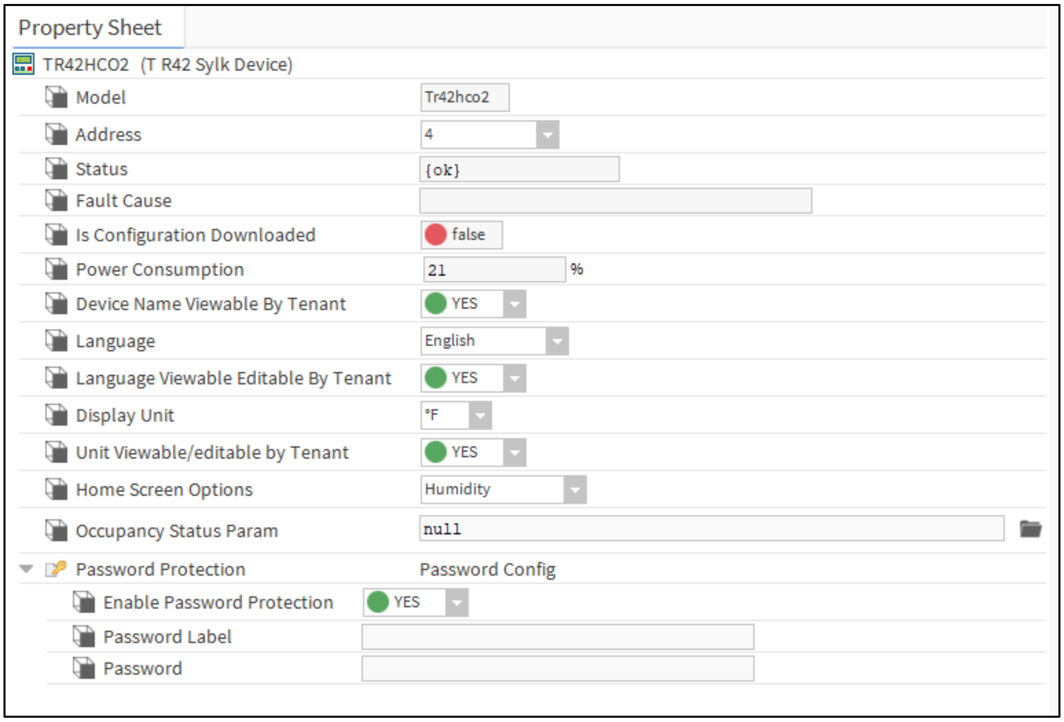



Figure 152: Total Sylk Power Consumption data checking

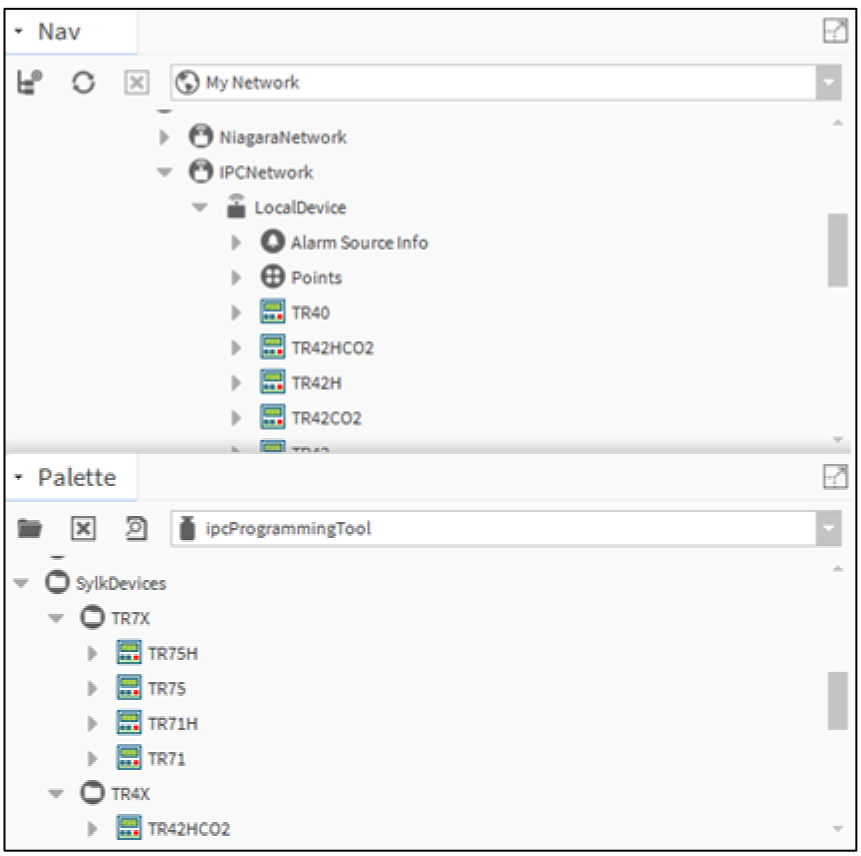



Figure 153: After Adding Sylk Device to Local Device Folder

## Configuring Sylk Device

Once you have added the Sylk device to the LocalDevice folder, you can configure parameters of the Sylk devices.

To configure a Sylk device:

1. Navigate to Station > Config > Drivers > IPCNetwork > LocalDevice and open Wire Sheet view.
2. Navigate to the pallet section, select ipcProgrammingTool > SylkParms.
3. Drag and drop the Sylk parameters modules from the SylkParams into the wire sheet or under the LocalDevice.
4. Double-click on the added Sylk parameters modules. This action opens Property Sheet of the Sylk parameter.

   In the property sheet, you can view the list of sylk device supported by the selected parameter.

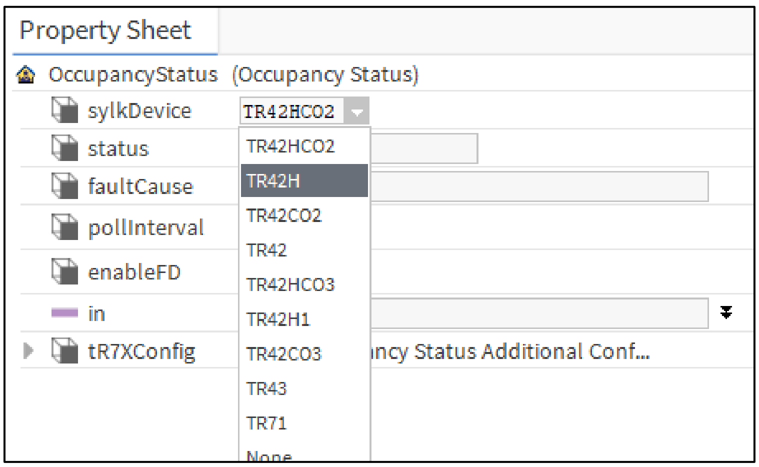



Figure 154: List of supported Sylk devices

5. To view all the Sylk parameters configured for the Sylk device, navigate to the Local Device > Views > select Sylk Parameter Summary.

| Sylk Device Name | Category | Param Name | Slot Path | Type |
|---|---|---|---|---|
| TR42H | Category | BypassTime | slot:/Drivers/IPCNetwork/localDevice/BypassTime | honeywellSylkDevice:BypassTimeParam |
| TR40H | Category | CO2 | slot:/Drivers/IPCNetwork/localDevice/CO2 | honeywellSylkDevice:CO2Param |
| TR71 | Category | ControllerValue | slot:/Drivers/IPCNetwork/localDevice/ControllerValue | honeywellSylkDevice:ValueFromControllerParam |
| TR43 | | FanCommand | slot:/Drivers/IPCNetwork/localDevice/FanCommand | honeywellSylkDevice:FanCommand |
| TR42HCO3 | Category | HUMIDITY | slot:/Drivers/IPCNetwork/localDevice/HUMIDITY | honeywellSylkDevice:HumidityParam |
| TR42 | Category | NetworkSetpoint | slot:/Drivers/IPCNetwork/localDevice/NetworkSetpoint | honeywellSylkDevice:NetworkSetpointParam |
| TR42 | | OccupancyOverrideCommand | slot:/Drivers/IPCNetwork/localDevice/OccupancyOverrideCommand | honeywellSylkDevice:OccupancyOverrideCommand |
| TR42H | | OccupancyStatus | slot:/Drivers/IPCNetwork/localDevice/OccupancyStatus | honeywellSylkDevice:OccupancyStatus |
| TR42H1 | Category | ROOMTEMP | slot:/Drivers/IPCNetwork/localDevice/ROOMTEMP | honeywellSylkDevice:TemperatureParam |
| TR71 | Category | SensorOffset | slot:/Drivers/IPCNetwork/localDevice/SensorOffset | honeywellSylkDevice:SensorOffsetParam |
| TR71 | | SystemCommand | slot:/Drivers/IPCNetwork/localDevice/SystemCommand | honeywellSylkDevice:SystemCommand |
| TR71 | | SystemStatus | slot:/Drivers/IPCNetwork/localDevice/SystemStatus | honeywellSylkDevice:SystemStatus |
| TR71 | Category | TimeField | slot:/Drivers/IPCNetwork/localDevice/TimeField | honeywellSylkDevice:TimeFieldParam |
| TR71 | Category | TimeOfDay | slot:/Drivers/IPCNetwork/localDevice/TimeOfDay | honeywellSylkDevice:TimeOfDayParam |
| TR71 | Category | ValueFromWallmodule | slot:/Drivers/IPCNetwork/localDevice/ValueFromWallmodule | honeywellSylkDevice:ValueFromWallModuleParam |

Figure 155: Sylk Parameter Summary View

For each point that you want to add, a separate configuration is required. You can filter the parameters by category. The category field of a parameter is user-defined.

With respective to workflow of the wall modules in CIPer Model 30 programming model, when you add OccupancyOverrideCommand module to the wire sheet, you need to explicitly add the OccupancyStatus module for the same. That is, in and out are separate.

When you commission Sylk device, the Sylk Device Resource Usage shown under TR7X device and properties get updated.

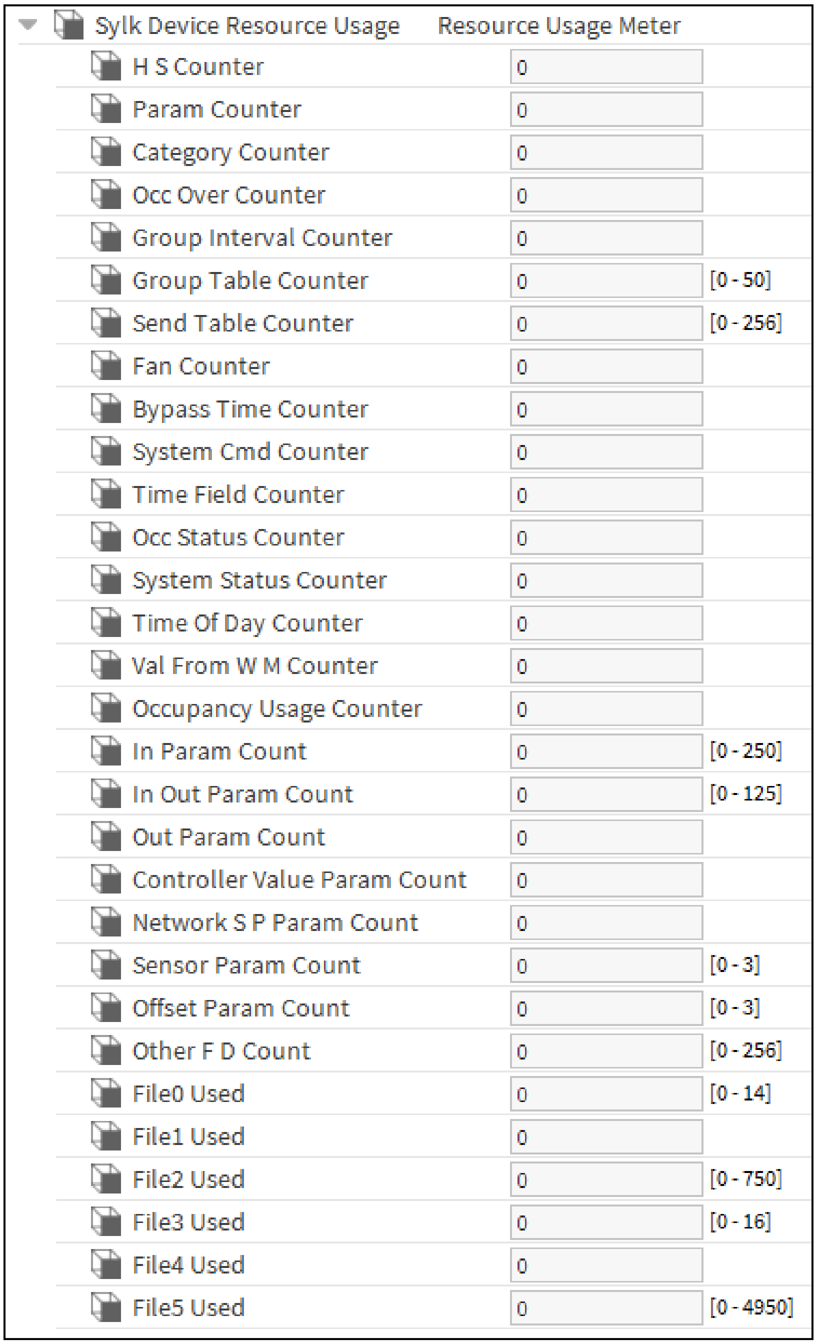



Figure 156: Sub-Menu of Sylk Device Resource Menu

A brief view to the different wall modules in CIPer Model 30 programming model and the parameters that these modules support is as follows:

Table 24: Sylk TR120X Modules and Parameters

| Parameter | Module-TR120X | |
|---|---|---|
| | TR120BusWallModule | TR120HBusWallModule |
| ROOMTEMP | Y | Y |
| HUMIDITY | N | Y |
| OccupancyOverrideCommand | Y | Y |
| ValueFromWallModule | Y | Y |
| TimeOfDay | Y | Y |
| SystemStatus | Y | Y |
| OccupancyStatus | Y | Y |
| ValueFromController | Y | Y |
| SystemCommand | Y | Y |
| TimeField | Y | Y |
| BypassTime | Y | Y |
| SensorOffset | Y | Y |
| HomeScreen | Y | Y |
| NetworkSetpoint | Y | Y |
| SylkTime | Y | Y |
| FanCommand | Y | Y |
| SylkSchedule | Y | Y |

Table 25: Sylk TR7X Modules and Parameters

| Parameter | Module-TR7X | | | |
|---|---|---|---|---|
| | TR75HSBusWallModule | TR75SBusWallModule | TR71HSBusWallModule | TR71SBusWallModule |
| ROOMTEMP | Y | Y | Y | Y |
| HUMIDITY | Y | N | Y | N |
| OccupancyOverrideCommand | Y | Y | Y | Y |
| ValueFromWallModule | Y | Y | Y | Y |
| TimeOfDay | Y | Y | Y | Y |
| SystemStatus | Y | Y | Y | Y |
| OccupancyStatus | Y | Y | Y | Y |
| ValueFromController | Y | Y | Y | Y |
| SystemCommand | Y | Y | Y | Y |
| TimeField | Y | Y | Y | Y |
| BypassTime | Y | Y | Y | Y |
| SensorOffset | Y | Y | Y | Y |
| HomeScreen | Y | Y | Y | Y |
| NetworkSetpoint | Y | Y | Y | Y |
| SylkTime | Y | Y | Y | Y |
| FanCommand | Y | Y | Y | Y |
| SylkSchedule | Y | Y | N | N |

Table 26: Sylk TR42X Modules and Parameters

| Parameter | Module-TR42X | | | |
|---|---|---|---|---|
| | TR42HCO2SBus-WallModule | TR42HSBusWall-Module | TR42CO2SBus-WallModule | TR42SBusWall-Module |
| ROOMTEMP | Y | Y | Y | Y |
| HUMIDITY | Y | Y | N | N |
| CO2 | Y | N | Y | N |
| OccupancyOverrideCom-mand | Y | Y | Y | Y |
| OccupancyStatus | Y | Y | Y | Y |
| BypassTime | Y | Y | Y | Y |
| NetworkSetpoint | Y | Y | Y | Y |
| FanCommand | Y | Y | Y | Y |

Table 27: Sylk TR40X Modules and Parameters

| Parameter | Module-TR40X | | | |
|---|---|---|---|---|
| | TR40HCO2SBus-WallModule | TR40HSBusWall-Module | TR40CO2SBus-WallModule | TR40SBusWall-Module |
| ROOMTEMP | Y | Y | Y | Y |
| HUMIDITY | Y | Y | N | N |
| CO2 | Y | N | Y | N |
| OccupancyOverrideCom-mand | N | N | N | N |
| OccupancyStatus | N | N | N | N |
| BypassTime | N | N | N | N |
| NetworkSetpoint | N | N | N | N |
| FanCommand | N | N | N | N |

Table 28: Sylk Zeleny Modules and Parameters

| Parameter | Module |
|---|---|
| | C7400S (Zeleny) |
| ROOMTEMP | Y |
| HUMIDITY | Y |

Table 29: Sylk Actuator Modules and Parameters

| Parameter | Module |
|---|---|
| | Sylk Actuator |
| SylkActuatorInputParam | Y |
| SylkActuatorOutputParam | Y |

Y: Yes, N: No

**Note:**

- Firmware Details:

  The firmware details available in the property sheet of the LocalDevice node get updated when a station is started up. You can get the firmware details manually by right-clicking the LocalDevice node, and select Actions, and click Ping.

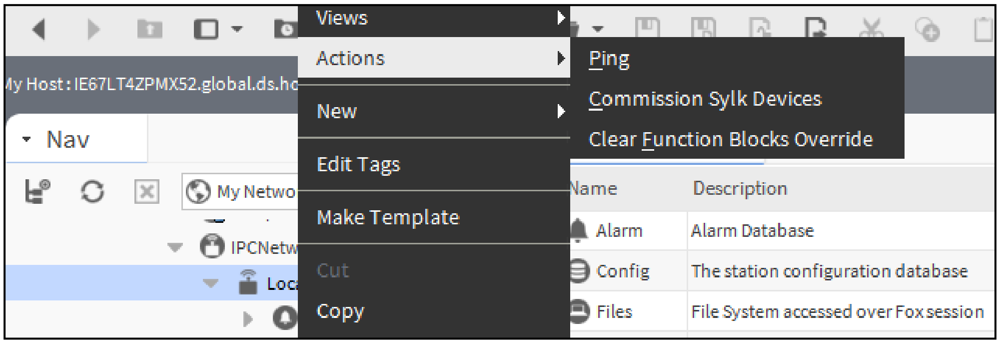



Figure 157: Manually Checking Firmware Details by Ping Action


Figure 158: Ping Action Showing Firmware Details

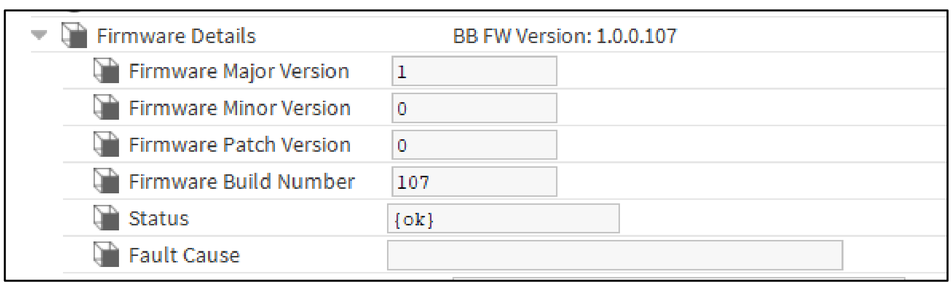



Figure 159: Property Sheet View of LocalDevice Showing Firmware Details Automatically Updated on Station Start up

- Sylk Download Status:

  The flag, Is Sylk Configuration Downloaded, in the property sheet of the LocalDevice node indicates if the Sylk device is downloaded after the configuration is modified. The flag is set to either true or false, and it represents multiple Sylk devices. If you modify any parameters in any Sylk device, the flag turns to false. So, you need to commission the Sylk device to get the modifications done downloaded. On successful commissioning of Sylk device, the flag turns to true. Now you can read and write the Sylk device.

  Is Sylk Configuration Downloaded flag is also present in the individual property sheet of the all the Sylk devices. So, if you modify a property in a Sylk device, the Sylk download status flag for the corresponding device turns to false.

## Deleting Sylk Device

If you want to remove the Sylk Device from a configured station, follow the below steps.

To delete a Sylk Device:

1. Right-click on the required Sylk device from the Nav window.
2. Select Delete.

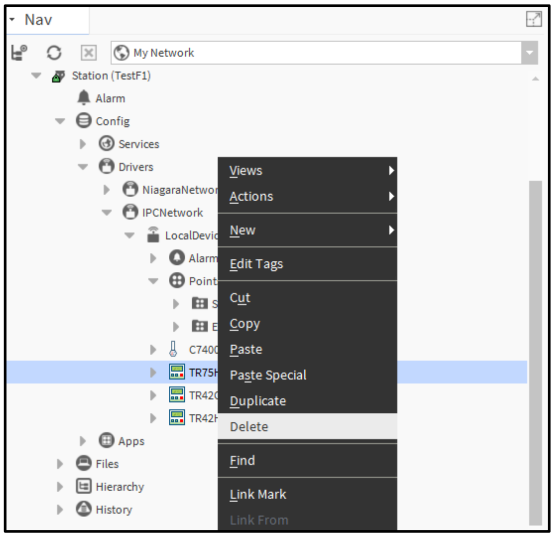



Figure 160: Deleating Sylk Device

| | |
|---|---|
| | Note: |

Whenever you add or delete any Sylk device or modify any Sylk params, effective Sylk device will not perfor read and write function.

This modification will not impact the read and write function of other commissioned Sylk device.

## Validate Sylk Device

Validating Sylk device basically to verify the configuration of added Sylk device. Also, it verifies the any incorrect value entered in parameters or limitation for the configured Sylk device.

You can perform validation for all the added Sylk Devices or single Sylk Device.

To validate all Sylk Device:

1. Navigate Config > Drivers > IPCNetwork and right-click LocalDevice.
2. On command list, select Actions > Validate Sylk Devices.

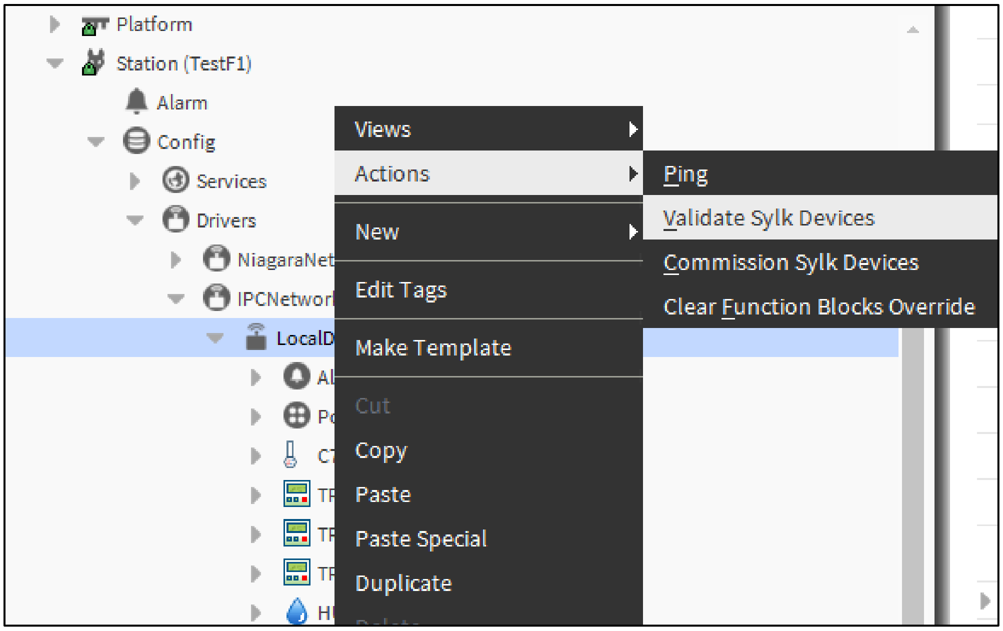



Figure 161: Sylk Devices Validation

To validate single Sylk Device:

1. Navigate Config > Drivers > IPCNetwork > LocalDevice and right-click on Sylk device.
2. On command list, select Actions > Validate Sylk Device.

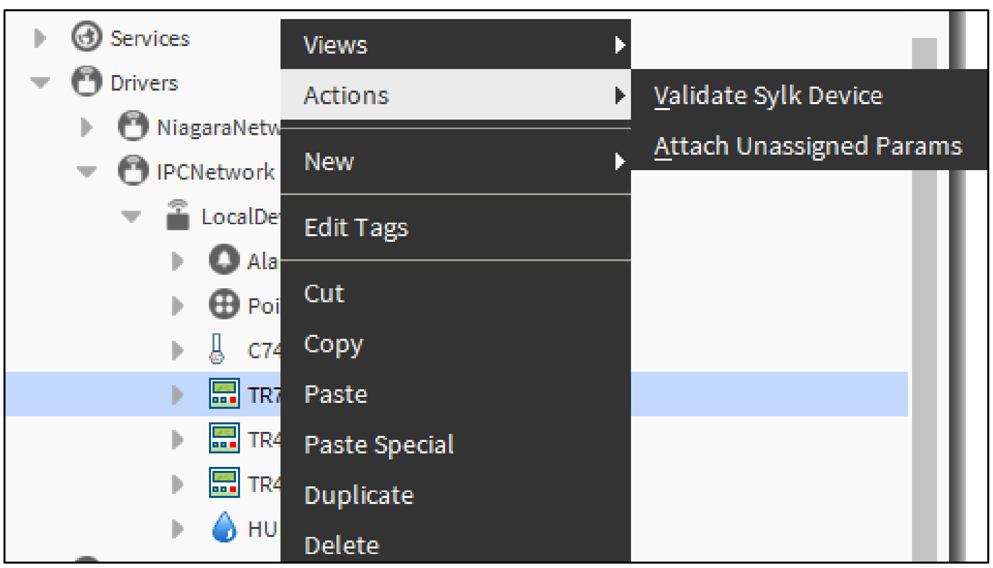



Figure 162: Sylk Device Validation

After you run the Validate Sylk Device, on the Job Log window, you view the status of the configured Sylk Devices.

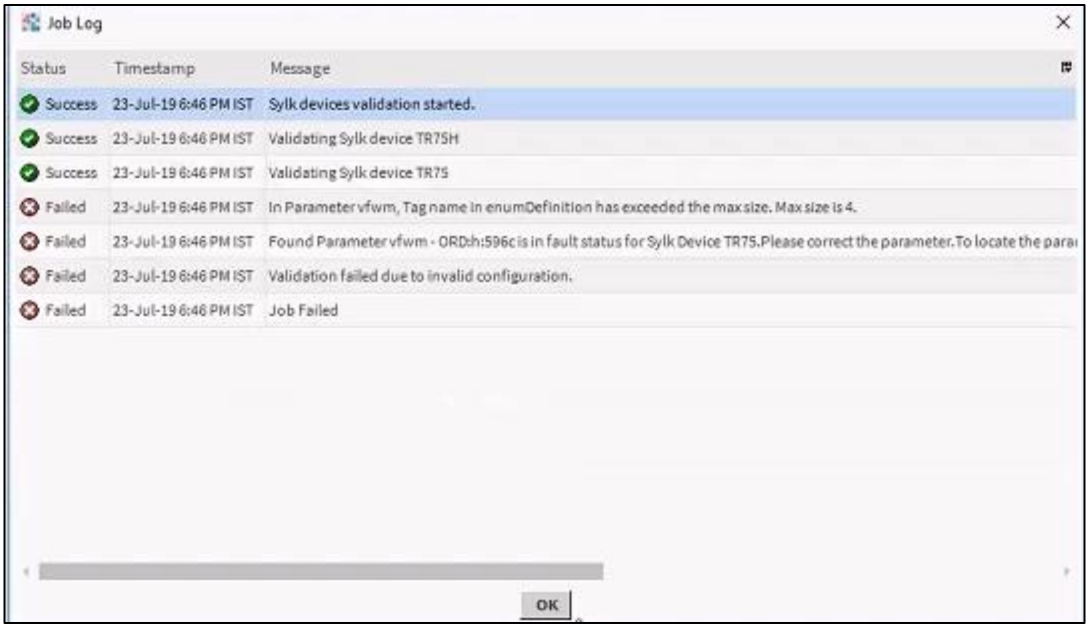


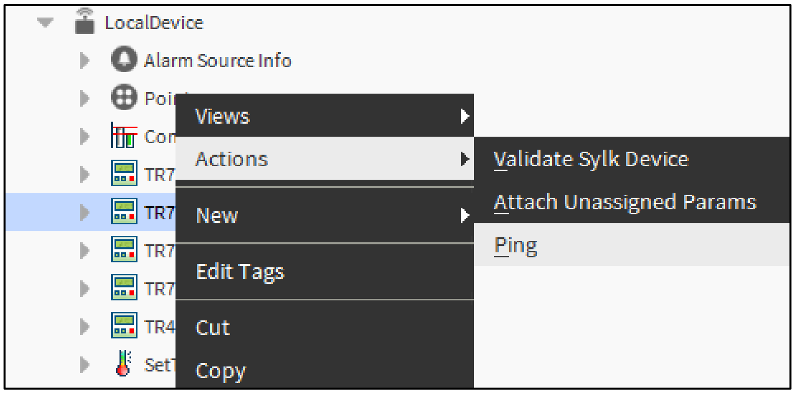

Figure 163: Sylk Device validation status

You can perform this operation in online mode as well as while configuring Sylk devices on workstation in offline mode.

## Sylk Ping

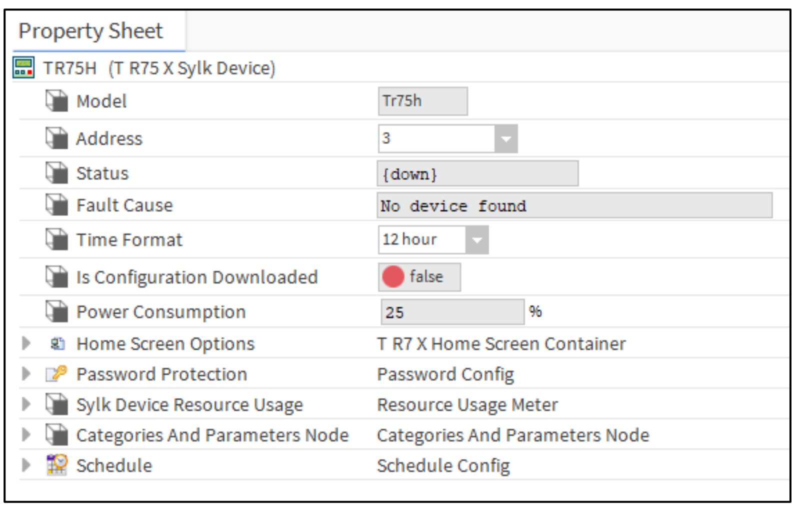
User can perform Sylk ping from a Sylk device or Local device. Sylk ping verifies and shows status based on device connected on the network at the configured address.

To run Sylk Ping, Navigate to the Sylk device in Nav window > Actions > Ping.




- If no device connected at configured address Status displays Down, in this case Fault Cause shows No device found and Sylk Device stops read write operation.

  All Sylk parameters associated with the Sylk device will also be updated with the same status and fault cause
- If user has configured TR75 and connected TR42 at the configured address then Status displays Down, in this case Fault Cause shows Incompatible device found and Sylk Device stops read write operation.
- If the device configured matching with the device connected on network, then Status displays Ok and Sylk Device resumes read write operation.




To perform Sylk Ping for multiple connected device, Navigate to the Local device in Nav window > Actions > Ping Sylk Devices.

| Note: |
|---|
| User can perform ping explicitly from Sylk device or Local device, on Sylk commission auto Sylk ping will be happen on all configured Sylk devices, to verify the compatibility. |

## Basic Sylk Devices

### SylkDevice C7400S Configuration

The C7400S Sylk Bus sensor is a combination temperature and humidity sensor, which is intended to be used in commercial roof top units for sensing air. The sensor is powered by and communicates on the Sylk Bus. The C7400S communicates temperature and humidity information digitally separately on the Sylk Bus Communication Protocol.

To add C7400S:

1. Navigate to the ipcProgrammingTool palette.

   If there is no palette visible on the left pane, on the Menu bar, select Window > Side Bars > Palette. The ipcProgrammingTool palette is displayed.

2. Navigate to IPCNetwork > Local Device.
3. Drag and Drop the C7400S module from the ipcProgrammingTool palette to Local Device in the Nav tree and double-click the Sequenced Control Program or Event Control Program folder to open wire sheet view.
4. Drag and drop the ROOMTEMP and HUMIDITY onto the wire sheet (C7400S supports only Temperature & Humidity Sensor).
5. Enter the name of the sensors and click Ok. The sensors are added and appear on the wire sheet.
6. Double-click the ROOMTEMP sensor on the wire sheet to configure its properties (refer Room Temperature Configuration).

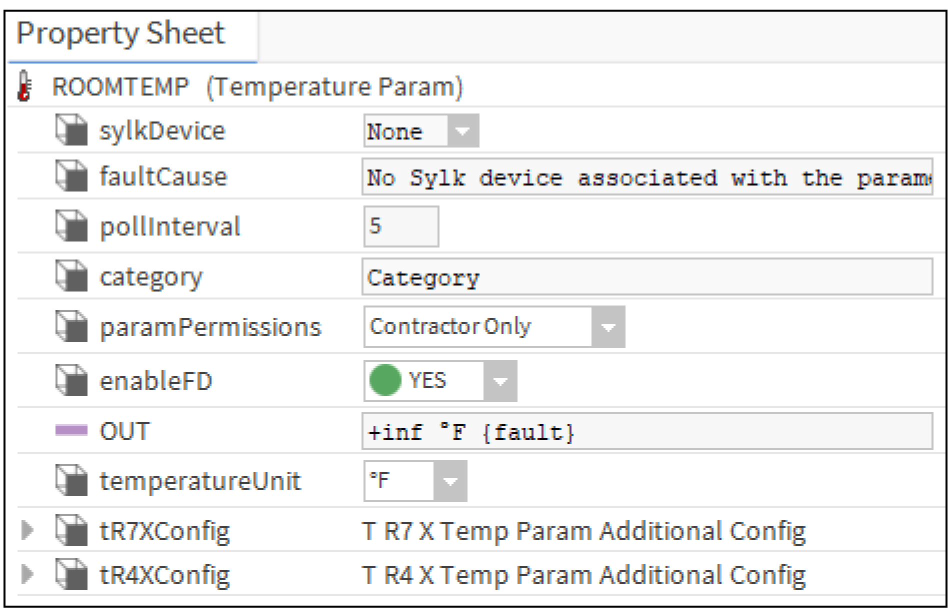



Figure 164: ROOMTEMP Property Sheet

7. Double-click the HUMIDITY sensor on the wire sheet to configure its properties (ref Humidity Configuration).

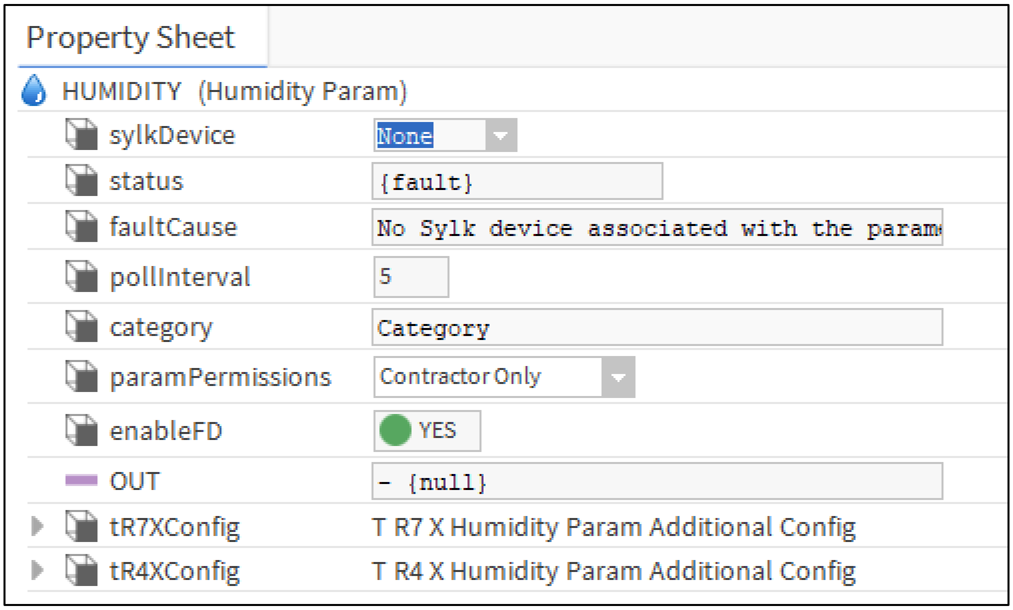



Figure 165: HUMIDITY Property Sheet

8. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

Note:

tR7XConfig and tR4X configuration is not applicable for C7400S sensor.

Table 30: Sylk Zeleny Modules and Parameters

| Parameter | Module |
|---|---|
| | C7400S (Zeleny) |
| ROOMTEMP | Y |
| HUMIDITY | Y |

## SylkDevice TR7X or TR4X Configuration

To add TR7X or TR4X:

1. Navigate to the ipcProgrammingTool palette.

   If there is no palette visible on the left pane, on the Menu bar, select Window > Side Bars > Palette. The ipcProgrammingTool palette is displayed.

2. Navigate to IPCNetwork > Local Device.

9. Drag and Drop the TR7X module from the ipcProgrammingTool palette to Local Device in the Nav tree double-click the Sequenced Control Program or Event Control Program folder to open wire sheet view.

3. Drag and drop the Sylk parameters modules from the SylkParams into the wire sheet or under the LocalDevice.
    - Note: While adding Sylk parameters to the Wiresheet, don't select HomeScreen.
4. Enter the name of the parameters and click OK. The sensors are added, and they appear on the wire sheet.
5. Drag and drop the HomeScreen parameter from the ipcProgrammingTool palette to the supported Sylk device present under Local Device in the Nav tree.

Following is the property sheet of a TR7X Sylk device.

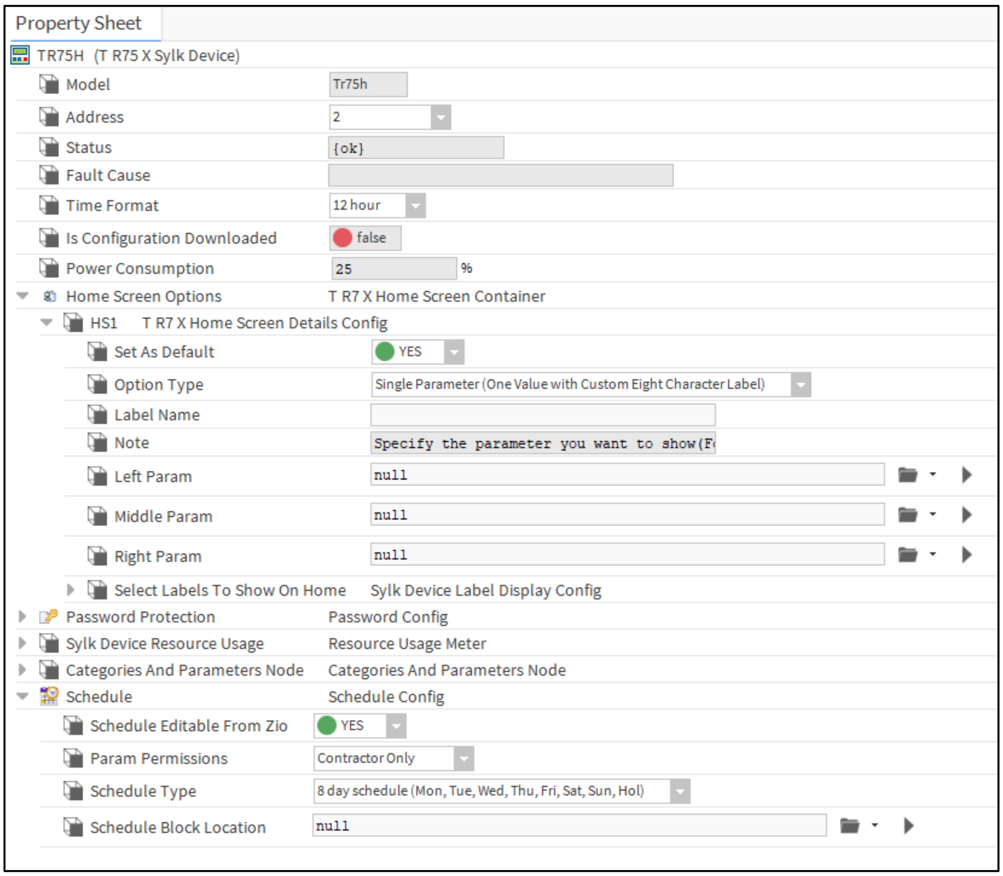



Figure 166: Property Sheet of TR7X Sylk Device

Following is the property sheet of a TR4X Sylk device.

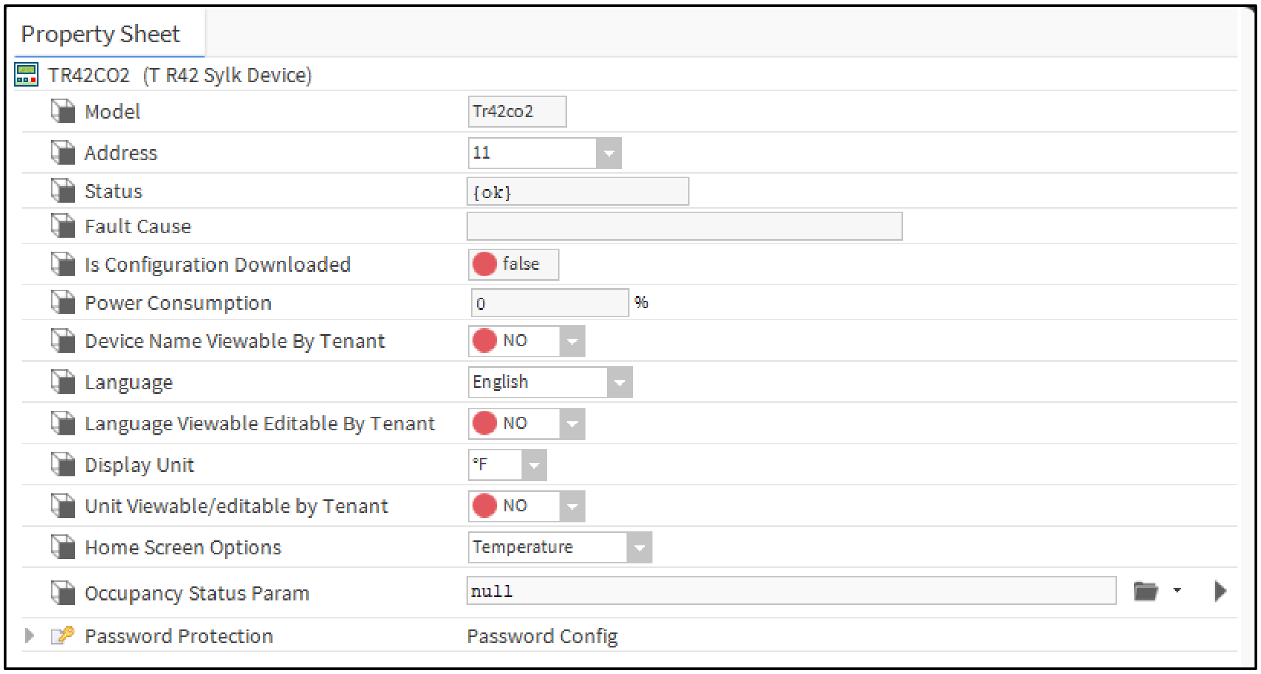



Figure 167: Property Sheet of TR4X Sylk Device

## Sylk Actuator

### Configuration of Sylk Actuator to the CIPer model 30 Controller

MS3103, MS3105, MS4103, MS4105, MS7403, MS7405, MS7503, MS7505, MS8103, MS8105 Spring Return Direct Coupled Actuators (DCA) are used within heating, ventilating, and air-conditioning (HVAC) systems. They can drive a variety of quarter-turn; final control elements requiring spring return fail-safe operation.

Applications includes:

- Volume control dampers, mounted directly to the drive shaft or remotely (with the use of accessory hardware).
- Quarter-turn rotary valves, such as ball or butterfly valves mounted directly to the drive shaft.
- Linear stroke globe or cage valves mounted with linkages to provide linear actuation.
- Available with cable on select models

### Configuration

### Inputs and outputs of the Sylk enabled actuators

As shown in Figure 1, Sylk actuator has two inputs and five outputs. All these inputs and outputs communicate with the CIPer model 30 controllers over a Sylk bus.

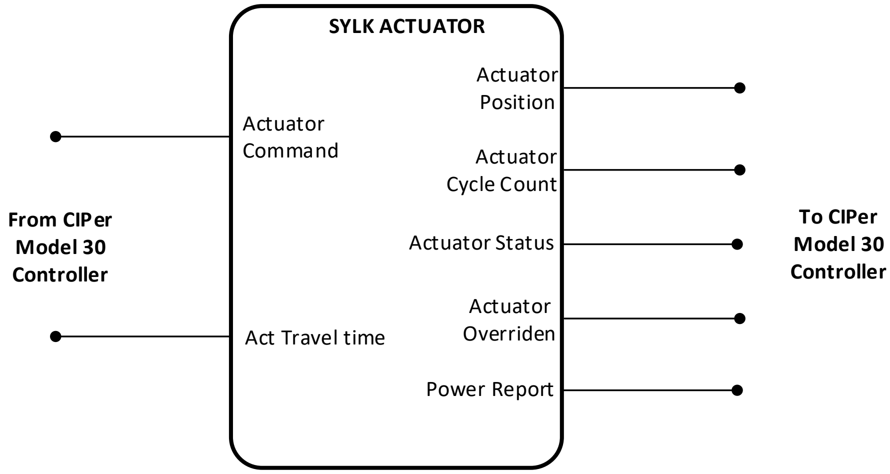


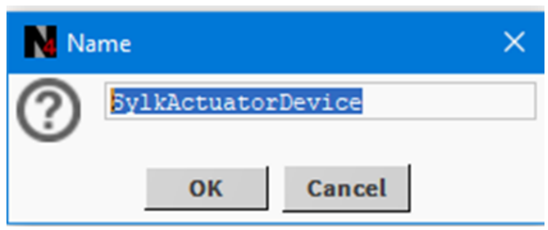

Figure 168: Inputs and Outputs of Sylk enabled actuator

| Inputs | |
|---|---|
| Actuator Command: | Actuator command comes from controller output. An Actuator moves as per the command. |
| Actuator Travel Time: | This is the stroke time or time required to travel from full close position to full open position. |
| **Outputs** | |
| Actuator Position: | This is a feedback from the actuator's current position ranging between 0 to 100 and can be mapped into the CIPer model 30 controller for monitoring. |
| Actuator Cycle Count: | It shows how many time actuators is cycled from open to close position. |
| Actuator Status: | Actuator status returns a number with the following meaning:<br>0- No Error<br>1- Under Voltage<br>2- Over Voltage<br>3- Stall<br>4- OverVoltage and Stall<br>5- UnderVoltage and Stall |
| Actuator Overridden: | It provides actuator overridden status if actuator is overridden externally.<br>This will return "True" if the unit is in Test Mode. This when the address pot is manually turned to position 6 |
| Power Report: | This is a relative measurement. It is only calculated for the last commanded move, value ranging between 0 to 100. |

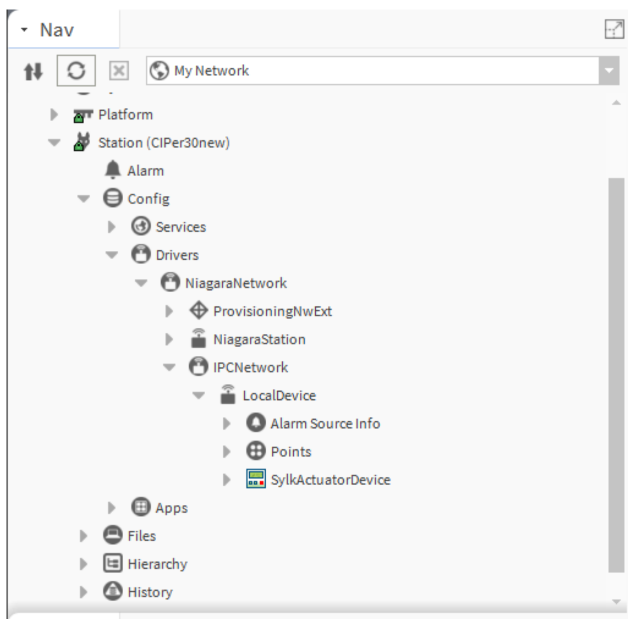
To perform action on Sylk actuator device, you need to add the Sylk actuator device to the workstation.

To add Sylk actuator device:

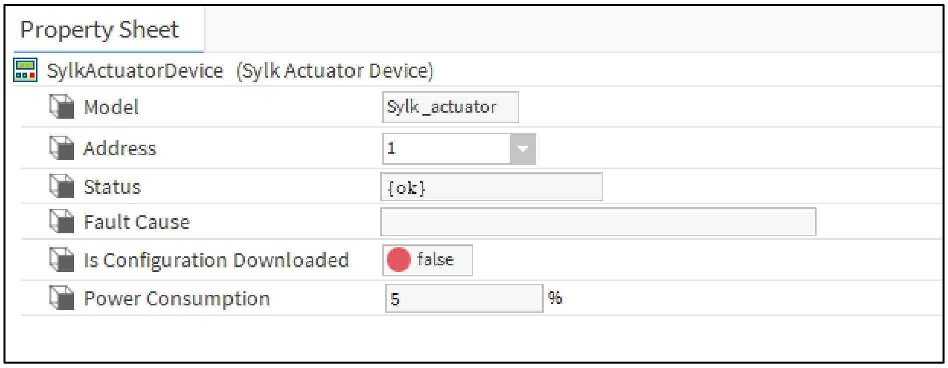
1. Navigate to Station > Config > Drivers > IPCNetwork > LocalDevice > Sequenced Control Program and open Wire Sheet view.
2. Navigate to the palette section select ipcProgrammingTool > SylkDevices.
3. Select the SylkActuatorDevice from the SylkDevices list and drop it to Wire Sheet or drop it under LocalDevice.
4. After adding the SylkActuatorDevice on the network, a window pops up to name the controller as shown in below figure. Name the controller and click on OK button




5. After adding the SylkActuatorDevice, it will appear under the network as shown in the below figure.




6. After adding the SylkActuatorDevice to Wire Sheet or under Localdevice. Double-click on the added devices, this action opens property sheet, you can check the power consumption and configure the sylk actuator device.




Also, user can check if the sylk configuration is downloaded status or not.

Configuring Sylk Device

Once you have added the SylkActuatorDevice to the LocalDevice folder, you can configure parameters of the SylkActuatorDevice.

Parameters Supporting Sylk Actuator

| Sylk Actuator Status Variables | | | |
|---|---|---|---|
| Parameter | SylkZelix | DiamondSylk | SylkZeus |
| Actuator Position | Y | Y | Y |
| Actuator Cycle Count | Y | Y | Y |
| Actuator Status | Y | Y | Y |
| Actuator Overridden | Y | Y | Y |
| Power Report | Y | N | N |
| Sylk Actuator Control Variable | | | |
| Actuator Timing | Y | Y | Y |
| Actuator Position Request | Y | Y | Y |

Configuring Sylk actuator inputs

1. Navigate to Station > Config > Drivers > IPCNetwork > LocalDevice > Sequenced Control Program and open Wire Sheet view.
2. Navigate to the palette section, select ipcProgrammingTool > SylkParms.
3. Drag and drop the Sylk parameters modules from the SylkActuatorInputParam into the wire sheet or under the LocalDevice > Sequenced Control Program.
4. Double-click on the SylkActuatorInputParam modules. This action opens Property Sheet of the Sylk parameter.
5. In the SylkActuator Input Param Property Sheet, select SylkDevice as SylkActuatorDevice and inputType as Actuator or Act Travel Time from drop down list.

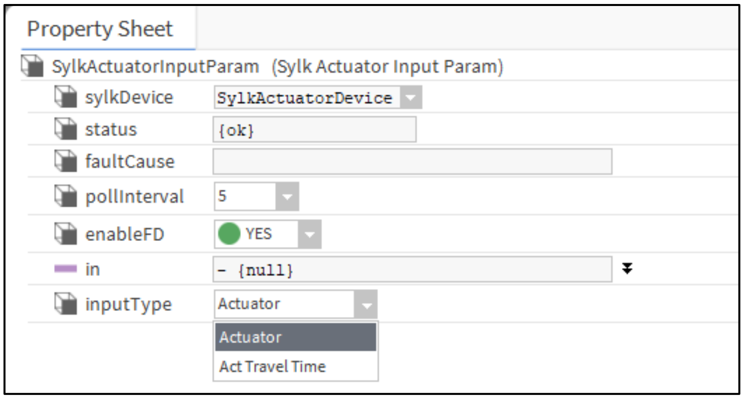



Figure 169: Configuring properties of SylkActutorInputParam

Configuring Sylk actuator output

1. Navigate to Station > Config > Drivers > IPCNetwork > LocalDevice > Sequenced Control Program and open Wire Sheet view.
2. Navigate to the pallet section, select ipcProgrammingTool > SylkParms.
3. Drag and drop the Sylk parameters modules from the SylkActuatorOutputParam into the wire sheet or under the LocalDevice > Sequenced Control Program.
4. Double-click on the SylkActuatorOutputParam modules. This action opens Property Sheet of the Sylk parameter.
5. In the SylkActuator Output Param Property Sheet, select SylkDevice as SylkActuatorDevice and outputType as Actuator Position / Actuator Cycle Count / Actuator Status / Actuator Overridden or Power Report from drop down list.

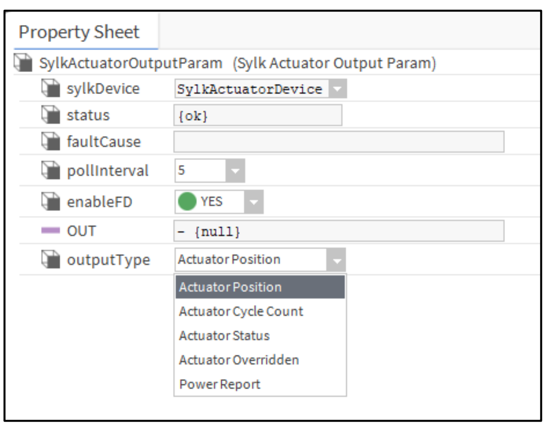



Figure 170: Configuring properties of SylkActutorOutputParam

| Note: |
| --- |
| For adding Input /Output point type for Sylk Actuator, a separate parameter configuration required. |

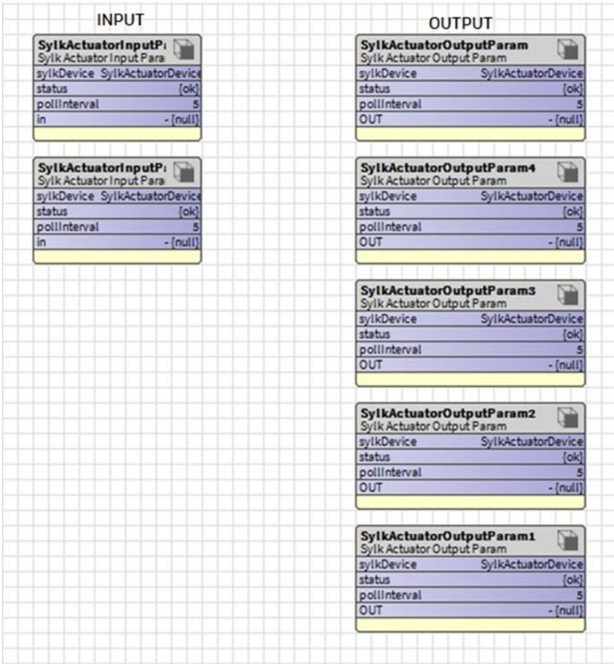
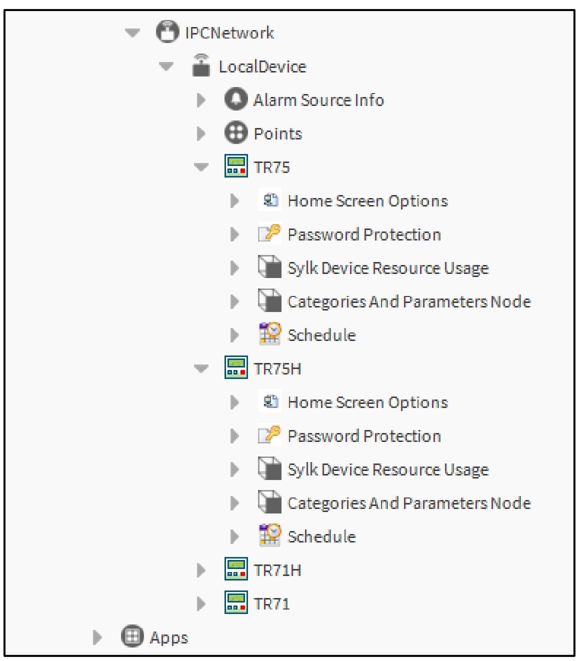



Figure 171: SylkActuator Inputs and Outputs

## Categories and Parameters Node

Categories and Parameters node allows user to reorder Sylk categories and Sylk parameters from workbench and browser. Sylk device will display the categories and parameters in the order specified in this place.




Reordering can be done once user is completed with Sylk device and parameters configuration.

| Note: |
| --- |
| Only TR7X Sylk devices supports reordering feature. |

Reordering Parameters and Categories

The parameters can be reorder across the categories as well as in a category. Also, user can reorder Categories.

To perform reordering

1. Navigate to TR7X from Nav window and double click Categories and Parameters Node.

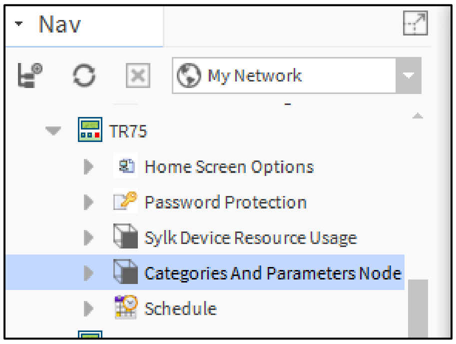



Figure 172: Categories and Parameters Node

2. Select Categories and Parameters Node and right click > Actions > Load Categories And Parameters.

   This will load all the categories and parameters associated with this Sylk device.

3. Right click on any Category from the Property Sheet and select Reorder.

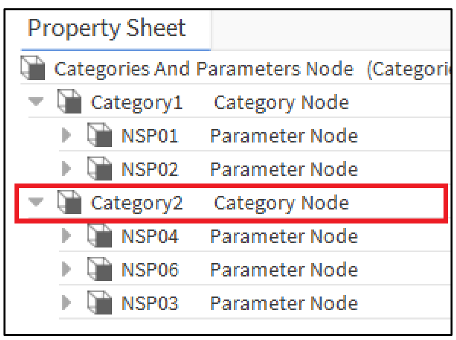



Figure 173: Categories and Parameters Node Property Sheet

4. Use Move Up or Move Down to reorder the parameter and Click Ok.

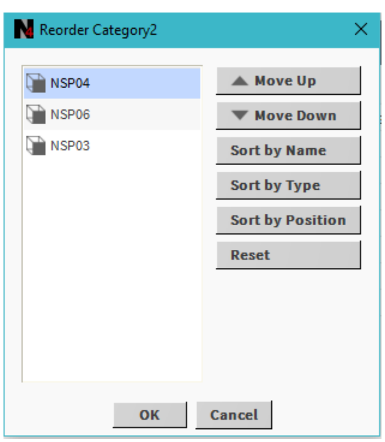
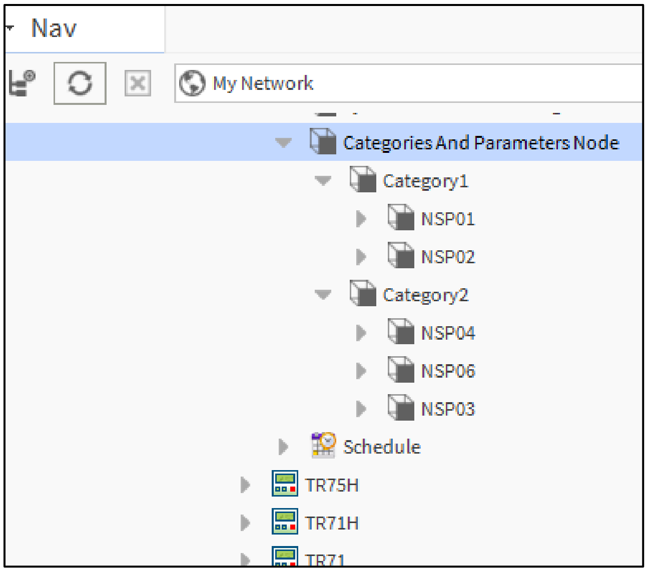



Figure 174: Category Reordering

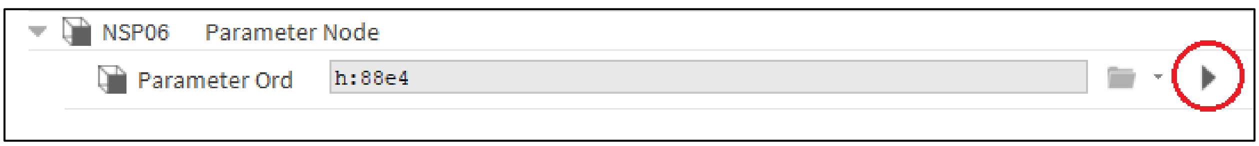
Similarly, you reorder for Categories also.

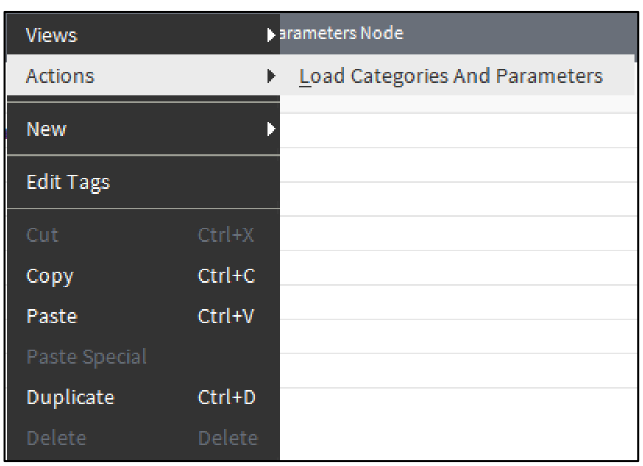
Also, you can perform the similar reordering from Nav window.




Once reordering is completed and Sylk commission is done, user can see the same order of categories and parameters in the connected Sylk device.

Categories and Parameters Node allows user to see all the parameters associated with that device, and also provides an option to navigate to the parameter using the ord.




> **Note:**
>
> To reflect any changes done in Sylk device and sylk parameter, user need to run below action.
>
> Navigate to Actions > Load Categories And Parameters.
>
> 
> 
>
> Figure 175: Load Categories And Parameters

Renaming Parameter and Categories

Categories and Parameters node allows user to rename categories as well as parameters from work-bench.

To rename Parameter

1. Select parameter from the category, right click and select Rename.
2. This action brings Rename window, enter the new name and click Ok.
3. Navigate to Categories and Parameters node, right click and select Actions > Load Categories And Parameters. This action loads the changes.
   - In the wiresheet, user can view the parameter is renamed with the new name.

Similarly, user can rename the Categories also.

To rename Categories

1. Select category, right click and select Rename.
2. This action brings Rename window, enter the new name and click Ok.
3. Navigate to Categories and Parameters node, right click and select Actions > Load Categories And Parameters. This action loads the changes in the category, simultaneously affecting all the parameters associated with the category.
   - In the wiresheet, user can view that category name under all parameters associated with it, is updated with new name.

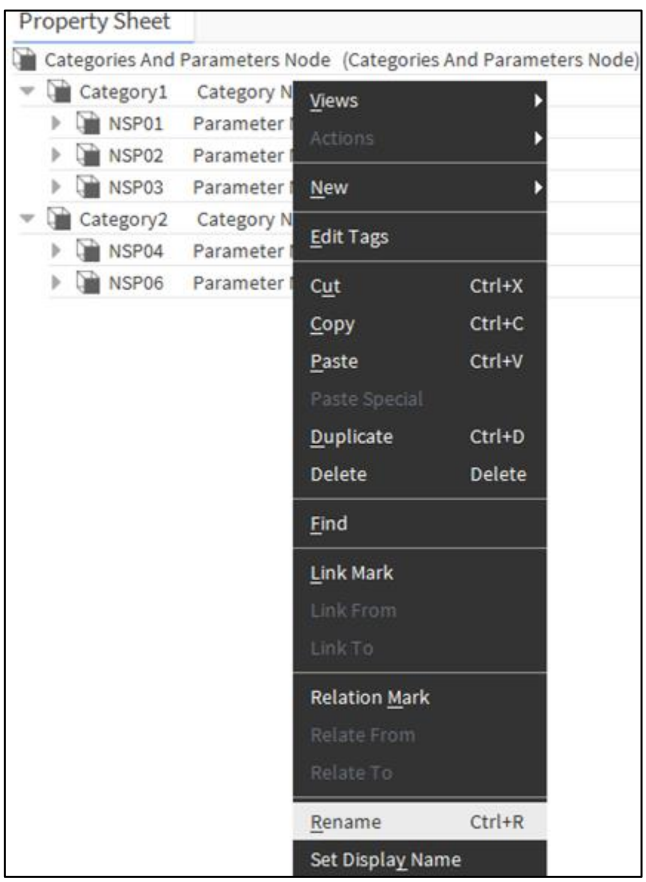



Figure 176: Renaming Parameter and Categories

### Disassociating Parameter and Categories

- Categories and Parameters node allows user to disassociating categories and parameters from the associated Sylk Device.
- To disassociate Parameter, select parameter from the category > right click and select **Delete**. This action disassociates that parameter from the Sylk device, and none will be displayed for that Sylk parameter.
- To disassociate Category, select category > right click and select **Delete**. This action disassociates all the parameters from the Sylk device, and none will be displayed for all the parameters that associated with that category.

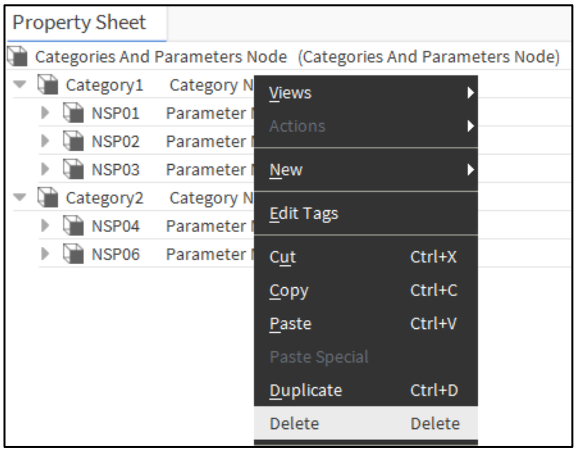
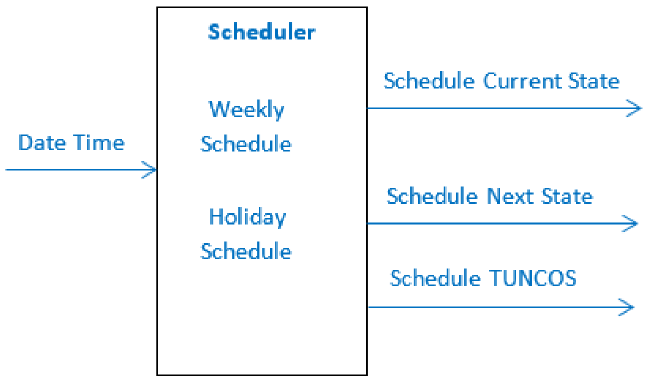



Figure 177: Disassociating Parameter and Categories

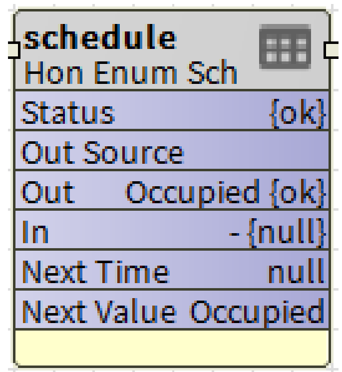
## Sylk Schedule

The Schedule function block helps you to configures priorities schedule tasks for the controller. The scheduler allows you to determine the controller current occupancy state for present day and defined occupancy status for next day. An external device via LON communications may update the time of day and date. This function calculates:

1. Current State
2. Next State
3. Time Until Next Change of State (TUNCOS) based on the date and time.




Logic Diagram




Function Block

Figure 178: Schedule Function

Also, you can set the activities that need to be performed when the system is in idle state or the system is down (not functioning). The schedule function calculates the current occupancy state, next state, and time until next change of state (TUNCOS) based on the date/time and the schedule. It makes them available as public variables.

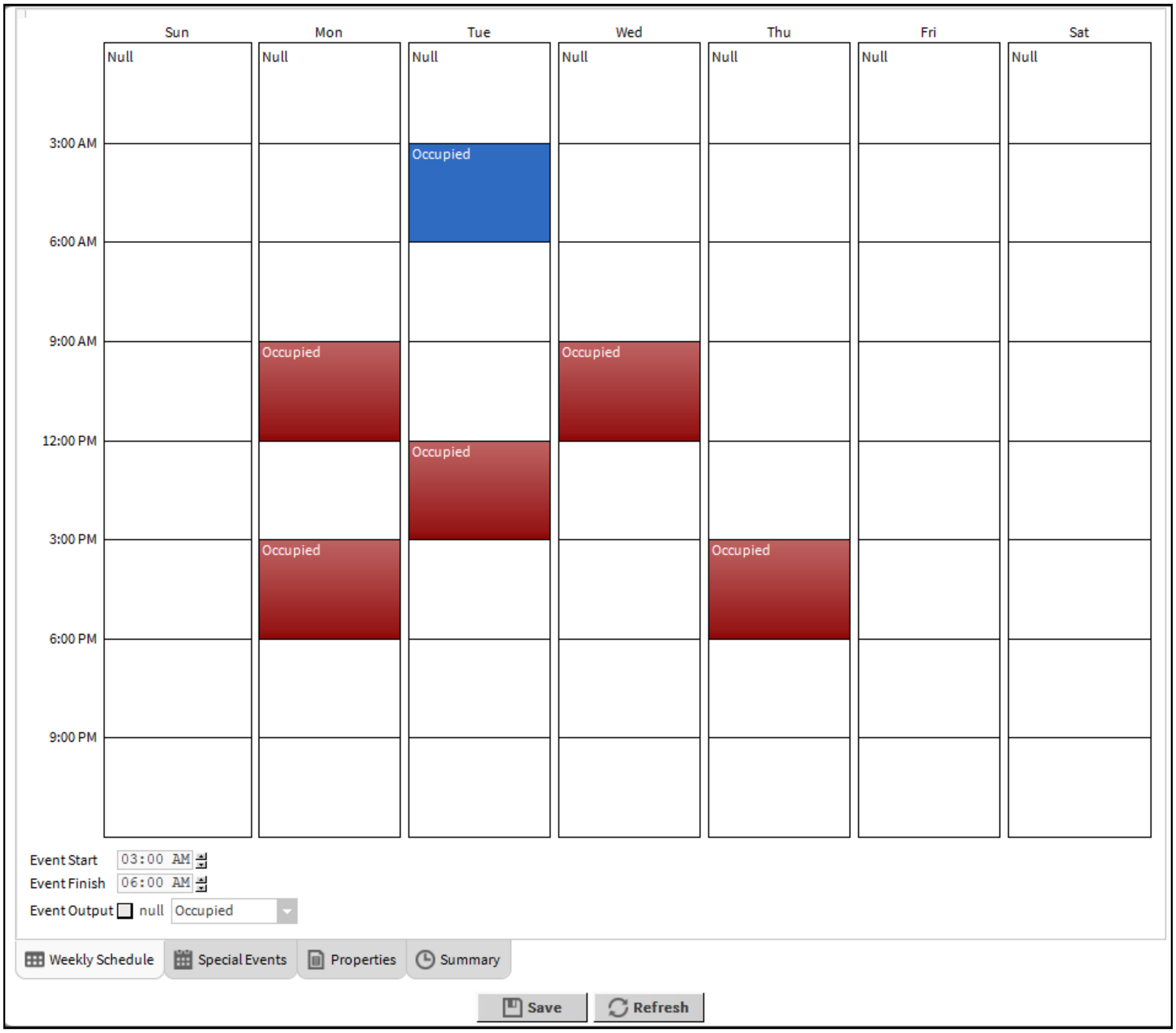



Figure 179: Schedule

## Inputs

Scheduler function block fetch data and time from the operating system. This input data utilized by the function block to deliver output.

## Outputs

- Current State: This state provide output for the preset day or current date. Following are the default output states.
  - Occupied means use the occupied setpoint.
  - Unoccupied means use the unoccupied setpoint.
  - Standby means use the standby setpoint.
- Next State: This state provides the output after the current occupancy state is complete.
  - Unoccupied means the next state is unoccupied.
  - Standby means the next state is standby.
  - OCCNUL means the next state unknown.

- TUNCOS: This is the time duration (in minutes) until the next change of state happen. The controller uses this to recover the setpoint.
- Range: 0 to 11520 minutes (8 days). 11520 minutes means the time until next change of state is unknown or further in the future.
- The enumeration values for all occupancy states are: Occ = 0, Unocc = 1, Bypass = 2, Standby = 3, Null = 255.

## Sylk Schedule Configuration

1. Double-click SylkSchedule on the wire sheet to configure the properties.

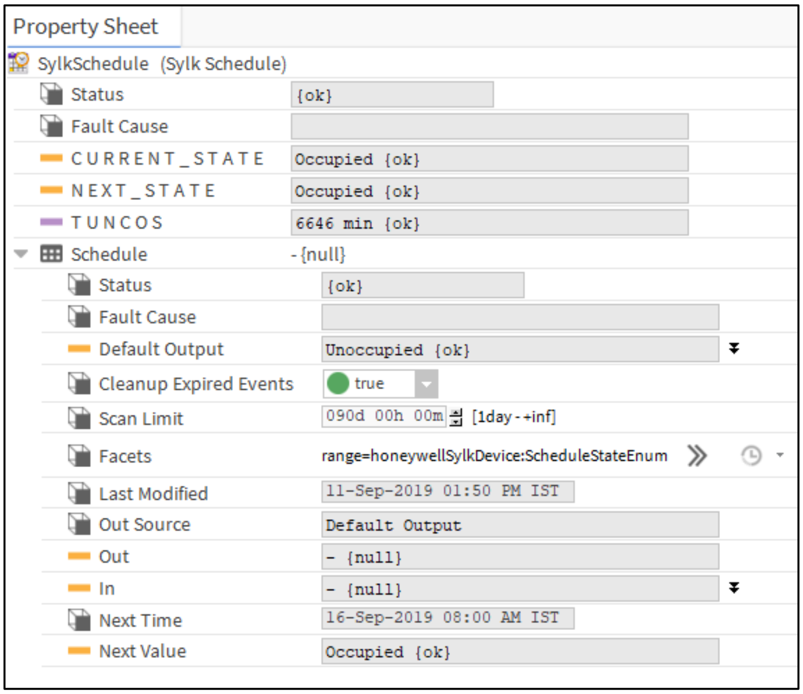



Figure 180: Property Sheet of SylkSchedule

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists. When there are more than one errors, the fault cause shows only the error which is at the top of the list of errors. If the error is corrected, the next error is shown.
- CURRENT_STATE: Provides is the occupancy state the controller must be in at this minute
- NEXT_STATE: Provides the occupancy state the controller goes to after the current state is complete
- TUNCOS: Provides the time (in minutes) until the next change of state
- Schedule: Internal block, and it is used to calculate TUNCOS

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

The Sylk commissioning process shows the validation error, when following conditions are not met.

Four events for a day and the holidays in Sylk schedule with same events.

## SylkSchedule

The SylkSchedule parameter is used in TR75 and TR75H devices. Following is the workflow of the SylkSchedule parameter.

1. Navigate to the Sequenced Control Program folder wire sheet.
2. Add TR75H module to the LocalDevice folder.
3. Navigate to the SylkSchedule parameter under ipcProgrammingTool palette.
4. Drag and drop the SylkSchedule parameter onto the wire sheet.

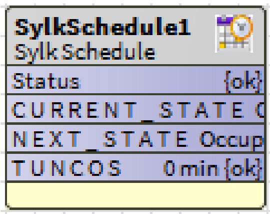



Figure 181: Function Block of SylkSchedule

5. Double-click the SylkSchedule block. The property sheet of the SylkSchedule block is displayed.
6. Click Schedule. The AX Scheduler is displayed.

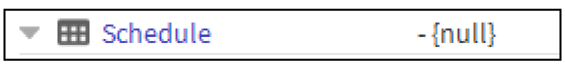



Figure 182: Schedule Option

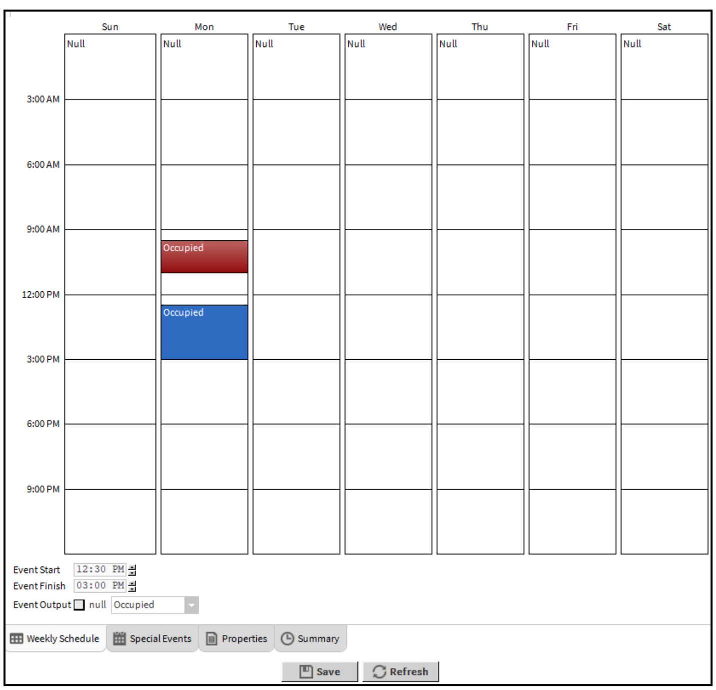



Figure 183: AX Scheduler

7. Add the schedule events as required and save the changes.
8. Open the property sheet of TR75H model.

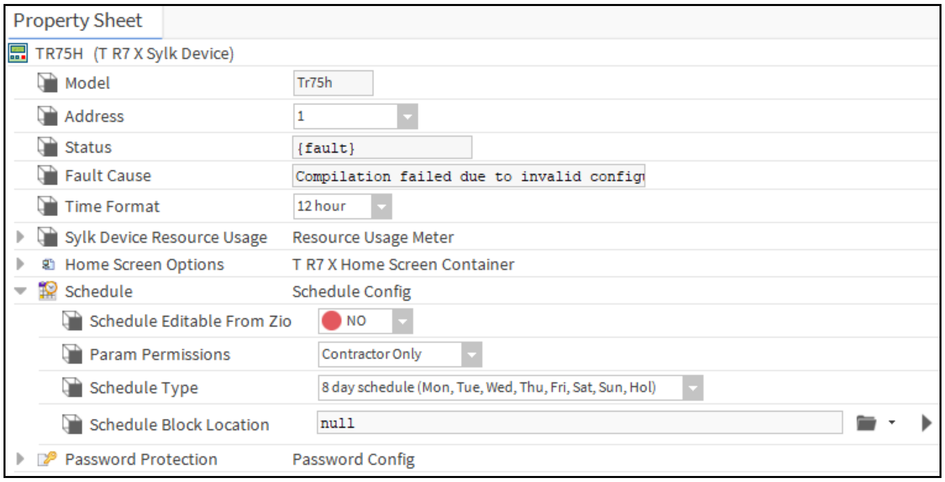



Figure 184: Property Sheet of TR75H Model

9. Set the Schedule Editable From Zio field to Yes.

10. Change Param Permissions (Contractor Only, Tenant Read only, and Tenant Write only) as per requirement.

11. Change the Schedule Type as per requirement (8 day schedule/ 7 day schedule/ 5-2-1 day schedule/ 5-2 day schedule).

12. Enter the schedule block location in the Schedule Block Location field.

- To Enter Scheduler Block Location:

1) Click Folder icon ( ) and select Component Chooser from the drop-down list.

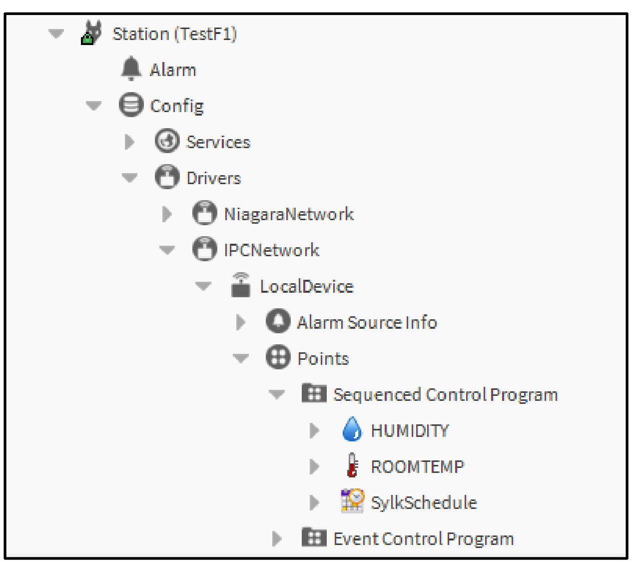
2) On the Select Ord window, Click Drivers > IPCNetwork > LocalDevice > Points > Sequenced Control Program > SylkSchedule, and click OK.

- This action fetch scheduler location to the Scheduler Block Location filed.

- Or

1) Go to Nav Window, navigate to the Station > Config > Drivers > IPCNetwork > LocalDevice > Points > Sequenced Control Program > Select SylkSchedule, and press Ctrl + L.

2) This action opens Ord window, copy the location, and paste in the Scheduler Block Location field.



13. Click Save.
    - Or click Refresh to cancel the changes.

Note:

- Sylk Commissioning fails if the number of events configured per day exceed three and if holidays that have schedule are not matching.
- The Niagara EnumSchedule block, present under schedule palette, is also supported. So, you can give the location of the Niagara schedule in the Schedule Block Location field.

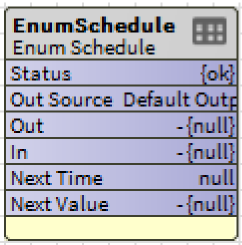



Figure 185: Function Block EnumSchedule

- Schedule configuration work flow changes - after changing any schedule changes in tool perform sylk commission

## Migrating Sylk Scheduled Events from SPYDER

Spyder To IPC Migrator tool capable of migrating the existing Spyder sylk scheduled events to CIPer Model 30 sylk scheduled events.

Follow the below steps to migrate Spyder sylk scheduled events to CIPer Model 30 sylk scheduled events:

1. Navigate to Spyder schedule function block, click Scheduling tab, and click the day of the week to select the day needs to be configured user wants to configure the schedule. For more details, refer Spyder User Guide.
2. Click Apply Event.

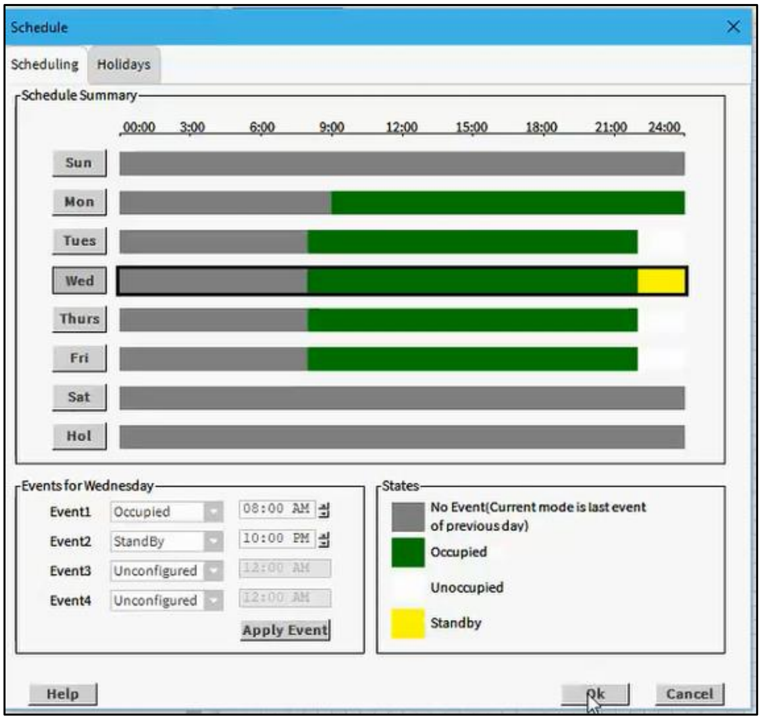
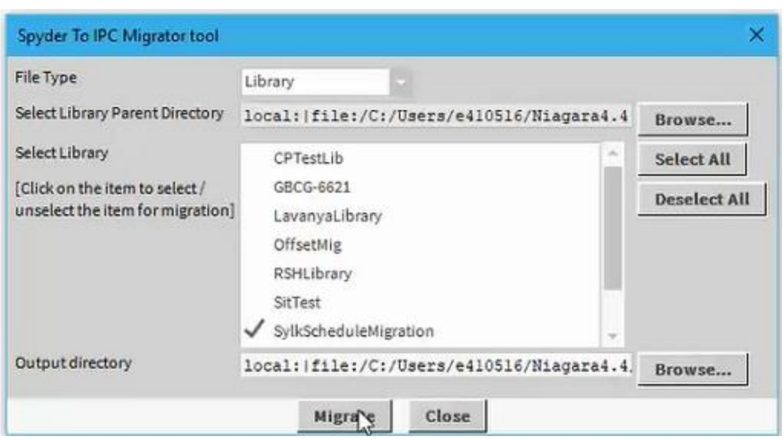



Figure 186: Scheduling View of Spyder Events

3. Navigate to Tools > Spyder To IPC Migrator tool.
4. Select the file type as Library in the File Type drop-down menu.
5. Click Browse next to the Select Niagara Home Directory field to select the path, where the list of libraries is available.
6. Browse to the Output directory, where the migrated applications are stored.
7. Click Migrate.




8. Spyder To IPC Migrator tool popup after successful migration, click Ok.
9. Navigate to CIper schedule function block, click Scheduling tab.

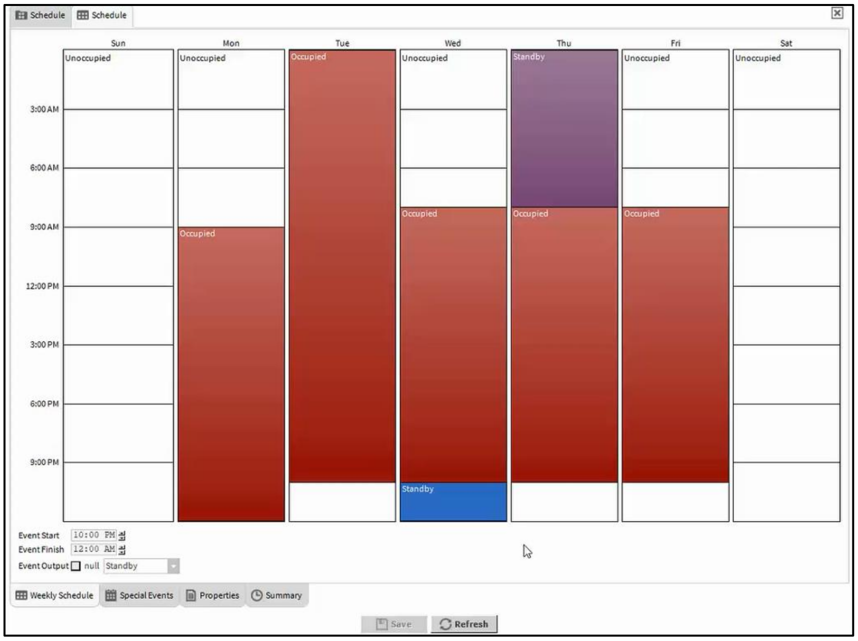



Figure 187: Scheduling View of CIPer Events

You can view all the migrated scheduled events from Spyder Sylk Schedule, displayed in CIPre Sylk Schedule window.

## Scheduling Enum Range

To define Enum range:

1. Navigate to Sylkschedule > Schedule Properties tab > Facets option, select default range (honeywell-SylkDevice:ScheduleStateEnum), and click (>>) icon.

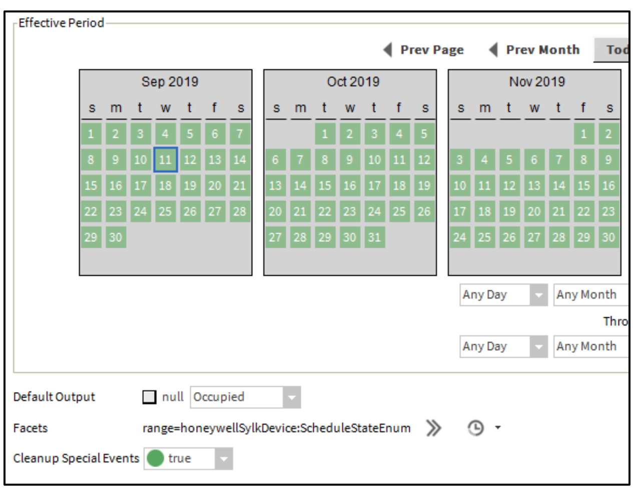



Figure 188: Defining Enum range

2. Select the default EnumRange, click (...) icon, and click (>>) icon.

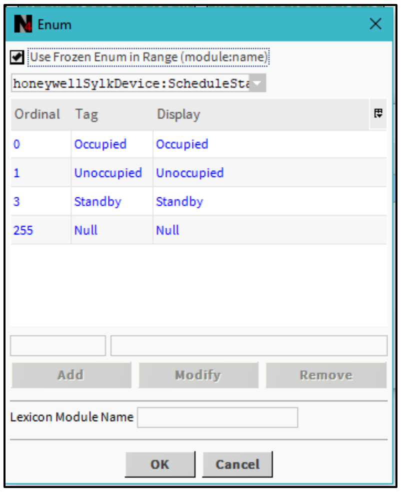



Figure 189: Default Enum Range

To modify and add the entry in the Zio Enum Library:

1. Select the entry.
2. Modify the Ordinal value, Display field, and Tag field and click Modify.

   Or

   Change the Ordinal value, Display field, and Tag field and click Add.

**Note:**

- A total of maximum 255 entries can be added in a Zio enum library.
- All the Holidays that are schedule in Spyder Schedular, after migration applicable to CIPer Sylk Schedular.
- All the unconfigured events in Spyder Schedular, displays Unscheduled in CIPer Schedular.
- Unscheduled events in Spyder Schedular, display Unoccupied as default output for CIPer Schedular.
- In configured Schedule events gaps also consider as event (Tool will always write 4th event as default event (eg., UnOccupied), so that user can edit the end time of 3rd event. While reading the schedule from wall module, tool will not create default-event except in case non-continuous events ie., while filling the gaps)

# Sylk Parameters

This section in the ipcProgrammingTool palette contains the list of parameters available. Based on your requirement, you can add the parameters for the Sylk devices.

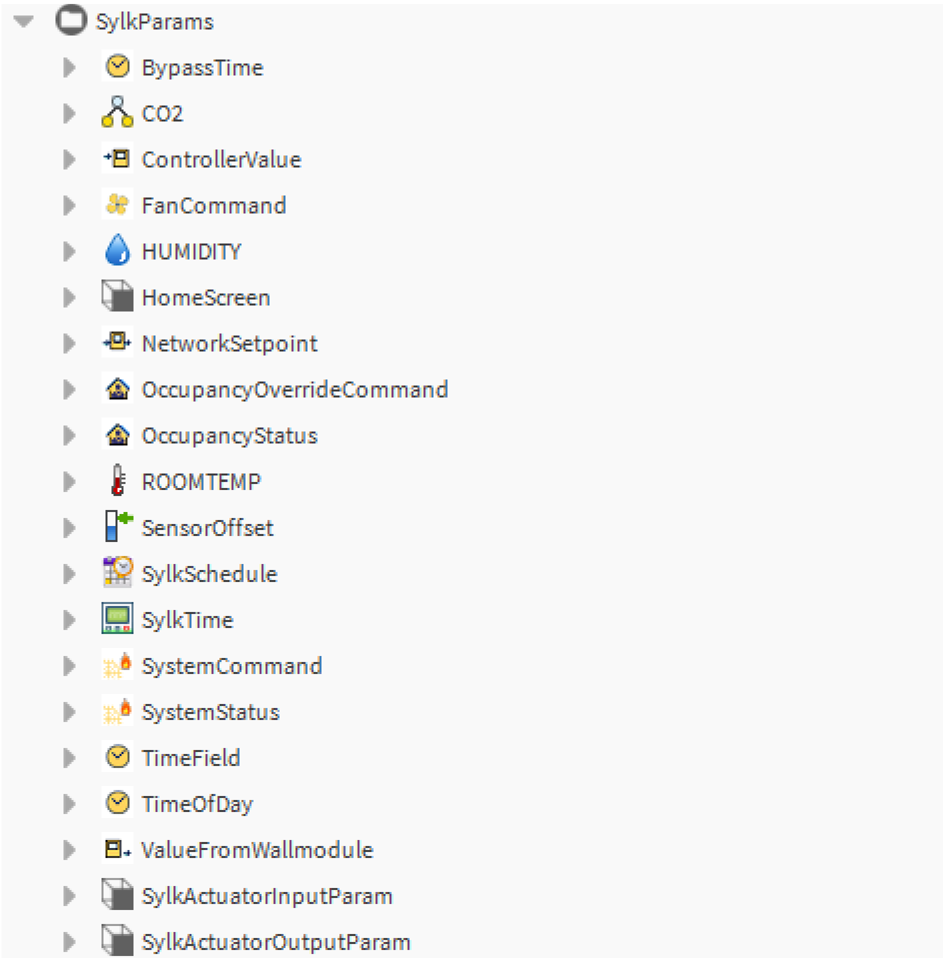



Figure 190: SylkParams View

Following are the Sylk parameters:

- BypassTime: To provide the bypass time to wall module
- CO2: CO2 concentration in the space
- ControllerValue: Value from the controller to the wall module
- FanCommand: To provide commands to the fan from wall module
- HUMIDITY: Percent humidity of the Space
- HomeScreen: To configure the LED display of wall module
- NetworkSetpoint: To provide the setpoints to the wall module
- OccupancyOverrideCommand: To override the Unoccupancy Mode to Occupied Mode
- OccupancyStatus: Occupancy status in the wall module
- ROOMTEMP: Temperature of the space
- SensorOffset: To provide the sensor offset to wall module

- SylkSchedule: To add occupancy schedule to the wall module
- SylkTime: to configure and change the value of Sylk module and platform time as well
- SystemCommand: To override the system command from the wall module
- SystemStatus: System status in the wall module
- TimeField: To configure the time format of wall module
- TimeOfDay: Time of the day to show on Sylk Device
- ValueFromWallModule: To provide the value from the wall module to the controller
- SylkActuatorInputParam: To provide input to Sylk Actuator.
- SylkActuatorOutputParam: To provide output from Sylk Actuator.

Note:

If you are using two network setpoints and if you connect the output of first block to the input of second block, the value of the second block does not reflect in the first block unless you connect the output of the second block to the input of the first block. The change in the workflow is handled in the migration process by providing the connection from destination component to the source component, so that the values are updated in the source component also.

## Bypass Time Configuration

1. Double-click BypassTime sensor on the wire sheet to configure the sensor properties.

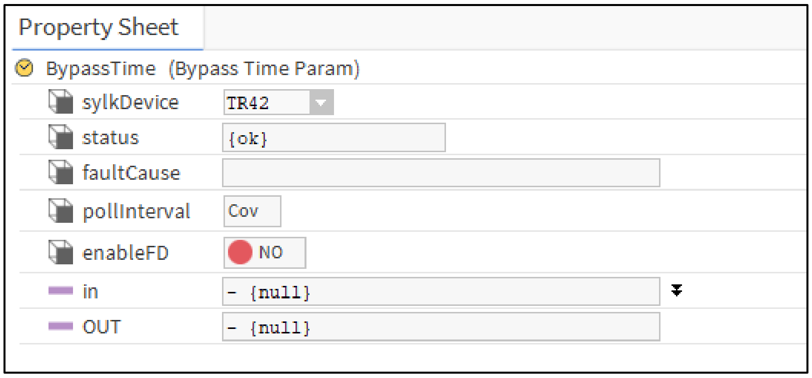



Figure 191: Property Sheet of BypassTime

- sylkDevice: Select the required TR7x device from the drop-down menu

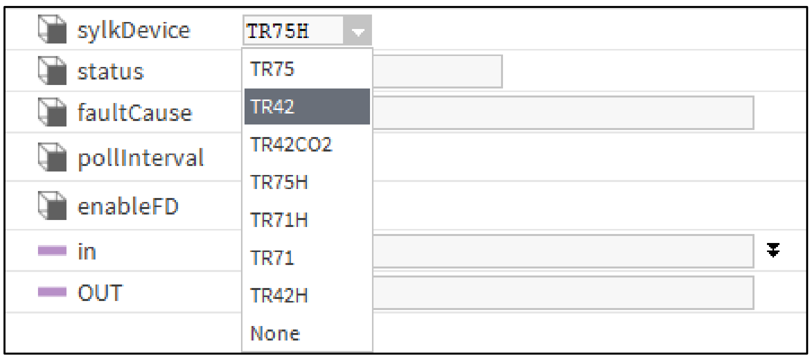



Figure 192: sylkDevice Drop-Down Menu

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists. When there are more than one errors, the fault cause shows only the error which is at the top of the list of errors. If the error is corrected, the next error is shown.
- pollInterval: Poll interval is the time between the end of a timeout period or completion of a network request, and the next request for data on the network
- enableFD: Fail Detect is the time until the CIPer Model 30 is notified of a failure on this point.
  - True: If the parameter has not received an update from the IPC network source in the fail detect time, then an alarm is generated and the present Value is set to Invalid. Note that fail detect time depends on the update rate configured.
  - False: False means the object retains the last value that was written to it until a IPC network source changes it or the Honeywell CIPer Model 30 has a power outage or reset
- In: To provide the Controller Bypass Time Value to the Wall module.
- OUT: To provide the wall module Bypass Time Value to the controller

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

## CO2 Configuration

1. Double-click CO2 sensor on the wire sheet to configure the sensor properties.

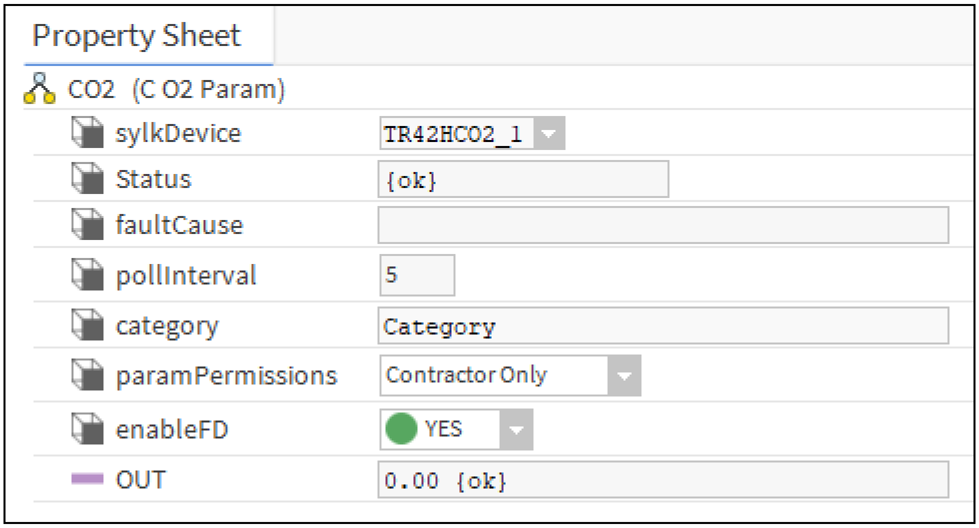



Figure 193: Property Sheet of CO2

- sylkDevice: Select the required TR4x device from the drop-down menu

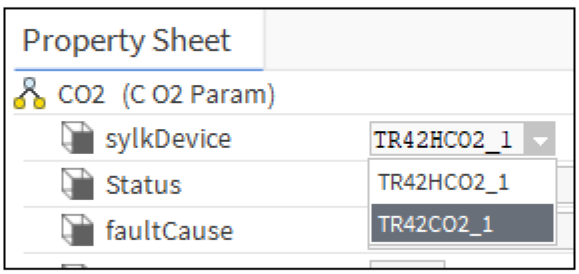



Figure 194: sylkDevice Drop-Down Menu

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists. When there are more than one errors, the fault cause shows only the error which is at the top of the list of errors. If the error is corrected, the next error is shown.
- pollInterval: Poll interval is the time between the end of a timeout period or completion of a network request, and the next request for data on the network.
- category: The category defined by the user while configuring Sylk parameters. By default, it shows Category as category
- paramPermissions: User can select the viewing option for the parameter. Select the parameter as viewable by Contractor Only or Tenant Read Only from the drop-down menu. If Contractor Only is selected, only the contractor can view the parameters in the wall module. If Tenant Read Only is selected, tenant can view the parameters, but cannot make any changes in the parameters.

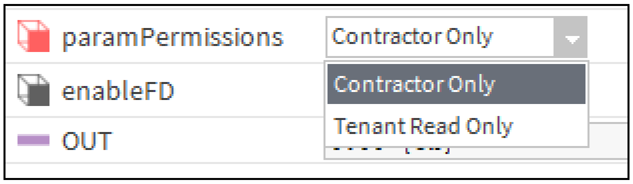



Figure 195: paramPermissions Drop-Down Menu

- enableFD: Fail Detect is the time until the CIPer Model 30 is notified of a failure on this point.
  - True: If the parameter has not received an update from the IPC network source in the fail detect time, then an alarm is generated and the presentValue is set to Invalid. Note that fail detect time depends on the update rate configured.
  - False: False means the object retains the last value that was written to it until a IPC network source changes it or the Honeywell CIPer Model 30 has a power outage or reset
- OUT: Shows the current value of the parameter
- temperatureUnit: Select the unit as Degree F or Degree C as per requirement

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

## Value From Controller Configuration

1. Double-click ControllerValue on the wire sheet to configure the properties.

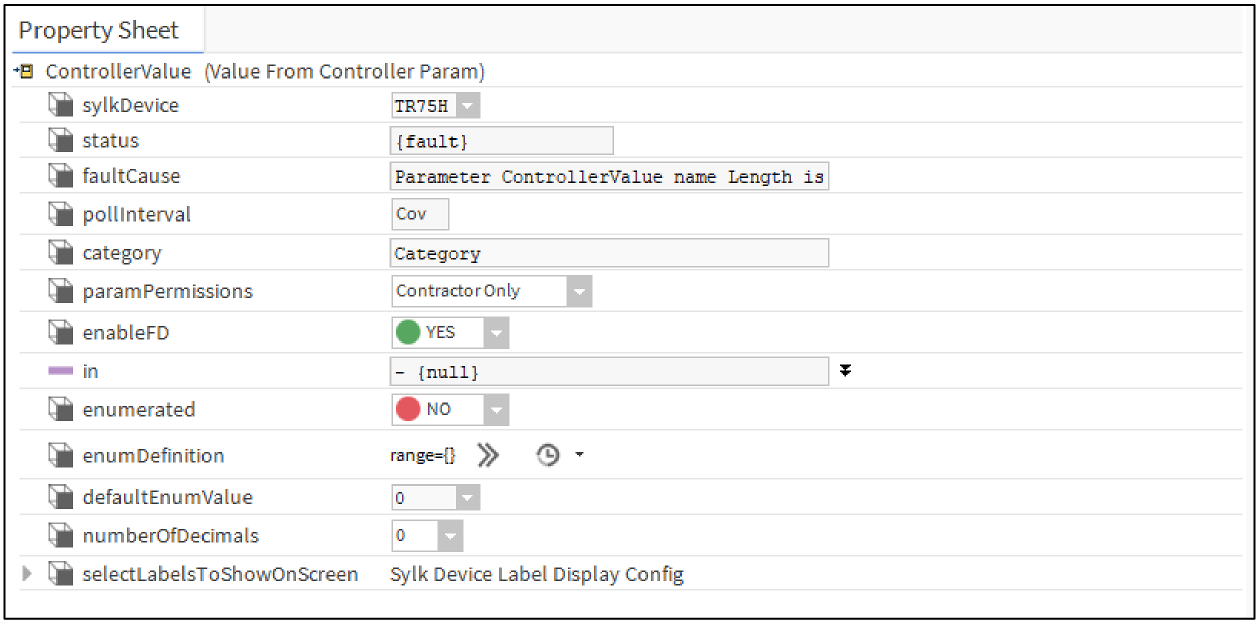



Figure 196: Property Sheet of ControllerValue

- sylkDevice: Select the required TR7x or TR4x device from the drop-down menu

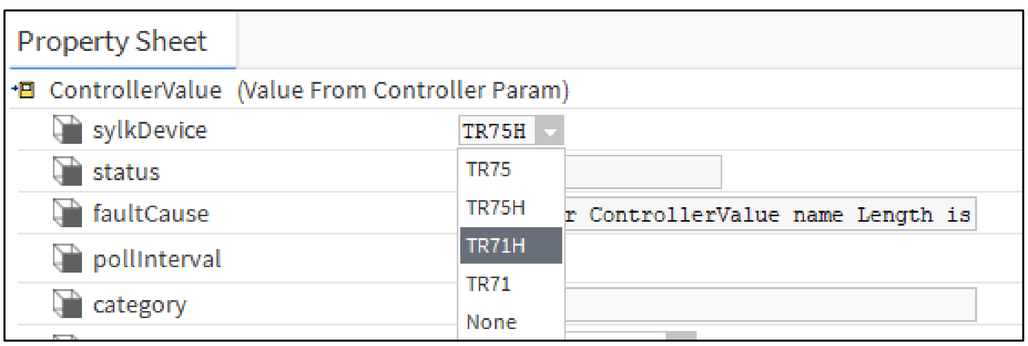



Figure 197: sylkDevice Drop-Down Menu

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists.
- pollInterval: Poll interval is the time between the end of a timeout period or completion of a network request, and the next request for data on the network
- category: The category defined by the user while configuring Sylk parameters. By default, it shows Category as category
- paramPermissions: User can select the viewing option for the parameter. Select the parameter as viewable by Contractor Only or Tenant Read Only from the drop-down menu. If Contractor Only is selected, only the contractor can view the parameters in the wall module. If Tenant Read Only is selected, tenant can view the parameters, but cannot make any changes in the parameters.

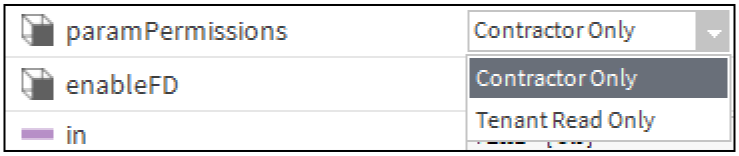



Figure 198: paramPermissions Drop-Down Menu

- enableFD: Fail Detect is the time until the CIPer Model 30 is notified of a failure on this point.
  - True: If the parameter has not received an update from the IPC network source in the fail detect time, then an alarm is generated and the presentValue is set to Invalid. Note that fail detect time depends on the update rate configured.
  - False: False means the object retains the last value that was written to it until a IPC network source changes it or the Honeywell CIPer Model 30 has a power outage or reset.
- in: To provide the Setpoints from the controller to the Wall module.
- enumerated: This option allows the user to provide Enum Setpoints
- enumDefinition: If the Enumerated is set to true then user can set the multiple states of the Enum setpoint.
  a. To set the Enum states of the Setpoint click on the >> button.


Figure 199: enumRange Parameter

  b. Set the states Ordinal & Display as per requirement. For example,

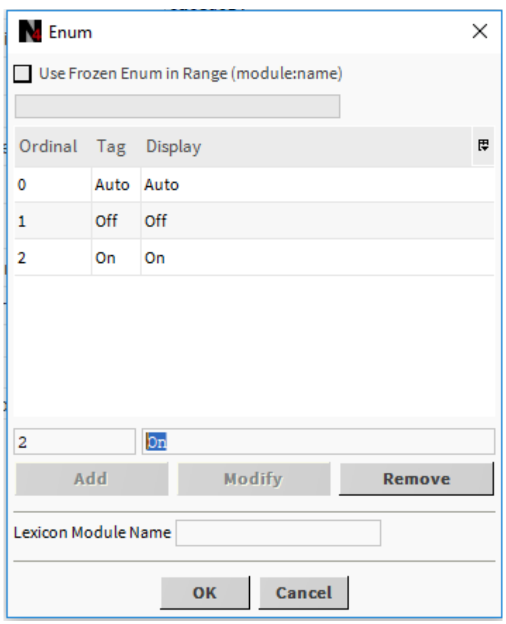



Figure 200: Enum Window

  c. Click Ok to save.
- defaultEnumValue: This option displays list of default Enum value from the defined enum range.
- numberofDecimals: Provide the value of numbers of decimal as per requirement.
- selectLabelsToShowOnScreen: Select the required option to show the parameter on the home screen shown below:

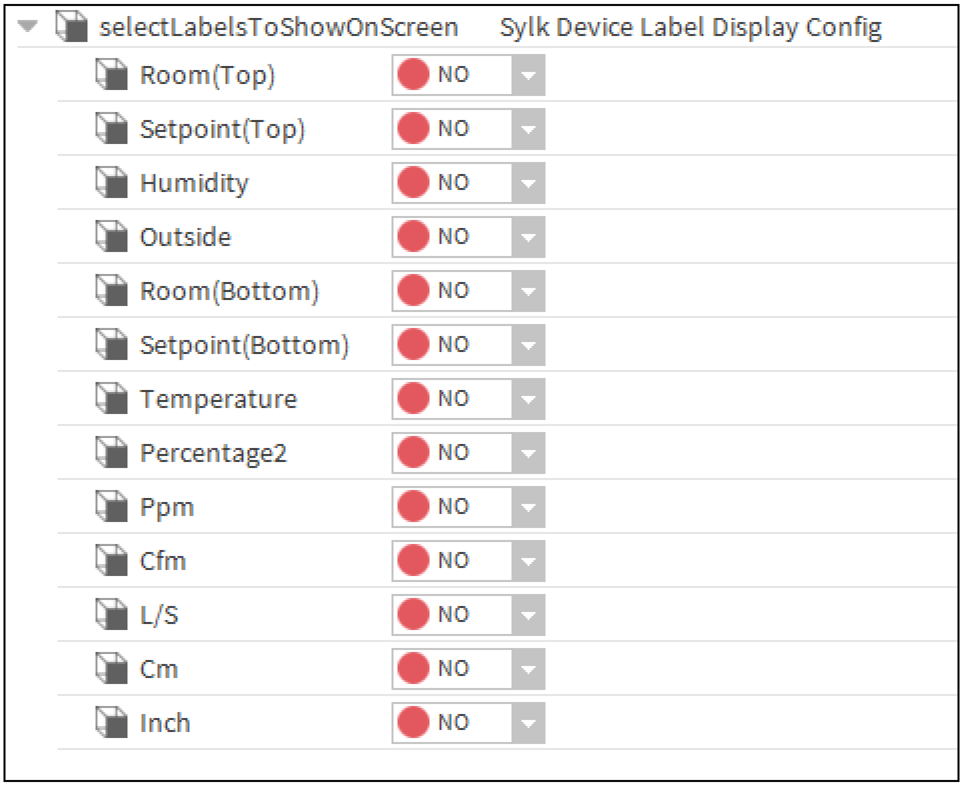



Figure 201: Sub-Menu of selectLabelsToShowOnScreen

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

## Fan Command Configuration

1. Double-click FanCommand on the wire sheet to configure the properties.

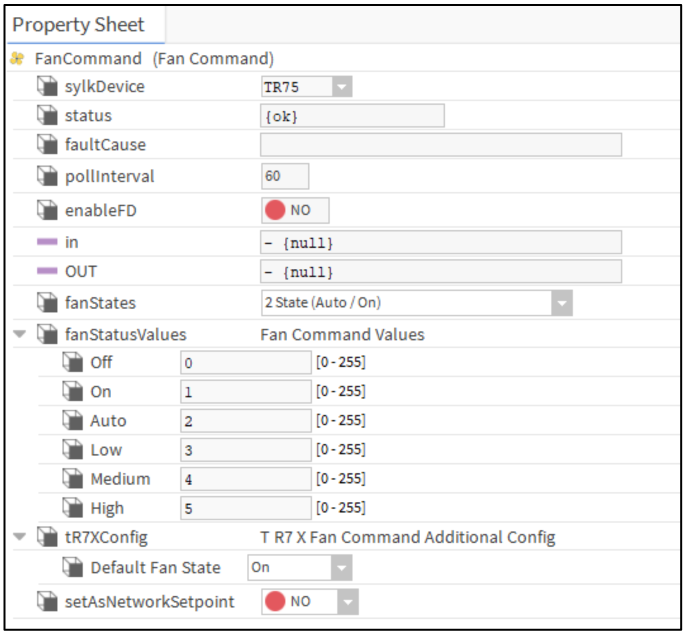



Figure 202: Property Sheet of FanCommand

- sylkDevice: Select the required TR7x or Tr4x device from the drop-down menu

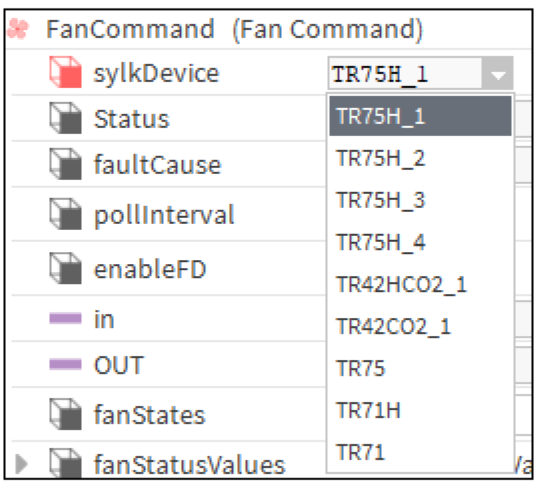



Figure 203: sylkDevice Drop-Down Menu

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists. When there are more than one errors, the fault cause shows only the error which is at the top of the list of errors. If the error is corrected, the next error is shown.
- pollInterval: Poll interval is the time between the end of a timeout period or completion of a network request, and the next request for data on the network
- enableFD: Fail Detect is the time until the CIPer Model 30 is notified of a failure on this point.

  - True: If the parameter has not received an update from the IPC network source in the fail detect time, then an alarm is generated and the present Value is set to Invalid. Note that fail detect time depends on the update rate configured.
  - False: False means the object retains the last value that was written to it until a IPC network source changes it or the Honeywell CIPer Model 30 has a power outage or reset
- In: To provide the Fan Enable Command from the controller to the Wall module.
- Out: To provide the Final Fan Enable Command from wall module to the controller
- fanStates: Select the States of the Fan command from the wall module as per requirement.

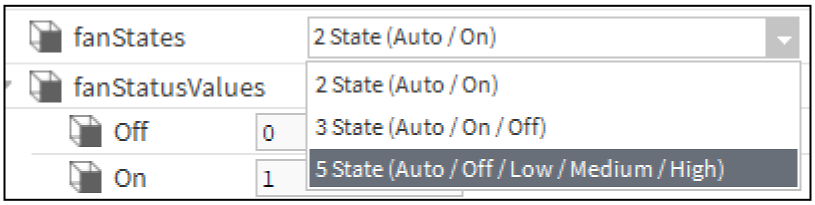



Figure 204: Sub-Menu of fanStates

- fanStatusValues: Set the Fan State Values as per requirement.

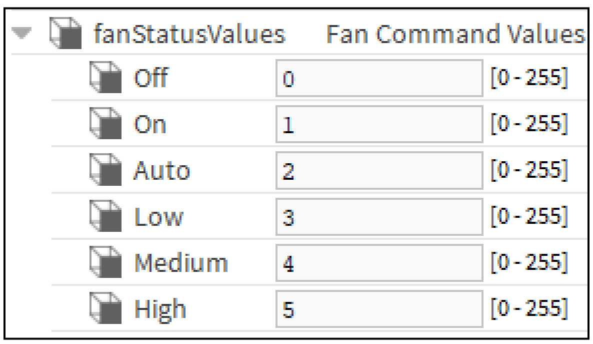



Figure 205: Sub-Menu of fanStatusValues

- tR7XConfig:
  - Default Fan State: Set the default state of Fan Command.

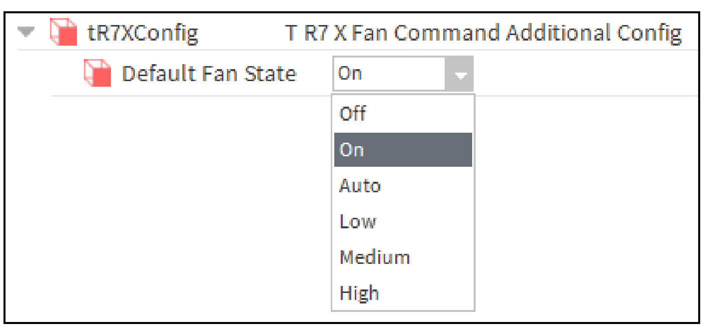



Figure 206: Sub-Menu of tR7XConfig

- SetAsNetworksetpoint: This option allows you to specify, if the param needs to be configured as network set point or output only param. If you select Yes, then you have access to write and read values, if No selected you only have access to read values from sylk device.

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

## Humidity Configuration

1. Double-click HUMIDITY sensor on the wire sheet to configure the sensor properties.

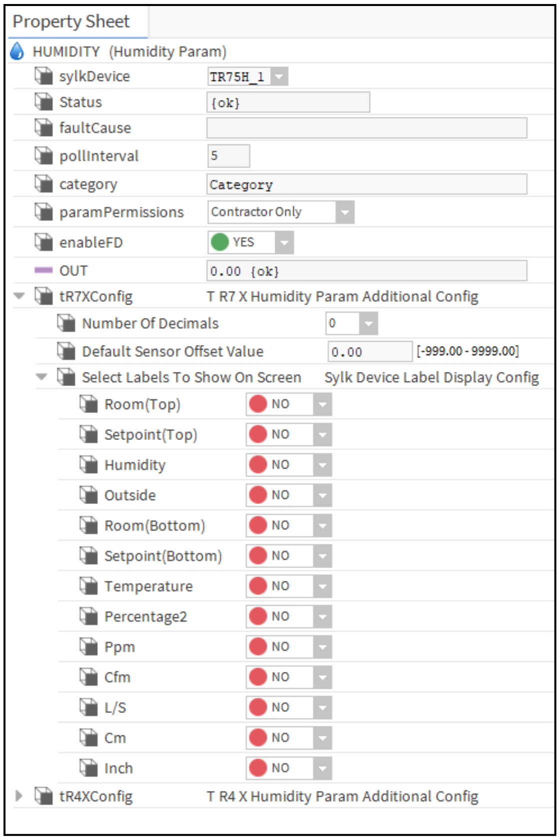



Figure 207: Property Sheet of HUMIDITY

- sylkDevice: Select the required TR7x or TR4x device from the drop-down menu.

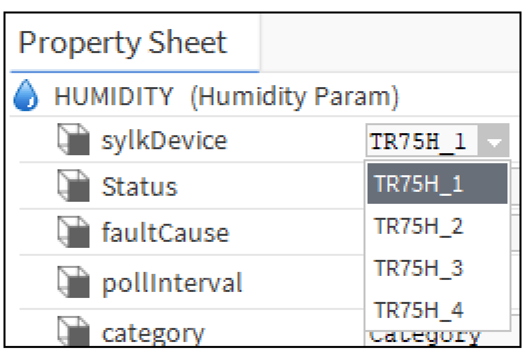



Figure 208: sylkDevice Drop-Down Menu

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists. When there are more than one errors, the fault cause shows only the error which is at the top of the list of errors. If the error is corrected, the next error is shown.
- pollInterval: Poll interval is the time between the end of a timeout period or completion of a network request, and the next request for data on the network

- category: The category defined by the user while configuring Sylk parameters. By default, it shows Category as category
- paramPermissions: User can select the viewing option for the parameter. Select the parameter as viewable by Contractor Only or Tenant Read Only from the drop-down menu. If Contractor Only is selected, only the contractor can view the parameters in the wall module. If Tenant Read Only is selected, tenant can view the parameters, but cannot make any changes in the parameters.

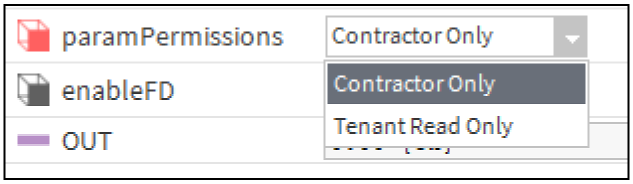



Figure 209: paramPermissions Drop-Down Menu

- enableFD: Fail Detect is the time until the CIPer Model 30 is notified of a failure on this point.
  - True: If the parameter has not received an update from the IPC network source in the fail detect time, then an alarm is generated and the presentValue is set to Invalid. Note that fail detect time depends on the update rate configured.
  - False: False means the object retains the last value that was written to it until a IPC network source changes it or the Honeywell CIPer Model 30 has a power outage or reset
- OUT: Shows the current value of the parameter
- tR7XConfig:
  - Number of Decimals: Enter the decimal accuracy
  - Default Sensor Offset Value: Enter the default value for sensor offset
  - Select Labels To Show On Screen: Select the Humidity option to show the parameter on the home screen.

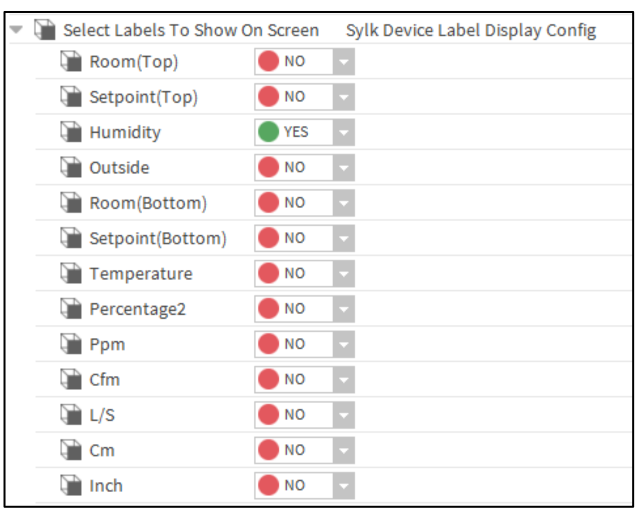



Figure 210: Sub-Menu of tR7XConfig

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

## HomeScreen Options Configuration

1. Double-click homeScreenOptions present under the respective TR7x module to configure the properties.

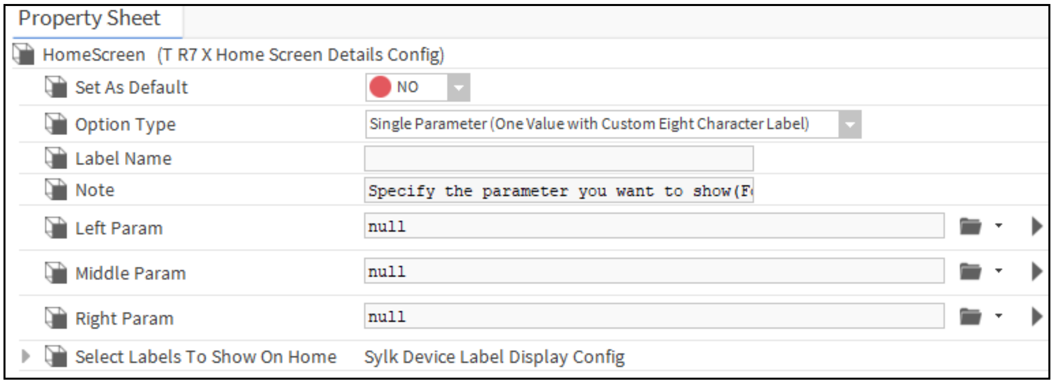



Figure 211: Property Sheet of Room Temperature

- Set As Default: Select this option if the current Home screen need to be set as default
- Option Type: Select the type of screen is required

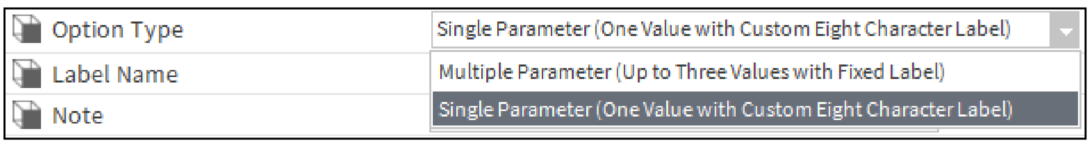



Figure 212: Option Type Drop-Down Menu

- Label Name: Add the label name as per requirement
- Note: "Specify the parameter you want to show."
- Left Parameter: Select the parameter need to be shown on left side of the home screen
- Middle Parameter: Select the parameter need to be shown on middle of the home screen
- Right Parameter: Select the parameter need to be shown on right side of the home
- Select Labels To Show On Home Screen: Select the required option (Room top or Room Bottom) to show the parameter on the home screen shown below:

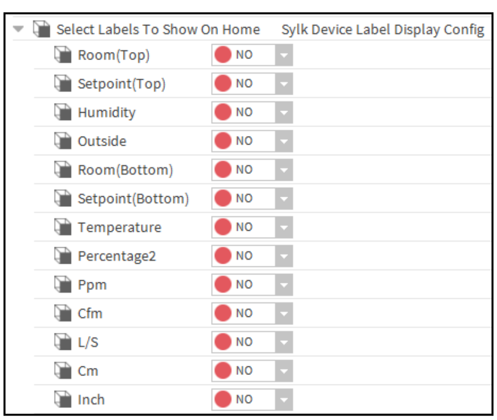



Figure 213: Sub-Menu of Select Labels To Show On Home

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

## Network Setpoint Configuration

These are outputs from the wall module such as room setpoint

1. Double-click NetworkSetpoint on the wire sheet to configure the properties.

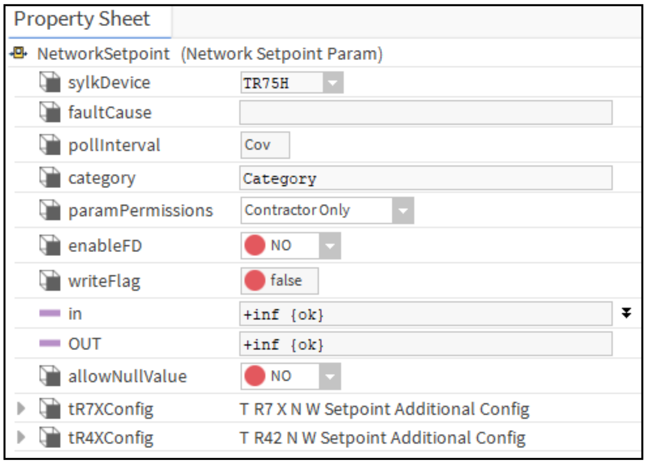



Figure 214: Property Sheet of NetworkSetpoint

- sylkDevice: Select the required TR7x or TR4x module from the drop-down menu

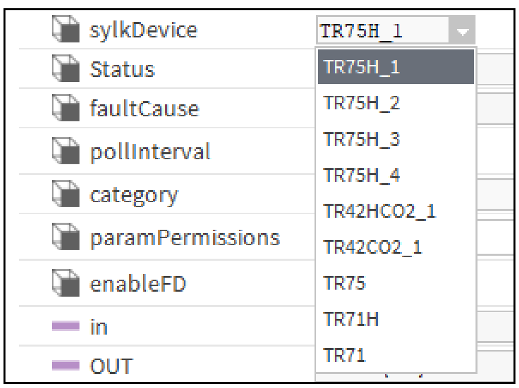



Figure 215: sylkDevice Drop-Down Menu

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists. When there are more than one errors, the fault cause shows only the error which is at the top of the list of errors. If the error is corrected, the next error is shown.
- pollInterval: Poll interval is the time between the end of a timeout period or completion of a network request, and the next request for data on the network
- category: The category defined by the user while configuring Sylk parameters. By default, it shows Category as category
- paramPermissions: User can select the viewing option for the parameter. Select the parameter as viewable by Contractor Only, Tenant Read Only, or Tenant Read Write from the drop-down menu. If Contractor Only is selected, only the contractor can view the parameters in the wall module. If Tenant Read Only is selected, tenant can view the parameters, but cannot make any changes in the parameters. If Tenant Read Write is selected, tenant can view as well as modify the parameters.

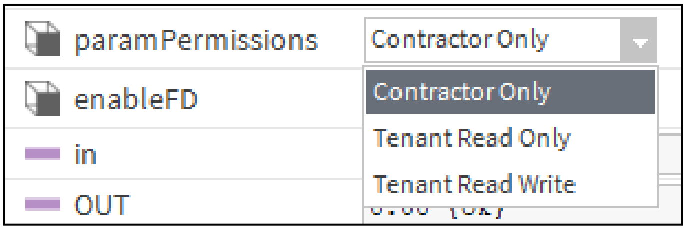



Figure 216: paramPermissions Drop-Down Menu

- enableFD: Fail Detect is the time until the CIPer Model 30 is notified of a failure on this point.
  - True: If the parameter has not received an update from the IPC network source in the fail detect time, then an alarm is generated and the presentValue is set to Invalid. Note that fail detect time depends on the update rate configured.
  - False: False means the object retains the last value that was written to it until a IPC network source changes it or the Honeywell CIPer Model 30 has a power outage or reset
- in: To provide the Controller Setpoint Value to the Wall module.
- OUT: To provide the wall module Setpoint Value to the controller
- allowNullValue: This option allows the user to provide Null value
- selectLabelsToShowOnScreen: Select the required option (Setpoint top or Setpoint Bottom) to show the parameter on the home screen shown below:

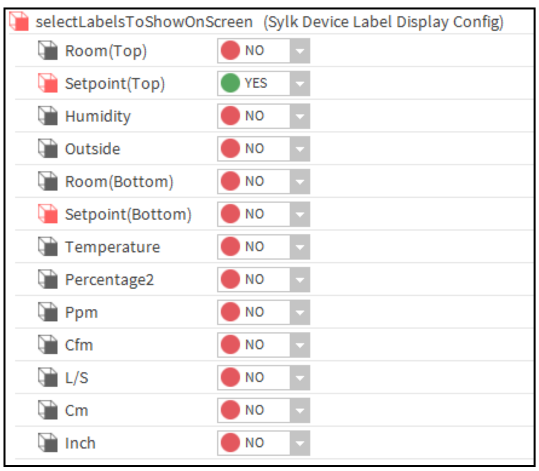



Figure 217: selectLabelsToShowOnScreen Drop-Down Menu

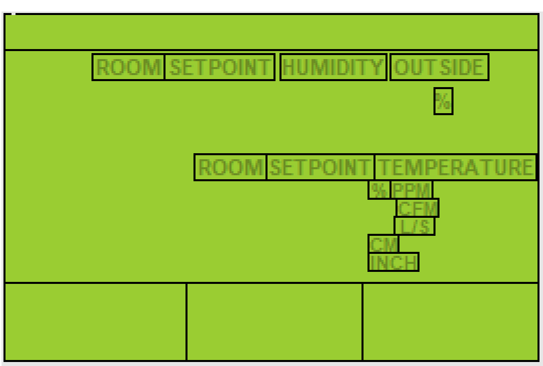



Figure 218: Display of Wall Module TR7X

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

## OccupancyOverrideCommand Configuration

1. Double-click OccupancyOverrideCommand on the wire sheet to configure the properties.

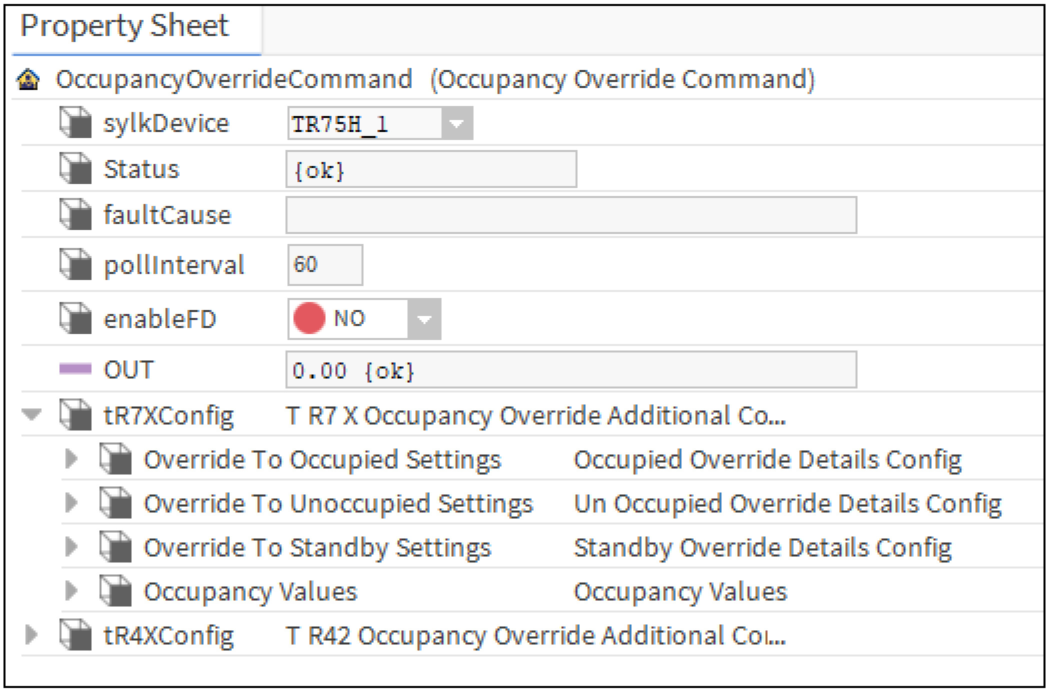



Figure 219: Property Sheet of OccupancyOverrideCommand

- sylkDevice: Select the required TR7x or TR4x device from the drop-down menu.

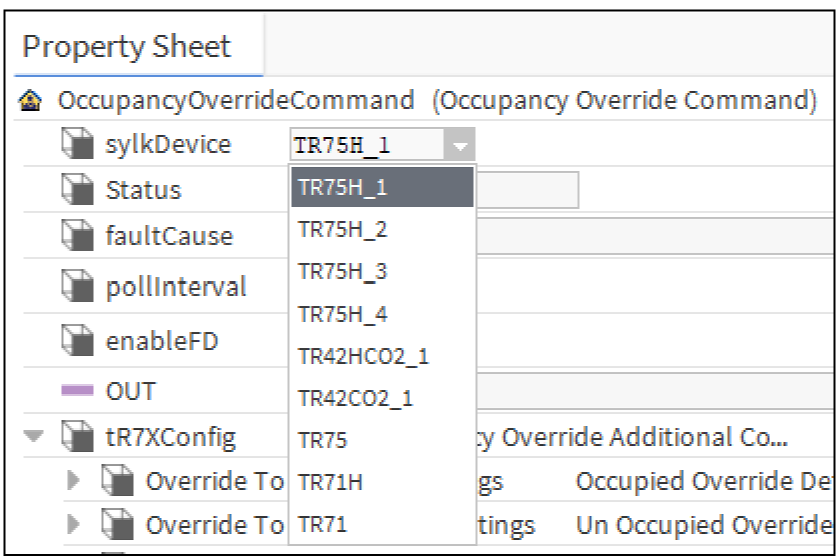



Figure 220: sylkDevice Drop-Down Menu

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists.
- pollInterval: Poll interval is the time between the end of a timeout period or completion of a network request, and the next request for data on the network.

- enableFD: Fail Detect is the time until the CIPer Model 30 is notified of a failure on this point.
  - True: If the parameter has not received an update from the IPC network source in the fail detect time, then an alarm is generated and the presentValue is set to Invalid. Note that fail detect time depends on the update rate configured.
  - False: False means the object retains the last value that was written to it until a IPC network source changes it or the Honeywell CIPer Model 30 has a power outage or reset
- OUT: Provide the Override Command to the wall module from the wall module
- tR7XConfig:
  - Override To Occupied Settings:

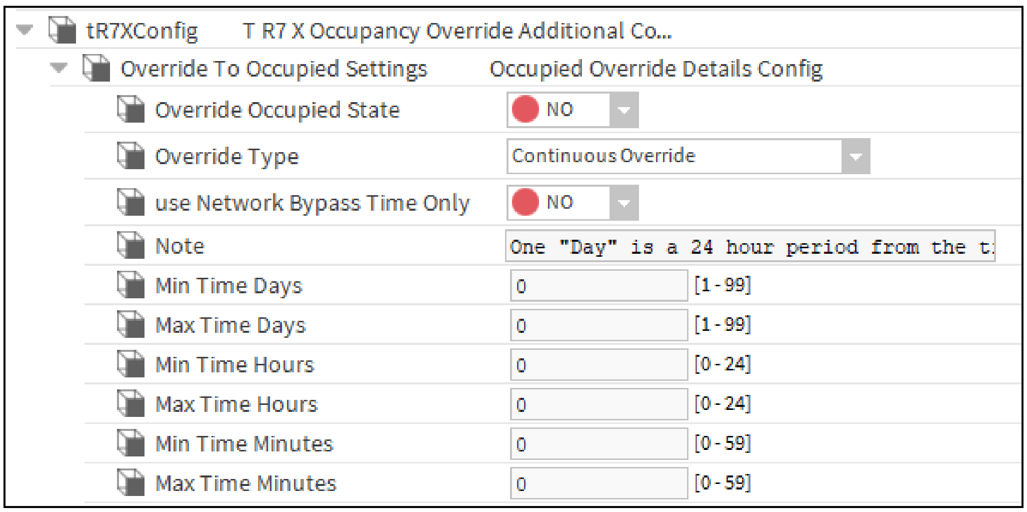



Figure 221: Sub-Menu of tR7XConfig

- Override Occupied State: To override the current occupancy state to Occupied mode
- Override Type:
  - Continuous Override: This will keep the system occupancy status in continuous occupied mode irrespective of system schedule
  - Timed Override: To provide the desired override timings in hours and minute format. User can override the system for the desired time by selecting the required days, Hours and minutes values. The Timed Override in Hours (Bypass) option is to override the system for required hours. The Timed Override in Minutes (Bypass) option is to override the system for required minutes.
- Use Network Bypass Time Only: By clicking yes in Use Network Bypass Time Only option disables all other delays option (for example, Min Time Days, and so on). The timed override details will be determined by the programmable controller configuration. This option only applies to Occupancy override settings. Unoccupied and Standby do not use bypass.
- Note: Shows the note One "Day" is a 24-hour period from the time the override is initiated.
- Min Time Days: Provide the Maximum Days for override to occupied mode.
- Max Time Days:  Provide the Minimum Days for override to occupied mode.
- Min Time Hours: Provide the Maximum Hours for override to occupied mode.
- Max Time Hours: Provide the minimum Hours for override to occupied mode.

- Min Time Minutes: Provide the Maximum Minutes for override to occupied mode.
- Max Time Minutes: Provide the Minimum Minutes for override to occupied mode.

- Override To Unoccupied Settings:

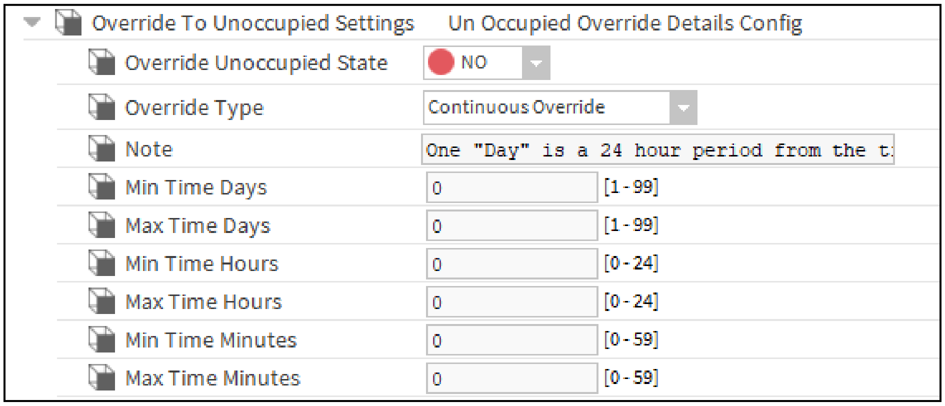



Figure 222: Sub-Menu of tR7XConfig-Un Occupied Override Details Config

- Override Unoccupied State: To override the current occupancy state to Unoccupied mode
- Override Type:
  - Continuous Override: This will keep the system occupancy status in continuous occupied mode irrespective of system schedule
  - Timed Override: To provide the desired override timings in hours and minute format. User can override the system for the desired time by selecting the required days, Hours and minutes values. The Timed Override in Hours (Bypass) option is to override the system for required hours. The Timed Override in Minutes (Bypass) option is to override the system for required minutes.
- Note: Shows the note One "Day" is a 24-hour period from the time the override is initiated
- Min Time Days: Provide the Maximum Days for override to unoccupied mode.
- Max Time Days: Provide the Minimum Days for override to unoccupied mode.
- Min Time Hours: Provide the Maximum Hours for override to unoccupied mode.
- Max Time Hours: Provide the minimum Hours for override to unoccupied mode.
- Min Time Minutes: Provide the Maximum Minutes for override to unoccupied mode.
- Max Time Minutes: Provide the Minimum Minutes for override to unoccupied mode.

- Override To Standby Settings:

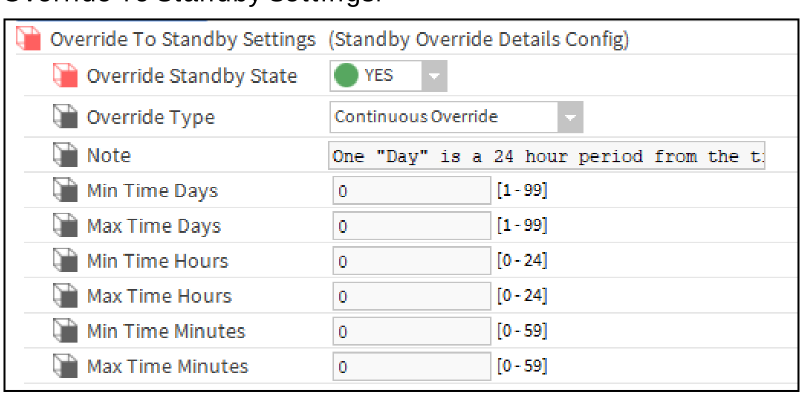


Figure 223: Sub-Menu of tR7XConfig-Standby Override Details Config

- Override Standby State: To override the current occupancy state to Standby mode
- Override Type:
  - ✓ Continuous Override: This will keep the system occupancy status in continuous occupied mode irrespective of system schedule
  - ✓ Timed Override: To provide the desired override timings in hours and minute format. User can override the system for the desired time by selecting the required days, Hours and minutes values. The Timed Override in Hours (Bypass) option is to override the system for required hours. The Timed Override in Minutes (Bypass) option is to override the system for required minutes.
- Note: Shows the note One "Day" is a 24-hour period from the time the override is initiated
- Min Time Days: Provide the Maximum Days for override to Standby mode
- Max Time Days:  Provide the Minimum Days for override to Standby mode
- Min Time Hours: Provide the Maximum Hours for override to Standby mode
- Max Time Hours: Provide the minimum Hours for override to Standby mode
- Min Time Minutes: Provide the Maximum Minutes for override to Standby mode
- Max Time Minutes: Provide the Minimum Minutes for override to Standby mode

o occupancyValues:
To define the values for the respective states.

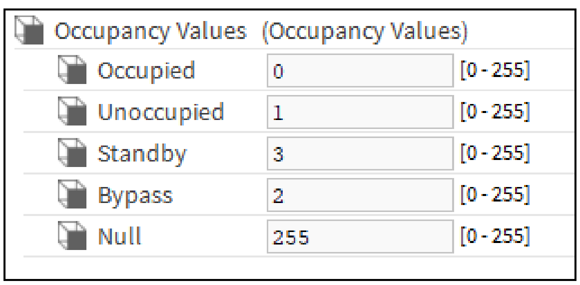



Figure 224: Sub-Menu of tR7X

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

## Occupancy Status Configuration

1. Double-click OccupancyStatus on the wire sheet to configure the properties.

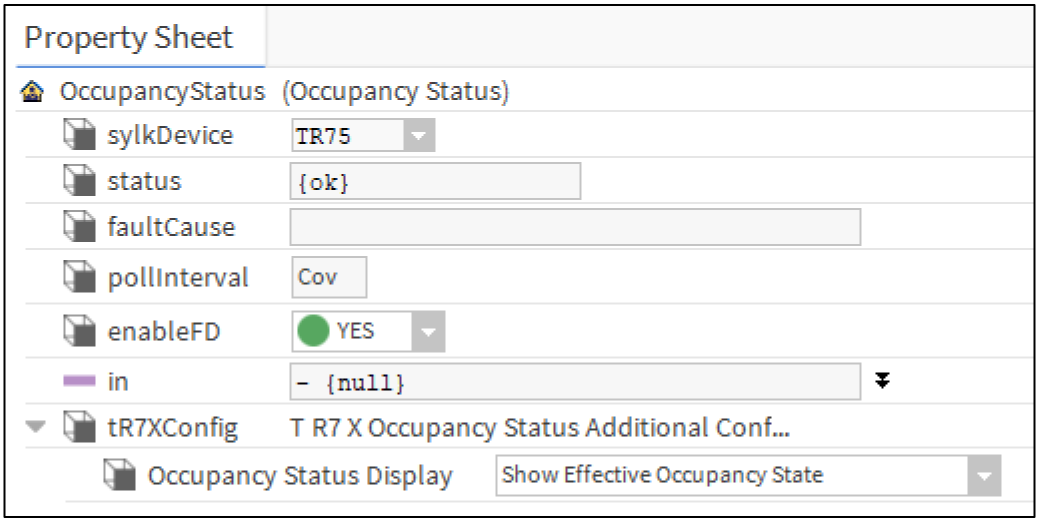



Figure 225: Property Sheet of OccupancyStatus

- sylkDevice: Select the required TR7x or TR4x device from the drop-down menu

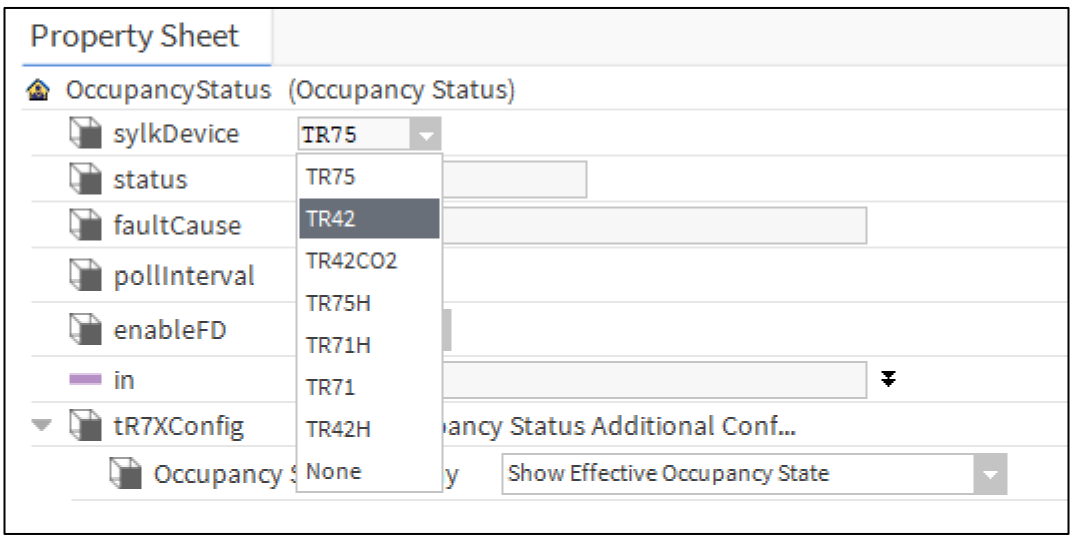



Figure 226: sylkDevice Drop-Down Menu

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists. When there are more than one errors, the fault cause shows only the error which is at the top of the list of errors. If the error is corrected, the next error is shown.
- pollInterval: Poll interval is the time between the end of a timeout period or completion of a network request, and the next request for data on the network
- enableFD: Fail Detect is the time until the CIPer Model 30 is notified of a failure on this point.
    - True: If the parameter has not received an update from the IPC network source in the fail detect time, then an alarm is generated and the presentValue is set to Invalid. Note that fail detect time depends on

the update rate configured.

- False: False means the object retains the last value that was written to it until a IPC network source changes it or the Honeywell CIPer Model 30 has a power outage or reset

- In: To provide the System Occupancy Status to the Wall module.
- tR7XConfig:
  - occupancyStatusDisplay: Select the Display option as per requirement

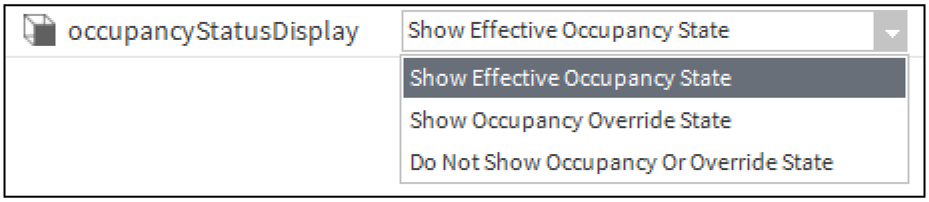



Figure 227: occupancyStatusDisplay Drop-Down Menu

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

## Room Temperature Configuration

1. Double-click ROOMTEMP sensor on the wire sheet to configure the sensor properties.

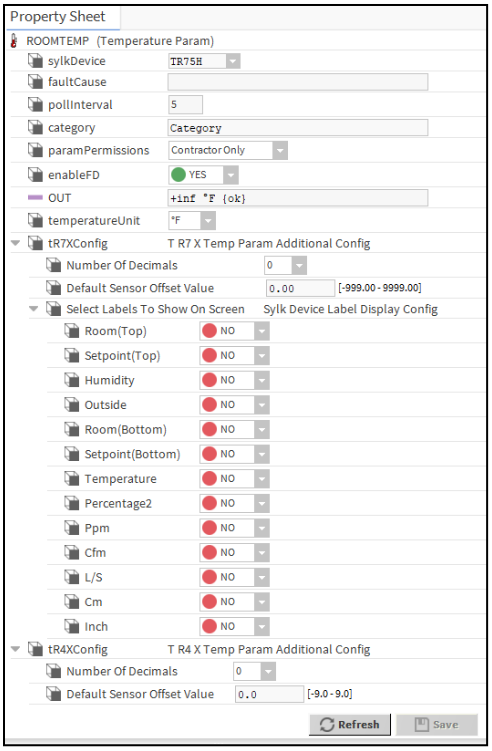



Figure 228: Property Sheet of ROOMTEMP

- sylkDevice: Select the required TR7x or TR4x device from the drop-down menu

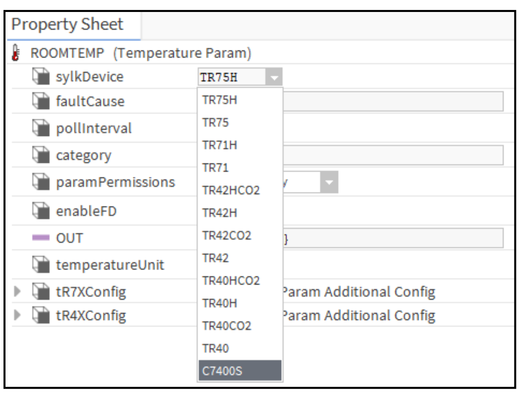



Figure 229: SylkDevices List

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists. When there are more than one errors, the fault cause shows only the error which is at the top of the list of errors. If the error is corrected, the next error is shown.
- pollInterval: Poll interval is the time between the end of a timeout period or completion of a network request, and the next request for data on the network

- category: The category defined by the user while configuring Sylk parameters. By default, it shows Category as category
- paramPermissions: You can select the viewing option for the parameters. Select Contractor Only or Tenant Read Only option from the drop-down of the parameter. If Contractor Only is selected, only the contractor can view the parameters in the wall module. If Tenant Read Only is selected, tenant can view the parameters, but cannot make any changes in the parameters.

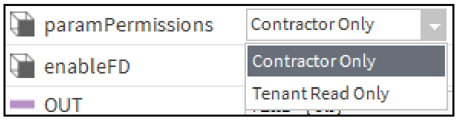



Figure 230:paramPermissions Drop-Down Menu

- enableFD: Fail Detect is the time until the CIPer Model 30 is notified of a failure on this point.
  - True: If the parameter has not received an update from the IPC network source in the fail detect time, then an alarm is generated and the presentValue is set to Invalid. Note that fail detect time depends on the update rate configured.
  - False: False means the object retains the last value that was written to it until a IPC network source changes it or the Honeywell CIPer Model 30 has a power outage or reset
- OUT: Shows the current value of the parameter
- temperatureUnit: Select the unit as Degree F or Degree C as per requirement
- tR7XConfig:
  - Number of Decimals: Enter the decimal accuracy
  - Default Sensor Offset Value: Enter the default value for sensor offset
  - Select Labels To Show On Screen: Select the required option (Room top or Room Bottom) to show the parameter screen.

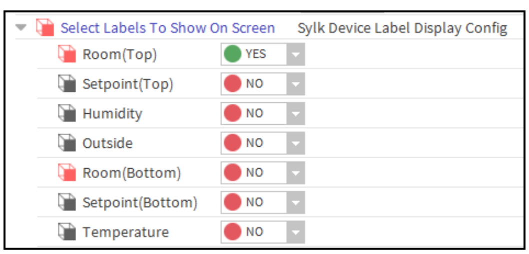



Figure 231: Sub-Menu of tR7XConfig

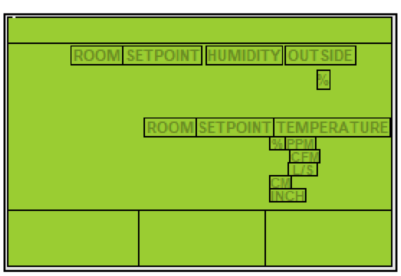



Figure 232: Display of Wall Module TR7X

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

## Sensor Offset Configuration

These are outputs from the wall module such as room setpoint

1. Double-click SensorOffset on the wire sheet to configure the properties.

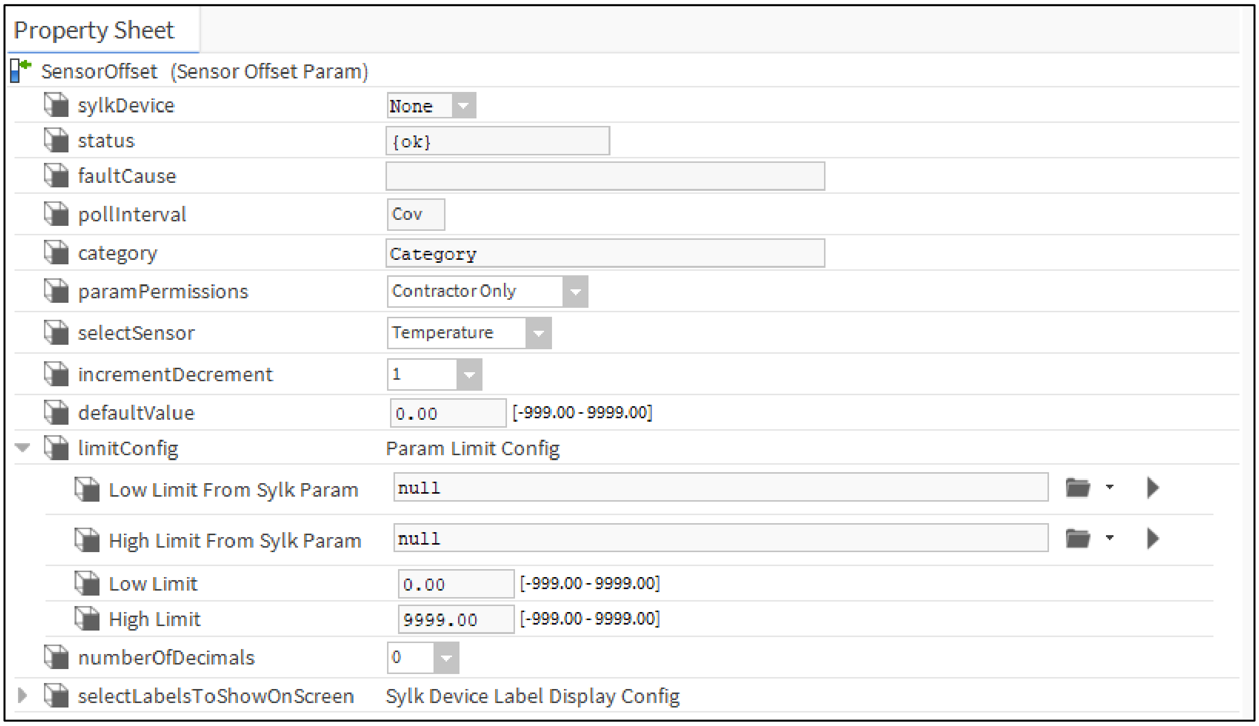



Figure 233: Property Sheet of SensorOffset

- sylkDevice: Select the required TR7x device from the drop-down menu

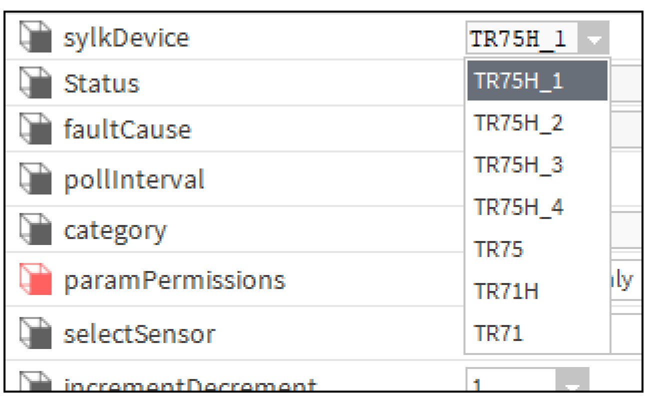



Figure 234: sylkDevice Drop-Down Menu

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists. When there are more than one errors, the fault cause shows only the error which is at the top of the list of errors. If the error is corrected, the next error is shown.
- pollInterval: Poll interval is the time between the end of a timeout period or completion of a network request, and the next request for data on the network

- category: The category defined by the user while configuring Sylk parameters. By default, it shows Category as category
- paramPermissions: User can select the viewing option for the parameter. Select the parameter as viewable by Contractor Only, Tenant Read Only, or Tenant Read Write from the drop-down menu. If Contractor Only is selected, only the contractor can view the parameters in the wall module. If Tenant Read Only is selected, tenant can view the parameters, but cannot make any changes in the parameters. If Tenant Read Write is selected, tenant can view as well as modify the parameters.

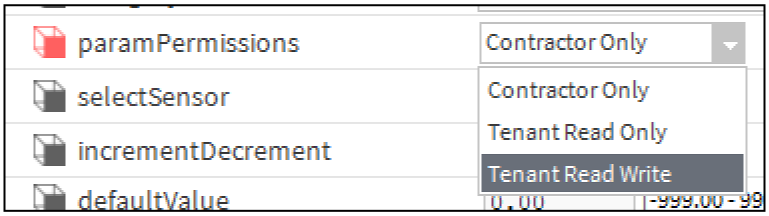



Figure 235: paramPermissions Drop-Down Menu

- sensorSelect: Select the respective sensor.

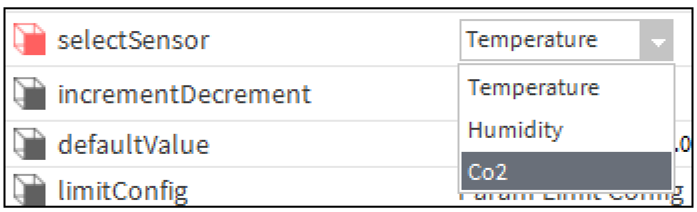



Figure 236: selectSensor Drop-Down Menu

- incrementDecrement: Value of the Offset to be increased or decreased at one step.
- defaultValue: Default value of the Offset.
- limitConfig: To set the high & low limits of the Offset

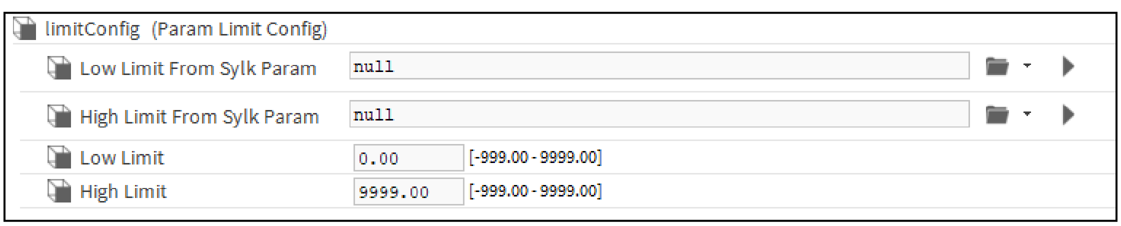
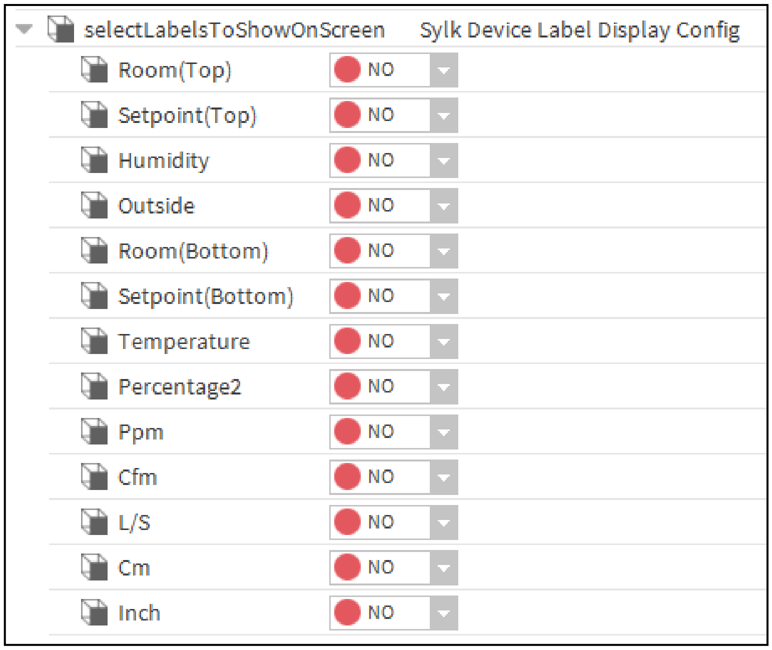
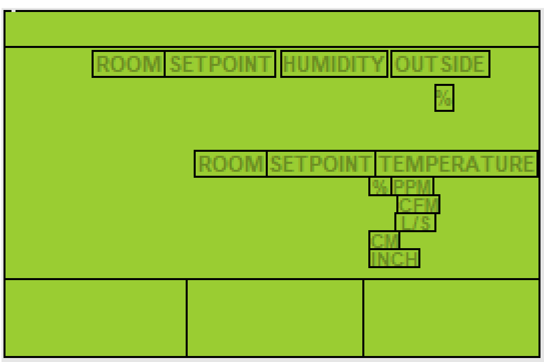



Figure 237: Sub-Menu of limitConfig

  - Low Limit From Sylk Param: To select the Low limit value from the other Sylk parameter of the respective system device
  - High Limit From Sylk Param: To select the High limit value from the other Sylk parameter of the respective system device
  - Low Limit: Set the constant value for High limit
  - High Limit: Set the constant value for Low limit
- numberOfDecimals: Provide the value of numbers of decimal as per requirement.
- selectLabelsToShowOnScreen: Select the required option (Setpoint top or Setpoint Bottom) to show the parameter on the home screen shown below:

- 

- Figure 238: selectLabelsToShowOnScreen Drop-Down Menu
- 

- Figure 239: Display of Wall Module TR7X

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

## Sylk Time Configuration

1. Double-click SylkTime on the wire sheet to configure the properties.

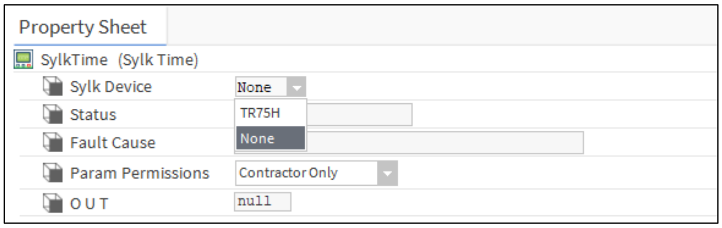



Figure 240: Property Sheet of SylkTime

- sylkDevice: Select the required TR7x device from the drop-down menu

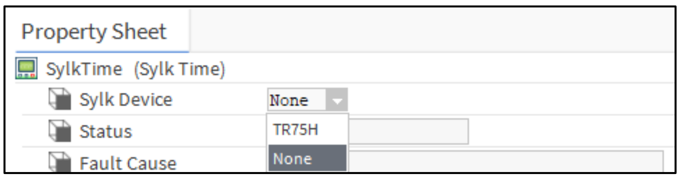
  
  

  Figure 241: sylkDevice Drop-Down Menu

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists. When there are more than one errors, the fault cause shows only the error which is at the top of the list of errors. If the error is corrected, the next error is shown.
- paramPermissions: User can select the viewing option for the parameter. Select the parameter as viewable by Contractor Only, Tenant Read Only, or Tenant Read Write from the drop-down menu. If Contractor Only is selected, only the contractor can view the parameters in the wall module. If Tenant Read Only is selected, tenant can view the parameters, but cannot make any changes in the parameters. If Tenant Read Write is selected, tenant can view as well as modify the parameters.

  Figure 242: paramPermissions Drop-Down Menu

- OUT: To use the time of the wall module.

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

| | Note: |
|---|---|
| User need to add Sylk time to change the IPC Controller time from sylk device and only one sylk time component is allowed per sylk device | |

## System Command Configuration

1. Double-click SystemCommand on the wire sheet to configure the properties.

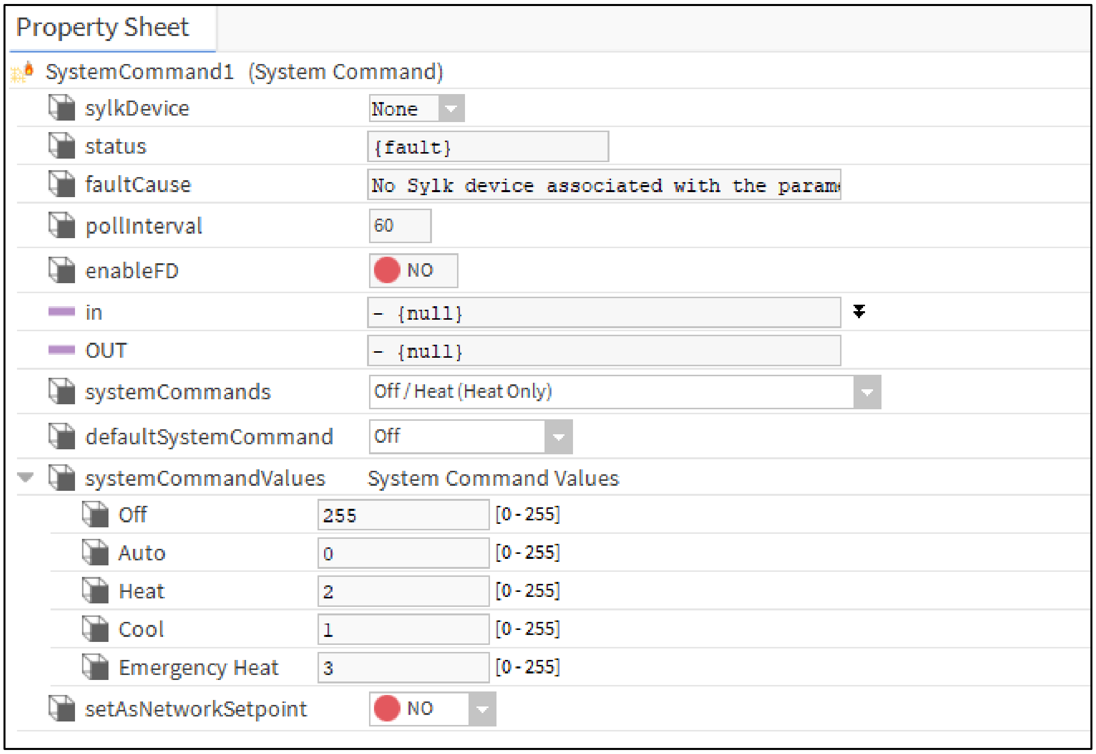



Figure 243: Property Sheet of SystemCommand

- sylkDevice: Select the required TR7x device from the drop-down menu

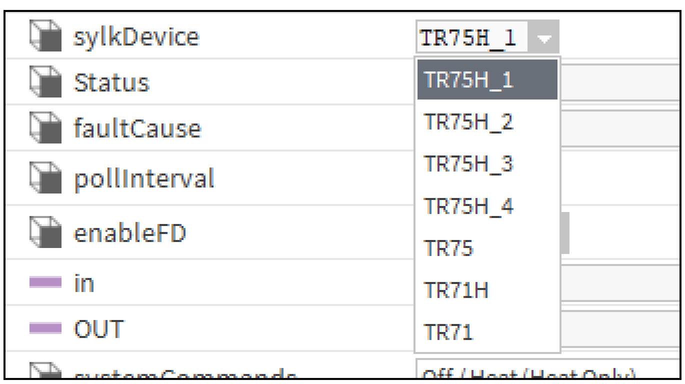



Figure 244: sylkDevice Drop-Down Menu

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists. When there are more than one errors, the fault cause shows only the error which is at the top of the list of errors. If the error is corrected, the next error is shown.
- pollInterval: Poll interval is the time between the end of a timeout period or completion of a network request, and the next request for data on the network
- enableFD: Fail Detect is the time until the CIPer Model 30 is notified of a failure on this point.

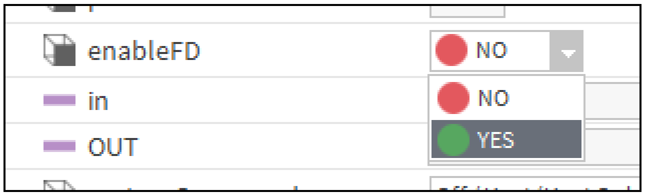



Figure 245: enableFD Drop-Down Menu

- True: If the parameter has not received an update from the IPC network source in the fail detect time, then an alarm is generated and the presentValue is set to Invalid. Note that fail detect time depends on the update rate configured.
- False: False means the object retains the last value that was written to it until a IPC network source changes it or the Honeywell CIPer Model 30 has a power outage or reset

- in: To provide the System Command from the controller to the Wall module.
- OUT: To provide the Final System Command from wall module to the controller
- systemCommands: Select the modes of the command from the wall module

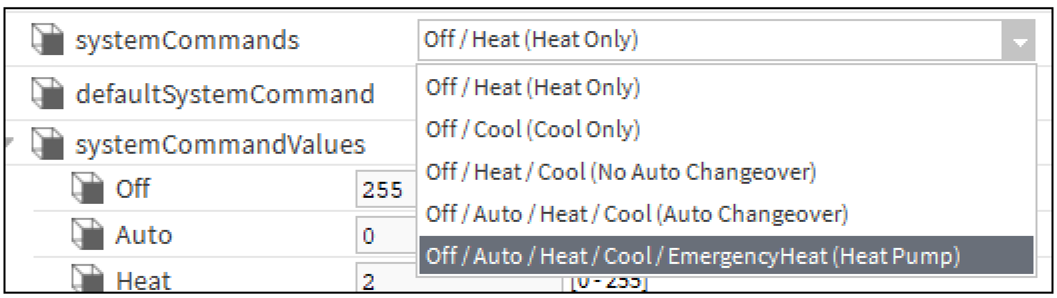



Figure 246: systemCommands Drop-Down Menu

- defaultSystemCommands: Select the Default value of the System Command mode.

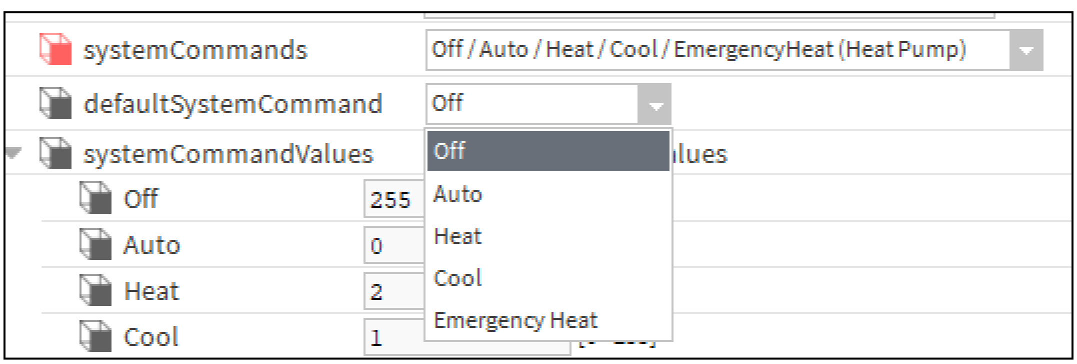



Figure 247: defaultSystemCommands Drop-Down Menu

- systemCommandValues: Set the System Command State Values as per requirement

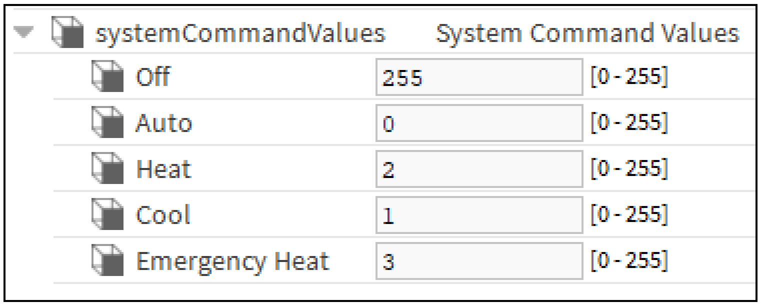



Figure 248: systemCommandValues Drop-Down Menu

- SetAsNetworksetpoint: This option allows you to specify, if the param needs to be configured as network set point or output only param. If you select Yes, then you have access to write and read values, if No selected you only have access to read values from sylk device.

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

## System Status Configuration

1. Double-click SystemStatus on the wire sheet to configure the properties.

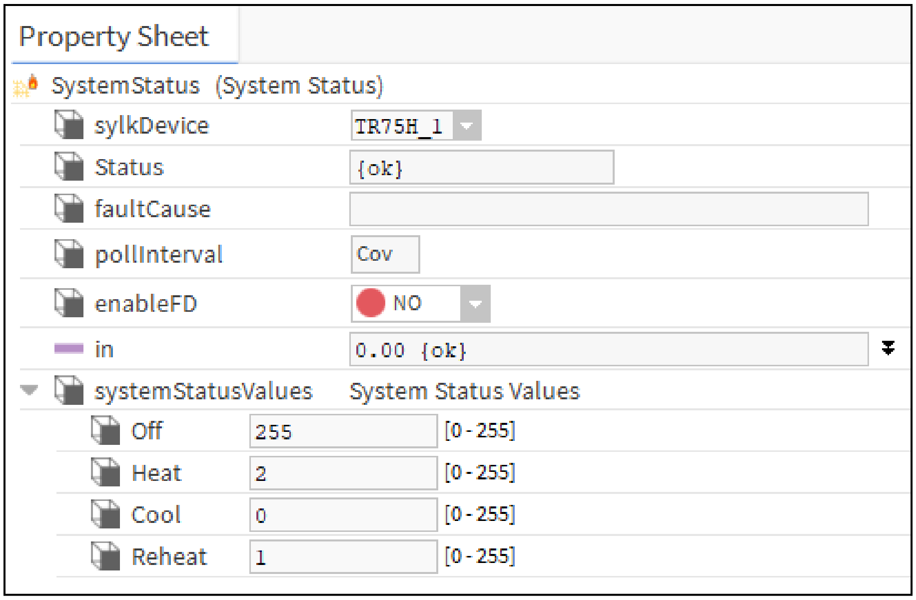



Figure 249: Property Sheet of SystemStatus

- sylkDevice: Select the required TR7x device from the drop-down menu

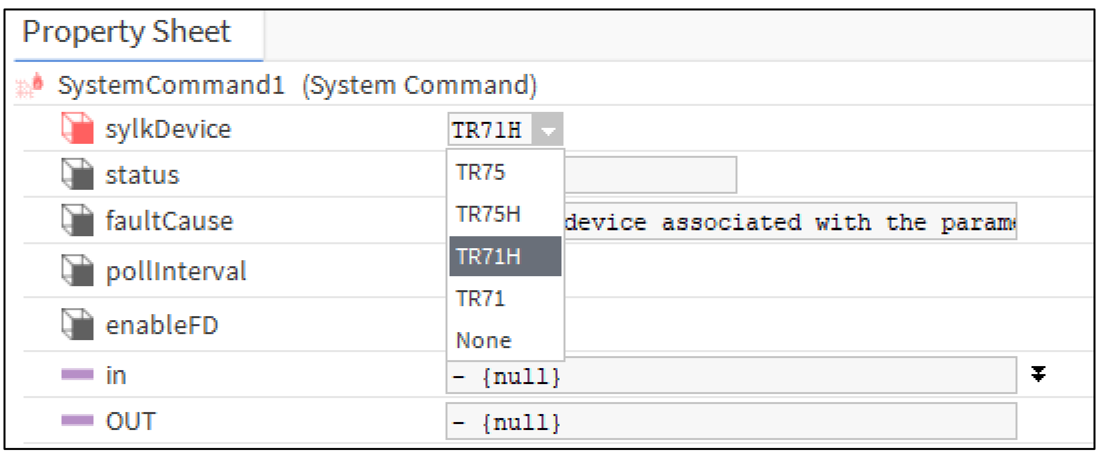



Figure 250: sylkDevice Drop-Down Menu

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists. When there are more than one errors, the fault cause shows only the error which is at the top of the list of errors. If the error is corrected, the next error is shown.
- pollInterval: Poll interval is the time between the end of a timeout period or completion of a network request, and the next request for data on the network
- EnableFD: Fail Detect is the time until the CIPer Model 30 is notified of a failure on this point.
    - True: If the parameter has not received an update from the IPC network source in the fail detect time, then an alarm is generated and the presentValue is set to Invalid. Note that fail detect time depends on

the update rate configured.

  - False: False means the object retains the last value that was written to it until a IPC network source changes it or the Honeywell CIPer Model 30 has a power outage or reset

- in: To provide the system Status from the controller to the Wall module.
- System Status Values: Set the System Sate as per requirement

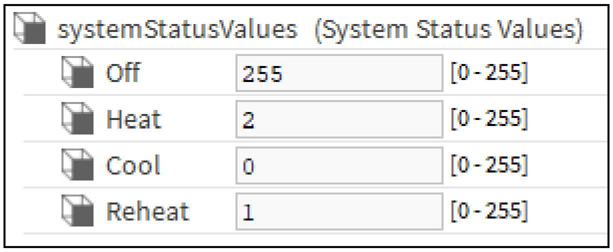



Figure 251: Sub-Menu of systemStatusValues

2. Click Save to save the changes made.

   Or if you do not want to save the changes, click Refresh and then No.

## Time Field Configuration

1. Double-click TimeField parameter on the wire sheet to configure the sensor properties.

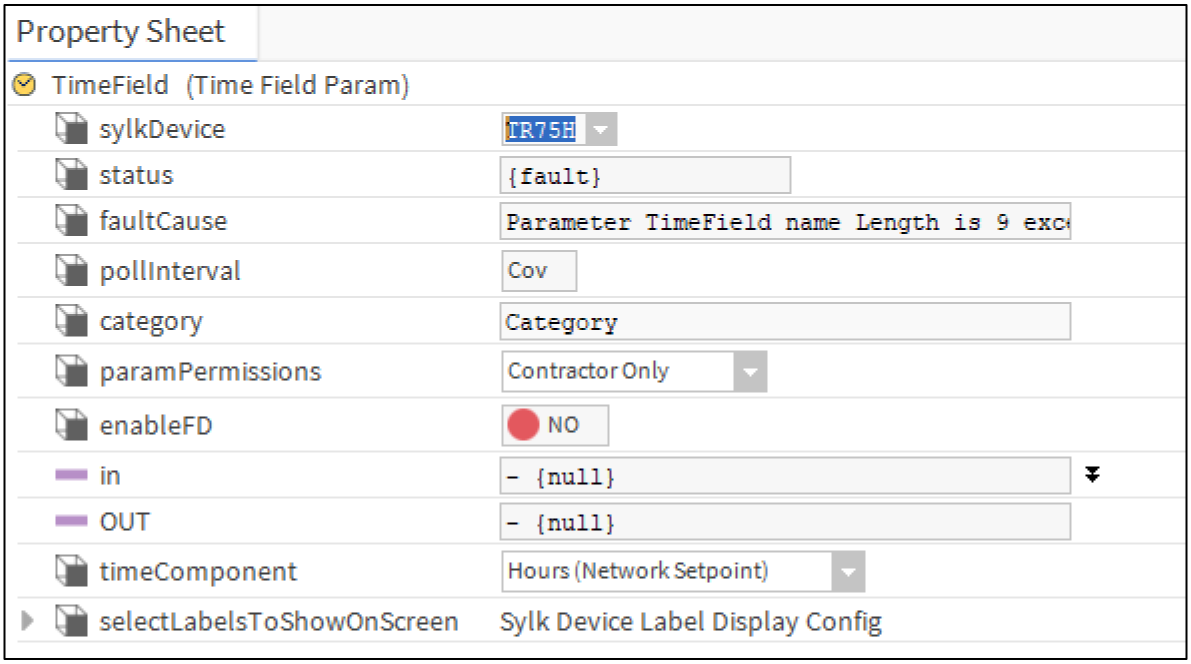
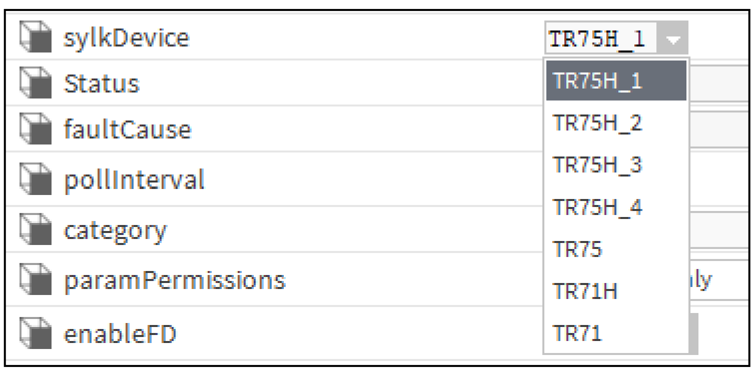



Figure 252: Property Sheet of TimeField

- sylkDevice: Select the required TR7x device from the drop-down menu




  - 
  - Figure 253: sylkDevice Drop-Down Menu
- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists.
- pollInterval: Poll interval is the time between the end of a timeout period or completion of a network request, and the next request for data on the network
- category: The category defined by the user while configuring Sylk parameters. By default, it shows Category as category
- paramPermissions: User can select the viewing option for the parameter. Select the parameter as viewable by Contractor Only or Tenant Read Only from the drop-down menu. If Contractor Only is selected, only the con-

tractor can view the parameters in the wall module. If Tenant Read Only is selected, tenant can view the parameters, but cannot make any changes in the parameters. If Tenant Read Write is selected, tenant can view as well as modify the parameters.

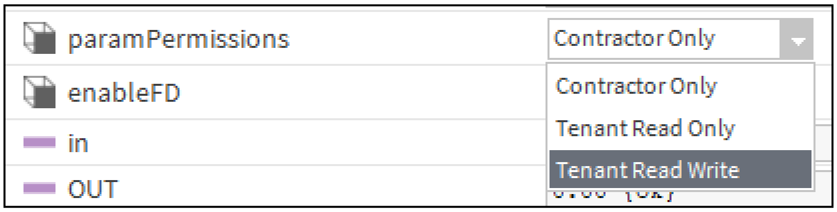



Figure 254: paramPermissions Drop-Down Menu

- enableFD: Fail Detect is the time until the CIPer Model 30 is notified of a failure on this point.

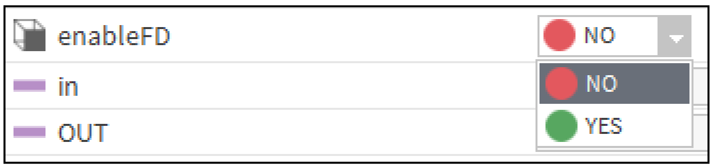



Figure 255: enableFD Drop-Down Menu

  - True: If the parameter has not received an update from the IPC network source in the fail detect time, then an alarm is generated and the presentValue is set to Invalid. Note that fail detect time depends on the update rate configured.
  - False: False means the object retains the last value that was written to it until a IPC network source changes it or the Honeywell CIPer Model 30 has a power outage or reset
- in: To provide the Controller time to the wall module.
- OUT: To provide the wall module time to the controller
- timeComponent: Set the parameter of the time as per requirement

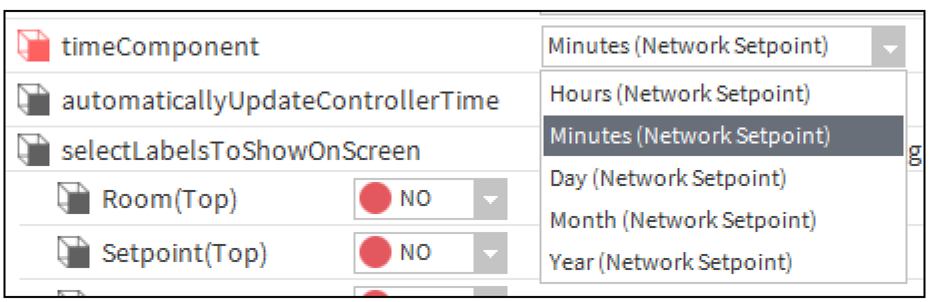
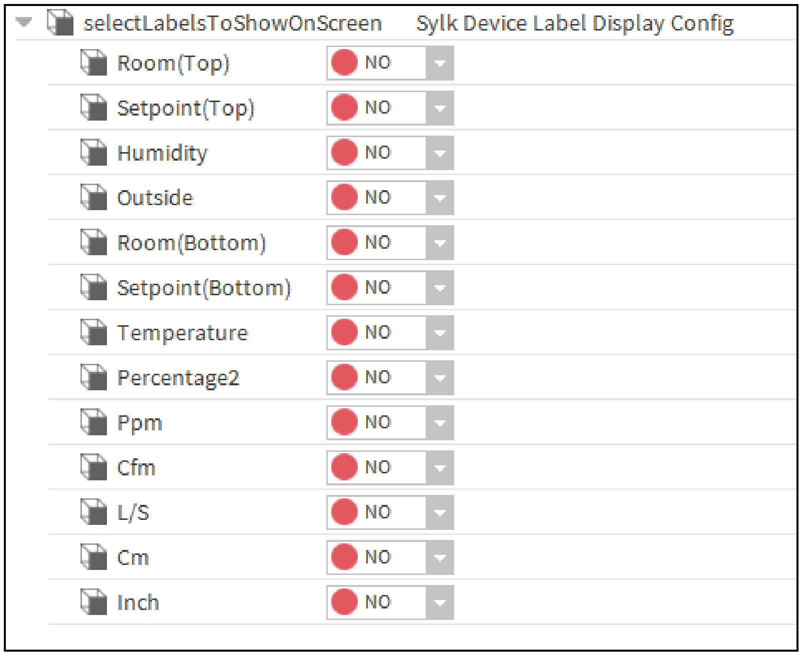
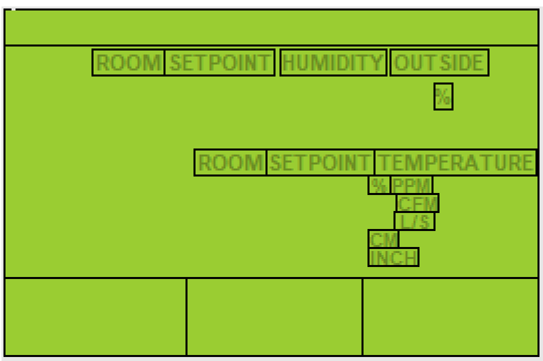



Figure 256: timeComponent Drop-Down Menu

- automaticallyUpdateControllerTime: To update the controller time as per the time of wall module.
- selectLabelsToShowOnScreen: Select the required option to show the parameter on the home screen shown below:




- Figure 257: Sub-Menu of selectLabelsToShowOnScreen




- Figure 258: Display of Wall Module TimeField

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

## Time of Day Configuration

1. Double-click TimeOfDay on the wire sheet to configure the properties.

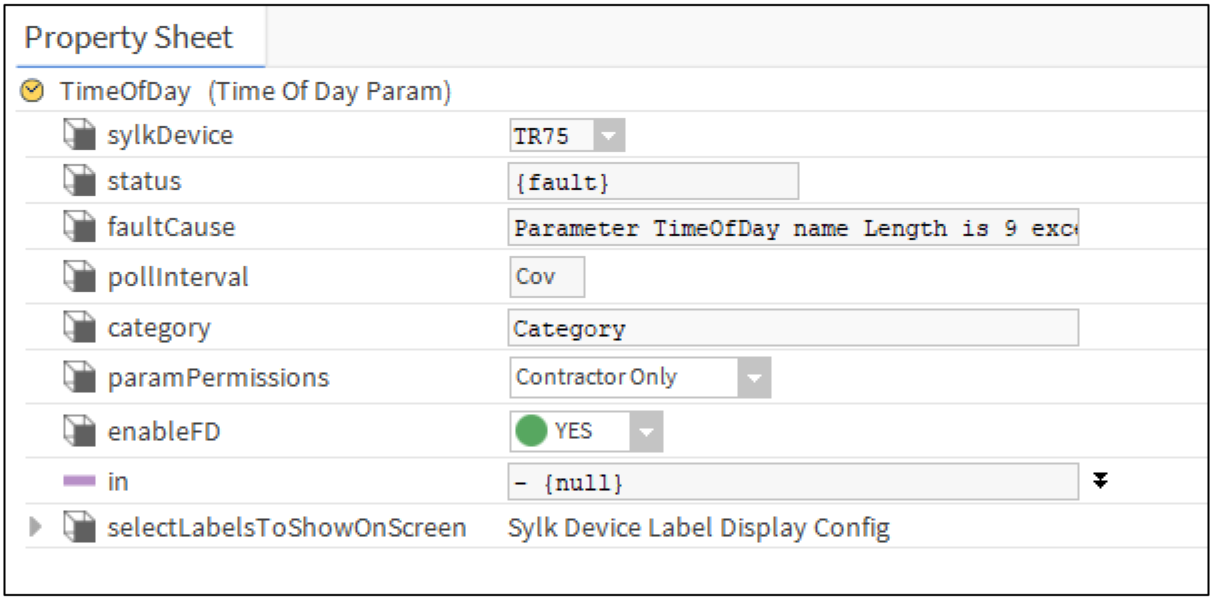
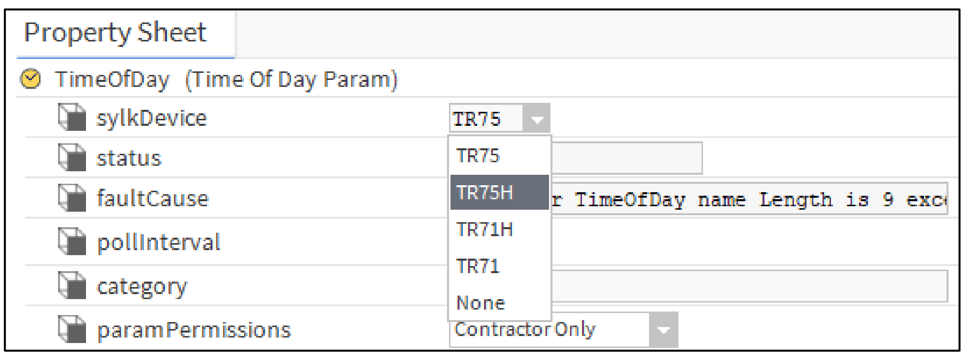


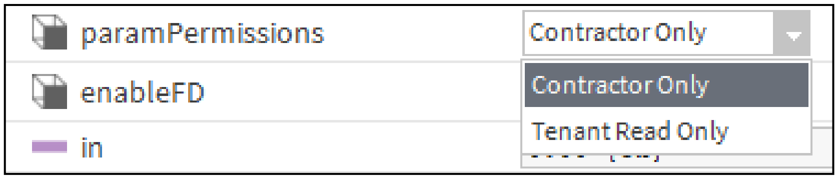

Figure 259: Property Sheet of TimeOfDay

- sylkDevice: Select the required TR7x device from the drop-down menu



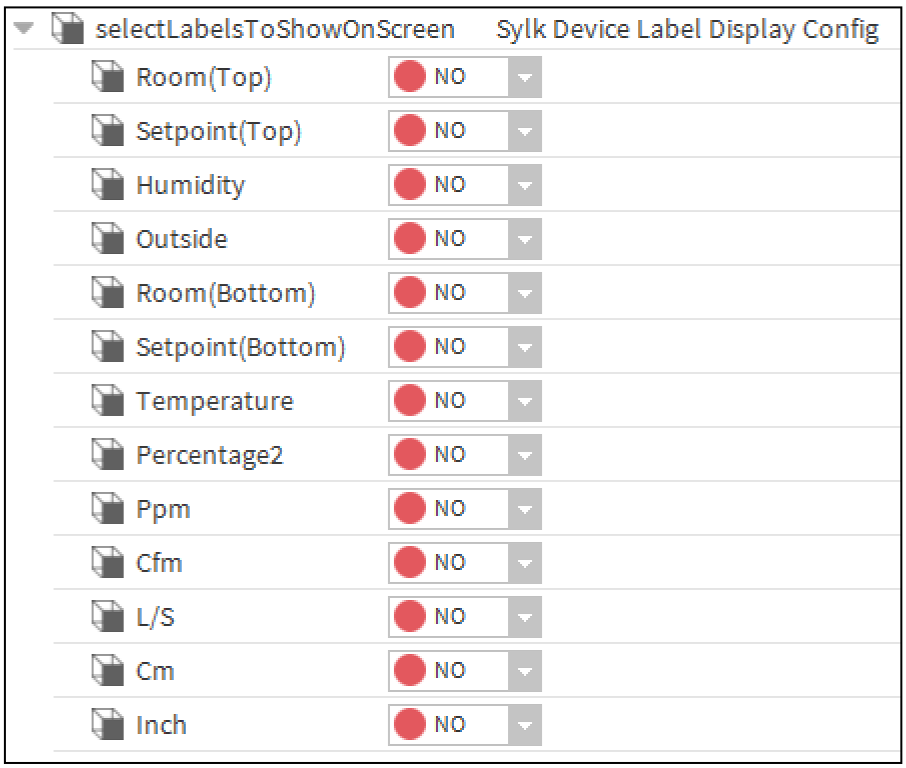

- Figure 260: sylkDevice Drop-Down Menu

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists.
- pollInterval: Poll interval is the time between the end of a timeout period or completion of a network request, and the next request for data on the network
- category: The category defined by the user while configuring Sylk parameters. By default, it shows Category as category
- paramPermissions: User can select the viewing option for the parameter. Select the parameter as viewable by Contractor Only or Tenant Read Only from the drop-down menu. If Contractor Only is selected, only the contractor can view the parameters in the wall module. If Tenant Read Only is selected, tenant can view the parameters, but cannot make any changes in the parameters.

- 

- Figure 261: paramPermissions Drop-Down Menu

- enableFD: Fail Detect is the time until the CIPer Model 30 is notified of a failure on this point.
  - True: If the parameter has not received an update from the IPC network source in the fail detect time, then an alarm is generated and the presentValue is set to Invalid. Note that fail detect time depends on the update rate configured.
  - False: False means the object retains the last value that was written to it until a IPC network source changes it or the Honeywell CIPer Model 30 has a power outage or reset
- in: To provide the system time to the Wall module.
- selectLabelsToShowOnScreen: Select the required option to show the parameter on the home screen shown below:

- 

- Figure 262: Sub-Menu of selectLabelsToShowOnScreen

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

## Value From Wall Module Configuration

These are outputs from the wall module such as sensor values.

1. Double-click ValueFromWallModule on the wire sheet to configure the properties.

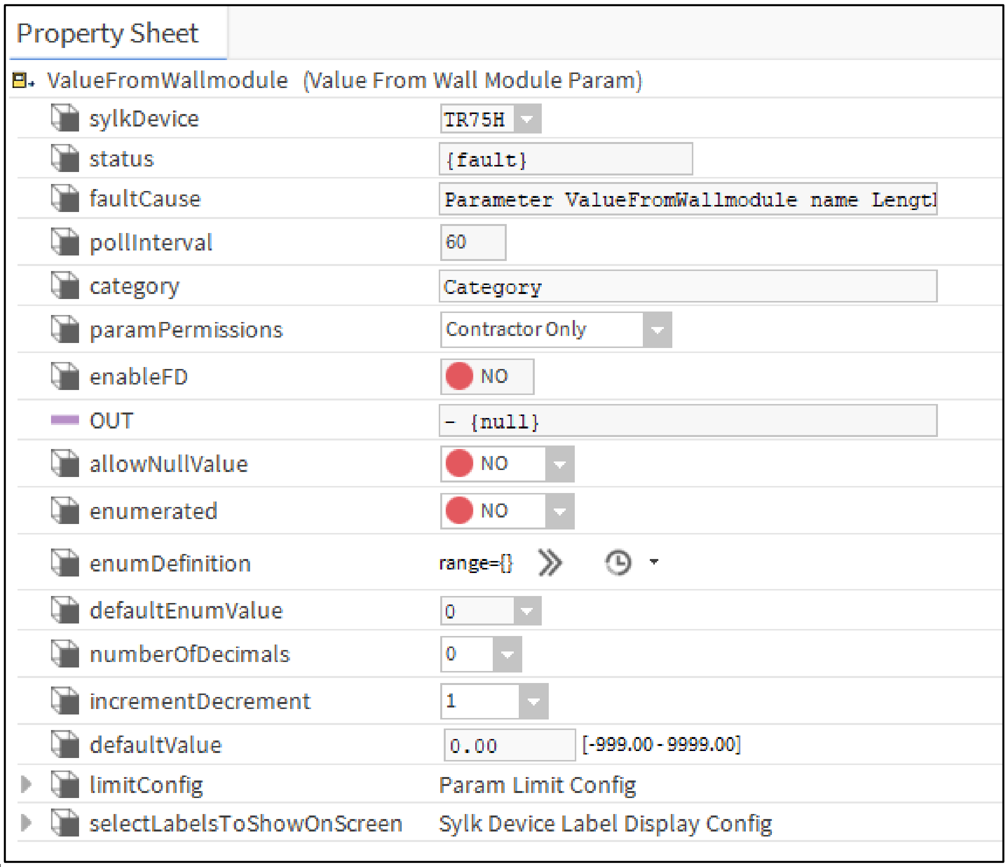



Figure 263: Property Sheet of ValueFromWallModule

- sylkDevice: Select the required TR7x device from the drop-down menu.

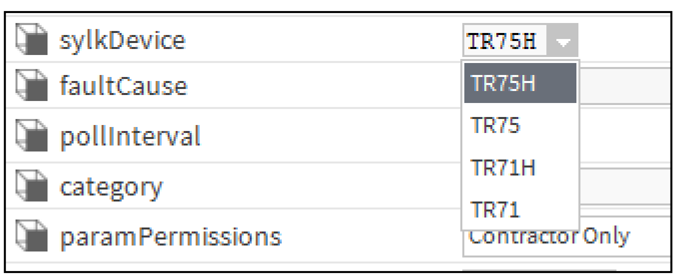



Figure 264: sylkDevice Drop-Down Menu

- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point. Indicates the reason why the parameter is in fault. This property is empty unless a fault exists. When there are more than one errors, the fault cause shows only the error which is at the top of the list of errors. If the error is corrected, the next error is shown.
- pollInterval: Poll interval is the time between the end of a timeout period or completion of a network request, and the next request for data on the network

- category: The category defined by the user while configuring Sylk parameters. By default, it shows Category as category
- paramPermissions: User can select the viewing option for the parameter. Select the parameter as visible by Contractor Only, Tenant Read Only, or Tenant Read Write from the drop-down menu. If Contractor Only is selected, only the contractor can view the parameters in the wall module. If Tenant Read Only is selected, tenant can view the parameters, but cannot make any changes in the parameters. If Tenant Read Write is selected, tenant can view as well as modify the parameters.


Figure 265: paramPermissions Drop-Down Menu

- enableFD: Fail Detect is the time until the CIPer Model 30 is notified of a failure on this point.
  - True: If the parameter has not received an update from the IPC network source in the fail detect time, then an alarm is generated and the presentValue is set to Invalid. Note that fail detect time depends on the update rate configured.
  - False: False means the object retains the last value that was written to it until a IPC network source changes it or the Honeywell CIPer Model 30 has a power outage or reset
- Out: Provides the respective value to the controller from the wall module
- allowNullValue: This option allows the user to provide Null value
- enumerated: This option allows the user to provide Enum Setpoints
- enumDefinition: If the Enumerated is set to true then user can set the multiple states of the Enum setpoint.

  a) To set the Enum states of the Setpoint click on the >> button.


Figure 266: enumRange Parameter

  b) Set the states Ordinal & Display as per requirement. For example,

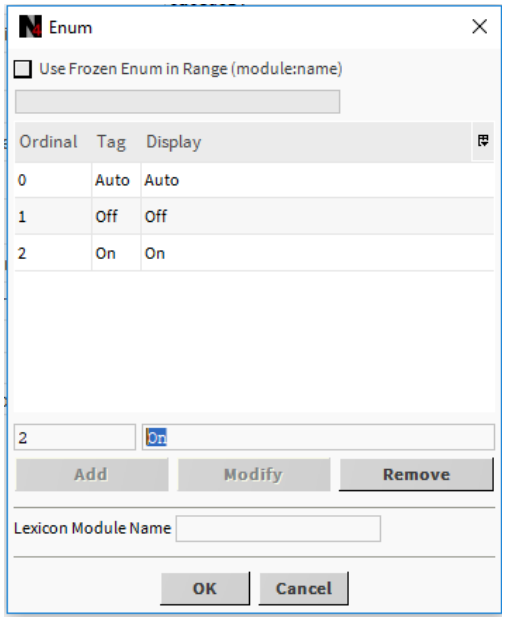



Figure 267: Enum Window

c) Click Ok to save.

- defaultEnumValue: This option displays list of default Enum value from the defined enum range.
- numberOfDecimals: Provide the value of numbers of decimal as per requirement.
- incrementDecrement: Value of the setpoint to be increased or decreased at one step.
- defaultValue: Default value of the setpoint.
- limitConfig: To set the high and low limits of the setpoint.

  If you provide the Low Limit From Sylk Param value, the value given in the Low Limit field is ignored.

  If you provide the High Limit From Sylk Param value, the value given in the High Limit field is ignored.

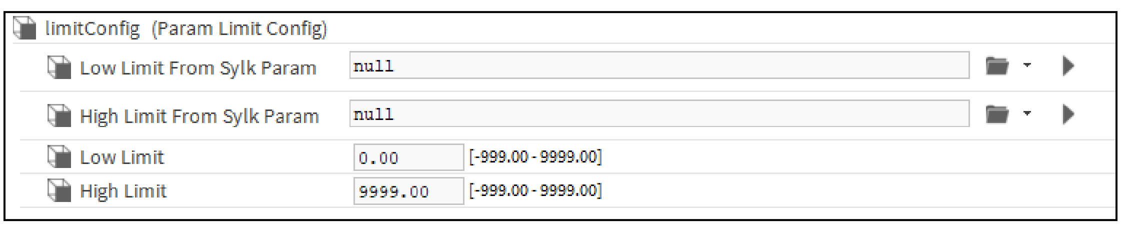



Figure 268: Prpoerty Sheet of limitConfig

  - Low Limit From Sylk Param: To select the Low limit value from the other Sylk parameter of the respective system device
  - High Limit From Sylk Param: To select the High limit value from the other Sylk parameter of the respective system device
  - Low Limit: Set the constant value for High limit
  - High Limit: Set the constant value for Low limit

- selectLabelsToShowOnScreen: Select the required option (Setpoint top or Setpoint Bottom) to show the parameter on the home screen shown below:

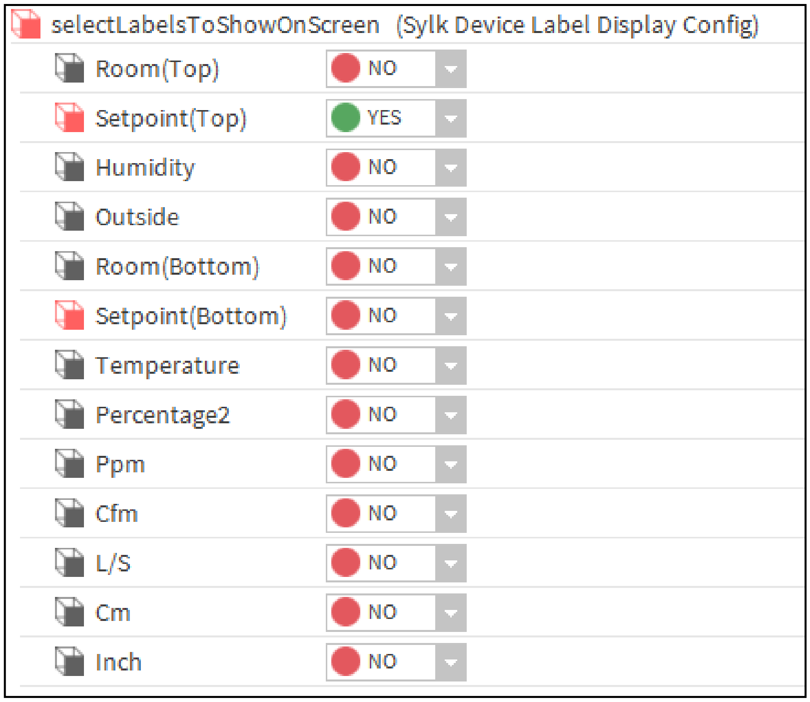



Figure 269: Sub-Menu of selectLabelsToShowOnScreen

2. Click Save to save the changes made.

   Or click Refresh and then No, if you do not want to save the changes.

## SylkActuatorInputParam Configuration

1. Double-click SylkActuatorInputParam on the wire sheet to configure the properties.

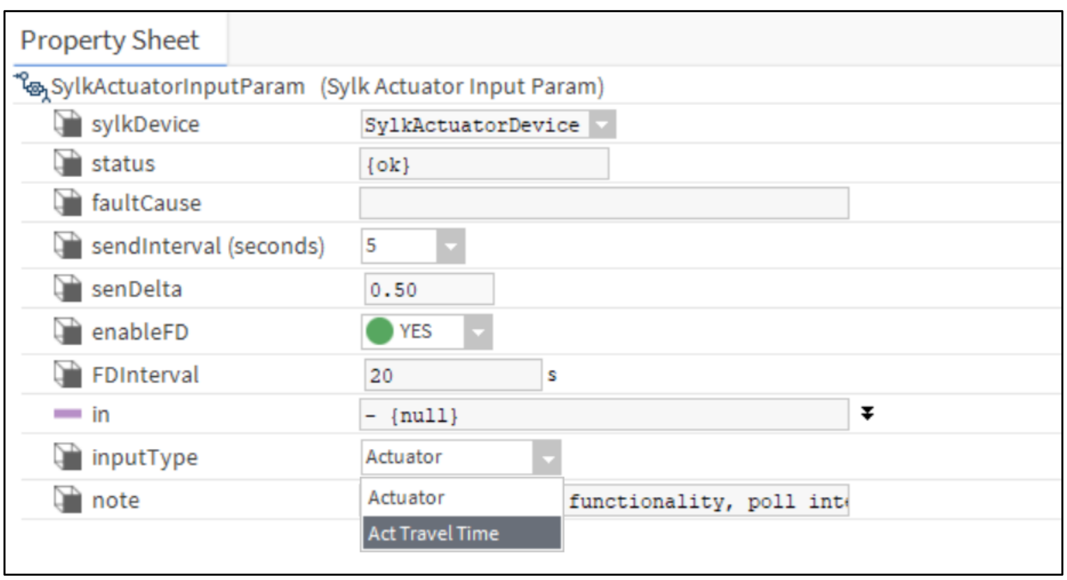



Figure 270 : Property of SylkActuatorInputParam

- sylkDevice: Select the required SylkActuator device from the drop-down menu
- Status: Read-only point. Shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.

- faultCause: Read-only point, displays the reason of the fault parameter. This property is empty unless a fault exists.
- sendInterval (seconds): sendInterval is the time interval for sending data from the controller to the network periodically with the configured sendInterval. Based on the user configured sendInterval (in seconds), the controller will send the input parameter value to the Sylk device.

  Example: If sendInterval value set to 5 seconds then in every 5 seconds, the controller will send input parameter value to the Sylk device.

  Available interval range can be configured from 5 seconds to 75 seconds. The default sendInterval value is 5 seconds.

  If sendInterval is set to Cov (change of value), the controller will send input parameter value to Sylk device, only when there is change in value on the input parameter. GPU (guaranteed periodic update) feature gets added with Cov selection. To avoid loss of data due to high network traffic, at every 60 secs controller updates input parameter value to the Sylk device.

| Note: |
|---|
| • If the Sylk device is configured at the higher address, then there is possibility that Sylk device will miss to read parameter periodic update on the network. It is recommended to set enableFD to "No." (If enableFD is set to "Yes" there is chance, that the Sylk parameter value will be set to default if it fails to receive parameter update which may impact its behavior).<br>• Use Cov only for critical points. Cov uses Sylk bandwidth. Excessive use of bandwidth can cause loss of data between Sylk devices. |

- senDelta: The delta value for Significant Event Notification is also known as SenDelta. It allows to propagate the input parameter value from controller to Sylk device, only when the difference between last sent value and current value is greater than or equal to senDelta. Otherwise it will not propagate the change of value to the Sylk device.

  Example: Let's assume senDelta is set to 2, last sent input parameter value is 2 and current value is 5. The difference between parameter values is 3 (5-2=3), which is greater than senDelta value (i.e. 2). This allows controller to send current input parameter value (i.e. 5) to the Sylk device.

  It is introduced to control increase in network traffic due to high frequency of change of value on input parameters. This can be used to tune the controller such that small changes of value are not propagated to the network. The default senDelta value is 0.5.

| Note: |
|---|
| • senDelta feature only applicable, if the sendInterval set to "Cov"<br>• A senDelta of zero means that any change of value is considered significant and is propagated to the network. |

- enableFD: Fail Detect is the time until the controller is notified of a failure on this point.
  - Yes: If enableFD set to "yes" then Sylk device expects periodic update on input parameter. If it fails to receive parameter update from the controller, then alarm will be generated and the Sylk input parameter value will be set to default value.
  - No: False means the parameter retains the last value that was written to it. It will set the value upon receiving updated values from the controller. It will not allow parameter value to be set to default value.
- FDInterval: FDInterval for Sylk actuator input parameter is 20 secs.

| Note: |
|---|
| To ensure proper functionality, sendInterval must be less than one-third of Fail Detect interval (if Fail Detect is enabled). |

- in: To provide the value from the controller to the Sylk device.
- inputType: This option allows you to select Sylk Actuator input. SylkActuatorInputParam can be configured into two type of input Actuator and Act Travel Time
  - Actuator: Actuator command comes from controller output. An actuator moves as per the command.
  - Actuator Travel Time: This is the stroke time or time required to travel from full close position to fully open position.

## SylkActuatorOutputParam Configuration

1. Double-click SylkActuatorOutputParam on the wire sheet to configure the properties.

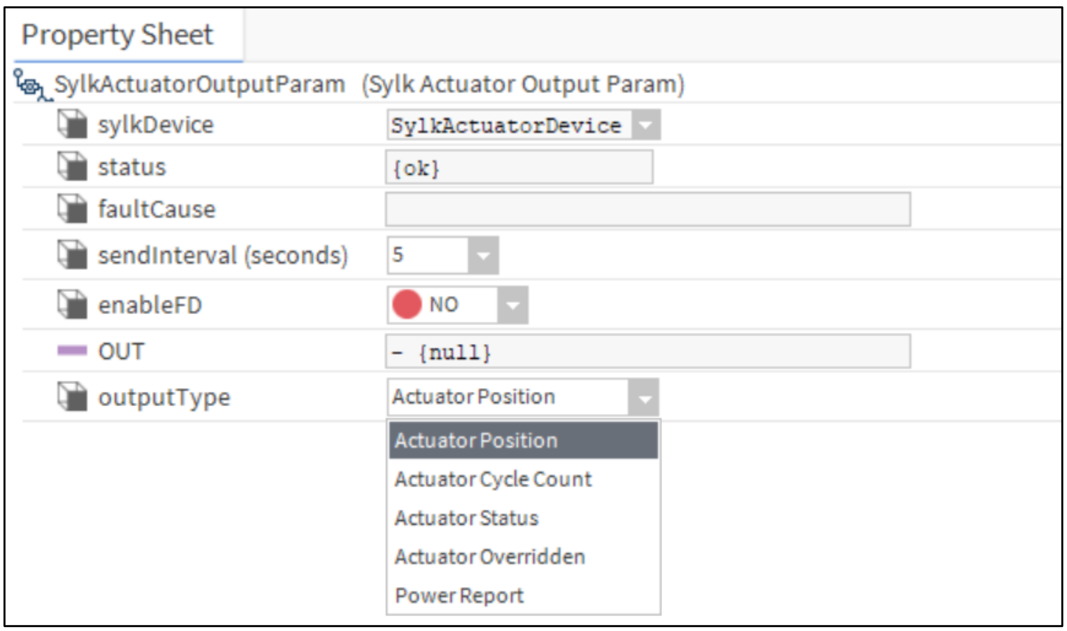



Figure 271: Property of SylkActuatorOutParam

- sylkDevice: Select the required SylkActuator device from the drop-down menu.
- Status: Read-only point, shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details see Sylk Component Status Behaviors.
- faultCause: Read-only point, displays the reason of the fault parameter. This property is empty unless a fault exists.
- sendInterval (seconds): sendInterval is the time interval for sending output parameter value from Sylk device to the network periodically. Based on the user configured sendInterval the Sylk device will send the output parameter value to the controller.

  Example: If sendInterval value set to 5 seconds then in every 5 seconds, the Sylk device will send output parameter value to the network.

Available interval range can be configured from 5 seconds to 75 seconds. The default sendInterval value is 5 seconds.

| Note: |
|---|
| • If the Sylk device is configured at the higher address, then there is possibility that Sylk device will miss to write parameter periodic update on the network. It is recommended to set enableFD to "No." (If enableFD is set to "Yes" there is chance, that the Sylk parameter value will be set to default if it fails to send parameter update which may impact its behavior.) <br> • Cov as sendInterval not supported when configuring for SylkActuatorOutputParam |

- enableFD: Fail Detect is the time until the controller is notified of a failure on this point.
  - Yes: If enableFD set to "yes" then controller expects periodic update on Sylk output parameter. If it fails to receive parameter update from the Sylk device, then alarm will be generated and the Sylk output parameter value will be set to default value.
  - No: False means the parameter retains the last value that was written to it. It will not allow parameter value to be set to default value.
- Out: Provides the respective value to the controller from the Sylk device.
- outputType: This option allows you to configure Sylk Actuator output. SylkActuatorOutputParam can be configured in the following type of outputs:
  - Actuator Position: This is a feedback from the actuator's current position ranging between 0 to 100 and can be mapped into the controller for monitoring.
  - Actuator Cycle Count: It shows how many time actuators is cycled from open to close position.
  - Actuator Status: Actuator status returns a number with the following meaning:
    - No Error
    - Under Voltage
    - Over Voltage
    - Stall
    - OverVoltage and Stall
    - UnderVoltage and Stall
  - Actuator Overridden: It provides actuator overridden status if actuator is overridden externally. This will return "True" if the unit is in Test Mode. This when the address pot is manually turned to position 6.
  - Power Report: This is a relative measurement. It is only calculated for the last commanded move, value ranging between 0 to 100.

# Event-Based Programming

In this wire sheet, the value of the output changes only if there is a change in the input values. The function block gets executed based on the order of the changes in the input values, that is, there is no specific order for executing the function blocks. This helps reduce the execution time and controller resource usage, which helps in fast response.

## Event-Based Execution

The event-based execution is more efficient when there is a conditional logic (if, then, or else) that does not involve feedback control loops, and where a quick response is required. However, based on how the event-based execution is deployed, system loading, and engine's priority, this execution provides faster control response to the events. The occasions, when there are too many events to execute or high processing demand from other system components in a heavily loaded system, you may notice slower response than average. This is an efficient method for a not-overloaded and low-noisy system, where supervisory services, data analysis, and event-based logic is the major concern.

However, for the program with feedback control loops, without sequential execution and periodic rate, the effective gain that is used to tune the loop is inconsistent and can result in random instability. Also, if the inputs to the loop are noisy, the loop responds to the noise events and not to the average signal value.

| Note: |
|---|
| CIPer Model 30 programming model function blocks which are under ipcProgrammingTool palette will not work in event-based control program. |

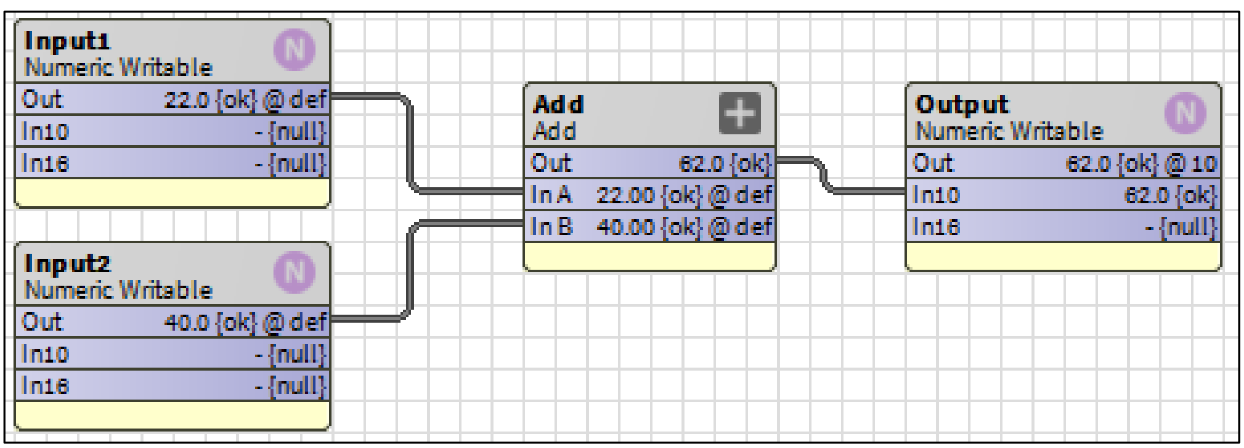



Figure 272: Add Function Block on Event Control Program Wire Sheet without Execution Order

# Sequential Programming

In this wire sheet, the value of the output changes continuously based on the execution of the function block in an ascending order. This prevents the race conditions in logic.

## Sequential Execution

The sequential execution of the function blocks consumes more consistent amount of processing power than the event-based execution, and eventually becomes less efficient, because all the logic of the function blocks runs every cycle even if there are no changes occurred. So, sequential programming should be used for the applications, which need execution to be controlled in an ascending order. For example, two function blocks A and B have execution order 1 and 2 respectively. So, the function block A is executed first, and then B.

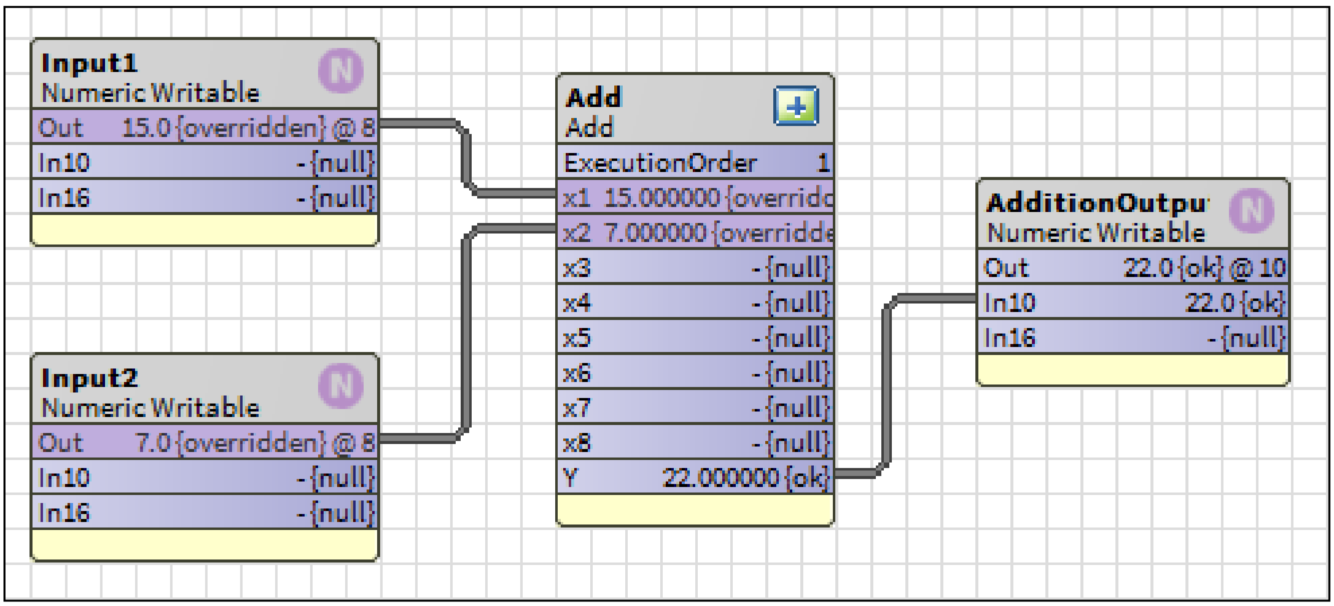



Figure 273: Add Function Block on Sequenced Control Program Wire Sheet with Execution Order

Note:

- Whenever required, you can change the execution order of the function blocks.
- When you add various function blocks onto the wire sheet for programming, the application starts executing the function blocks at the same time. If you want the application to start the execution only after you complete adding and linking the function blocks, you can do so by selecting the Request Sequenced Control Engine Stop or Start Sequenced Control Engine options available on right-clicking the Sequenced Control Program folder in the Nav tree and selecting Actions.

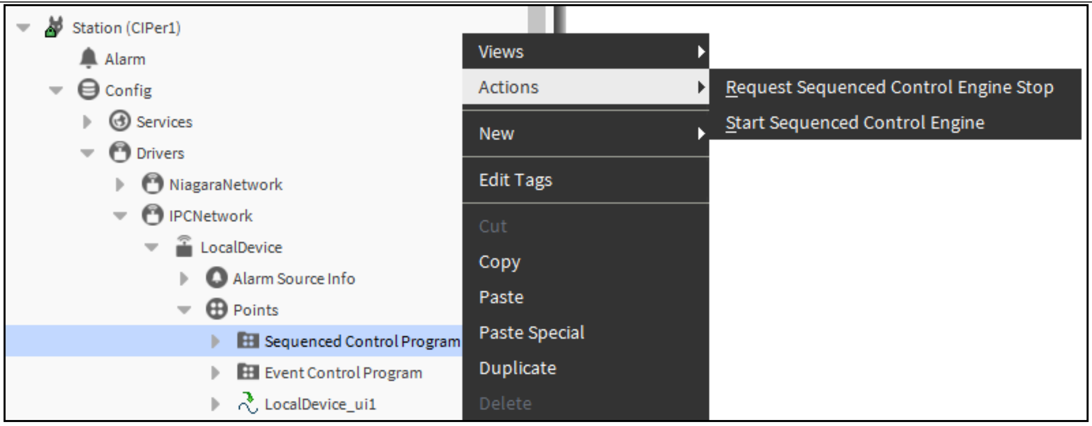



Figure 274: Request Sequenced Control Engine Stop and Start Sequenced Control Engine Options

# Function Block Library

Function blocks are the library of the objects used to implement any custom application logic for wide variety of HVAC applications. A function block has inputs and outputs. A Function block receives inputs from physical inputs, network inputs, or from output of another function block. The function block processes the received input data and produces an output. Processing depends upon the function block type.

All function blocks are available in the ipcProgrammingTool palette in the WEBStation N4 application. The function block for CIPer Model 30 also provides offline expansion I/O configuration support, which helps you to connect more inputs and outputs.

## Common Behavior Overview

Following are the common behaviors, which are applicable for all function blocks.

### Execution Time

The execution time displays the default time (write synchronization) that is continuous 5-minute Interval. If required you can adjust this, or set to Daily or Manual.

### Status

It shows the component status at the last check.

### Facets

It gives the facets in use by the parent proxy point.

### Out Save

The Out Save feature saves the value of the component which was last saved. Suppose, you have enabled the Out Save feature. The output value is 50 and it is saved, and later the output value changes to 60, which is not yet saved. In this case, if the controller stops functioning, the Out Save displays the output value as 50, because 50 was the value which was saved last. If the Out Save feature is disabled, the output is some garbage value.

### Function Blocks Details

Function blocks are classified into following categories:

- Analog Function Blocks
- Control Function Blocks
- Logic Function Blocks
- Math Function Blocks
- DataFunction Blocks
- ZoneControl Blocks
- BuiltIn
- Utils Functional Blocks

Any of these objects except SylkDevices can be dragged onto the wire sheet of SequencedControlProgram. SylkDevices need to be dragged and dropped onto the LocalDevice folder. Make the connections between physical points, software points, and function blocks to create a Control Program or an Application.

Following figure shows the function blocks and objects of the CIPer Model 30 Control Application.

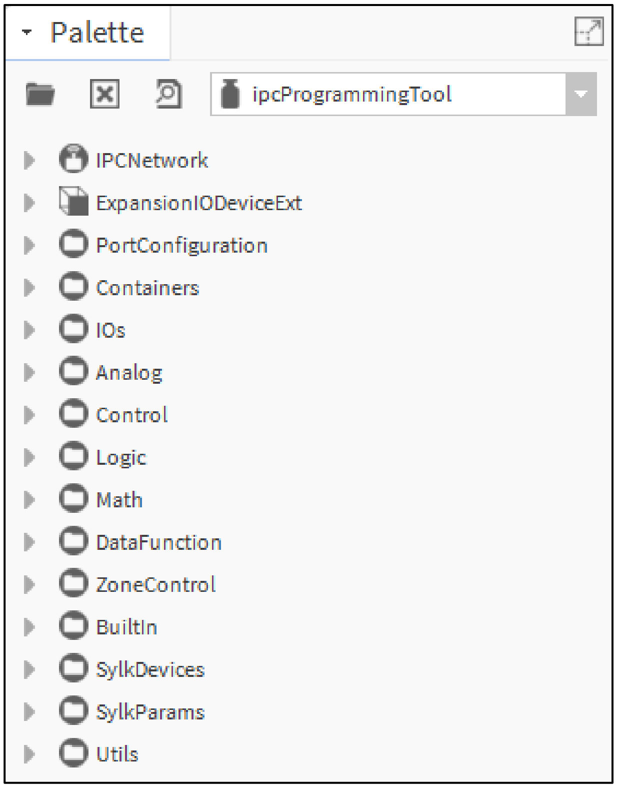



Figure 275: ipcProgrammingTool Palette

As shown in the above figure, the ipcProgrammingTool palette contains the following items:

- IPCNetwork: IPCNetwork to communicate with CIPer device
- ExpansionIODeviceExt: Adds Expansion IO Devices
- PortConfiguration: Configures the Network Port.
- Containers: Contains folder to add program logic and I/O modules
- IOs: Configures the floating motor output
- Analog: Analog function block
- Control: Control function block
- Logic: Logic function block
- Math: Math function block
- DataFunction: DataFunction block
- ZoneControl: Zone Arbitration function block
- BuiltIn: BuiltIn function block
- SylkDevices: Contains all types of Sylk Modules
- SylkParams: Contains all possible default parameters for Sylk Devices

- SylkSchedule: Contains schedule for Sylk Devices
- Utils: Utility function block

You need to add logic for CIPer Model 30 controller programming model in SequencedControlProgram and EventControlProgram folders. SequencedControlProgram acts same as JACE logic, whereas the EventControlProgram is event-based, that is you need to provide the Iteration Interval for each block for execution.

You need to use CIPer Model 30 programming model for SequencedControlProgram, and Kit Control for EventControlProgram. To add Boolean Writable, Numeric Writable, and Enum Writable, you need to use Kit Control.

Various capabilities provided in a function block are:

- Add Function Block
- Configure Function Block
- Override Output of Function Block
- Clear Overridden Output of Function Block
- Delete Function Block
- Remove non-required Pin Slots

## Adding a Function Block

You can add new function block with Add Function Block feature.

To add a function block:

1. Navigate to the ipcProgrammingTool palette. If the ipcProgrammingTool palette is not visible on the left pane, on the Menu bar, select Window > Side Bars > Palette. The ipcProgrammingTool palette is displayed.
2. Navigate to IPCNetwork > Local Device > Points in the Nav tree and double-click the SequencedControlProgram or EventControlProgram. The wire sheet is displayed at the right-side area of the screen.
3. Drag and drop the required function block from the ipcProgrammingTool palette onto the wire sheet.
4. Enter the name of the function block and click Ok. The function block is added and it appears on the wire sheet.

## Configuring a Function Block

You can modify the properties of a function block with the Configure Function Block feature.

To configure a function block:

1. Add the required function block onto the wire sheet of an Application Logic, Program, or Macro. See Add Function Block section for more details.
2. Double-click the required function block on the wire sheet and the AX Property Sheet of the function block is displayed.
3. Enter the required information in the available input fields. The input fields differ for each function block.
4. Click Save to save the changes. The Save button is enabled when there is a change in at least one of the input fields. Or click Refresh on the property sheet, and then Yes on the dialog box to save the changes.
5. Click Refresh after changing the input field values to revert to the last saved settings and click No on the dialog box. Or click Cancel on the dialog box to return to the property sheet view. See the following example.

**Note:**

You can also configure a function block by double-clicking the required function block in the palette pane. The AX Property Sheet of the selected function block is displayed at the right-side area of the screen.

Example:

1. Navigate to the Maximum function block in the Analog function block folder.
2. Navigate to IPCNetwork > Local Device > Points in the Nav tree and double-click SequencedControlProgram or EventControlProgram. The wire sheet is displayed on the right-side area of the screen.
3. Drag and drop the Maximum function block onto the wire sheet of SequencedControlProgram or EventControlProgram.
4. Double-click the function block. The property sheet is displayed along with the parameter values as shown in.

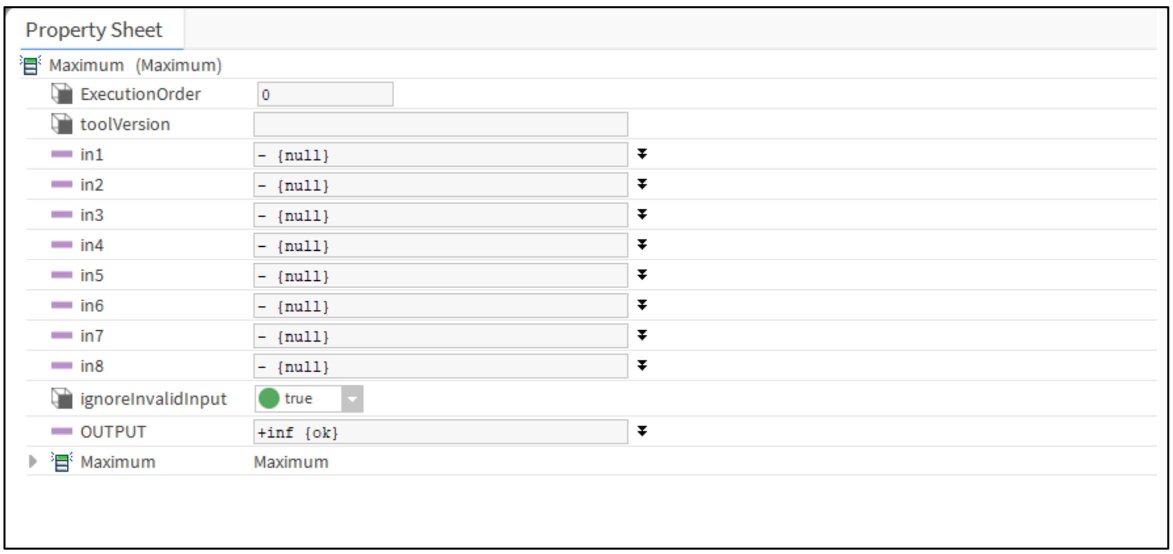



Figure 276: Property Sheet of Maximum Function

You can modify the properties which are editable.

## Overriding Output of a Function Block

You can override the output value of the function blocks and provide the duration for which you want to override the output.

To override output of a function block:

1. Right-click the required function block present on the SequencedControlProgram wire sheet.

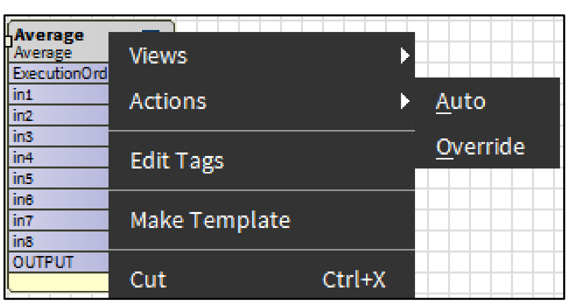


Figure 277: Override Option in Actions Menu of Function Block

2. Select Actions and then click Override. The Override window is displayed.

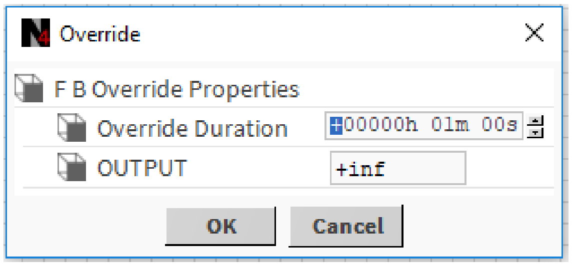



Figure 278: Override Window

3. Enter the duration for which you want to override the function block in the hours-minutes-seconds format. Use the (up and down arrows) to increase or decrease the duration respectively.
4. Enter the value to override the output in the OUTPUT input field.
5. Click Ok to save the override properties.

   Or

   Click Cancel, if do not want to save the changes.

**Note:**

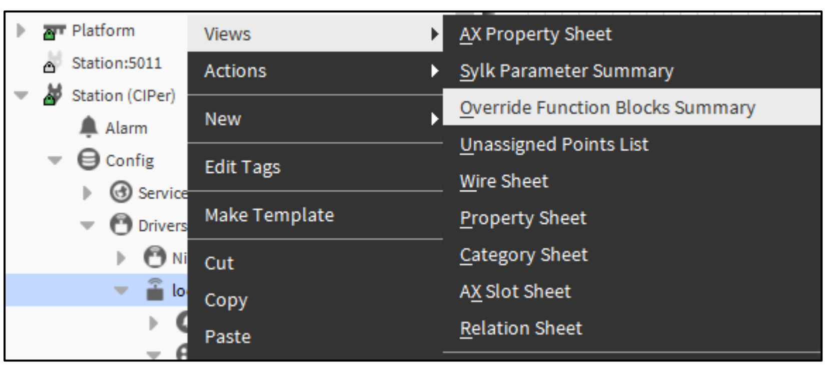
- Once the duration of the override is passed, the Output automatically changes to Auto.
- To view the list of function blocks that are overridden, right-click LocalDevice, click Views, and then Override Function Blocks Summary.
- 


  Figure 279: Override Function Blocks Summary Option in View Menu of LocalDevice

## Clearing Overridden Output of a Function Block

To clear overridden output of a function block:

1. Right-click the required function block present on the SequencedControlProgram wire sheet.

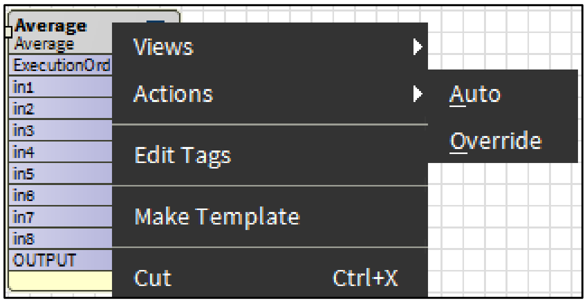



Figure 280: Override Option in Actions Menu of Function Block

2. Select Actions and then click Auto. The overridden value of the output is no more applicable, and the function block is not executed to calculate the output.

Note:

You can also clear the overridden outputs of multiple function blocks at a time by right-clicking LocalDevice, clicking Actions, and then Clear Function Blocks Override.

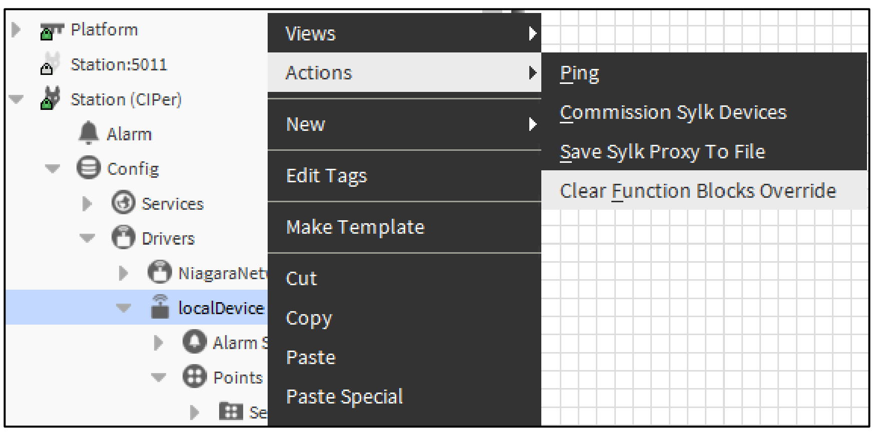



Figure 281: Clear Function Blocks Override in Actions Menu of LocalDevice

## Deleting a Function Block

You can delete a function block from the wire sheet with Delete option.

To delete a function block:

3. On the wire sheet, select the required function block.
4. Click the Delete button on the keyboard or right-click the function block and select Delete. The function block is deleted from the wire sheet along with bindings to it, if any.

## Removing a Non-Required Pin Slot

Every function block has inputs and outputs. Many times, all these inputs and outputs are not required in the logic. In such situations, you can remove unwanted pin slots of the function block.

To remove unwanted pin slots from the function block:

1. Right-click on the function block and select Pin Slots. The Pin slots window appears.
2. Click the pins which are not required and click OK. The respective function block appears on the wire sheet without the removed pin slots.

Example: As shown in the following figure, AND function block with only two inputs connected, in1 and in2. From in3 to FalseDelay inputs are not utilized.

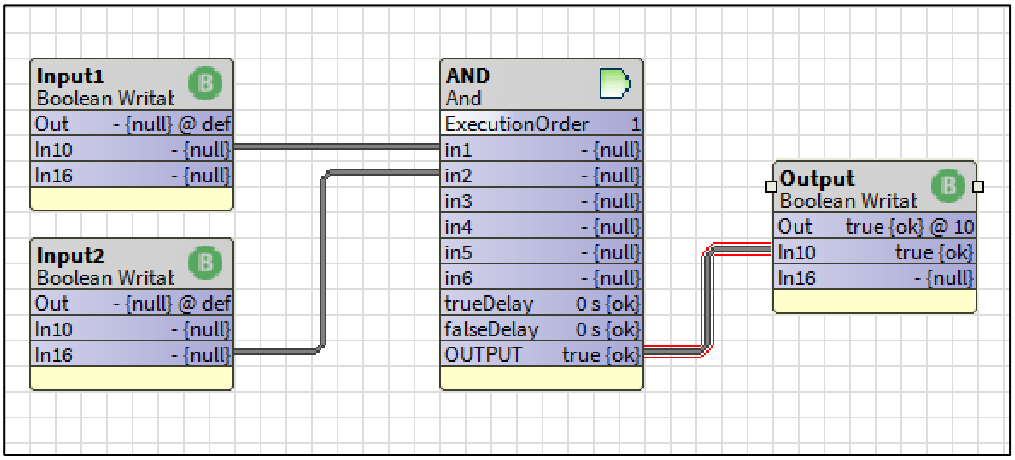



Figure 282: AND Function Block with All Inputs

After removing the non-required pin slots, the logic appears as shown in the following figure.

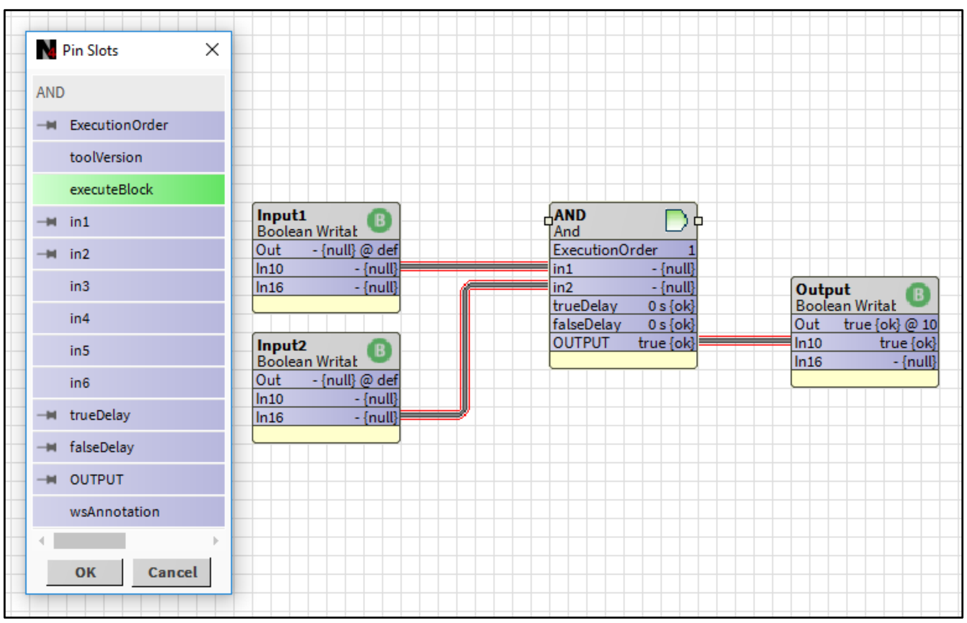



Figure 283: AND Function Block with Required Inputs

# Analog Function Blocks

The CIPer Model 30 programming model provides the following analog function blocks that you can configure and use to build the application logic:

- AnalogLatch
- Average
- Compare
- DecisionBox
- Edge
- Encode
- HystereticRelay
- Maximum
- Minimum
- Priority Select
- Select
- Switch

## AnalogLatch

This function block latches the Y output to the value of the X input, when the latch input transitions from FALSE to TRUE. The output remains the same and is retained until the next transition happens from FALSE to TRUE. At each FALSE to TRUE transition, the output Y is latched to the current X input. The logic inputs for analog latch are as shown in the following table.

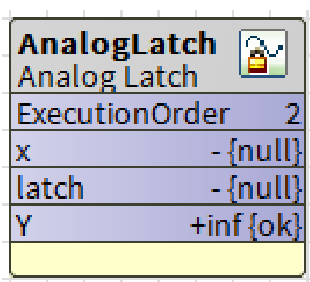



Figure 284: Function Block of AnalogLatch

## Logic Inputs

Table 31: Logic Inputs of Analog Latch

| Input Name | Input Value | Logic Value | Description |
|---|---|---|---|
| latch | Unconnected | 0 | Output remains zero, because there is no input value to cause a latch. |
| | VAL != 0.0 | 1 | Latch the input X to the output on FALSE to TRUE transitions (no negation) |
| | invalid | 0 | Output remains as it was. |

## Analog Inputs

The following table shows the details about analog inputs of an analog latch.

Table 32: Analog Inputs of Analog Latch

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| x | >= – infinity | < + infinity | unconnected | X = invalid |
| | | | invalid | X = invalid |

## Output

Table 33: Output of Analog Latch

| Output Name | Range | Description |
|---|---|---|
| Y | Any floating-point value | The value of X, when the latch input changes from FALSE to TRUE. |

**Note:**

If both the X and latch inputs are unconnected, the output is invalid.

If the input is invalid, the output transits to invalid, when the latch input changes from FALSE to TRUE.

You can negate the latch input to cause a TRUE to FALSE transition to latch X to Y.

In each iteration, the analog latch keeps a track of the last state of the latch input, so that it knows when a FALSE to TRUE transition occurs.

On power up/reset, the last latch value is set to FALSE, regardless of the negation configuration.

Example: An illustration to explain the behavior of the Analog Latch is shown in the following figures.

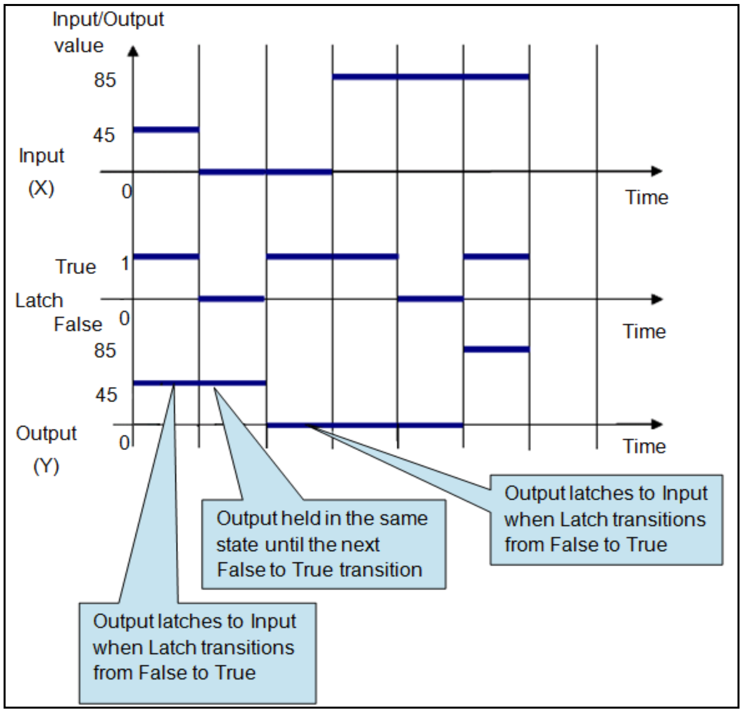
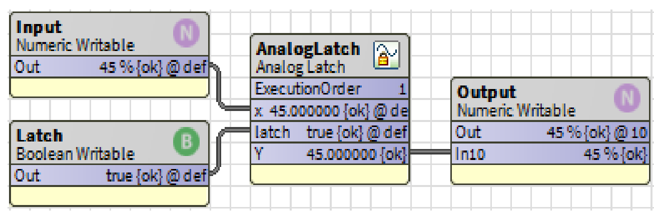
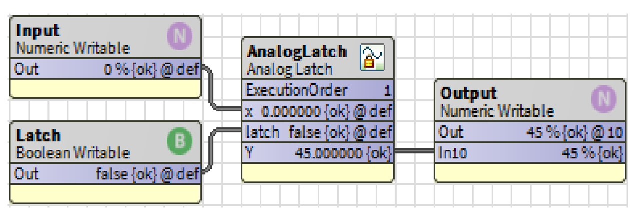
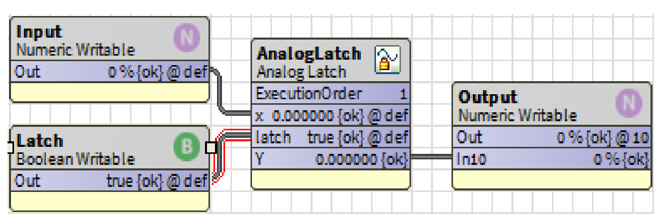
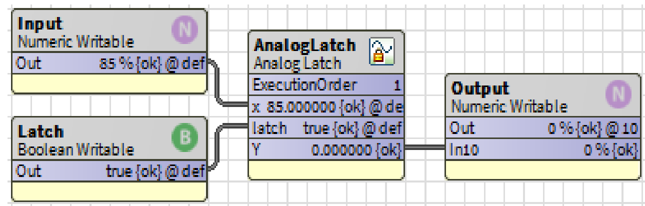
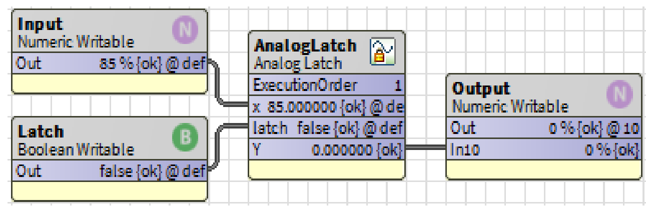
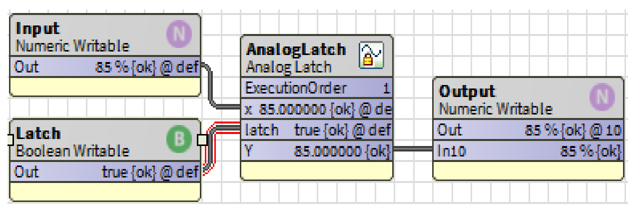


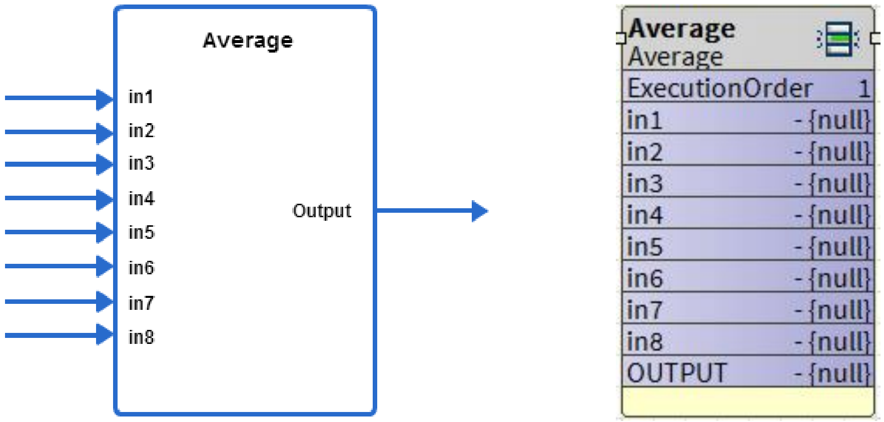

Figure 285: Behavior of Analog Latch

| Case | Input | Latch | Output | Description | Image |
|---|---|---|---|---|---|
| 1 | 45 | True | 45 | Output latches to input when Latch Transition from False to True |  |
| 2 | 0 | False | 45 | Output held in the same state until the next False to True transition |  |
| 3 | 0 | True | 0 | Output latches to input when Latch Transition from False to True |  |

| Case | Input | Latch | Output | Description | Image |
|---|---|---|---|---|---|
| 4 | 85 | True | 0 | Output held in the same state until the next False to True transition |  |
| 5 | 85 | False | 0 | Output held in the same state until the next False to True transition |  |
| 6 | 85 | True | 85 | Output latches to input when Latch Transition from False to True |  |

## Average

This function block calculates the average of 8 inputs. The output is set to the average value of inputs.




Logic Diagram

Function Block

Figure 286: Average Function

| Note: |
|---|
| The output returns an invalid value, if no inputs are connected or if all inputs are invalid. |

Ignore invalid inputs: If this option is selected, function block considers only valid inputs while determining the average of the inputs. If this option is not selected, and any input becomes invalid then output also becomes invalid.

## Inputs

Table 34: Inputs of Average Function

| Input Name | Range | | Input Value | Ignore Invalid Input | Description |
|---|---|---|---|---|---|
| | Low | High | | | |
| in1-8 | >= – infinity | <+ infinity | unconnected | | Not used in the calculation. If all inputs are unconnected, output is invalid. |
| in1-8 | >= – infinity | <+ infinity | invalid | false | If any input is invalid, output is invalid. |
| in1-8 | >= – infinity | <+ infinity | invalid | true | Output considers only valid inputs while determining the average of the inputs. |
| in1-8 | >= – infinity | <+ infinity | valid | | Calculates the average of 8 inputs or those set as constant. |

## Outputs

Table 35: Outputs of Average Function

| Output Name | Range | Description |
|---|---|---|
| OUTPUT | Any floating-point number | Average of all the inputs |

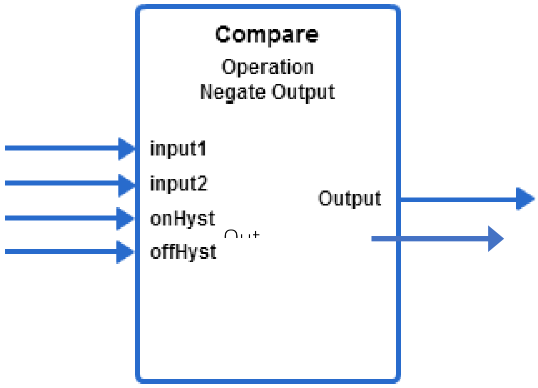
## Configuration

Table 36: Configuration of Average Function

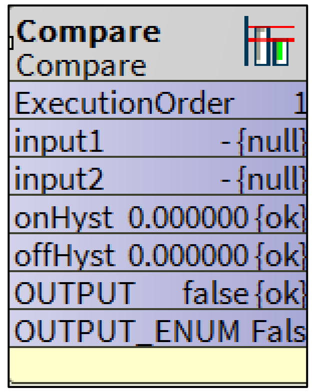
| Name | Range | Description |
|---|---|---|
| Invalid Flag | 0, 1 | See above table. |

## Compare

This function compares two inputs with each other.




Logic Diagram




Function Block

Figure 287: Compare Function

| Note: |
| --- |
| It is possible to create invalid numbers by combining large values of input 2, and on and off hysteresis. The behavior is dependent on the operation selected, value of input 1, and the compiler. (That is, the simulator may have a behavior different from the product). |

You can make the following comparison calculations using the Compare function block:

- Input1 less than input2
- Input1 greater than input2
- Input1 equal to input2

Additionally, ON and OFF hysteresis analog inputs are provided which you can use to make comparison calculations.

| Note: |
| --- |
| The output returns an invalid value if no inputs are connected or if all inputs are invalid. |

### Analog Inputs

Table 37: Inputs of Compare Function

| Input Name | Range | | Input Value | Description |
| --- | --- | --- | --- | --- |
| | Low | High | | |
| input1-2 | >= – infinity | < +infinity | unconnected | out = 0 |
| | | | invalid | out = 0 |

| | | | | |
|---|---|---|---|---|
| onHyst | 0 | < +infinity | unconnected | val = 0 |
| | | | invalid | val = 0 |
| offHyst | 0 | < +infinity | unconnected | val = 0 |
| | | | invalid | val = 0 |

## Setpoints

Table 38: Setpoints of Compare Function

| Operation | Description |
|---|---|
| Equals | • The output is set to TRUE if (Input 2 – On Hyst) <= Input 1 <= (Input 2 + Off Hyst). |
| Less than | • The output is set to TRUE if Input 1 < (Input 2 – On Hyst).<br>• The output does not change if (Input 2 – On Hyst) <= Input1 < (Input 2 + Off Hyst).<br>• The output is set to FALSE if Input1 >= (Input 2 + Off Hyst). |
| Greater than | • The output is set to TRUE if Input 1 > (Input 2 + On Hyst).<br>• The output does not change if (Input 2 – Off Hyst) < Input1 <= (Input 2 + On Hyst).<br>• The output is set to FALSE if Input1 <= (Input 2 - Off Hyst). |

## Outputs

Table 39: Outputs of Compare Function

| Output Name | Range | Description |
|---|---|---|
| OUTPUT | False (0) or True (1) | Comparison of inputs |
| OUT-PUT_ENUM | Null or False (0) or True (1) | Comparison of inputs |

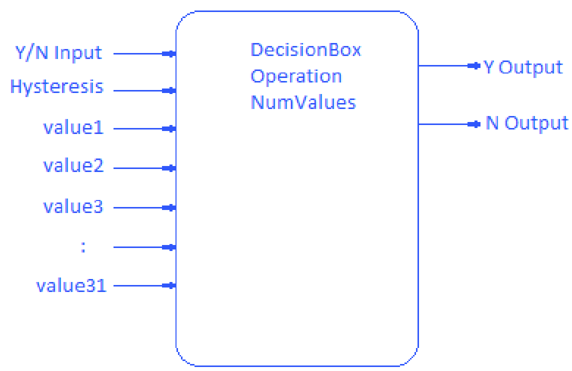
## Configuration

Table 40: Configuration of Compare Function

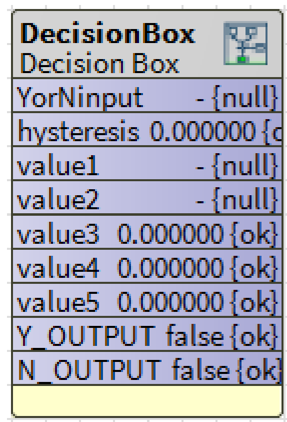
| Name | Range | Description |
|---|---|---|
| operation | 0, 2 | equals (0), less than (1), greater than (2) |

## DecisionBox

The newly added DecisionBox in the CIPer Model 30 programming model compares the analog value (value1) with one or more reference values (value2 to value31). Here the comparison operation is based on the selected operation like Equals, Greater Than, Greater Than Or Equal To, Less Than, Less Than Or Equal To, Any Equal, and In Any Range.




Logic Diagram




Function Block

Figure 288: DecisionBox Function

### Logic Inputs

Table 41: Logic Inputs of DecisionBox Function

| Input Name | Input Value | Logic Value | Description |
|---|---|---|---|
| Y/N Input | VAL != 0.0 | 1 | Enable input, possibly chained from another DecisionBox. |
| | 0 | 0 | |
| | Unconnected | 0 | |
| | Invalid | 0 | |

### Analog Inputs

Table 42: Analog Inputs of DecisionBox Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| HysteresisPar | 0 | 65535 | valid | Hysteresis to be applied for operations 2,3,4,5. Ignored for other operations. |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| | | | uncon-nected | HysteresisPar = 0 |
| | | | invalid | HysteresisPar = 0 |
| | | | < 0 | HysteresisPar = 0 |
| | | | > 65535 | HysteresisPar = 65535 |
| Value #1 | >= –∞ | <+∞ | valid | Value to be compared to reference value(s) |
| | | | uncon-nected | Value #1= 0 |
| | | | invalid | Value #1= 0 |
| Value #2 | >= –∞ | <+∞ | valid | Reference value |
| | | | uncon-nected | Value #2= 0 |
| | | | invalid | Value #2= 0 |
| Value #3 | >= –∞ | <+∞ | valid | Reference value |
| | | | uncon-nected | Value #3= 0 |
| | | | invalid | Value #3= 0 |
| Value #4 | >= –∞ | <+∞ | valid | Reference value |
| | | | uncon-nected | Value #4= 0 |
| | | | invalid | Value #4= 0 |
| Value #5 | >= –∞ | <+∞ | valid | Reference value |
| | | | uncon-nected | Value #5= 0 |
| | | | invalid | Value #5= 0 |

Additional Value inputs obtained via extension blocks are treated same as Value #5.

## Outputs

Table 43: Outputs of DecisionBox Function

| Output Name | Range | Description |
|---|---|---|
| Y-Output | False (0) or True (1) | Defaults to 0 on startup |
| N-Output | False (0) or True (1) | Defaults to 0 on startup |

## Configuration

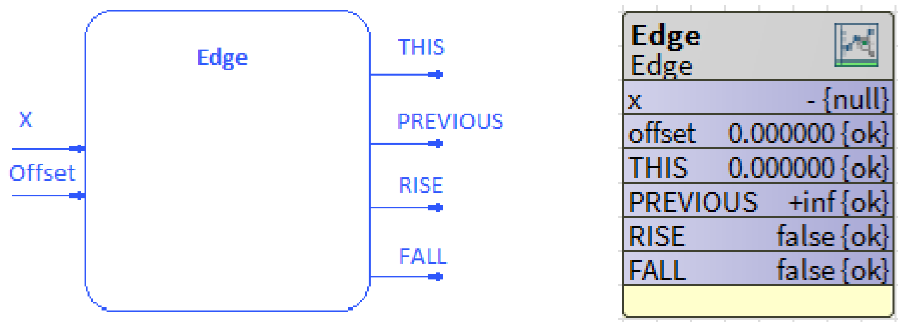
Table 44: Configuration of Decision Box Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| Operation | 1 | 7 | < 1 | Operation = invalid. Block is disabled. |
| | | | 1 | Operation = “=” |
| | | | 2 | Operation = ”>” |
| | | | 3 | Operation = ”>=” |
| | | | 4 | Operation = ”<” |
| | | | 5 | Operation = ”<=” |
| | | | 6 | Operation = Value #1 equal to any reference value |
| | | | 7 | Operation = Value #1 in any range defined by pairs of reference values |
| | | | > 7 | Operation = invalid. Block is disabled. |
| NumValues | 2 | 30 | valid | Total number of value inputs, including Value #1, Value #2, and so on and any input provided via extension blocks. Used for operations 6 and 7. |
| | | | < 2 | NumValues = invalid. Block is disabled. |

| Operation | Numvalue | YorNInput | Value 1 | Value 2 | Value 3 | Value 4 | Value 5 | Y_Output | N_Output | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| Equal | 2 or more (Value 1 & Value2 will be used & Others value will be ignored) | False | 80 | 80 | 85 | 90 | 95 | False | False | YorNInput need to be true to enable Decision box. Others value will be ignored |
| Equal | | True | 80 | 80 | 85 | 90 | 95 | True | False | Y_Output is true as Value1 is equal to Value2 |
| Equal | | True | 85 | 80 | 85 | 90 | 95 | True | True | N_Output is true as Value1 is not equal to Value2 |
| GreaterThan | | True | 85 | 80 | 85 | 90 | 95 | True | False | Y_Output is true as Value1 is greater than Value2 |
| AnyEqual | 3or More | True | 85 | 80 | 85 | 90 | 95 | True | False | Y_Output is true as Value1 is equal to Value3 i.e any of the values |
| InAnyRange | 3 | True | 84 | 80 | 85 | 90 | 95 | True | False | Y_Output is true as Value1 is in range of Value2 & Value3 |

## Edge

The Edge block monitors the X input for the significant change in value. Analog outputs of the block post the current (THIS) and PREVIOUS values of X. Logical outputs indicate when a significant change has been detected, in either the rising or falling direction. A change is considered significant only if it exceeds the specified offset from the previous value.




Logic Diagram

Function Block

Figure 289: Edge Function

| Note: |
|---|
| • Because each iteration of the block compares only the current and previous values of X, it is possible for slow changes in X to occur over time without causing the Rise/Fall flags to fire.<br>• A transition of X either to or from INVALID is not considered as a Rise or Fall event. |

For example, if the offset value is 5, it is possible for X to increase by 2 in each iteration of the block without ever causing a Rise event to fire.

### Analog Inputs

Table 45: Analog Inputs of Edge Function

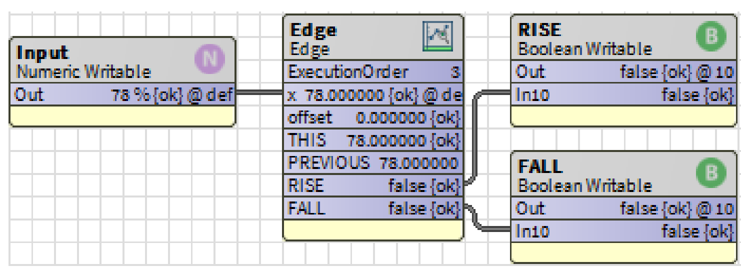
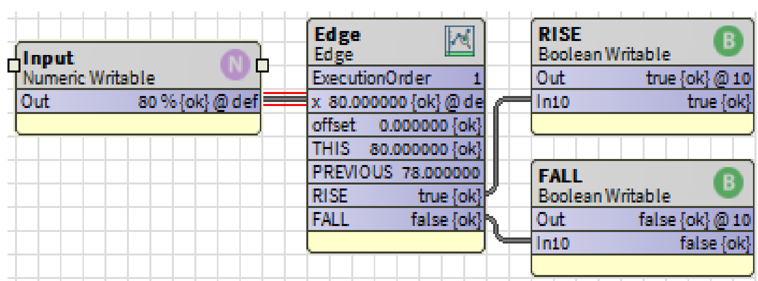
| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| X | >= –∞ | <+∞ | valid | Value being monitored for change |
| | | | unconnected | X = invalid |
| | | | invalid | X = invalid |
| Offset | 0 | <+∞ | valid | Amount of change in X required to be considered significant |
| | | | unconnected | Offset = 0 |

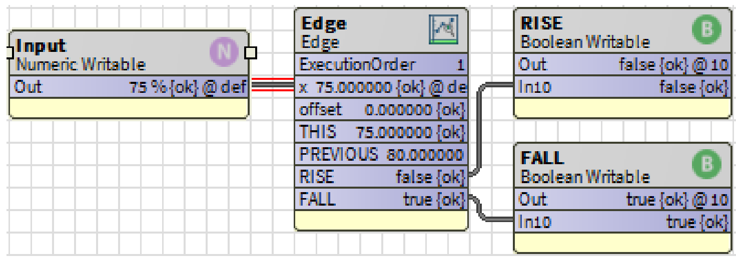
| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| | | | invalid | Offset = 0 |
| | | | < 0 | Offset = 0 |

Outputs

Table 46: Outputs of Edge Function

| Input Name | Range | Input Value |
|---|---|---|
| THIS | Any floating-point value | Value from X in this cycle |
| PREVIOUS | Any floating-point value | Value from X in last cycle |
| RISE | False (0) or True (1) | A significant increase in X is detected |
| FALL | False (0) or True (1) | A significant decrease in X is detected. |

| Input Value | This | Previ-ous | Ris e | Fall | Result |
|---|---|---|---|---|---|
| 78 | 78 | 78 | Fals e | False | As the Input value is same as the Previous and This value, the Rise and Fall both are False<br> |
| 80 | 80 | 78 | Tru e | False | Rise is True for few moment as input value is greater than the previous value<br> |
| 75 | 75 | 80 | Fals | True | Fall is True for few moment as input value is Lower |

| Input Value | This | Previ-ous | Rise | Fall | Result |
|---|---|---|---|---|---|
| | | | e | | than the previous value   |

## Encode

This function translates enumerations of a digital value into different enumeration numbers, allowing standard and custom enumerations to be combined and used together.

If the value of inEnum input does not match with any of the in1 to in9 values, OUTPUT value is equal to the value of inEnum input and FIRE output is equal to FALSE. If disable input is TRUE, the OUTPUT value is equal to inEnum input.

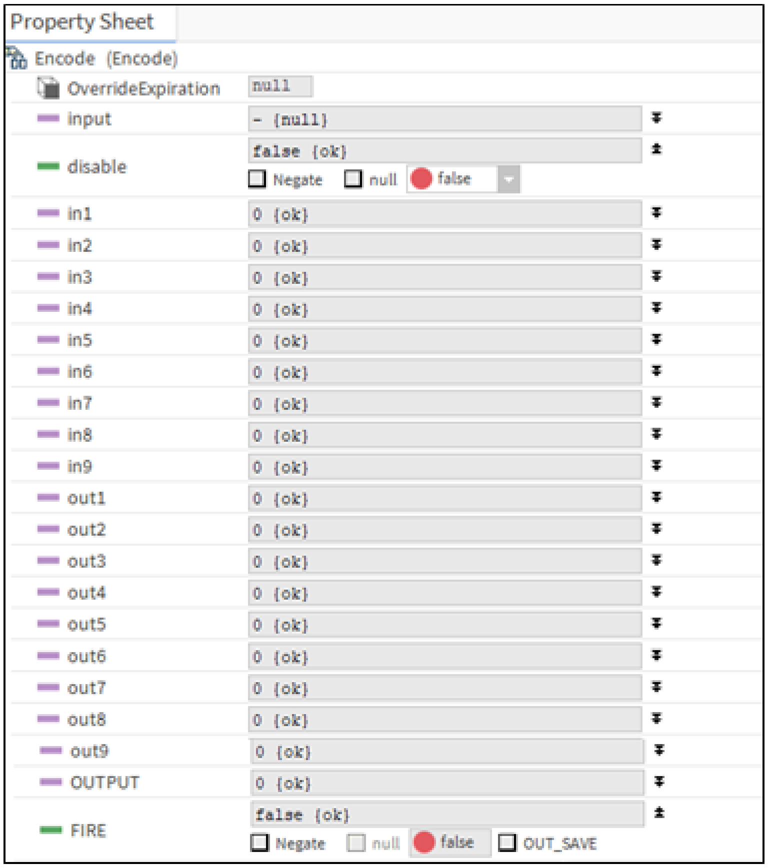
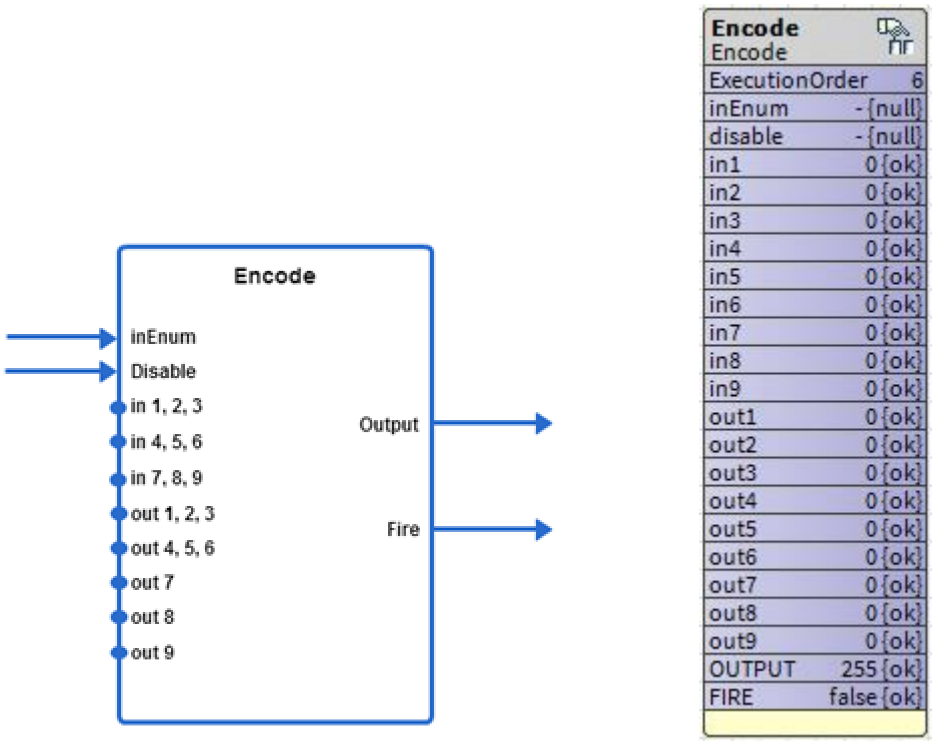



Figure 290: Ecode Property Sheet

Unconfigured is taken as high priority compared to Negate. If nothing is configured and Negate is applied nothing will get effected. Unconfigured will be taken as high priority.




Logic Diagram Function Block

Figure 291: Encode Function

| Note: |
|---|
| • To check IO pins configuration, go to terminal assignment view, click refresh button. It reflects the changes on terminal.<br>• Terminal assignment view shows only physical points. |

### Analog Inputs

Table 47: Analog Inputs of Encode Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| inEnum | 0 | 255 | unconnected | Val = 0 |
| | | | invalid | Val = 0 |
| | | | Val matches an input value | Output = matching input's output value |
| | | | Val matches two or more input values | Output = matching input's first output value |
| disable | 0 | 255 | unconnected | Val = 0 |
| | | | invalid | Val = 0 |
| | | | VAL !=0 | All mappings disable, pass input to output |
| | | | Val=0 | Enable mappings |
| in1, 2, 3 | 0 | All mappings disable, pass input to output Val=0 Enable mappings | 0xAABBCC | See the note above this table. Input 1 value 0xAA maps to output 1 value.<br>Input 2 value 0xBB maps to output 2.<br>Input 3 value 0xCC maps to output 3 |
| in4, 5, 6 | 0 | 16777215.0 | 0xDDEEFF | See the note above this table. Input 4 value 0xDD maps to output 4 value.<br>Input 5 value 0xEE maps to output 5.<br>Input 6 value 0xFF maps to output 6. |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| in7, 8, 9 | 0 | 16777215.0 | 0xGGHHII | See the note above this table. Input 7 value 0xGG maps to output 7 value.<br>Input 8 value 0xHH maps to output 8.<br>Input 9 value 0xII maps to output 9. |
| out1, 2, 3 | 0 | 16777215.0 | 0xaabbcc | See the note above this table. Input 1 value 0xaa maps to output 1 value.<br>Input 2 value 0xbb maps to output 2.<br>Input 3 value 0xcc maps to output 3. |
| out4, 5, 6 | 0 | 16777215.0 | 0xddeeff | See the note above this table. Input 4 value 0xdd maps to output 4 value.<br>Input 5 value 0xee maps to output 5.<br>Input 6 value 0xff maps to output 6. |
| out7 | 0 | 255 | 0xgg | Input 7 value 0xgg maps to output 7 value. |
| out8 | 0 | 255 | 0xhh | Input 8 value 0xhh maps to output 8 value. |
| out9 | 0 | 255 | 0xii | Input 9 value 0xii maps to output 9 value. |

| | Note: |
|---|---|

- in1, 2, 3, in4, 5, 6, in7, 8, 9, out1, 2, 3, and out4, 5, 6 are created by taking each individual input value (0-255), converting into a hex byte (0x00 – 0xFF), and putting first value in Most Significant Byte, second value in middle and third value in Least Significant Byte.
- The result gives an integer value that must be stored as a float. So, if in1 is 1, in2 is 2 and in3 is 3 then the integer would be 0x010203=66051, and the float value stored as a parameter would be 66051.0.
- The tool prompts you for individual in1 out9 values and do the conversion both to and from the packed structure.

Analog Outputs

Table 48: Analog Outputs of Encode Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| Out | 0 | 255 | See description | If an input matches a block mapping and Disable is FALSE, output block mapping.<br>If input does not match a block mapping or if Disable is TRUE, the output = input. |
| fire | 0 | 1 | See description | If Disable is FALSE and input matches a block mapping, fire output is TRUE.<br>If Disable is TRUE, fire is TRUE. |

For example, to map a standard HVAC enumeration to a custom enumeration, the standard HVAC enumeration and desired mapping is as shown in the following table.

Table 49: Standard HVAC Enumeration and Desired Mapping

| In Parameter | Input Enumeration Configurations | Input Range | Out Parameter # | Output Enumerations | Output Range |
|---|---|---|---|---|---|
| in1 | HVAC_AUTO | 0 | out1 | COOL_MODE | 0 |
| in2 | HVAC_HEAT | 1 | out2 | HEAT_MODE | 2 |
| in3 | HVAC_MORNING_WARM_UP | 2 | out3 | HEAT_MODE | 2 |
| in4 | HVAC_COOL | 3 | out4 | COOL_MODE | 0 |
| in5 | HVAC_NIGHT_PURGE | 4 | out5 | NIGHT_MODE | 7 |

| In Parameter | Input Enumeration Configurations | Input Range | Out Parameter # | Output Enumerations | Output Range |
|---|---|---|---|---|---|
| in6 | HVAC_PRECOOL | 5 | out6 | COOL_MODE | 0 |
| in7 | HVAC_OFF | 6 | out7 | OFF_MODE | 255 |
| in8 | HVAC_TEST | 7 | out8 | OFF_MODE | 255 |
| in9 | HVAC_EMERGENCY_HEAT | 8 | out9 | EMERG_HEAT | 3 |
| Block 2 passed through | HVAC_FAN_ONLY | 9 | Block 2 not used | Pass through (output =9) (Does not require mapping because the output is the same as the input.) | |
| Block 2 In 1 | HVAC_NUL | 255 | Block 2 Out1 | REHEAT_MODE | 1 |

The first encode function block parameters are:

- in1, 2, 3: 0, 1, 2 = 0x000102 = 258
- in4, 5, 6: 3, 4, 5 = 0x030405 = 197637
- in7, 8, 9: 6, 7, 8 = 0x060708 = 395016
- out 1, 2, 3: 0, 2, 2 = 0x000202 = 514
- out4, 5, 6: 0, 7, 0 = 0x000700 = 1792
- out7: 255
- out8: 255
- out9: 3

The second encode function block parameters are:

- in1, 2, 3: 255, 0, 0 = 0xFF0000 = 16711680
- in4, 5, 6: 0, 0, 0 = 0x000000 = 0
- in7, 8, 9: 0, 0, 0 = 0
- out1, 2, 3: 1, 0, 0 = 0x010000 = 65535
- out4, 5, 6: 0, 0, 0 = 0
- out7: 0
- out8: 0
- out9: 0

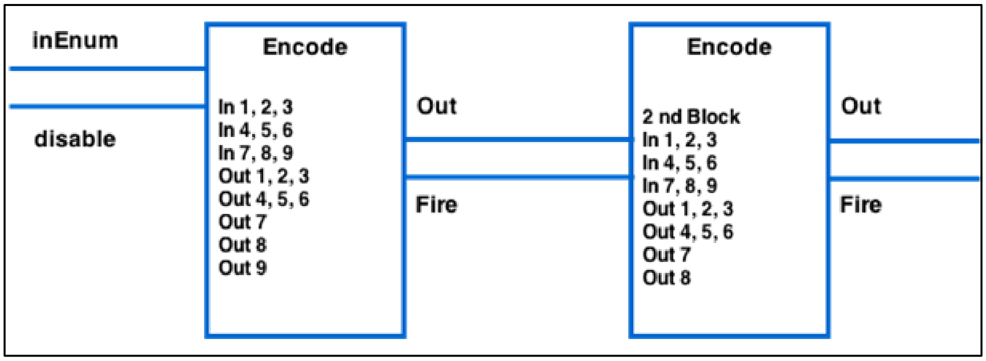



Figure 292: Encode Function Block Parameters

## HystereticRelay

This function sets the output to TRUE if input value is greater than onVal and to FALSE if input value is less than offVal.

minOn: When output transits from TRUE to FALSE, it remains TRUE for the period specified by the minOn input. This input can be a constant, or can take value from other function blocks or physical/network inputs.

minOff: When output transits from FALSE to TRUE, it remains FALSE for the period specified by the minOff input.

In each iteration, the function block keeps track of the current minimum on or off time. On power, up/reset this timer is cleared.

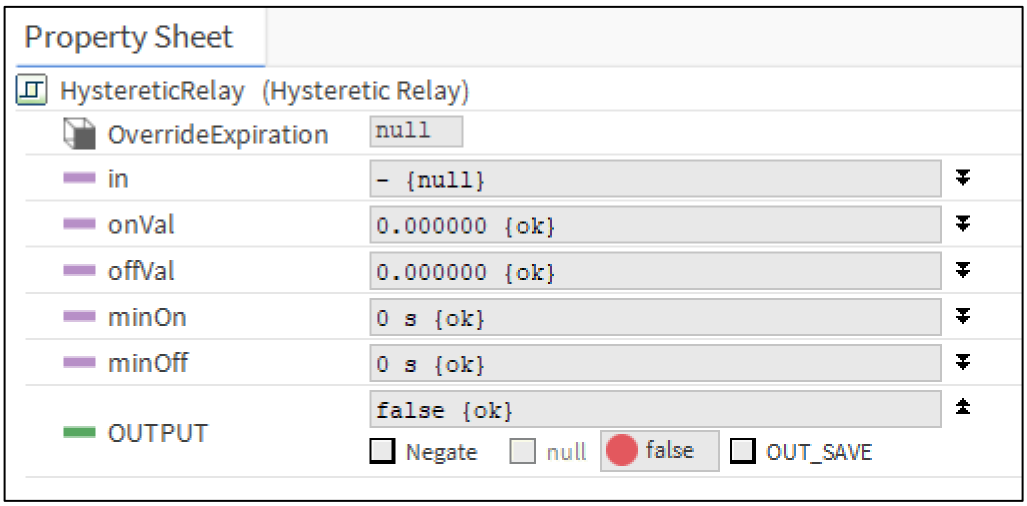
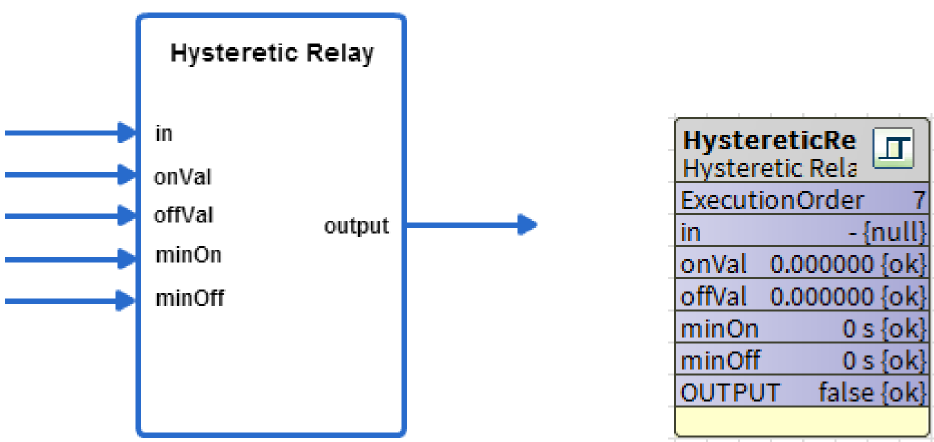



Figure 293: HystereticRelay Property Sheet




Logic Diagram Function Block

Figure 294: Hysteretic Relay Function

### Analog Inputs

Table 50: Inputs of Hysteretic Relay Function

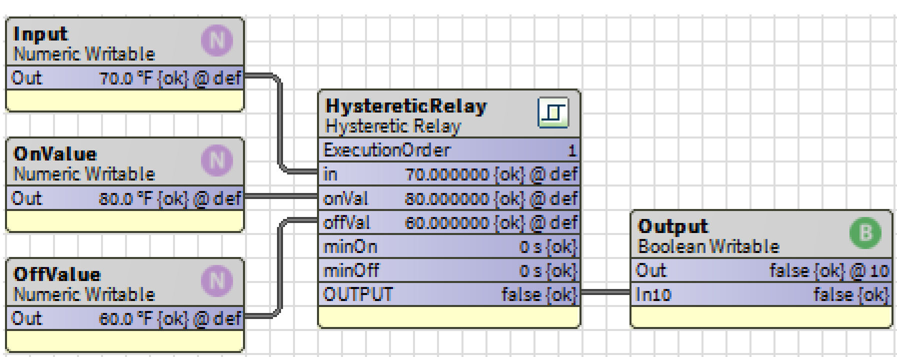
| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| In | >= – in-finity | <+ infinity | unconnected | val = invalid Output = FALSE |
| | | | invalid | val = invalid Output = FALSE |
| onVal | >= – in-finity | <+ infinity | unconnected | val = invalid Output = FALSE |
| | | | invalid | val = invalid Output = FALSE |
| offVal | >= – in-finity | <+ infinity | unconnected | val = FALSE Output = invalid |
| | | | invalid | val = invalid Output = FALSE |
| minOn (sec) | 0 | 65535 | unconnected | val = 0 |
| | | | invalid | val = 0 |
| minOff (sec) | 0 | 65535 | unconnected | val = 0 |
| | | | invalid | val = 0 |

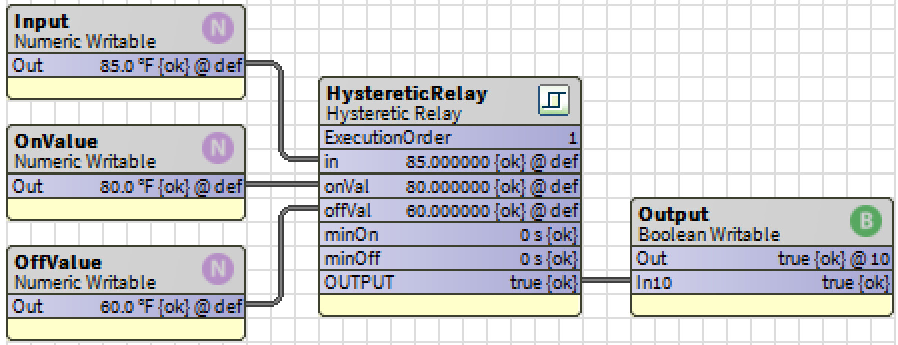
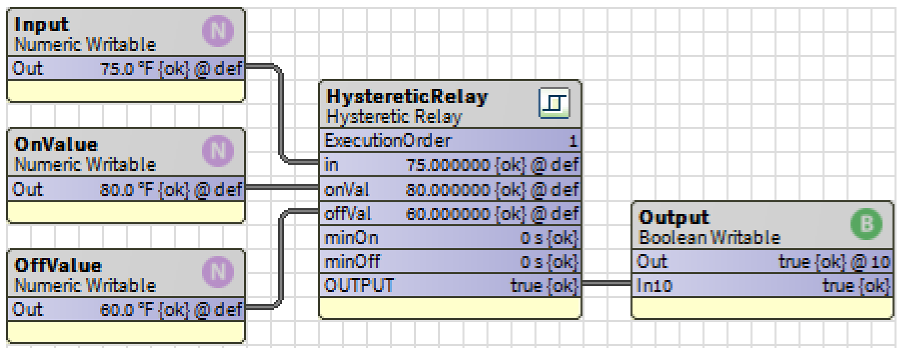
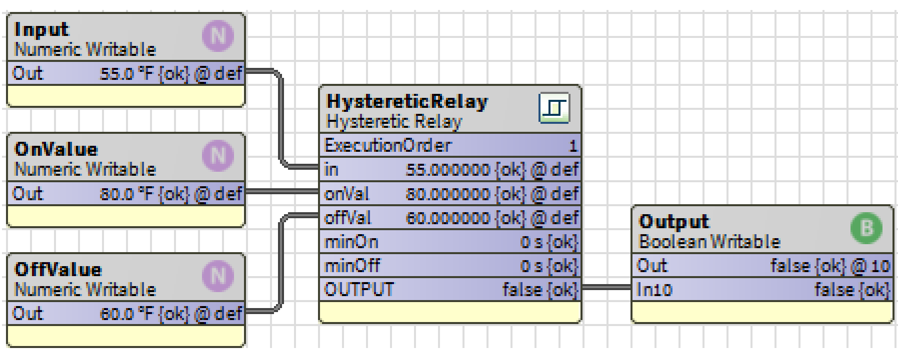
Output

Table 51: Output of Hysteretic Relay Function

| Output | Name Range | Description |
|---|---|---|
| OUT-PUT | False (0) or true (1) | The output is set to TRUE at onVal and FALSE at offVal while honoring minOn and minOff times. |

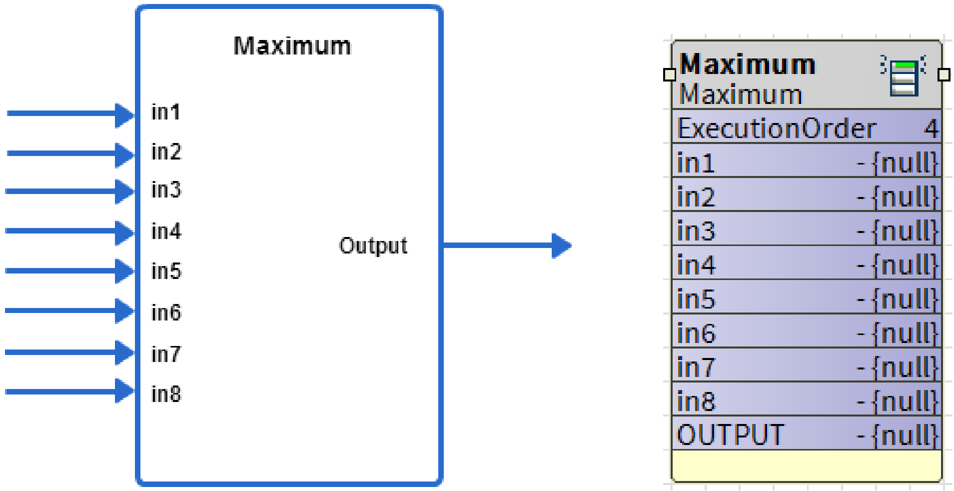
Example:

| Input value | Output Value | Remarks |
|---|---|---|
| 70 | False | The Output is False as the Input is lower than the On Value.  |

<table>
<tr><td>85</td><td>True</td><td>The Output is True as the Input is greater than the On Value.<br><br></td></tr>
<tr><td>75</td><td>True</td><td>The Output is still true as the Input is greater than the Off Value.<br><br></td></tr>
<tr><td>55</td><td>False</td><td>The Output is False as the Input is lower than the Off Value<br><br></td></tr>
</table>

## Maximum

This function calculates the maximum of 8 inputs (connected inputs or inputs set as constant). The output is set to the largest value of all the inputs.




Logic Diagram　　Function Block

Figure 295: Maximum Function

| Note: |
|---|
| If one or more inputs are selected as constant, any previous connection from the outputs of other function blocks to maximum function block is removed automatically and the maximum of the selected constant values is set as the output. |

Ignore invalid inputs: If this option is selected, function block considers only valid inputs while determining the maximum of the inputs. If this option is not selected, and any input becomes invalid then output also becomes invalid.

## Inputs

Table 52: Inputs of Maximum Function

| Input Name | Range | | Input Value | Ignore Invalid Input | Description |
|---|---|---|---|---|---|
| | Low | High | | | |
| in1-8 | >= – infinity | <+ infinity | unconnected | | Not used in the calculation. If all inputs are unconnected, output is invalid. |
| in1-8 | >= – infinity | <+ infinity | invalid | false | If any input is invalid, output is invalid. |

| Input Name | Range | | Input Value | Ignore Invalid Input | Description |
|---|---|---|---|---|---|
| | Low | High | | | |
| in1-8 | >= – infinity | <+ infinity | invalid | true | Output considers only valid inputs while determining the maximum of the inputs. |
| in1-8 | >= – infinity | <+ infinity | valid | | Calculates the maximum of 8 inputs or those set as constant. |

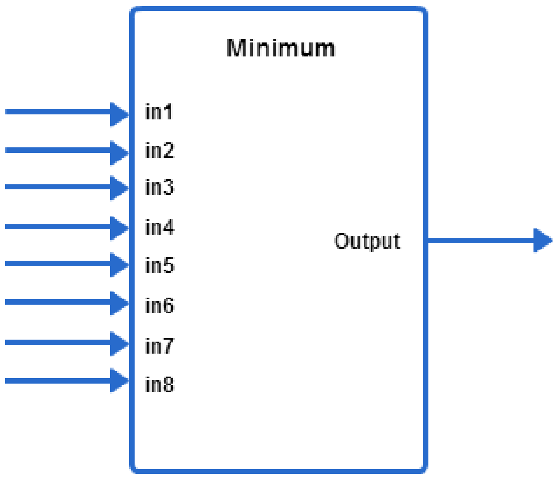
Outputs

Table 53: Outputs of Maximum Function

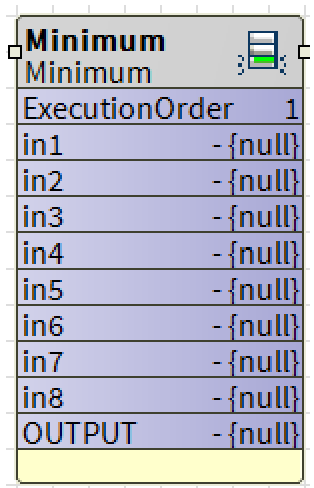
| Output Name | Range | Description |
|---|---|---|
| OUTPUT | Any floating-point value | Maximum of the inputs |

## Minimum

This function calculates the minimum of 8 inputs or those set as constant. The output is set to the smallest input. Unused/invalid inputs are ignored.




Logic Diagram




Function Block

Figure 296: Minimum Function

Ignore invalid inputs: If this option is selected, function block considers only valid inputs while determining the minimum of the inputs. If this option is not selected, and any input becomes invalid then output also becomes invalid.

The Ignore invalid inputs option is not supported in the PVL6436A, PVL6438N, and PUL6438 models and therefore this configuration cannot be downloaded to those models.

### Analog Inputs

Table 54: Inputs of Minimum Function

| Input Name | Range | | Input Value | Ignore Invalid Input | Description |
|---|---|---|---|---|---|
| | Low | High | | | |
| in 1-8 | >= – infinity | <+ infinity | unconnected | | Not used in the calculation.<br>If all inputs are unconnected, output is invalid. |
| in1-8 | >= – infinity | <+ infinity | invalid | False | If any input is invalid, output is invalid. |
| in1-8 | >= – infinity | <+ infinity | invalid | True | Output considers only valid inputs while determining the minimum of the |

| Input Name | Range | | Input Value | Ignore Invalid Input | Description |
|---|---|---|---|---|---|
| | Low | High | | | |
| | | | | | inputs. |
| in1-8 | >= – in-finity | <+ in-finity | valid | | Calculates the minimum of 8 inputs or those set as constant. |

### Outputs

Table 55: Outputs of Minimum Function

| Output | Range | Description |
|---|---|---|
| OUTPUT | Any floating-point value | Minimum of the inputs |

### Configuration

Table 56: Configuration of minimum Function

| Name | Range | Description |
|---|---|---|
| Invalid Flag | 0, 1 | See above table. |

## PrioritySelect

This function allows one to four inputs in any combination to be individually enabled to override the default. The output is the input with its highest priority enabled TRUE.

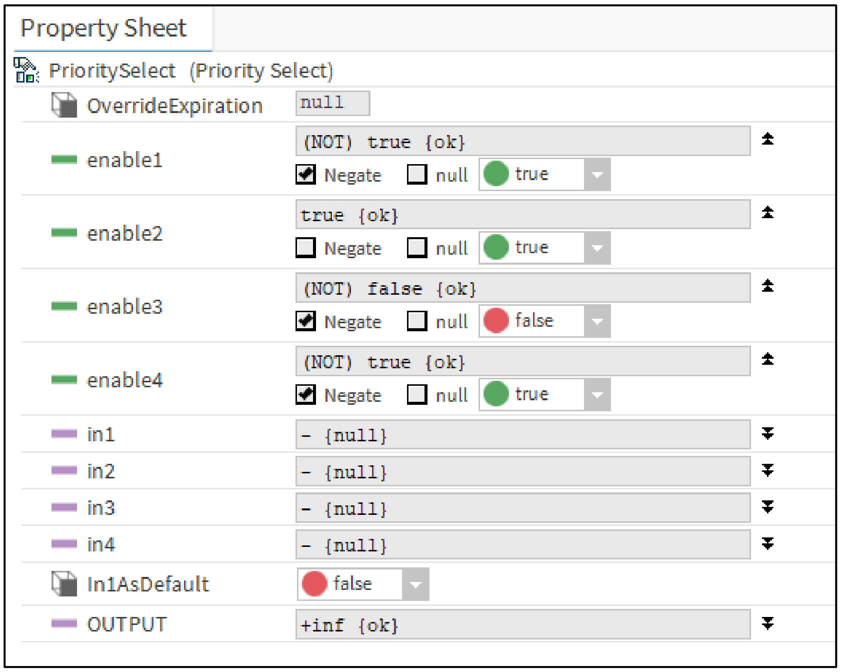



Figure 297: PrioritySelect Property Sheet

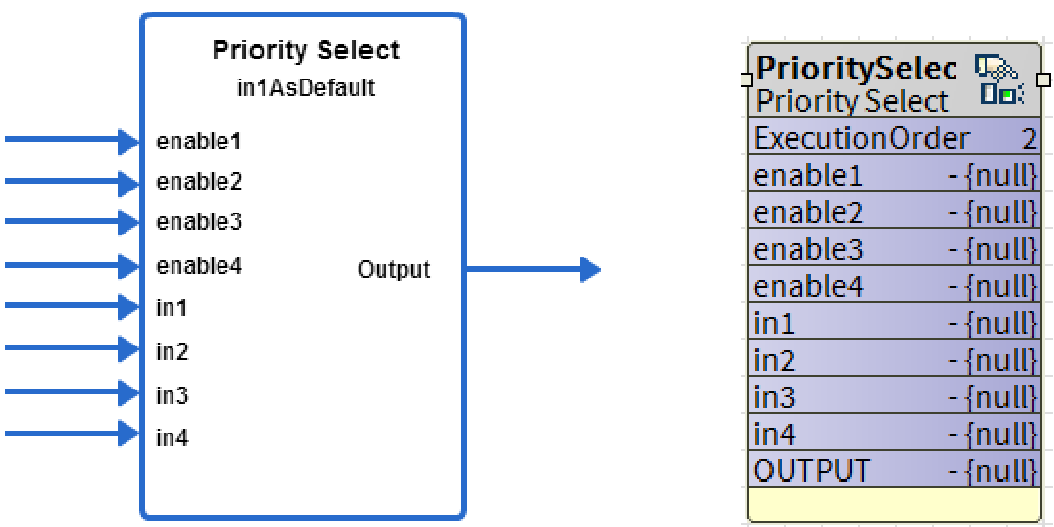



Figure 298: Priority Select Function

Unconfigured is taken as high priority compared to Negate. If nothing is configured and Negate is applied nothing will get effected. Unconfigured will be taken as high priority.

## Logic Inputs

Table 57: Logic Inputs of PrioritySelect Function

| Input Name | Input Value | Logic Value | Description |
|---|---|---|---|
| enable1 -4 | VAL != 0.0 0 | 1 | |
| | 0 | 0 | |
| | unconnected | 0 | |
| | invalid | 0 | |

## Analog Inputs

Table 58: Analog Inputs of PrioritySelect Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| in1-4 | >= – infin-ity | <+ infinity | uncon-nected | val = invalid |
| | | | invalid | val = invalid |
| | | | Valid | Val = value of in1-4 |

## Setpoint

Table 59: Setpoint of PrioritySelect Function

| Name | Range/Value | Description |
|---|---|---|
| In1 As De-fault | Yes | Output is set to Input 1 even if all Enable Inputs 1-4 are invalid. |
| | No | Output is set to Invalid if all Enable Inputs 1-4 are disabled. |

## Output

Table 60: Output of PrioritySelect Function

| Output Name | Range | Description |
|---|---|---|
| OUTPUT | Any floating-point value | The output is set to the input that is enabled.<br>• If all inputs are unconnected, output is invalid.<br>• If all enable inputs are disabled, and all inputs are invalid, the output is invalid.<br>• If In1asDefault is enabled, output is input1, even if all enable inputs are disabled. |

| Output Name | Range | Description |
|---|---|---|
| | | • When SetIn1asDefault is disabled/enabled and if at least one enable input is enabled, output is the input with its highest priority enabled TRUE. The priority order among enable inputs is:<br>1. enable1<br>2. enable2<br>3. enable3<br>4. enable4 |

Based on the In1asDefault option and the Enable options selected, the output is set as Input as shown in the table. Output Based on In1AsDefault

| Name | Range | Description |
|---|---|---|
| In1AsDefault | False (0) or True (1) | You can specify if input 1 should be used as the default (Yes/No). If none of the enable inputs are true and input 1 is configured to be the default, the output is set to input 1. If none of the enable inputs are true and input 1 is not the default, the output is set to Invalid. |

Table 61: Output Based on In1AsDefault

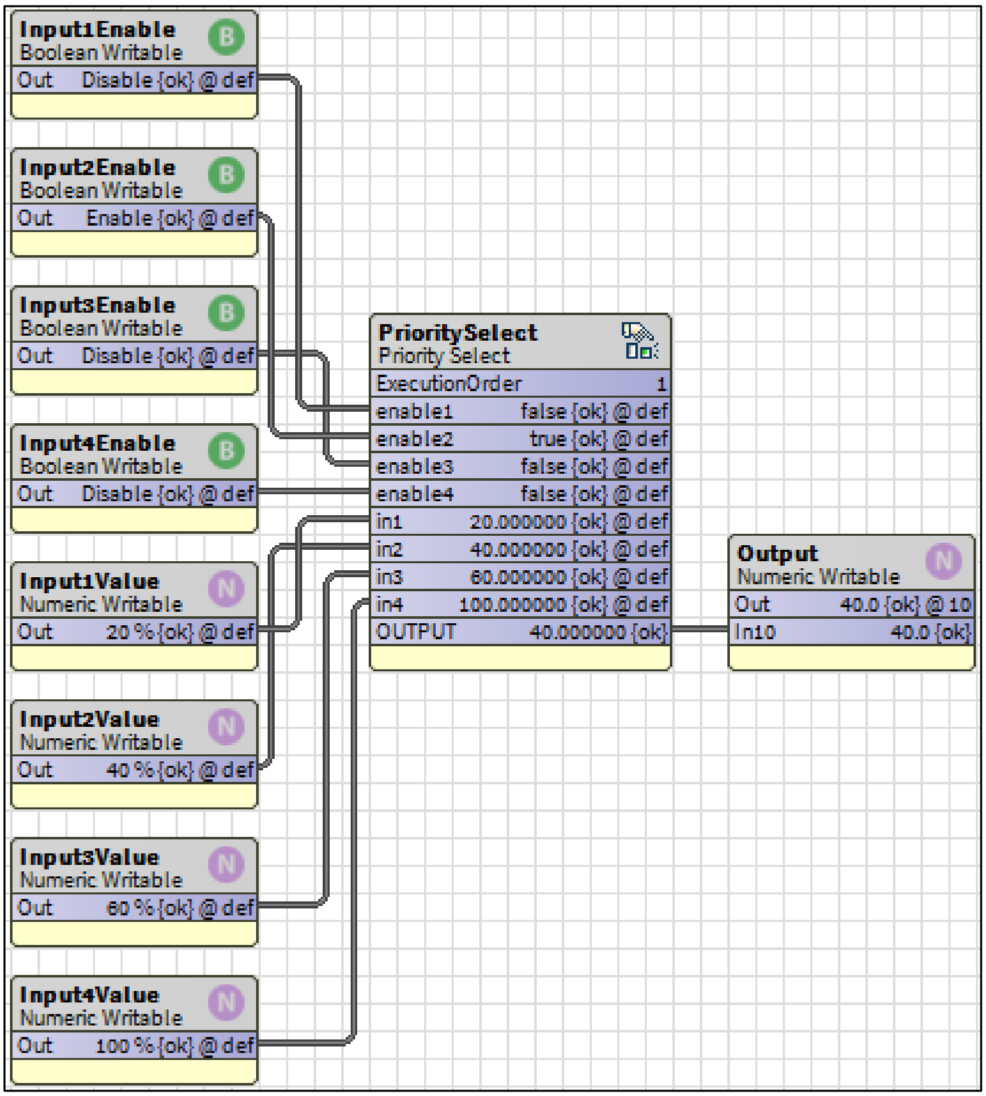
| In1asDefault | Enable Inputs 1–4 | Output |
|---|---|---|
| Enabled | Disabled | Output is set to Input1. |
| | Enabled | Output is set to highest enabled input. |
| Disabled | Disabled | The output is invalid. |
| | One or more inputs is Enabled | Output is set to one of the Inputs 1-4 based on the priority order:<br>1. Enable1<br>2. Enable2<br>3. Enable3<br>4. Enable4 |

Note:

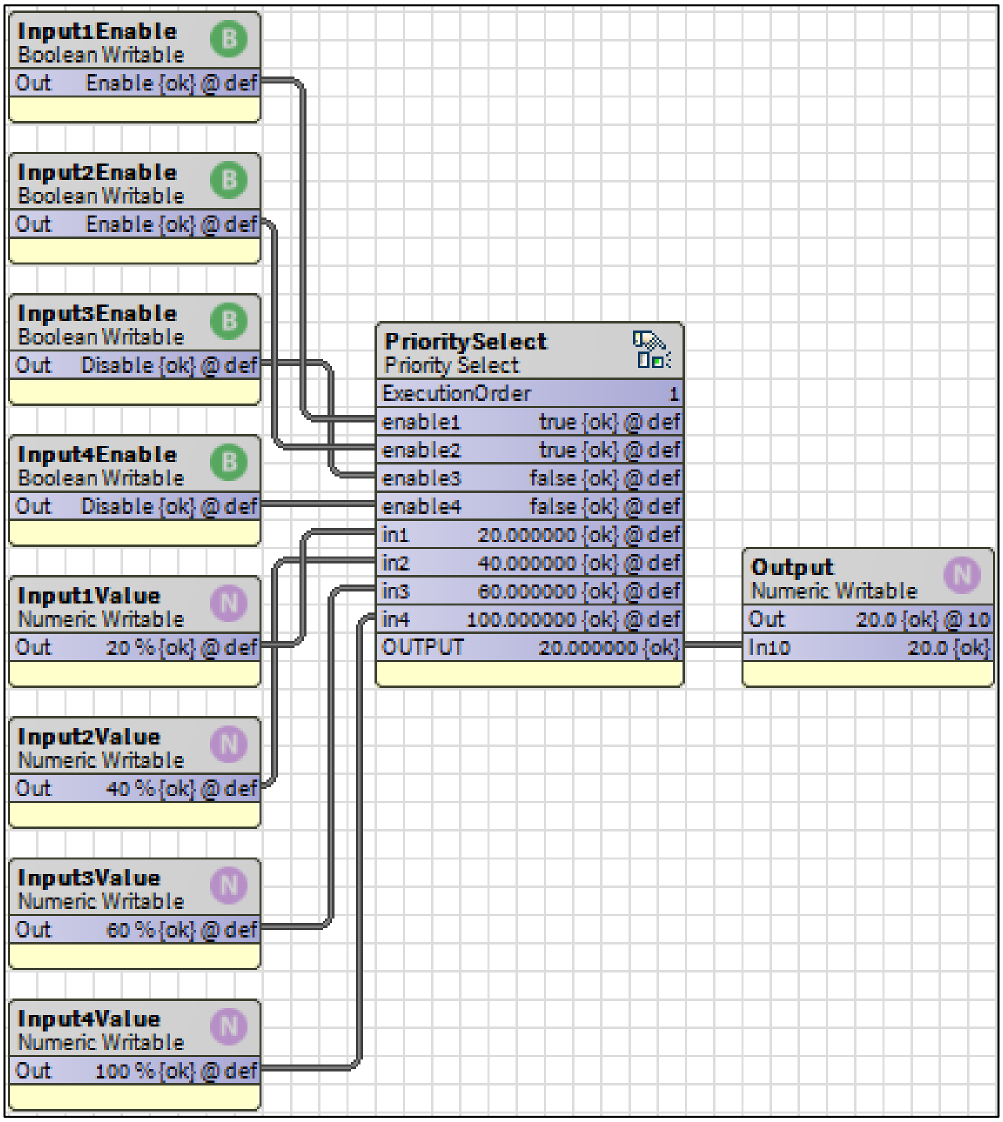
- Enable 1 has the highest priority and if it is enabled, output is taken as Input1.
- If Enable 1 is disabled, Enable 2 has the next highest priority, and if Enable 2 is enabled, output is taken as Input 2.
- Enable 3 has the third highest priority and if Enable 1 and Enable 2 are disabled, the output is taken as Input 3.
- Enable 4 has the least priority and output is set to Input 4 only if Enable 1-3 is disabled.

Example:

The Output value is same as Input 2 as Enable2 is enabled.



As Input1 has higher priority, thus out is same as Input 1 Value.



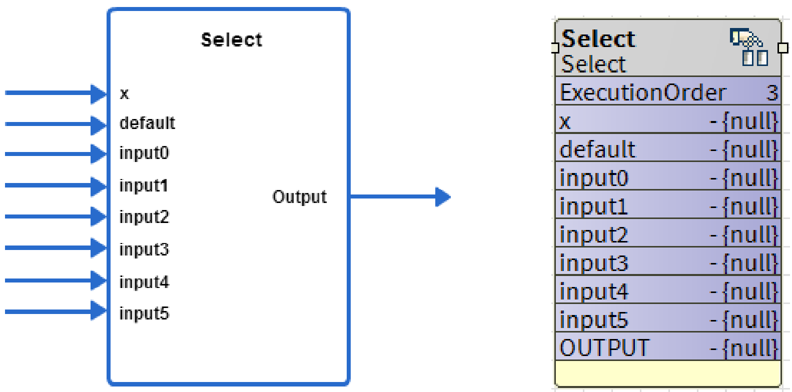
## Select

The Select function block selects one of the 6 input values to be transferred to the output. The input selected depends on the value of X.

The default input allows multiple select function blocks to be tied together by chaining the outpu00t of one block to the default input of the next.

When the Select function blocks are chained, all chained blocks receive the same input, but different offsets. So, the select function blocks examine different ranges of the input values.

When (X) selects one of the 6 inputs, the output equals the value on the input (X-offset). Otherwise, the output equals the value on the default input.




Logic Diagram    Function Block

Figure 299: Select Function

### Analog Inputs

Table 62: Analog Inputs of Select Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| x | 0 | 255 | | selection index |
| | | | uncon-nected | val = invalid |
| | | | invalid | val = invalid |
| default | >= –∞ | <+∞ | | Default value to be output when x selection in-dex is invalid or out of range |
| | | | uncon-nected | val = invalid |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| | | | invalid | val = invalid |
| input0 to 5 | >= –∞ | <+∞ | | inputs selected by the index (x – offset) |
| | | | uncon-nected | val = invalid |
| | | | invalid | val = invalid |

Output

Table 63: Output of Select Function

| Output Name | Range | Description |
|---|---|---|
| Output | Any floating-point value | Output = input (x-offset) |

Setpoint

Table 64: Setpoints of Select Function

| Name | Range | Description |
|---|---|---|
| Offset | 0 - 255 | Output value depends upon the input value x and offset. Output value is the value equal to the input (any one of the input0-5) selected by the input x-offset.<br>input0: 1<br>input1: 2<br>input2: 3<br>input3: 4<br>input4: 5<br>input5: 6<br>offset: 2<br>If input x is 4, (x - offset = 4 - 2) = 2<br>Output is the value of input 2. From the input values, out-put=3. |

| Note: |
| --- |
| If any input is invalid, the output is invalid. |

Output = Position determined by the value (X - Offset).

- If the value of (X – Offset) is greater than 6, the default value is taken as the Output.
- If the value (X - Offset) is a floating-point number between 0 and 6, the position is determined as:
    - 0.10 – 0.99, 0 is returned and Input 0 is taken as Output
    - 1.10 – 1.99, 1 is returned and input 1 is taken as Output
    - 2.10 – 2.99, 1 is returned and input 2 is taken as Output
    - 3.10 – 3.99, 1 is returned and input 3 is taken as Output
    - 4.10 – 4.99, 1 is returned and input 4 is taken as Output
    - 5.10 – 5.99, 1 is returned and input 5 is taken as Output

Example 1:

X = 100, Offset = 97, default = 10

Output = 100 – 97 = 3, and hence input 3 is taken as the output.

Example 2:

X = 100.6, Offset = 95.2, default = 10

Output = 100.6 – 95.2 = 5.4, and hence input 5 is taken as the output.

Example 3:

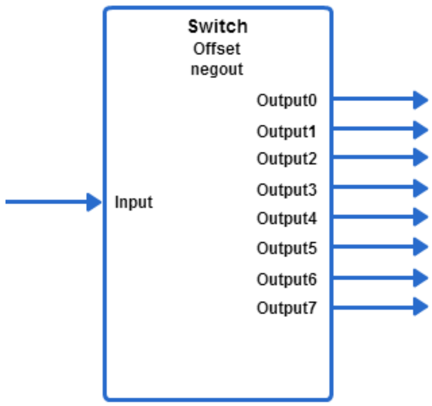
X = 100, Offset = 5.2, default = 10

Output = 100 – 5.2 = 94.4, and hence default value 10, is taken as the output.

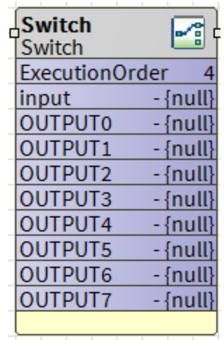
## Switch

This function takes an enumerated typed input and subtracts a user-defined offset to determine which output to set TRUE, holding all others FALSE.

The valid range of the input minus the offset is 0 through 7. The output X (0 through 7) is TRUE if input – offset = X, else, it is FALSE.




Logic Diagram




Function Block

Figure 300: Switch Function

### Analog Inputs

Table 65: Analog Inputs of Switch Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| input | 0 | 255 | unconnected | val = invalid, all outputs off. Note 1 |
| | | | invalid | val = invalid, all outputs off. Note 1 |
| | | | in - offset > 8 | All outputs off. See the note in the Encode function block section. |
| | | | in - offset < 0 | All outputs off. See the note in the Encode function block section. |

## Output

Table 66: Outputs of Switch Function

| Output Name | Range | Description |
|---|---|---|
| OUTPUT 0 - 7 | False (0) or True (1) | The output [0-7] is TRUE if input – offset = X, otherwise it is FALSE. |

## Setpoint

Table 67: Setpoint of Switch Function

| Output Name | Range | Description |
|---|---|---|
| offset | 0 - 255 | Used to determine which Output is set to TRUE based on the expression (input - offset) Output |

## Configuration

Table 68: Configuration of Switch Function

| Output Name | Range | Description |
|---|---|---|
| offset | 0 - 255 | Base number from which input matches are measured. |

Output = Output position determined by the value (input – Offset).

- If the value of (input – Offset) is greater than 7, all outputs are taken as FALSE.
- If the value (input – Offset) is a floating-point number between 0 and 8, the position is determined as:
  - 0.10 – 0.99, 0 is returned, Output 0 is TRUE and all other outputs are FALSE
  - 1.10 – 1.99, 1 is returned, Output 1 is TRUE and all other outputs are FALSE
  - 2.10 – 2.99, 2 is returned, Output 2 is TRUE and all other outputs are FALSE
  - 3.10 – 3.99, 3 is returned, Output 3 is TRUE and all other outputs are FALSE
  - 4.10 – 4.99, 4 is returned, Output 4 is TRUE and all other outputs are FALSE
  - 5.10 – 5.99, 5 is returned, Output 5 is TRUE and all other outputs are FALSE
  - 6.10 – 6.99, 6 is returned, Output 6 is TRUE and all other outputs are FALSE
  - 7.10 – 7.99, 7 is returned, Output 7 is TRUE and all other outputs are FALSE

Example 1:

Input = 100, Offset = 97

Output = 100 – 97 = 3, and hence Output 3 is made TRUE and all other outputs are made FALSE.

Example 2:

X = 100.6, Offset = 95.2

Output = 100.6 – 95.2 = 5.4, and hence Output 5 made TRUE and all other outputs are made FALSE.

Example 3:

X = 100, Offset = 5.2

Output = 100 – 5.2 = 94.4, and hence all Outputs are made FALSE.

# Control Function Blocks

The CIPer Model 30 programming model provides the following control function blocks that you can configure and use to build the required application logic:

- AIA
- Cycler
- FlowControl
- PID
- RateLimit
- Stager
- StageDriver

## AIA

This function is an Adaptive Integral Action controller (AIA). You can use AIA in place of the PID (Proportional Integral Derivative), because it works better than PID, when delays in the process being controlled cause integral windup resulting in undershoot or overshoot that leads to instability.

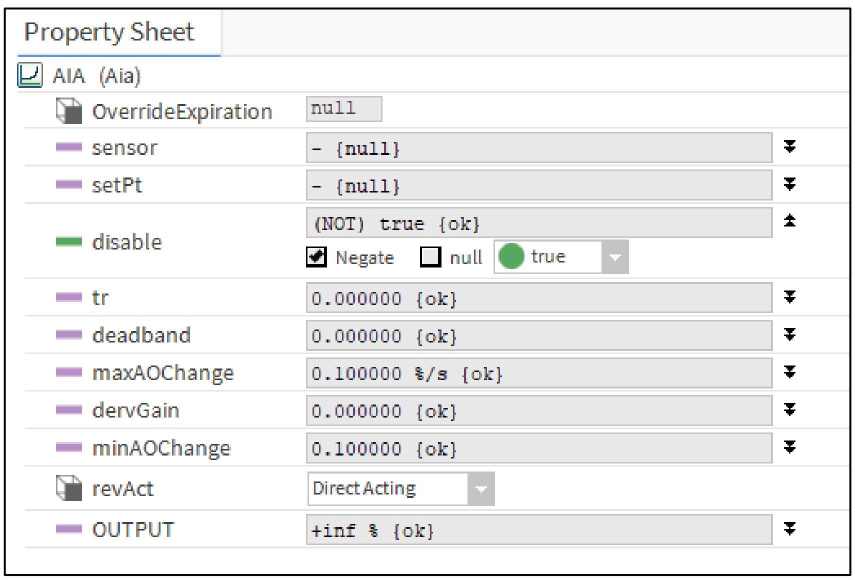
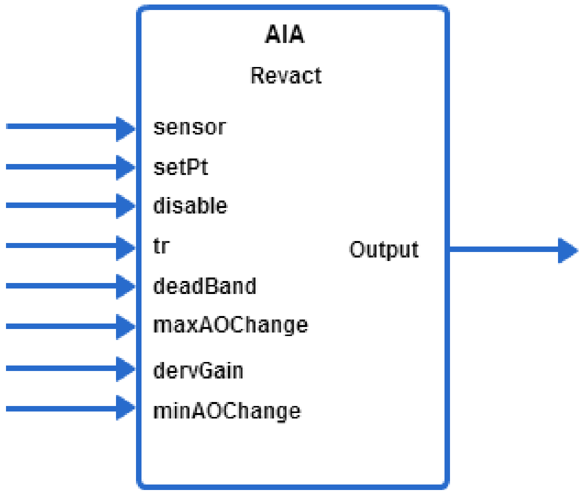
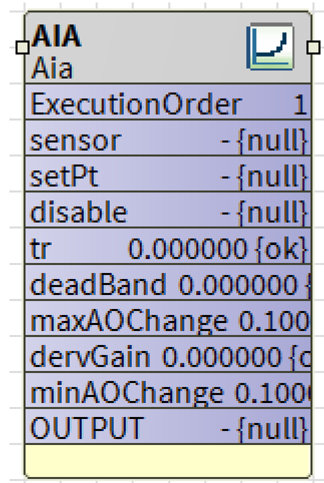



Figure 301: AIA Property Sheet

Unconfigured is taken as high priority compared to Negate. If nothing is configured and Negate is applied nothing will get effected. Unconfigured will be taken as high priority.




Logic Diagram




Function Block

Figure 302: AIA Function

Err = Sensor – Set Point.

If Direct/Reverse is set to reverse, Err term is set to – Err.

tr (throttling range): It is error value that results in an output change of the maximum value (maxAOChange) from one step to the next. maxAOChange is the maximum amount in percentage that output changes for a single cycle of the control (1 second). This is typically set to 100 percent / (Actuator speed (second/full stroke)).

Deadband: Error value must be greater than the deadband absolute value to reflect any change in the output.

EffErr = Err – deadband

If Err > 0, ErrSign = 1 else ErrSign = –1.

If |Err| < deadband, AbsErr = 0.

Otherwise (|Err| > deadband), AbsErr = |Err| – deadband

Output = Output + ErrSign * [(maxAOChange – minAO) * (AbsErr / (ThrottlingRange – Deadband)) * 3 + MinAO)].

In each iteration, the function block keeps track of the old proportional error. On power up/reset this is cleared.

Whenever there is a certain configuration change in AIA block (like derivative gain) if users expect behavior like Spyder controller, then they need to Stop and start the Sequenced Control Engine.

Steps to stop and start the sequenced control engine

1. Right click on Sequenced Control Program
2. Invoke Action request sequenced Control engine stop
3. Then invoke Start Sequenced Control Engine to start the engine

### Logic Inputs

Table 69: Logic Inputs of AIA Function

| Input Name | Input Value | Logic Value | Description |
|---|---|---|---|
| disable | unconnected | 0 | AIA function runs. |
| | VAL != 0.0 | 1 | Disable AIA function. Output is set to 0. |
| | 0 | 0 | AIA function runs. |
| | invalid | 0 | AIA function runs. |

### Analog Input

Table 70: Analog Inputs of AIA Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| sensor | >= – infinity | < +infinity | unconnected | AIA function is disabled. Output is set to 0. |
| | | | invalid | AIA function is disabled. Output is set to 0. |
| setPt | >= – infinity | < +infinity | unconnected | AIA function is disabled. Output is set to 0. |
| | | | invalid | AIA function is disabled. Output is set to 0. |
| tr | 0 | < +infinity | unconnected | AIA function is disabled. Output is set to 0. |
| | | | invalid | AIA function is disabled. Output is set to 0. |
| | | | VAL <= 0 | AIA function is disabled. Output is set to 0. |
| maxAOChange (%/sec) | 0 < | 100 | unconnected | maxAOChange = 1.1%/sec |
| | | | invalid | maxAOChange = 1.1%/sec |
| | | | 0 | maxAOChange = 1.1%/sec |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| | | | VAL < low | maxAOChange = 1.1% /sec |
| | | | VAL > high | maxAOChange = 1.1% /sec |
| deadband | 0 | < tr | unconnected | Disable Dead Band action. |
| | | | invalid | Disable Dead Band action. |
| | | | VAL < low<br>OR<br>VAL >+ tr | DB = 0 |
| | | | 0 | disable Dead Band action. |
| derivGain | 0 | <+ | unconnected | val = 0 |
| | | | invalid | val = 0 |
| minAO-Change | 0 < | <= maxAO-Change | unconnected | minAOChange = 0 |
| | | | invalid | minAOChange = 0 |
| | | | VAL < 0 | minAOChange = 0 |
| | | | VAL>=maxAO-Change | minAOChange = 0 |

Output

Table 71: Output of AIA Function

| Out-put Name | Range | Description |
|---|---|---|
| OUT-PUT | 0 to +100% | Output = Output + ErrSign * NonLin (AbsErr, ThrottlingRange, maxAOChange, minAOChange) |

Setpoint

Table 72: Setpoint of AIA Function

| Name | Range | Description |
|---|---|---|
| revAct | 0 = Direct acting<br>1 = Reverse acting | User-specified revAct value |

Configuration

Table 73: Configuration of AIA Function

| Name | Range | Description |
|---|---|---|
| revAct | 0 to 1 | You can specify revAct: 0 = Direct acting, 1 = reverse acting. |

## Cycler

This function is a generic stage driver or a Thermostat Stage Cycler dependent on the value of the CPH (Cycles per Hour) parameter (cph = 0 means stager functionality, and cph = 1 - 60 gives thermostat cycler functionality). The input range (In) for the Cycler block is -200 to +200.

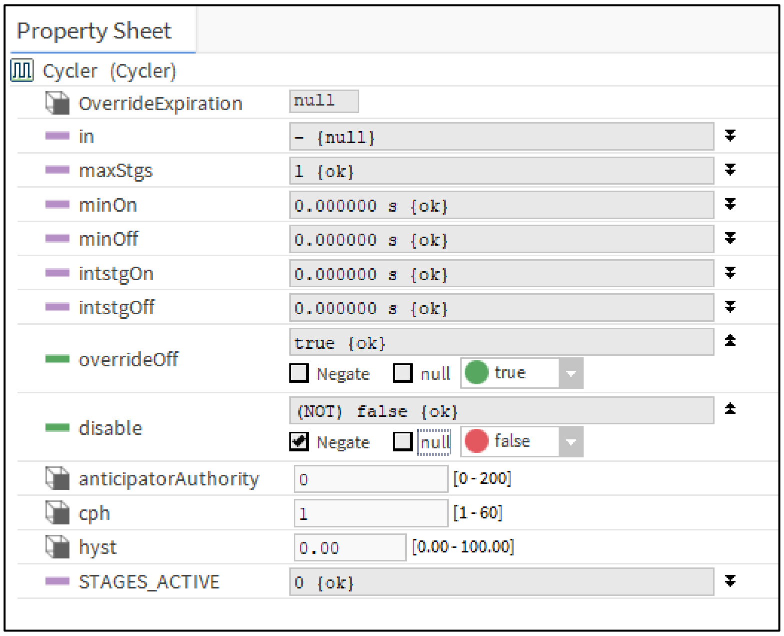
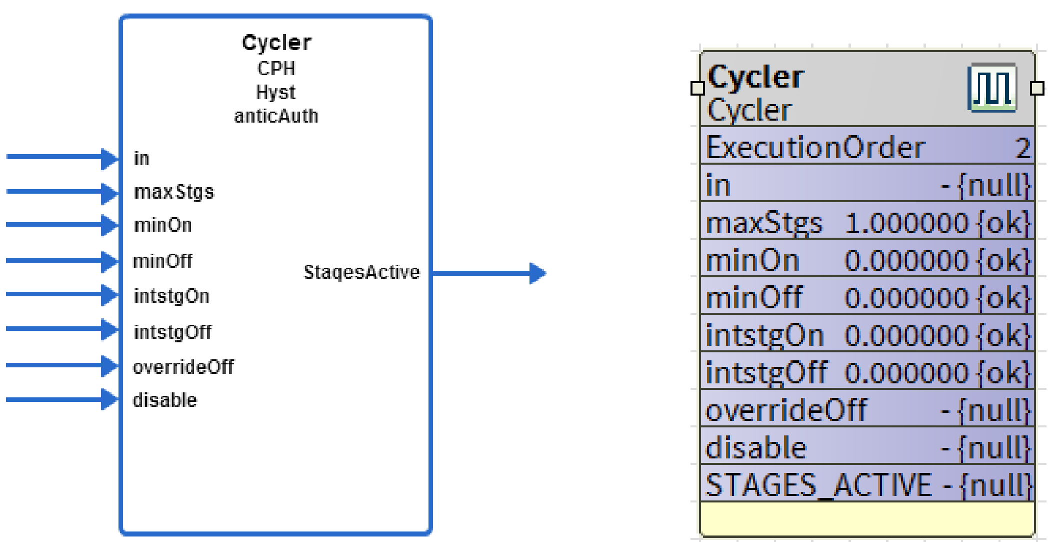



Figure 303: Cycler Property Sheet

Unconfigured is taken as high priority compared to Negate. If nothing is configured and Negate is applied nothing will get effected. Unconfigured will be taken as high priority.




Logic Diagram

Function Block

Figure 304: Cycler Function

## Logic Inputs

Table 74: Logic Inputs of Cycler Function

| Input Name | Input Value | Logic Value | Description |
|---|---|---|---|
| disable | unconnected | 0 | Normal operation |
| | VAL != 0.0 | 1 | Disable block, output = 0 |
| | 0 | 0 | Normal operation |
| | invalid | 0 | Normal operation |
| override Off | unconnected | 0 | Normal operation |
| | VAL != 0.0 | 1 | Turns off stages as minOn time allows |
| | 0 | 0 | Normal operation |
| | invalid | 0 | Normal operation |

Analog Inputs

Table 75: Analog Inputs of Cycler Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| In (%) | 0 | 100 | uncon-nected | stgsAct = 0 |
| | | | invalid | in = 0% |
| maxStgs | 1 | 255 | uncon-nected | stgsAct = 0 |
| | | | invalid | maxStgs = 1 |
| minOn (sec) | 0 | 65535 | uncon-nected | stgsAct = 0 |
| | | | invalid | stgsAct = 0 |
| minOff (sec) | 0 | 65535 | uncon-nected | stgsAct = 0 |
| | | | invalid | stgsAct = 0 |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| intstgOn (sec) | 0 | 65535 | uncon-nected | stgsAct = 0 |
| | | | invalid | stgsAct = 0 |
| intstgOff (sec) | 0 | 65535 | uncon-nected | stgsAct = 0 |
| | | | invalid | stgsAct = 0 |

## Output

Table 76: Output of Cycler Function

| Output Name | Range | Description |
|---|---|---|
| STAGES_ ACTIVE | 0 to +100% | The number of stages active (on) |

## Configuration

1. Specify CPH from 0 to 60.
2. Specify Anticipator Authority from 0 to 200%. Typical value is 100%.
3. Specify hysteresis from 0 to 100.

Table 77: Configuration of Cycler Function

| Name | Range/Value | Description |
|---|---|---|
| cph | 0 to 60 | You can specify CPH from 0 to 60. |
| hyst | 0 to 100 | You can specify hysteresis from 0 to 100. A value of 0 results in an effective hysteresis of: hyst = 100/maxStgs when CPH = 0 (stager operation) or hyst = anticipatorAuthority/max-Stgs/2 when CPH != 0 (cycler operation). |
| anticipatorAuthority | 0 to 200 | You can specify Anticipator Authority from 0 to 200%. Typical value is 100%. |
| intstgOff (sec) | 0 to 255 | You can specify intstgOff time in seconds under conditions. |

## Cycler Functionality

The Cycler function is the traditional anticipator cycling algorithm used in Honeywell thermostats. Input is either P or PI space temperature error in % (0-100). Standard (recommended) settings are cph = 3 for cooling, cph = 6 for heating, anticAuth = 100%, hyst = 100%/maxStgs/2. Note that for multiple stage cyclers the PID block feeding this block should have an appropriately large throttling range to achieve smooth behavior.

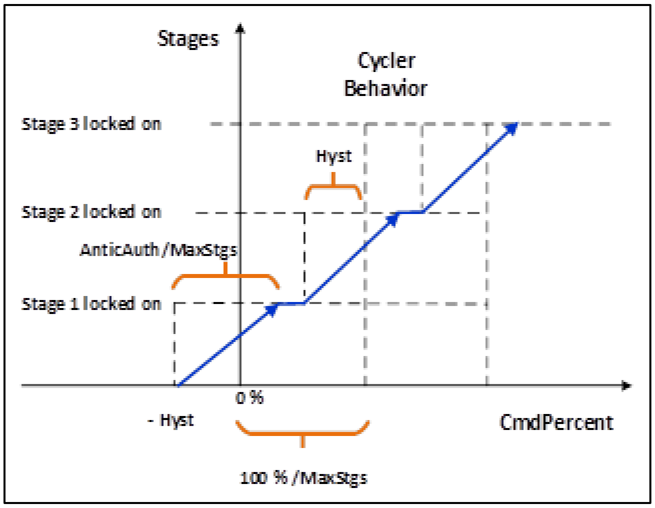
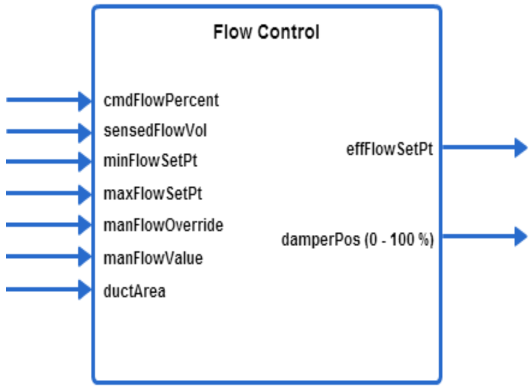


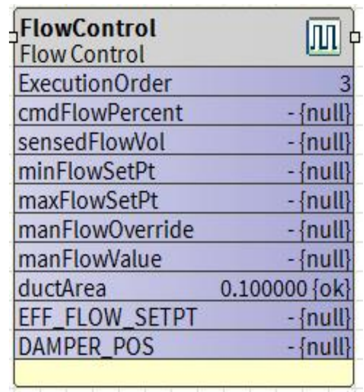

Figure 305: Cycler Functionality

## FlowControl

This function is a Variable Air Volume (VAV) Damper Flow Controller. Traditionally, this is the second half of a pressure independent VAV box cascade control strategy. Typically, the input would come from the output of a PID block controlling space temperature.




Logic Diagram




Function Block

Figure 306: Flow Control Function

Whenever there is a certain configuration change in Flow Control block (like duct area) if users expect behavior like Spyder controller, then they need to Stop and start the Sequenced Control Engine.

Steps to stop and start the sequenced control engine

1. Right click on Sequenced Control Program
2. Invoke Action request sequenced Control engine stop
3. Then invoke Start Sequenced Control Engine to start the engine

### Analog Inputs

Table 78: Analog Inputs of Flow Control Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| cmdFlowPercent (%) | 0 | < + infinity | unconnected | cmdFlowPercent = 0 |
| | | | invalid | Same as unconnected |
| sensedFlowVol | > = – infinity | < + infinity | unconnected | damperPos = cmdFlowPercent |
| | | | invalid | damperPos = cmdFlowPercent |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| minFlowSetPt | > = – infinity | < + infinity | unconnected | Switch to Pressure-dependent mode.<br>minFlowSetPt = 20<br>maxFlowSetPt = 100<br>effFlowSetPt = invalid |
| | | | invalid | Same as unconnected |
| maxFlowSetPt | > = – infinity | < + infinity | unconnected | Switch to Pressure-dependent mode.<br>minFlowSetPt = 20<br>maxFlowSetPt = 100<br>effFlowSetPt = invalid |
| | | | invalid | Same as unconnected |
| manFlowOverride | > = – infinity | < + infinity | unconnected | Normal operation |
| | | | invalid | Same as unconnected |
| manFlowValue | 0 | < + infinity | unconnected | value = invalid |
| | | | invalid | Same as unconnected |
| ductArea | > 0 | < + infinity | invalid | effFlowSetPt = invalid & damperPos = (100* minFlowSetPt / maxFlowSetPt) |
| | | | unconnected | Same as invalid |
| | | | VAL < = 0 | Same as invalid |

## Output

Table 79: Outputs of Flow Control Function

| Output Name | Range | Description |
|---|---|---|
| EFF_FLOW_SETPT | Any floating-point value | Effective Flow setpoint |
| DAMPER_POS | Any floating-point value | Damper position |

## Configuration

Table 80: Configuration of Flow Control Function

| Name | Range/Value | Description |
|---|---|---|
| units | 0 to 2 | You can specify the units from 0 to 2. 0 = flow (cfm), area(ft**2); 1 = flow (Lps), area (m**2); 2 = flow (cmh), area (m**2). Default is zero (0). |
| motor-Speed | 1 to 255 | You can specify the motor speed from 1 to 255 seconds per 90 degrees. Default is 90. |

- Specify the units from 0 to 2.
    - 0 = flow (cfm), area($ft^2$)
    - 1 = flow (Lps), area ($m^2$)
    - 2 = flow (cmh), area ($m^2$)
- Specify the motor speed from 1 - 255 sec per 90 °. Default is 90.

The Flow Controller function calculates an effective flow control set point (effFlowSetPt) and outputs 0% - 100% command to drive a VAV box damper. The commanded flow set point (in percent) from a temperature control signal is mapped into the effective flow set point such that 0% maps to the min flow set point and 100% maps to the max flow set point. The sensedFlowVol input is the volumetric flow into the box, if it is invalid (sensor fails) the damper is driven in a pressure dependent mode where:

Output = 100% * (minSP/maxSP) + (1 – minSP/maxSP) * cmdPercent.

If either flow MinSP, or MaxSP is invalid, the Output = 20% + .8 * cmdPercent.

The Units parameter sets the units being used for the flow sensor, set points, and the duct area where 0 = cfm (flow) and $ft^2$ (area), 1 = L/s (flow) and $m^2$ (area), 2 = $m^3$/hr (flow) and $m^2$ (area). The cmdFlowPercent input is the input in percent from the temperature control logic. The DuctArea is the area of an air flow duct and you can calculate it by using the height and width of the duct, if the duct is in square shape (for circular duct it can be calculated by using diameter of the duct). DuctArea is required for the control algorithm.

The control loop is implemented in air velocity to simplify loop tuning. The motorSpeed parameter is the time the actuator being used takes to travel a full 90° stroke in seconds (this is used to automatically adjust the control gains).

The manFlowOverride input allows the flow set point to be selectively overridden based on the following codes: (taken from snvt_hvac_overid)

- 0 and all others not listed = no override (normal operation)

- 2 = effFlowSetPt is set to the ManFlowValue input
- 6 = effFlowSetPt is set to the minFlowSetPt input
- 7 = effFlowSetPt is set to the maxFlowSetPt input

Manual flow override is particularly useful when trying to make the box easy to be balanced.

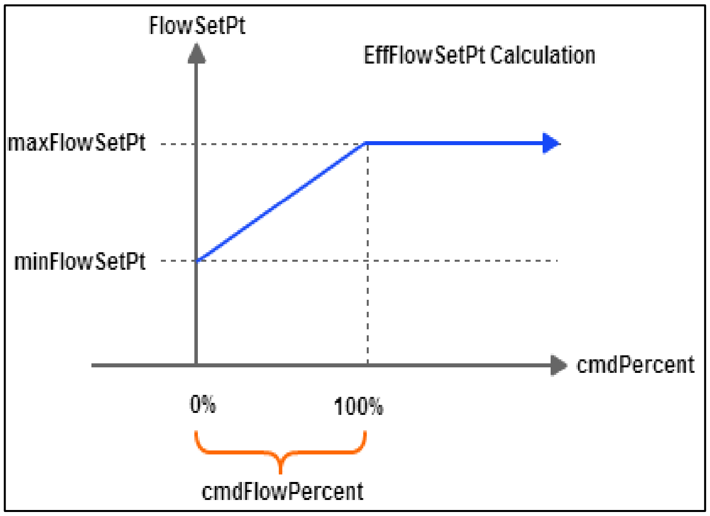
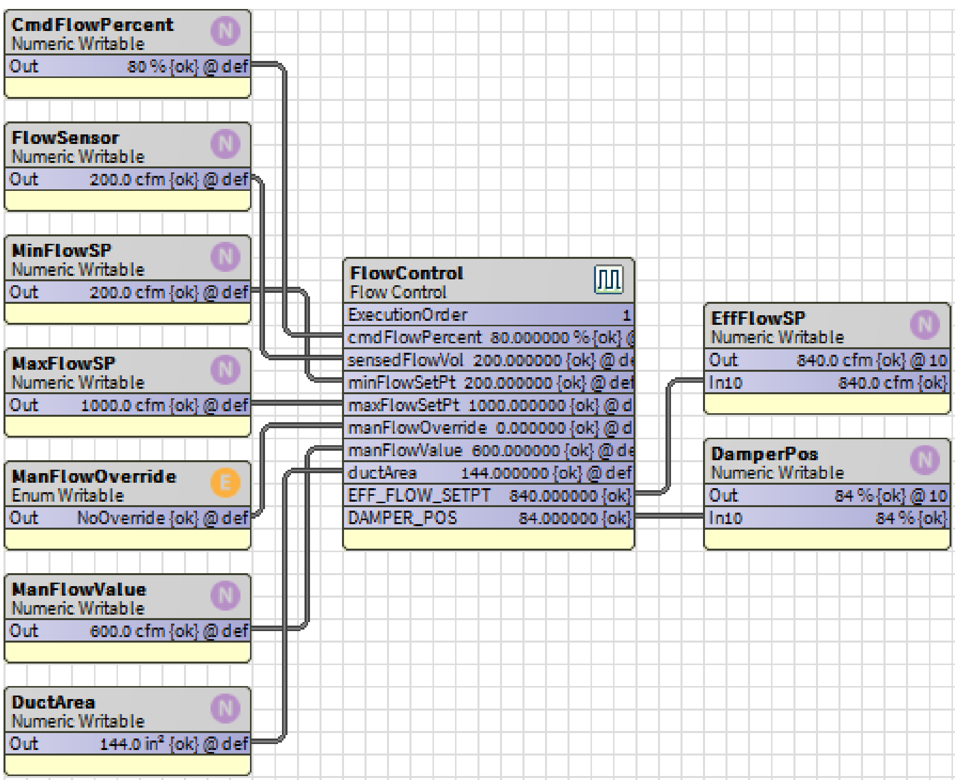
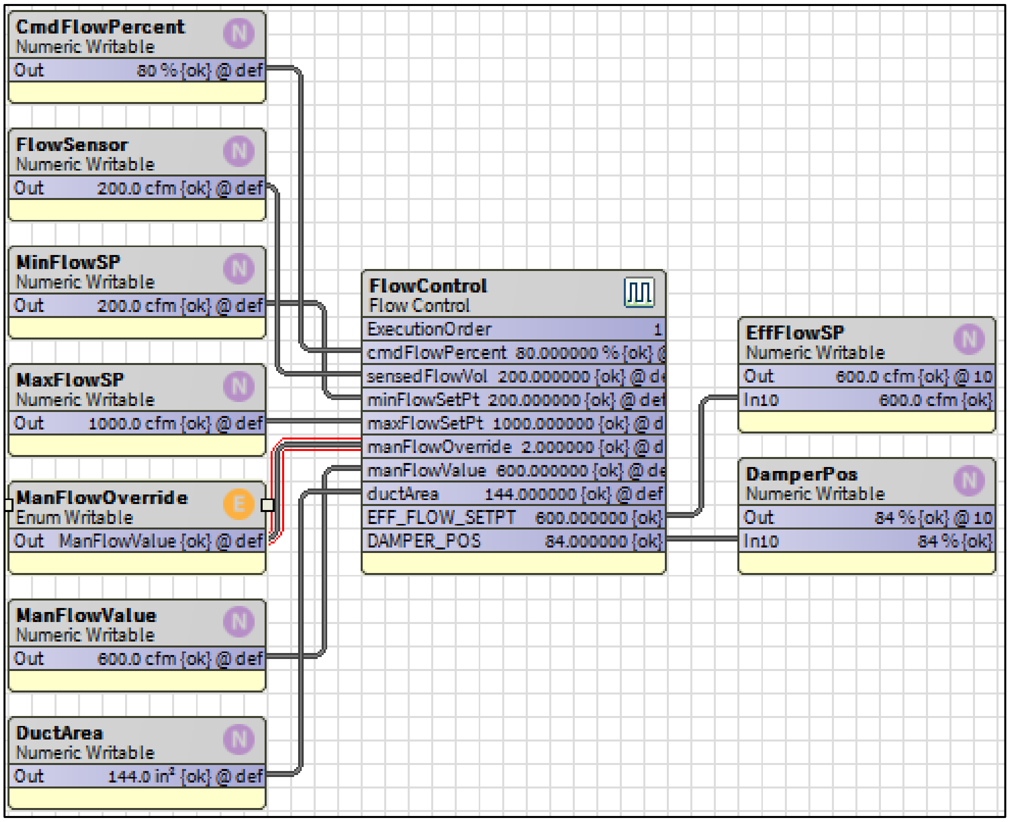



Figure 307: Effective Flow Setpoint Calculation

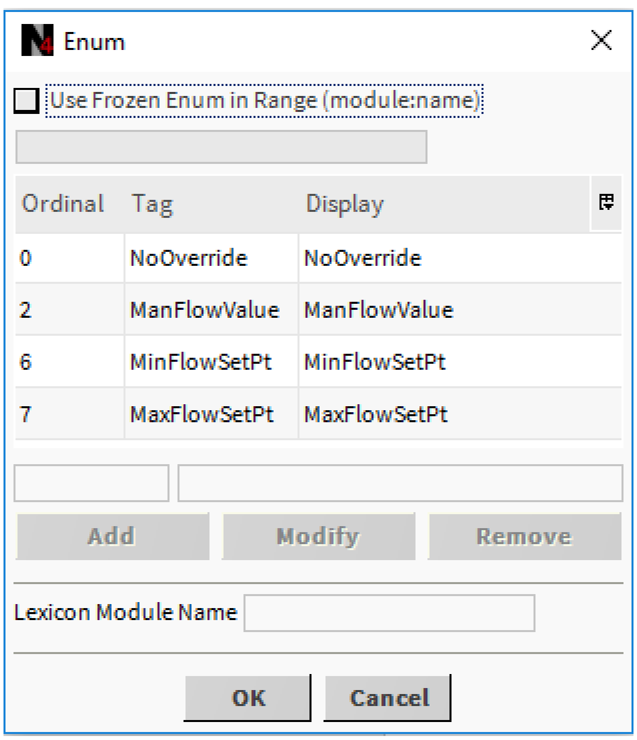
Example: The flow Setpoint value will be calculated based in the Cmd Percent value, Min Flow Setpoint & Max Flow Setpoint as below:




But if the Manual Flow Override value is other than 0 then the Flow Setpoint value will be either Min Flow Setpoint or Max Flow Setpoint or Man Flow Value based upon selection.




The setting for Man Flow Override need to be similar as below:



## PID

The Proportional Integral Derivative (PID) controller compares a measured value from a process with a reference setpoint value. The difference (or error signal) is then used to calculate a new value for a manipulatable input to the process that brings the process measured value back to its require setpoint. Unlike simpler control algorithms, the PID controller can adjust the process outputs based on the history and rate of change of the error signal, which gives more accurate and stable control. The output limit is -200 to +200 and you can choose the option 0 to 100 separately.

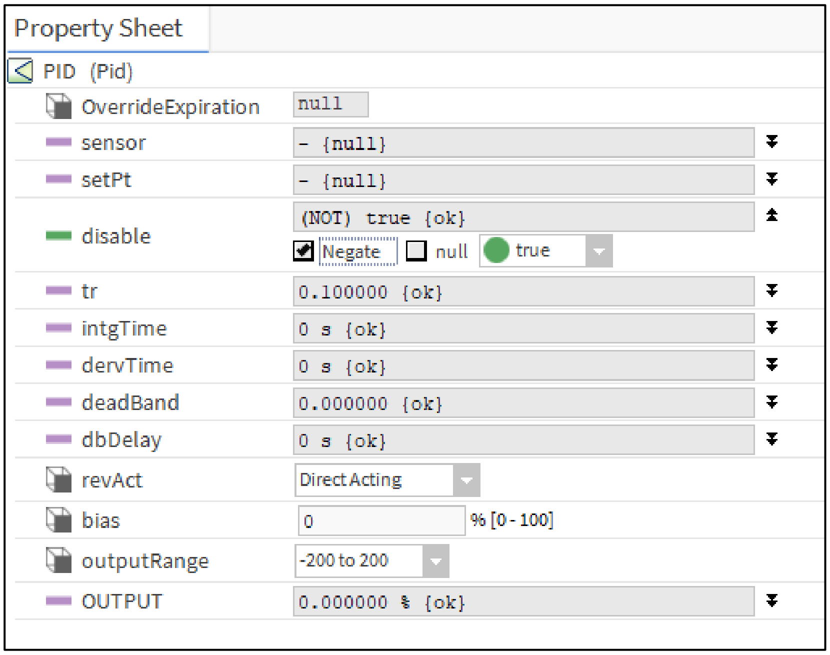
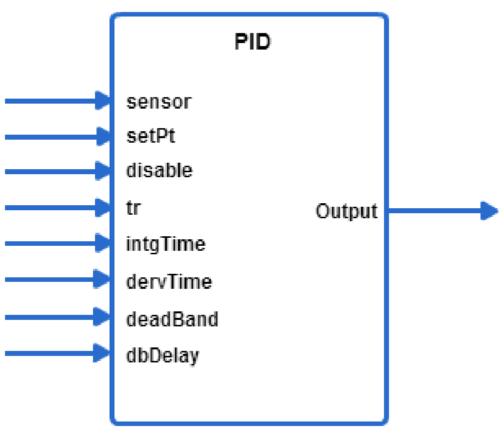
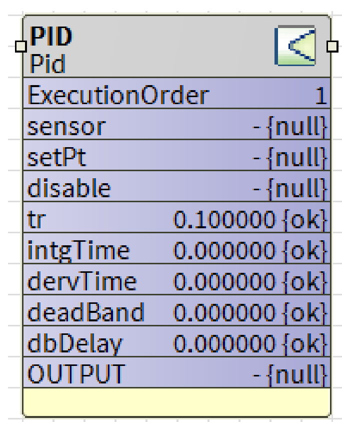



Figure 308: PID Property Sheet

Unconfigured is taken as high priority compared to Negate. If nothing is configured and Negate is applied nothing will get effected. Unconfigured will be taken as high priority.




Logic Diagram




Function Block

Figure 309: PID Function Block

In a PID loop, correction is calculated from the error in three ways:

- Cancel out the current error directly (Proportional)

- The amount of time the error has continued uncorrected (Integral)
- Anticipate the future error from the rate of change of the error over time (Derivative)
  - Err = Sensor – Set Point
  - Kp = 100/Proportional Band
  - Ti = Integral Time (seconds)
  - Td = Derivative Time (second)
  - Bias = Proportional offset (%)

Output (%) =Bias +Kp * Err + Kp/Ti $\int_0^t (Err)dt$ + Kp* Td*dErr/dt

Whenever there is a certain configuration changes in PID block (like integral time) if users expect behavior like Spyder controller, then they need to Stop and start the Sequenced Control Engine.

Steps to stop and start the sequenced control engine

1. Right click on Sequenced Control Program
2. Invoke Action request sequenced Control engine stop
3. Then invoke Start Sequenced Control Engine to start the engine

## Logic Inputs

Table 81: Logic Inputs of PID

| Input Name | Input Value | Logic Value | Description |
|---|---|---|---|
| disable | unconnected | 0. | PID function runs. |
| | VAL != 0.0 | 1 | PID function is disabled. Output set to zero |
| | 0 | 0 | PID function runs. |
| | invalid | 0 | PID function runs. |

## Analog Inputs

Table 82: Analog Inputs of PID

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| sensor | >=– infin-ity | <+ infinity | uncon-nected | PID function disabled. Output is set to 0. |
| | | | invalid | |
| setPt | | <+ infinity | uncon-nected | PID function disabled. Output is set to 0 |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| | >=– infin-ity | | invalid | Same as unconnected |
| tr | 0< | <+ infinity | uncon-nected | PID function disabled. Output is set to 0. |
| | | | invalid | Same as unconnected |
| | | | 0 | PID function disabled. Output is set to 0 |
| | | | VAL < low | val = low |
| intgTime (sec). | 0 | <+ infinity | uncon-nected | PID function disabled. Output is set to 0. |
| | | | invalid | Disable Integral Action. |
| | | | 0 | Disable Integral Action. |
| | | | VAL < low | IT = low |
| dervTime (sec) | 0 | <+ infinity | uncon-nected | Disable Derivative action. |
| | | | invalid | Disable Derivative action |
| | | | 0 | Disable Derivative action. |
| | | | VAL < low | DT = low |
| deadBand | 0 | < tr | uncon-nected | Same as 0 input |
| | | | invalid | Same as 0 input |
| | | | VAL < low or VAL >= tr | DB = 0 |
| | | | 0 | Disable deadband action |
| dbDelay | 0 | 65565 | | Deadband delay |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| (sec) | | | uncon-nected | Same as 0 input |
| | | | invalid | Same as 0 input |
| | | | 0 | Deadband action enabled without delay |
| | | | VAL < low | Dead band delay = low |

## Output

Table 83: Output of PID

| Output Name | Range | Description |
|---|---|---|
| OUTPUT | –200 to +200% or 0-100 | Output (%) =Bias +Kp * Err + Kp/Ti $\int_0^t (Err)dt$ + Kp* Td*dErr/dt |

## Setpoints

Table 84: Setpoints of PID

| Name | Range/Value | Description |
|---|---|---|
| revAct | • 0 to 2 | You can specify revAct: 0 = Direct acting, 1 = reverse acting, 2 = sign of the tr input determines direction: (+) = Direct acting, (-) = Reverse acting. |
| bias | 0 to 100% | You can specify the Bias: 0 to 100%. |
| Output Range | 0 to100 or -200 to +200 | You can specify the required Output range |

## Configuration

1. Specify Action
   - 0 = Direct acting
   - 1 = Reverse acting
2. Specify the bias: 0 to 100%.

When Disable/Initialize input is TRUE, the Output and the integral are set to 0 and the block stops running.

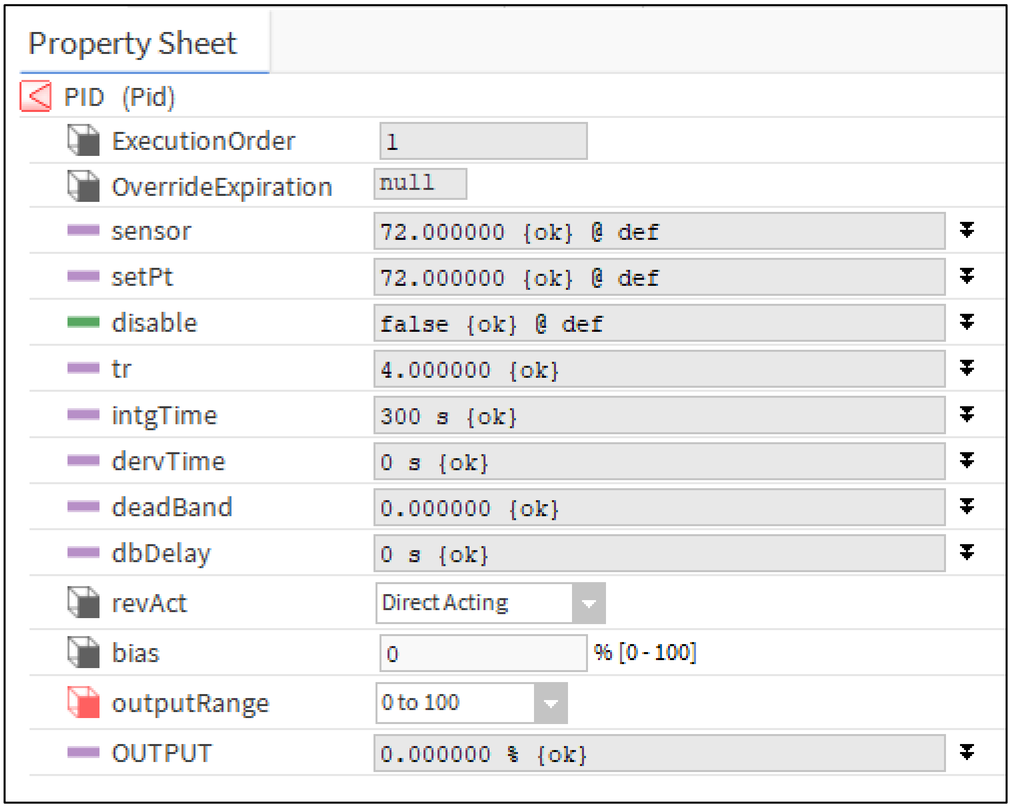
If Direct/Reverse is set to reverse, Err term is set to –Err.

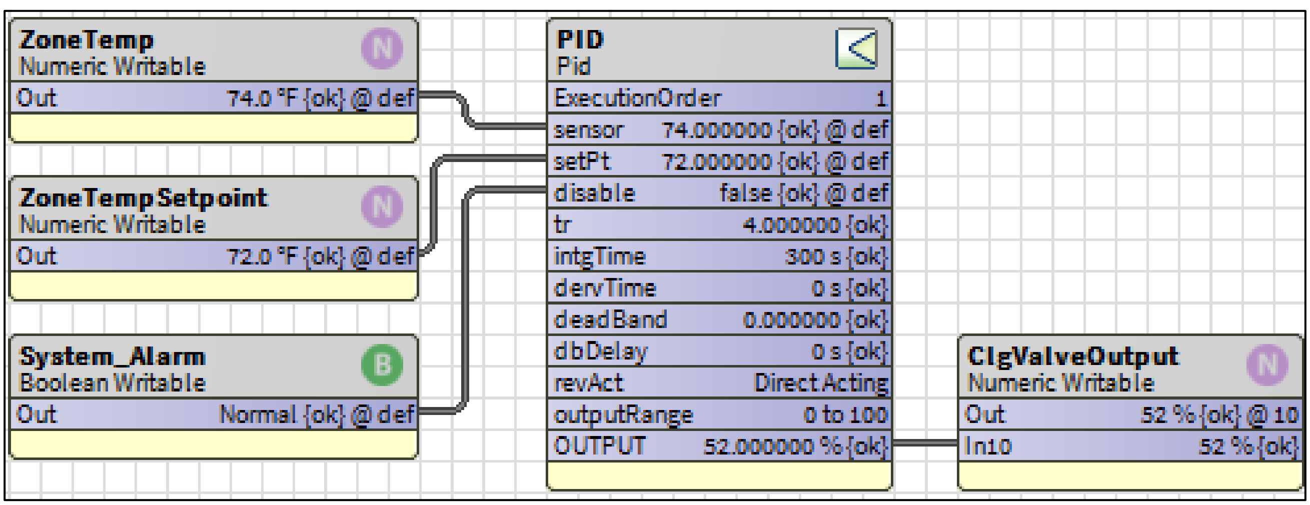
When Err < Deadband, Err is set to zero until Deadband Delay time has elapsed and Err is still in the dead band.

To prevent integral wind up, the integral portion of the total error output is limited to 100%.

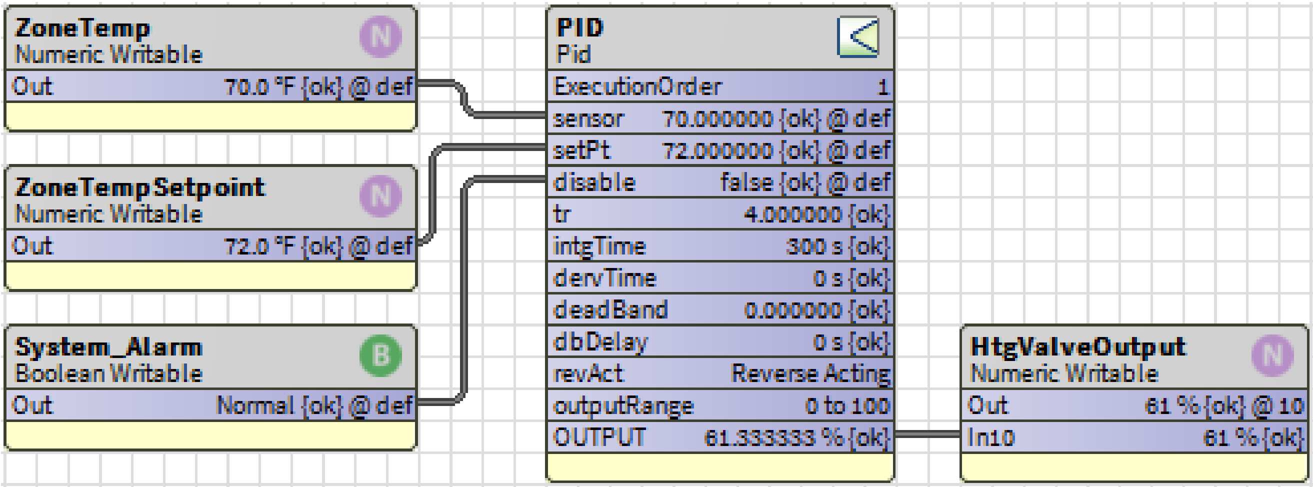
From iteration to iteration, the Function Block keeps track of the old proportional error, integral error, and dead-band timer. On power up/reset these are cleared.

Example: To control a Cooling Coil Valve the PID need to be configured as Direct acting and set the Parameters as per requirement.




To control a Heating Coil Valve the PID need to be configured as Reverse Acting.


ZoneTemp
Numeric Writable
Out 70.0 °F {ok} @ def
ZoneTempSetpoint
Numeric Writable
Out 72.0 °F {ok} @ def
System_Alarm
Boolean Writable
Out Normal {ok} @ def
PID
Pid
ExecutionOrder 1
sensor 70.000000 {ok} @ def
setPt 72.000000 {ok} @ def
disable false {ok} @ def
tr 4.000000 {ok}
intgTime 300 s {ok}
dervTime 0 s {ok}
deadBand 0.000000 {ok}
dbDelay 0 s {ok}
revAct Reverse Acting
outputRange 0 to 100
OUTPUT 61.333333 % {ok}
HtgValveOutput
Numeric Writable
Out 61 % {ok} @ 10
In10 61 % {ok}

## RateLimit

This function creates an output that follows the input but prevents the output from changing faster that the specified rates depending on direction.

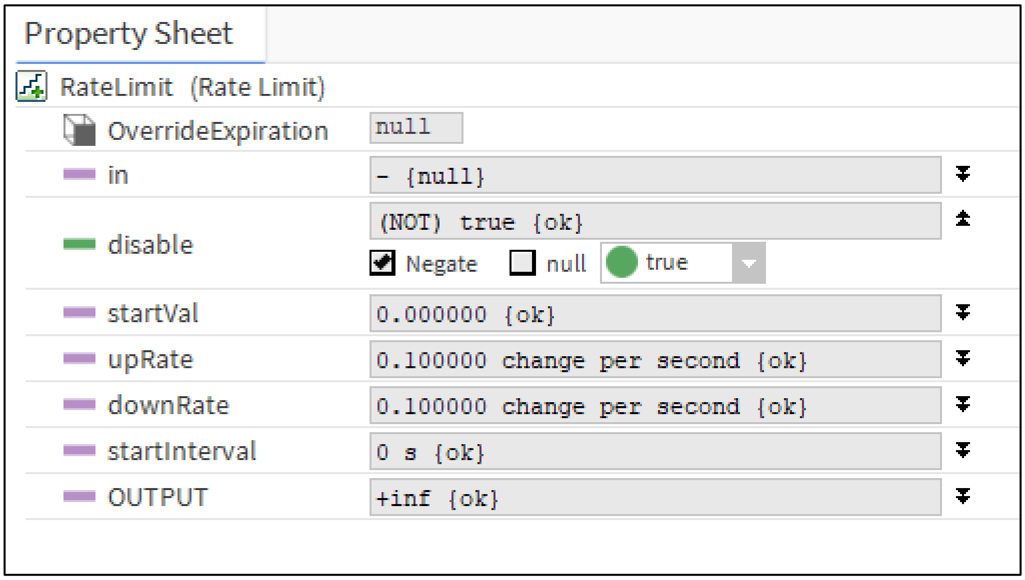



Figure 310: RateLimt Property Sheet

Unconfigured is taken as high priority compared to Negate. If nothing is configured and Negate is applied nothing will get effected. Unconfigured will be taken as high priority.

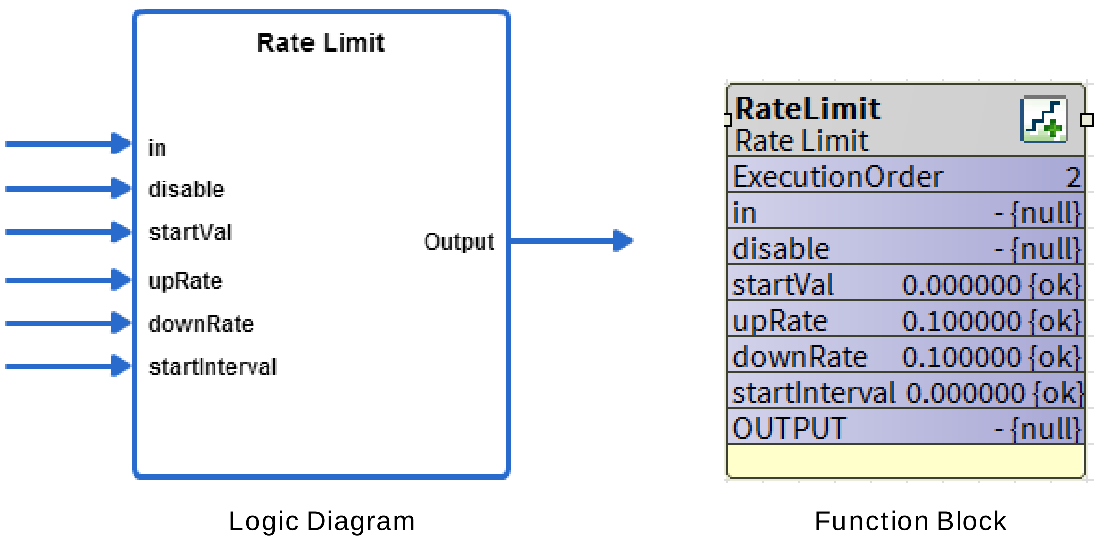



Figure 311: Rate Limit Function

### Logic Inputs

Table 85: Logic Inputs of Rate Limit Function

| Input Name | Input Value | Logic Value | Description |
|---|---|---|---|
| disable | unconnected | 0. | The function executes. |

| Input Name | Input Value | Logic Value | Description |
|---|---|---|---|
| | VAL != 0.0 | 1 | The function is disabled. |
| | 0 | 0 | The function executes. |
| | invalid | 0 | The function executes. |

## Analog Inputs

Table 86: Analog Inputs of Rate Limit Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| in | >= – infinity | <+ infinity | unconnected | In= 0.0 |
| | | | invalid | In = Invalid |
| | | | Valid | In = value |
| startInterval (sec) | 0 | 65535 | unconnected | Start interval = 0 |
| | | | invalid | Start interval = 0 |
| | | | 0<val<max float | Limit Start interval value 0 to 65535.0 sec. |
| | | | <0 | StartInterval = 0 |
| startVal. | >= – infinity | <+ infinity | | Output assumes the start value when the function is disabled |
| | | | unconnected | If disable=1 then Out=in |
| | | | invalid | If disable=1 then Out=in |
| upRate (chg/sec) | 0< | <+ infinity | unconnected | No limit on upRate |
| | | | invalid | No limit on upRate |
| | | | 0 | No limit on upRate |
| | | | <0 | upRate = 0 (no limit on upRate) |
| downrate (chg/sec) | 0< | <+ infinity | unconnected | Rate no limit on downRate |
| | | | invalid | No limit on downRate |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| | | | 0 | No limit on downRate |
| | | | <0 | downRate=0 (no limit on upRate) |

## Output

Table 87: Output of Rate Limit Function

| Output Name | Range | Description |
|---|---|---|
| OUTPUT | Any floating-point value | Rate limit |

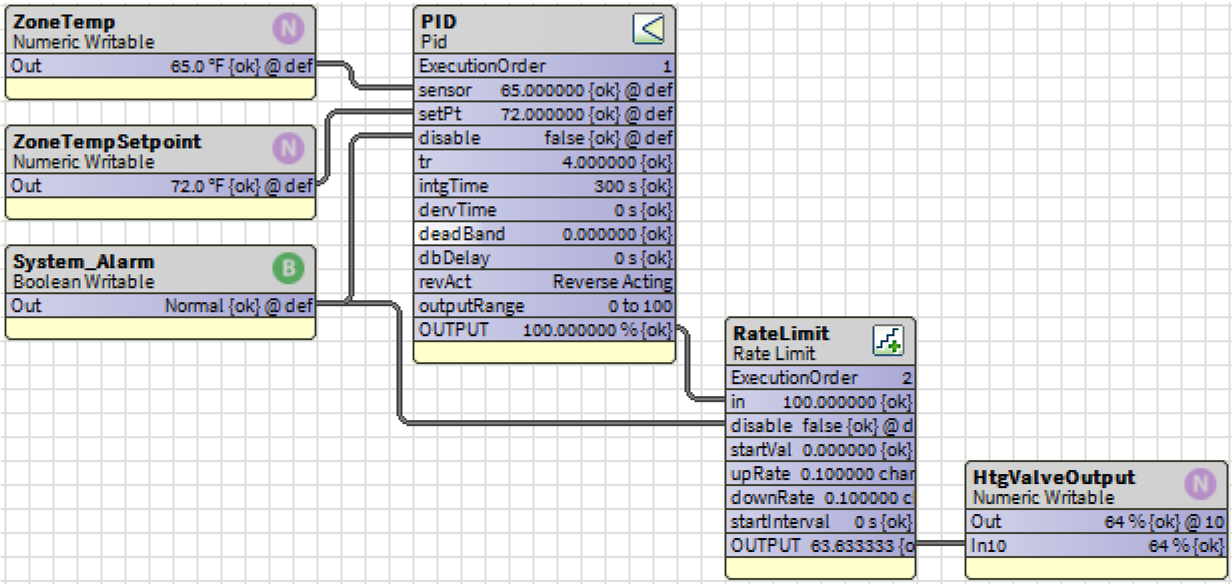
## Operation

The value StartInterval (Sec) limits the output after the rate limit function is enabled (disable input set to 0) and the StartInterval time is still in process. RateLimit uses the startVal input as the default output during disable.

- If the RateLimit function is disabled (disable input set to 1) the output is set to StartVal.
- After RateLimit is enabled (disable set to 0), the StartInterval timer counts down from the StartInterval number of seconds and during this time the output is Ratelimited.
- When the timer expires (and RateLimit is enabled), the out value is exactly what the input (in) is set to and there is no longer rate limiting.
- If the StartInterval second is set to 0 (and ratelimit is enabled), the output is Ratelimited.
- During RateLimit the output moves at the maximum allowed rate towards the new input value each second.
- UpRate controls the rate in a more positive direction, and DownRate controls the rate in a more negative direction. UpRate set to zero means the uprate limit is not enforced. DownRate set to zero means the downrate limit is not enforced.
- Out is set to StartVal before rate limiting is enabled (disable set to 0).
- From iteration to iteration, the Function Block keeps track of the start timer. On power/up/reset, this is set to the StartInterval.

Where cph = 0 means stager functionality, and cph = 1-60 gives thermostat cycler functionality.

Example: Rate Limit is used to slow down the PID output changes


ZoneTemp
Numeric Writable
Out 65.0 °F {ok} @ def
ZoneTempSetpoint
Numeric Writable
Out 72.0 °F {ok} @ def
System_Alarm
Boolean Writable
Out Normal {ok} @ def
PID
Pid
ExecutionOrder 1
sensor 65.000000 {ok} @ def
setPt 72.000000 {ok} @ def
disable false {ok} @ def
tr 4.000000 {ok}
intgTime 300 s {ok}
dervTime 0 s {ok}
deadBand 0.000000 {ok}
dbDelay 0 s {ok}
revAct Reverse Acting
outputRange 0 to 100
OUTPUT 100.000000 % {ok}
RateLimit
Rate Limit
ExecutionOrder 2
in 100.000000 {ok}
disable false {ok} @ d
startVal 0.000000 {ok}
upRate 0.100000 char
downRate 0.100000 c
startInterval 0 s {ok}
OUTPUT 63.633333 {o
HtgValveOutput
Numeric Writable
Out 64 % {ok} @ 10
In10 64 % {ok}

## Stager

This function is a generic stage driver or a Thermostat Stage Cycler dependent on the value of the CPH parameter. The input range (In) for the stager block is -200 to +200.

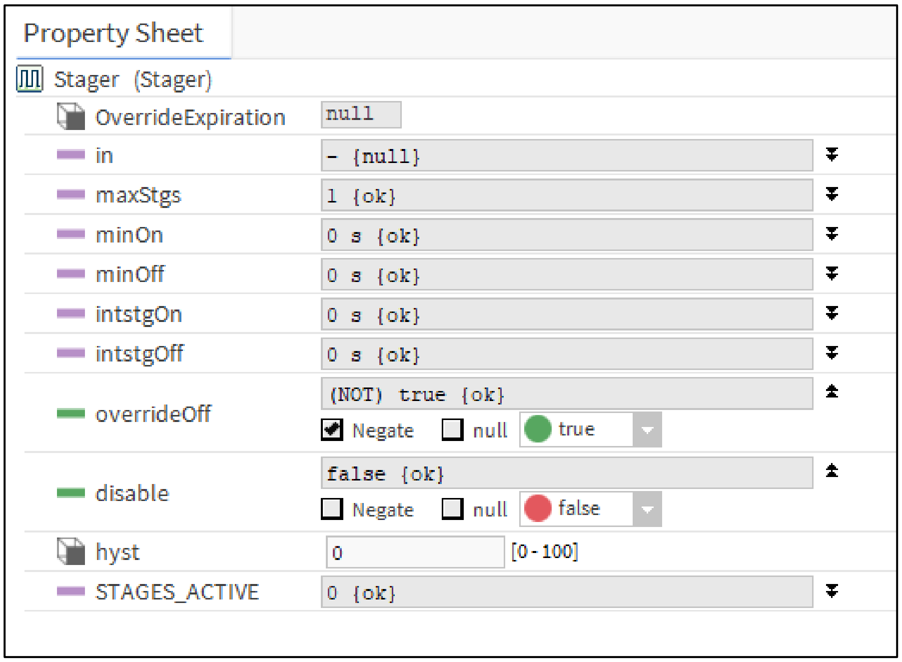
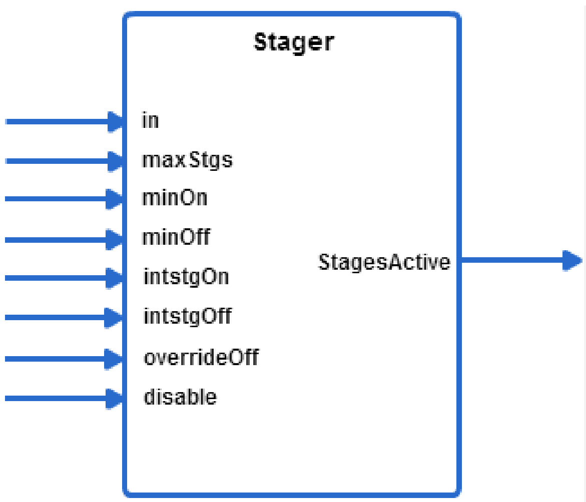
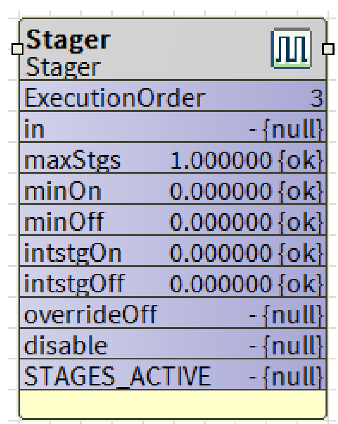



Figure 312: Stager Property Sheet

Unconfigured is taken as high priority compared to Negate. If nothing is configured and Negate is applied nothing will get effected. Unconfigured will be taken as high priority.




Logic Diagram




Function Block

Figure 313: Stager Function Block

## Logic Inputs

Table 88: Logic Inputs of Stager Function

| Input Name | Input Value | Logic Value | Description |
|---|---|---|---|
| disable | unconnected | 0 | Normal operation |
| | VAL != 0.0 | 1 | Disable block, output = 0 |
| | 0 | 0 | Normal operation |
| | invalid | 0 | Normal operation |
| override Off | unconnected | 0 | Normal operation |
| | VAL != 0.0 | 1 | Turns off stages as minOn time allows |
| | 0 | 0 | Normal operation |
| | invalid | 0 | Normal operation |

### Analog Inputs

Table 89: Analog Inputs of Stager Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| in % | 0 | 100 | unconnected | stgsAct = 0 |
| | | | invalid | in = 0% |
| maxStgs | 1 | 255 | unconnected | stgsAct = 0 |
| | | | Invalid | maxStgs = 0 |
| minOn (sec) | 0 | 65535 | unconnected | stgsAct = 0 |
| | | | invalid | stgsAct = 0 |
| minOff (sec) | 0 | 65535 | unconnected | stgsAct = 0 |
| | | | invalid | stgsAct = 0 |
| intstgOn (sec) | 0 | 65535 | unconnected | stgsAct = 0 |
| | | | invalid | stgsAct = 0 |
| intstgOff | 0 (sec) | 65535 | unconnected | stgsAct = 0 |
| | | | invalid | stgsAct = 0 |

## Output

Table 90: Output of Stager Function

| Output Name | Range | Description |
|---|---|---|
| STAGES_ ACTIVE | 0 to +100% | The number of stages active (on) |

## Setpoints

Table 91: Setpoint of Stager Function

| Name | Range / Value | Description |
|---|---|---|
| hyst | 0 to 100 | User-specified value |

## Configuration

Specify hysteresis from 0 to 100.

## Stager Functionality

The Stager Function takes a 0-100 percent (typically PID error) input and determines how many stages to turn on. The 0-100 percent input range is divided evenly between how many stages are configured in maxStgs. The first stage is turned on at CmdPercent > 0 and off at CmdPercent < – Hyst. As shown in following illustration the general criterion for turning on stage N is:

CmdPercent > (N –1) * 100% / maxStgs.

For turning off stage N the criterion is:

CmdPercent < (N –1) * 100% / maxStgs – Hyst

From iteration to iteration, the Function Block keeps track of the on timer, off timer, anticipator, and CPH multiplier. On power up/reset, the off timer and anticipator are cleared, the on timer is set equal to the inter-stage on time and the CPH multiplier is recalculated.

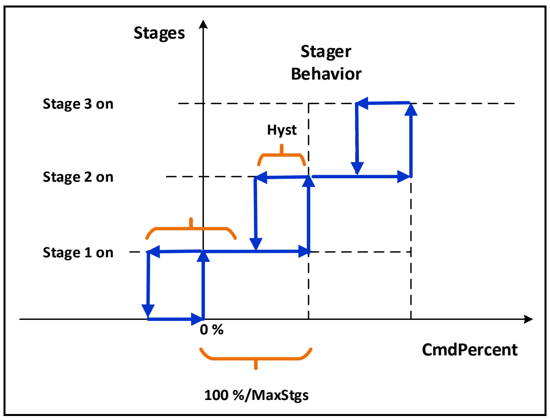



Figure 314: Stager Functionality

When override is TRUE, active stages are shed (turned off) based on min on and interstage timers regardless of the CmdPercent input. Output is the number of stages active (0-MaxStages) which can be sent to the StageDriver function block. Configuration parameters include:

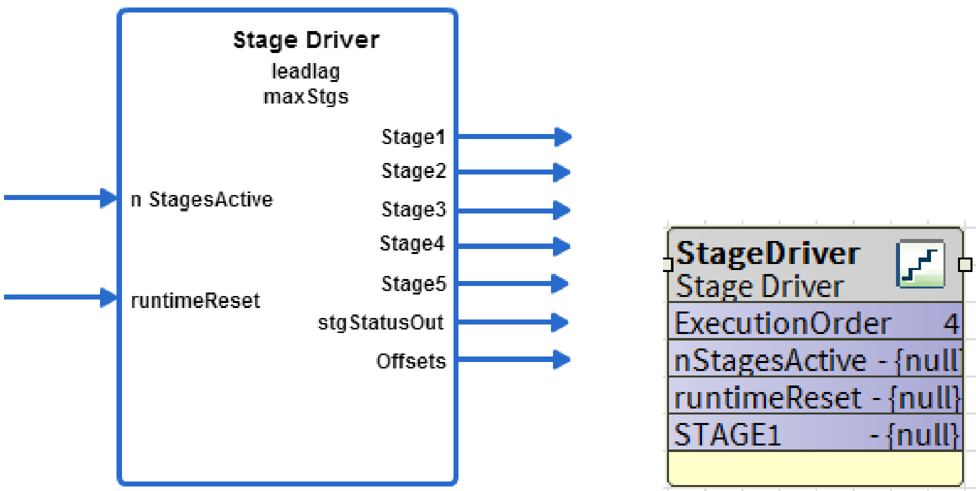
- maxStgs is the maximum stages available to turn on.
- CPH (non-zero) is max cycle rate in Cycles Per Hour when input is halfway between stages available, and AnticAuth is at the default value (100%). CPH = 0 means the Stager logic is performed and has no other effect.
- Hyst is the switching differential around the switch points in % error. (Range: 0 < Hyst < 100 / maxStgs)
- AnticAuth (cycler only (CPH != 0)) is the anticipator authority, which allows adjustment of the cycling behavior. It represents the max amount of “fake” error in % that is input into the switching logic when maxStgs are turned on. (Range 0 < AnticAuth < 200.)
- minOnTime is the minimum time a stage must be on once it is turned on.
- minOffTime is the minimum time a stage must be off once it is turned off.
- intstgOn is the minimum time before the next stage can be turned on after the previous one is turned on.
- intstgOff is the minimum time before the next stage can be turned off after the previous one is turned off.

## StageDriver

The StageDriver function takes the number of active stages as an input and determines which stages to energize or de-energize based on the lead/lag strategy chosen while configuring the stage driver control block. The stage driver works with StageDriverAdd to distribute additional stages above those provided in the stage driver and maintains nonvolatile runtime total and digital stage status information for each stage.

When the equal runtime strategy is selected, the engineering tool must allocate a block of ControlNonvolatile public variables for use as run timers, and convey the base PVID of that block to the STAGEDRIVER function block via the offset output. When other strategies are selected (STD or FOFO), no run timers are required, and the offset output content is ignored.

The stgStatusOut output is also a legacy artifact that is no longer required. No connection is made to the STAGEDRIVER_ADD block(s) associated with the STAGEDRIVER. Instead, the STAGEDRIVER_ADD blocks must follow the STAGEDRIVER block directly (in the execution order).




Logic Diagram

Function Block

Figure 315: Stage Driver Function

### Analog Inputs

Table 92: Analog Inputs of Stage Driver Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| nStagesActive | 0 | 255 | unconnected | Stages all off |
| | | | invalid | Stages all off |
| runtimeReset | 0 | 255 | Unconnected | No action to reset; runtime can accumulate |
| | | | Invalid | No action; runtime can accumulate |
| | | | Value=0 | No action; runtime can accumulate |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| | | | 1<=VAL<=255 | Stage runtime for stage VAL is reset to 0; runtime for this stage does not accumulate—should be reset VAL to 0 to allow accumulation of runtime. |

Outputs

Table 93: Outputs of Stage Driver Function

| Output Name | Range | | Description |
|---|---|---|---|
| | Low | High | |
| Stage1 | 0 | 1 | Stage 1 output |
| Stage2 | 0 | 1 | Stage 2 output |
| Stage3 | 0 | 1 | Stage 3 output |
| Stage4 | 0 | 1 | Stage 4 output |
| Stage5 | 0 | 1 | Stage 5 output |
| stgStatusOut | | | Output values to connect to StageDriverAdd block. The floating number must be converted to an integer and then converted to a two-byte value. The upper byte (value right shifted 8 bits) is the maxStgs info and the lower byte (value AND 0xFF) is the stageStatus offset to reference the starting location in digital memory for the stageStatus bytes. |
| offset | | | Float value has two components – after conversion to a two-byte unsigned integer value, the upper byte is the offset of the number of nonvolatile entries to get to the start of the stage runtime storage (used only for leadLag=LL-RUNEQ) and the lower byte is the offset of number of digital memory locations to the start. |

Configuration

Specify the maximum number of stages (maxStgs) from 1 to 255.

Specify the lead lag

- LL_FOLO = 0  first on last off
- LL_FOFO = 1  first on first off
- LL_RUNEQ = 2  runtime equalization for lowest runtime

If the leadLag is outside of the range of 0 - 2 then stages are initialized to off and not commanded.

## Inputs

nStagesActive is the input number of stages to be distributed to on/off values to individual stages.

runtimeReset is the stage number. Runtime to be reset to 0 (zero), if the lead-lag parameter is set to LL RUNTIME, 0 (zero) or unconnected results in no reset occurring.

This value must be returned to 0 (zero) to allow the reset stage number to resume counting. It is valid only when leadLag is set to LL RUNTIME. The stage runtime values are only allocated and updated if the leadLag config is set to LL RUNTIME. The runtime for each stage is stored as a floating-point number in intervals of 1 min.

The stages are sampled once a minute and if the stage is on, the stage runtime accumulator number for that stage is incremented by one minute. The range of values for an integer number stored as a float is from – 16,777,216 to 16,777,216. If the runtime is stored in minutes starting at 0 to 16, 777, and 216, the range of runtime is from 0 to 31.92 years of run time.

## Outputs

Stage1, stage2, stage3, stage4, and stage5 are individual outputs that represent on or off values. These are outputs that are turned on in a different order depending on the leadLag strategy.

stgStatusOut is connected from StageDriver to the StageDriverAdd block and gives a floating-point number combined to hold two pieces of information, offset in the Common Memory to the StageBitStatus values and maximum number of stages are available. This information is used by the StageDriverAdd to find the correct offset to command which stages to turn on or off.

The floating value can be converted into an integer and ANDed with 0xFF. It gives the value of the stageStatus Offset. The floating value stgStatusOut converted to an integer and right-shifted 8 bits gives the byte value of the maxStgs. These values are needed to allow the StageDriverAdd to work properly.

The values in stgStatusOut are created by the StageDriver stage and no tool calculation is required.

Offsets store the public Variable ID to a float value created by the tool to allocate storage memory and reference for stage status in digital memory and stage runtime in nonvolatile memory. There are two offsets stored inside the float value, one for runtime, and one for stage status. The offset, float value right-shifted 8 bits gives the number of nonvolatile float values from the beginning nonvolatile index (offset), where the runtime values are stored (one runtime value offset for each stage configured), and the offset ANDED with 0xff gives the number of digital values from the base, where the stageStatus is stored (one byte per up to 8 stages configured). Each digital memory location takes up 1 byte storage in calculating the offset.

Example

If three nonvolatile were already assigned and four digital outputs were already assigned before adding a StageDriver stage of nine stages with runtime accumulation, the offset float value would be 256 (3) + 4 = 772.0.

That means the tool would have 8 nonvolatile runtime locations starting at offset 3 from the base of nonvolatile memory. The tool allocates digital memory of two bytes for the stage status starting at offset of 4 from the base of digital memory. The tool sets this float value for offsets and allocates the memory, and then StageDriver uses this information to know where to look for stageStatus and stage runtime information.

The Float value that stores Offsets is composed of two values

- offsetStageRuntimer (byte)

  The float value converted to an integer and shifted 8 bits specifies the variable quantity offset to be applied to the beginning of nonvolatile memory variable numbers that indicates the starting variable number used to store the individual stage runtime values. This number is calculated by the configuration tool and is not changeable.

- offsetStageStatus (byte)

  The float value converted to an integer and ANDed with 0xFF specifies the variable number offset to be applied to the beginning of the digital memory area that indicates the starting variable number used to store the individual stage on/off values. This number is calculated by the configuration tool and is not changeable. This value is exported to other stages through the StageBitStatus output.

## Parameters

leadLag (Byte param:UBYTE) specifies whether the staging strategy should be:

- First on, last off (LL FILO = 0 - standard)
- First on, first off (LL FOFO = 1 - Rotating)
- Run time accumulation where next ON has lowest runtime and next OFF has highest runtime (LL RUNEQ = 2 - Runtime Accumulation)

Runtime Accumulation selection requires the tool to allocate Nonvolatile memory and Set the Offsets value.

Example:

LL FILO: Consider that a Stage Driver function block is configured with 3 stages and with LL FILO settings for a boiler application.

1. If nStageActiveinput value=1, Stage 1 is turned on.
2. If nStageActiveinput value=2, Stage 1 and stage 2 are turned on.
3. If nStageActiveinput value becomes 3, or greater than 3 then all 3 states are turned on.

Assume that nStageActiveinput is 3 and it becomes 2 then stage 3 is turned OFF first and stage 1 and 2 remain ON. Stage 3 is turned OFF, because it is last stage. Stage 1 always comes on first and is always the last stage to turn off.

LL FIFO: If the stage driver is configured as LL_FIFO then stage driver operates the stages on a basis of First On First Off.

Number of Active stages = value of nStagesActive input

If stage driver with three stages and with LL_FIFO is configured and if,

1. nStageActive input becomes 1 then stage driver turns on first stage.
2. nStageActive input becomes 2 then stage driver turns on second stage.
3. nStageActive input becomes 1 then Stage 1 is turned OFF.
4. nStageActive input again becomes 2 then Stage 3 is turned ON.

nStageActive input becomes 1 then Stage 2 is turned OFF as it was the first stage.

RUNEQ: If the stage driver is configured with LL_RUNEQ, it accumulates the run time of every active stage. When staging down is required, it turns OFF the stage which has largest runtime.

Number of active stages=value of nStageActive input

If the stage driver is configured with three stages and LL_RUNEQ setting,

1. If nActiveStages input value is zero, all stages are turned OFF. If it is considered as an initial condition, accumulated active time for every stage is 0 minutes.

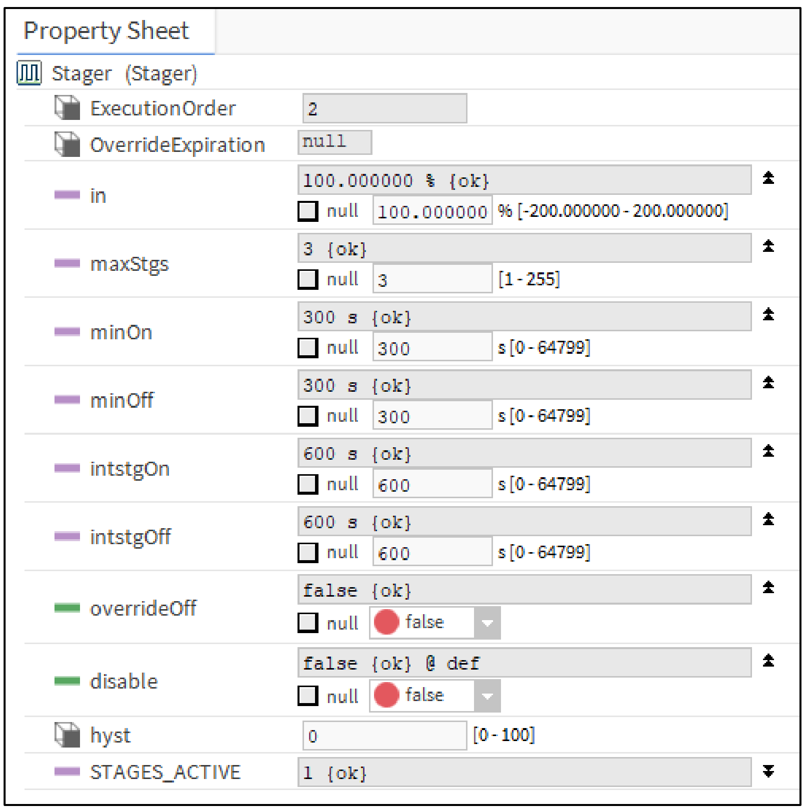
2. If nActiveStages value becomes one, first stage is turned ON. If the stager remains in this condition for two minutes, runtime of first stage is two minutes.
3. If nActiveStages value becomes 2 then second stage turns ON. If stager remains in this condition for 2 minutes, runtime of first stage is 4 minutes and runtime of the second stage is 2 minutes.
4. If nActiveStages value becomes 1 then first stage is turned OFF, and second stage remains ON as second stage has least time.
5. If nActiveStages value becomes 2 then third stage is turned ON as third stage has 0 runtime as compared to first stage.

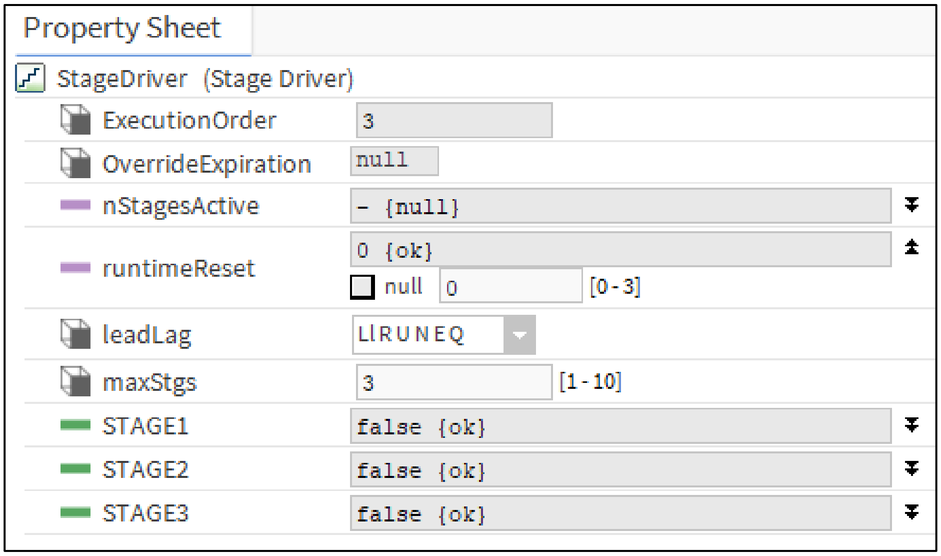
During staging up, the least runtime stage is turned ON and during staging down, the stage with highest run time is turned OFF. The aim is to equal runtime of every stage.

maxStgs (Byte param:UBYTE) specifies how many total stages nStagesActive can reach. maxStgs can go up to a total of 20 stages.

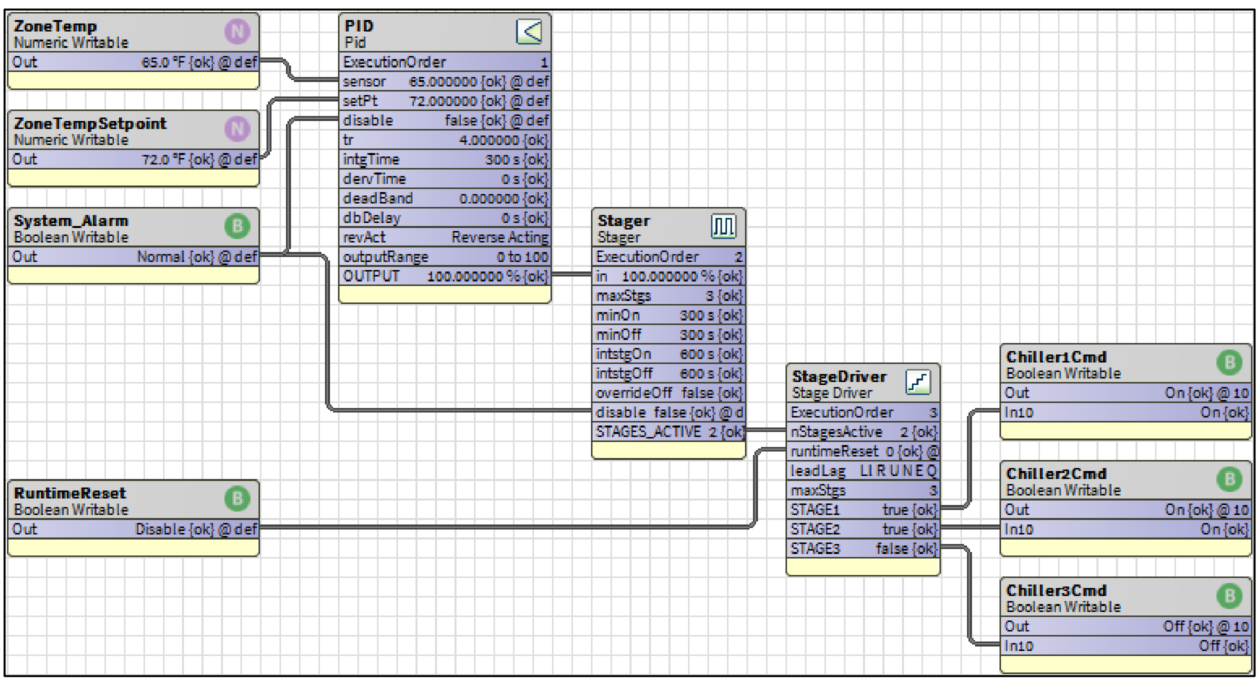
Example: Below is an example of using Stager and Stage Driver to enable 3 Chillers with Runtime equalization

The setting for Stager & Stage Driver are as below






If Stager & Stage Driver are directly connected, then the value of Max Stages need to match exactly.




**Note:**

Due to limitations of Niagara, only 95 stages can be seen on the wire sheet. To see, say stage number 200, do one of the following:

- Select the stages (in this case, stage 200) that needs to be seen by right-clicking them in the Block Configuration table under Show Stages and select Show.
- Invoke the link editor on the wire sheet. Select the Source and Target (in this case, stage 200).

# Logic Function Blocks

The CIPer Model 30 programming model provides the following logic function blocks that you can configure and use to build the required application logic:

- AND
- OneShot
- OR
- XOR

Inputs to the logic function blocks may come from either Digital or Floating-point variables.

For digital inputs:

- 0 = FALSE
- 1-255 = TRUE

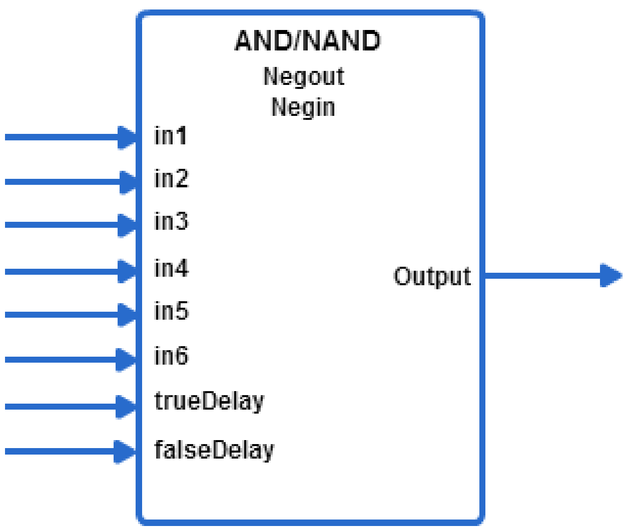
For floating point variables:

- = FALSE
- Any nonzero number = TRUE

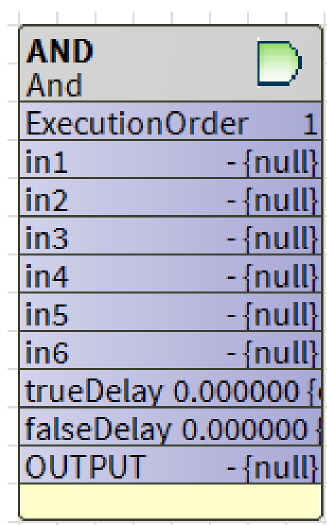
An output sent to a digital variable is 0 or 1. Similarly, an output sent to a float point variable is 0.0 or 1.0.

## AND

AND output becomes TRUE, if all inputs are TRUE. This function is a six-input AND Function Block. Each input may be individually inverted (NOT). The following table shows the basic operation of AND function block.




Logic Diagram




Function Block

Figure 316: AND Function

Table 94: Basic Operation of AND

| Input | | Output (X NAND Y = XY) |
|---|---|---|
| (X) | (Y) | |
| 0 (FALSE) | 0 (FALSE) | 0 (FALSE) |
| 0 (FALSE) | 1 (TRUE) | 0 (FALSE) |
| 1 (TRUE) | 0 (FALSE) | 0 (FALSE) |
| 1 (TRUE) | 1 (TRUE) | 1 (TRUE) |

The following table shows basic operation of NAND function block.

Table 95: Basic Operation of NAND

| Input | | Output (X NAND Y = $\overline{XY}$) |
|---|---|---|
| (X) | (Y) | |
| 0 (FALSE) | 0 (FALSE) | 1 (TRUE) |
| 0 (FALSE) | 1 (TRUE) | 1 (TRUE) |
| 1 (TRUE) | 0 (FALSE) | 1 (TRUE) |
| 1 (TRUE) | 1(TRUE) | 0 (FALSE) |

Unconnected or invalid inputs default to TRUE, without negation, to have no effect on the result.

In each iteration, the function block keeps track of the last computed output value and the current true or false delay time. These values are cleared on power up/reset.

## Logic Inputs

Table 96: Logic Inputs of AND Function

| Input Name | Input Value | Logic Value | Description |
|---|---|---|---|
| in1-6 | VAL ! = 0.0 | 1 | |
| | 0 | 0 | |
| | unconnected | 1 | Inputs with a "not" interpreted as logic 1 when disconnected. |
| | invalid | 1 | Negin does not affect the invalid logic value. |

## Analog Inputs

Table 97: Analog Inputs of AND Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| trueDelay | 0 | 3276 7 | uncon- nected | val = 0<br>It is the minimum time the computed output must stay True before the output changes from False to True. |
| (sec) | | | invalid | val = 0 |
| falseDelay | 0 | 3276 7 | uncon- nected | val = 0<br>It is the minimum time the computed output must stay False before the output changes from True to False. |
| (sec) | | | invalid | val = 0 |

## Output

Table 98: Outputs of AND Function

| Input Name | Low | Description |
|---|---|---|
| OUT- PUT | Any floating- point value | Output = AND/NAND (inputs).<br>Negating the Output makes AND function block operate like a NAND func- tion block. |

Example

1. Set In1- In6 = 1, and True delay = 2, and False delay = 6.

In this case, the output is set to 1 after a time delay of 2 seconds as specified by the True delay.

2. Set In1 = 0, In2 - In6 = 1, and True delay = 2, and False delay = 6.

In this case, the output is set to 0 after a time delay of 6 seconds as specified by the False delay.

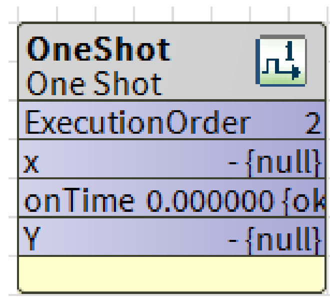
## OneShot

In the OneShot function block, when x transitions from False to True, y is set to True for onTime seconds.

The onTime is limited to the range 0 - 65535 sec. An onTime of zero keeps the output OFF irrespective of changes occur in the x input.

Both the x input and y outputs have an option to be negated. In each iteration, the function block keeps track of the last input and the onTime. On power, up/reset, this track is cleared.


Logic Diagram




Function Block

Figure 317: OneShot Function

### Logic Inputs

Table 99: Logic Inputs of OneShot Function

| Input Name | Input Value | Logic Value | Description |
|---|---|---|---|
| X | uncon-nected | N/A | For an invalid input make output be OFF (ON if output is negated). Clear the timer.<br>Must go from FALSE to TRUE (or TRUE to FALSE (Negated)) |
| | VAL != 0.0 | 1 | |
| | 0 | 0 | |
| | invalid | N/A | Same as unconnected. |

## Analog Inputs

Table 100: Analog Inputs of OneShot Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| onTime(sec) | 0 | 65535 | uncon-nected | onTime =0 |
| | | | invalid | onTime =0 |
| | | | < 0 | 0 |
| | | | >65535 | 65535 |

## Output

Table 101: Output of OneShot Function

| Input Name | Low | Description |
|---|---|---|
| Y | False (0) or True (1) | When x transitions from FALSE to TRUE, y is set to TRUE (1) for on-Time seconds. Negation on input or output reverses the logic as appropriate. |

Example

The Input is a square wave of 2 sec amplitude. The time transition diagram of the Output for different onTimes of 1 and 5 seconds is illustrated as shown in the following figure.

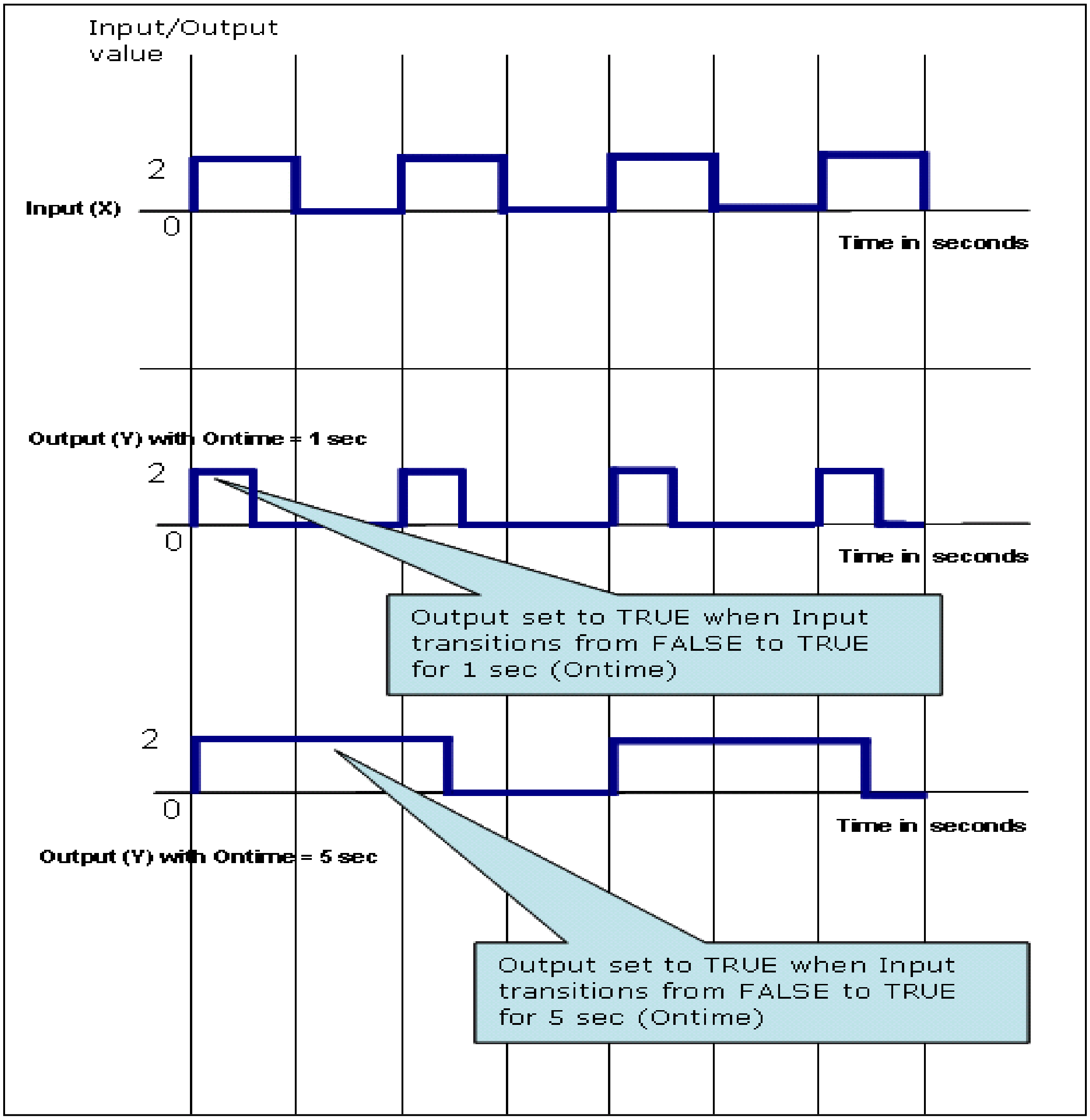



Figure 318: Time Transition Diagram of 1 sec and 5 Sec

# OR

The OR output becomes TRUE if at least one input is TRUE. This function is a six input OR. Each input may be individually inverted (NOT).

Following table shows basic operation of OR function block.

Table 102: Basic Operation of OR

| Input | | Output |
|---|---|---|
| (X) | (Y) | (X OR Y = X + Y) |
| 0 (FALSE) | 0 (FALSE) | 0 (FALSE) |
| 0 (FALSE) | 1 (TRUE) | 1 (TRUE) |
| 1 (TRUE) | 0 (FALSE) | 1 (TRUE) |
| 1 (TRUE) | 1 (TRUE) | 1 (TRUE) |

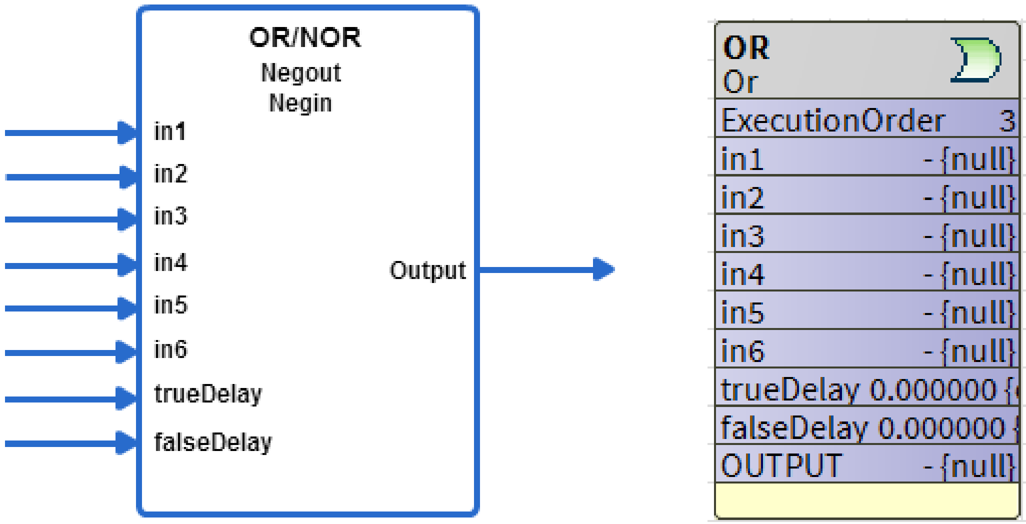
Following table shows basic operation of NOR function block.

Table 103: Basic Operation of NOR

| Input | | Output |
|---|---|---|
| (X) | (Y) | (X NOR Y = $\overline{\mathbf{X}+\mathbf{Y}}$ ) |
| 0 (FALSE) | 0 (FALSE) | 1 (TRUE) |
| 0 (FALSE) | 1 (TRUE) | 0 (FALSE) |
| 1 (TRUE) | 0 (FALSE) | 0 (FALSE) |
| 1 (TRUE) | 1(TRUE) | 0 (FALSE) |

Unconnected or invalid inputs default to True, without negation, to have no effect on the result.

In each iteration, the function block keeps track of the last computed output value and the current true or false delay time. These values are cleared on power up/reset.




Logic Diagram

Function Block

Figure 319: OR Function

## Logic Inputs

Table 104: Logic Inputs of OR Function

| Input Name | Input Value | Logic Value | Description |
| --- | --- | --- | --- |
| in1-6 | VAL != 0.0 | 1 | |
| | 0 | 0 | |
| | uncon-nected | 0 | Inputs with a "not" interpreted as logic 0 when discon-nected. |
| | 0 | 0 | Negin does not affect the invalid logic value |

## Analog Inputs

Table 105: Analog Inputs of OR Function

| Input Name | Range | | Input Value | Description |
| --- | --- | --- | --- | --- |
| | Low | High | | |
| trueDelay (sec) | 0 | 32767 | uncon-nected | val = 0 |
| | | | invalid | val = 0 |
| falseDelay (sec) | 0 | 32767 | uncon-nected | val = 0 |
| | | | invalid | val = 0 |

## Output

Table 106: Output of OR Function

| Input Name | Range | Description |
|---|---|---|
| OUTPUT | False (0) or True (1) | Output = OR/NOR (inputs) |

## XOR

The XOR output becomes TRUE if exactly one input is TRUE. This function is a six input XOR. Each input may be individually inverted (NOT).

Table 107: Basic Operation of XOR

| Input | | Output<br>(X OR Y = $\overline{\mathbf{XY}} + \overline{\mathbf{YX}}$ ) |
|---|---|---|
| (X) | (Y) | |
| 0 (FALSE) | 0 (FALSE) | 0 (FALSE) |
| 0 (FALSE) | 1 (TRUE) | 1 (TRUE) |
| 1 (TRUE) | 0 (FALSE) | 1 (TRUE) |
| 1 (TRUE) | 1 (TRUE) | 0 (FALSE) |

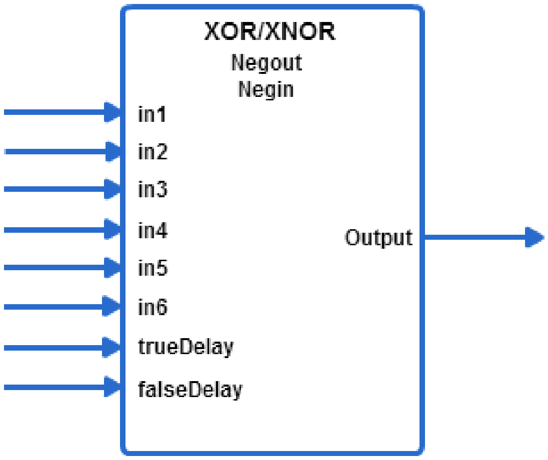
Following shows basic operation of XNOR function block.

Table 108: Basic Operation of XNOR

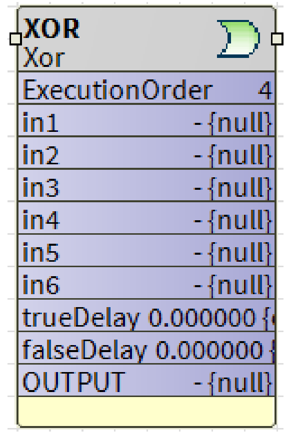
| Input | | Output<br>(X NOR Y = $\mathbf{XY} + \overline{\mathbf{XY}}$ ) |
|---|---|---|
| (X) | (Y) | |
| 0 (FALSE) | 0 (FALSE) | 1 (TRUE) |
| 0 (FALSE) | 1 (TRUE) | 0 (FALSE) |
| 1 (TRUE) | 0 (FALSE) | 0 (FALSE) |
| 1 (TRUE) | 1(TRUE) | 1 (TRUE) |

Unconnected or invalid inputs default to True, without negation, to have no effect on the result.

In each iteration, the function block keeps track of the last computed output value and the current true or false delay time. These values are cleared on power up/reset.




Logic Diagram




Function Block

Figure 320: XOR Function

## Logic Inputs

Table 109 Logic Inputs of XOR Function

| Input Name | Input Value | Logic Value | Description |
|---|---|---|---|
| in1-6 | VAL != 0.0 | 1 | |
| | 0 | 0 | |
| | uncon-nected | 0 | Inputs with a "not" interpreted as logic 0 when discon-nected. |
| | 0 | 0 | Negin does not affect the invalid logic value |

## Analog Inputs

Table 110: Analog Inputs of XOR Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| onTime (sec) | 0 | 32767 | uncon-nected | val = 0 |
| | | | invalid | val = 0 |
| | 0 | 32767 | uncon-nected | val = 0 |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| | | | invalid | val = 0 |

Output

Table 111: Output of XOR Function

| Input Name | Low | Description |
|---|---|---|
| OUTPUT | Any floating-point value | Output = XOR/XNOR (inputs) |

# Math Function Blocks

The CIPer Model 30 programming model provides the following Math function blocks that you can configure and use to build the required application logic:

- Add
- Digital Filter
- Divide
- Enthalpy
- Exponential
- Flow Velocity
- Limit
- Logarithm
- Multiply
- Ratio
- Reset
- Square Root
- Subtract

## Add

Math functions operate on and produce single precision floating-point numbers. In the absence of any other restrictions, if the result overflows the range of a single precision floating-point number (approximately minus 3.4e38 to plus 3.4e38), the result returned is invalid.

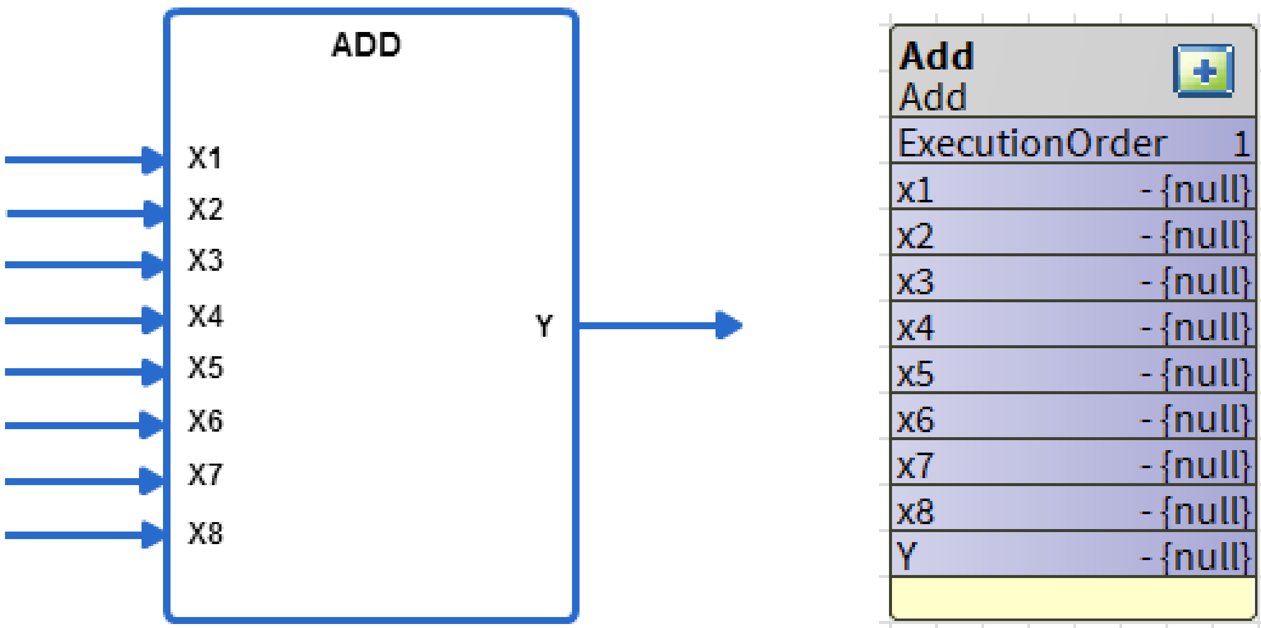
| Note: |
|---|
| You can connect both Analog and Digital inputs to this function block. |

TailOperation: The output value is based on one of four specified property values:

- No Change: The actual result is returned.
- Absolute: The absolute (modulus or non-negative) value of the result is returned. For example, if the output is -3, the result is 3.
- Integer: The integer value of the result is returned. For example, if the output is 3.25, the result is 3.
- Fractional: The fractional value of the result is returned. For example, if the output is 3.25, the result is .25.

Ignore invalid inputs:

If this option is selected, function block considers only valid inputs while determining the addition of the inputs. If this option is not selected, and any input becomes invalid then the output becomes invalid too.




Logic Diagram Function Block

Figure 321: Add Function

## Inputs

Table 112: Inputs of Add Function

| Input Name | Range | | Input Value | Ignore Invalid Input | Description |
|---|---|---|---|---|---|
| | Low | High | | | |
| x1-x8 | >= – infinity | <+ infinity | unconnected | | Not used in the calculation. If all inputs are unconnected, output is zero. |
| x1-x8 | >= – infinity | <+ infinity | invalid | false | If any input is invalid, output is invalid. |
| x1-x8 | >= – infinity | <+ infinity | invalid | true | Output considers only valid inputs while determining the addition of the inputs. |
| x1-x8 | >= – infinity | <+ infinity | valid | | Calculates the addition of 8 inputs or those set |

| Input Name | Range | | Input Value | Ignore Invalid Input | Description |
|---|---|---|---|---|---|
| | Low | High | | | |
| | | | | | as constant. |

Output

Table 113: Output of Add Function

| Input Name | Low | Description |
|---|---|---|
| Y | Any floating-point value | Output is the sum of inputs x1 through x8. |

## Digital Filter

This function digitally filters the input.

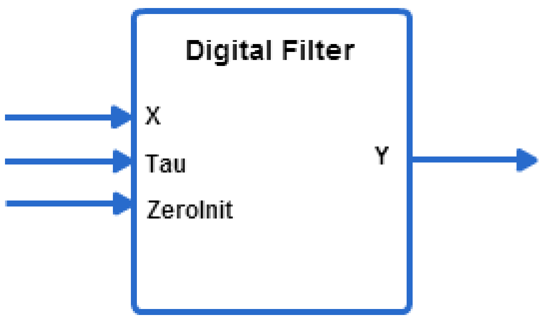
$Y_{new} = Y_{old} + (X - Y_{old}) * (1 - exp^{(-t/Tau)})$.

$$Y_{new}=Y_{old}-\left(X-Y_{old}\right)-e^{\frac{-t}{\tau}}$$

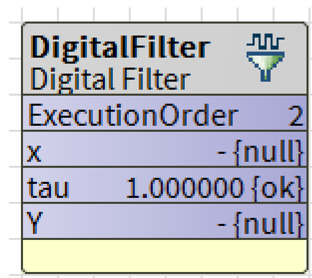
Where, t = 1 sec and Tau is in the range 0 - 65535 sec.

The output can be initialized to zero (zeroInit=TRUE) or the first valid input value (zeroInit=FALSE).

From iteration to iteration, the Function Block keeps track of the tau multiplier (1 – exp $^{(-t/Tau)}$). On power, up/re-set, this is recalculated




Logic Diagram




Function Block

Figure 322: Digital Filter Function

### Inputs

Table 114: Inputs of Digital Filter Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| X | >= – infinity | <+ infinity | Uncon-nected | The output is invalid. |
| | | | Invalid | Output is set to invalid and filter rei-nitializes when the input returns to valid. |

### Output

Table 115: Output of Digital Filter Function

| Input Name | Low | Description |
| --- | --- | --- |
| Y | Any floating-point value | $Y_{new} = Y_{old} + (X - Y_{old}) * (1 - \exp^{(-t/Tau)})$. |

Setpoint

Table 116: Setpoint of Digital Filter Function

| Input Name | Low | Description |
| --- | --- | --- |
| Y | Any floating-point value | $Y_{new} = Y_{old} + (X - Y_{old}) * (1 - \exp^{(-t/Tau)})$. |

| Note: |
| --- |
| Both Analog and Digital inputs can be connected as inputs to this function block. |

Example 1:

Set In1 (X) = 4, tau = 2.0, Set ZeroInit = 1 (initializes filter to 0.0)

$Y_{new} = Y_{old} + (X - Y_{old}) * (1 - \exp^{(-t / Tau)})$

In the first iteration,

$Y_{old} = 0; Y_{new} = Y_{old} + (X - Y_{old}) *$
$(1 - \exp(-t/Tau))$

$Y_{new} = 0 + (4 - 0) * (1 - 2.718 (-1 / 2))$
$= 0 + 4 * (0.393)$
$= 1.572$

In the second iteration,

$Y_{old} = 1.572; X = 4; Y_{new} = 1.57 +$
$(4 - 1.57) * (0.393)$

$Y_{new} = 2.52$

In the third iteration,

$Y_{new} = 2.52 + (4 - 2.52) * (0.393)$
$= 3.107$

The iterations continue until the input is reached.

Example 2:

Set In1 (X) = 4, tau = 2.0, Set ZeroInit = 0 (initializes filter to first valid value)

$Y_{new} = Y_{old} + (X - Y_{old}) * (1 - \exp^{(-t / Tau)})$

In the first iteration,

$Y_{new} = X$

$= 4$

In the second iteration, if X = 6

$Y_{new} = Y_{old} + (X - Y_{old}) * (1 - \exp^{(-t / Tau)})$

$= 4 + (6 - 4) * (0.393)$

$= 4 + 0.786$

$= 4.786$

In the third iteration, if X = 6

$Y_{new} = Y_{old} + (X - Y_{old}) * (1 - \exp^{(-t / Tau)})$

$= 4.786 + (6 - 4.786) * (0.393)$

$= 5.263.$

The iterations continue until the input is reached.

## Divide

This function divides one input by the other.

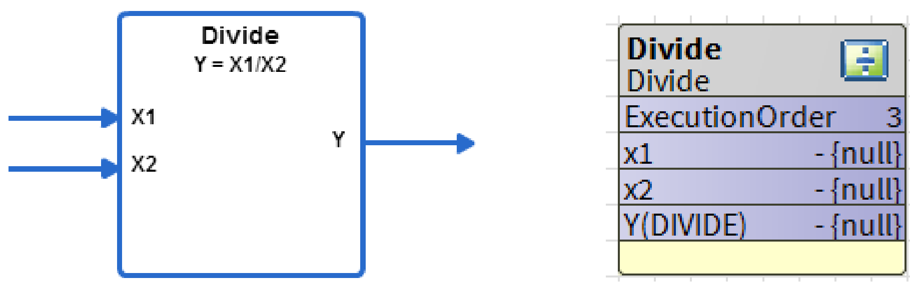
Y = x1 / x2. Division by 0 results in an invalid output. If the result overflows the range of a single precision floating point number (approximately minus $3.4e^{38}$ to plus $3.4e^{38}$) the result returned is invalid.

divOperation: The result of the division is based on one of two specified property values. For Modulo, the output is the remainder of the division; for Divide, the output is the quotient.

TailOperation: The output value is based on one of four specified property values:

- No Change: The actual result is returned.
- Absolute: The absolute (modulus or non-negative) value of the result is returned. For example, if the output is -3, the result is 3.
- Integer: The integer value of the result is returned. For example, if the output is 3.25, the result is 3.
- Fractional: The fractional value of the result is returned. For example, if the output is 3.25, the result is .25.

| Note: |
|---|
| Both Analog and Digital inputs can be connected as inputs to this function block. |




Logic Diagram

Function Block

Figure 323: Divide Function

### Analog Inputs

Table 117: Analog Inputs of Divide Function

| Input Name | Range | | Input Value | Ignore Invalid Input | Description |
|---|---|---|---|---|---|
| | Low | High | | | |
| x1, x2 | >= – infinity | <+ infinity | unconnected | | Not used in the calculation. If all inputs are unconnected, output is set to invalid. |
| | | | invalid | false | If any input is invalid, output |

| Input Name | Range | | Input Value | Ignore Invalid Input | Description |
|---|---|---|---|---|---|
| | Low | High | | | |
| | | | | | is invalid. |
| | | | invalid | true | Output considers only valid inputs while determining the division of the inputs. |
| | | | valid | | Calculates the division of two inputs or those set as constant. |

Output

Table 118: Output of Divide Function

| Output Name | Range | Description |
|---|---|---|
| Y out | Any floating-point value | Y= x1/x2 |

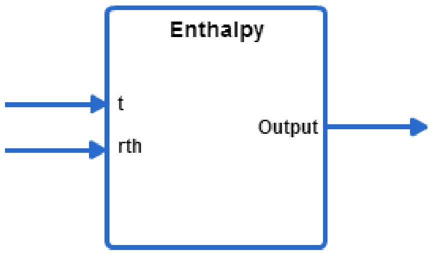
Configuration

Table 119: Configuration of Divide Function

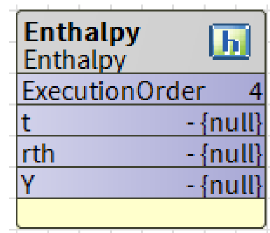
| Name | Range | Description |
|---|---|---|
| invalidFlag | 0 to 1 | See above table. |
| divOperation | 0 to 1 | Divide operation to be performed:<br>0 = division/quotient. Result is x1/x2.<br>1 = modulo/remainder. The result is x1 modulo x2, that is,the remainder f, where<br>x1 = a*x2 + f for some integer a and 0 <= f < x2. |
| tailOperation | 0 to 3 | Operation to be applied to the result just prior to output:<br>0 = no change to result<br>1 = absolute value (result is made positive if negative)<br>2 = integer part. The fractional part is truncated to 0. The sign is retained<br>3 = fractional part. The integer part is truncated to 0. The result is always positive. |

## Enthalpy

This function computes the enthalpy (BTU/LB) based on the temperature (°F) and relative humidity (%) inputs. Relative humidity (rh) is limited to 0% to 100%. Temperature is limited to 0°F - 120°F.




Logic Diagram




Function Block

Figure 324: Enthalpy Function

### Analog Inputs

Table 120: Analog Inputs of Enthalpy Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| t | 0°F | 120°F | uncon-nected | output = invalid |
| (F) | | | invalid | output = invalid |
| | | | VAL < low | T = low |
| | | | VAL > high | T = high |
| rth (%) | 0 | 100 | uncon-nected | output = invalid |
| | | | invalid | output = invalid |
| | | | VAL < low | RH = low |
| | | | VAL > high | RH = high |

### Output

Table 121: Output of Enthalpy Function

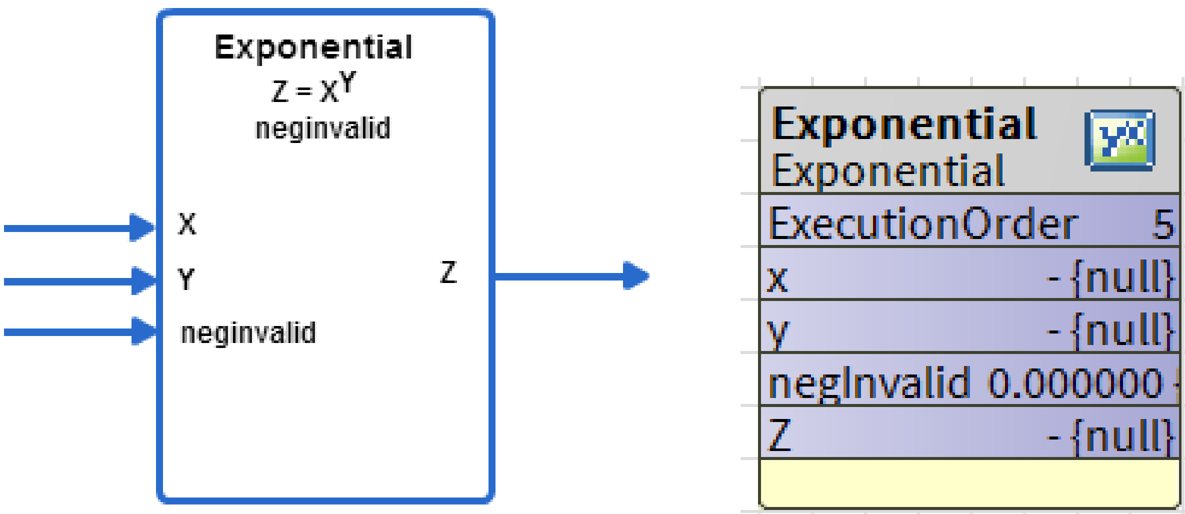
| Input Name | Low | Description |
|---|---|---|
| Y | Any floating-point value | Output = Enthalpy (t, rth) |

## Exponential

This function raises y to the power of x. x and y are floating point numbers. The application designer is limited to two function blocks (these types) per device. Unconnected inputs are treated as 0. Invalid inputs result in an invalid output. The negInvalid input determines whether the operation should proceed with a negative base and non-integer exponent, operating on the absolute value of the base, or return invalid. The negInvalid input does not affect an unconnected or invalid input. If both the x and y inputs are disconnected, the output z is 1.

TailOperation: The output value is based on one of four specified property values:

- No Change: The actual result is returned.
- Absolute: The absolute (modulus or non-negative) value of the result is returned. For example, if the output is -3, the result is 3.
- Integer: The integer value of the result is returned. For example, if the output is 3.25, the result is 3.
- Fractional: The fractional value of the result is returned. For example, if the output is 3.25, the result is .25.




Logic Diagram　　Function Block

Figure 325: Exponential Function

### Analog Inputs

Table 122: Analog Inputs of Exponential Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| onTime (sec) | 0 | 65535 | unconnected | onTime = 0 |
| | | | invalid | onTime = 0 |
| | | | < 0 | 0 |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| | | | >65535 | 65535 |

## Output

Table 123: Output of Exponential Function

| Input Name | Low | Description |
|---|---|---|
| Y | Any floating-point value | When x transitions from FALSE to TRUE, y is set to TRUE (1) for on-Time seconds. |

## FlowVelocity

This function computes the flow and velocity based on the measured pressure and the K factor.

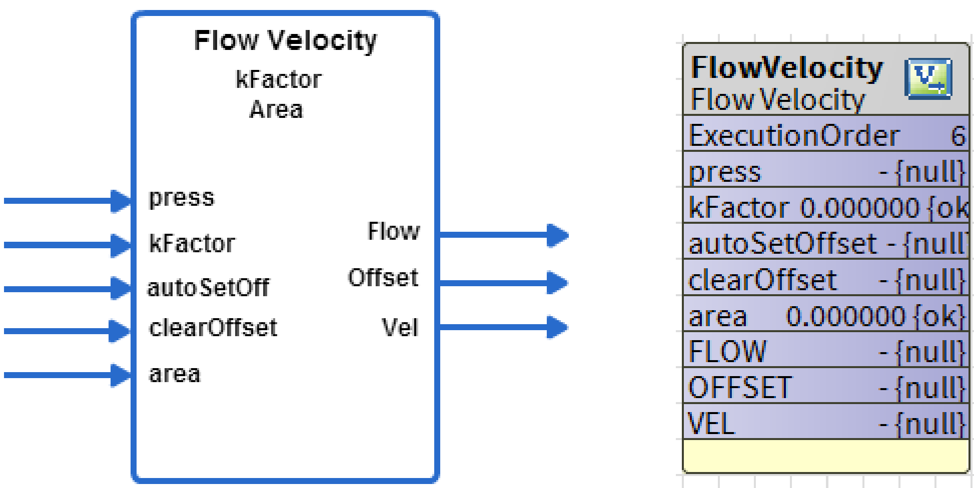
$$flow = K\sqrt{\Delta P - offset}$$

And

Vel = flow/area

Where:

- K = Flow coefficient (K-Factor) representing the actual flow in ft^3/min corresponding to a velocity pressure sensor output of 1 w.g.
- DeltaP = Flow sensor output pressure in inches
- Water gauge (inw)
- OFFSET = Correction pressure (inw) to adjust for zero
- FLOW = Airflow in $ft^3$/min (CFM)
- VEL = Flow velocity in ft/min
- area = Duct area in $ft^2$




Logic Diagram    Function Block

Figure 326: Flow Velocity Function

### Analog Inputs

Table 124: Analog Inputs of Flow Velocity Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| press | >= – infinity | < + infinity | unconnected | Output is set to invalid |
| | | | invalid | Output is set to invalid |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| | | | > –0.002425 and < 0.002425 inw | Flow and vel = 0 |
| autoSet-Offset | >= – in-finity | < + in-finity | Unconnected | No effect on output |
| | | | Invalid | No effect on output |
| clearOffset | >= – in-finity | < + in-finity | ! = 0 | Set offset = incoming press |
| | | | unconnected or invalid | No effect on output |
| | | | ! = 0 | Set offset = 0 |
| area | >= – in-finity | < + in-finity | Invalid or < = 0; value in $ft^2$ | Velocity is set to invalid |
| kFactor | >= – in-finity | < + in-finity | unconnected | Output is set to invalid |
| | | | invalid | Output is set to invalid |
| | | | < = 0 | kFactor = 1015 |

## Output

Table 125: Output of Flow Velocity Function

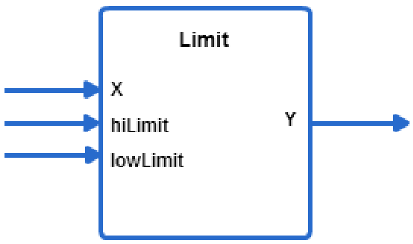
| Input Name | Range | | Description |
|---|---|---|---|
| | Low | High | |
| FLOW | > = – in-finity | <+ in-finity | Flow value ($ft^3$/min) |
| OFFSET | > = – in-finity | <+ in-finity | Input press, offset correction (inches water column). Not used for hardware connection. Stores Flow offset amount. |
| VEL | > = – in-finity | <+ in-finity | Flow velocity (ft/min) |

## Limit

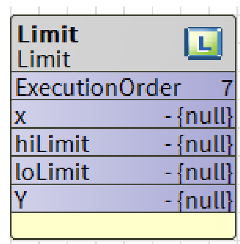
This function limits the input to the low and high limits.

If the value of input (x) is:

- Lower than the lowLimit, value of output is set to lowLimit
- Higher than the hiLimit, output is set to hiLimit
- Between the lowLimit and hiLimit, output is set to input




Logic Diagram




Function Block

Figure 327: Limit Function

### Analog Inputs

Table 126: Analog Inputs of Limit Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| x | >= – infinity | <+ infinity | unconnected | Output is set to invalid |
| | | | invalid | Output is set to invalid |
| | | | x< lowLimit | Output is set to lowLimit |
| | | | lowLimit > hiLimit | Limits not enforced (not enforced means Y is always set to X.) |
| | | | lowLimit < x < hiLimit | Output set to x x>hiLimit Output set to hiLimit |
| hiLimit | >= – infinity | <+ infinity | unconnected | hiLimit not enforced |
| | | | invalid | hiLimit not enforced |
| lowLimit | >= –infinity | <+ infinity | unconnected | lowLimit not enforced |
| | | | invalid | lowLimit not enforced |

## Output

Table 127: Output of Limit Function

| Input Name | Range | Description |
|---|---|---|
| Y | Any floating-point value | Y = Limit (x, lowLimit, hiLimit) |

## Multiply

This function multiplies one input with the other.

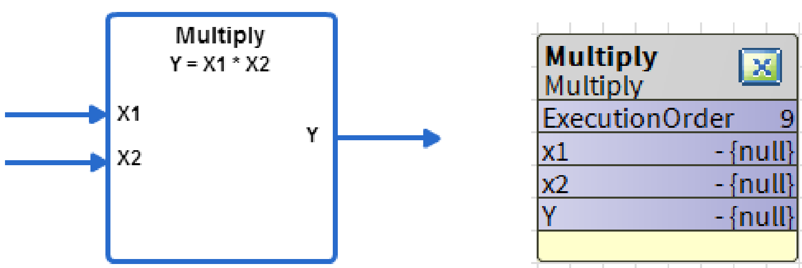
y = x1 multiplied by x2. If the result overflows the range of a single precision floating point number (approximately minus $3.4e^{38}$ to plus $3.4e^{38}$), the result returned is invalid.

| Note: |
|---|
| Both Analog and Digital inputs can be connected as inputs to this function block. |

TailOperation: The output value is based on one of four specified property values:

- No Change: The actual result is returned.
- Absolute: The absolute (modulus or non-negative) value of the result is returned. Example: If the output is -3, the result is 3.
- Integer: The integer value of the result is returned. For example, if the output is 3.25, the result is 3.
- Fractional: The fractional value of the result is returned. For example, if the output is 3.25, the result is .25.

Ignore invalid inputs: If this option is selected, function block considers only valid inputs while determining the multiplication of the inputs. If this option is not selected and any input becomes invalid, output also becomes invalid.




Logic Diagram    Function Block

Figure 328: Multiply Function

## Analog Inputs

Table 128: Analog Inputs of Multiply Function

| Input Name | Range | | Input Value | Ignore Invalid Input | Description |
|---|---|---|---|---|---|
| | Low | High | | | |
| x1, x2 | >= – infinity | <+ infinity | unconnected | | Unconnected inputs are set to zero. If all inputs are unconnected, output is set to zero. |
| | | | invalid | false | If any input is invalid, output is invalid. |
| | | | invalid | true | Output considers only valid inputs while determining the multiplication of the inputs. |
| | | | valid | | Calculates the multiplication of two inputs or those set as constant. |

## Output

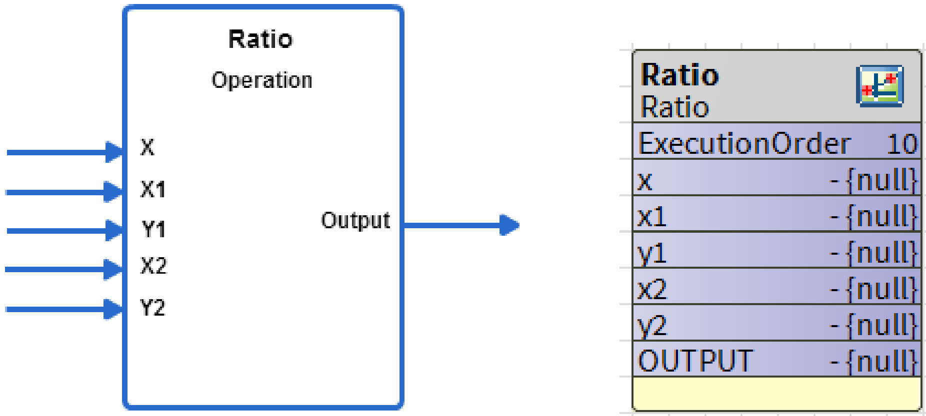
Table 129: Output of Multiply Function

| Input Name | Range | Description |
|---|---|---|
| Y | Any floating-point value | Y= x1 * x2 |

## Ratio

This function converts the input X to the output Y based on the line defined by x1, y1, x2, and y2.

Output (Y) = y1 + (((x – x1) * (y2 – y1)) / (x2 – x1))




Logic Diagram Function Block

Figure 329: Ratio Function

### Analog Inputs

Table 130: Analog Inputs of Ratio Function

| Input Name | Range | | Input Value | Ignore Invalid Input | Description |
|---|---|---|---|---|---|
| | Low | High | | | |
| x | >= – infinity | <+ infinity | unconnected | true | Output set to invalid. |
| | | | | false | Output set to invalid. |
| | | | invalid | true | Output set to y1. |
| | | | | false | Output set to invalid. |
| x1-x2 | >= – infinity | <+ infinity | unconnected | true | Output set to invalid. |
| | | | | false | Output set to invalid. |
| | | | invalid | true | Output set to y1. |
| | | | | false | Output set to invalid. |
| y1 | >= – infinity | | | true | Output set to invalid. |

| Input Name | Range | | Input Value | Ignore Invalid Input | Description |
|---|---|---|---|---|---|
| | Low | High | | | |
| | | <+ infinity | unconnected | false | Output set to invalid. |
| | | | invalid | true | Output set to invalid. |
| | | | | false | Output set to invalid. |
| Y2 | >= – infinity | <+ infinity | unconnected | true | Output set to invalid. |
| | | | | false | Output set to invalid. |
| | | | invalid | true | Output set to y1. |
| | | | | false | Output set to invalid. |

## Output

Table 131: Output of Ratio Function

| Input Name | Range | Description |
|---|---|---|
| OUTPUT | Any floating-point value | Out Ratio (X, X1, Y1, X2, Y2) |

## Setpoints

Table 132: Setpoints of Ratio Function

| Input Name | Range/Value | Description |
|---|---|---|
| operation | Unlimited<br>Vav_Flow_Balance<br>Endpoint_Limited | |

## Unlimited

The output is based on the line defined by x1, x2, y1, y2. The behavior of the function block is as illustrated in the following figure.

Y = y1 + (((x – x1) * (y2 – y1)) / (x2 – x1))

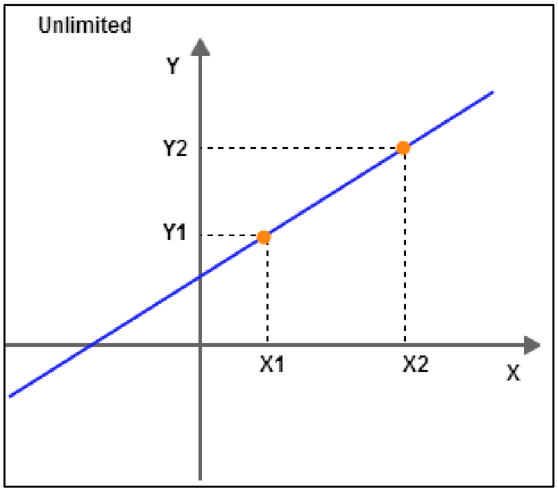



Figure 330: Unlimited

## VAV Flow Balance

The output is based on the line defined by x1, x2, y1, y2. The slope of the line is as shown in the following illustration.

- When x1 = 0,

  OUTPUT = 0
- When x >= x1,

  OUTPUT = y1 + (((x - x1) * (y2 - y1)) / (x2 - x1))
- When x < x1,

  OUTPUT = ((x * y2) / x2)

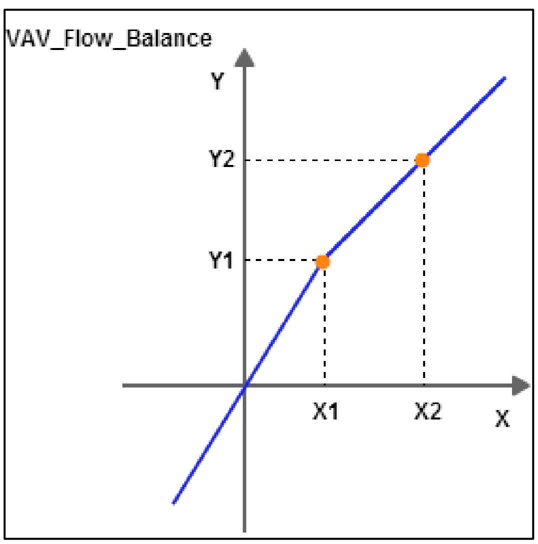



Figure 331: VAV Flow Balance

## Endpoint Limited

The output is based on the line defined by x1, x2, y1, y2. The slope of the line is as shown in the following illustration. Beyond points x1 and x2, the output is limited to the point y1 and y2 respectively. The output is held between the point y1 and y2.

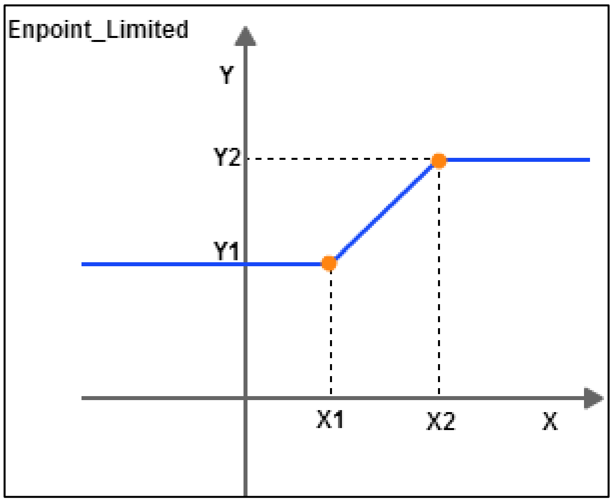
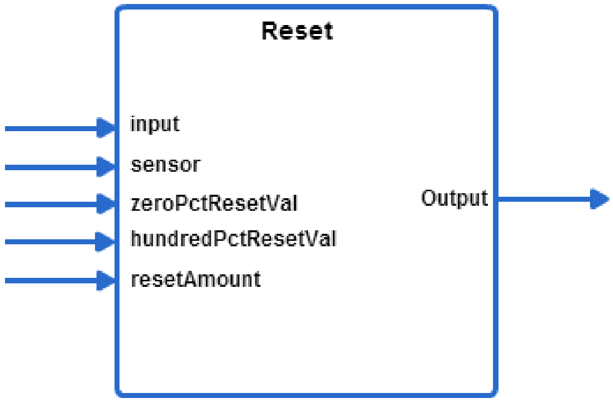



Figure 332: Endpoint Limited

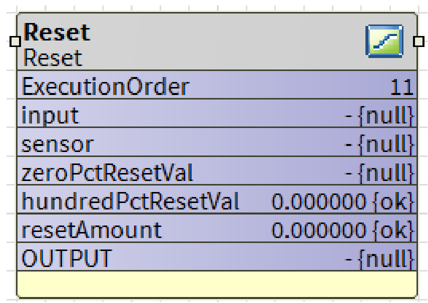
- When x1 < x2 and x <= x1 OUTPUT = Y1
- When x1 < x2 and x >= x2 OUTPUT = Y
- When x1 > x2 and x <= x2 OUTPUT = Y2
- When x1 > x2 and x >= x1 OUTPUT = Y1

## Reset

This function computes the reset value based on the relation of the input to the reset parameters.




Logic Diagram




Function Block

Figure 333: Reset Function

### Analog Inputs

Table 133: Analog Inputs of Reset Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| input | >= – infinity | <+ in-finity | unconnected | Output is set to invalid |
| | | | invalid | Output is set to invalid |
| sensor | >= – infinity | <+ in-finity | unconnected | Output is set to invalid |
| | | | invalid | Output = input |
| zeroPctResetVal | >= – infinity | <+ in-finity | unconnected | Output is set to invalid |
| | | | invalid | Output = input |
| | | | 0%RV = 100%RV | Output is set to invalid |
| hundredPctReset-Val | >= – infinity | <+ in-finity | unconnected | Output is set to invalid |
| | | | invalid | Output = input |
| | | | 0%RV = 100%RV | Output is set to input |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| resetAmount | >= – infinity | <+ in-finity | unconnected | Output is set to invalid |
| | | | invalid | Output = input |

## Output

Table 134: Output of Reset Function

| Input Name | Range | Description |
|---|---|---|
| OUTPUT | Any floating-point value | Y = Reset (input, sensor, 0%, 100%, reset amount) |

## Working

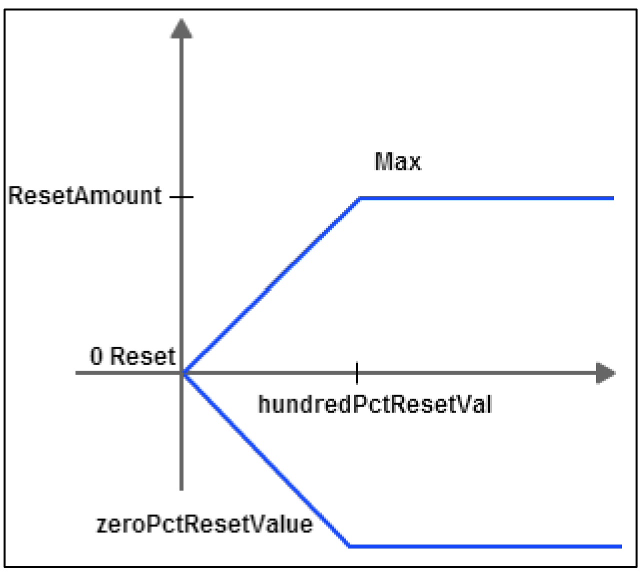



Figure 334: Working of Reset

Table 135: Input and Output of Reset Function

| If Input Condition is | Output |
|---|---|
| • Input is unconnected<br>• Input is invalid<br>• Sensor is unconnected<br>• zeroPctResetVal is unconnected<br>• hundredPctResetVal is unconnected<br>• resetAmount is unconnected | Output = invalid |

| If Input Condition is | Output |
|---|---|
| • Sensor is invalid<br>• Sensor < zeroPctResetVal<br>• zeroPctResetVal is invalid<br>• hundredPctResetVal is invalid<br>• resetAmount is invalid<br>• hundredPctResetVal = zeroPctResetVal | Output = input |
| Sensor > hundredPctResetVal | Output = input + resetAmount |
| If none of the above conditions are satisfied | Output = input + ((sensor – zeroPctResetVal) / hundredPctResetVal – zeroPctResetVal)) * re-setAmount |

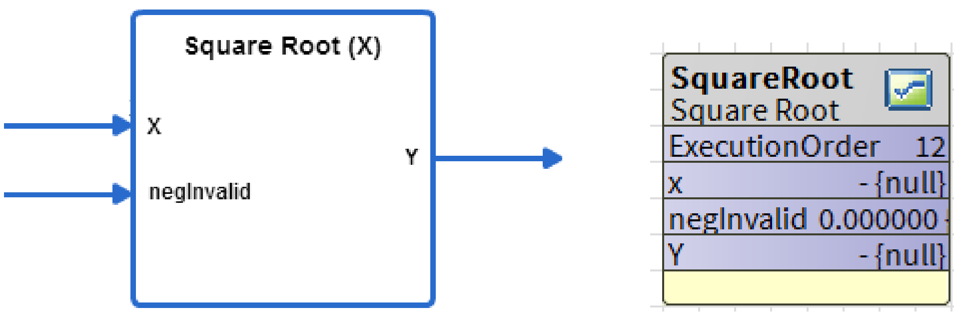
## Square Root

This function takes the square root of the input. The Output Y is the Sqrt (x), where x is the input. The behavior of a negative x input is controlled by the parameter negInvalid.

| Note: |
|---|
| Negative values are treated as absolute values. Example: Square root of -9801 is given as 99, taking the absolute value of –9801 as 9801. |

TailOperation: The output value is based on one of four specified property values:

- No Change: The actual result is returned.
- Absolute: The absolute (modulus or non-negative) value of the result is returned. For example, if the output is -3, the result is 3.
- Integer: The integer value of the result is returned. For example, if the output is 3.25, the result is 3.
- Fractional: The fractional value of the result is returned. For example, if the output is 3.25, the result is .25.




Logic Diagram

Function Block

Figure 335: Square Root Function

### Analog Inputs

Table 136: Analog Inputs of Square Root Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| X | >=– infinity | <+ infinity | uncon- | Y= 0 |
| | | | invalid | Output is set to invalid |
| | | | x1 < 0 | See the description for negInvalid input. |
| negInvalid | 0 | 1 | 0 | Use the square root of the absolute value. |
| | | | 1 | If the input is negative, the output is invalid. The default value is 0. |
| | | | unconnected | Y = sqrt(X), output is invalid for neg x1 |
| | | | invalid | Y = sqrt(X), output is invalid for neg x1 |

## Output

Table 137: Output of Square Root Function

| Input Name | Range | Description |
| --- | --- | --- |
| Y | Any floating-point value | Y= Sqrt (X) |

## Subtract

This function subtracts one input from the other. Y = x1 – x2. If the result overflows the range of a single precision floating point number, (approximately minus $3.4e^{38}$ to plus $3.4e^{38}$) the result returned is invalid.

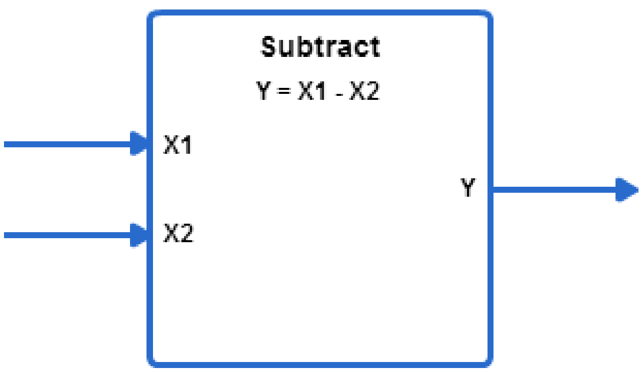
| Note: |
|---|
| Both analog and digital inputs can be connected as inputs to this function block. |

Ignore invalid inputs:

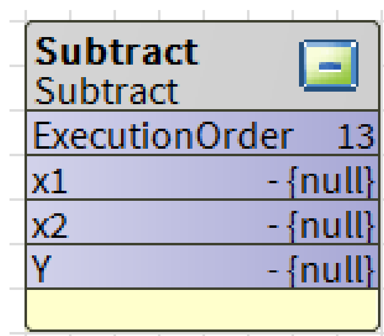
If this option is selected, the function block considers only valid inputs while determining the subtraction of the inputs. If this option is not selected and any input become invalid, output also becomes invalid

TailOperation: The output value is based on one of four specified property values:

- No Change: The actual result is returned.
- Absolute: The absolute (modulus or non-negative) value of the result is returned. For example: if the output is -3, the result is 3.
- Integer: The integer value of the result is returned. For example, if the output is 3.25, the result is 3.
- Fractional: The fractional value of the result is returned. For example, if the output is 3.25, the result is .25.




Logic Diagram




Function Block

### Analog Inputs

Table 138: Analog Inputs of Subtract Function

| Input Name | Range Low | Range High | Input Value | Ignore Invalid Input | Description |
|---|---|---|---|---|---|
| x1, x2 | >= – infinity | <+ infinity | unconnected | | Unconnected inputs are set to zero. If all inputs are unconnected, output is set to zero. |
| | | | invalid | false | If any input is invalid, output is invalid. |

| Input Name | Range | | Input Value | Ignore Invalid Input | Description |
|---|---|---|---|---|---|
| | Low | High | | | |
| | | | invalid | true | Output considers only valid inputs while determining the subtraction of the inputs. |
| | | | valid | | Calculates the subtraction of two inputs or those set as constant. |

### Output

Table 139: Output of Subtract Function

| Input Name | Range | Description |
|---|---|---|
| Y | Any floating-point value | Y= x1 - x2 |

## Logarithm

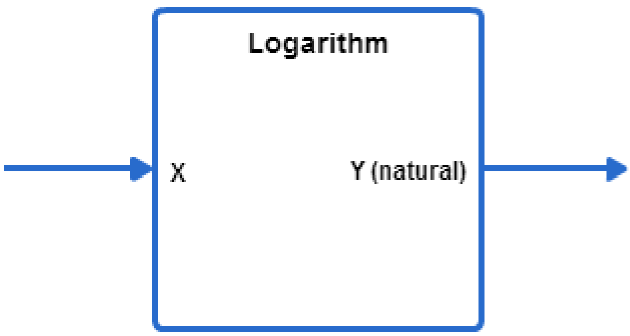
This function takes either the natural logarithm

($\log_e(x) = \ln(x)$) or logarithm base 10 ($\log_{10}(x)$) of the input, depending on the configuration setting eOR10.

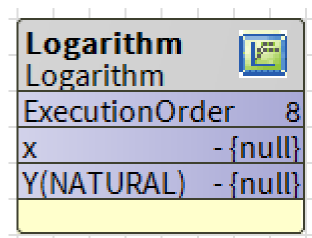
$Y = \log_e(X)$ or $Y = \log_{10}(X)$.

eOR10: The Log operation depends on the selection of this configuration property. In this property, you can select a natural or base10 log.

- Natural: Natural logarithm ($\log_e(x) = \ln(x)$) of the input
- Base10: logarithm base 10 ($\log_{10}(x)$) of the input
- TailOperation: The output value is based on one of four specified property values:
- No Change: The actual result is returned.
- Absolute: The absolute (modulus or non-negative) value of the result is returned. For example, if the output is –3, the result is 3.
- Integer: The integer value of the result is returned. For example, if the output is 3.25, the result is 3.
- Fractional: The fractional value of the result is returned. For example, if the output is 3.25, the result is .25.




Logic Diagram




Function Block

Figure 336: Logarithm Function

### Analog Inputs

Table 140: Analog Inputs of Subtract Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| x | > 0 | + infinity | unconnected | Output is set to invalid |
| | | | invalid | Output is set to invalid |
| | | | x<=0 | Output is set to invalid |

### Output

Table 141: Output of Subtract Function

| Input Name | Range | Description |
|---|---|---|
| | Any floating-point value | Y= Log (x) |

## Data Function Blocks

The CIPer Model 30 programming model provides the following Data Function blocks that can be configured and use to build the required application logic:

- Counter
- Override
- RunTimeAccumulate

### Counter

This function counts leading edge transitions of the input. If enable is True and the input transitions from False to True, the count is incremented or decremented by the count value. Positive values on count value increment the count. Negative values decrement the count. If the preset is True, the count is set to the Preset Value. From iteration to iteration, the Function Block keeps track of the previous state of the input so that it can detect a transition. On power up/reset, this is cleared.

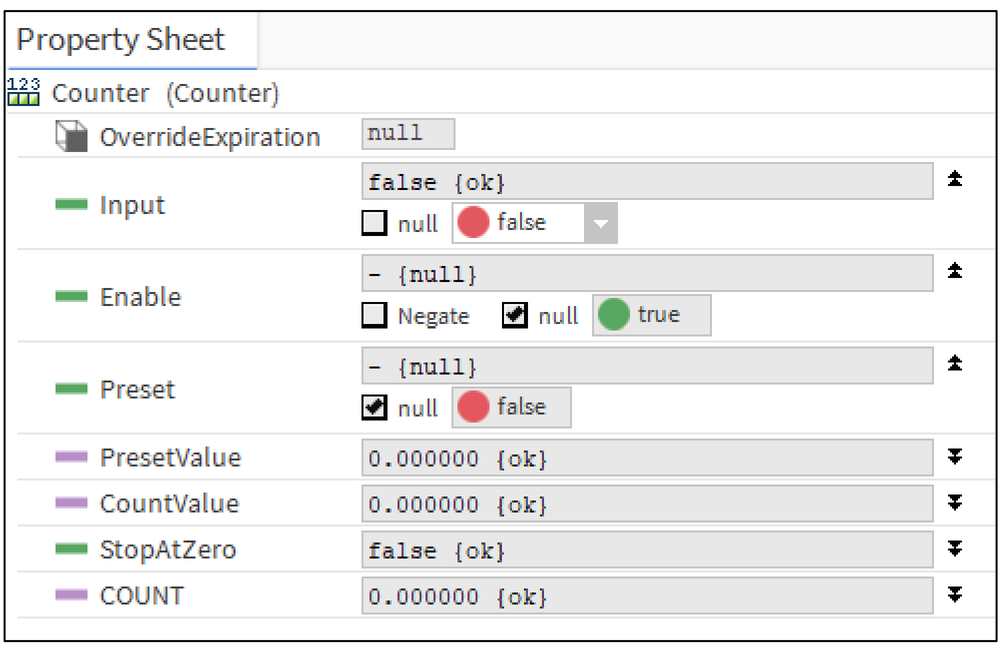
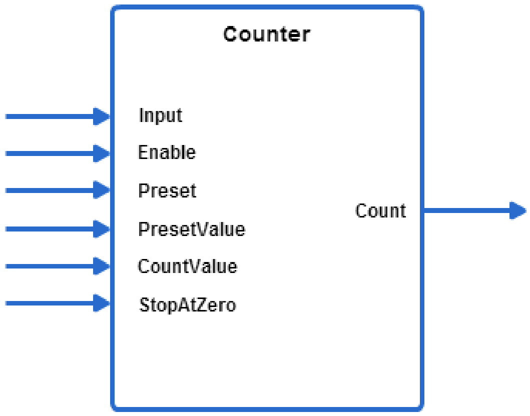
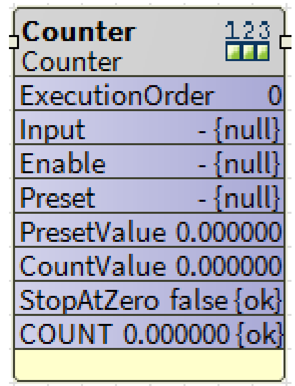



Figure 337: Counter Property Sheet

Unconfigured is taken as high priority compared to Negate. If nothing is configured and Negate is applied nothing will get effected. Unconfigured will be taken as high priority.




Logic Diagram




Function Block

Figure 338: Counter Function

## Logic Inputs

Table 142: Logic Inputs of Counter Function

| Input | Input Value | Logic | Description |
|---|---|---|---|
| Input | uncon- | 0 | Set Input = False |
| | invalid | 0 | Set Input = False |
| | 0 | 0 | Input is False |
| | VAL != 0.0 | 1 | Input is True |
| Enable | uncon- | 1 | Set Enable = True |
| | invalid | 1 | Set Enable = True |
| | 0 | 0 | Set Enable = False |
| | VAL != 0.0 | 1 | Set Enable = True |
| Preset | uncon- | 0 | Set Preset = False |
| | invalid | 0 | Set Preset = False |
| | 0 | 0 | Set Preset = False |
| | VAL != 0.0 | 1 | Set Preset = False |
| StopAtZero | uncon- | 0 | Set Stop At Zero = False. The default value is False. |
| | invalid | 0 | Set Stop At Zero = False. |

| Input | Input Value | Logic | Description |
|---|---|---|---|
| | 0 | 0 | Stop At Zero is False. The count is unaffected by a zero |
| | VAL != 0.0 | 1 | Stop At Zero is True. Stops counting at zero if counting down from a positive count or up from a negative count. |

## Analog Inputs

Table 143: Analog Inputs of Counter Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| Count Value | >= – infinity | <+ infinity | unconnected | Set Count Value = 1.0. The default value = 1.0 |
| | | | Invalid | Set Count Value = 1.0 |
| | | | VAL < low | Set Count Value = 1.0 |
| | | | VAL > high | Set Count Value = 1.0 |
| Preset Value | >= – infinity | <+ infinity | unconnected | Set Preset Value = 0.0 |
| | | | Invalid | Set Preset Value = 0.0 |
| | | | VAL < low | Set Preset Value = 0.0 |
| | | | VAL> high | Set Preset Value = 0.0 |

## Output

Table 144: Output of Counter Function

| Output Name | Range | Description |
|---|---|---|
| COUNT | Any floating-point number | Counter value |

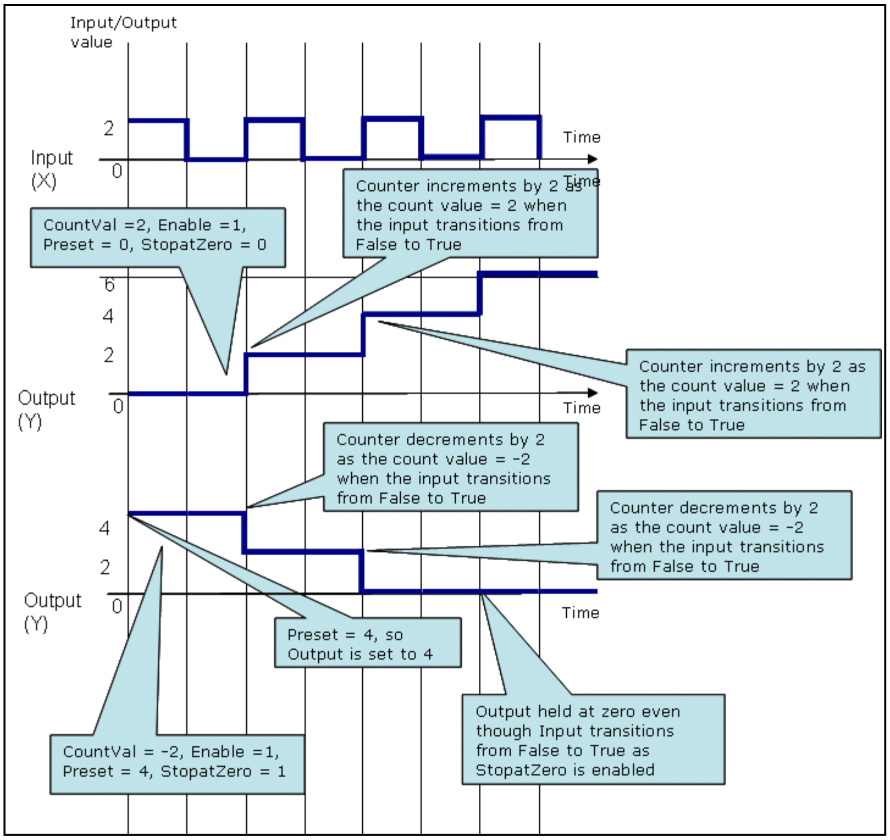
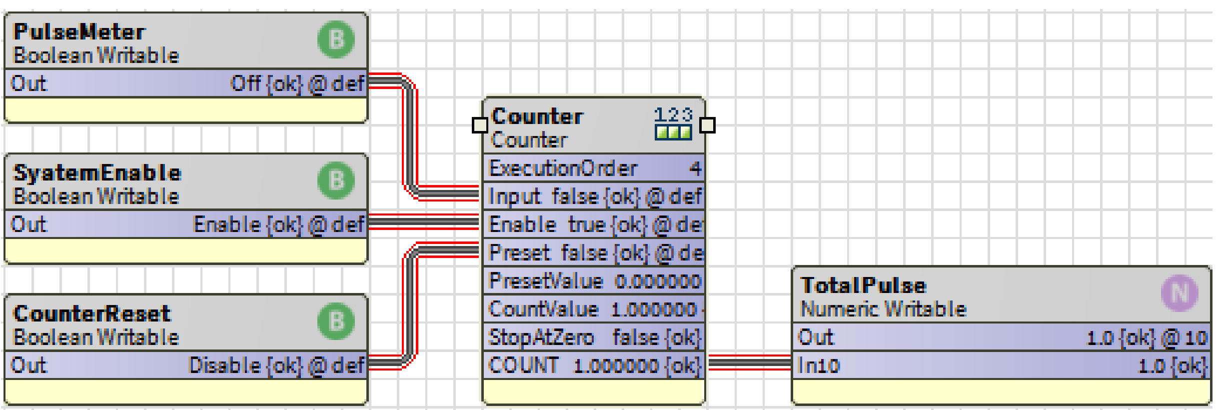



Figure 339: Transition versus Time with Positive and Negative Count Values

Example: The counter setting to accumulate Pulse meter signal. (Note: set the Count value 1 or more as per requirement)



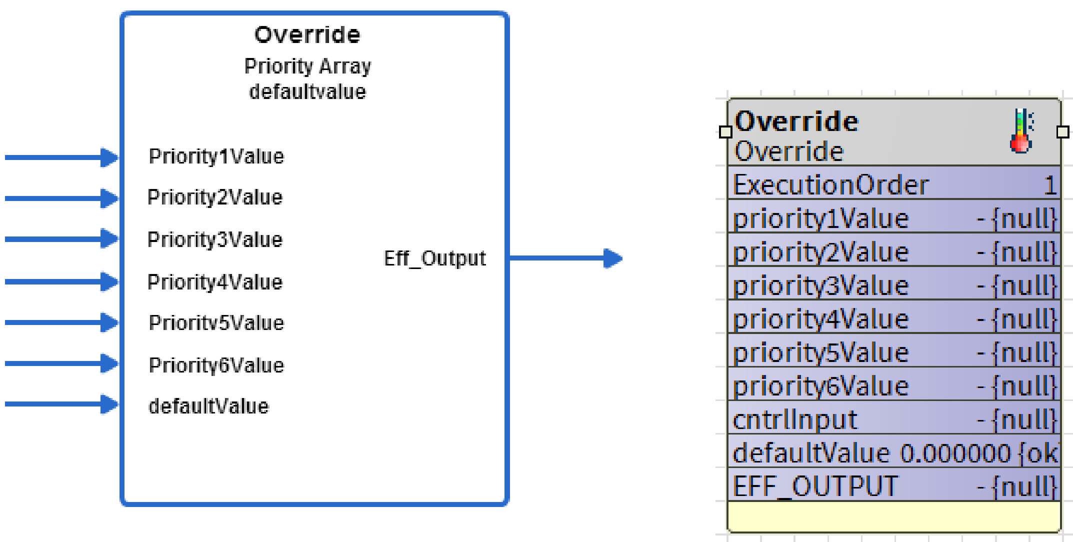
## Override

This function sets the output to the highest priority input that is not invalid. The Priority1 value has the highest priority and cntrlInput the lowest priority. This function block checks if the Inputs are not invalid in the following order:

1. priority1Value
2. priority2Value
3. priority3Value
4. priority4Value
5. priority5Value
6. priority6Value
7. CtrlInput

The first value that is not invalid in the order of priority is set as the output. If all inputs are invalid or unconnected, the output is set to the defaultValue.

This function block corresponds to the BACnet priority array implementation with the replacement of the BACnet NULL state with invalid.




Logic Diagram

Function Block

Figure 340: Override Function

## Analog Inputs

Table 145: Analog Inputs of Override Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| priority1Value through priority6Value | >= –∞ | <+∞ | Unconnected or invalid | Output = highest priority input (priority1Val is top priority and cntrlInput is lowest priority) that is not invalid or unconnected. If no inputs are valid, use defaultValue. |
| cntrlInput | >= –∞ | <+∞ | Unconnected or invalid | Output = highest priority input (priority1Val is top priority and cntrlInput is lowest priority) that is not invalid or unconnected. If no inputs are valid, use defaultValue. |
| defaultValue | >= –∞ | <+∞ | unconnected | defaultValue = invalid |
| | | | invalid | defaultValue = invalid |

## Output

Table 146: Output of Override Function

| Output Name | Range | | Description |
|---|---|---|---|
| effOutput | >= –∞ | <+∞ | effOutput = highest priority input that is not invalid. |

Example

Set the Inputs to the following:

- Priority 1 Value = Invalid
- Priority 2 Value = Invalid
- Priority 3 Value = 50
- Priority 4 Value = 60
- Priority 5 Value = –20
- Priority 6 Value = 80
- Ctrl Input = 30

The output is set as 50. Priority 1 and Priority 2 values are invalid. The next highest priority value (Priority 3 value = 50) is set as the output.

An invalid input to this function block could arise when the output of the Minimum function block can be connected whose input is invalid.

## RuntimeAccumulate

This function accumulates runtime whenever the input is True (non-zero) and enable is True. If Preset is True, runtime is set equal to the Preset Value. Runtime is provided in four outputs of seconds, minutes, hours, and days. In each iteration, the function block keeps track of the run time seconds. On power up/reset, this track is cleared.

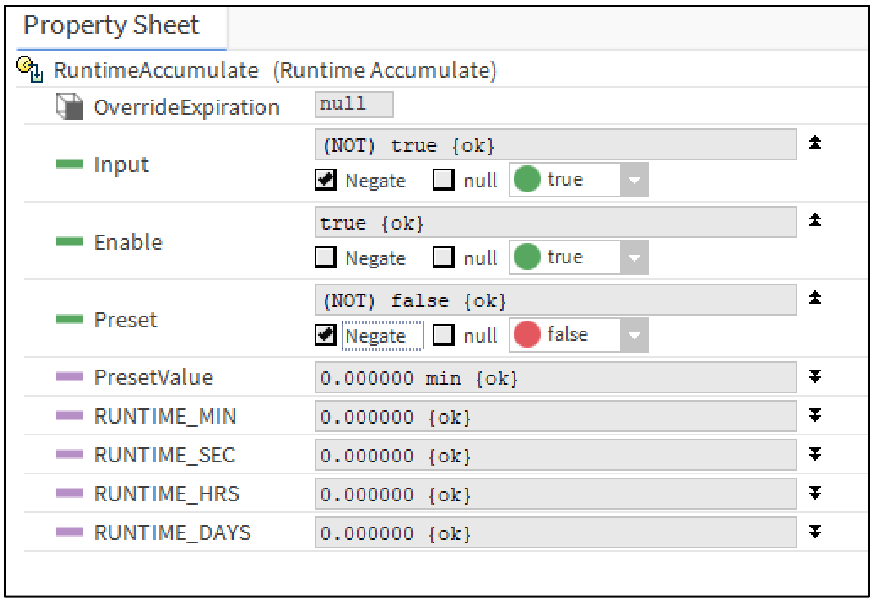
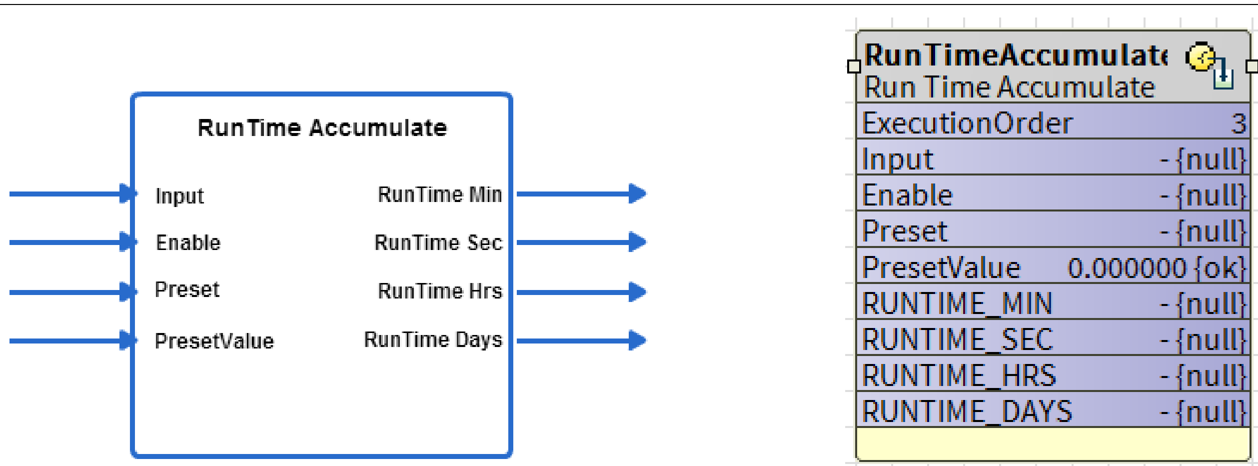



Figure 341: Runtime Accumulate Property Sheet

Unconfigured is taken as high priority compared to Negate. If nothing is configured and Negate is applied nothing will get effected. Unconfigured will be taken as high priority.

| Note: |
|---|
| On power up/reset, only the Runtime Sec output is set to zero. The other three outputs, RUNTIME_MIN, RUNTIME_HRS, and RUNTIME_DAYS are stored and not lost. |




Logic Diagram

Function Block

Figure 342: Runtime Accumulate Function

## Logic Inputs

Table 147: Logic Inputs of Runtime Accumulate Function

| Input Name | Input Value | Logic Value | Description |
|---|---|---|---|
| Input | uncon- | 0 | Set Input = False |
| | invalid | 0 | Set Input = False |
| | 0 | 0 | Input is False |
| | VAL != 0.0 | 1 | Input is True |
| Enable | uncon- nected | 1 | Set Enable = True |
| | invalid | 1 | Set Enable = True |
| | 0 | 0 | Enable is False |
| | VAL != 0.0 | 1 | Enable is True |
| Preset | uncon- nected | 0 | Set Preset = False |
| | invalid | 0 | Set Preset = False |
| | 0 | 0 | Preset is False |
| | VAL != 0.0 | 1 | Preset is True |

## Analog Inputs

Table 148: Analog Inputs of Runtime Accumulate Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| PresetValue | 0 | <+8 | unconnected | Set Preset Value = 0.0 (in minutes) |
| | | | invalid | Set Preset Value = 0.0 |
| | | | VAL < low | Set Preset Value = 0.0 |
| | | | VAL > high | Set Preset Value = 0.0 |

## Output

Table 149: Outputs of Runtime Accumulate Function

| Output Name | Range | Description |
|---|---|---|
| RUNTIME_MIN | Any floating-point number >= | Runtime minutes |

| Output Name | Range | Description |
|---|---|---|
| RUNTIME_SEC | Any floating-point number >= | Runtime seconds |
| RUNTIME_HRS | Any floating-point number >= | Runtime hours |
| RUNTIME_DAYS | Any floating-point number >= | Runtime days |

## Operation

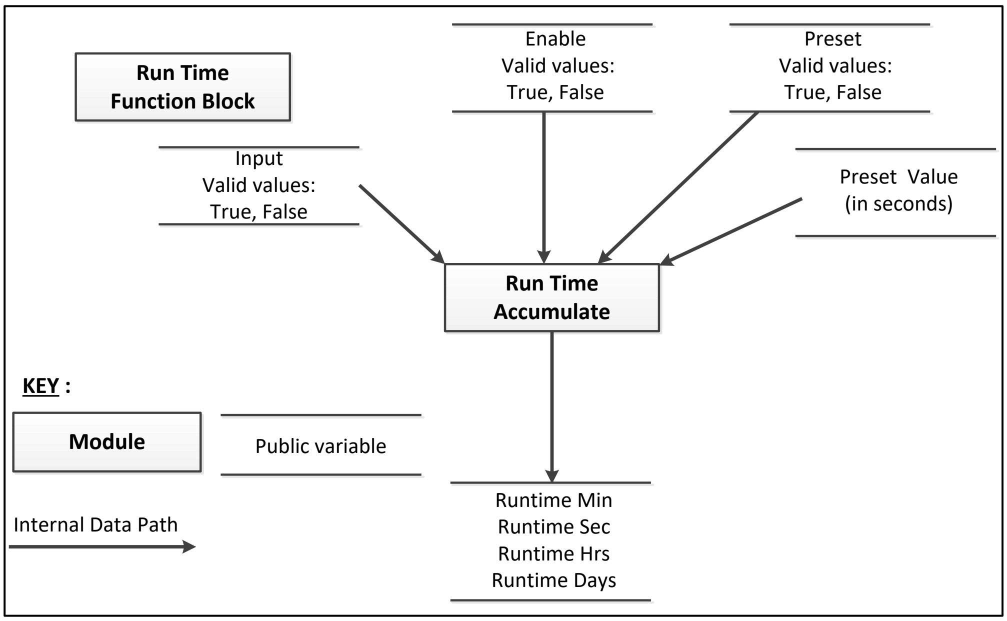



Figure 343: RunTime Function Block

Run time is always accumulated internally in minutes. It is reported in 4 different units of seconds, minutes, hours and days. Run time Min is saved over a power outage and reset. If a power outage or reset occurs, the controller could lose up to one minute of runtime. RUNTIME_MIN, RUNTIME_HRS, and RUNTIME_DAYS are calculated at every iteration from the RUNTIME_MIN.

RUNTIME_HRS and RUNTIME_DAYS outputs are fractional units to the nearest minute. RUNTIME_SEC is RUNTIME_MIN multiplied by 60. The preset input should be used to set the runtime to an initial value in minutes.

Runtime Accumulate is run every second. The state of the input, enable, and the preset are examined by the function block when it is run. Momentary transitions of the inputs between invocations of the function block are not detected. If the runtime reaches 16,277,216 minutes, it stops.

RUNTIME_MIN is effectively limited to 16, 277,216 minutes (31 years).

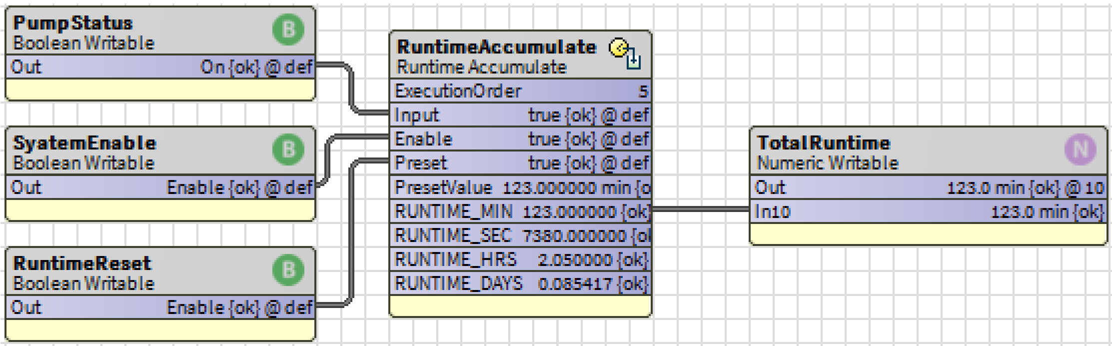
Example

Connect an output from another block to the Input. Connect a digital input to Preset. Set the Preset Value to 123. Set the PresetValue to 255 (TRUE).

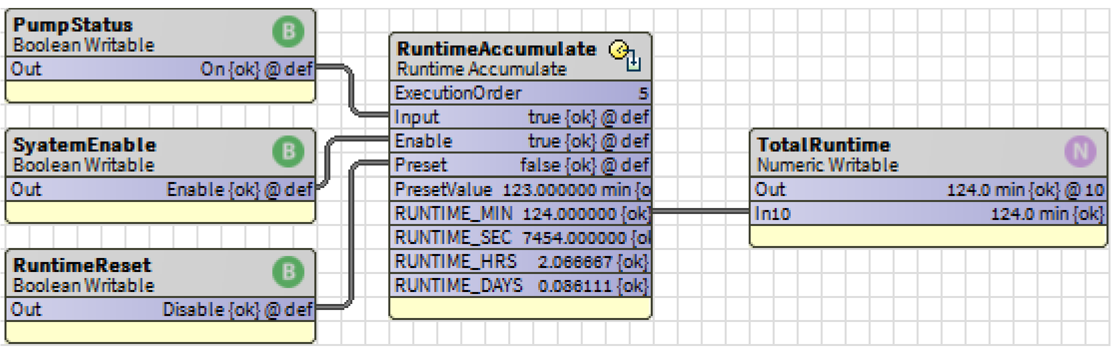
The four outputs are as follows:

- RUNTIME_MIN = 123

- RUNTIME_SEC = 7380
- RUNTIME_HRS = 2.05
- RUNTIME_DAYS = 0.085416




Once you release the Preset point, the block will start accumulating the runtime value from the preset value.



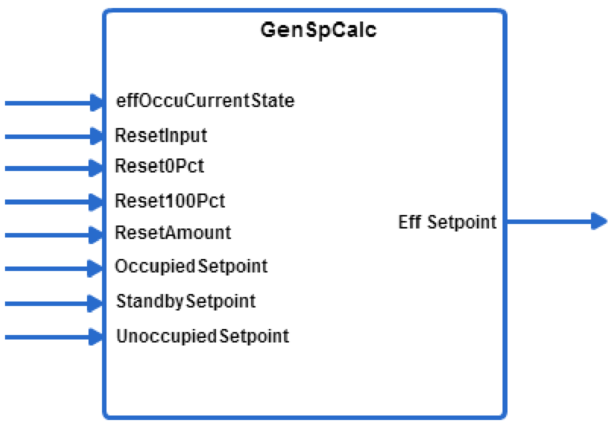
# ZoneControl Function Blocks

The CIPer Model 30 programming model provides the following Zone Arbitration function blocks that you can configure and use to build the required application logic:

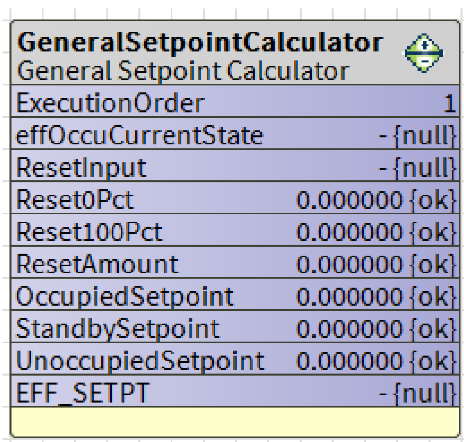
- GeneralSetpointCalculator
- OccupancyArbitrator
- SetTemperatureMode
- TemperatureSetPointCalculator

## GeneralSetpointCalculator

This function does a generic setpoint calculation, including reset. It uses the three configuration parameters, that is effective occupancy, current state, and reset input to calculate the effective setpoint.




Logic Diagram




Function Block

Figure 344: General Set Point Calculator Function

## Analog Inputs

Table 150: Analog Inputs of General Set Point Calculator Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| Eff Occ Current state | 0 | 3 | uncon-nected | Eff Occ Current state = 0 (OCC) |
| | | | invalid | Eff Occ Current state = 0 (OCC) |
| | | | VAL < low | Eff Occ Current state = 0 (OCC) |
| | | | VAL > high | Eff Occ Current state = 0 (OCC) |
| Reset Input | >= –∞ | <+∞ | uncon-nected | Reset Input = Invalid |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| | | | invalid | Reset Input = Invalid |
| | | | VAL < low | Reset Input = Invalid |
| | | | VAL > high | Reset Input = Invalid |
| Reset 0Pct | >= –∞ | <+∞ | uncon-nected | Reset 0Pct = Invalid |
| | | | invalid | Reset 0Pct = Invalid |
| | | | Val < low | Reset 0Pct = Invalid |
| | | | Val > high | Reset 0Pct = Invalid |
| Reset 100Pct | >= –∞ | <+∞ | uncon-nected | Reset 100Pct = Invalid |
| | | | invalid | Reset 100Pct = Invalid |
| | | | Val < low | Reset 100Pct = Invalid |
| | | | Val > high | Reset 100Pct = Invalid |
| Reset Amount | >= –∞ | <+∞ | uncon-nected | Reset Amount = Invalid |
| | | | invalid | Reset Amount = Invalid |
| | | | Val < low | Reset Amount = Invalid |
| | | | Val > high | Reset Amount = Invalid |
| Occupied Setpoint | >= –∞ | <+∞ | uncon-nected | Occupied Setpoint = Invalid |
| | | | invalid | Occupied Setpoint = Invalid |
| | | | Val < low | Occupied Setpoint = Invalid |
| | | | Val > high | Occupied Setpoint = Invalid |
| Standby setpoint | >= –∞ | <+∞ | uncon-nected | Standby Setpoint = Invalid |
| | | | invalid | Standby Setpoint = Invalid |
| | | | Val < low | Standby Setpoint = Invalid |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| | | | Val > high | Standby Setpoint = Invalid |
| Unoccupied set-point | >= –∞ | <+∞ | uncon-nected | Unoccupied Setpoint = Invalid |
| | | | invalid | Unoccupied Setpoint = Invalid |
| | | | Val < low | Unoccupied Setpoint = Invalid |
| | | | Val > high | Unoccupied Setpoint = Invalid |

State value:

- Occ = 0
- Unocc=1
- Bypass =2
- Standby = 3
- Null = 255

## Output

Table 151: Output of General Set Point Calculator Function

| Input Name | Range | Description |
|---|---|---|
| Eff Setpoint | Any floating-point number | Effective Setpoint |

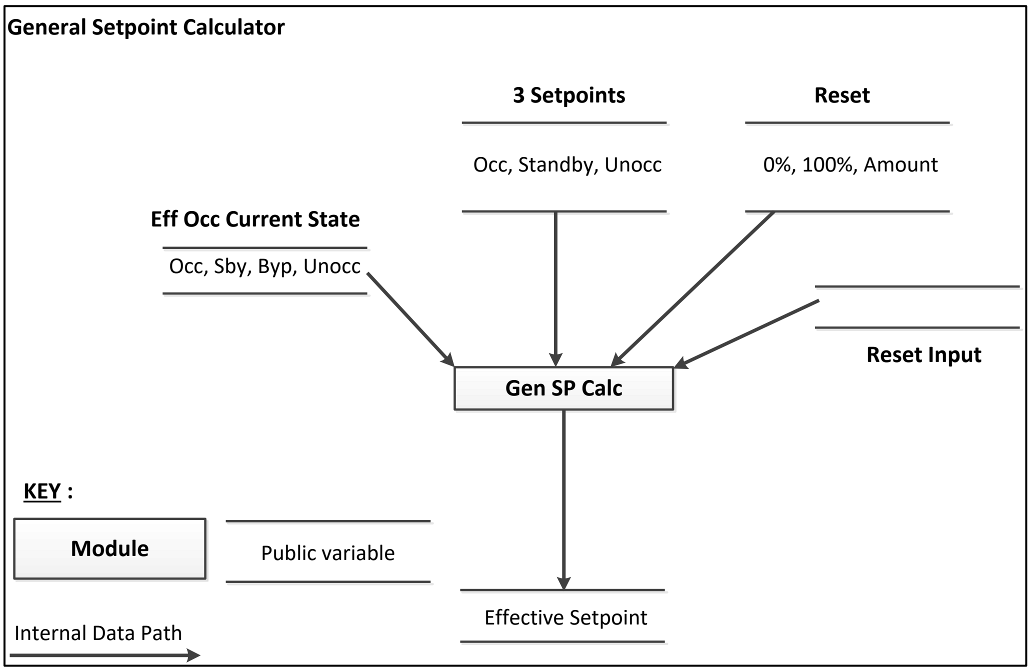



Figure 345: General Set Point Calculator

## Reset Input

Reset allows to change the effective setpoint either in the direction of increased energy savings or in the direction of increased comfort. The Reset Amount (+/-) is positive or negative to accommodate energy savings versus comfort. The reset value varies between zero and the reset amount and is proportional to the Reset Input with respect to the Reset 0% and Reset 100% parameters.

> **Note:**
>
> Ensure that the Reset 0% and Reset 100% parameters are in the same engineering unit as the Reset Input. The Reset Amount should be in the same units as the configured setpoints.

Positive reset values are added to the setpoint and negative resets are subtracted. Reset only applies in the occupied mode. Reset 0% can be any relation to Reset 100%. The following illustration shows Reset 0% less than Reset 100% with a positive reset amount. If any of the Reset Input, Reset 0%, Reset 100% or Reset Amount parameters is invalid, the Reset value is set to zero (0).

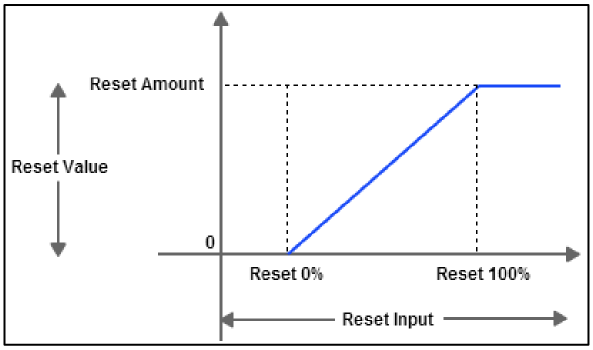



Figure 346: Reset Calculation: Positive Amount 0% < 100%

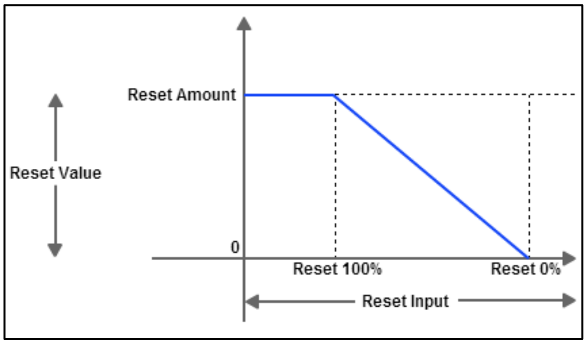



Figure 347: Reset Calculation: Positive Amount 100% < 0%

Eff Occ Current State

The effective occupancy current state comes from a scheduler. The valid values are:

- Occupied
- Unoccupied
- Bypass
- Standby
- Null

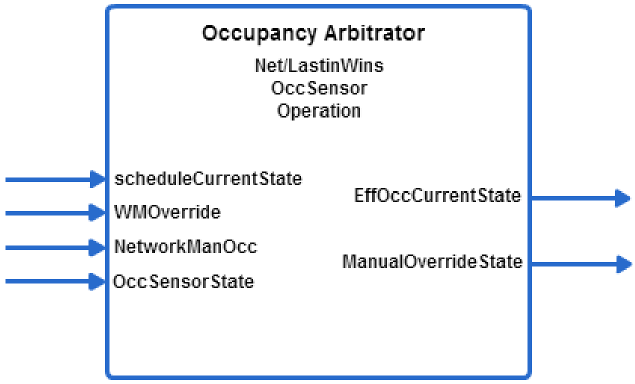
The General Setpoint Calculator uses the three configured setpoints: effective occupancy, current state, and reset input to determine the effective setpoint. If a setpoint is invalid, INVALID is propagated to the output as appropriate.

Table 152: Valid values of Effective Occupancy Current State

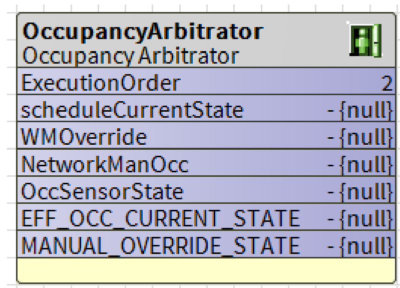
| Eff Occ Current State | Eff Setpoint |
|---|---|
| UNOCC | Result = unoccupied setpoint |
| STANDBY | Result = standby setpoint |
| OCC | Result = occupied setpoint + reset |
| BYPASS | Result = occupied setpoint + reset |
| NULL | Result = occupied setpoint + reset |

## OccupancyArbitrator

This function computes the present Effective Occupancy Current State and the Manual Override State.




Logic Diagram




Function Block

Figure 348: Occupancy Arbitrator

### Inputs

Table 153: Inputs of Occupancy Arbitrator

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| scheduleCurrent-State | 0 | 1,3,255 | uncon-nected | Schedule Current State = 255 (OC-CNUL) |
| | | | invalid | Schedule Current State = 255 (OC-CNUL) |
| | | | VAL < low | Schedule Current State =0 (OCC) |
| | | | VAL > high | Schedule Current State = 255 (OC-CNUL) |
| WMOverride | 0 | 1-3,255 | uncon-nected | WM Override = 255 (OCCNUL) |
| | | | invalid | WM Override = 255 (OCCNUL) |
| | | | VAL < low | WM Override = 0 (OCC) |
| | | | VAL > high | WM Override = 255 (OCCNUL) |
| NetworkManOcc | 0 | 1-3,255 | uncon-nected | Network Man Occ = 255 (OCCNUL) |
| | | | invalid | Network Man Occ = 255 (OCCNUL) |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| | | | VAL < low | Network Man Occ = 0 (OCC) |
| | | | VAL > high | Network Man Occ = 255 (OCCNUL) |
| OccSensorState | 0 | 1, 255 | uncon-nected | Occ Sensor State = 255 (OCCNUL) |
| | | | invalid | Occ Sensor State = 255 (OCCNUL) |
| | | | VAL < low | Occ Sensor State = 0 (OCC) |
| | | | VAL > high | Occ Sensor State = 255 (OCCNUL) |

State values:

Occ = 0 Standby = 3

Unocc=1 Null = 255

Bypass =2

## Outputs

Table 154: Outputs of Occupancy Arbitrator

| Output Name | Range | Description |
|---|---|---|
| EFF_OCC_CUR-RENT_STATE | 0 to 3 (Occupied, Unoccupied, Bypass, Standby) | Effective Occupancy Cur-rent state |
| MANUAL_OVER-RIDE_STATE | 0 to 3, 255 (Occupied, Unoccupied, Bypass, Standby, Null) | Manual Override State |

## Configuration

Specify Net wins (0) or Last in wins (1). Specify the occupancy sensor operation.

There are 3 choices: Conference room (0), Unoccupied Cleaning Crew (1), and Unoccupied Tenant (2).

Table 155: Configuration of Occupancy Arbitrator

| Name | Range | Description |
|---|---|---|
| net-LastInWins | 0 to 1 | You can specify Net wins (0) or Last in wins (1). |
| occSenso-rOper | 0 to 2 | You can specify the occupancy sensor operation. There are 3 choices: Con-ference room (0), Unoccupied Cleaning Crew (1), and Unoccupied Tenant |

| Name | Range | Description |
|---|---|---|
| | | (2). See the bottom of the Occupancy Arbitrator table for differences between these options. |

## Operation

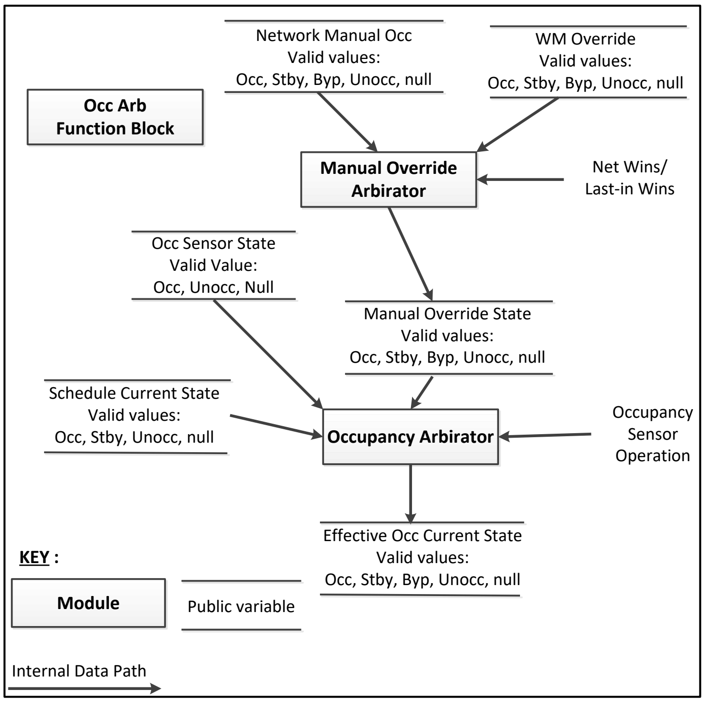



Figure 349: Occupancy Arbitrator Function

### Manual Override Arbitration Mechanism

Manual override arbitration mechanism determines the value of MANUAL_OVERRIDE_STATE. This value is used as an input to the Occupancy Arbitrator.

The Manual Override Arbitrator uses either a Net Wins or a Last in Wins scheme to evaluate the inputs. Net Wins means the network command always takes precedence over the wall module command.

With Last in Wins, the last override source is used to determine the final state. If multiple sources change the state in the same second, they are evaluated in the order: Network Man Occ, WM Override. Each second the function block is called; the algorithm looks for a change of state to Network Man Occ or WM Override. If either of these changed state, appropriate action is taken. Generally, a new command on any input, cancels prior action by another source.

Table 156: Truth Table of Net Wins and Last Wins

| Net Wins/ Last in Wins | Network Man Occ (note2) | WM Override (note 2) | RESULT: Manual Override State | Comment |
|---|---|---|---|---|
| Last in Wins | OCC | Don't Care | OCC | Result set to Network Man Occ. |
| Last in Wins | UNOCC | Don't Care | UNOCC | Result set to Network Man Occ. |
| Last in Wins | BYPASS | Don't Care | BYPASS | Result set to Network Man Occ. |
| Last in Wins | STANDBY | Don't Care | STANDBY | Result set to Network Man Occ. |
| Last in Wins | OCCNUL | Don't Care | OCCNUL | Override canceled. |
| Last in Wins | Don't Care | OCC | OCC | Result set to the wall module override. |
| Last in Wins | Don't Care | STANDBY | STANDBY | Result set to the wall module override. |
| Last in Wins | Don't Care | BYPASS | BYPASS | Result set to the wall module override. |
| Last in Wins | Don't Care | UNOCC | UNOCC | Result set to the wall module override. |
| Last in Wins | Don't Care | OCCNUL | OCCNUL | Override canceled. |

Note:

Any other input value not listed, is not a valid state. If received, it is treated as OCCNUL.

For last in wins, the value in the above table was just changed from another state and this is the current state.

The manual override command is hold ON for the Bypass time outside the function block when Manual Override command is triggered. The function block does not have an ability to hold the Manual Override Command for the required Bypass time or don't have any parameter for setting the bypass time.

If manual Override Command is coming from ConventionalWallModule then it is held On for the bypass time by the wall module after triggering of the command.

If network Manual Override Command is there, additional logic in the control program (or other device from the Manual Override Command is received over a Lon network) is need to be implemented for the bypass time. It need to be implemented.

From iteration to iteration of the Function Block, the Occupancy Arbitrator keeps track of the last state of the Network Man Occ and WM Override inputs so that it knows when a transition occurs. On power up/reset the last latch value is set to FALSE, regardless of the negation configuration. Override is canceled, after a power outage. The Network Man Occ and WM Override inputs must reset themselves after a power outage.

Network Manual Occupancy Input

NetworkManOcc is a method to command the occupancy state from a network workstation or a node. You may write logic to combine these, if both are required for the application. NetworkManOcc can command the state to be occupied, unoccupied, standby, bypass or null. It is required that the workstation (nviManOccCmd) or network node (nviBypass) performs any timing needed that is bypass.

WM Override Input

WM Override is a method to command the occupancy state from a locally wired wall module. WM Override can command the state to be occupied, unoccupied, standby, bypass or null. It is required that the function block wired to this input, perform any timing needed that is bypass.

| Note: |
|---|
| The current T7770 wall module function doesn’t support occupied or standby override, but future wall modules might. |

Occupancy Arbitration Mechanism

The Occupancy Arbitrator computes the effective occupancy status. The inputs of the Effective Occupancy Arbitrator include the Schedule Current State, Occ Sensor State, and Manual Override State. The Manual Override State comes from Network Man Occ or WM override.

The Effective Occupancy Arbitrator sets the Effective Occ Current State. Valid states of current state are:

- OCC: The space is occupied.
- UNOCC: The space is unoccupied.
- BYPASS: The space is occupied, though it is not scheduled to be occupied.
- STANDBY: The space is in a standby state, somewhere between occupied and unoccupied.

OCCNUL is not a valid output. If all inputs are OCCNUL, the output is set to occupied.

Table 157: Truth Table of Valid States of Effective Occ Current State

| Manual Override State | Schedule Current State | Occ Sensor State | Occ Sensor Operation | RESULT: effOcc Current-State | Comments | Follows LonMark SCC |
|---|---|---|---|---|---|---|
| OCC | Don’t Care | Don’t Care | Don’t Care | OCC | Result = Manual Override State | Yes. |

| Manual Override State | Schedule Current State | Occ Sensor State | Occ Sensor Operation | RESULT: effOcc Current-State | Comments | Follows LonMark SCC |
|---|---|---|---|---|---|---|
| STANDBY | Don't Care | Don't Care | Don't Care | STANDBY | Result = Manual Override State | Yes |
| UNOCC | Don't Care | Don't Care | Don't Care | UNOCC | Result = Manual Override State | Yes |
| BYPASS | OCC | Don't Care | Don't Care | OCC | The result stays at occupied because bypass isn't effective when scheduled for occupied | Yes |
| BYPASS | STANDBY | Don't Care | Don't Care | BYPASS | The result stays at bypass. | Yes |
| BYPASS | UNOCC | Don't Care | Don't Care | BYPASS | Result = bypass | Yes |
| BYPASS | OCCNUL | OCC | Don't Care | OCC | The result follows occupancy sensor | Yes |
| BYPASS | OCCNUL | UNOCC | Don't Care | BYPASS | The result follows manual override | Yes |
| BYPASS | OCCNUL | OCCNUL | Don't Care | OCC | When occupancy sensor is null, default to occupied. | Yes |
| OCCNUL | STANDBY | Don't Care | Don't Care | STANDBY | Result = scheduled state. | Yes |
| OCCNUL | OCC | OCC | Don't Care | OCC | All say we are Occupied. | Yes |
| OCCNUL | OCC | UNOCC | Don't Care | STANDBY | We are scheduled to be occupied, but the room is unoccupied, | Yes |

| Manual Override State | Schedule Current State | Occ Sensor State | Occ Sensor Opera-tion | RESULT: effOcc Current-State | Comments | Fol-lows Lon-Mark SCC |
|---|---|---|---|---|---|---|
| | | | | | so go to standby to save energy. | |
| OCCNUL | OCC | OC-CNUL | Don't Care | OCC | Sensor not present so use schedule. | Yes |
| OCCNUL | UNOCC | UN-OCC | Don't Care | UNOCC | All say we're unoccu-pied. | Yes |
| OCCNUL | UNOCC | OC-CNUL | Don't Care | UNOCC | Sensor not present so use schedule | Yes |
| OCCNUL | OCCNUL | OCC | Don't Care | OCC | Result -= occupancy sensor state. | Yes |
| OCCNUL | OCCNUL | UN-OCC | Don't Care | UNOCC | Result -= occupancy sensor state. | Yes |
| OCCNUL | OCCNUL | OC-CNUL | Don't Care | OCC | Result = occupied be-cause the LonMark SCC sets a null occu-pancy sensor to Occu-pied. | Yes |
| OCCNUL | UNOCC | OCC | Confer-ence Room | UNOCC | Stay unoccupied re-gardless of what the sensor says that is save energy. | Yes |
| OCCNUL | UNOCC | OCC | Clean-ing Crew | STANDBY | We are scheduled to be unoccupied, but the room is occupied, so go to standby for the comfort of the cleaning crew. | No |
| OCCNUL | UNOCC | OCC | Tenant | OCC | We are scheduled to | No |

| Manual Override State | Schedule Current State | Occ Sensor State | Occ Sensor Operation | RESULT: effOcc Current-State | Comments | Follows LonMark SCC |
|---|---|---|---|---|---|---|
| | | | | | be unoccupied, but the room is occupied, so, go to occupied for the comfort of the tenant. | |

## SetTemperatureMode

This function automatically calculates the effective temperature control mode based on the control type, system switch setting, network mode command, temperature set points, supply temperature and space temperature.

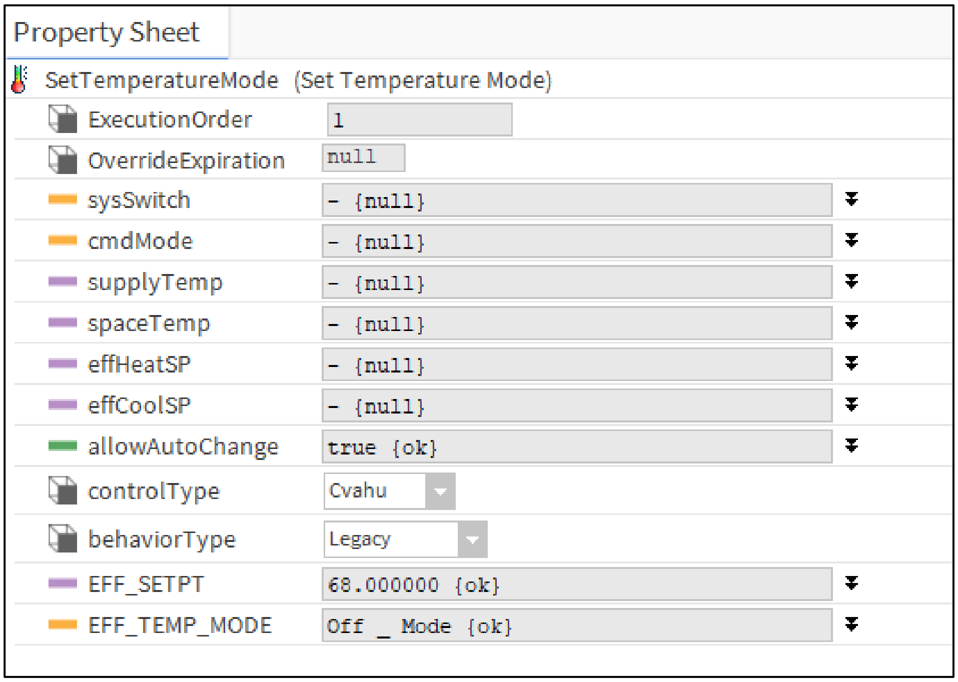
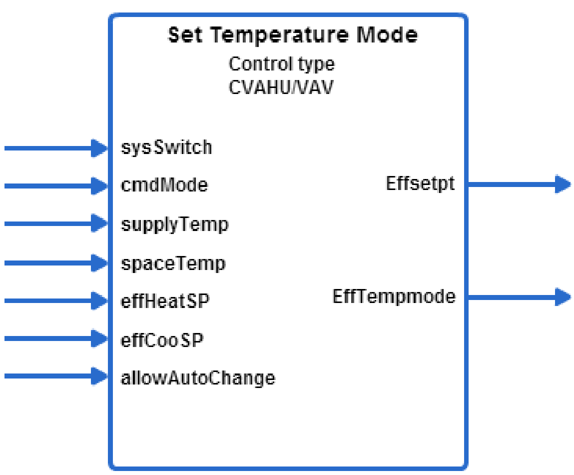


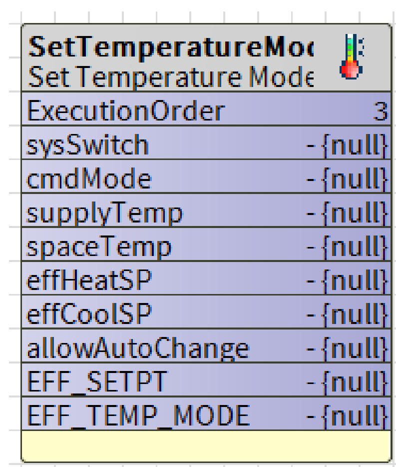

Figure 350: SetTemperatureMode Property Sheet

From iteration to iteration, the Function Block keeps track of the previous command mode and the effective temperature mode. On power up/reset, these are cleared.

effTempMode indicates the current Mode determined by input states and arbitrated by control logic. SetTempMode does not generate all the possible Modes available. The following table shows the meaning of the valid enumerated values. The following table shows the meaning of valid enumerated values.




Logic Diagram




Function Block

Figure 351: Set Temperature Mode Function

The following table shows the meaning of valid enumerated values.

Table 158: Meanings of Valid Enumerated Values

| effTempMode | Meaning |
|---|---|
| COOL_MODE = 0 | Cool air is being supplied to the node via the central air supply and cooling energy is being supplied to the controlled space. |
| REHEAT_MODE = 1 | Cool air is being supplied to the node via the central air supply. The air is being reheated by a local Heat source. |
| HEAT_MODE = 2 | Heated air is being supplied to the node via the central air supply and heated air is being supplied to the controlled space |
| EMERG_HEAT = 3 | Emergency Heat is being supplied to the node via the central air supply. |
| OFF_MODE = 255 | The controller is commanded off. |

## Analog Input

Table 159: Analog Inputs of Set Temperature Mode Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| sysSwitch | 0 | 255 | uncon-nected | System Switch = SS_AUTO(0) |
| | | | invalid | System Switch = SS_AUTO(0) |
| | | | VAL < low | System Switch = SS_AUTO(0) |
| | | | VAL > high | System Switch = SS_AUTO(0) |
| cmdMode | 0 | 255 | uncon-nected | val = CMD_AUTO_MODE(0) |
| | | | invalid | val = CMD_AUTO_MODE(0) |
| | | | VAL < low | val = CMD_AUTO_MODE(0) |
| | | | VAL > high | val = CMD_AUTO_MODE(0) |
| supplyTemp | 0 | 255 | uncon-nected | Supply Temp = invalid |
| | | | invalid | Supply Temp = invalid |
| | | | Val < low | SupplyTemp = low |
| | | | Val > high | SupplyTemp = high |
| spaceTemp | 0 | 255 | uncon-nected | SpaceTemp = invalid |
| | | | invalid | SpaceTemp = invalid |
| | | | Val < low | SpaceTemp = low |
| | | | Val > high | SpaceTemp = high |
| effHeatSP | >=- | <+ | uncon-nected | EffHeatSp = 68 |
| | | | invalid | EffHeatSp = 68 |
| effCoolSP | >=- | <+ | uncon-nected | EffCoolSp = 75 |
| | | | invalid | EffCoolSp = 75 |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| allowAuto-Change | 0 | 1 | uncon-nected | allowAutoChange=1 |
| | | | invalid | allowAutoChange=1 |
| | | | Val < low | allowAutoChange=1 |
| | | | Val > high | allowAutoChange=1 |

## Outputs

Table 160: Output of Set Temperature Mode Function

| Output Name | Range | | Description |
|---|---|---|---|
| | Low | High | |
| effSetpt | 0.0 | 255 | If effTempMode=COOL_MODE then val= effCoolSetPt, else val=eff-HeatSetPt |
| effTemp-Mode | 0 | 255 | See arbitration table for VAV and CVAHU behavior |

## Configuration

Specify the control Type (controlType)

- 0 – CVAHU
- 1 – VAV

Specify the Behavior Type

- Legacy – For normal output (Spyder Behavior)
- Enhanced – For enhanced output. If no device is connected sysSwitch considered as SS Auto as default input for both CVAHU and VAV and EFF Temp mode output displays Heat mode.

| Output Name | Range | | Description |
|---|---|---|---|
| | Low | High | |
| con-trolType | 0 | 1 | You can specify the control Type<br>0 = CVAHU; 1 = VAV |

## Input Enumerations

Table 161: Input Enumerations of Set Temperature Mode Function

| sysSwitch | |
|---|---|
| SS_AUTO | = 0 |
| SS_COOL | = 1 |
| SS_HEAT | = 2 |
| SS_EMERG_HEAT | = 3 |
| SS_OFF | = 255 |
| cmdMode | |
| CMD_AUTO_MODE= 0 | = 0 |
| CMD_ HEAT_MODE = 1 | = 1 |
| CMD_COOL_MODE = 2 | = 2 |
| CMD_OFF_MODE = 3 | = 3 |
| CMD_ EMERG_HEAT_MODE = 4 | = 4 |
| CMD_NUL_MODE = 255 | = 255 |

The CVAHU arbitration logic for ControlType = 0 (CVAHU) is summarized in the following table.

Table 162: CVAHU Arbitration Logic for ControlType = 0 (CVAHU)

| Space Temp | sysSwitch | cmdMode | effTempMode |
|---|---|---|---|
| X | X | CMD_OFF(3) | OFF_MODE(255) |
| X | X | CMD_EMERG_HEAT_MODE(4) | EMERG_HEAT(3) |
| X | X | CMD_COOL_MODE(2) | COOL_MODE(0) |
| X | X | CMD_HEAT_MODE(1) | HEAT_MODE(2) |
| X | X | ENUMERATION (5) through ENU-MERATION (254) | HEAT_MODE(2) |
| X | SS_COOL (1) | CMD_AUTO_MODE(0), CMD_NUL_MODE(255) | COOL_MODE (0) |
| X | SS_HEAT (2) or ENU-MERATION(4) through ENUMERATION (254) | CMD_AUTO_MODE(0), CMD_NUL_MODE(255) | HEAT_MODE(2) |
| X | SS_EMER-GENCY_HEAT(3) | CMD_AUTO_MODE(0), CMD_NUL_MODE(255), | EMERG_HEAT(3) |
| X | SS_OFF (255) | CMD_AUTO_MODE(0), CMD_NUL_MODE(255) | OFF_MODE(255) |

| Space Temp | sysSwitch | cmdMode | effTempMode |
|---|---|---|---|
| INVALID | SS_AUTO (0), invalid, unconected, or a non-listed enumeration. | CMD_AUTO_MODE(0), CMD_NUL_MODE(255) | HEAT_MODE(2) |
| VALID | SS_AUTO (0), invalid, unconected, or a non-listed enumeration. | CMD_AUTO_MODE(0), CMD_NUL_MODE(255), | COOL_MODE(0) or HEAT_MODE(2)<br>(See the note following this table.) |

Note:

- X means don’t care.
- If allowAutoChange = 1 then allow to switch between HEAT_MODE and COOL_MODE.
- Must have valid effHeatSP and effCoolSP. If allowAutoChange = 1 and effHeatSp > effCoolSp then effHeatSp is internally set to effCoolSP.

The VAV Mode arbitration logic for controlType = 1 the following table summarizes (VAV):

Table 163: VAV Mode Arbitration Logic for controlType = 1

| Space Temp | sysSwitch | Supply Temp | cmdMode | effTempMode |
|---|---|---|---|---|
| X | X | X | CMD_OFF_MODE (3) | OFF_MODE (255) |
| X | X | X | CMD_EMERG_HEAT_MODE (4) | HEAT_MODE (2) |
| X | X | X | ENUMERATION (5) through ENUMERATION (254) | COOL_MODE (0) |
| Valid | X | <70.0 | CMD_AUTO_MODE (0)<br>CMD_HEAT_MODE (1)<br>CMD_NUL_MODE (255) | COOL_MODE (0) or<br>REHEAT_MODE (1)<br>(See the note above this table.) |
| Valid | X | 70.0 To 75.0 | CMD_AUTO_MODE (0)<br>CMD_HEAT_MODE (1) | COOL_MODE (0)<br>REHEAT_MODE |

| Space Temp | sysSwitch | Supply Temp | cmdMode | effTempMode |
|---|---|---|---|---|
| | | | CMD_COOL_MODE (2)<br>CMD_NUL_MODE (255) | (1)<br>HEAT_MODE (2)<br>(See the note above this table for transition between cool mode and reheat mode.) |
| Valid | X | >75 | CMD_AUTO_MODE (0)<br>CMD_HEAT_MODE (1)<br>CMD_NUL_MODE (255) | HEAT_MODE (2) |
| Valid | X | Invalid or unconnected | CMD_HEAT_MODE (1) | HEAT_MODE (2) |
| Valid | X | Invalid or unconnected | CMD_COOL_MODE (2) | COOL_MODE (0) |
| Valid | SS_COOL (1) | Invalid or unconnected | CMD_AUTO_MODE (0)<br>CMD_NUL_MODE (255) | COOL_MODE (0) |
| Valid | SS_HEAT (2) | Invalid or unconnected | CMD_AUTO_MODE (0)<br>CMD_NUL_MODE (255) | HEAT_MODE (2) |
| Valid SS_ | EMERGENCY_HEAT (3) | Invalid or unconnected | CMD_AUTO_MODE (0)<br>CMD_NUL_MODE (255) | HEAT_MODE (2) |
| Valid | SS_OFF (255) | Invalid or unconnected | CMD_AUTO_MODE (0)<br>CMD_NUL_MODE (255) | OFF_MODE (255) |
| Valid | SS_AUTO (0), invalid, unconnected, or a non-listed enumeration. | Invalid or unconnected | CMD_AUTO_MODE (0)<br>CMD_NUL_MODE (255) | COOL_MODE (0) or<br>REHEAT_MODE (1)<br>(See the note |

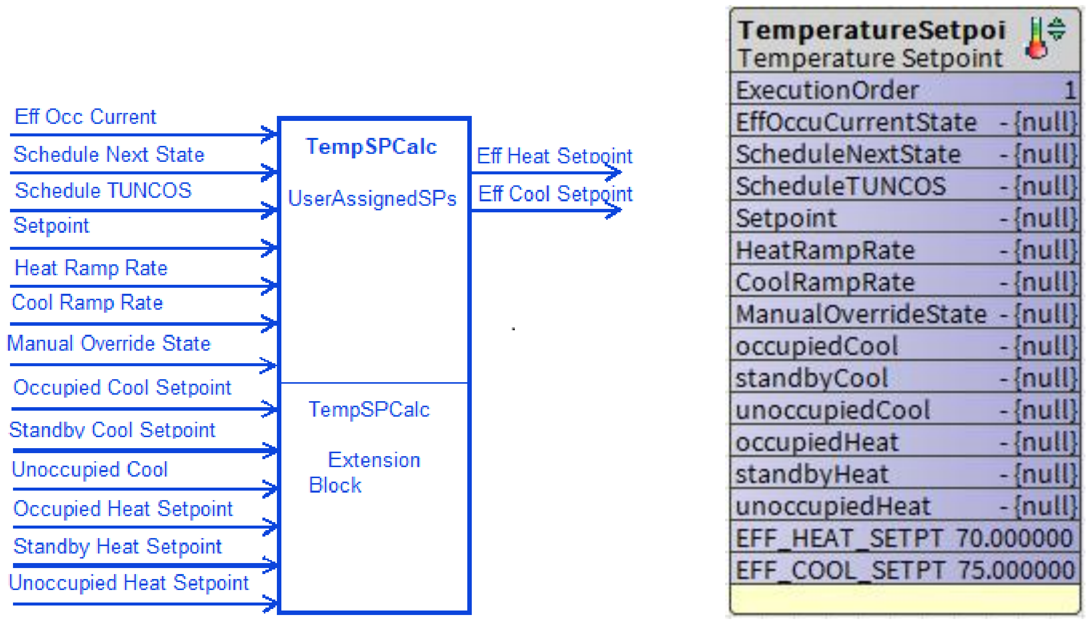
| Space Temp | sysSwitch | Supply Temp | cmdMode | effTempMode |
|---|---|---|---|---|
| | | | | above this table.) |
| Inva-lid | SS_AUTO (0), invalid, un-connected, or a non-listed enumeration. | Invalid or un-connected | CMD_AUTO_MODE (0)<br>CMD_NUL_MODE (255) | COOL_MODE (0) |

| Note: |
|---|

- X means don't care.
- If allowAutoChange = 1 then allow to switch between REHEAT_MODE and COOL_MODE. Must have valid effHeatSP and effCoolSP.
- If in cool mode and spaceTemp < effHeatSetPt and space temp < effCoolSetPt – 1.0 then go to reheat mode. If in reheat mode and spacetemp > effCoolSetPt and spacetemp > effHeatSetPt + 1.0 then go to cool mode.

## TemperatureSetpointCalculator

This function calculates the current Effective Heat setpoint and Effective Cool setpoint based on the current schedule information, occupancy override, and intelligent recovery information.




Logic Diagram    Function Block

Figure 352: Temperature Set Point Calculator Function

### Analog Inputs

Table 164: Inputs of Temperature Set Point Calculator Function

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| EffOccCurrent-State | 0 | 3 | uncon-nected | Eff Occ Current State = 0 (OCC) |
| | | | invalid | Eff Occ Current State = 0 (OCC) |
| | | | VAL < low | Eff Occ Current State = 0 (OCC) |
| | | | VAL > high | Eff Occ Current State = 0 (OCC) |
| ScheduleN-extState | 0 | 1, 3, 255 | uncon-nected | Schedule Next State = 255 (OCCNUL) |
| | | | invalid | Schedule Next State = 255 (OCCNUL) |
| | | | VAL < low | Schedule Next State = 255 (OCCNUL) |
| | | | VAL > high | Schedule Next State = 255 (OCCNUL) |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| ScheduleTUNCOS (min) | 0 | 11520 | uncon-nected | Schedule TUNCOS = 11520 |
| | | | invalid | Schedule TUNCOS = 11520 |
| | | | VAL < low | Schedule TUNCOS = 0 |
| | | | VAL > high | Schedule TUNCOS = 11520 |
| Setpoint | >= –∞ | <+∞ | uncon-nected | Setpoint = 0 |
| | | | invalid | Setpoint = 0 |
| | | | VAL < low | Setpoint = 0 |
| | | | VAL > high | Setpoint = 0 |
| HeatRampRate | 0 | <+∞ | uncon-nected | Heat Ramp Rate = 0 |
| | | | invalid | Heat Ramp Rate = 0 |
| | | | VAL < low | Heat Ramp Rate = 0 |
| | | | VAL > high | Heat Ramp Rate = 0 |
| CoolRampRate | 0 | <+∞ | uncon-nected | Cool Ramp Rate = 0 |
| | | | invalid | Cool Ramp Rate = 0 |
| | | | VAL < low | Cool Ramp Rate = 0 |
| | | | VAL > high | Cool Ramp Rate = 0 |
| ManualOver-rideState | 0 | 3,255 | uncon-nected | Manual Override State = 255 (OC-CNUL) |
| | | | invalid | Manual Override State = 255 (OC-CNUL) |
| | | | VAL < low | Manual Override State = 255 (OC-CNUL) |
| | | | VAL > high | Manual Override State = 255 (OC-CNUL) |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| occupiedCool | >= –∞ | <+∞ | uncon-nected | Setpoint = 75 |
| | | | invalid | Setpoint = 75 |
| | | | VAL < low | Setpoint = 75 |
| | | | VAL > high | Setpoint = 75 |
| standbyCool * | >= –∞ | <+∞ | uncon-nected | Setpoint = 78 |
| | | | invalid | Setpoint = 78 |
| | | | VAL < low | Setpoint = 78 |
| | | | VAL > high | Setpoint = 78 |
| unoccupiedCool * | >= –∞ | <+∞ | uncon-nected | Setpoint = 85 |
| | | | invalid | Setpoint = 85 |
| | | | VAL < low | Setpoint = 85 |
| | | | VAL > high | Setpoint = 85 |
| occupiedHeat * | >= –∞ | <+∞ | uncon-nected | Setpoint = 70 |
| | | | invalid | Setpoint = 70 |
| | | | VAL < low | Setpoint = 70 |
| | | | VAL > high | Setpoint = 70 |
| standbyHeat * | >= –∞ | <+∞ | uncon-nected | Setpoint = 67 |
| | | | invalid | Setpoint = 67 |
| | | | VAL < low | Setpoint = 67 |
| | | | VAL > high | Setpoint = 67 |
| unoccupiedHeat * | >= –∞ | <+∞ | uncon-nected | Setpoint = 55 |

| Input Name | Range | | Input Value | Description |
|---|---|---|---|---|
| | Low | High | | |
| | | | invalid | Setpoint = 55 |
| | | | VAL < low | Setpoint = 55 |
| | | | VAL > high | Setpoint = 55 |

Occ State enumeration: Occ = 0, Unocc=1, Bypass =2, Standby = 3, Null = 255.

* - extension block inputs (Note: extension block PVID# = Block PVID# - 9)

## Outputs

Table 165: Outputs of Temperature Set Point Calculator Function

| Output Name | Range | Description |
|---|---|---|
| EFF_HEAT_SETPT | Any floating-point number | Effective Heat Setpoint |
| EFF_COOL_SETPT | Any floating-point number | Effective Cool Setpoint |

## Operation:

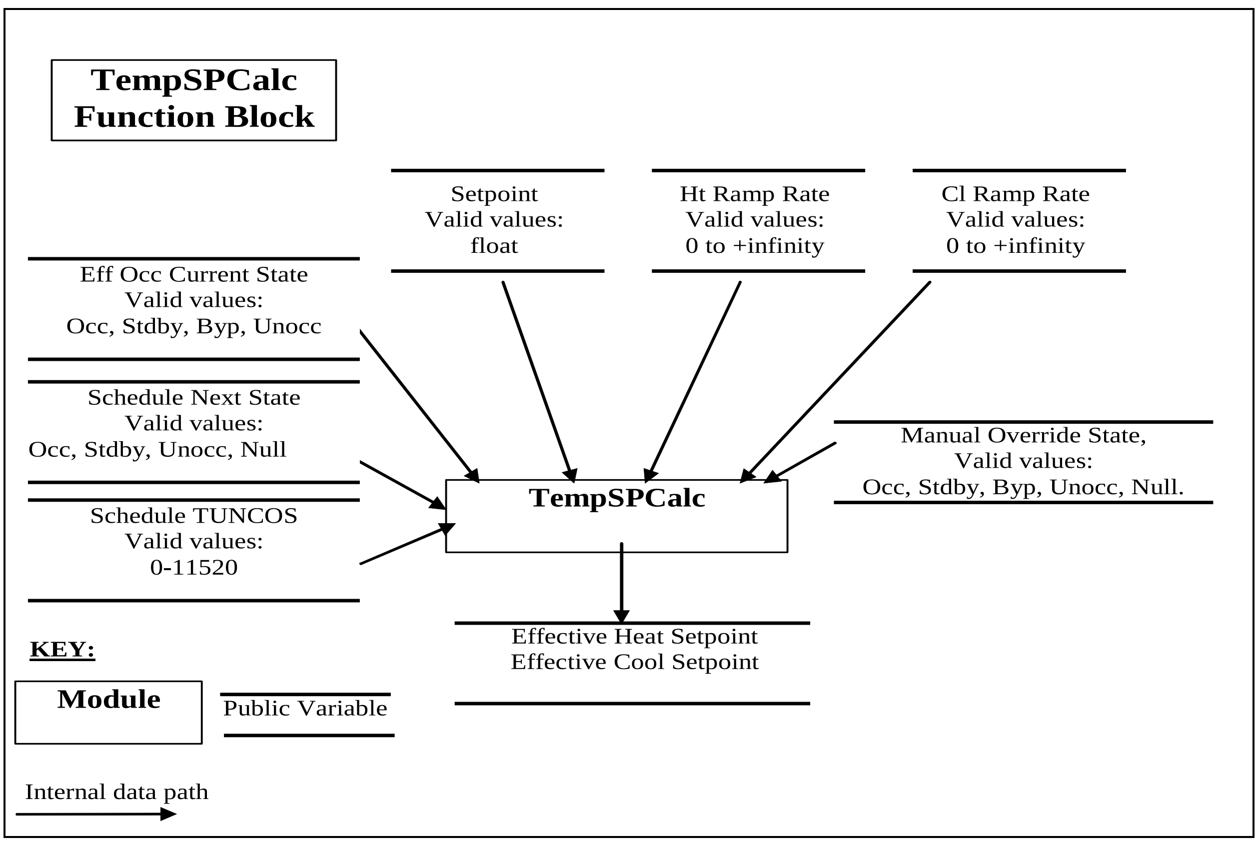



Figure 353: Temperature Setpoint Function Block

The Temperature Setpoint Calculator uses the 6 programmed-set-points, effective occupancy current state, scheduled next state and TUNCOS, center/offset setpoint, manual override state, recovery heat and cool ramp rates to determine the effective heat setpoint and effective cool setpoint.

The algorithm flow is:

1. Verify inputs are within range.
2. Compute the occupied and standby heat and cool setpoints based on the setpoint input and programmed setpoints.
3. If the effective occupancy current state is in unoccupied mode and not in manual override, calculate the recovery ramps.
4. If the effective occupancy current state is in occupied or bypass mode, use the occupied setpoints.
5. If the effective occupancy current state is in standby mode, use the standby setpoints.

## Programmed Set Points

The control block uses six temperature set-points. There are three set points of occupied, standby and unoccupied for heating and the same for cooling. All six can be changed from the Network via network variables. The Temperature Setpoint calculator doesn't place any restrictions on relationships between the set points and other inputs and the resulting calculations. This function block depends on the Tools writing TempSetpoints to enforce the range and relationship.

For reference, the LonMark Space Comfort Controller profile defines TempSetpoints as having a range of 10°C to 35°C with the following relationship unoccupiedHeat ≤ standbyHeat ≤ occupiedHeat ≤ occupiedCool ≤ standbyCool ≤ unoccupiedCool.

## Setpoint Input

This input allows the temperature setpoint for the occupied and standby mode to be changed via the wall module and/or network. This input can be either center or offset setpoint. If the input is less than 10, it is treated as offset setpoint. If the input is greater than or equal to 10, it is treated as center setpoint. Ensure the results are within the desired range. That is, it is possible to combine the setpoint input and the programmed heat and cool setpoints and get an effective setpoint outside of the unoccupied setpoints.

## Offset Setpoint

The setpoint acts in offset mode, that is relative setpoint, when the value on the Setpoint input is less than 10. The setpoint input adjusts the programmed occupied and standby heating and cooling set-points up and down by the amount on the input. You must ensure the input range is less than +10 for offset setpoint to be used. The setpoint input doesn't affect the unoccupied setpoints. During bypass, the occupied setpoints are adjusted. If the setpoint input is not connected or the sensor has failed, the offset is zero. You must ensure consistent units. That is, if the Setpoint input is in degrees F, the programmed setpoints should also be in degrees F.

- Occupied cool setpoint = programmed occupied cool setpoint + Setpoint input.
- Occupied heat setpoint = programmed occupied heat setpoint + Setpoint input.
- Standby cool setpoint = programmed standby cool setpoint + Setpoint input.
- Standby heat setpoint = programmed standby heat setpoint + Setpoint input.

### Center Setpoint

If the value on the Setpoint input is greater than or equal to 10, it is used as the center setpoint, that is absolute setpoint. If an invalid setpoint is on the Setpoint input, the programmed setpoints are used. The individual heat/cool setpoints for occupied and standby mode then derive from the Setpoint input minus/plus half the zero-energy bands calculated from the programmed setpoints.

For example:

zeb_occ = programmed occupied cool – programmed occupied heat

zeb_standby = programmed standby cool – programmed standby heat.

Occupied cool setpoint = setpoint + ½ zeb_occ

Occupied heat setpoint = setpoint - ½ zeb_occ

Standby cool setpoint = setpoint + ½ zeb_standby

Standby heat setpoint = setpoint - ½ zeb_standby

### Manual Override State

The Manual Override State is required to turn off recovery if in manual mode. If the Manual Override State is any value other than null then the algorithm doesn't know the scheduled next state and setpoint recovery is NOT done.

Note: Manual Override State doesn't affect the effective occupancy state. The OccArb function block already handles this. The effective setpoints never go to the state commanded by the Manual Override state input. Manual Override State just affects recovery as stated above.

### Effective Occupied State

This is used by the algorithm to determine the setpoints for the current occupancy state. When the Effective Occupancy Current state is occupied or bypass, use the occupied setpoints. When the Effective Occupancy Current state is standby, use the standby setpoints. When the Effective Occupancy Current state is unoccupied, recover the setpoint to the next state of occupied or standby. No recovery is done if in manual mode. See Adaptive Intelligent Recovery section.

### Heating and Cooling Ramp Rates

These are used by the adaptive recovery algorithm to recover the heating and cooling setpoints from their unoccupied values.

### Schedule Next State and TUNCOS

These are used by the adaptive recovery algorithm to recover the heating and cooling setpoints from their unoccupied values.

### Adaptive Intelligent Recovery

Set point recovery applies to setpoint changes associated with the following schedule state changes:

- Unoccupied to Standby
- Unoccupied to Occupied

Setpoint changes from occupied or standby to unoccupied state, changes from occupied to standby state, and changes from standby to occupied state use a step change in setpoint.

The heating or cooling recovery ramp begins before the next state transition time. During the recovery ramps, the heating and cooling set points are ramped from the unoccupied setpoint to the next state setpoint. The setpoint ramps is at the target setpoint 10 minutes prior to the occupied/standby event time.

This allows the HVAC equipment an extra 10 minutes to get the space temperature to the target setpoint during recovery.

| Note: |
| --- |
| Recovery is not done, if manual occupancy is in effect. |

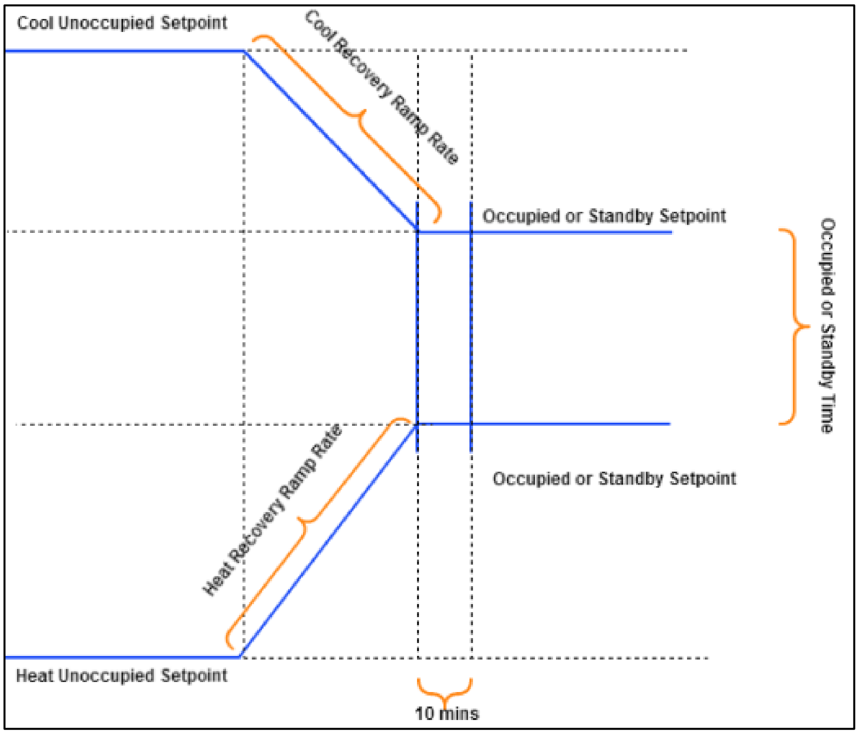



Figure 354: Heat and Cool Recovery Ramps

You can provide the heat and cool recovery ramp rates to the Temperature Setpoint Calculator. These can be constants, values calculated using the Ratio function block using outdoor air temperature, or some other method.

Heating and cooling recovery ramp rates can be any value greater than or equal to zero and have units of °/hr. A ramp rate of 0°/hr. means no recovery ramp for that mode. This means the setpoint steps from one setpoint to the other at the event time that is no extra 10 minutes. You must ensure consistent units. That is, the ramp rates should be in the same units as the setpoints.

| Note: |
| --- |
| If you program a rate of 1°/Hr. and have more than 192° spread between OCC and UNOCC set points, the algorithm is in recovery immediately when going to UNOCC. This is because the maximum TUNCOS is 11520 minutes times 1°/Hr. = 192° maximum delta. |

## TUNCOS Mesa

The controller implements the “TUNCOS Mesa” feature. This feature, also known as the "Smith Mesa" after Gary Smith implemented it in the T7300 series 1000. The TUNCOS Mesa is added to the algorithm to ensure the HVAC equipment gets the space temperature up to setpoint by the occupied time. The recovery algorithm subtracts 10 minutes from the TUNCOS and uses that to calculate the setpoint ramps.

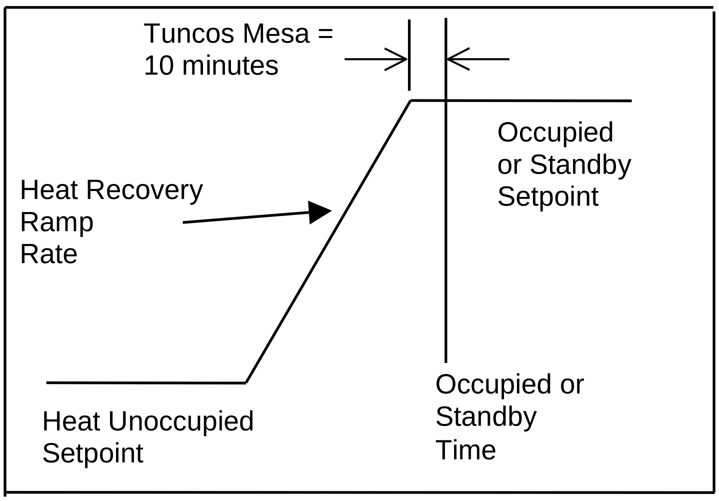



Figure 355: TUNCOS Mesa Heat Recovery Ramp (Cool is upside down)

## Effective Setpoint Limiting

This algorithm does not ensure that the effective cooling setpoint doesn’t go above the unoccupied cooling setpoint and the effective heating setpoint doesn’t go below the unoccupied heating setpoint. No check is made to ensure the effective heat and cool setpoints stay a minimum distance apart.

# BuiltIn

The CIPer Model 30 programming model provides the following BuiltIn blocks that can be configured and use to build the required application logic:

- ConventionalWallModule
- Utils Function Blocks

## ConventionalWallModule

The CIPer Model 30 programming model provides the ConventionalWallModule function blocks that you can configure and use to build the required application logic:

The CIPer Model 30 programming model supports configuring the conventional wall module (7-wire).

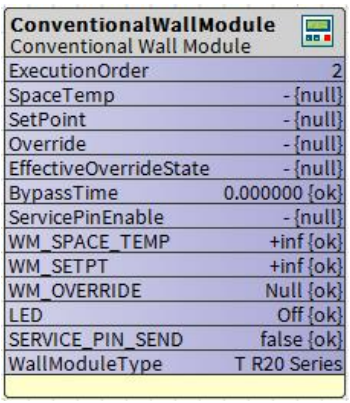



Figure 356: ConventionalWallModule Function Block

To configure the conventional wall module:

1. Expand the BuiltIn folder in the ipcProgrammingTool palette.
2. Drag the ConventionalWallModule onto the wire sheet. The conventional wall module block appears on the wire sheet.
3. Double-click the ConventionalWallModule block and the Properties Sheet is displayed. You can configure the properties for the conventional wall module in the Property Sheet.

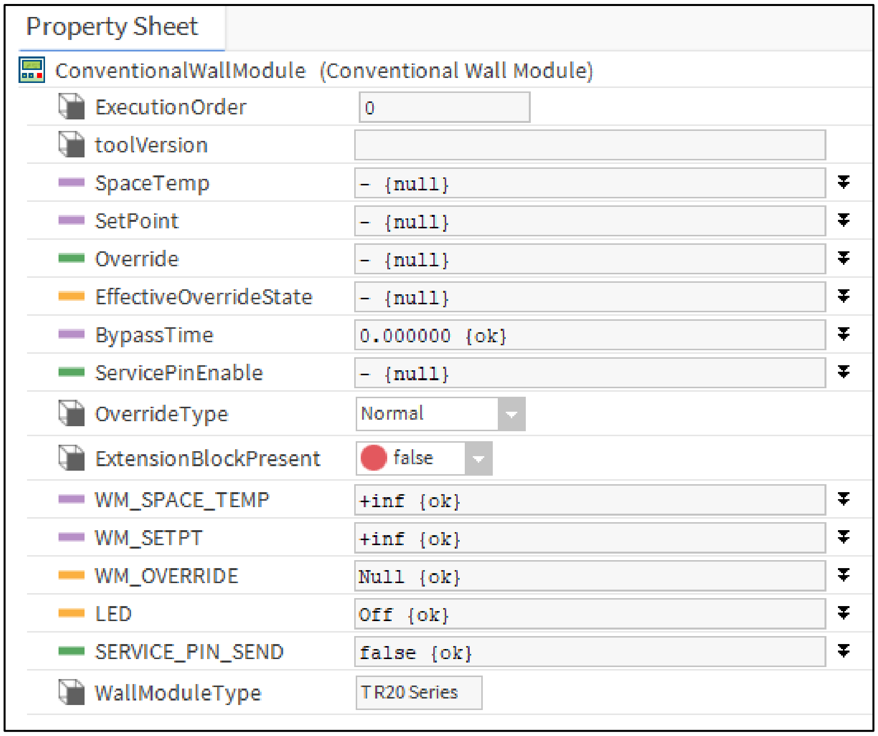



Figure 357: Property Sheet of ConventionalWallModule

## Override Type

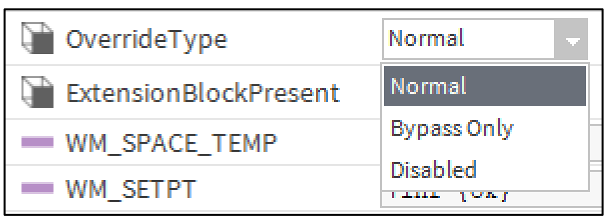



Figure 358: Override Options

You can select one of three options available:

- NORMAL_OVERRIDE: If this option is selected, you can override the system in Unoccupied and Bypass Mode using Override button. If analog output is connected to the LED input of the wall module, it gives feedback of the overridden state.

  To override the system in Unoccupied Mode, press the Override button till LED starts blinking. To remove overridden state, press the Override button till LED turns off.

  To override the System in Bypass Mode, press the override button till LED turns ON. To remove overridden state, wait till bypass time (see Bypass Time in the same table) expires or press the Override button and wait till LED turns OFF.

- BYAPSS_ONLY_OVERRIDE: If this option is selected, you can override the system in only Bypass Mode.

  To override the System in Bypass Mode, press the override button till LED turns ON. To remove overridden state, wait till bypass time (see Bypass Time in the same table) expires or press the override button and wait till LED turns OFF.

- OVERRIDE_DISABLE: You cannot override the system in any mode. Override button has no effect.

| | Note: |
|---|---|

If the user connects the sensors or hardware inputs directly to the conventional wall module then the wall module will automatically calibrate the output based on the current data. But if the user connects the parameters via passthrough or other blocks then user need to do manual commissioning to reflect the changes in the output.

To do Manual commissioning select the Conventional Wall Module on the wiresheet -> right click -> select Commission Wall Module. Also, if user changes any configuration or connection of the Conventional Wall Module then the user need to do manual commissioning to reflect the changes.

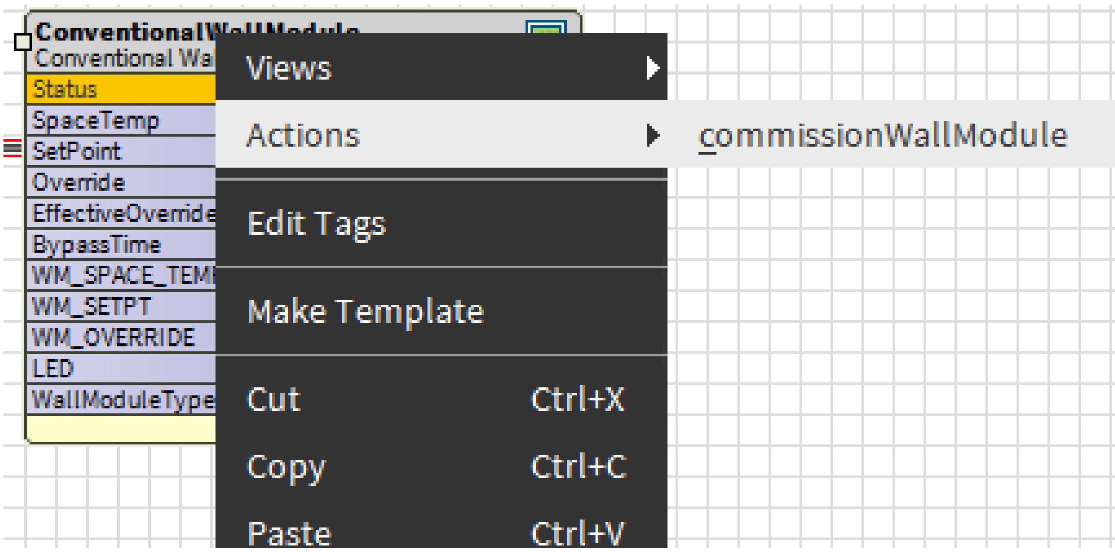



Figure 359: Manual Commissioning of Conventional wall Module

# Utils Function Blocks

The utility function block provides the PassThru, TextBlock, SystemTime, and Tuncos application blocks for programming.

## PassThru

This object is used to provide an input and output slot to the application block so that inputs and outputs can be connected to that application block.

Example: A logic is created with application block as shown in the following figure. It calculates the averages two temperature inputs and transfers the average value to the network outputs. As shown at the right-side area of the following figure, the application block does not have slots. The PassThru block helps creating these slots and then you can connect inputs and outputs to the slots created to the application block.

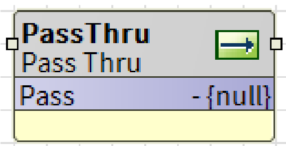



Figure 360: PassThru Function Block

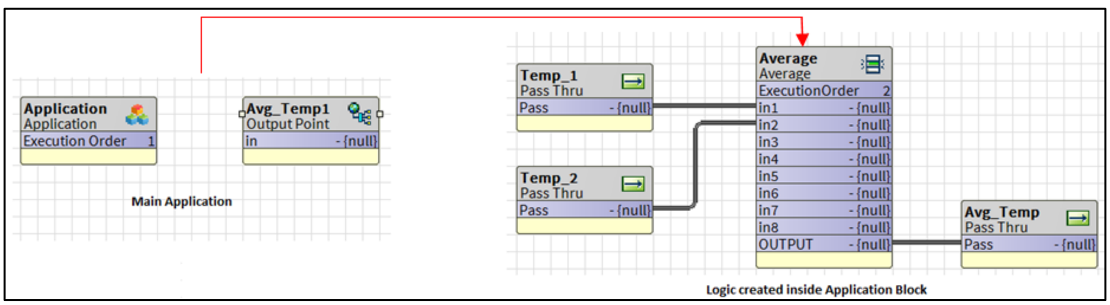



Figure 361: PassThru Example

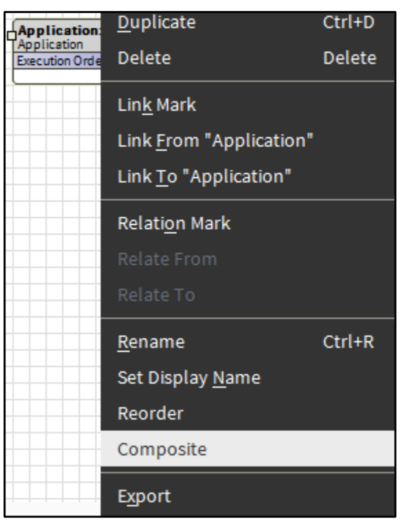



Figure 362: Navigating to Composite

To create slots:

1. Right-click the application block and select Composite. The Composite window is displayed. At the right pane of the window all points of which slots can be created appears.
2. Create slots as shown in the following figure and click Ok. Input and output slots are created to the Application block.
3. Connect inputs to input slots and output to the output slots. In this way, data is passed to and from Application logic through pass through object. Input and output slots can be created to function blocks also.
4. In the following example, Average function block is used. Its inputs and outputs can be exposed to Application block.

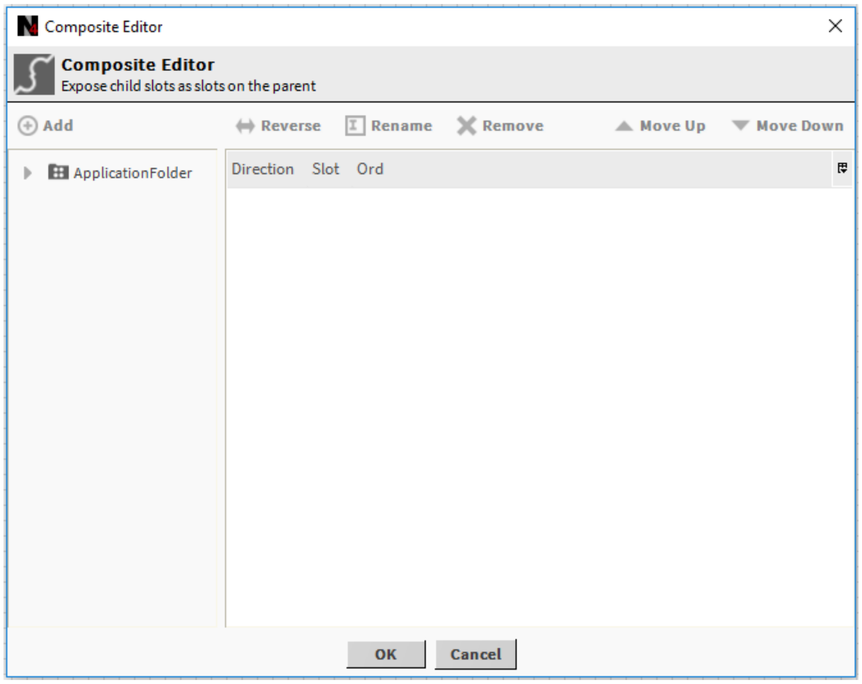



Figure 363: Composite Editor Window

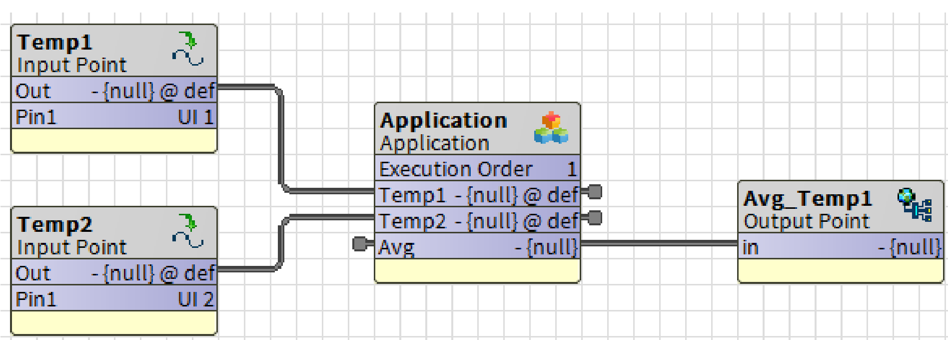



Figure 364: Wire Sheet View of Composite

## TextBlock

The TextBlock in the utility module provides you the ability to add the text boxes onto the wire sheet. You can add the text of your choice.

To use TextBlock:

1. Drag and drop the TextBlock from palette pane onto the wire sheet.
2. Double-click the TextBlock. The property Sheet is displayed.
3. Enter the text in the Text field.
    a. Set the Foreground and Background color properties.
    b. Select the font and font size from the respective drop-down menus.
    c. Apply the bold, italic, underline styles.
    d. Select the border and selectable property values from respective drop-down menus.
4. Click Save to save the changes.

Following figure shows the Property Sheet view of the TextBlock.

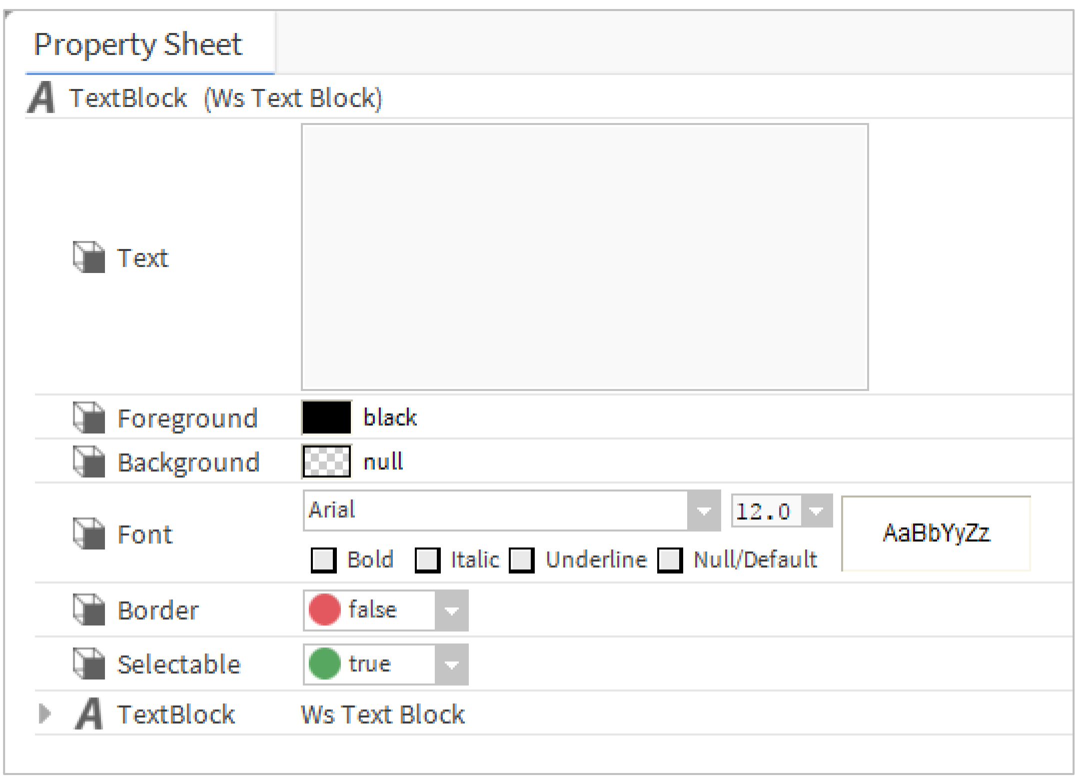



Figure 365: Property Sheet View of TextBlock

## SystemTime

When you drag and drop the SystemTime block onto the wire sheet, the local time value which is configured in the CIPer Model 30 is displayed. You can configure the value of SystemTime block by configuring the Property Sheet of the SystemTime block or in the Platform Administration. The changes made at any one of these two places reflect at the other place.

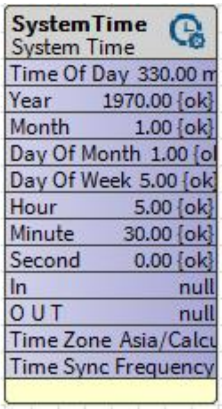



Figure 366: SystemTime Block

## Tuncos

The time until next change of state (TUNCOS) represents the time taken by the system to change from one or current state to another or next state. You cannot use the Tuncos block separately and must use it with SylkSchedule block. By default, the Tuncos parameter is provided in the SylkSchedule block.

To add the Tuncos function block onto the wire sheet:

1. Open the wire sheet of the SequencedControlProgram folder by navigating to Station > Config > Drivers > IPCNetwork > LocalDevice > Points > SequencedControlProgram.
2. Navigate to the Tuncos block in the Utils function block of ipcProgrammingTool palette.
3. Drag and drop the Tuncos block onto the wire sheet. The Tuncos block is displayed on the wire sheet.

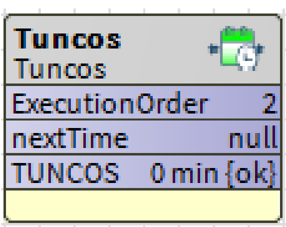



Figure 367: Tuncos Block

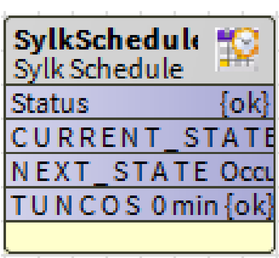



Figure 368: SylkSchedule Block with TUNCOS Parameter

# Custom Palette File

You can create and use a custom palette file to store any CIPer Model 30 object application, macro, device, function blocks, and IOs from a station. The custom palette file can be used to share across stations and among multiple users. This file acts as a repository, but an object cannot be configured that exists in the palette. You can later copy and paste or drag and drop these objects from the custom palette to the station.

## Creating Custom Palette File

To create a custom palette file:

1. On the Nav sidebar, navigate to the drive, where the custom palette file needs to be created.
2. Right-click the drive and click New > New Folder. A new folder is created.

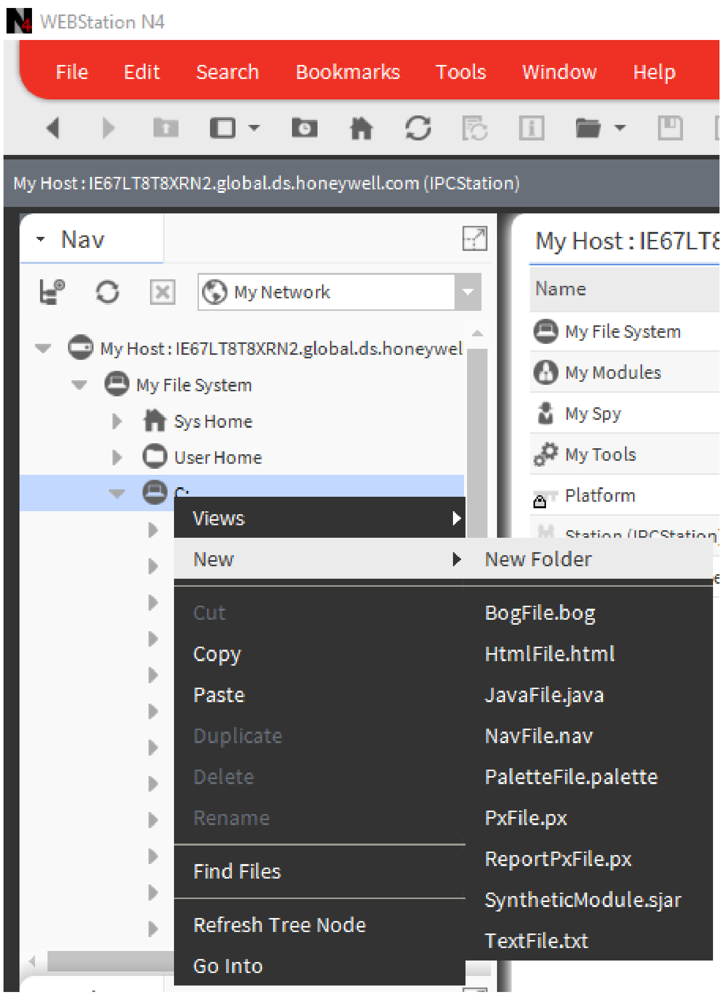



Figure 369: Creating New Folder

3. Enter a name for the new folder and click Ok.
4. Right-click the newly created folder and click New > PaletteFile.palette.

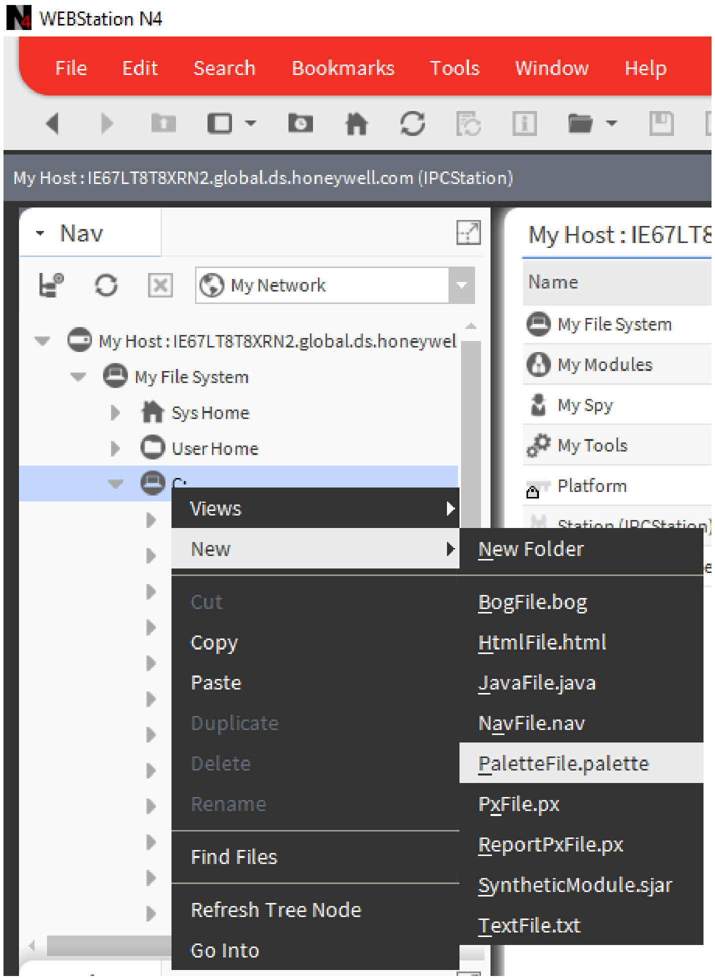



Figure 370: Creating New Palette File

5. Enter a name for the palette file and click Ok. A new palette file is created.
6. Expand the newly created folder to view the palette file that is created.
7. Double-click the palette file to open its wire sheet.
8. On the Palette pane, click the  (Open Palette) icon. The Open Palette window is displayed.

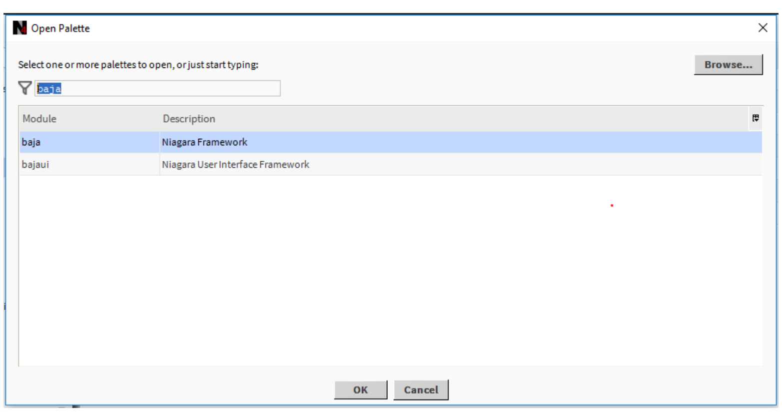



Figure 371: Select Baja Module

9. Select Baja module and click Ok. The UnrestrictedFolder folder is available in the Baja Palette (Palette pane with Baja module selected).

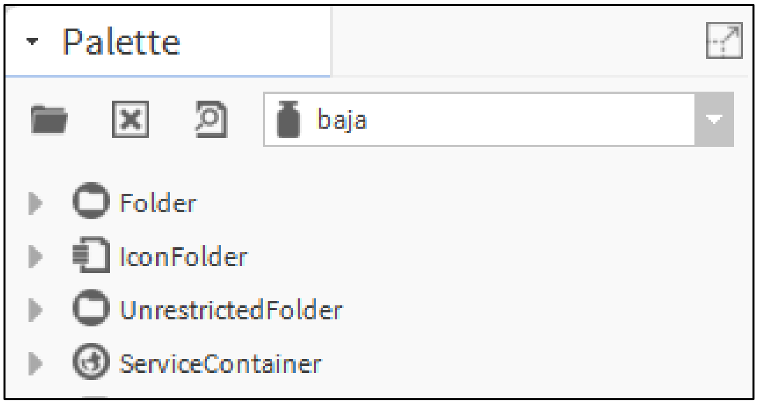



Figure 372: Unrestricted Folder in Baja Module

10. Drag the UnrestrictedFolder into the folder with the palette file that is created. A .bog file is displayed in the folder which contains the UnrestrictedFolder.
11. Double-click the UnrestrictedFolder.bog file that is added to the newly created folder, to open its wire sheet.

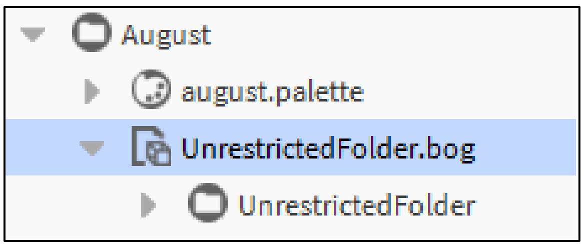



Figure 373: Unrestricted Folder Structure

12. Right-click the folder to rename it. This is the Unrestricted folder, where all CIPer Model 30 objects can be stored.

**Note:**

You can double-click the UnrestrictedFolder in the Nav tree to open its wire sheet and drag the UnrestrictedFolder object from the Palette onto the wire sheet. This has the effect of nesting folders within the palette file. This enables the categorization of objects that are stored in the palette file.

For example, an UnrestrictedFolder can be dragged from the Baja palette (Palette with Baja module selected) onto the wire sheet of the palette file in the Nav tree and name it Applications. Then double-click the Applications folder on the wire sheet, drag another UnrestrictedFolder object from the Baja palette, and name it VAV Applications. This creates the VAV Applications folder nested under the Applications folder in a tree structure in the custom palette file folder that is created.

# Adding Items to Custom Palette File

To add any CIPer Model 30 object such as an IO or function block to the custom palette:

1. Browse to the Sequence Control Program that needs to be saved in the custom palette file by clicking Station > Drivers > IPCNetwork > LocalDevice > Points > SequencedControlProgram.

   Or

2. Station > Drivers > IPCNetwork > LocalDevice > Points > EventControlProgram in the Nav tree.
3. Right-click any object such as a function block or IO and select Copy.

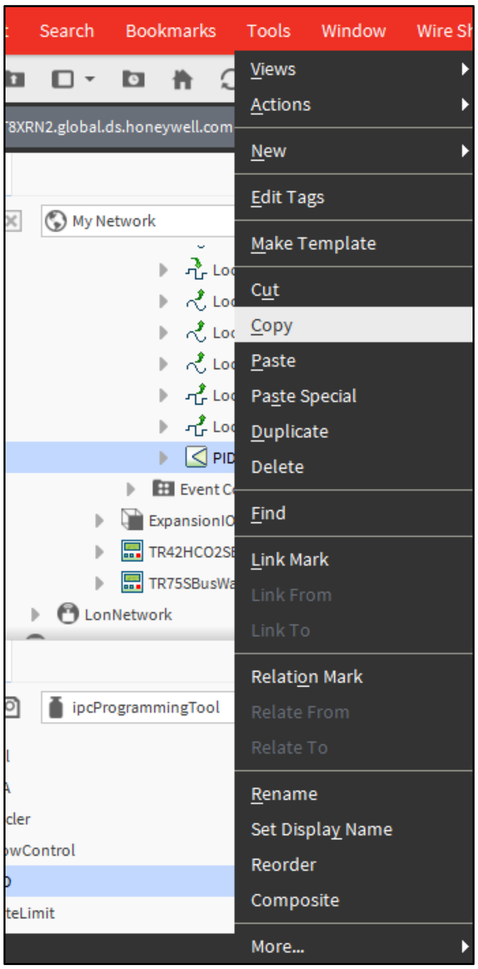



Figure 374: Copy Option

4. Navigate to the folder that is created under the custom palette file (Applications or VAV Applications as given in the above Note) and right-click it and select Paste.

   Or

   Drag and drop the object to the wire sheet of the folder (Applications or VAV Applications as given in the above Note) under the custom palette file.

   Or

Drag and drop a CIPer Model 30 object directly onto the folder (Applications or VAV Applications as given in the above Note) under the custom palette file in the Nav tree.

5. The object is saved under the folder in the custom palette file.
6. Right-click the file in the custom palette file and click Save.
7. Right-click the custom palette file and click Close to close the custom palette file.

# Closing Custom Palette File

To close the custom palette file, right-click the custom palette file, and click Close.

| Note: |
|---|
| If a custom palette file or the workbench is closed without saving the contents of the custom palette file, the newly added contents are not saved and are not be available next time. |

The components can be reused from the custom palette file in any application logic that is created by dragging and dropping the desired object from the custom palette file to the wire sheet of SequencedControlProgram or EventControlProgram as required.

## Adding Device to Custom Palette File

Adding a device to the custom palette file is like adding an IPC object, but it has some specific steps that you need to perform additionally.

To add a device to the custom palette file:

1. Browse to the device that needs to be saved in the custom palette file by clicking Station > Drivers > IPCNetwork > LocalDevice.
2. Right-click the device and select Copy.
3. Navigate to the folder that is created under the custom palette file (Applications or VAV Applications as given in the note under Creating Custom Palette File section), right-click it, and select Paste.
4. Right-click the custom palette file and select Save. The device is saved under the folder in the custom palette file.

**Honeywell Building Technologies**
Honeywell International Inc.
1985 Douglas Drive North
Golden Valley, MN 55422
customer.honeywell.com


31-00237-03 | Rev 06-2021