Honeywell

Niagara IRM

FUNCTION BLOCKS USER GUIDE

EN2B-0415GE51-R0423

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.

Sylk™ is a trademark of Honeywell International Inc.

TRADEMARKS

CentraLine is a trademark of Honeywell Inc.

BACnet and ASHRAE are registered trademarks of American Society of Heating, Refrigerating and Air-Conditioning Engineers. Microsoft and Windows are registered trademarks, and Windows Internet Explorer are trademarks of Microsoft Corporation. Java and other Java-based names are trademarks of Sun Microsystems Inc. and refer to Sun's family of Java-branded technologies. Mozilla and Firefox are trademarks of the Mozilla Foundation. Echelon, LON, LonMark, LonTalk, and LonWorks are registered trademarks of Echelon Corporation.

Tridium, JACE, Niagara Framework, NiagaraAX Framework, Sedona Framework and Vykon are registered trademarks, and Workbench, WorkPlaceAX, and AXSupervisor, are trademarks of Tridium Inc. All other product names and services mentioned in this publication that is known to be trademarks, registered trademarks, or service marks are the property of their respective owners.

DOCUMENT HISTORY

Version	Published	Update
01-18	Initial Release	New release
02-21	Whole Document	Updated table colour and figure caption colour.
04-21	Added New Function Blocks	Added UIO, IRMWs Text Block FB and Updated Save Permanent FB.
06-21	Updated few Function Blocks	Updated Compare, EnumSchedule, Floating, FlowSensor, CurrentDateTime and SystemCommand Function Blocks.
08-21	Updated Function Block	Updated System Command Function Block.
01-22	Updated Function Block	Updated Save Permanent Function Block.
02-22	Updated Function Block	Updated Physical Points Function Block, Actuator Position Feedback, PidA, Support for TR120 device, Sylk Actuator Input, Sylk Actuator Output and Out Save feature for all the function blocks.
03-22	Updated Function Block	Updated PWN function block.
08-22	Updated Function Block	Updated BACnet Numeric value, BACnet Boolean value, BACnet Enum value, Floating Function Block, and Stage123 Outp Function Block.
09-22	Updated Function Block	Updated the description of wired wall module parameters.
10-22	Updated Function Block	Updated the description of Stage Driver function block.

TABLE OF CONTENT

About IRM Function Blocks 20

Applicable Technical Literature **20**

Abbreviation **22**

General Description **23**

About honIrmControl **26**

Physical Points Function Blocks 29

BO Terminal **30**

Input 30

Output 30

Parameter 30

BI Terminal **32**

Input 32

Output 32

Parameter 33

UI Terminal **34**

Input 35

Output 35

Parameter 36

AO Terminal **40**

Input 40

Output 40

Parameter 40

UIO Terminal **42**

Input 42

Output 44
Parameter 45
Service Pin Terminal 48
Output 48
Flow Sensor 49
Output 49
Parameter 50
Actuator Position Feedback 51
Input 51
Output 51
Parameter 51
Onboard IO Package 52
Onboard IO IRM CVAV Unitary Packages 52
Onboard IO IRM CVAV VAV Package 53
Onboard IO IRM IP VAV Packages 54

Bacnet Objects Function Blocks.. 55

Bacnet Numeric Input 56
Input 56
Output 56
Parameter 57
Bacnet Boolean Input 60
Input 60
Output 60
Parameter 62
Bacnet Numeric Output 64
Input 64
Output 65
Parameter 65
Bacnet Boolean Output 69
Input 69
Output 70
Parameter 71
Bacnet Enum Output 73

Input 73
Output 73
Parameter 74
Bacnet Numeric Value 76
Inputs 76
Output 77
Parameter 78
Bacnet Boolean Value 80
Input 80
Output 81
Parameter 82
Bacnet Enum Value 84
Input 84
Output 85
Parameter 86
Ref In 88
Input 88
Output 88
Parameter 89
Ref Out 91
Input 91
Output 91
Parameter 91
Arithmetic Function Blocks 93
Add 94
Input 95
Output 95
Parameter 95
Aggregation 97
Input 98
Output 98
Parameter 98
Divide 100

Input 100
Output 100
Parameter 101
Psychrometric 102
Input 102
Output 103
Parameters 103
Exponential 105
Input 105
Output 106
Parameter 106
Limit 107
Input 108
Output 108
Parameter 109
Linear Graph 111
Input 112
Output 112
Parameters 112
Math Operation 117
Input 117
Output 117
Parameter 118
Multiply 120
Input 120
Output 120
Parameter 121
Negative 122
Inputs 122
Outputs 122
Parameter 123
Reset 124
Input 126
Output 126

Parameter 126
Subtract 131
Input 131
Output 131
Parameter 132
Digital Filter 133
Input 133
Output 134
Parameter 134
Flow Velocity 135
Input 135
Output 136
Parameter 136
Encode 138
Input 139
Output 139
Parameter 140

Bit Functions Function Blocks 144

Bit And 145
Input 146
Output 146
Parameter 146
Bit Or 149
Input 149
Output 150
Parameter 150
Numeric To Bit 153
Input 153
Output 153
Parameter 154

Control Loop Function Blocks 156

Aia 157

Input 158
Output 159
Parameter 159
PidA 161
Input 163
Output 166
Parameter 166
Pid 169
Input 171
Output 171
Parameter 172
Stager 175
Input 175
Output 176
Parameter 176
Stage Driver 180
Input 181
Output 181
Parameter 182
Flow Control 184
Input 185
Output 185
Parameter 185
Cycler 189
Input 190
Output 190
Parameter 191
Conversion Function Block........ 196
Binary To Numeric 196
Input 197
Output 197
Parameter 198

Comparison Function Blocks 199

Compare **200**
Input 201
Output 201
Parameter 201
Equal Null **205**
Input 205
Output 205
Parameter 205
Greater Than **207**
Input 207
Output 208
Parameter 208
Greater Than Equal **209**
Input 209
Output 210
Parameter 210
Less Than **211**
Input 211
Output 211
Parameter 212
Less Than Equal **213**
Input 213
Output 213
Parameter 214

Select Switch Function Blocks 215

Binary Select **216**
Input 216
Output 217
Parameter 217
Binary Select Prio **218**
Input 219

Output 219
Parameter 219
Binary Select Multi 221
Input 221
Output 222
Parameter 222
Binary Select Multi Prio 224
Input 225
Output 226
Parameter 226
Change Select 228
Input 229
Output 229
Parameter 230
Max Select Multi 231
Input 232
Output 232
Parameter 233
Min Select Multi 235
Input 236
Output 236
Parameter 237
Numeric Select 239
Input 240
Output 241
Parameter 241
Valid Select Prio 243
Input 244
Output 244
Parameter 244
Valid Select Multi Prio 246
Input 247
Output 248
Parameter 248

Numeric Switch **250**
Input 251
Output 251
Parameter 251

Logic Function Blocks 253

And **254**
Input 255
Output 255
Parameter 255
Or **257**
Input 258
Output 258
Parameter 258
Xor **260**
Input 261
Output 261
Parameter 261
Not **263**
Input 263
Output 263
Parameter 263
Rs Flip Flop **265**
Input 265
Output 266
Parameter 266
Sr Flip Flop **268**
Input 269
Output 269
Parameter 269
Trigger **271**
Input 271
Output 271
Parameter 272

Timer Function Blocks 275

One Shot **276**
- Input 277
- Output 277
- Parameter 277

Rate Limit **278**
- Input 279
- Output 280
- Parameter 280

Time Ramp **282**
- Input 283
- Output 284
- Parameter 284

Time Delay **286**
- Input 286
- Output 286
- Parameter 287

Multi function Timer **289**
- Input 289
- Output 290
- Parameter 290

Runtime Accumulate **293**
- Input 293
- Output 294
- Parameter 294

Date and Time Function Blocks 296

Enum Schedule **297**
- Output 297
- Parameter 298
- Schedule Configuration 299

Configuring Schedule Event with Supported Device 300
Configuring Schedule Block Location 301
Configure Enum Range 302
Define Enum range 302
Add Entry in Enum library 303
Modify Enum library 304
Additional Configuration 304
Special Events 306
Creating Event for Multiple Days 308
Default Output 308
Calendar 309
Parameters 309
Current Date Time 311
Input 311
Output 311
Parameters 312
Date And Time Operation 313
Input 313
Output 314
Parameters 314
Util Function Blocks 316
Error 317
Input 317
Parameter 318
Pass Thru 319
Input 319
Output 319
Prev Value 320
Input 320
Output 320
Parameter 321
Const1 Numeric 322
Output 322

Parameter 322
Const2 Numeric 323
Output 323
Parameter 323
Const5 Numeric 325
Output 325
Parameter 325
Const1 Boolean 327
Output 327
Parameter 327
Const2 Boolean 328
Output 328
Parameter 328
Const5 Boolean 329
Outputs 329
Parameter 329
Save Permanent 331
Input 332
Output 333
Parameter 333
Evaluate Bacnet Status Flags 336
Input 336
Output 336
Parameter 337
Text A 339
System A 340
Input 340
Output 340
Parameter 340
IrmWsTextBlock Function Block 343
IrmWsTextBlock 343
Input 343

Outputs Function Blocks ... 345

Floating ... 346
Input ... 347
Output ... 350
Parameter ... 351
Pwm ... 359
Input ... 360
Output ... 360
Parameter ... 361
Stg123 Outp ... 363
Input ... 366
Output ... 367
Parameter ... 367

Light Function Block ... 371

Light A ... 372
Input ... 374
Output ... 381
Parameter ... 383

Sunblind Function Block ... 400

Blind A ... 401
Input ... 407
Output ... 413
Parameters ... 414
Push Button Behavior ... 418

Wall module Function Block ... 421

Wm Config Hvac A ... 421
Input ... 423
Output ... 444
Tr40 Model ... 454
Parameter – General ... 454

Relative Setpoint 473
Absolute Setpoint 473

Modbus Function Block 476

Modbus Device Settings 477

Modbus Device 477

Output 478
Parameter 478

Modbus Configuration Screen 479

Modbus Read Point 480

Input 480
Output 481
Parameter 482
Add Modbus Read Point 484

Modbus Write Point 486

Input 486
Output 487
Parameter 488
Add Modbus Write Point 489

Data Function Block 491

Alarm 492

Inputs 493
Output 493
Parameter 494

Counter 495

Inputs 496
Output 496
Parameters 496

Zone Arbitration Function Blocks 499

General SetPoint Calculator 500

Input 501
Output 502

Parameter 502
Occupancy Arbitrator 506
Input 506
Output 507
Parameter 508
Set Temperature Mode 512
Input 513
Output 514
Parameter 515
Temperature Setpoint Calculator 520
Input 521
Output 523
Parameter 523

Sylk Device and Parameters 527

Sylk Parameters Description 528
Sylk In Parameters 528
Sylk Out Parameters 528
Sylk In Out Parameters 528
Sylk Device Supported Sylk Parameters 529
Sylk Component Status Behaviors 532

Sylk In Parameters 533

Controller Value 533
Inputs 534
Output 534
Parameters 534
Occupancy Status 537
Inputs 537
Output 537
Parameters 537
System Status 539
Inputs 539
Output 539

Parameters 539
Time of Day 541
Inputs 541
Output 541
Parameters 541
Sylk Actuator Input 543
Inputs 543
Output 544
Parameters 544

Sylk Out Parameters 546

CO2 546
Inputs 547
Output 547
Parameters 547
Humidity 549
Parameters 549
Occupancy Override Command 551
Inputs 551
Output 551
Parameters 552
ROOMTEMP 558
Inputs 558
Output 558
Parameters 559
Sensor Offset 561
Inputs 561
Output 561
Parameters 561
Value From Wall Module 564
Inputs 564
Output 564
Parameters 565
Sylk Actuator Output 568

Inputs 568
Output 568
Parameters 568

Sylk In Out Parameters...... 571

Bypass Time 571
Inputs 572
Output 572
Parameters 572
Fan Command 574
Inputs 574
Output 574
Parameters 575
Network Setpoint 577
Inputs 577
Output 577
Parameters 577
System Command 582
Inputs 583
Output 583
Parameters 583
Time Field 587
Inputs 587
Output 587
Parameters 587
Sylk Time 589
Input 589
Output 589
Parameter 589
Home Screen 590

CHAPTER

1 ABOUT IRM FUNCTION BLOCKS

This document is intended to serve as a guide for using the IRM function blocks. These function blocks are part of the honIrmControl Palette and are used to create a custom application logic for various HVAC applications.

The function block has inputs and outputs. The function block receives inputs from physical inputs, network inputs, or outputs of other blocks and then processes the input data to produce output. The processing is determined by the function block type.

Applicable Technical Literature

Table 1 Applicable Technical Literature

Document Title	Document Number
Merlin NX IP VAV Product Datasheet	EN0Z-1073GE51
Merlin NX MSTP VAV Product Datasheet	EN0Z-1072GE51
Merlin NX IP and MSTP VAV Installation Instructions	EN1Z-1076GE51
Merlin NX IP and MSTP VAV Mounting Instructions	EN1Z-1074GE51
Global VAV Balancing Tool User Guide	EN2Z-1086IE67
Honeywell Connect Mobile User Guide	31-00472
Merlin NX Compact VAV Product Datasheet	EN0Z-1061GE51
Merlin NX Compact VAV Installation Instructions	EN1Z-1061GE51
Merlin NX Compact VAV Mounting Instructions	MU1Z-1061GE51
IRM Engineering Guide	EN2B-0414GE51
Honeywell Unitary Controller 24V - Datasheet	31-00571
Honeywell Unitary Controller 24V - Mounting Instructions	31-00572
Honeywell Unitary Controller 24V - Installation Instructions	31-00614
IP VAV Product Datasheet	EN0B-0301-IE10
MSTP VAV Product Datasheet	EN0B-0300-IE10
IP and MSTP VAV Installation Instructions	EN1B-0301-IE10

Table 1 Applicable Technical Literature

Document Title	Document Number
IP and MSTP VAV Mounting Instructions	MU1B-0300-IE10
IRM Migration Guide	EN2B-0428-IE67

Abbreviation

A **AI:** Analog Input **AIA:** Adaptive Integral Action **AO:** Analog Output **AV:** Analog Value	N **nan:** Not a Number
B **BI:** Binary Input **BO:** Binary Output **BTU/LB:** British Thermal Units per pound **BV:** Binary Value	P **Par:** Parameter **PID:** Proportional Integral and Derivative
C **CV:** Controlled Value	S **Sec:** Second **SP:** Setpoint
D **DI:** Digital Input **DDC:** Direct Digital Control	U **UI:** Universal Input
F **FB:** Function Block	W **W:** Watt **WM:** Wallmodule
I **+Inf:** Infinitely large number **-Inf:** Infinitely small number **In:** Input **Inv:** Invalid **I/O:** Input Output	
M **Max:** Maximum **Min:** Minimum **MI:** Multistate Input **MO:** Multistate Output **MV:** Multistate Value	

General Description

Enable Input

Many function blocks have an Enable input. "Enabled" indicates that the function block will be executed. "Disabled" means that the function block or some of its features are disabled. If the input is not connected, its value is null, which means "Enabled." The underlined value is the default value.

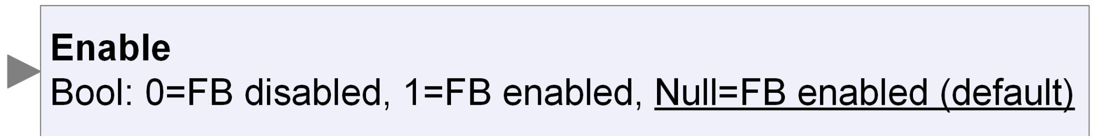


Fig. 1 Enable Default Value

Output Function block with Interlock Input and Virtual Position Output

When set to 1, the interlock input enables the block's output, and when set to 0, the block's output is closed (0 %) or off (0 %).

Example: When the fan is turned on, the electrical reheat is activated.

If logic is required to check whether the output is open or running (valve or fan open or running), always use the virtual output because the output model and all times are represented here. A BACnet override and a valve exercise are also covered.

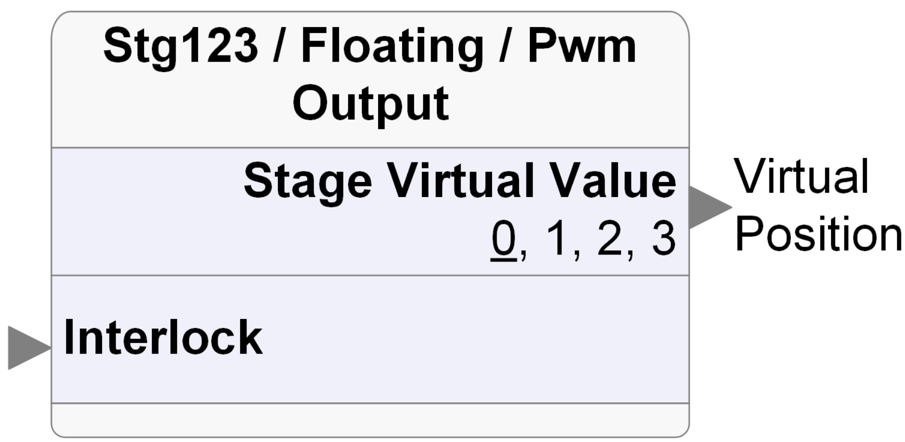


Fig. 2 Output Function blocks with Interlock Input and Virtual Position Output

Null Value

The "null" value can be set and checked just like any other value in the Logic. All unconnected inputs are set to "null."

Non-connected Input and Output

An unconnected Input always has the value "null." An Output has no default value. The value is calculated based on the inputs. Each parameter has a default value, which is highlighted in the images. The default can also be a "null" value.

Periodic and Event Program Folder

There is a "Periodic program" folder with a cycle time of 500 ms, as well as an "Event program" folder. Hardware inputs are typically connected to the Event program folder. The event program folder is executed every 1000 ms or whenever an input changes. Points can be transferred between folders.

BACnet and Physical Output Principle

Physical outputs added to the controller do not automatically generate the BACnet points. The output is first written to a BACnet function block. The value can be overwritten or set to Out-Of-Service using BACnet. The BACnet function block's output is connected to the physical output.

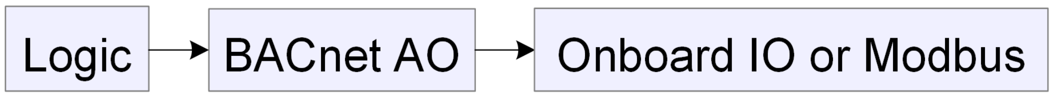


Fig. 3 BACnet and Physical Output Principle

Failure Handling

In the event of a known error in the function block, log the error along with the Execution Order of the corresponding Function block (that is, when dividing by 0).

The function block "Error" field displays the most recent error on the current wiresheet. There should be no mistakes at the end of the programming. Every Wire sheet should have a "Error" function as the last function block.

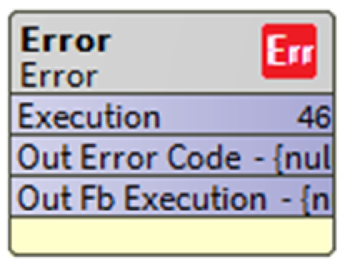


Fig. 4 Failure Handling

Mix of Different Data Types

For easy programming, all data types can be connected to the inputs and outputs. False is equivalent to 0, and True is equivalent to 1.

Negate Input and Output

For some function blocks, the inputs and outputs in the function block can be negated to improve legibility and avoid additional negator function blocks.

Data Types

Enumerations in BACnet begin with the number 1, and it is recommended that all enums begin with the same number.

Input, Output, and Parameter Slots

The output slots are at the top of the function block, and the inputs are at the bottom. This improves the arrangement of the function block on the wire sheet and corresponds to the "naming conventions of AX developments." The parameters are only visible when the "Property Sheet" is opened. Outputs require a relatively large amount of memory, namely 6 bytes of RAM.

Input has higher Priority than Parameter

There are parameters marked with (In & Par). In addition to this parameter, a corresponding input slot has the same name but without "Par" at the end since the names must be unique. If the input is not connected or the input has the value "null," then the parameter is used. The input slot, therefore, has priority over the parameter. The input slot is described in detail in the documentation, while the parameter shows the default value.

About honIrmControl

The honIrmControl module contains the library of IRM control components such as function blocks, Sylk modules, and BACnet device components for programming the applications.

Steps to open honIrmControl:

Step 1. Open the **Palette** and click on the folder icon.

Step 2. On the search field, enter honIrmControl, and click **OK.** This adds the honIrmControl module to the palette.

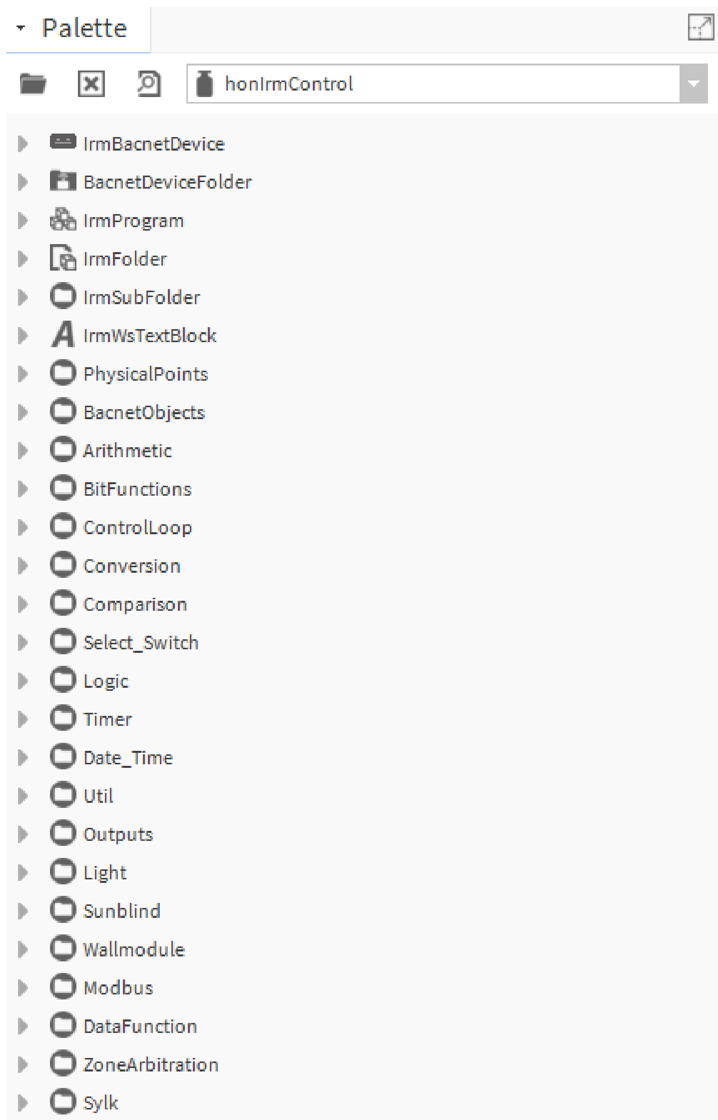


Fig. 5 honIrmControl Palette

The honIrmControl palette includes the following function blocks.

Table 2 Function blocks Description

Function Block	Description
Physical Points Function Blocks	Physical Points are logical objects that are used to build application logic. Depending on the model selected, default (fixed) physical points for that model are made available.
Bacnet Objects Function Blocks	A Bacnet Object is a data item such as temperature, switch value, or actuator state. The Bacnet objects represented by point parameters.
Arithmetic Function Blocks	Arithmetic function blocks are math components. Each component type provides a specific math function, such as Add, Average, Divide, Minimum, Maximum, Reset, AbsValue, and so on.
Bit Functions Function Blocks	Bit Functions blocks are used to configure and build the required application using bit components.
Control Loop Function Blocks	The Control Loop function block is used to configure and build the required application logic using control components.
Conversion Function Block	The Conversion function blocks is used to assign three different numeric values to a numeric output based on a binary input.
Comparison Function Blocks	The Comparison function block is used to configure and build the required application logic using comparison components.
Select Switch Function Blocks	The Select Switch function block is used to configure and build the required application logic using switch components.
Logic Function Blocks	The Logic function block is used to configure and build the required application using various like AND, OR, XOR, and NOT logic using switch components.
Timer Function Blocks	The Time function block is used to configure and build the required application logic using time components.
Util Function Blocks	The Util function block is used to configure and build the required application logic using util components.
IrmWsTextBlock Function Block	It allows you to customize the graphics to add better text in the application.
Outputs Function Blocks	The Outputs function block is used to configure and build the required application logic using output components.
Light Function Block	Light A function block is used to control lighting for On or Off, dimming, or stairwell applications.
Sunblind Function Block	The Blind A function block is used to control a sunblind, roller shutter or an electrically driven window.
Wall module Function Block	The Wall module function block is used to configure and build the required application logic using Wall module components.

Table 2 Function blocks Description (Continued)

Function Block	Description
Modbus Function Block	The Modbus function block is used to configure and build the required application using Modbus device logic.
Data Function Block	The Data Function function block is used to configure and build the required application logic using data components.
Zone Arbitration Function Blocks	The Zone Arbitration function block is used to configure and build the required application logic using zone components.
Sylk Device and Parameters	The Sylk function block is used to configure and build the required application logic using Sylk components.

Note: *Except for PhysicalPoints, any of these objects can be dragged onto the wire sheet of the Periodic or Event programs. Drag and drop PhysicalPoints into the Onboard IO folder. Create a Control Program or an Application by connecting physical points, BACnetObjects, and function blocks.*

Backward compatibility to IRM Function Blocks

Spyder Classic is compatible with a wide range of function blocks. The slot names are frequently added to the CentraLine-N4 kitControl. Additional inputs, outputs, or parameters are frequently included. An additional function block is sometimes used to accomplish the function.

CHAPTER

2 PHYSICAL POINTS FUNCTION BLOCKS

Following terminal points are available in the physical points function block:

- BO Terminal
- BI Terminal
- UI Terminal
- AO Terminal
- UIO Terminal
- Service Pin Terminal
- Flow Sensor
- Actuator Position Feedback
- Onboard IO Package

These terminal points can be configured and used to build the required application logic.

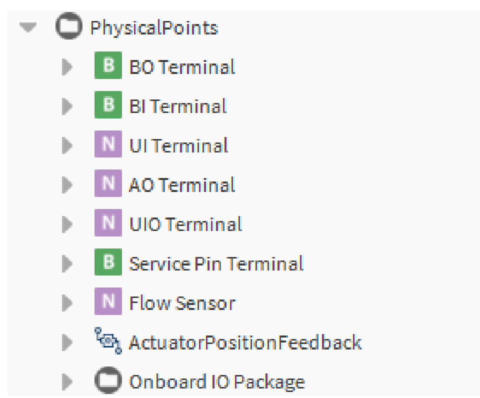


Fig. 6 Physical Points Function Blocks

BO Terminal

The BO Terminal is used for relay outputs from the controllers.

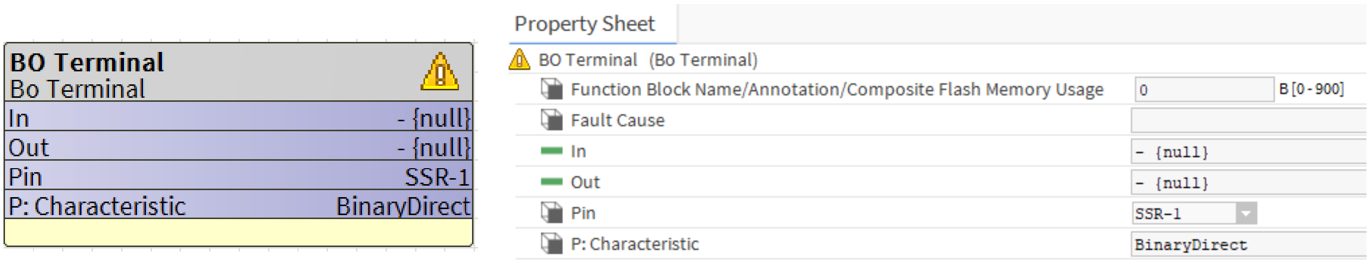


Fig. 7 BoTerminal Function Block and Property Sheet

Input

Table 3 Input of BO Terminal

Input Name	Description
In	This is a boolean point.

Output

Table 4 Output of BO Terminal

Output Name	Description
Out	Status of the connected relay.

Parameter

Table 5 Parameters of BO Terminal

Parameter Name	Description
Fault Cause	This field is read-only. Indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
Pin	Displays a list of supported pins based on the configured controller.
P: Characteristic	This field is read-only. Displays the **Pin** characteristic.

Examples:

Example 1: BO Terminal (Relay Output)

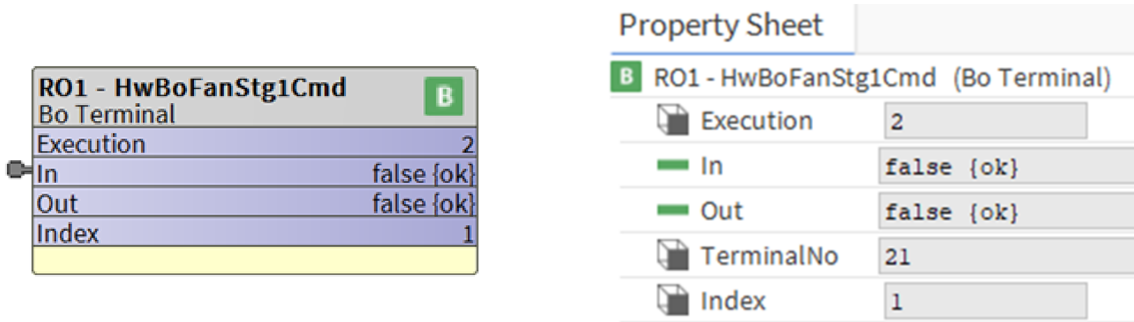


Fig. 8 RO1– HwBoFanStg1Cmd Function Block and Property

Example 2: BO Terminal (Triac Output)

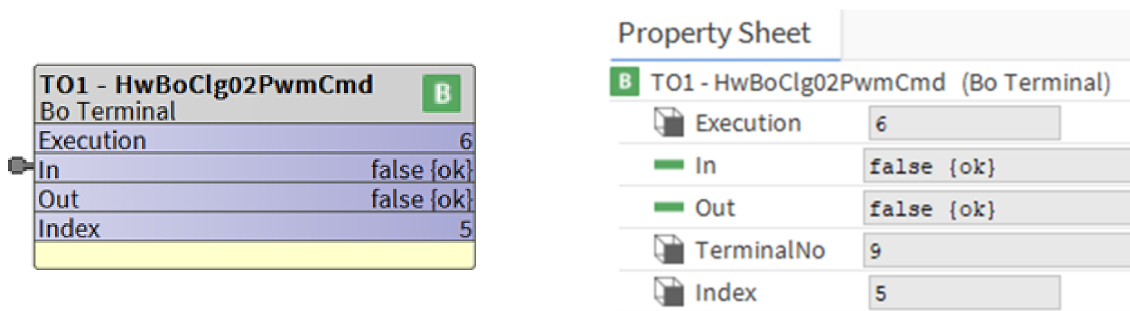


Fig. 9 TO1– HwBoClg02PwmCmd Function Block and Property

BI Terminal

The BI Terminal is used for relay outputs from the controllers.

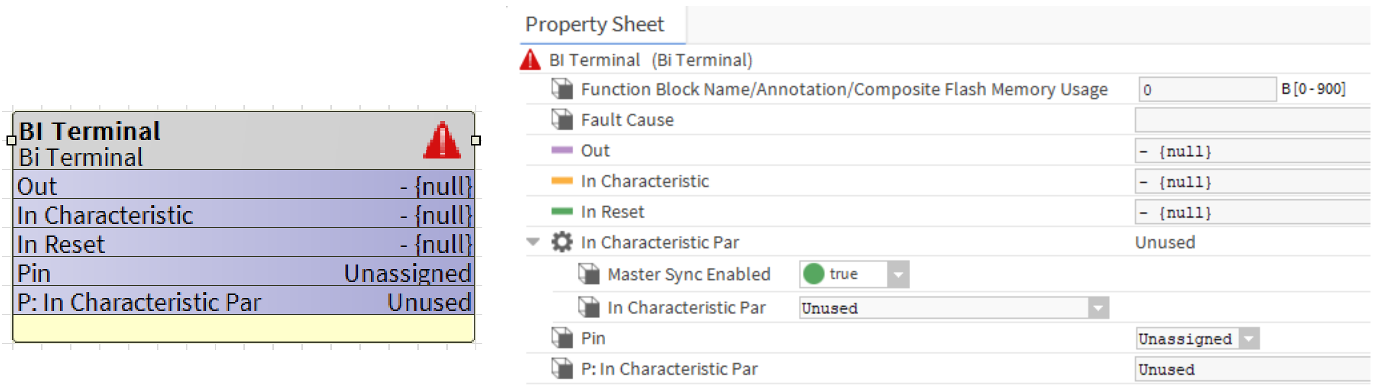


Fig. 10 BI Terminal Function Block and Property Sheet

Input

Table 6 Inputs of BI Terminal

Input Name	Description
In Characteristics	Characteristics specifies the input. **-1:** Unused **0:** Binary Reverse Dry Ct **1:** Binary Direct Dry Ct **2:** SO Pulse Counter Dry Ct **3:** SO Pulse Counter Ext VDC **4:** Binary Reverse Ext Volt **5:** Binary Direct Ext Volt
In Reset	If InCharacteristic = Pulse Counter then a change from 0->1 resets the pulse counter, shown on **Out** to 0. **Booloean:** 0 = False, 1 = True.

Output

Table 7 Output of BI Terminal

Output Name	Description
Out	Status of the connected relay.

Parameter

Table 8 Parameters of BI Terminal

Parameter Name	Description
Fault Cause	This field is read-only. Indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
In Characteristic Par	Characteristics specify the output. **-1:** Unused (default) **0:** Binary Reverse Dry Ct **1:** Binary Direct Dry Ct **2:** SO Pulse Counter Dry Ct **3:** SO Pulse Counter Ext Volt **4:** Binary Reverse Ext Volt **5:** Binary Direct Ext Volt
Pin	Displays a list of supported pins based on the configured controller.
P: Characteristic	This field is read-only. Displays the configured characteristic.

Examples:

Example 1: BI Terminal (Occupancy Sensor)

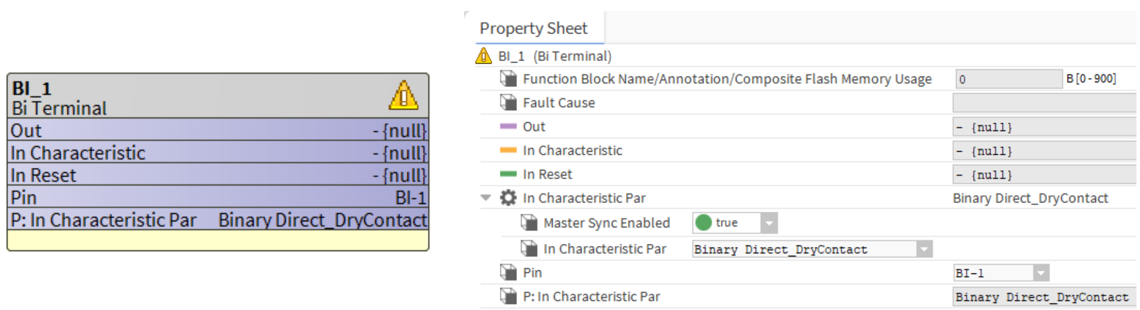


Fig. 11 BI– Occupancy Sensor Function Block and Property

UI Terminal

The UI Terminal is used to connect all universal inputs (such as analog and digital inputs). The UI Terminal function block reads the switches and sensors that are directly wired (connected to the UIx terminals on the controller).

The counter displays the total number of pulses (one pulse is a rising edge followed by a falling edge) on the UI terminal since the controller's last reset due to power-up or reset. The Output value is equal to Null after power-up and reset, indicating that previous count values have been lost. The counter begins with the value 0 for the first pulse (rising edge followed by the falling edge is one pulse) and counts up to 2147483647. Then it overflows to the value 1 on the scale.

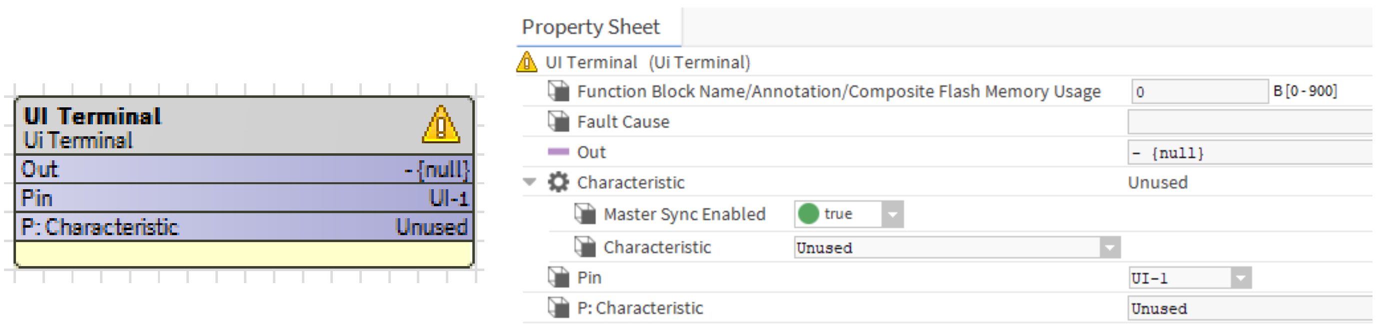


Fig. 12 UI Terminal Functional Block and Property Sheet

Pulse Input Specification

Max. frequency: 15 Hz (54,000 pulses per hour)

Min. pulse width: 20 ms (debounced)

Min. time between two pulse: 33 ms

Max. chatter: 5 ms

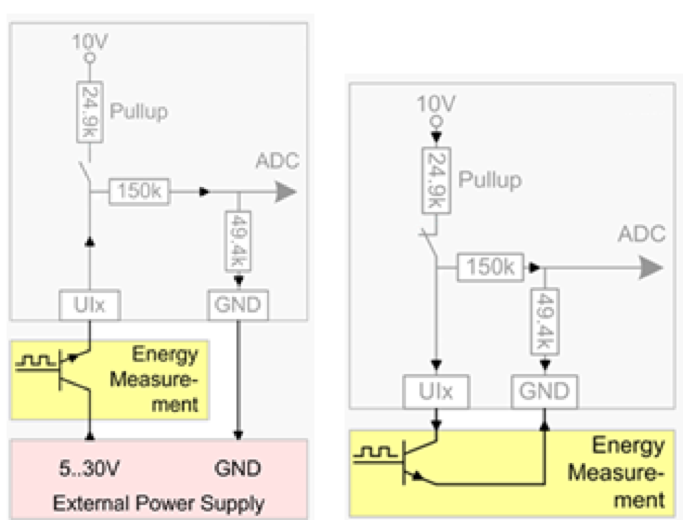


Fig. 13 Wiring Diagram for S0 Counter

Note: *The current flows on the left or right depending on the Energy managements low or high pulse input.*

Input

Table 9 Input of UI Terminal

Output Name	Description
NA	NA

Output

Table 10 Output of UI Terminal

Output Name	Description
Out	• For Unused, **Out = 0**. • For the characteristic Binary Reverse. If resistance seen at the input terminal is less than 10 KOhm, **Out** is False. If resistance seen at the input terminal is greater than 20 KOhm, **Out** is True. • For the characteristic Binary Direct. If resistance seen at the input terminal is less than 10 KOhm, **Out** is True. If resistance seen at the input terminal is greater than 20 KOhm, **Out** is False. • For the characteristics NTC10, NTC20, Pt1000, Ni1000TK5000, **Out** is -58 to 302 °F or -50 to 150 °C. In the case of a sensor break (open), **Out** is +infinity. In the case of sensor short, **Out** is -infinity. • For the characteristics 0 - 10 VDC, **Out** is 0 - 100 %. For the characteristics 2 - 10 V. Input voltage >= 2 V and <=10 V, **Out** = 0 – 100 %. Input Voltage >=1.5 V and < 2 V, **Out** = 0 %. Input Voltage < 1.5 V, **Out** = +infinity (sensor short condition on UI). • For the SetPt10k Characteristics, **Out** = 0 - 100 % from 0 Ohm to 10000 Ohm. • For the characteristics S0 Counter with external voltage or S0 Counter with dry contact, **Out** = Null, 1 - 2147483647.

Parameter

Table 11 Parameters of UI Terminal

Parameter Name	Description
Fault Cause	This field is read-only. Indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
Characteristic	Various characteristics can be selected as given below: • Unused • BinaryReverse • 0-10 VDC • MilliOhms • 2-10 VDC • BinaryDirect • S0 Counter with external voltage • S0 Counter with dry contact • 0-10 mA • 4-10 mA • NTC10 • NTC20 • Pt1000 • SetPt10kCharacteristics • Ni1000TK5000 • NTC10Type3 ***Note:*** *The UI terminal may not support all of the listed characteristics.*
Pin	Displays a list of supported pins based on the configured controller.
P: Characteristic	This field is read-only. Display the configured characteristic.

Examples

Example 1: UI Terminal (Room temperature)

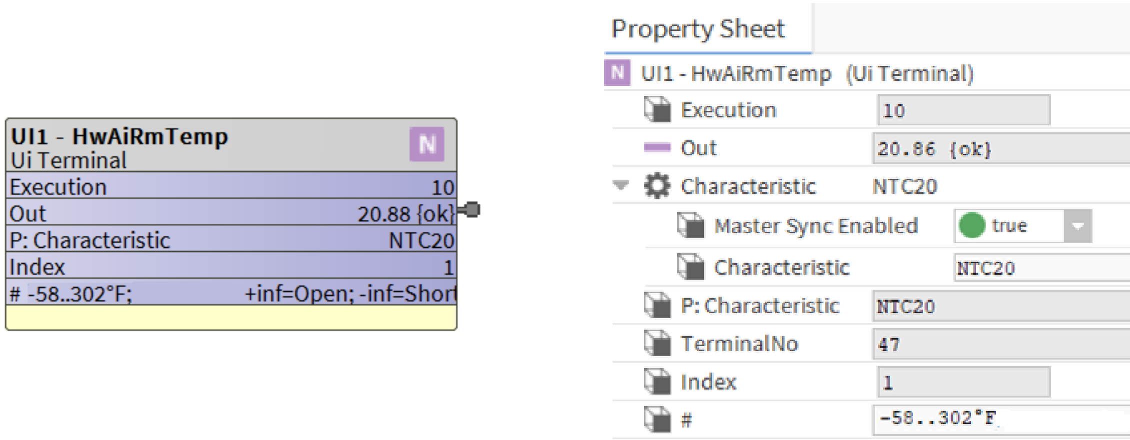


Fig. 14 UI1 – HwRmTemp Function Block and Property

Example 2: UI Terminal (Room CO2)

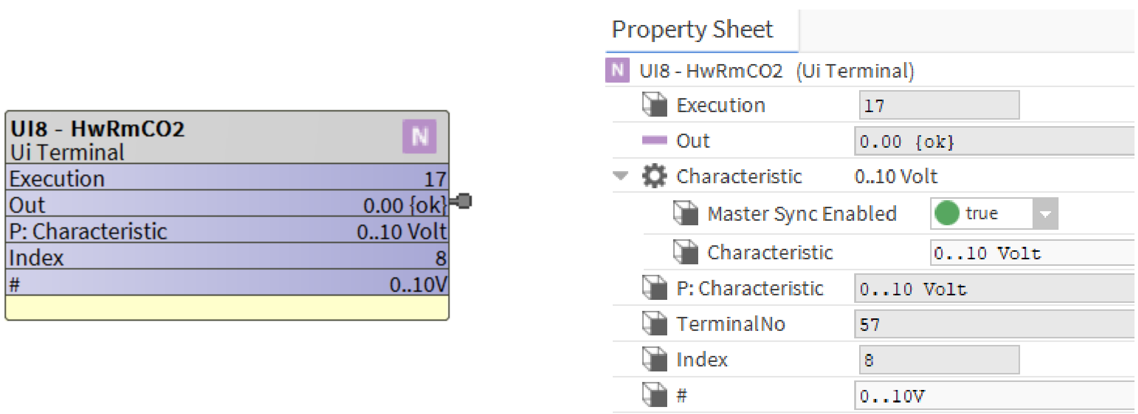


Fig. 15 UI8 – HwRmCO2 Function Block and Property

Example 3: UI Terminal (Occupancy Sensor)

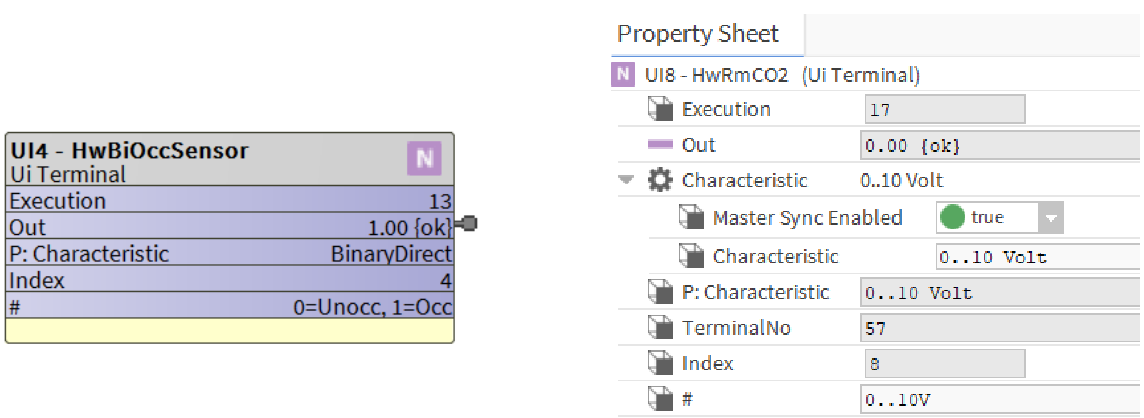


Fig. 16 UI4 – HwBiOccSensor Function Block and Property

Example 4: UI Terminal (Window Contact)

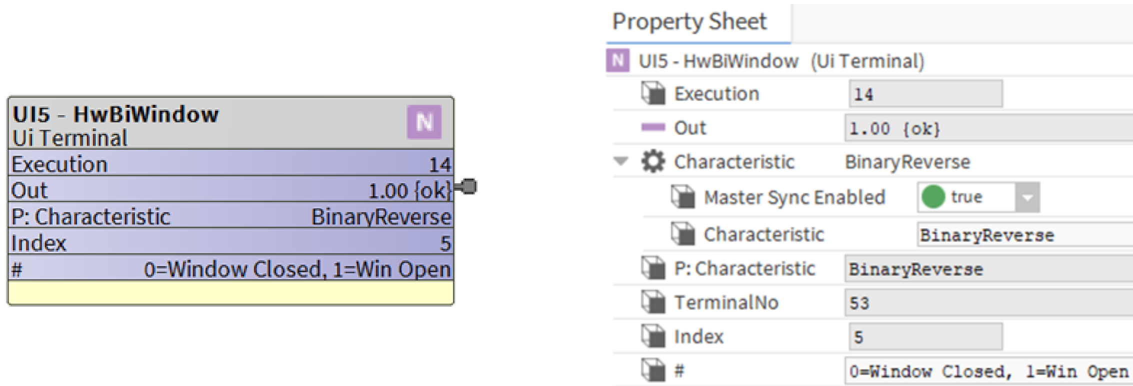


Fig. 17 UI5 – HwBiWindow Function Block and Property

Example 5: UI Terminal (S0 Counter with External Voltage)

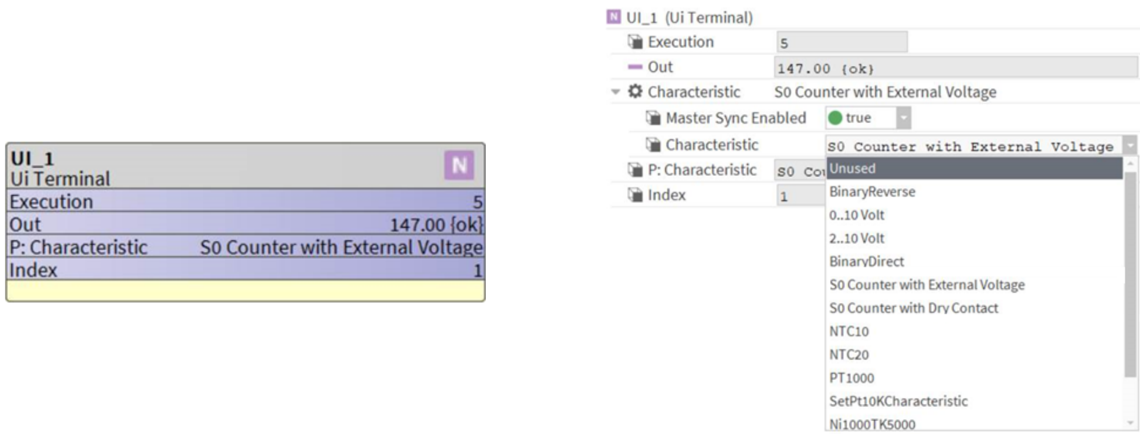


Fig. 18 S0 Counter with External Voltage Function Block and Property

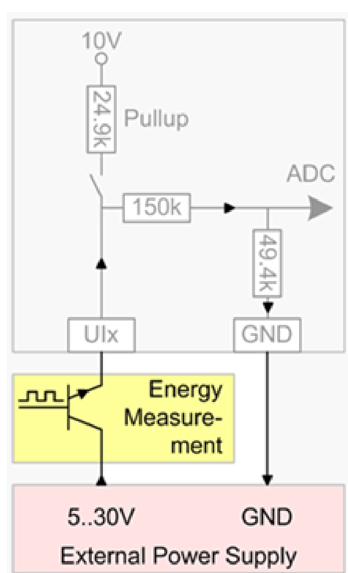


Fig. 19 S0 Counter with external Voltage

Example 6: UI Terminal (S0 Counter with Dry Contact)

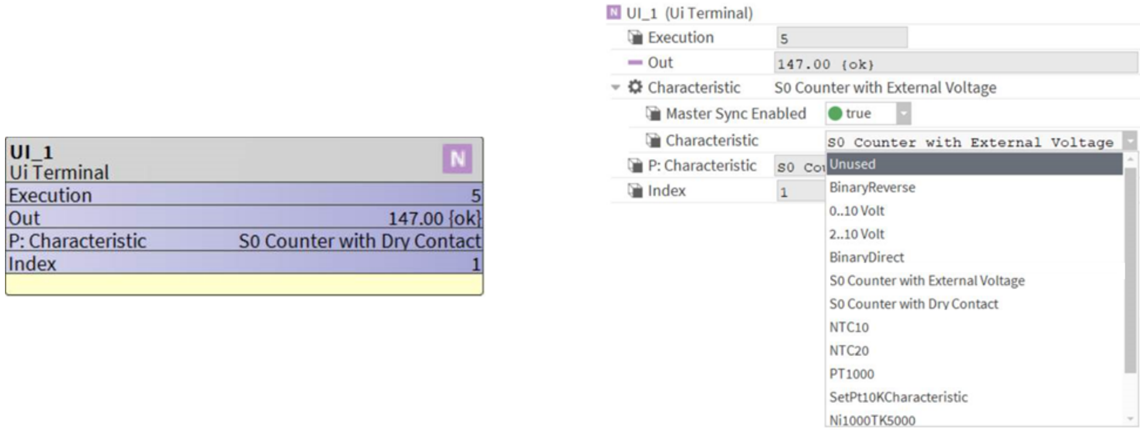


Fig. 20 S0 Counter with Dry Contact Function Block and Property

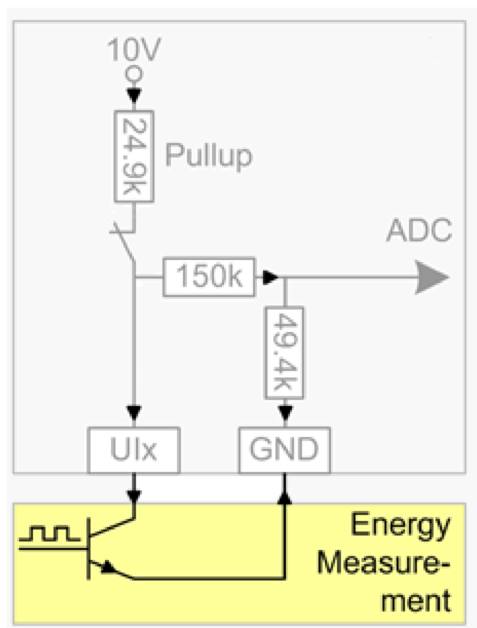


Fig. 21 S0 Counter with dry contact

Note: *The current flows on the left or right depending on the Energy managements low or high pulse input.*

AO Terminal

The AO Terminal is used to control the analog output of the control filed device.

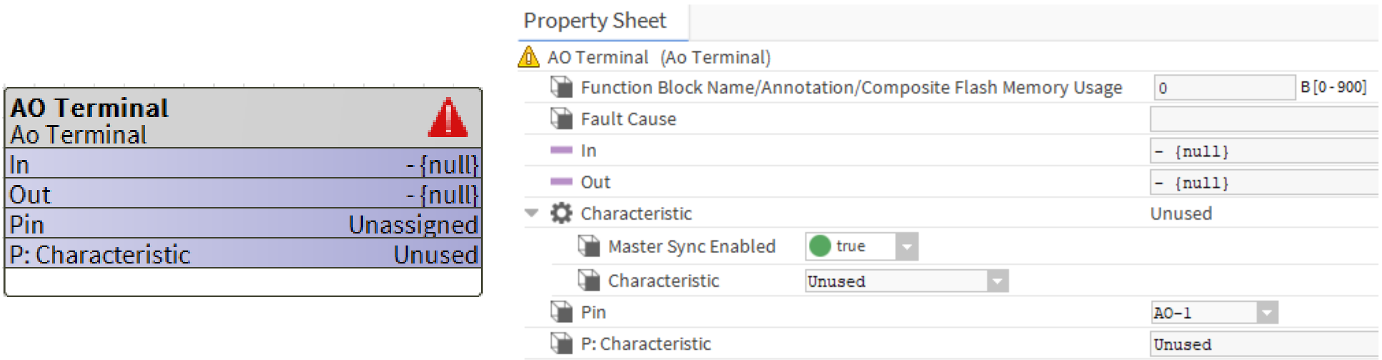


Fig. 22 AO Terminal Function Block and Property

Input

Table 12 Input of AO Terminal

Input Name	Description
In	This is a numeric point.

Output

Table 13 Output of AO Terminal

Output Name	Description
Out	Analog output.

Parameter

Table 14 Parameters of AO Terminal

Parameter Name	Description
Characteristics	The following characteristics can be selected: • Unused • Direct 0 - 10 VDC • Direct 1 - 10 VDC • Direct 2 - 10 VDC • Reverse 0 - 10 VDC • Reverse 1 - 10 VDC • Reverse 2 - 10 VDC • 4 - 20 mA

Table 14 Parameters of AO Terminal (Continued)

Parameter Name	Description
Fault Cause	This field is read-only. Indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
Characteristic	The following characteristics can be selected: • Unused • Direct 0 - 10 VDC • Direct 1 - 10 VDC • Direct 2 - 10 VDC • Reverse 0 - 10 VDC • Reverse 1 - 10 VDC • Reverse 2 - 10 VDC • 4 - 20 mA
Pin	Displays a list of supported pins based on the configured controller.
P: Characteristic	This field is read-only. Displays the configured characteristic.

Examples

Example 1: AO Terminal (Cooling valve Control)

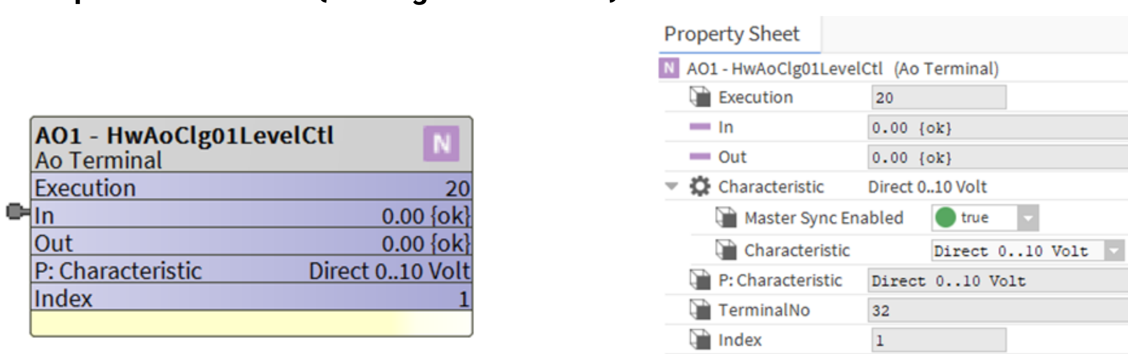


Fig. 23 AO1 – HwAoClgo1LevelCtl Function Block and Property

Example 2: AO Terminal (Fan Speed Control)

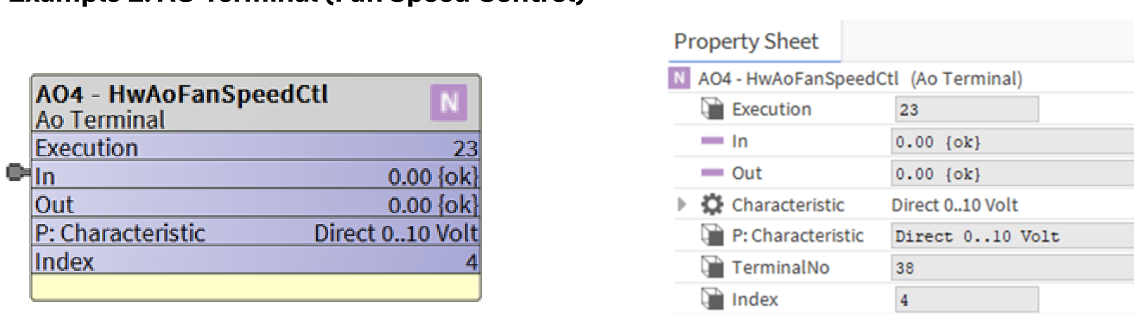


Fig. 24 AO4 – HwAoFanSpeedCtl Function Block and Property

UIO Terminal

This function block is used to configure the hardware inputs and outputs. UIO means that the hardware does not have a fixed number of AI, BI, BO, or AO, but each terminal is configured via this function block regardless of whether it is an AI, BI, BO, or AO.

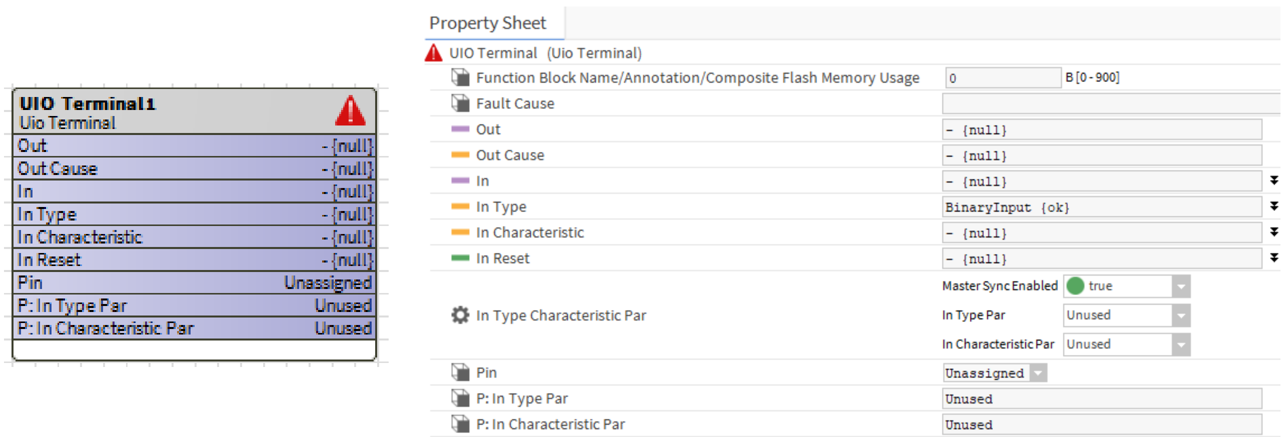


Fig. 25 UIO Function Block and Property

Input

Table 15 Inputs of UIO Function block

Input Name	Description
In	This input is used if the **InType** is Binary output or Analog output. • If the **InType** is configured as Binary input or Analog input, **In** is ignored. • If **InType** is Binary Output, the **In** value is 0 for false and 1 for true. • If **InType** is Analog Output, the **In** value is a percentage value in the range 0-100 %.
In Type	This input defines the hardware type of UIO. If the type is selected, the more exact usage is configured via the characteristics. **-1: Not used**, **Out** = **Null**, **OutCause** = **Not Configured**. 0 is Binary Input. The connected contact or pulses is read, and the result is output on **Out**. **1: Binary Output**. The connected device is switched ON or OFF according to the input **In**. **Out** = **In**. **2: Analog Input.** The connected sensor is read, and the result is output on **Out**. **3: Analog Output**. The connected device gets a voltage or current according to the input **In**. **Out** = **In**.

Table 15 Inputs of UIO Function block (Continued)

Input Name	Description
In Characteristics	Characteristics specify the use of input or output defined via **InType**. The values are not unique. • **InType = Not used** , Characteristics is ignored. • **InType = Binary Input** **1:** Not Used, **Out = Null, OutCause = Not configured** **0:** Binary Reverse Dry Ct **0 V:** Contact is closed and the output is True. **1:** Binary Direct Dry Ct **2:** S0 Pulse Counter Dry Ct **3:** S0 Pulse Counter Ext Volt **4:** Binary Reverse Ext Volt **5:** Binary Direct Ext Volt • **InType = Binary Output** **-1:** Not Used **0:** Binary Reverse **1:** Binary Direct If In = 1, then the Uio get 0 V, otherwise 10 V. • **InType = Analog Input** **-1:** Not Used, **Out = Null, OutCause = Not configured** **1:** 0 - 10 VDC **2:** milliOhms **3:** 2 - 10 VDC **8:** 4 - 20 mA **9:** 0 - 20 mA **32:** NTC10KType2 **33:** NTC20K **34:** PT100 **36:** PT1000 **37:** BALCO500 **38:** SetPT10KCharacteristic **39:** Ni1000TK5000 **40:** NTC10KTType3 **41:** 10K3A1 **42:** PT3000 **43:** NickelClassBDIN43760 **44:** JOHNSON A99

Table 15 Inputs of UIO Function block (Continued)

Input Name	Description
	• **InType** = **Analog Output** **-1:** Not Used **1:** Direct 0 - 10 VDC **2:** Direct 1 - 10 VDC **3:** Direct 2 - 10 VDC **4:** Reverse 0 - 10 VDC **5:** Reverse 1 - 10 VDC **6:** Reverse 2 - 10 VDC **7:** Direct 0 - 20 mA **8:** Direct 4 - 20 mA **9:** Direct 0 - 11 VDC **10:** Reverse 0 - 11 VDC **11:** Reverse 0 - 20 mA **12:** Reverse 4 - 20 mA **Enum:** See above values, Null
InReset	If **InType** = **Binary Input** and **InCharacteristic** = **Pulse Counter,** a change from 0->1 resets the pulse counter, shown on **Out** to 0. **Boolean:** 0 = False, 1 = True.
P: In Type Par	This shows on the Function block Level the configuration parameter **InTypePar.** The value is read-only. This value is independent of the input **InType**. The effective value between the input and the parameter is not displayed here, only the value of the parameter is displayed.
In Type Characteristic Par	• **In Type Par:** This parameter is also available as an input, please refer to the description of the input **InType**. Default is Used. • **In Type Characteristic Par:** This parameter is also available as an input, please refer to the description of the input **InCharacteristics**. Default is Not Used.

Output

Table 16 Outputs of UIO Function block

Output Name	Description
Out	If **InType** = Binary Output or Analog Output, **Out** = In. **Numeric:** 32-Bit floating-point value (float).

Table 16 Outputs of UIO Function block (Continued)

Output Name	Description
Out Cause	This output has a number of conditions that can occur, particularly when reading analog values. **1:** Not configured **2:** Normal If **Intype = Analog** input and Characteristic is a temperature sensor. **3:** Sensor open **4:** Sensor Short If Characteristic is 2 - 10 VDC. **5:** Under Range If Characteristic is 0 - 10 V, 2 - 10 V, 0 - 20 mA, 4 - 20 mA or milliOhms. **6:** Over Range

Parameter

Table 17 Parameters of UIO Function block

Parameter Name	Description
In Type Par	See **InType** for description. Default = Not Used.
Fault Cause	This field is read-only. Indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
Pin	Displays a list of supported pins based on the configured controller.
P: In Type Par	This parameter is also available as an input, please refer to the description of the input **InType**. Default is Unused.
P: InCharacteristics Par	This parameter is also available as an input, please refer to the description of the input **InCharacteristics**. Default is unused.

Examples

Example 1: UIO Terminal (Occupancy Sensor)

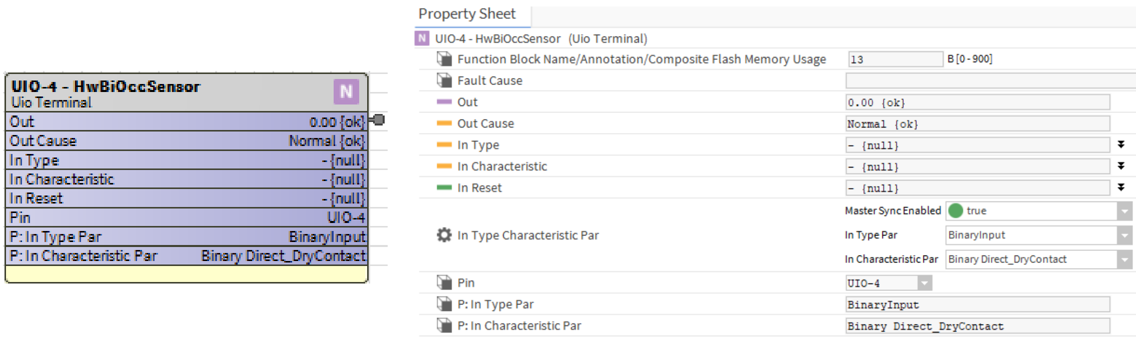


Fig. 26 UIO – HwBiOccSensor Function Block and Property

Example 2: UIO Terminal (Space Temperature)

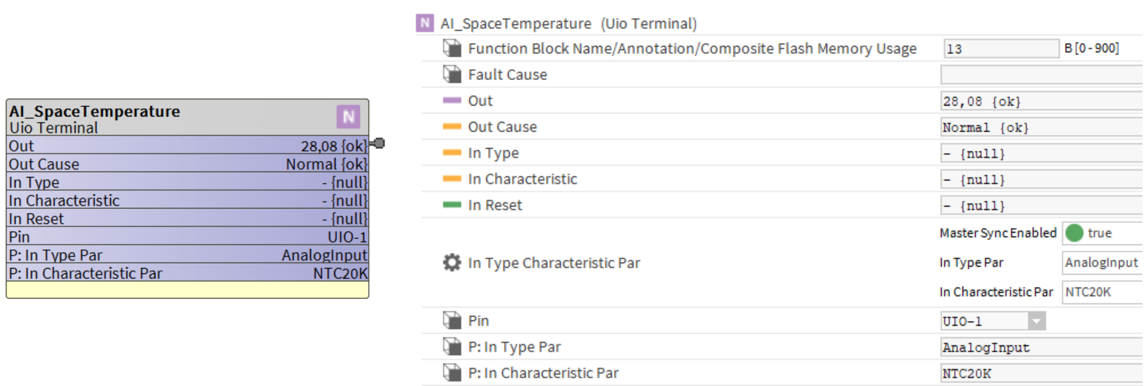


Fig. 27 UIO – SpaceTemperature Function Block and Property

Example 3: UIO Terminal (Cooling)

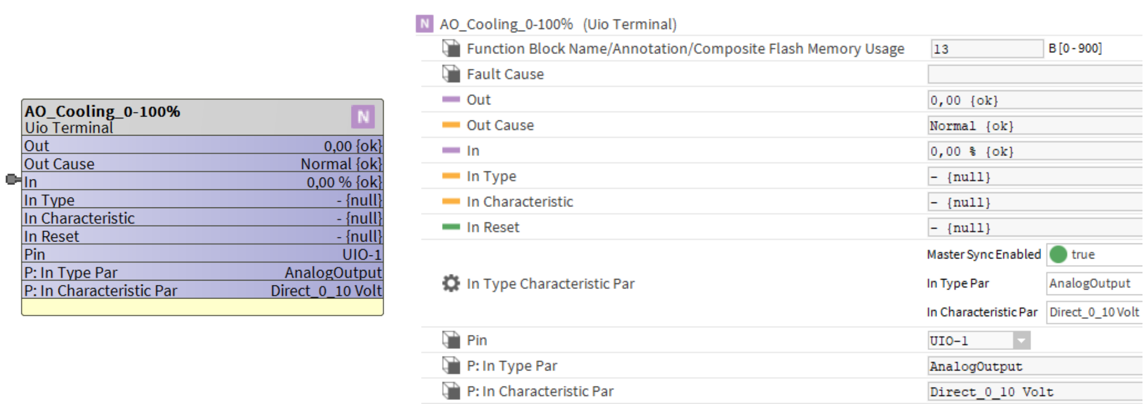


Fig. 28 UIO – Cooling Function Block and Property

Example 5: UIO Terminal (Reheat with external SSR)

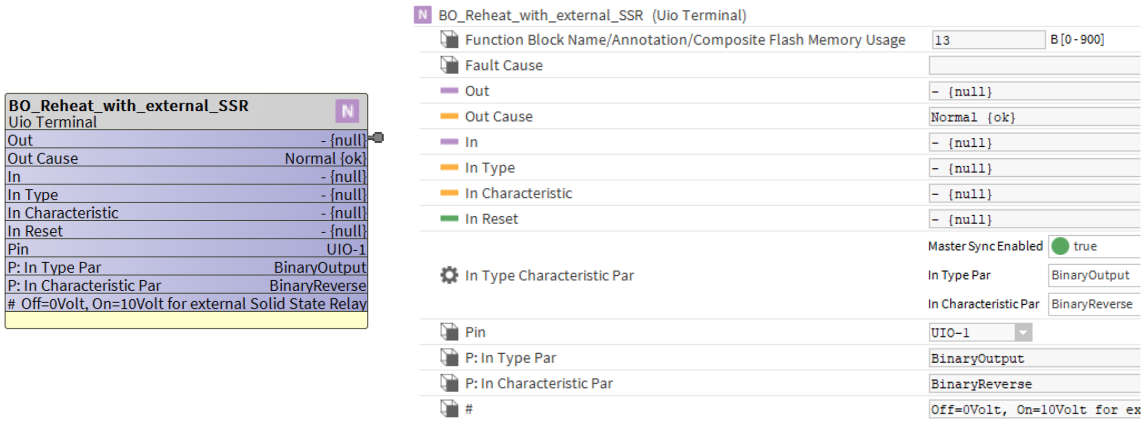


Fig. 29 UIO – Reheat with external SSR Function Block and Property

Service Pin Terminal

This block indicates the status of the controller's service pin.

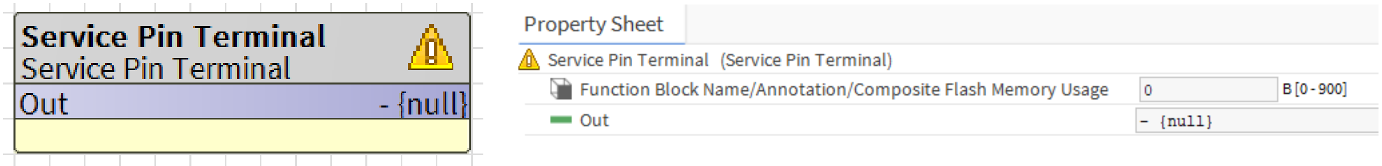


Fig. 30 ServicePinTerminal Function Block

Output

Table 18 Output Of Service Pin Terminal

Output Name	Description
Out	If the Service Pin button is pressed on the controller, the output becomes “true.”

Flow Sensor

In addition to the UI terminal, an integrated sensor for measuring differential pressure is included. If the air flow calibration process for this sensor was successful, an air flow is calculated from the differential pressure.

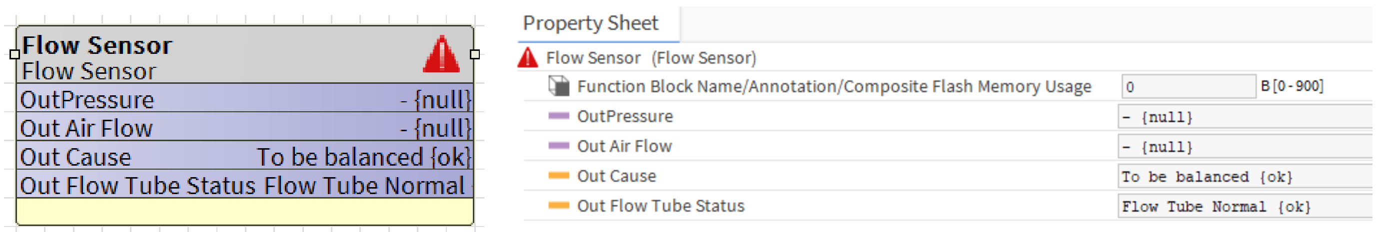


Fig. 31 FlowSensor Function Block

Output

Table 19 Outputs of Flow Sensor

Output Name	Description
Out Pressure	The output pressure represents the measured differential pressure. The value range is determined by the measuring range of the pressure sensor. The value may be negative depending on the mounting position. The engineering unit is determined by the global engineering unit configuration selected in the control manager. • In the SI-System, the engineering unit is Pascal • In the Imperial-System, the engineering unit is inH2O. • If the sensor is defective, the value is Null. **Numeric:** 32-bit floating point value, Null
Out Air Flow	The air flow is always positive, regardless of whether the differential pressure is positive or negative. ***Note:*** *The air flow sensor will give air flow even if the box is not calibrated. It will use the default flow calibration factor .* The engineering unit is configured via the parameter Air Flow Unit. **Numeric:** 32-Bit floating point value, Null
Out Cause	Different balancing states are displayed using the Cause output, (ENUM): **1:** To be balanced (Air flow balancing is not done). **2:** Set point balanced (Air flow balancing is done using Setpoint balancing method). **3:** Min-Max balanced (Air flow balancing is done using Min-Max balancing method). **4:** Set point balanced with pre-calculated flow calibration factor (Balancing was completed by directly entering the calibrated K factor).

Table 19 Outputs of Flow Sensor (Continued)

Output Name	Description
Out Flow tube Status	It is used to determine if the Airflow tube connection is reversed. Typically, users should connect total pressure to (+) and static pressure tube to (-). If its connected in the opposite way then Out Flow Tube Status will display status "Flow Tube Reversed".

Parameter

Table 20 Parameter of Flow Sensor

Parameter Name	Description
Air Flow Unit	The parameter specifies the engineering unit of the output **Out** Air Flow. Furthermore, if there are issues with the installation, the sensor can be completely deactivated without changing the wiresheet Logic. **ENUM:** **1:** CFM (Cubic feet per minute) **2:** CM / hr (Cubic meter per hour) **3:** Lt / s (Liter per second) **Null:** CFM (Cubic feet per minute)

Actuator Position Feedback

Actuator Position Feedback function block is used to get outputs from the actuator.

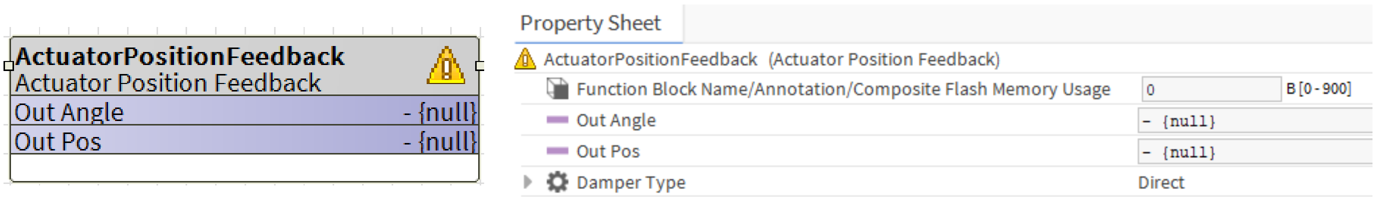


Fig. 32 ActuatorPositionFeedback Function Block and Property

Input

Table 21 Input of Actuator Position Feedback

Input Name	Description
NA	NA

Output

Table 22 Outputs of Actuator Position Feedback

Output Name	Description
Out Angle	It gives the angle of the actuator.
Out Pos	It shows the position of the actuator.

Parameter

Table 23 Parameter of Actuator Position Feedback

Parameter Name	Description
Damper Type	Allows to set the damper movement **Direct**: 0 -100 % **Reverse**:100 - 0 %

Onboard IO Package

The Onboard IO package is a palette available for IRM CVAV (Unitary, Compact VAV) and IRM IP VAV (IP or MSTP).

Onboard IO IRM CVAV Unitary Packages

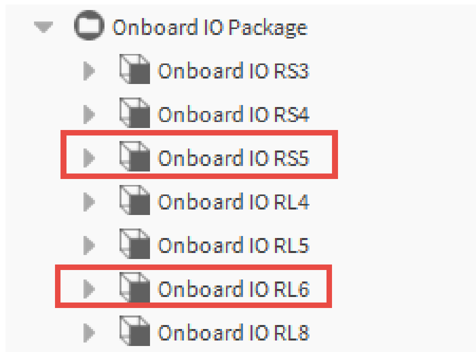


Fig. 33 Onboard IO Unitary Packages

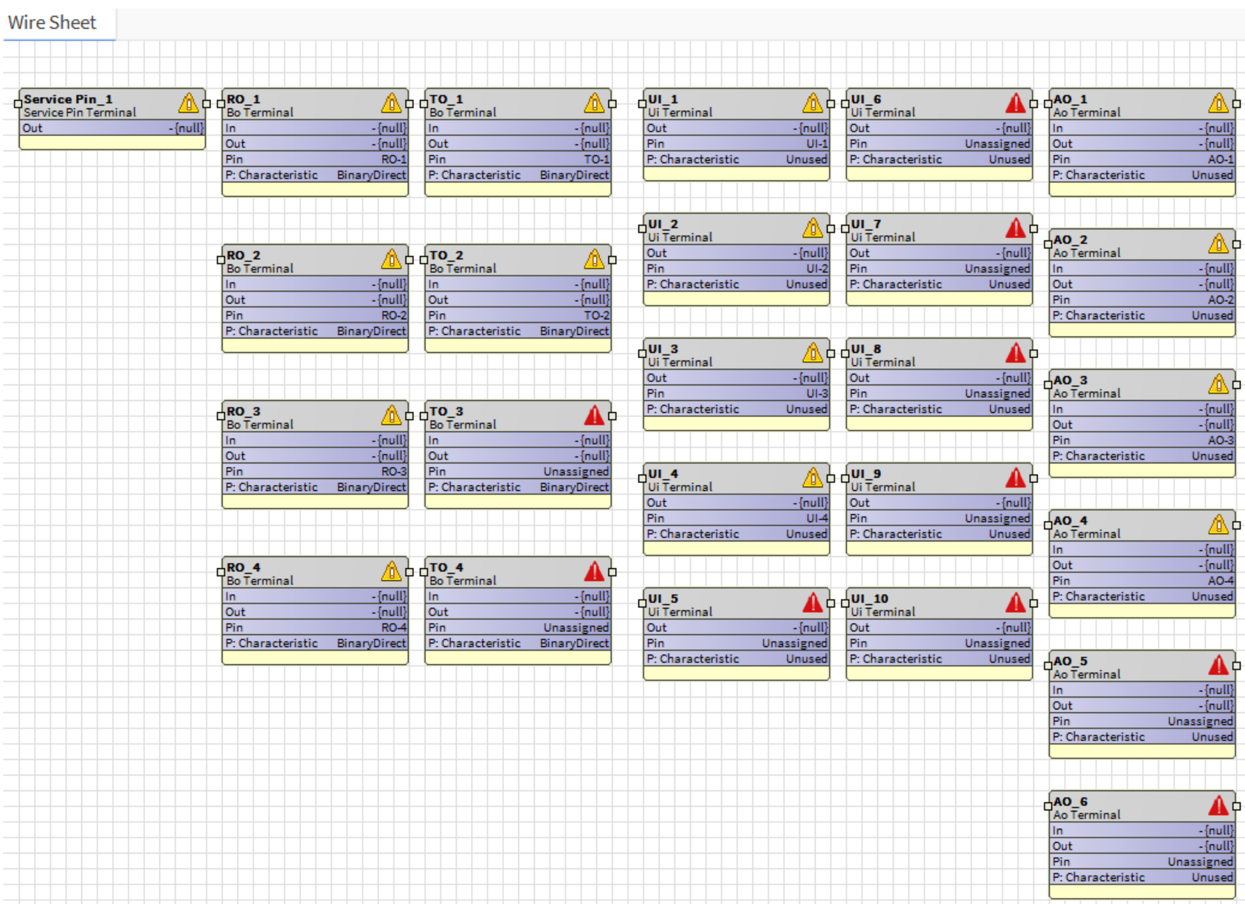


Fig. 34 Onboard IO Unitary Function Block

Onboard IO IRM CVAV VAV Package

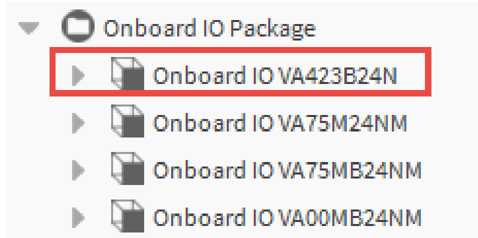


Fig. 35 Onboard IO Compact VAV Package

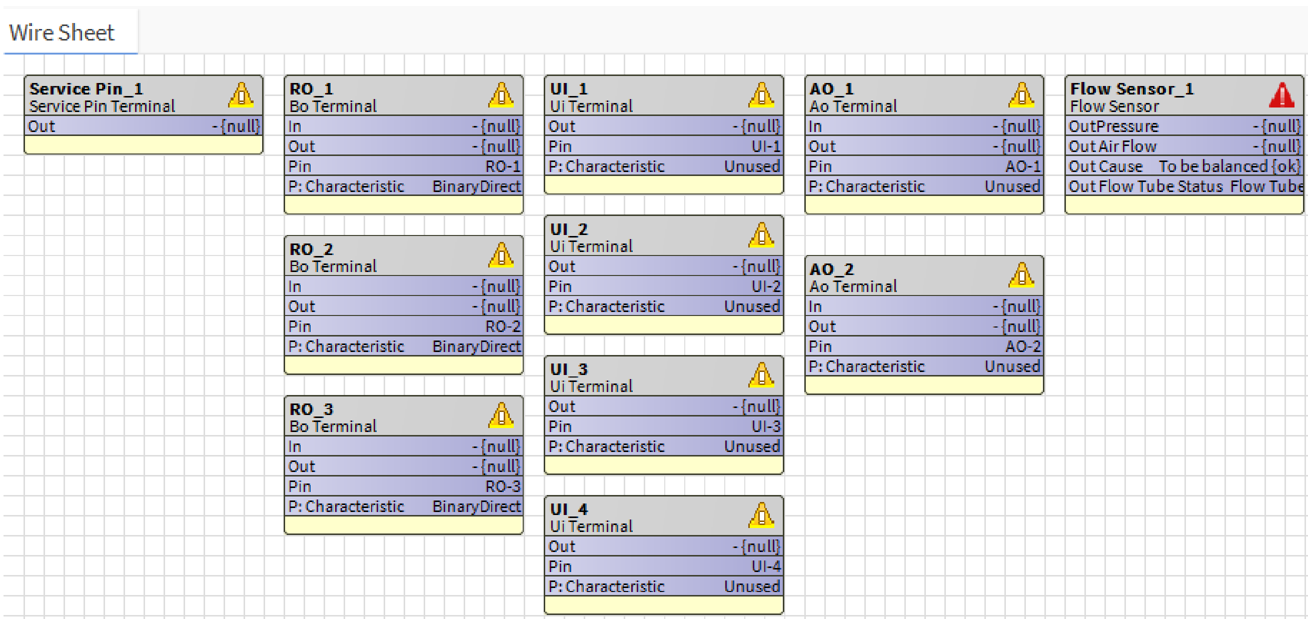


Fig. 36 Onboard IO Compact VAV Function Block

Onboard IO IRM IP VAV Packages

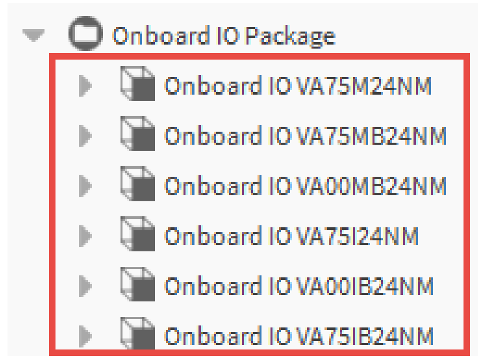


Fig. 37 Onboard IO IP and MSTP Packages

Fig. 38 Onboard IO IP and MSTP Function Block

CHAPTER

3

BACNET OBJECTS FUNCTION BLOCKS

The following BACnet Objects function blocks are available in the honIrmControl Palette and can be configured and used to create the required application logic:

- Bacnet Numeric Input
- Bacnet Boolean Input
- Bacnet Numeric Output
- Bacnet Boolean Output
- Bacnet Enum Output
- Bacnet Numeric Value
- Bacnet Boolean Value
- Bacnet Enum Value
- Ref In
- Ref Out

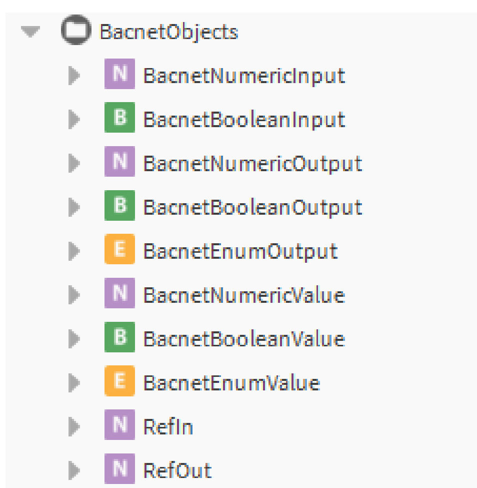


Fig. 39 BACnet Objects Function Blocks

Bacnet Numeric Input

The BACnet Numeric Input represents a ‘Analog Input’ object over BACnet.

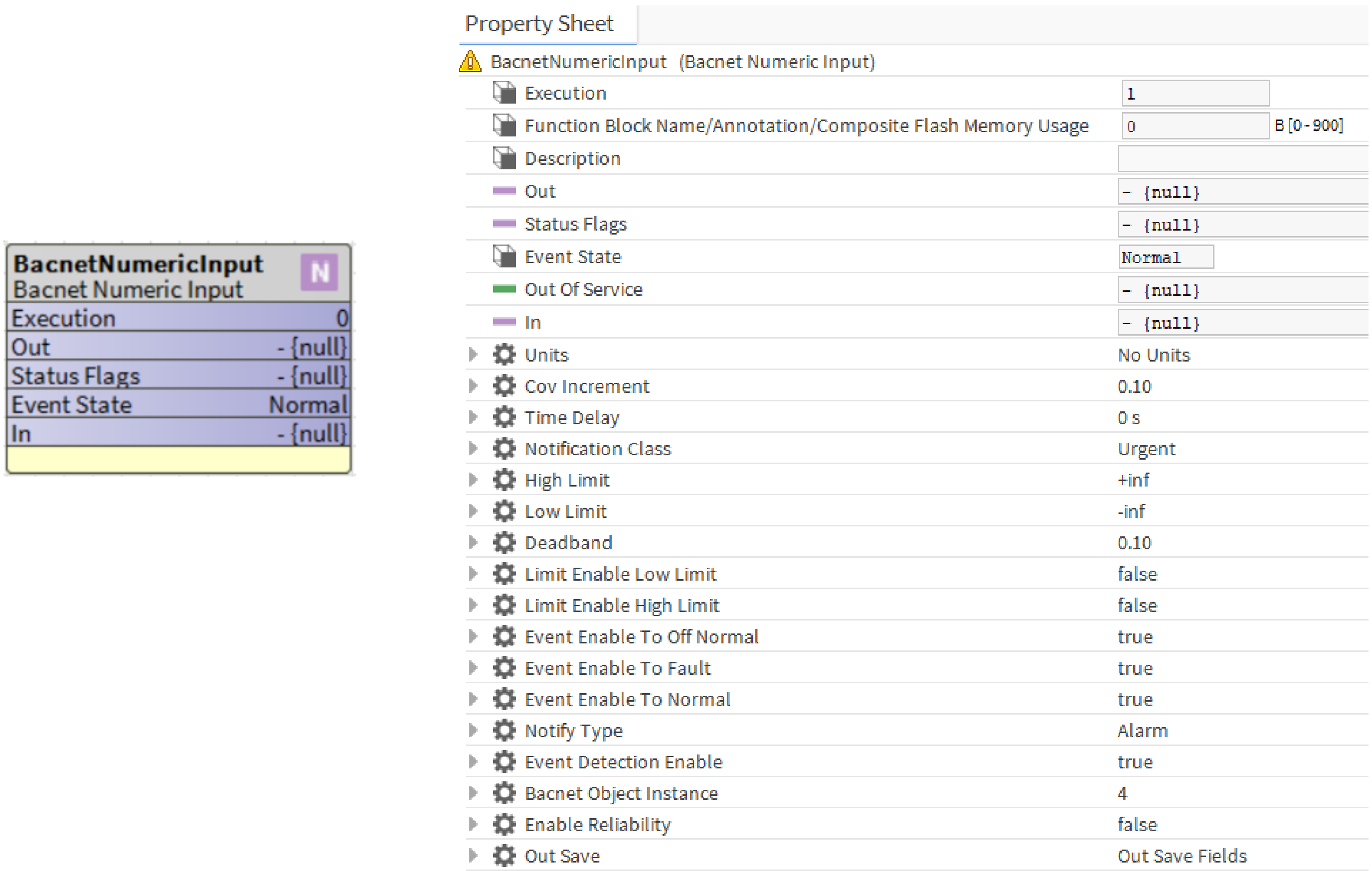


Fig. 40 BacnetNumericInput Function Block and Property Sheet

Input

Table 24 Input of Bacnet Numeric Input

Input Name	Description
In	It is a 32-Bit floating input value.

Output

Table 25 Outputs of Bacnet Numeric Input

Output Name	Description
Out	• When **Out Of Service** is “false”- **Out** is **In** value. • When **Out Of Service** is “true”– **Out** is the present Value written over BACnet.

Table 25 Outputs of Bacnet Numeric Input (Continued)

Output Name	Description
Status Flags	Displays the status of this function block. It can be any one or a 'OR' combination of the following values: **1:** In Alarm **2:** Fault **4:** Overwritten **8:** Out Of Service **Example:** • A value of 1 means the function block is in **In Alarm** state. • A value of 9 means the function block is in **Out Of Service** and in **In Alarm** state.
Event State	Displays the present event state of this function block. It can be any one of the following values: **0:** Normal **1:** Fault **3:** High Limit **4:** Low Limit
Out Of Service	It shows whether the function block is in out of service state or not. • '**Out Of Service**' can be changed via BACnet. Setting '**Out Of Service**' to true separates '**In**' and '**Out**' so that '**In**' values are no longer visible on '**Out**.' • When '**Out Of Service**' is true, BACnet can overwrite '**Out**' (via **Present Value** property).

Parameter

Table 26 Parameters of Bacnet Numeric Input

Parameter Name	Description
Units	Defines the **Out** values of the engineering unit.
Cov Increment	Defines the change in **Out** delta value that will trigger COV update notifications to other BACnet bus recipients.
Time Delay	Defines the time delay (in seconds); once conditions are met, the **High Limit** or **Low Limit** or **Normal** transition is set to off.
Notification Class	Defines the notification class, which can be one of the following: • Urgent • High • Low • User Defined

Table 26 Parameters of Bacnet Numeric Input (Continued)

Parameter Name	Description
High Limit	Defines the high limit. If the **Limit Enable High Limit** is defined as "true" and the **Out** value exceeds this limit, then the condition exists for at least the defined **Time Delay** period. • **Event State** is changed to **High Limit**. • **Status Flags** is set to **In Alarm**. • An event notification is sent to all recipients listed by the **Notification Class** (if '**Event Enable to Off Normal**' is defined true).
Low Limit	Defines the low limit. If the **Limit Enable Low Limit** is defined as "true" and the **Out** value fails below this limit, then the condition exists for at least the defined **Time Delay** period. • **Event State** is changed to **Low Limit**. • **Status Flags** is set to **In Alarm**. • An event notification is sent to all recipients listed by the **Notification Class** (if '**Event Enable to Off Normal**' is defined true).
Deadband	Defines the value of the deadband. While transitioning from an off-normal state (**High Limit** or **Low Limit**) to a normal state, the '**Out**' value must remain between the range – (**Low Limit** + **Deadband**) and (**High Limit** – **Deadband**) for at least the defined **Time Delay** period, then only. • **Event State** is changed to **Normal**. • **Status Flag** is cleared-off of **In Alarm**. • An event notification is sent to all recipients listed by the **Notification Class** (if '**Event Enable To Normal**' is defined true).
Limit Enable Low Limit	When set to "true," the low limit alarm is detected based on the configured **Low Limit** value.
Limit Enable High Limit	When set to "true," the high limit alarm is detected based on the configured **High Limit** value.
Event Enable to Off Normal	When set to "true," event notifications for **Off-Normal** event transitions are sent over BACnet to the recipients specified by the **Notification Class**.
Event Enable to Fault	When set to "true," event notifications for **Fault** event transitions are sent over BACnet to recipients specified by the **Notification Class**.
Event Enable to Normal	When set to "true," event notifications for **Normal** event transitions are sent over BACnet to recipients specified by the **Notification Class**.
Notify Type	Define the following notification types: • **Alarm:** Makes all BACnet event notifications 'Alarm' type originating from this function block instance. • **Event:** Makes all BACnet event notifications 'Event' type originating from this function block instance.

Table 26 Parameters of Bacnet Numeric Input (Continued)

Parameter Name	Description
Event Detection Enable	Setting to “true” enables: • Detection of **High Limit**/ **Low Limit/ Fault/ Normal** events in ‘**Event State**.’ • Setting or resetting of **In Alarm /Fault/ Overwritten/ Out Of Service** flags in **Status Flags**. Setting it to “False” enables: • **Event State** will always remain **Normal**. • **Status Flags** will always remain 0.
Bacnet Object Instance	It shows the instance number of this Bacnet object function block.
Enable Reliability	Setting to “true,” enables the detection of open loop or short loop states. These states show up in the Reliability property of this function block instance. Reliability property is only exposed over Bacnet.
Out Save	• **Master Sync Enabled:** If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Example 1: BacnetNumericInput

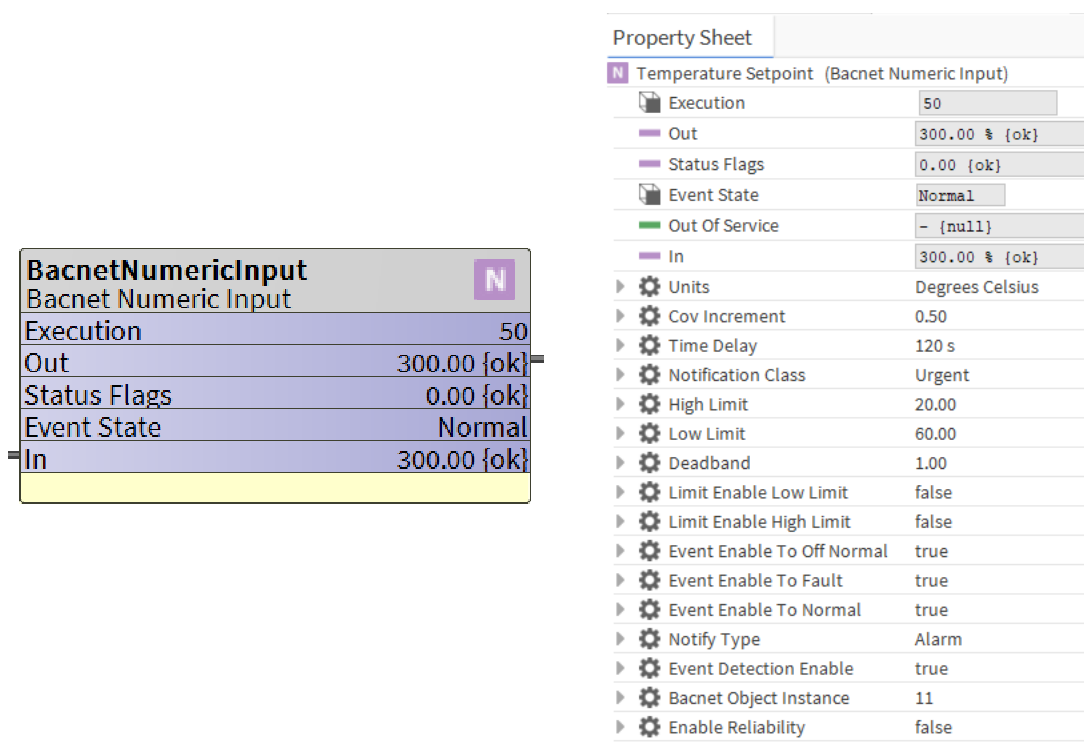


Fig. 41 BacnetNumericInput Function Block and Property Sheet

Bacnet Boolean Input

The Bacnet Boolean Input Function Block, represents a ‘Binary Input’ object over BACnet.

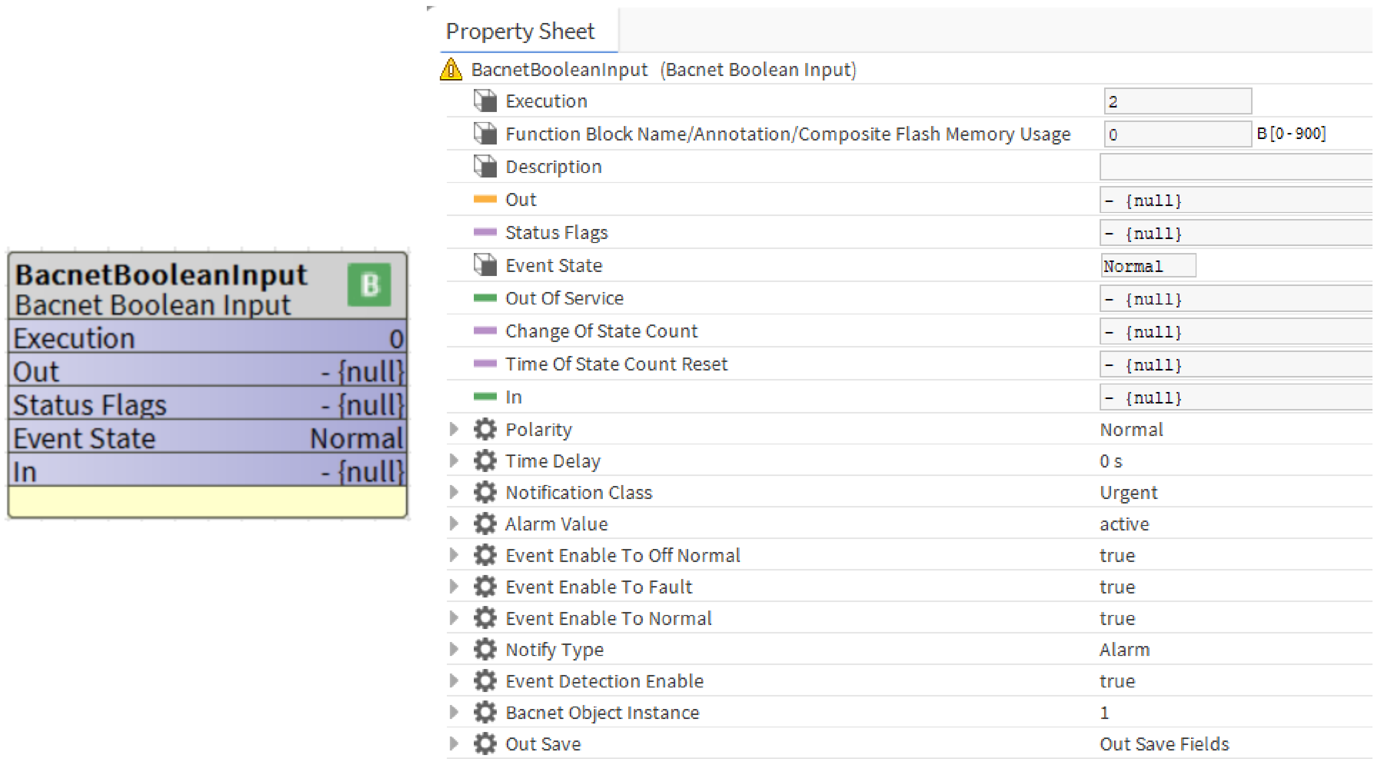


Fig. 42 BacnetBooleanInput Function Block and Property Sheet

Input

Table 27 Input of Bacnet Boolean Input

Input Name	Description
In	It is a boolean input value.

Output

Table 28 Outputs of Bacnet Boolean Input

Output Name	Description
Out	• When **Out Of Service** is “false”- **Out** is **In** value. • When **Out Of Service** is “true”– **Out** is the present Value written over BACnet.

Table 28 Outputs of Bacnet Boolean Input (Continued)

Output Name	Description
Status Flags	Displays the status of this function block. It can be any one or a ‘OR’ combination of the following values: **1:** In Alarm **2:** Fault **4:** Overwritten **8:** Out Of Service **Example:** • A value of 1 means the function block is in the ‘**In Alarm**’ state. • A value of 9 means the function block is in ‘**Out Of Service**’ as well as in the ‘**In Alarm**’ state.
Event State	Displays the present event state of this function block. It can be any one of the following values: **0:** Normal **1:** Fault **2:** Off Normal
Out Of Service	It shows whether the function block is in out of service state or not. • ‘**Out Of Service**’ is modifiable over BACnet. Setting ‘**Out Of Service**’ to true decouples ‘**In**’ from ‘**Out**,’ that is, ‘**In**’ values no longer appear on ‘**Out**.’ • When ‘**Out Of Service**’ is true, ‘**Out**’ can be overwritten over BACnet (via **Present Value** property). • '**Out Of Service**' can be changed via BACnet. Setting '**Out Of Service**' to true separates '**In**' and '**Out**' so that '**In**' values are no longer visible on '**Out**.' • When '**Out Of Service**' is true, BACnet can overwrite '**Out**' (via **Present Value** property).
Change of State Time	Displays the last time ‘**Out**’ state was changed.
Change of State Count	Displays the total count of ‘**Out**’ state changes. This adjustable over BACnet.
Time Of State Count Reset	Displays the last time “**Change of State Count**” was reset over BACnet.

Parameter

Table 29 Parameters of Bacnet Boolean Input

Parameter Name	Description
Polarity	Defines the relationship between '**In**' and '**Out**' options. • **Normal:** '**Out**' is the same as '**In**,' that is, if '**In**' is 'active,' then '**Out**' is 'active.' • **Reverse:** '**Out**' is NOT of '**In**,' that is, if '**In**' is 'active,' then '**Out**' is 'inactive.'
Time Delay	Defines the time delay (in seconds); once conditions are met, the Off-Normal or **Normal** transition is set to off.
Notification Class	Defines the notification class, which can be one of the following: • Urgent • High • Low • User Defined
Alarm Value	Defines the value of '**Out**' that will trigger an '**Off-Normal**' alarm state.
Event Enable To Off Normal	When set to "true," event notifications for **Off-Normal** event transitions are sent over BACnet to the recipients specified by the **Notification Class**.
Event Enable to Fault	When set to "true," event notifications for **Fault** event transitions are sent over BACnet to recipients specified by the **Notification Class**.
Event Enable to Normal	When set to "true," event notifications for **Normal** event transitions are sent over BACnet to recipients specified by the **Notification Class**.
Notify Type	Defines following notification types: • **Alarm:** Makes all the BACnet event notifications originating out of this instance of the function block, of type 'Alarm.' • **Event:** Makes all the BACnet event notifications originating out of this instance of the function block, of type 'Event.'
Event Detection Enable	Setting it to "true" enables: • Detection of **Off-Normal/Fault/Normal** events in **Event State**. • Setting or resetting of **InAlarm/ Fault/ Overwritten/ Out Of Service** flags in **Status Flags**. Setting it to "false" enables: • **Event State** will always remain **Normal**. • **Status Flags** will always remain 0.
Bacnet Object Instance	It shows the instance number of this Bacnet object function block.

Table 29 Parameters of Bacnet Boolean Input (Continued)

Parameter Name	Description
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out**: To enable or disable the **Out** feature.

Example 1: BacnetBooleanInput (Fan Off Alarm)

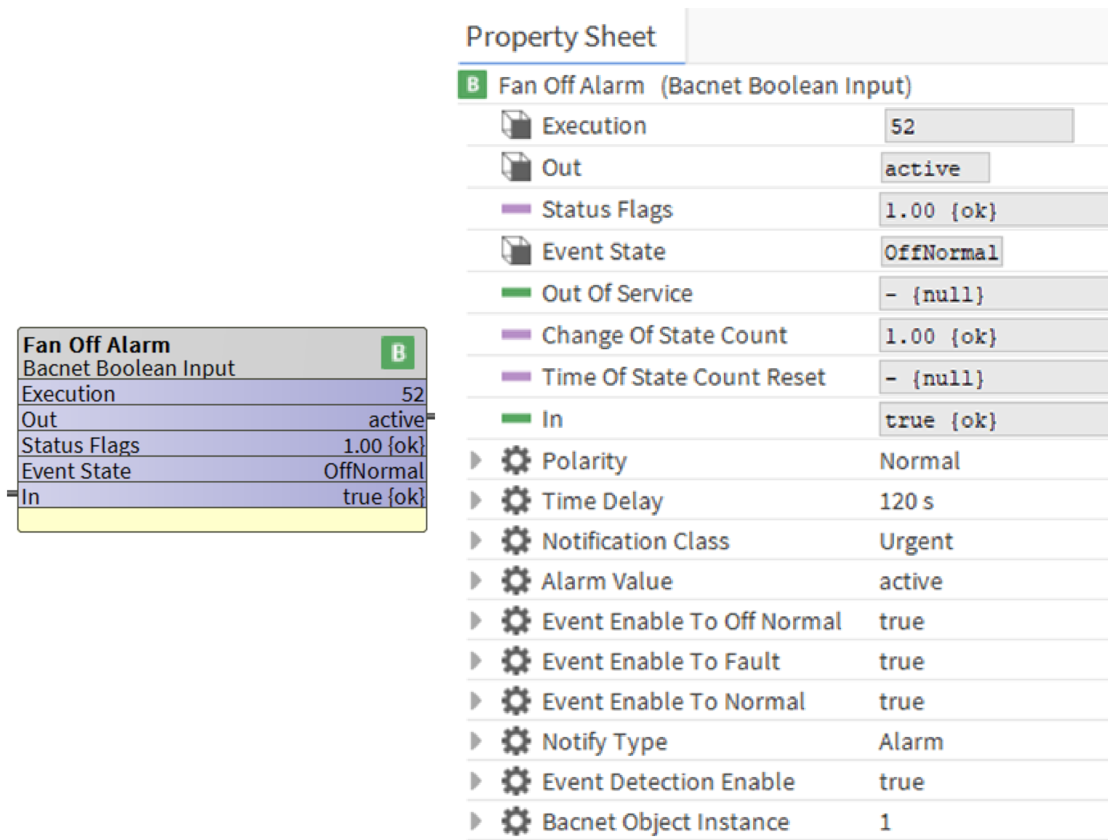


Fig. 43 BacnetBooleanInput Function Block and Property Sheet

Bacnet Numeric Output

The Bacnet Numeric Output exposes a raw object over BACnet as an ‘Analog Output’ BACnet object.

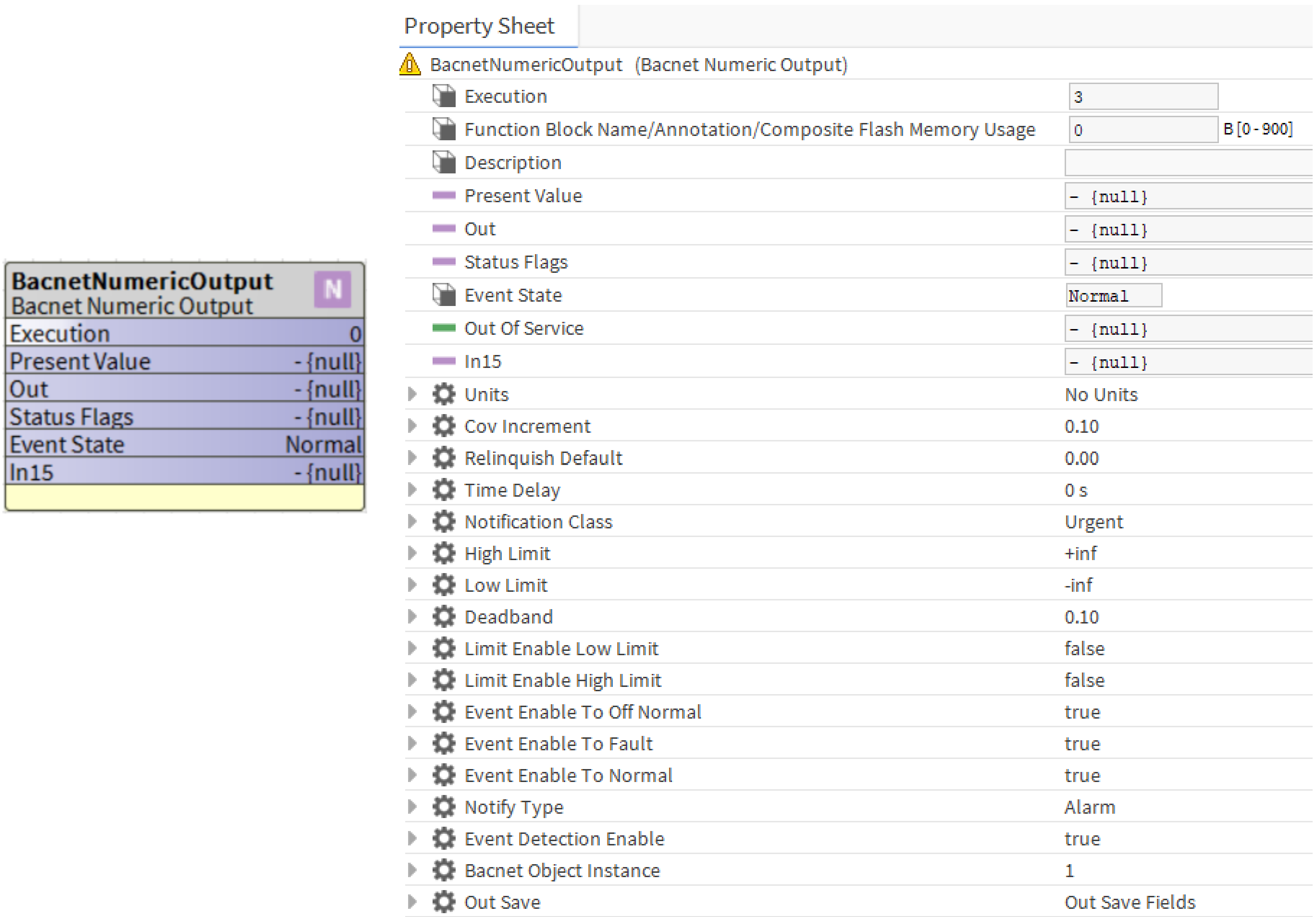


Fig. 44 BacnetNumericOutput Function Block and Property Sheet

Input

Table 30 Input of Bacnect Numeric Output

Input Name	Description
In 15	It is a 32-Bit floating input point.

Output

Table 31 Outputs of Bacnet Numeric Output

Output Name	Description
Present Value	Displays the highest priority valid value from '**Priority Array**.' 'BacnetNumericOutput' maintains a Priority Array of size 16, exposed over BACnet. • Index 1 is the highest and index 16 is the lowest. • 'In15' value shows up at Priority Array Index 15. • The Rest of the Index or Priority values can be updated over BACnet. • '**Present Value**' always reflects the first highest priority (from 1 to 16) valid value (not NULL).
Out	• When **Out Of Service** is "false"– '**Out**' is the same as '**Present Value**.' • When **Out Of Service** is "true"– '**Out**' retains the last value it had before '**Out Of Service**' was set to "true."
Status Flags	Displays the status of the function block. It can be any one or a 'OR' combination of the following values: **1:** In Alarm **2:** Fault **4:** Overwritten **8:** Out Of Service **Example:** • A value of 1 means the function block is in the '**In Alarm**' state. • A value of 9 means the function block is in '**Out Of Service**' as well as in the '**In Alarm**' state.
Event State	Displays the present event state of this function block. It can be any one of the following values: **0:** Normal **1:** Hight Limit **4:** Low Limit
Out Of Service	It shows whether the function block is in out of service state or not. '**Out Of Service**' is modifiable over BACnet. Setting '**Out Of Service**' to "true" decouples '**Present Value**' from '**Out**,' that is, '**Present Value**' value no longer appears on '**Out**.' When '**Out Of Service**' is "true," 'Out' will retain the last value that it had before '**Out Of Service**' was set to "true."

Parameter

Table 32 Parameters of Bacnet Numeric Output

Parameter Name	Description
Units	Defines the '**Out**' value's engineering unit.
Cov Increment	Defines the delta value change in '**Out**' that will trigger COV update notifications to other recipients on the BACnet bus.

Table 32 Parameters of Bacnet Numeric Output (Continued)

Parameter Name	Description
Relinquish Default	Defines the value that shows up on '**Out**' when 'In15' is not connected, and the priority array doesn't have any valid value.
Time Delay	Defines the time delay (in seconds); once conditions are met, the **High Limit** or **Low Limit** or **Normal** transition is set to off.
Notification Class	Defines the notification class from one of the following: • Urgent • High • Low • User Defined
High Limit	Defines the high limit. If the **Limit Enable High Limit** is defined as "true" and the **Out** value exceeds this limit, then the condition exists for at least the defined **Time Delay** period. • **Event State** is changed to **High Limit**. • **Status Flags** is set to **In Alarm**. • An event notification is sent to all recipients listed by the **Notification Class** (if '**Event Enable to Off Normal**' is defined true).
Low Limit	Defines the **Low Limit**. If the **Limit Enable Low Limit** is defined as "true" and the **Out** value fails below this limit, then the condition exists for at least the defined **Time Delay** period. • **Event State** is changed to **Low Limit**. • **Status Flags** is set to **In Alarm**. • An event notification is sent to all recipients listed by the **Notification Class** (if '**Event Enable to Off Normal**' is defined true).
Deadband	Defines the value of the deadband. While transitioning from an off-normal state (**High Limit** or **Low Limit**) to a normal state, '**Out**' value must remain between the range – (**Low Limit** + **Deadband**) and (**High Limit** – **Deadband**) for at least the defined **Time Delay** period. • **Event State** is changed to **Normal**. • **Status Flags** is cleared-off of **In Alarm**. • An event notification is sent to all recipients listed by the **Notification Class** (if '**Event Enable To Normal**' is defined true).
Limit Enable Low Limit	When set to "true," the low limit alarm is detected based on the configured **Low Limit** value.
Limit Enable High Limit	When set to "true," the high limit alarm is detected based on the configured **High Limit** value.
Event Enable To Off Normal	When set to "true," event notifications for **Off-Normal** event transitions are sent over BACnet to the recipients specified by the **Notification Class**.

Table 32 Parameters of Bacnet Numeric Output (Continued)

Parameter Name	Description
Event Enable To Fault	When set to "true," event notifications for Fault event transitions are sent over BACnet to recipients specified by the **Notification Class**.
Event Enable To Normal	When set to "true," event notifications for Normal event transitions are sent over BACnet to recipients specified by the **Notification Class**.
Notify Type	Defines the following notification types: • **Alarm:** Makes all the BACnet event notifications originating out of this instance of the function block, of type 'Alarm.' • **Event:** Makes all the BACnet event notifications originating out of this instance of the function block, of type 'Event.'
Event Detection Enable	Setting it to "true" enables: • Detection of **High Limit**/ **Low Limit/ Fault/ Normal** events in '**Event State**.' • Setting or resetting of **In Alarm/ Fault/ Overwritten/ Out Of Service**' flags in '**Status Flags**.' If it is set to "false': • **Event State** will always remain 'Normal.' • **Status Flags** will always remain 0.
Bacnet Object Instance	It shows the instance number of this Bacnet object function block.
Out Save	• **Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Present Value:** To enable or disable the Present value feature. • **Out**: To enable or disable the **Out** feature. ***Note:*** *The Out Save feature is only configurable via tools for Present Value and Out slot. As a result, after a power cycle, BACnet Output loses its priority array values, but the most recent OUT/Present Value is retained and appears on Priority Slot 16. This value, which belonged to which priority index prior to the power cycle event, will be lost.*

Example 1: BacnetNumericOutput (Cooling Valve Control)

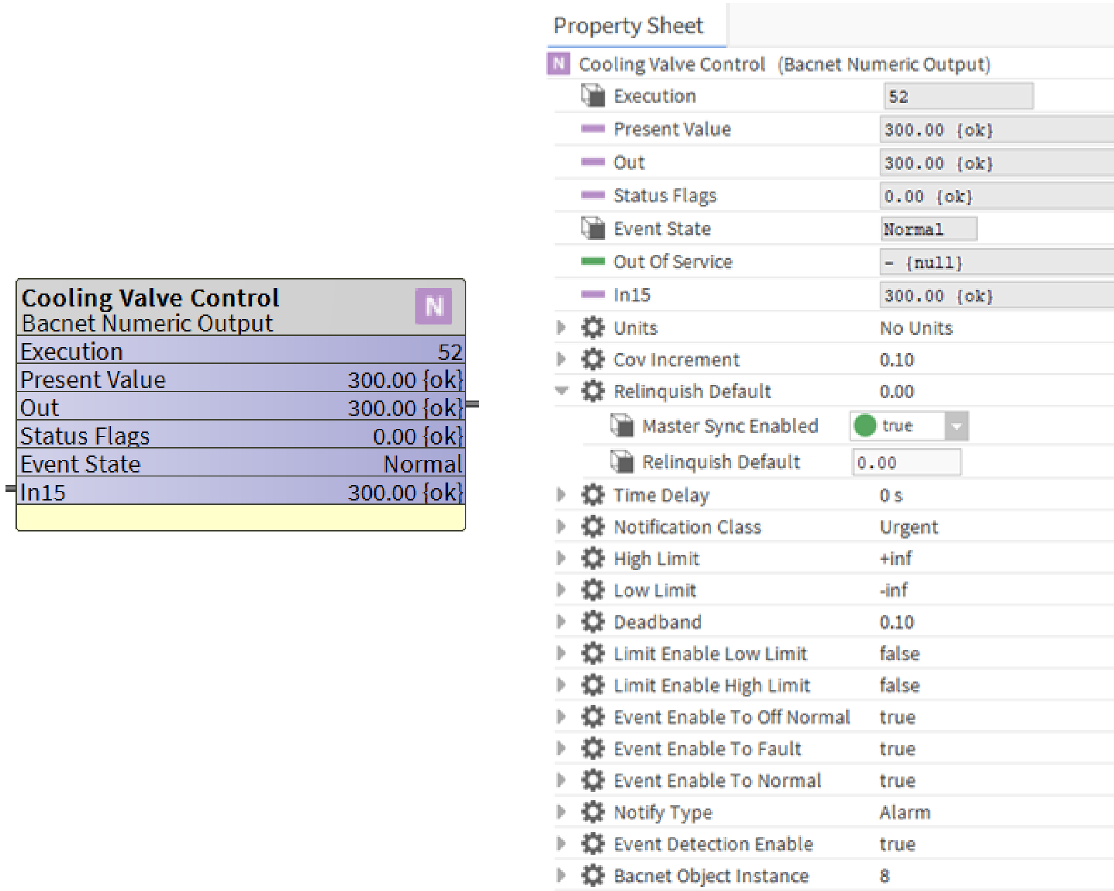


Fig. 45 BacnetNumericOutput Function Block and Property Sheet

Bacnet Boolean Output

The Bacnet Boolean Output exposes a boolean type raw object over BACnet as a ‘Binary Output’ BACnet object.

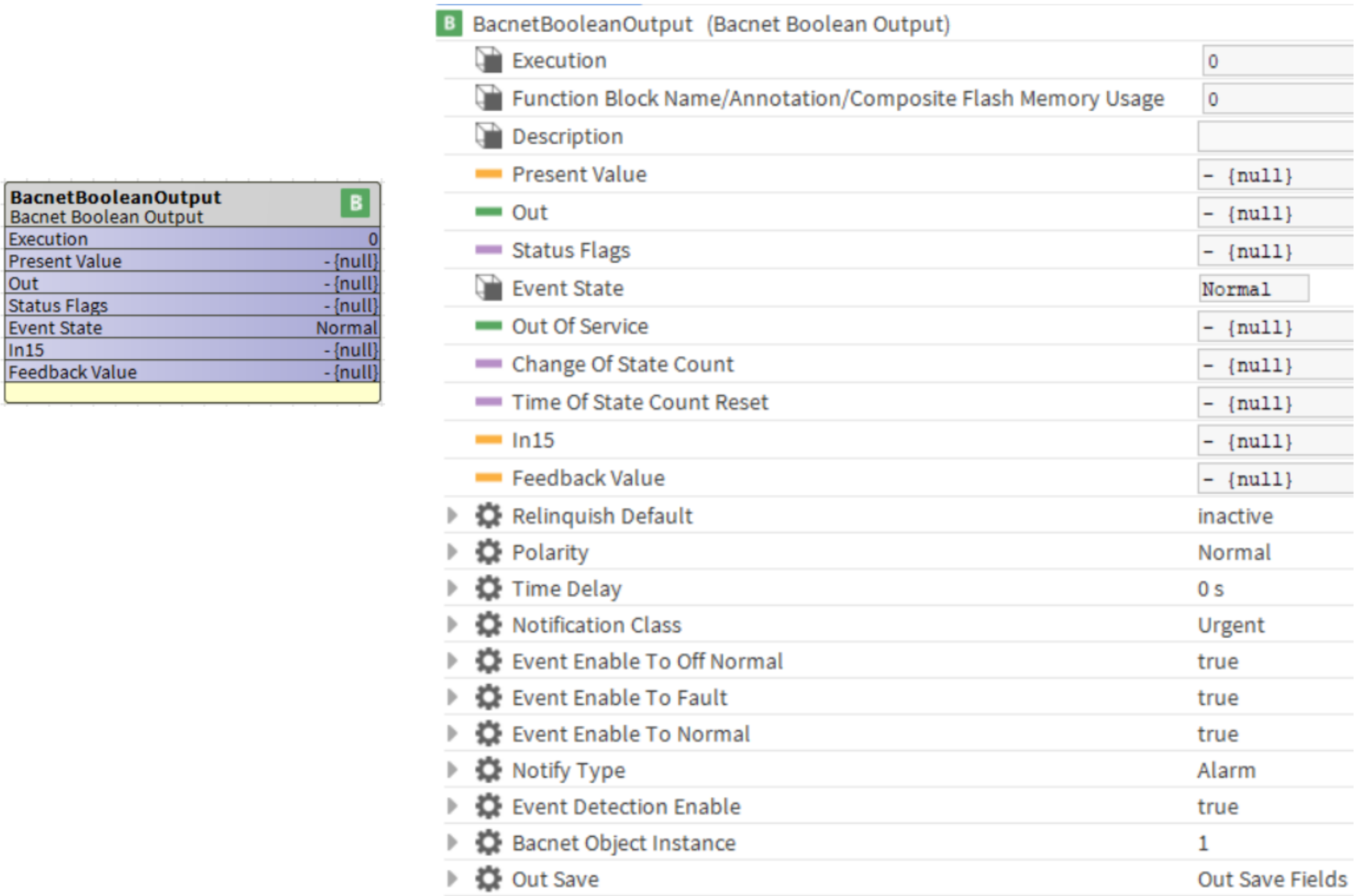


Fig. 46 BacnetBooleanOutput Function Block and Property Sheet

Input

Table 33 Inputs of Bacnet Boolean Output

Input Name	Description
In15	It is a boolean input point.
Feedback Value	The boolean value that gets compared against ‘**Out**.’ if unequal, then an ‘**Off-Normal**’ alarm state gets triggered.

Output

Table 34 Outputs of Bacnet Boolean Output

Output Name	Description
Present Value	Displays the highest priority valid value from ‘Priority Array.’ ‘BacnetBooleanOutput’ maintains a Priority Array of size 16, exposed over BACnet. • Index 1 is the highest, and index 16 is the lowest. • ‘In15’ value shows up at Priority Array Index 15. • The Rest of the Index or Priority values can be updated over BACnet. • ‘**Present Value**’ always reflects the first highest priority (from 1 to 16) valid value (not NULL).
Out	• When **Out Of Service** is False – ‘**Out**’ is the same as ‘**Present Value**.’ • When **Out Of Service** is “true”– ‘**Out**’ retains the last value it had before ‘**Out Of Service**’ was set to “true.”
Status Flags	Displays the status of this function block. It can be any one or a ‘OR’ combination of the following values: **1:** In Alarm **2:** Fault **4:** Overwritten **8:** Out Of Service **Example:** • A value of 1 means the function block is in the ‘**In Alarm**’ state. • A value of 9 means the function block is in ‘**Out Of Service**’ as well as in the ‘**In Alarm**’ state.
Event State	Displays the present event state of this function block. It can be any one of the following values: **0:** Normal **1:** Fault **2:** Off Normal
Out Of Service	It shows whether the function block is in out of service state or not. • ‘**Out Of Service**’ is modifiable over BACnet. Setting ‘**Out Of Service**’ to true decouples ‘**Present Value**’ from ‘**Out**,’ that is, ‘**Present Value**’ no longer shows up on ‘**Out**.’ • When ‘**Out Of Service**’ is true, ‘**Out**’ will retain the last value which was before ‘**Out Of Service**’ was set to True.
Change of State Time	Displays the last time ‘**Out**’ state was changed.
Change of State Count	Displays the total count of ‘**Out**’ state changes. This is modifiable over BACnet.
Time of State Count Reset	Displays the last time “**Change of State Count**” was reset over BACnet.

Parameter

Table 35 Parameters of Bacnet Boolean Output

Parameter Name	Description
Relinquish Default	Defines the value that appears on **Out** when 'In15' is not connected, and the priority array doesn't have any valid value.
Polarity	Defines the relationship between '**Present Value**' and '**Out**' options are: • **Normal:** '**Out**' is the same as '**Present Value**,' that is, if '**Present Value**' is 'active,' then '**Out**' is 'active.' • **Reverse:** '**Out**' is NOT of '**Present Value**,' that is, if '**Present Value**' is 'active,' then '**Out**' is 'inactive.'
Time Delay	Defines the time delay (in seconds); once conditions are met, the Off Normal or Normal transition is set to off.
Notification Class	There are four notification classes: • Urgent • High • Low • User Defined
Event Enable To Off Normal	When set to "true," event notifications for **Off-Normal** event transitions are sent over BACnet to the recipients specified by the **Notification Class**.
Event Enable to Fault	When set to "true," event notifications for Fault event transitions are sent over BACnet to recipients specified by the **Notification Class**.
Event Enable to Normal	When set to "true," event notifications for Normal event transitions are sent over BACnet to recipients specified by the **Notification Class**.
Notify Type	Defines the following notification types: • **Alarm:** Makes all the BACnet event notifications originating out of this instance of the function block, of type 'Alarm.' • **Event:** Makes all the BACnet event notifications originating out of this instance of the function block, of type 'Event.'
Event Detection Enable	Setting it to "true" enables: • Detection of **Off-Normal/ Fault/ Normal** events in **Event State**. • Setting or resetting of **In Alarm/ Fault/ Overwritten**/ **Out Of Service** flags in **Status Flags**. If it is set to "false": • **Event State** will always remain Normal. • **Status Flags** will always remain 0.
Bacnet Object Instance	It shows the instance number of this BACnet object function block.

Table 35 Parameters of Bacnet Boolean Output (Continued)

Parameter Name	Description
Out Save	• **Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Present Value:** To enable or disable the **Present Value** feature. • **Out**: To enable or disable the **Out** feature. ***Note:*** *The Out Save feature is only configurable via tools for Present Value and Out slot. As a result, after a power cycle, BACnet Output loses its priority array values, but the most recent OUT/Present Value is retained and appears on Priority Slot 16. This value, which belonged to which priority index prior to the power cycle event, will be lost.*

Example 1: BacnetBooleanOutput (Damper Command)

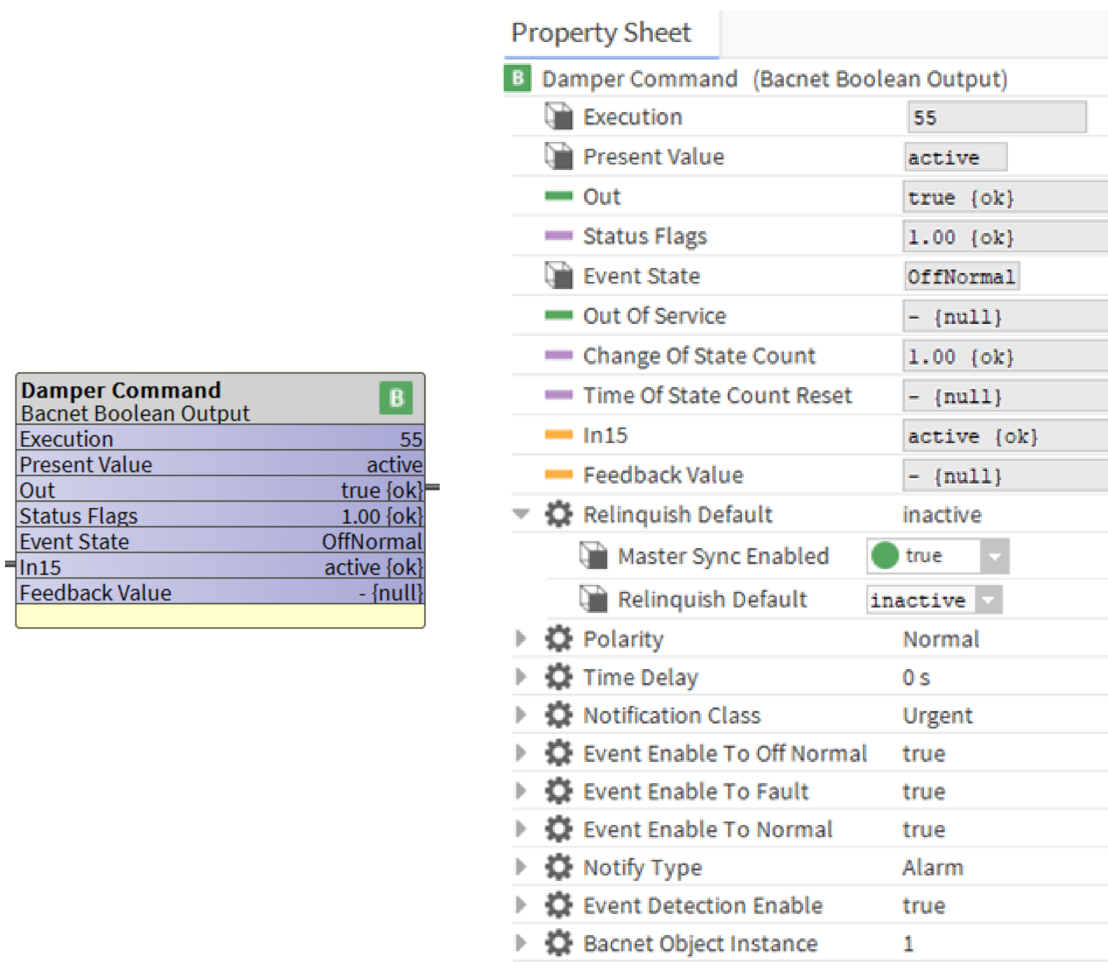


Fig. 47 BacnetBooleanOutput Function Block and Property Sheet

Bacnet Enum Output

The Bacnet Enum Output converts input value(s) (Public Variable(s)) into a raw network variable output that is published onto the BACnet network.

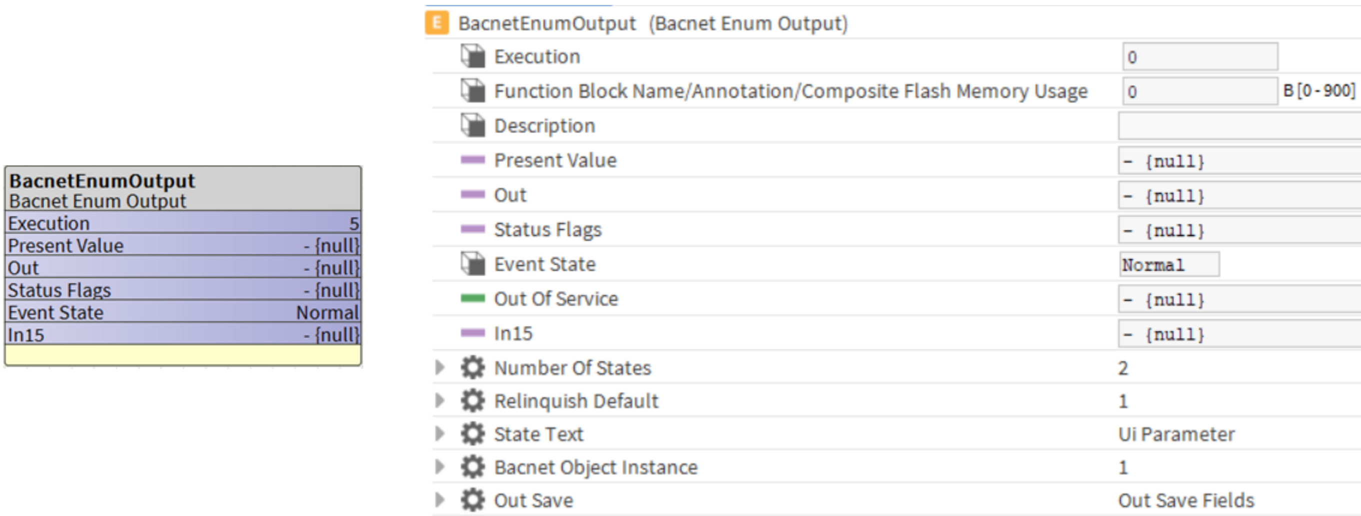


Fig. 48 BacnetEnumOutput Function Block and Property Sheet

Input

Table 36 Input of Bacnet Enum Output

Input Name	Description
In15	It is a 32-Bit unsigned input point.

Output

Table 37 Outputs of Bacnet Enum Output

Output Name	Description
Preset Value	Displays the highest priority valid value from ‘Priority Array.’ ‘BacnetEnumOutput’ maintains a Priority Array of size 16, exposed over BACnet. • Index 1 is the highest, and index 16 is the lowest. • ‘In15’ value shows up at Priority Array Index 15. • The Rest of the Index or Priority values can be updated over BACnet. • **Present Value** always reflects the first highest priority (from 1 to 16) valid value (not NULL).
Out	• When **Out Of Service** is “false”– ‘**Out**’ is the same as ‘**Present Value**.’ • When **Out Of Service** is “true”– ‘**Out**’ retains the last value it had before ‘**Out Of Service**’ was set to True.

Table 37 Outputs of Bacnet Enum Output (Continued)

Output Name	Description
Status Flags	Displays the status of this function block. It can be any one or a 'OR' combination of the following values: **8:** Out Of Service **Example:** • A value of 8 means the function block is in the '**Out Of Service**' state. • A value of 3 means the function block is not in the '**Out Of Service**' state.
Event State	Displays the present event state of this function block. It will always be in the following state: **0:** Normal
Out Of Service	It shows whether the function block is in out of service state or not. • '**Out Of Service**' is modifiable over BACnet. Setting '**Out Of Service**' to "true" decouples '**Present Value**' from '**Out**,' that is, '**Present Value**' no longer shows up on '**Out**.' • When '**Out Of Service**' is "true," '**Out**' will retain the last value that it had before '**Out Of Service**' was set to True.

Parameter

Table 38 Parameters of Bacnet Enum Output

Parameter Name	Description
Number Of States	Defines the total number of discreet states starting from 1 that '**Out**' can have.
Relinquish Default	Defines the value that appears on **Out** when 'In15' is not connected and the priority array doesn't have any valid value.
State text	Facets define enumeration values and the associated texts for BACnet data points. The enumeration values can be defined in the range 1 - 16.
Bacnet Object Instance	It shows the instance number of this BACnet object function block.

Table 38 Parameters of Bacnet Enum Output (Continued)

Parameter Name	Description
Out Save	• **Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Present Value:** To enable or disable the **Present Value** feature. • **Out:** To enable or disable the **Out** feature. ***Note:*** *The Out Save feature is only configurable via tools for Present Value and Out slot. As a result, after a power cycle, BACnet Output loses its priority array values, but the most recent OUT/Present Value is retained and appears on Priority Slot 16. This value, which belonged to which priority index prior to the power cycle event, will be lost.*

Example 1: BacnetEnumOutput (Fan Speed)

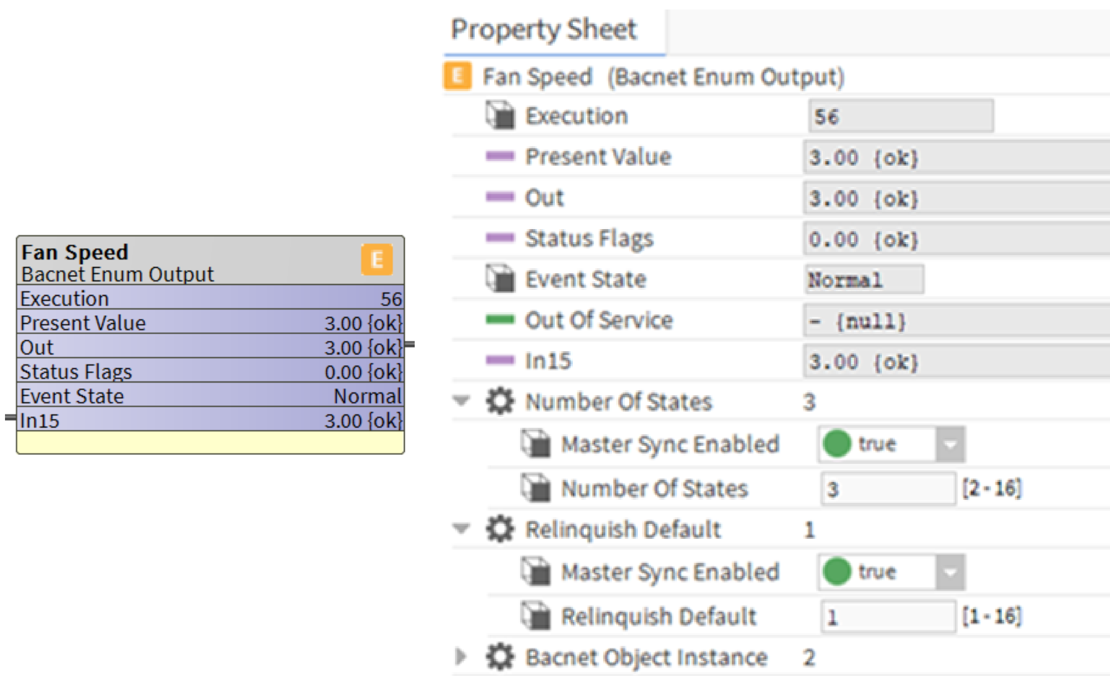


Fig. 49 BacnetEnumOutput Function Block and Property Sheet

Bacnet Numeric Value

The Bacnet Numeric Value exposes a raw object over BACnet as an ‘Analog Value’ BACnet object.

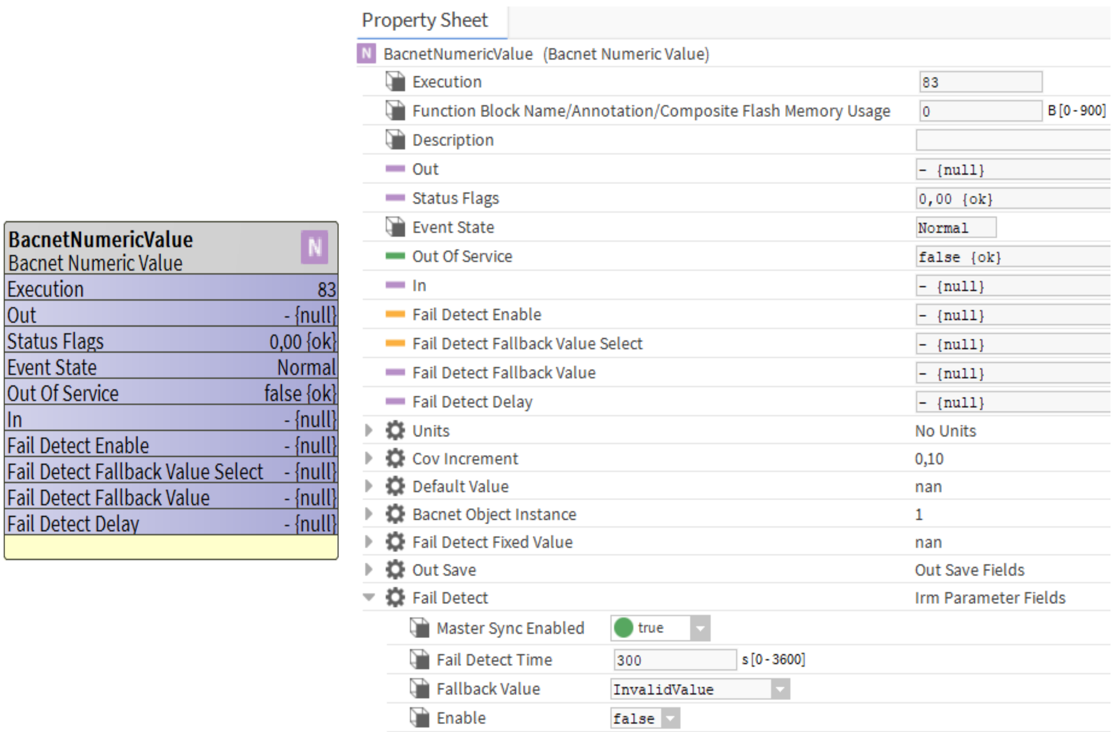


Fig. 50 BacnetNumericValue Function Block and Property Sheet

Inputs

Table 39 Input of Bacnet Numeric Value

Input Name	Description
In	It is a 32-Bit floating input point.
Fail Detect Enable	**0:** Disable **1:** Enable Set it to true to enable the fail detection feature. ***Note:*** *This feature enables the monitoring of periodic updates of a function block over a network.*
Fail Detect Fallback Value Select	**0:** InvalidValue **1:** LastknownGoodValue **2:** FixedValue InvalidValue: ‘Out’ is set as NULL LastknownGoodValue: If the last ‘Out’ value was not NULL, then ‘Out’ is retained as it is; otherwise, a value from the ‘Default Value’ parameter goes to Out. FixedValue: “Fail Detect Fixed Value” goes to ‘Out.’

Table 39 Input of Bacnet Numeric Value (Continued)

Input Name	Description
Fail Detect Fallback Value	Define the value that should be set to ‘Out’ (when failure is detected), when “Fail Detect Enable” is True, and “Fail Detect Fallback Value” is set to "Fixed Value." It is a 32-bit floating point value.
Fail Detect Delay	Update interval in seconds within which ‘Present Value’ should get written over BACnet periodically. ‘Present Value’ not written within this interval will result in failure, and ‘Out’ will be set with the value as configured in ‘Fail Detect Fallback Value Select.’ It is a 32 bit integer value with a range of 0 to 3600 sec.

Output

Table 40 Outputs of Bacnet Numeric Value

Output Name	Description
Out	• When **Out Of Service** is “false” – ‘**Out**’ is the same as ‘**In**’ if ‘**In**’ is connected and its value is not NULL. • When ‘**In**” is NULL, ‘**Out**’ takes value from the ‘Default Value’ parameter. This value will also be exposed as ‘**Present Value**’ over BACnet. ‘**Out**’ will change when ‘**Present Value**’ is changed over BACnet, or the ‘Default Value’ parameter is changed via Tool. • When **Out Of Service** is “true” – ‘**Out**’ is decoupled from ‘**In**.’ ‘**Out**’ is the ‘**Present Value**’ written over BACnet.
Status Flags	Displays the status of this function block. It can be 0 or the following value: **2:** Fault **8:** Out Of Service **Example:** • A value of 8 means the function block is in the ‘**Out Of Service**’ state. • A value of 2 means the function block is in the ‘Fault’ state. • A value of 0 means the function block is not in the ‘**Out Of Service**’ state.
Event State	Displays the present event state of this function block. It will always be in the following state: **0:** Normal
Out Of Service	It shows whether the function block is in out of service state or not. • ‘**Out Of Service**’ is modifiable over BACnet. Setting ‘**Out Of Service**’ to “true,” decouples ‘**In**’ from ‘**Out**,’ that is, ‘**In**’ values no longer show up on ‘**Out**.’ • When ‘**Out Of Service**’ is “true,” ‘**Out**’ can be overwritten over BACnet (via **Present Value** property).

Parameter

Table 41 Parameters of Bacnet Numeric Value

Parameter Name	Description
Units	Defines the '**Out**' value's engineering unit.
Cov Increment	Defines the delta value change in '**Out**' that will trigger COV update notifications to other recipients on the BACnet bus.
Default Value	Defines the value that appears on '**Out**' at controller startup when '**In**' is not connected.
Bacnet Object Instance	It shows the instance number of this BACnet object function block.
Fail Detect	**Fail Detect** only works if "**In**" is NULL and "Fail Detect Enable" is set to true. • **Fail Detect Time**: Update interval in seconds within which '**Present Value**' should get written over BACnet periodically. '**Present Value**' not written within this interval will result in failure, and '**Out**' will be set with the value as configured in 'Fallback Value.' • **Fail Detect Value:** Select the Fail Detect Value type. '**InvalidValue**': '**Out**' is set as NULL '**LastknownGoodValue**': If the last '**Out**' value was not NULL, then '**Out**' is retained as it is; otherwise, a value from the 'Default Value' parameter goes to **Out**. '**FixedValue**' - "Fail Detect Fixed Value" goes to '**Out**.' • **Enable**: Set it to true to enable the fail detection feature. ***Note:*** *This feature enables the monitoring of periodic updates of a function block over a network.*
Fail Detect Fixed Value	Define the value that should be set to '**Out**' (when failure is detected), when "Fail Detect Enable" is True, and "Fail Detect Fallback Value" is set to "Fixed Value."
Out Save	• **Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. If the present value is changed via BACnet , then the same value will be set to present value after power cycle. If there is no value saved then first time it will use the default value. • **Out**: To enable or disable the **Out** feature.

Example 1: BacnetNumericValue (Temperature Effective Setpoint)

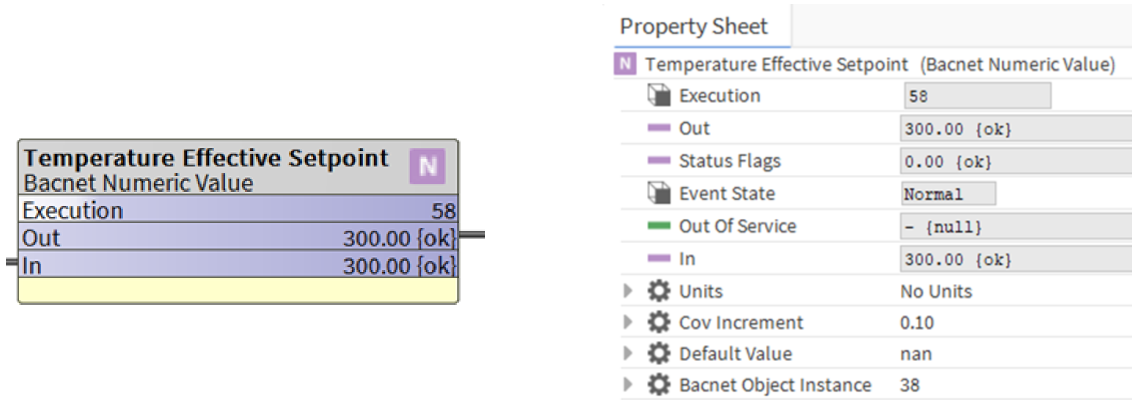


Fig. 51 BacnetNumericValue Function Block and Property Sheet

Bacnet Boolean Value

The Bacnet Boolean Value exposes a boolean type raw object over BACnet as a ‘Binary Value’ BACnet object.

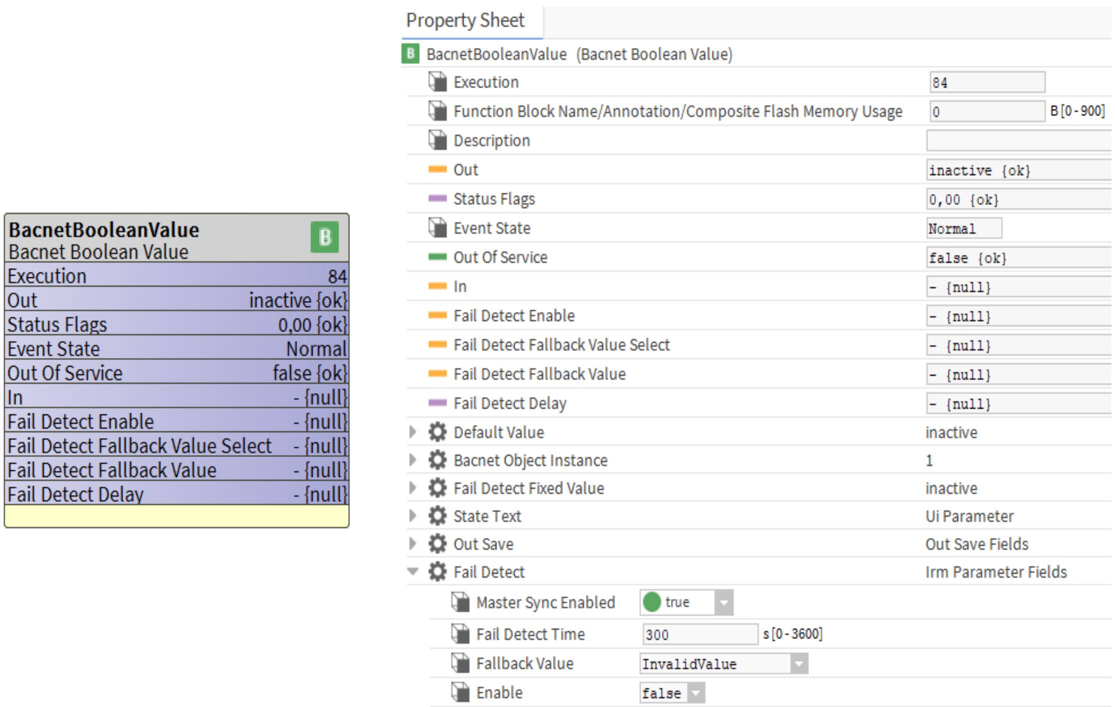


Fig. 52 BacnetBooleanValue Function Block and Property Sheet

Input

Table 42 Inputs of Bacnet Boolean Value

Input Name	Description
In	It is a boolean input point.
Fail Detect Enable	**0:** Disable **1:** Enable Set it to true to enable the fail detection feature. ***Note:*** *This feature enables the monitoring of periodic updates of a function block over a network.*
Fail Detect Fallback Value Select	**0:** InvalidValue **1:** LastknownGoodValue **2:** FixedValue InvalidValue: ‘Out’ is set as NULL LastknownGoodValue: If the last ‘Out’ value was not NULL, then ‘Out’ is retained as it is; otherwise, a value from the ‘Default Value’ parameter goes to Out. FixedValue: “Fail Detect Fixed Value” goes to ‘Out.’

Table 42 Inputs of Bacnet Boolean Value (Continued)

Input Name	Description
Fail Detect Fallback Value	Define the value that should be set to ‘Out’ (when failure is detected), when “Fail Detect Enable” is True, and “Fail Detect Fallback Value” is set to "Fixed Value." 0 = false, 1 = True. It is a boolean point value.
Fail Detect Delay	Update interval in seconds within which ‘Present Value’ should get written over BACnet periodically. ‘Present Value’ not written within this interval will result in failure, and ‘Out’ will be set with the value as configured in ‘Fail Detect Fallback Value Select.’ It is a 32 bit integer value with a range of 0 to 3600 sec.

Output

Table 43 Outputs of Bacnet Boolean Value

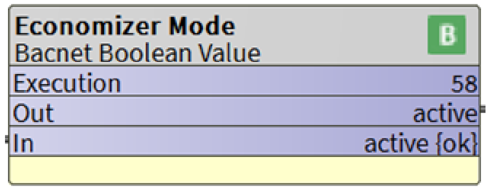
Output Name	Description
Out	• When **Out Of Service** is “false” – ‘**Out**’ is the same as ‘**In**’ if ‘**In**’ is connected and its value is not NULL. • When ‘**In**” is NULL, ‘**Out**’ takes value from the ‘Default Value’ parameter. This value will also be exposed as ‘**Present Value**’ over BACnet. ‘**Out**’ will change when ‘**Present Value**’ is changed over BACnet, or the ‘Default Value’ parameter is changed via Tool. • When **Out Of Service** is “true” – ‘**Out**’ is decoupled from ‘**In**.’ ‘**Out**’ is the ‘**Present Value**’ written over BACnet.
Status Flags	Displays the status of this function block. It can be 0 or the following value: **2:** Fault **8:** Out Of Service **Example:** • A value of 8 means the function block is in the ‘**Out Of Service**’ state. • A value of 2 means the function block is in the ‘Fault’ state. • A value of 0 means the function block is not in the ‘**Out Of Service**’ state.
Event State	Displays the present event state of this function block. It will always be in the following state – **0: Normal**
Out Of Service	It shows whether the function block is in out of service state or not. • **‘Out Of Service**’ is modifiable over BACnet. Setting ‘**Out Of Service**’ to “true,” decouples ‘**In**’ from ‘**Out**,’ that is, ‘**In**’ values no longer show up on ‘**Out**.’ • When ‘**Out Of Service**’ is “true,” ‘**Out**’ can be overwritten over BACnet (via **Present Value** property).

Parameter

Table 44 Parameters of Bacnet Boolean Value

Parameter Name	Description
Default Value	Defines the value of ‘**Out**’ at controller startup when ‘**In**’ is not connected.
Bacnet Object Instance	It shows the instance number of this Bacnet object function block.
Fail Detect	**Fail Detect** only works if “**In**” is NULL and “Fail Detect Enable” is set to true. • **Fail Detect Time**: Update interval in seconds within which ‘**Present Value**’ should get written over BACnet periodically. ‘**Present Value**’ not written within this interval will result in failure, and ‘**Out**’ will be set with the value as configured in ‘Fallback Value.’
	• **Fail Detect Value:** Select the Fail Detect Value type. ‘**InvalidValue**’: ‘**Out**’ is set as NULL ‘**LastknownGoodValue**’: If the last ‘**Out**’ value was not NULL, then ‘**Out**’ is retained as it is; otherwise, a value from the ‘Default Value’ parameter goes to **Out**. ‘**FixedValue**’: “Fail Detect Fixed Value” value goes to ‘**Out**.’ • **Enable**: Set it to true to enable fail detection feature. ***Note:*** *This feature enables the monitoring of periodic update sof a function block over a network.*
Fail Detect Fixed Value	Define the value that should be set to ‘**Out**’ (when failure is detected), when “Fail Detect Enable” is True, and “Fail Detect Fallback Value” is set to "Fixed Value."
State text	Facets define enumeration values and the associated texts for BACnet data points. The enumeration values can be defined in the range 1-16.
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. If the present value is changed via BACnet , then the same value will be set to present value after power cycle. If there is no value saved then first time it will use the default value. • **Out**: To enable or disable the **Out** feature.

Example 1: BacnetBooleanValue (Economizer Mode)



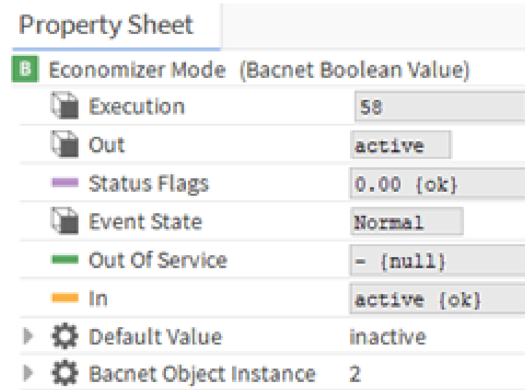


Fig. 53 BacnetBooleanValue Function Block and Property Sheet

Bacnet Enum Value

The Bacnet Enum Value exposes a raw object over BACnet as a ‘Multi-State Value’ BACnet object.

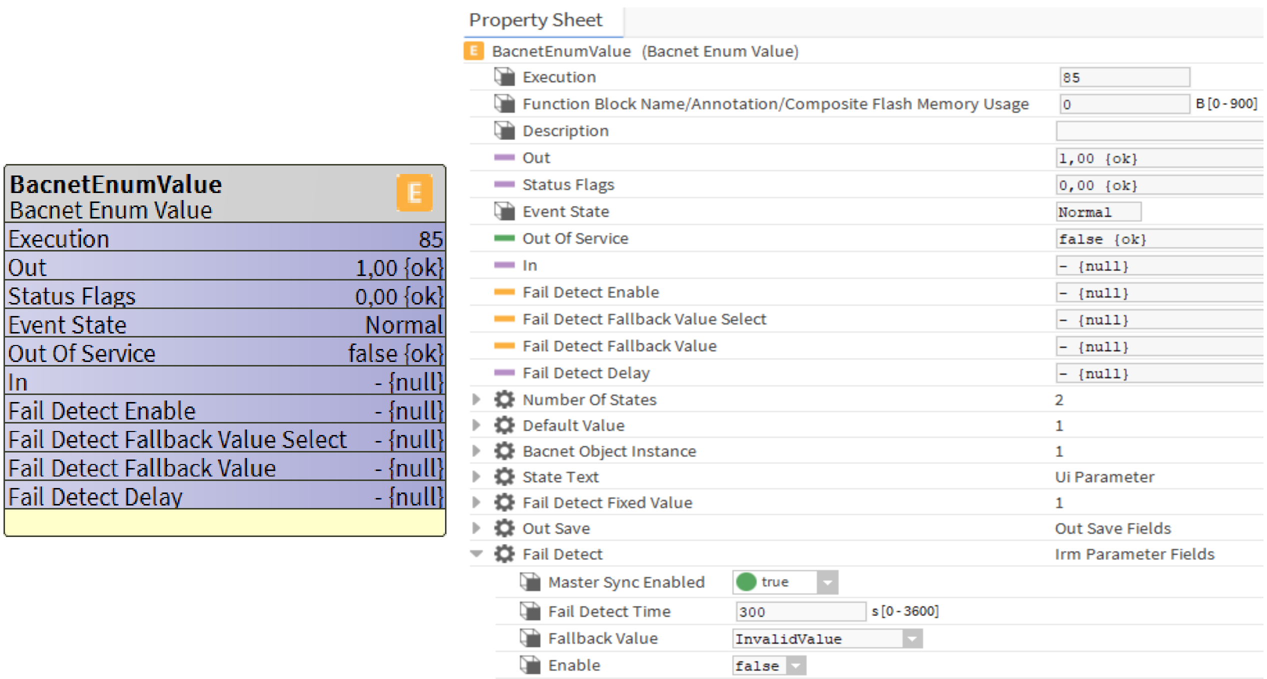


Fig. 54 BacnetEnumValue Function Block and Property Sheet

Input

Table 45 Inputs of Bacnet Enum Value

Input Name	Description
In	It is a 32-Bit floating input point.
Fail Detect Enable	**0:** Disable **1:** Enable Set it to true to enable the fail detection feature. ***Note:*** *This feature enables the monitoring of periodic updates of a function block over a network.*
Fail Detect Fallback Value Select	**0:** InvalidValue **1:** LastknownGoodValue **2:** FixedValue InvalidValue: ‘Out’ is set as NULL LastknownGoodValue: If the last ‘Out’ value was not NULL, then ‘Out’ is retained as it is; otherwise, a value from the ‘Default Value’ parameter goes to Out. FixedValue: “Fail Detect Fixed Value” goes to ‘Out.’

Table 45 Inputs of Bacnet Enum Value (Continued)

Input Name	Description
Fail Detect Fallback Value	Define the value that should be set to 'Out' (when failure is detected), when "Fail Detect Enable" is True, and "Fail Detect Fallback Value" is set to "Fixed Value." It is a Enum value with a range of 1 to 16.
Fail Detect Delay	Update interval in seconds within which 'Present Value' should get written over BACnet periodically. 'Present Value' not written within this interval will result in failure, and 'Out' will be set with the value as configured in 'Fail Detect Fallback Value Select.' It is a 32-bit integer value with range 0 to 3600 sec.

Output

Table 46 Outputs of Bacnet Enum Value

Output Name	Description
Out	• When **Out Of Service** is "false" – '**Out**' is the same as '**In**' if '**In**' is connected and its value is not NULL. • When '**In**" is NULL, '**Out**' takes value from the 'Default Value' parameter. This value will also be exposed as '**Present Value**' over BACnet. '**Out**' will change when '**Present Value**' is changed over BACnet, or the 'Default Value' parameter is changed via Tool. • When **Out Of Service** is "true" – '**Out**' is decoupled from '**In**.' '**Out**' is the '**Present Value**' written over BACnet.
Status Flags	Displays the status of this function block. It can be 0 or the following value: **2:** Fault **8:** Out Of Service **Example:** • A value of 8 means the function block is in the '**Out Of Service**' state. • A value of 2 means the function block is in the 'Fault' state. • A value of 0 means the function block is not in the '**Out Of Service**' state.
Event State	Displays the present event state of this function block. It will always be in the following state: **0:** Normal
Out Of Service	It shows whether the function block is in out of service state or not. • **'Out Of Service**' is modifiable over BACnet. Setting '**Out Of Service**' to "true," decouples '**In**' from '**Out**,' that is, '**In**' values no longer show up on '**Out**.' • When '**Out Of Service**' is "true," '**Out**' can be overwritten over BACnet (via **Present Value** property).

Parameter

Table 47 Parameters of Bacnet Enum Value

Parameter Name	Description
Number Of States	Defines the total number of discreet states starting from 1 that '**Out**' can have.
Default Value	Defines the value of '**Out**' at controller startup when '**In**' is not connected.
Bacnet Object Instance	It shows the instance number of this BACnet object function block.
State text	Facets define enumeration values and the associated texts for BACnet data points. The enumeration values can be defined in the range 1-16.
Fail Detect Fixed Value	Define the value that should be set to '**Out**' (when failure is detected), when "Fail Detect Enable" is True, and "Fail Detect Fallback Value" is set to "Fixed Value."
Out Save	• **Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. If the present value is changed via BACnet , then the same value will be set to present value after power cycle. If there is no value saved then first time it will use the default value. • **OUT**: To enable or disable the **Out** feature.
Fail Detect	**Fail Detect** only works if "**In**" is NULL and "Fail Detect Enable" is set to true. • **Fail Detect Time:** Update interval in seconds within which '**Present Value**' should get written over BACnet periodically. '**Present Value**' not written within this interval will result in failure, and '**Out**' will be set with the value as configured in 'Fallback Value.' • **Fail Detect Value:** Select the Fail Detect Value type. '**InvalidValue':** '**Out**' is set as Null. '**LastknownGoodValue':** If the last '**Out**' value was not NULL, then '**Out**' is retained as it is; otherwise, a value from the 'Default Value' parameter goes to **Out**. '**FixedValue':** "Fail Detect Fixed Value" goes to '**Out**.' • **Enable:** Set it to true to enable fail detection feature. ***Note:*** *This feature enables the monitoring of periodic updates of a function block over a network.*

Example 1: BacnetEnumValue (Effective Control Mode)

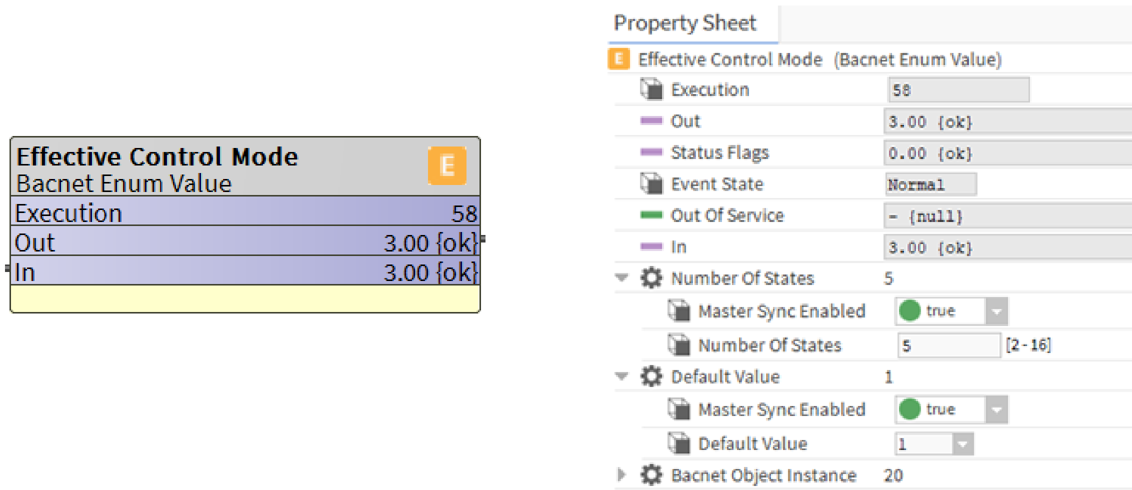


Fig. 55 BacnetEnumValue Function Block and Property Sheet

Ref In

The Ref In function block makes it easier to get a value from a BACnet object on a remote BACnet device. RefIn subscribes to or reads the BACnet object of the remote BACnet device on a regular basis once it has been correctly mapped to the remote BACnet object.

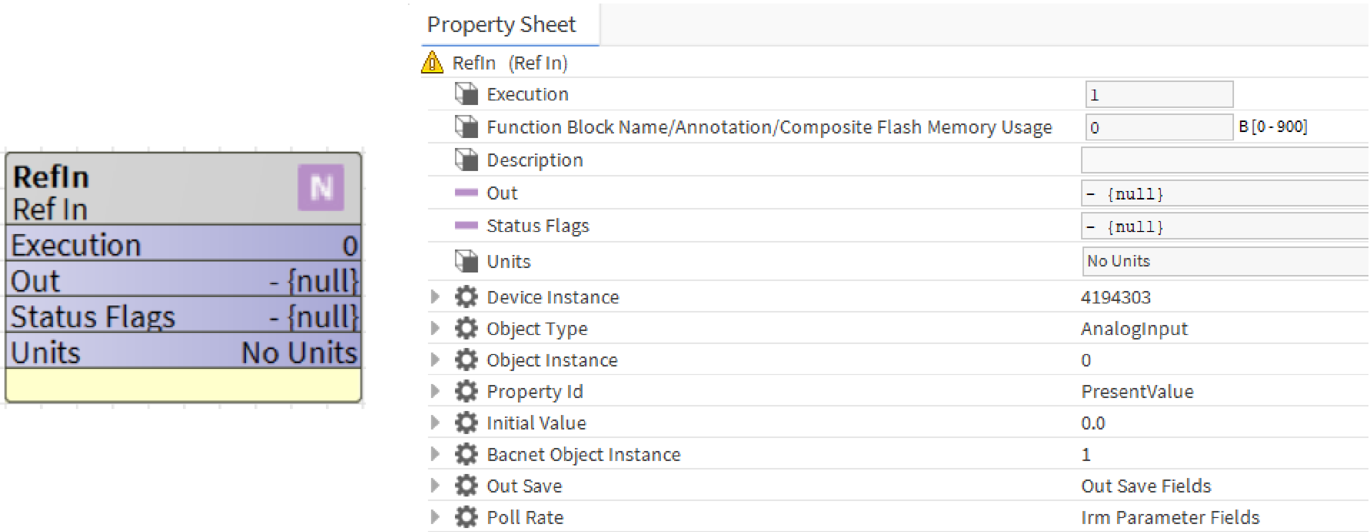


Fig. 56 RefIn Function Block and Property Sheet

Input

Table 48 Input of RefIn

Input Name	Description
NA	NA

Output

Table 49 Outputs of RefIn

Output Name	Description
Out	• When **Out Of Service** is "false"– **Out** is the last read value of the linked BACnet object property. • When **Out Of Service** is "true"– **Out** retains the last value it had before **Out Of Service** was set to True.
Status Flags	Displays the status of this function block. It can be any one or a 'OR' combination of the following values: **2:** Fault **8:** Out Of Service **Example:** A value of 2 means the function block is in the '**In Alarm**' state.
Units	Displays the unit of the value retrieved from the remote BACnet object.

Parameter

Table 50 Parameters of RefIn

Input Name	Description
Device Instance	Defines the remote BACnet device instance, whose BACnet object needs to be linked to this RefIn.
Object Type	Defines the type of the BACnet object that needs to be linked. Nine types of Object Type are presently supported: • AnalogInput • AnalogOutput • AnalogValue • BinaryInput • BinaryOutput • BinaryValue • MultistateInput • MultistateOutput • MultistateValue
Object Instance	Defines the instance number of the BACnet object to be linked to this RefIn.
Property Id	Defines the property id to be read. Right now, the **Present Value** is only supported.
Initial Value	Defines the initial value at controller startup that should show up on **Out** before the first successful poll.
Bacnet Object Instance	Shows the instance number of this BACnet object.
Out Save	• **Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out**: To enable or disable the **Out** feature.

Table 50 Parameters of RefIn (Continued)

Input Name	Description
Poll Rate	**Mode:** Defines whether RefIn shall subscribe COV for the linked object's property value or it would do Polling. **Poll Rate:** Define how frequently (in minutes) RefIn should poll for this linked object's property value when Mode is defined as Polling. • If Mode is defined as COV, RefIn subscribes for COV (change of value) of the property value for the linked BACnet object. • If successful, then, as and when linked object property value changes, the linked object itself asynchronously sends COV updates to RefIn. In this mode, RefIn renews its COV subscription every 15 minutes. • If the COV subscription request fails, RefIn automatically fallback to Polling mode internally. • If Mode is defined as Polling, RefIn reads the property value of the linked BACnet object every poll interval as defined in Poll Rate. **Poll Rate Unit**: Defines the Poll Rate unit.

Example 1: RefIn

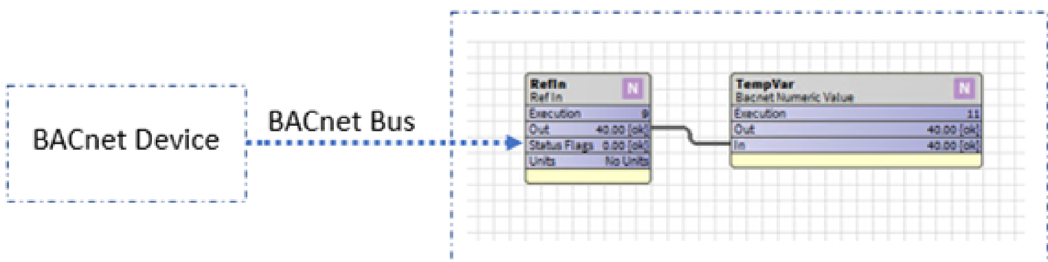


Fig. 57 RefIn Function Block

Ref Out

The Ref Out function block allows a BACnet object on a remote BACnet device to be updated with new values. The Ref Out update to the BACnet object value of the remote BACnet device on a regular basis after being correctly mapped to it.

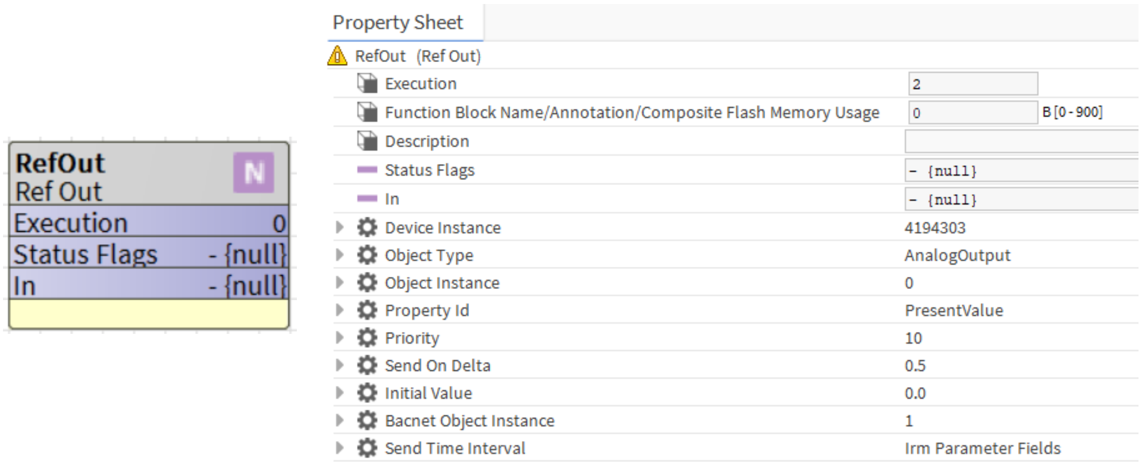


Fig. 58 RefOut Function Block and Property Sheet

Input

Table 51 Inputs of RefOut

Input Name	Description
In	It can be a 32-Bit floating or unsigned or a boolean input point.

Output

Table 52 Outputs of RefOut

Output Name	Description
Status Flags	Displays the status of this function block. It can be any one or a 'OR' combination of the following values: **2:** Fault **8:** Out Of Service **Example:** A value of 2 means the function block is in the '**In Alarm**' state.

Parameter

Table 53 Parameters of RefOut

Input Name	Description
Device Instance	Defines the remote BACnet device instance, whose BACnet object needs to be linked to this 'RefOut.'

Table 53 Parameters of RefOut (Continued)

Input Name	Description
Object Type	Defines the type of the BACnet object that needs to be linked. Six types of 'Object Type' are presently supported: • AnalogOutput • AnalogValue • BinaryOutput • BinaryValue • MultistateOutput • MultistateValue
Object Instance	Defines the instance number of the BACnet object to be linked to this 'RefOut.'
Property Id	Defines the property id to be updated. Right now, the present value is only supported
Priority	Defines the priority for the update requests (Bacnet WriteProperty) initiated on the linked remote BACnet object property. • Valid values are 1 to 16. • Default is set to 10. • If priority is 0, the write requests are initiated without any priority.
Send On Delta	Defines the delta change in the '**In**' value that would prompt an immediate update (via Bacnet Write Property requests) of the '**In**' value to the linked BACnet object's property.
Initial Value	Defines the initial value at startup that gets written to the linked object's property value when '**In**' is not connected.
Bacnet Object Instance	It shows the instance number of this Bacnet object.
Send Time Interval	Defines how frequently (in minutes) the '**In**' value would get updated (via Bacnet Write Property requests) to the linked BACnet object's property when '**In**' is in a stable state.

Example 1: RefOut

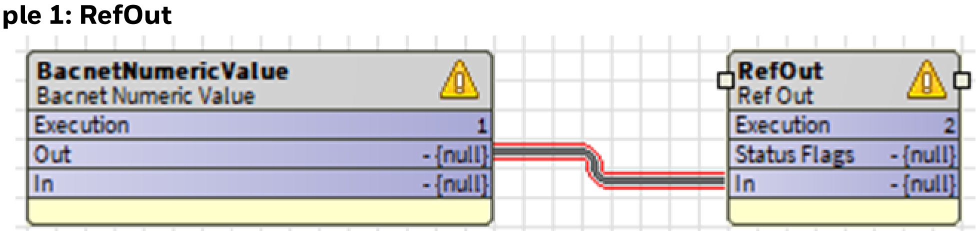


Fig. 59 RefOut Function Bloc

CHAPTER

4 ARITHMETIC FUNCTION BLOCKS

The following Arithmetic function blocks are available in the honIrmControl Palette and can be configured and used to create the required application logic:

- Add
- Aggregation
- Divide
- Psychrometric
- Exponential
- Limit
- Linear Graph
- Math Operation
- Multiply
- Negative
- Reset
- Subtract
- Digital Filter
- Flow Velocity
- Encode

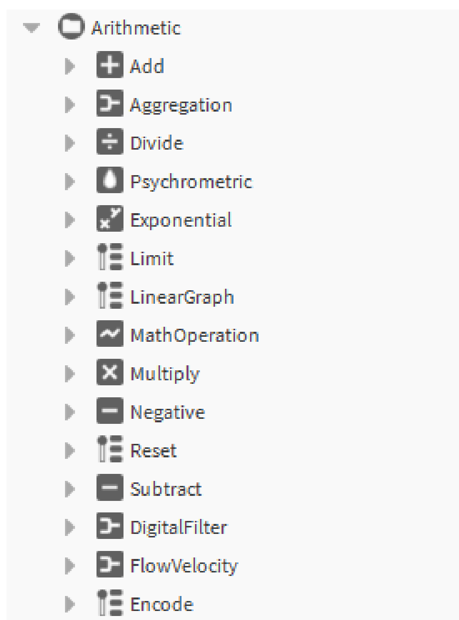


Fig. 60 Arithmetic Function Blocks

Add

The Add function block adds input A through H. In H can be either standard input or a parameter In H Par. In addition, there is a parameter called Ignore Inv In. The Master Sync Enabled function can be activated for parameters. The “No Of Inputs” is the amount of inputs that are considered for the output calculation.

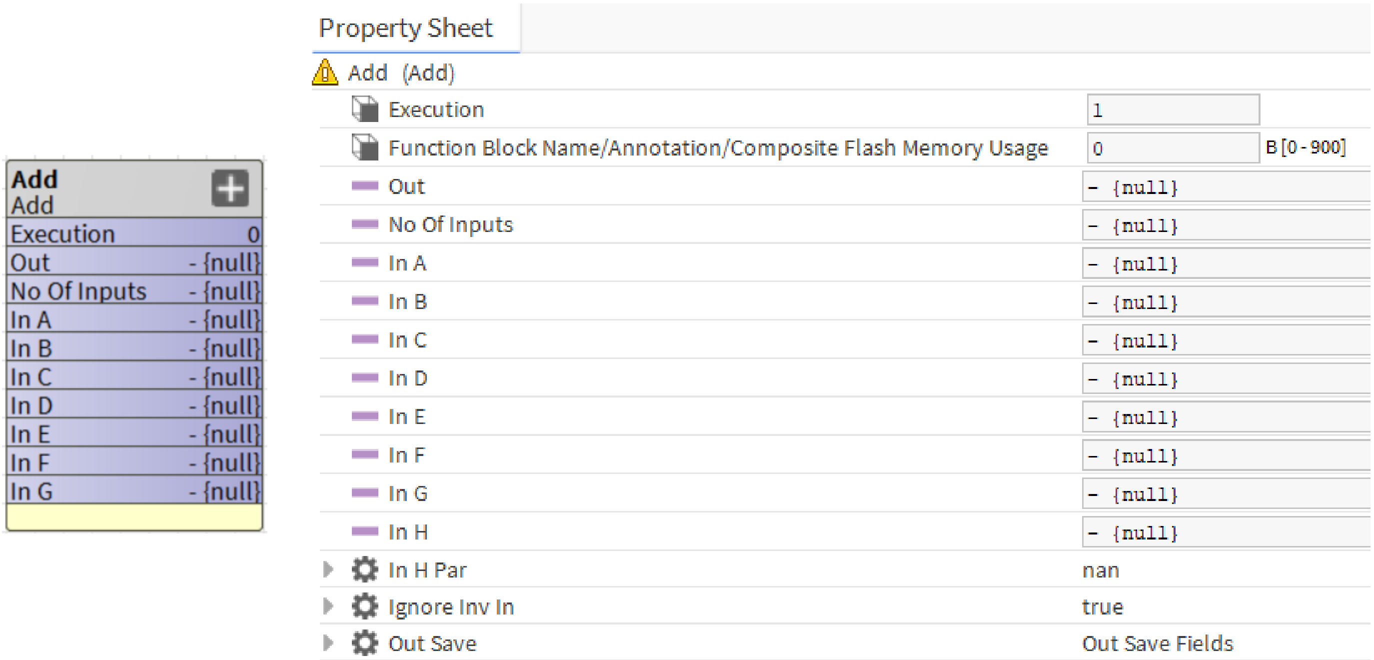


Fig. 61 Add Function Block and Property Sheet

Inputs must be valid to get considered for the calculation of the output. Valid values can be positive or negative. Inputs with "null" or "0.00" are ignored and not considered in the calculation. Inputs that are not connected have the value "null" and are ignored during the calculation (depending on Ignore Inv In).

Note: *Parameter values are only visible in the Property Sheet but not in the Function Block view.*

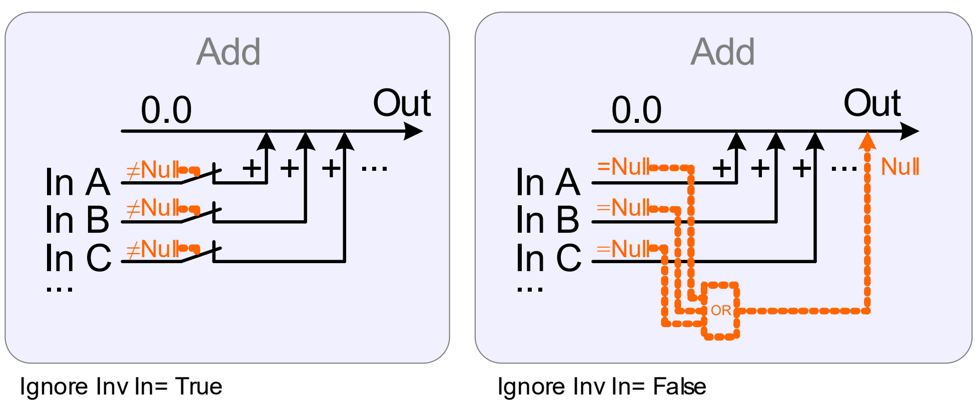


Fig. 62 Add – Logic Diagram

Input

Table 54 Inputs of Add

Input Name	Description
In A – In H	This input slot is of Numeric type.
In H (In & Par)	This input slot is of type Numeric. Since **In H** is also available as a parameter, the slot is hidden by default. This allows you to add a constant without using another function block.

Output

Table 55 Outputs of Add

Output Name	Description
Out	Outputs the sum of inputs A through H depending on the parameters. If any of the input have the value "null", then the following applies: **0.00:** If Ignore Inv In is set to “true.” **null:** If Ignore Inv In is set to “false.”
No of Inputs	The "**No of Inputs**" refers to the number of inputs taken into account when calculating **Out**. If **Out** is "null, the “**No of Inputs**” becomes "null" because no calculation is performed.

Parameter

Table 56 Parameters of Add

Parameter Name	Description
In H Par (In & Par)	If **In H** is "null," then **In H Par** is used for the calculation. Default is "nan."
Ignore Inv In	• True means that only inputs with a value not equal to "null" are considered. • False means that as soon as an input has a value of "null," the output **Out** is set to "null." • Default is "true."
Out Save	• **Master Sync Enabled:** If you set it to “TRUE,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Examples

Example 1: Add (Ignore Inv In set to false)

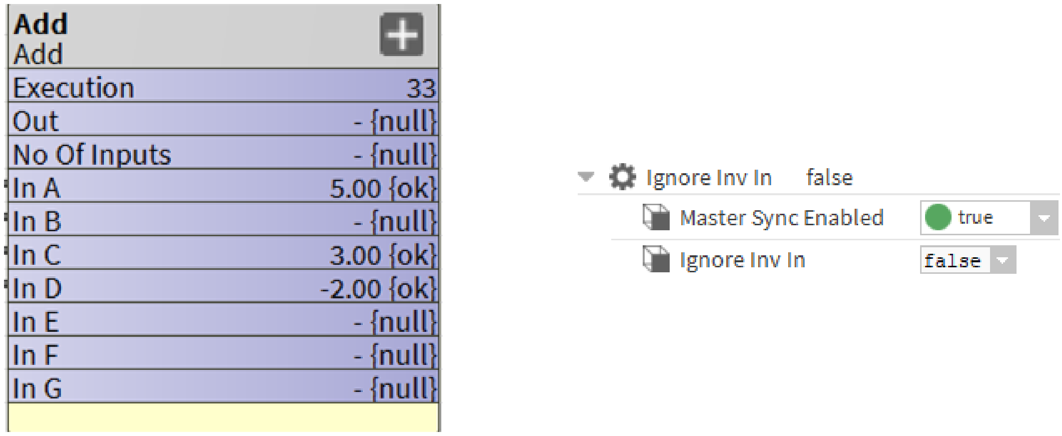


Fig. 63 Add with Ignore Inv In set to false

Out is "null" because one of the inputs is "null." **No of Inputs** is "null" because **Out** is "null."

Example 2: Add (Ignore Inv In set to true)

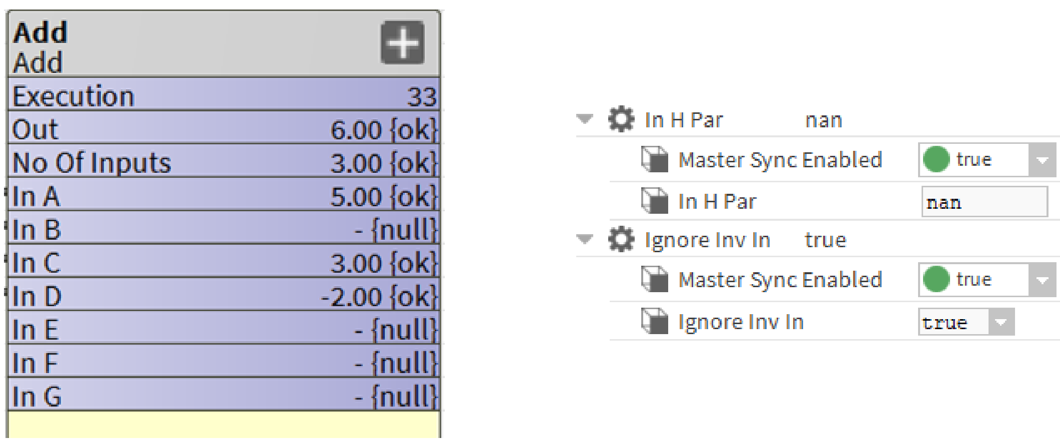


Fig. 64 Add with Ignore Inv In set to true

Number of inputs = 3, because only valid inputs are summarized. **Out** is 6, because **In B** is ignored because of the "null" value. Thus, the addition looks as following: **Out** = **In A** + In C + In D = 5 + 3 - 2 = 6.

In D - **In H** are not connected, and therefore they are ignored. **In H Par** has the value "nan," and therefore, it is also ignored.

Aggregation

The aggregation function block brings together multiple inputs into a single output. The methods, like average calculation min or max selection, can be set under the parameter operation. It is possible to change the operation without changing the function block or the connections.

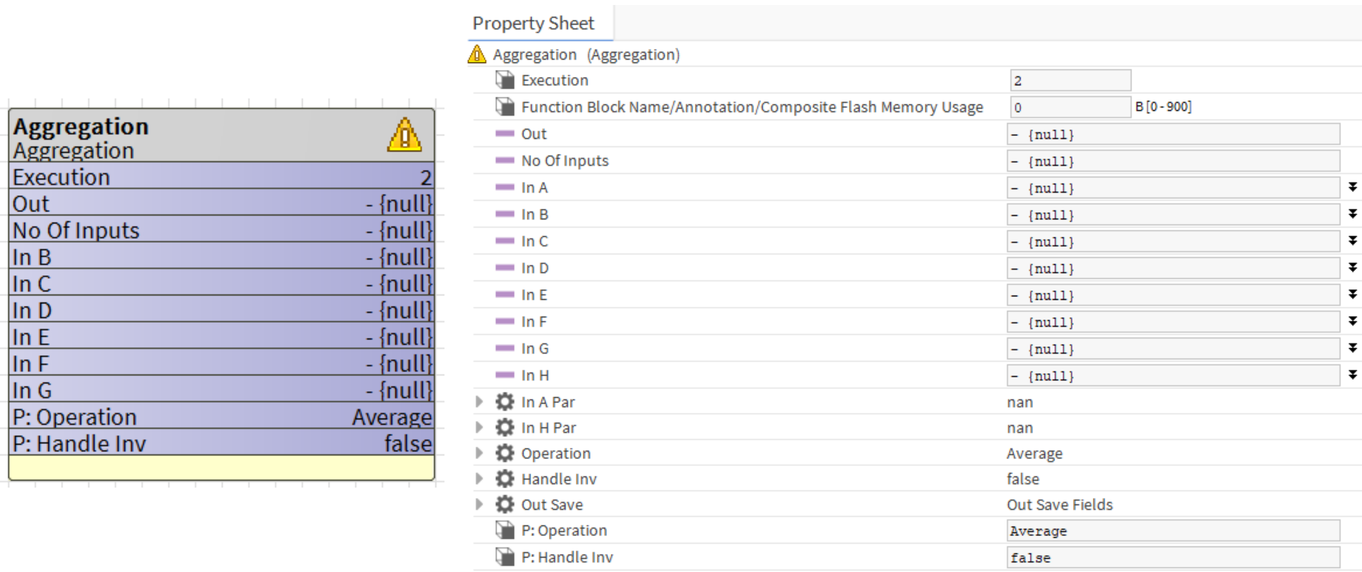


Fig. 65 Aggregation – Wire Sheet and Property Sheet

In H has the highest priority if 2 inputs are the same.

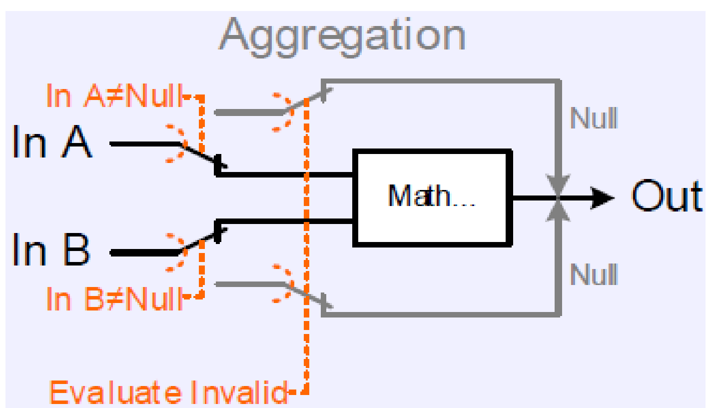


Fig. 66 Aggregation – Logic Diagram

Input

Table 57 Inputs of Aggregation

Input Name	Description
In A (In & Par)	The value of the **In A** is used to perform the desired aggregation. This input is also available as a parameter, which can be helpful for a min or max arbitration.
In B – In G	The inputs B - G values are used to perform the desired aggregation.
In H (In & Par)	The value of the input H is used to perform the desired aggregation. This input is also available as a parameter, which can be helpful for a min or max arbitration.

Output

Table 58 Outputs of Aggregation

Output Name	Description
Out	The function block merges multiple inputs into a single output. The parameter Operation can be used to set this method, which includes average calculation and min or max selection. The operation can be changed without changing the function block connections.
No of Inputs	The **No of Inputs** refers to the number of inputs taken into account when calculating **Out**. • If **Out** is "null, the **No of Inputs** becomes "null" because no calculation is performed.

Parameter

Table 59 Parameters of Aggregation

Parameter Name	Description
In A Par (In & Par)	• If **In A** is “null,” **In H Par** is used as the parameter. This is used for a minimum or maximum arbitration. • If **In A** Par is set to “null,” it is ignored because the parameter Handle Inv is set to Ignore Inputs with null.
In H Par (In & Par)	• If **In H** is “null,” **In H Par** is used as the parameter. This is used for a minimum or maximum arbitration. • If **In H Par** is set to “null,” it is ignored because the parameter Handle Inv is set to Ignore Inputs with null.

Table 59 Parameters of Aggregation (Continued)

Parameter Name	Description
Operation	The function block combines multiple inputs into a single output. The type of combination is set via this parameter. Example for Average: 3 different temperature sensors are connected to the inputs A-C, and the average value is output at the **Out**. The output **No of Inputs** = 3. There are three operations: Average, Maximum, and Minimum. • **Average:** Calculates the average of the inputs. • **Maximum:** Maximum output of the applied inputs. • **Minimum:** Minimum output of the applied inputs.
Handle Inv	If this option is set to false, all inputs with a “null” value are ignored. This is the default value. **True:** If any Input is “null,” then **Out** = “null.” Therefore, set all inputs to a meaningful value (for example, 0 or 100).
Out Save	• **Master Sync Enabled:** If you set it to “TRUE,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Examples

Example 1: Aggregation (Handle Inv set to false)

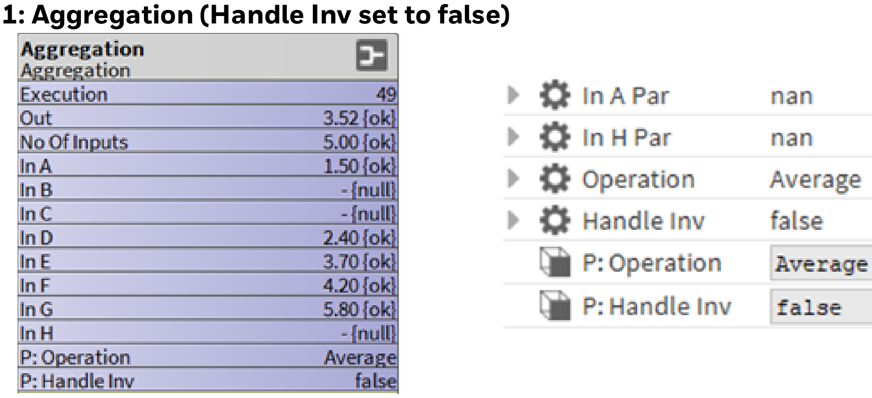


Fig. 67 Aggregation with Handle Inv set to false

Example 2: Aggregation (Handle Inv set to true)

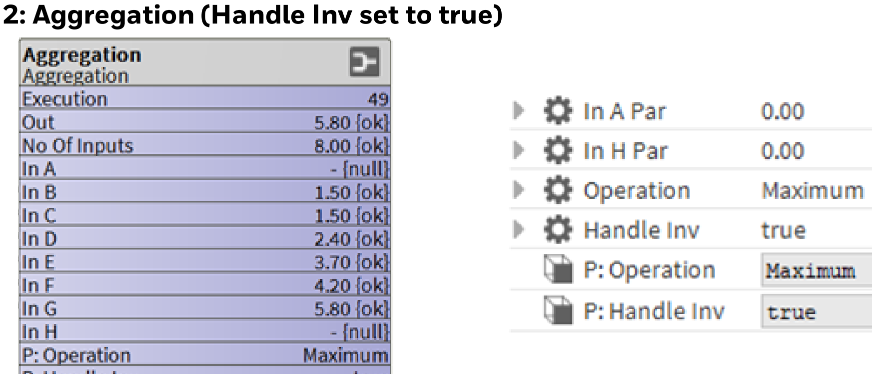


Fig. 68 Aggregation with Handle Inv set to true

Divide

The divide function takes two inputs and divides them by each other. Depending on the operation, the output is either Dividend / Divisor or Dividend modulo Divisor. When you divide by zero, you get an error.

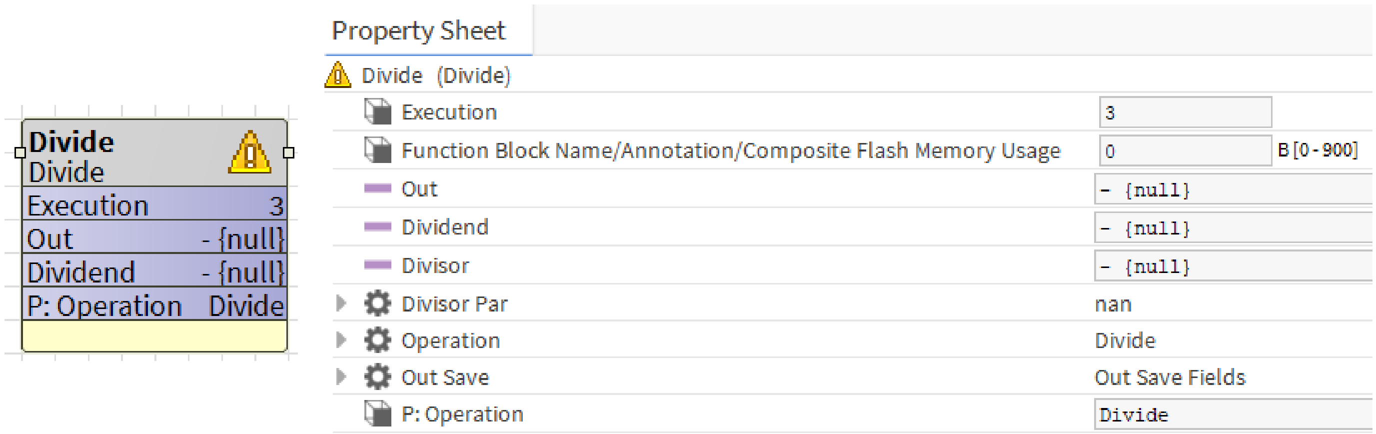


Fig. 69 Divide – Wire Sheet and Property Sheet

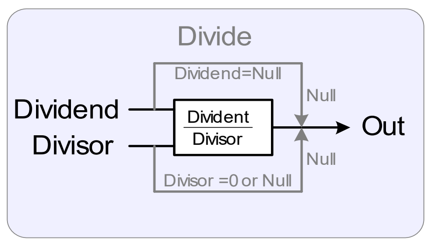


Fig. 70 Divide Logic Diagram

Input

Table 60 Inputs of Divide

Input Name	Description
Dividend	• If the operation is set to Divide, **Out** = **Dividend** or **Divisor**. • If the operation is set to Modulo, **Out** = **Dividend modulo** or **Divisor**.
Divisor (In & Par)	• If the operation is set to Divide, **Out** = **Dividend** or **Divisor**. • If the operation is set to Modulo, **Out** = **Dividend modulo** or **Divisor**.

Output

Table 61 Output of Divide

Output Name	Description
Out	• If the operation is set to Divide, **Out** = **Dividend** or **Divisor**. • If the operation is set to Modulo, **Out** = **Dividend modulo Divisor**. • If the **Dividend** = “null” or the **Divisor** = “null” or “0”, the output is “null.”

Parameter

Table 62 Parameters of Divide

Parameter Name	Description
Divisor Par (In & Par)	If **Divisor** is "null," **Divisor Par** is used as a parameter. **Divisor Par** must be unequal to 0.
Operation	There are two operations: • **Modulo: Out** = **Dividend modulo** or **Divisor** • **Divide: Out** = **Dividend** or **Divisor**
Out Save	• **Master Sync Enabled:** If you set it to "TRUE," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** Save feature.

Examples

Example 1: Divide (Operation set to Divide)

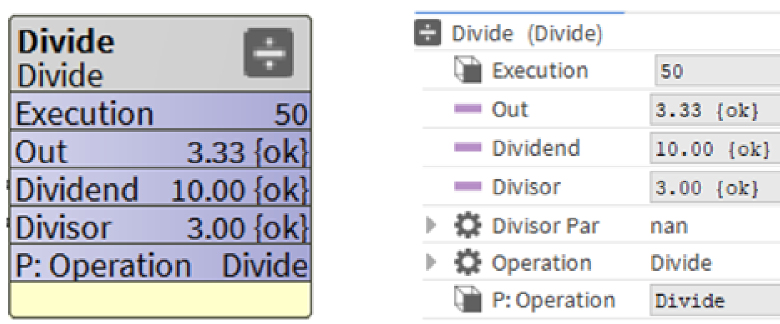


Fig. 71 Divide Operation set to Divide

Example 2: Divide (Operation set to Modulo)

10 modulo 3 = 1 because (3 x 3) + 1

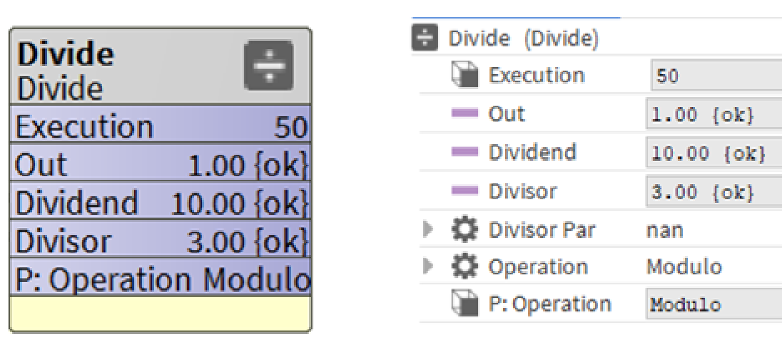


Fig. 72 Divide Operation set to Modulo

Psychrometric

Psychrometric function block calculates different temperature and relative humidity values like enthalpy, dewpoint, or absolute humidity.

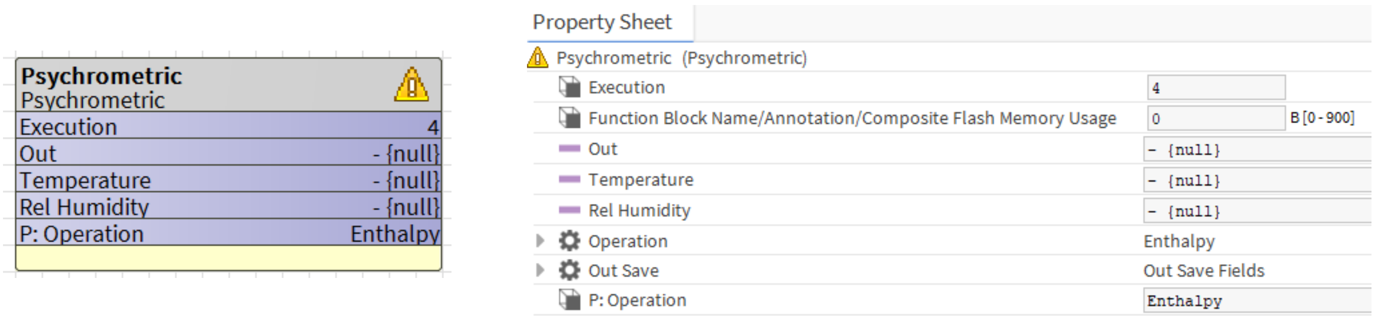


Fig. 73 Psychrometric Function Block and Property Sheet

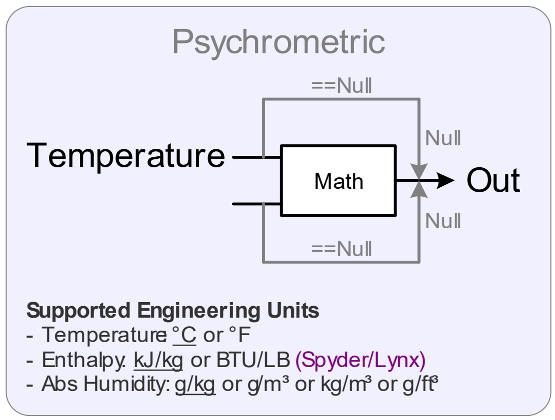


Fig. 74 Psychrometric – Logic Diagram

Input

Table 63 Inputs of Psychrometric

Input Name	Description
Temperature	This is a 32-bit floating point input slot. It depends on the global setting for the controller.
Rel Humidity	This is a 32-bit floating point input slot. It is measured in percentage (%).

Output

Table 64 Outputs of Psychrometric

Output Name	Description
Out	**Output:** Enthalpy (kJ/kg or BTU/ LB), Dewpoint, or Absolute humidity (g/kg, kg/m3, g/m3) (Depends on the selected operation).

Parameters

Table 65 Parameters of Psychrometric

Parameter Name	Description
Operation	There are three operations: Enthalpy, Dewpoint, and AbsHumidity. • **Enthalpy:** This function computes the enthalpy (kJ/kg or BTU/LB) based on the temperature (F° or °C) and relative humidity (%) inputs. Enthalpy is a measure of the energy present in the air. The warmer and more humid the air, the higher the enthalpy. Enthalpy is important for heat recovery. • **Dewpoint:** The dew point is the temperature at which air begins to form dew, for example, on a cold-water pipe. A cooling ceiling should not be supplied with colder water than the calculated dew point of the air in the room; otherwise, drops can form on the ceiling and drip down. • **AbsHumidity:** The absHumidity indicates the humidity in relation to the volume. The value is measured in the supply air and is used to control the humidification of the supply air.
Out Save	• **Master Sync Enabled:** If you set it to “TRUE,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** Save feature.

Examples

Example 1: Psychrometric (Operation set to Enthalpy)

Enthalpy = 42.3 kJ/kg

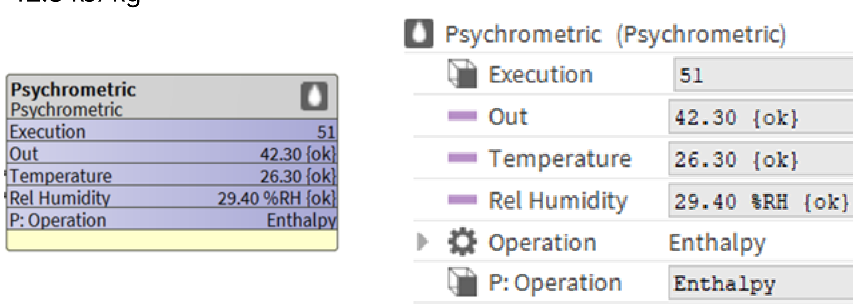


Fig. 75   Psychrometric with Enthalpy

Example 2: Psychrometric (Operation set to Dewpoint)

Dewpoint = 44.6 °F (7.03 °C)

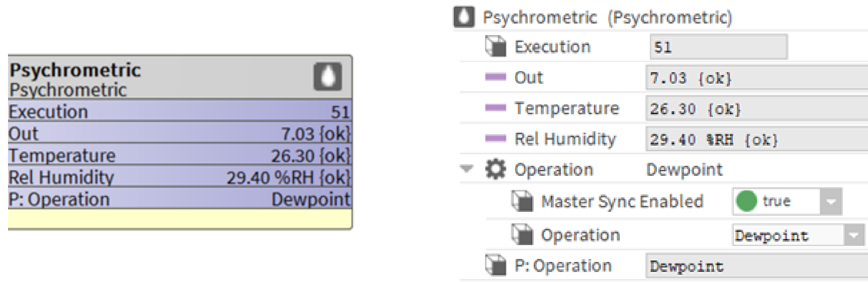


Fig. 76 Psychrometric with Dewpoint

Example 3: Psychrometric (Operation set to Absolute Humidity)

Absolute Humidity = 6.23 kg/m3

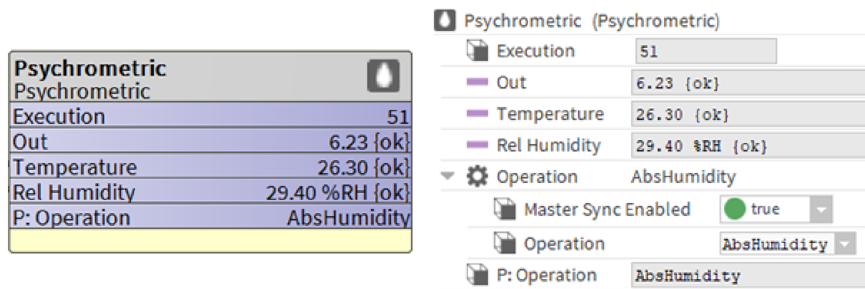


Fig. 77 Psychrometric with Absolute Humidity

Exponential

The Exponential function block carries out a potentiation calculation **Out = Base $^{\text{(Exponent)}}$**.

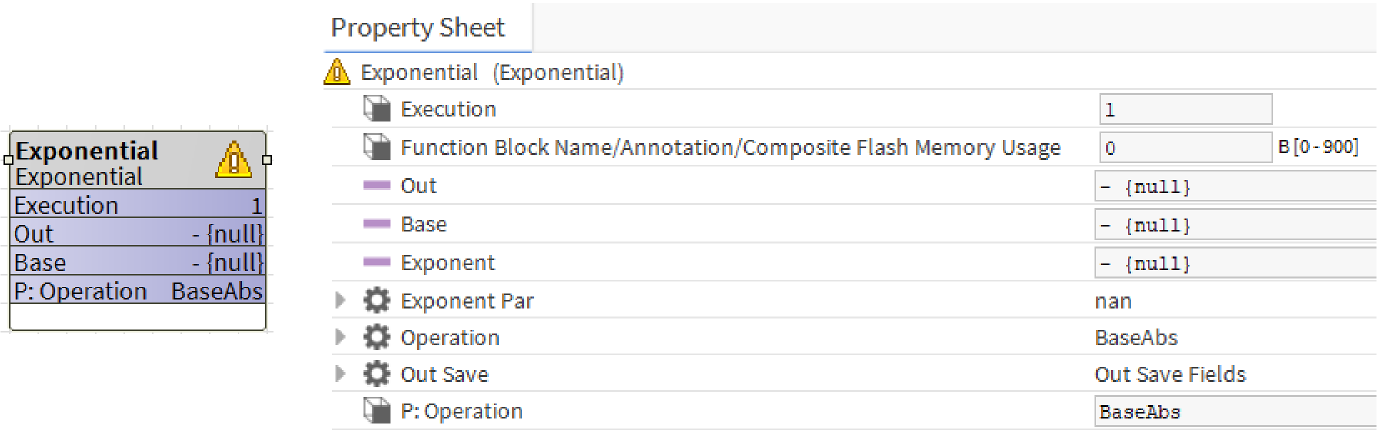


Fig. 78 Exponential – Wire Sheet and Property Sheet

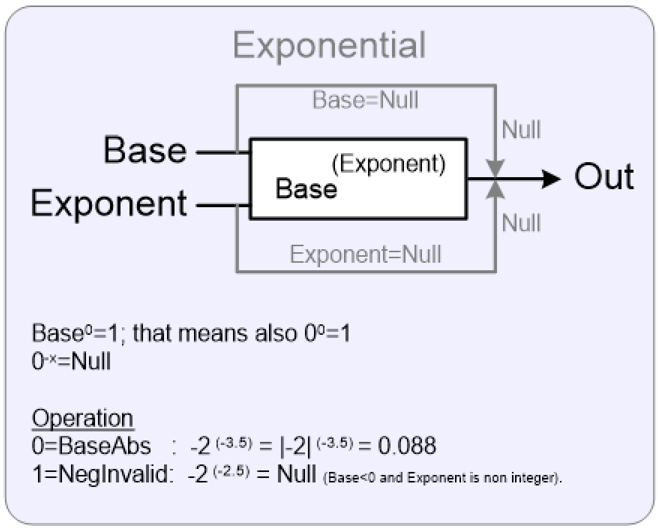


Fig. 79 Exponential - Logic Diagram

Input

Table 66 Inputs of Exponential

Input Name	Description
Base	The Base number states what is being multiplied.
Exponent (In & Par)	The Exponent states how many times the base number is being multiplied.

Output

Table 67 Outputs of Exponential

Output Name	Description
Out	An exponential calculation is a product of multiplying a number by itself. **Out: Base** $^{(\mathbf{Exponent})}$

Parameter

Table 68 Parameters of Exponential

Parameter Name	Description
Exponent Par (In & Par)	If the Exponent is “null,” Exponent Par is used as a parameter.
Operation	There are two operations: BaseAbs and NegInvalid. For correct calculation, use NegInvalid. To avoid “null” values in very rare cases, use BaseAbs (recommendation). • **BaseAbs: Out** is calculated if both base and exponent are negative and the exponent is not an integer value. • **NegInvalid: Out** becomes negative if both base and exponent are negative and the exponent is not an integer value.
Out Save	• **Master Sync Enabled:** If you set it to “TRUE,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Examples

Example 1: Exponential (Operation set to BaseAbs)

Result = 0.125 (NegInvalid would have the same result)

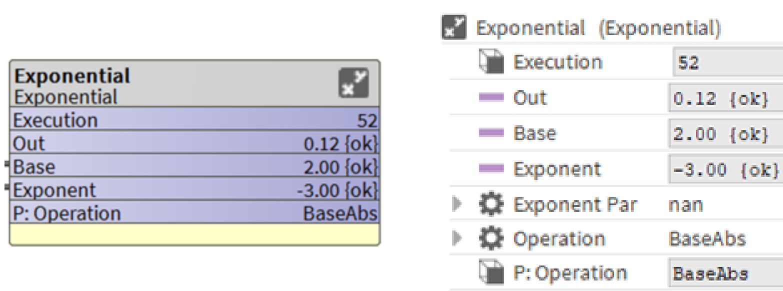


Fig. 80 Exponential – Example with BaseAbs

Value examples depending on operation.

BaseAbs: Out = Base (Exponent). If Base and Exponent are both negative, Out =|Base| Exponent.

Example: $2^{(3)} = 8$, $-2^{(3)} = -8$, $-2^{(-3)} = -0.125$, $-2.5^{(-3)} = -0.064$, $-2^{(-3.5)} = |2|^{(-3.5)} = 0.088$. $0^{(-3)}$ = “null”.

Limit

The Limit function block restricts the input to the low and high limits for the following In value:

- Lower than the **Limit Low**, **Out** is set to **Limit Low**.
- Higher than the **Limit High**, **Out** is set to **Limit High**.
- Between the **Limit Low** and **Limit High**, **Out** is set to **In**.

The two boolean outputs, Out Min and Out Max, can be used to determine whether the limitation has taken place.

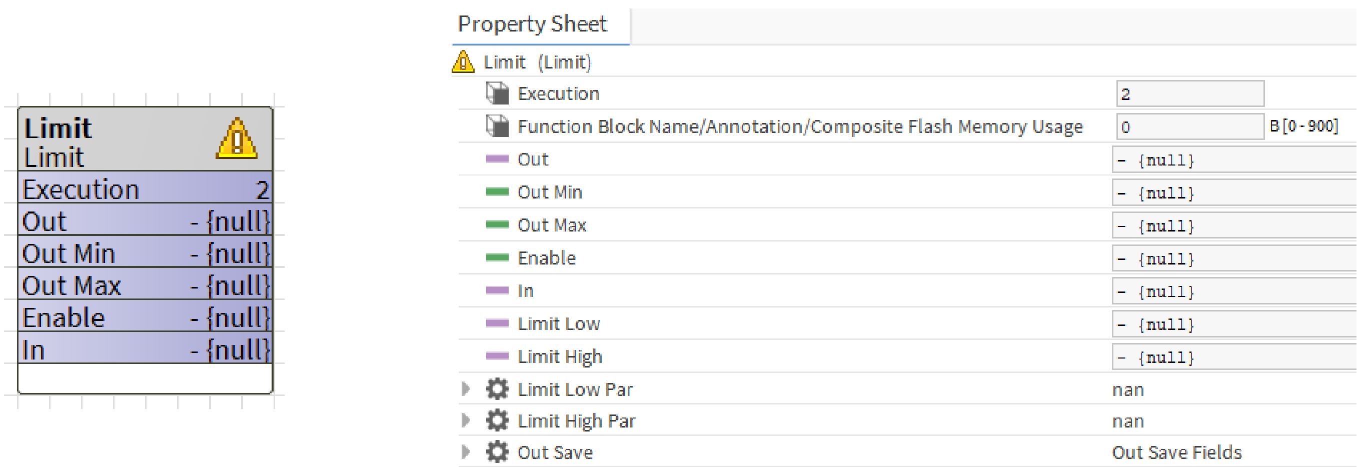


Fig. 81 Limit - Wire Sheet and Property Sheet

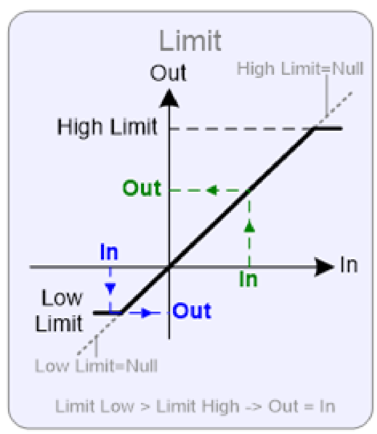


Fig. 82 Limit – Logic Diagram Curve

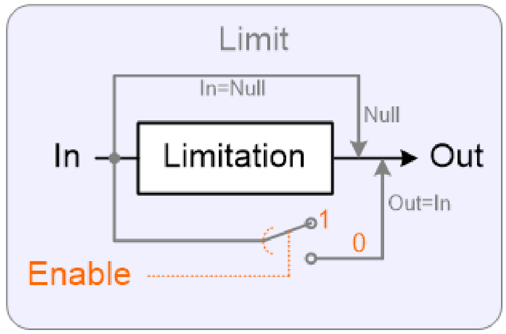


Fig. 83 Limit – Logic Diagram Flow

Input

Table 69 Inputs of Limit

Input Name	Description
Enable	The limitation occurs if Enable = “true.” **false:** No limitation **true or null:** Limitation
In	This is the input value that is limited at **Out** if enabled and **Limit Low** or **Limit High** are set.
Limit Low (In & Par)	This is the value for the lower limit.
Limit High (In & Par)	This is the value for the higher limit.

Output

Table 70 Outputs of Limit

Output Name	Description
Out	**Out = In,** but limited between low limit and high limit. If the low and high limits are configured and if enable is set accordingly. There is no limitation if **Limit Low** > **Limit High** (**Out = In**).
Out Min	If In value is limited to the lower limit, output is “true”.
Out Max	If In value is limited to the higher limit, the output is “true”.

Parameter

Table 71 Parameters of Limit

Parameter Name	Description
Limit Low Par (In & Par)	• If **Limit Low** is "null," **Limit Low Par** is used as a parameter. • If the parameter value is "null," **Limit Low** function is disabled.
Limit High Par (In & Par)	• If **Limit High** is "null," **Limit High Par** is used as a parameter. • If the parameter value is "null," **Limit High** function is disabled.
Out Save	• **Master Sync Enabled:** If you set it to "TRUE," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature. • **Out Min:** To enable or disable the **Out Min** feature. • **Out Max:** To enable or disable the **Out Max** feature.

Examples

Example 1: Limit (Limit set to 0 to 100)

Out = 100. Out Max = "true".

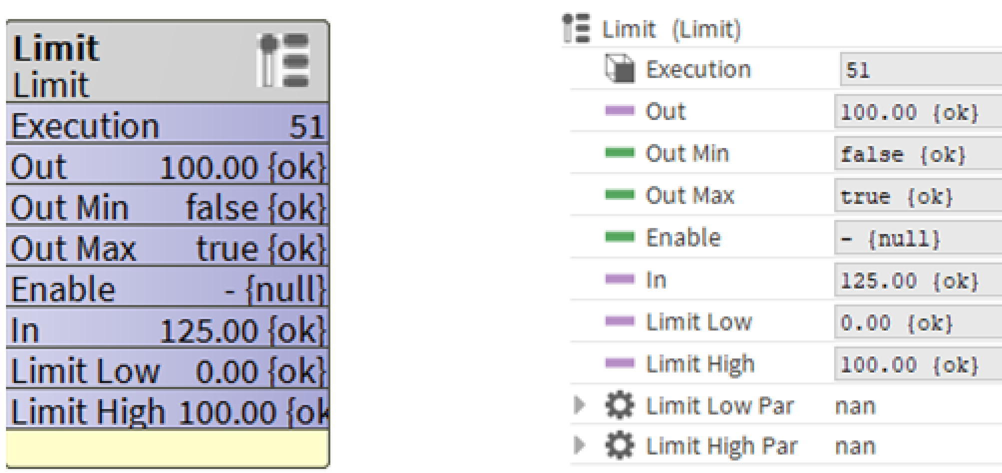


Fig. 84 Limit with High Limitation

Example 2: Limit (No high limitation)

Out = In. No limitation because Limit High is disabled (null). Low Limit is still working.

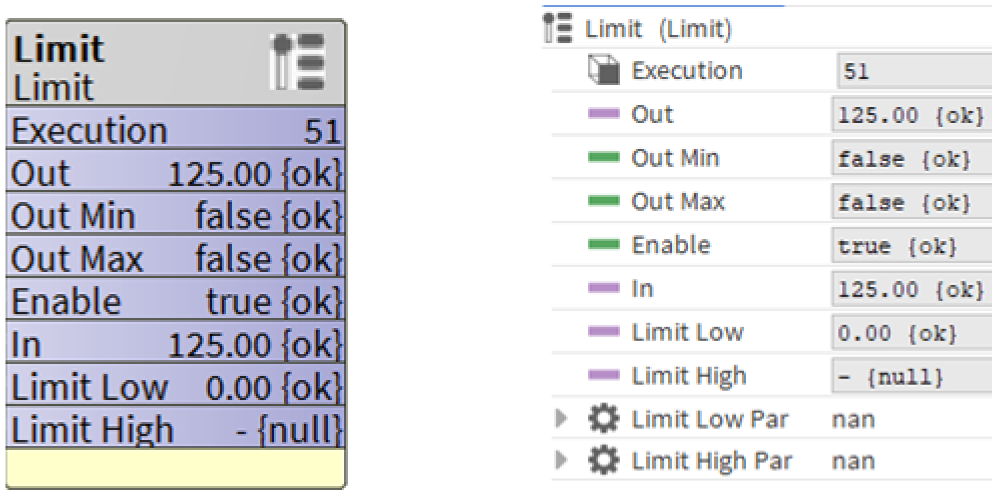


Fig. 85 Limit with disabled High Limitation

Linear Graph

The Linear Graph function block converts the input In to the output **Out** based on the linear graph curve defined by **X1, X2, Y1**, and **Y2** and based on the **Operation**.

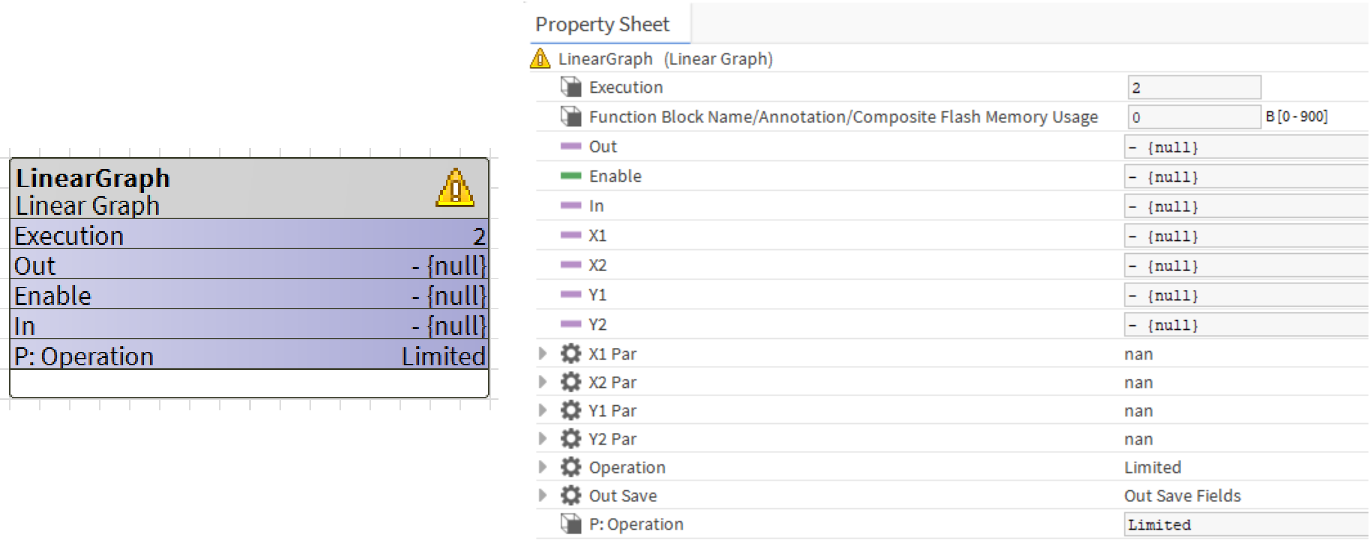


Fig. 86 Linear Graph - Wire Sheet and Property Sheet

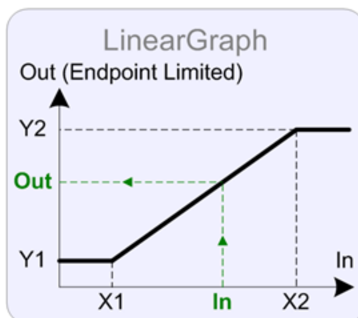


Fig. 87 Linear Graph – Logic Diagram – Endpoint Limited

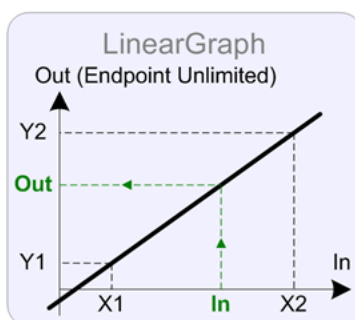


Fig. 88 Limit – Logic Diagram - Endpoint Unlimited

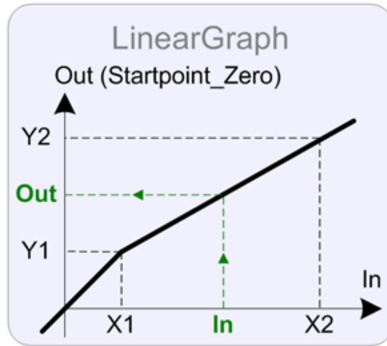


Fig. 89 Limit – Logic Diagram - Startpoint Zero

Input

Table 72 Inputs of Linear Graph

Input Name	Description
Enable	If it is set to “true”, it enables the function block.
In	This is numeric input value.
X1 (In & Par)	X Value from point 1.
X2 (In & Par)	X Value from point 2.
Y1 (In & Par)	Y Value from point 1.
Y2 (In & Par)	Y Value from point 2.

Output

Table 73 Outputs of Linear Graph

Output Name	Description
Out	Linear graph curve calculated as following: **Out (In) = Y1 + (((In – X1) * (Y2 – Y1)) / (X2 – X1))**

Parameters

Table 74 Parameters of Linear Graph

Parameter Name	Description
X1 Par (In & Par)	If **X1** is “null,” **X1 Par** is used as a parameter. X Value from point 1.
X2 Par (In & Par)	If **X2** is “null,” **X2 Par** is used as a parameter. X Value from point 2.
Y1 Par (In & Par)	If **Y1** is “null,” **Y1 Par** is used as a parameter. Y Value from point 1.

Table 74 Parameters of Linear Graph (Continued)

Parameter Name	Description
Y2 Par (In & Par)	If **Y2** is “null,” **Y2 Par** is used as a parameter. Y Value from point 2.
Operation	There are three operations: • Endpoint Limited • Endpoint Unlimited • Startpoint_Zero (In IRM controller called as Vav_Flow_Balance)
Out Save	• **Master Sync Enabled:** If you set it to “TRUE,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Examples

Example 1: Linear Graph (Cooling 2 calculation)

Converting from PID Cooling Output 0 - 100 % into 50 - 100 % for Cooling 02 to open the first Clg01. If Clg01 is on 100 %, open Clg02.

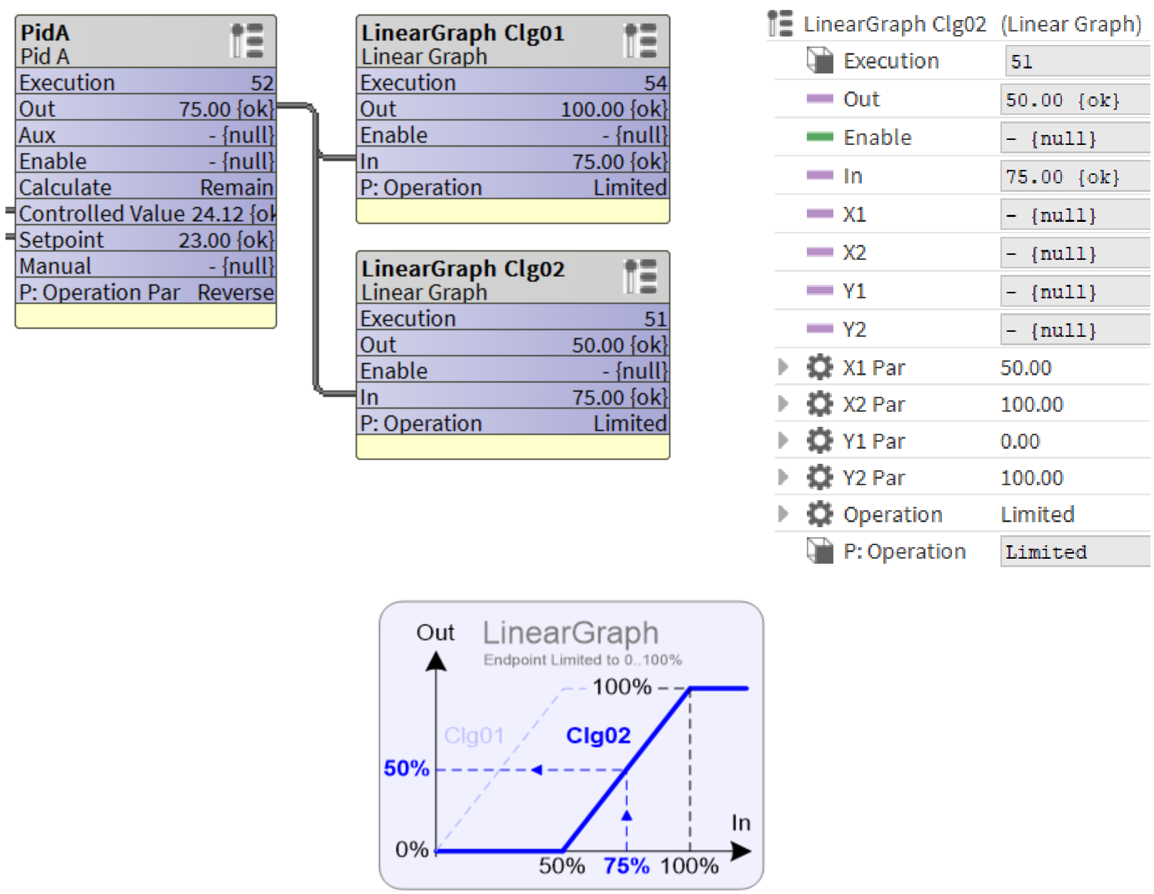
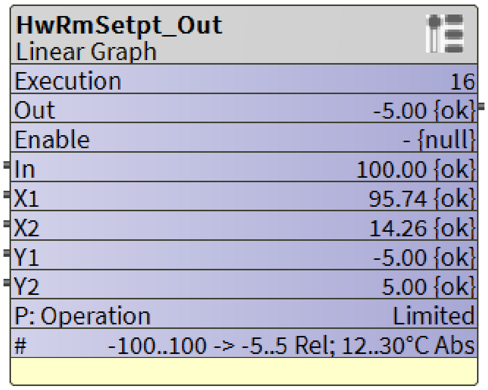
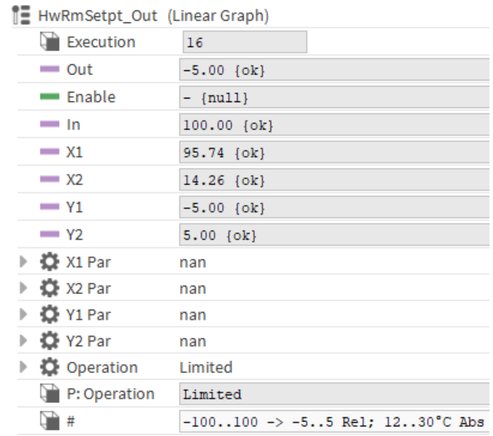


Fig. 90 Linear Graph – Cooling 2 calculation

Example 2: Linear Graph (Relative Setpoint from conventional WM)

Conventional wired wall module with relative setpoint.





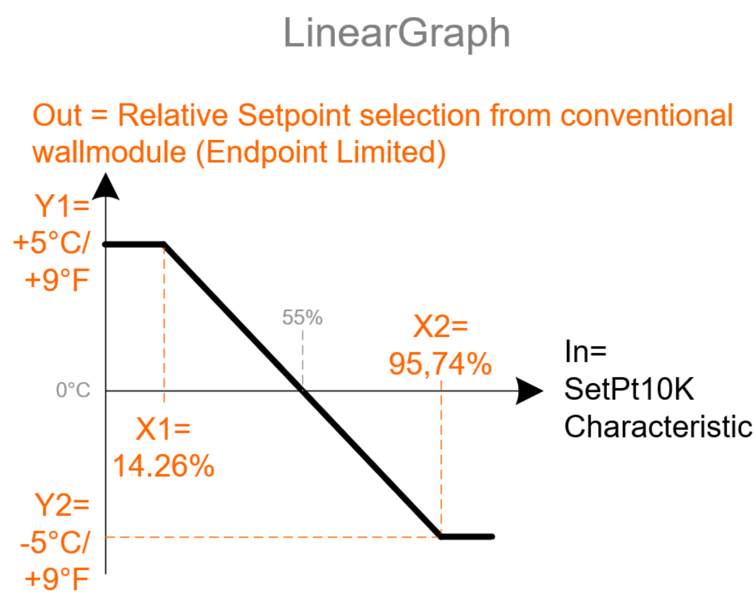
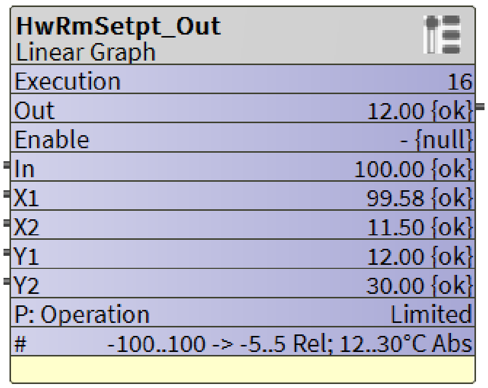
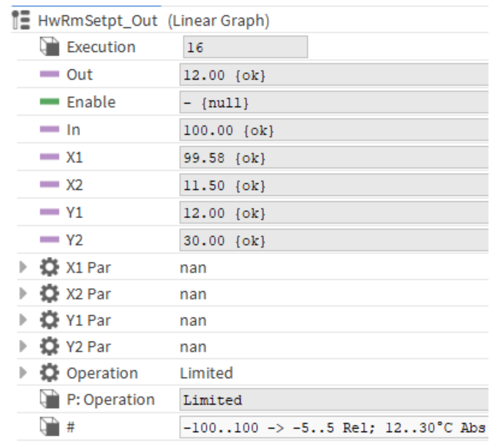


Fig. 91 Linear Graph – Relative Setpoint from conventional WM

Example 3: Linear Graph (Absolute Setpoint from conventional WM)

Conventional wired wall module with absolute setpoint °F (°C).





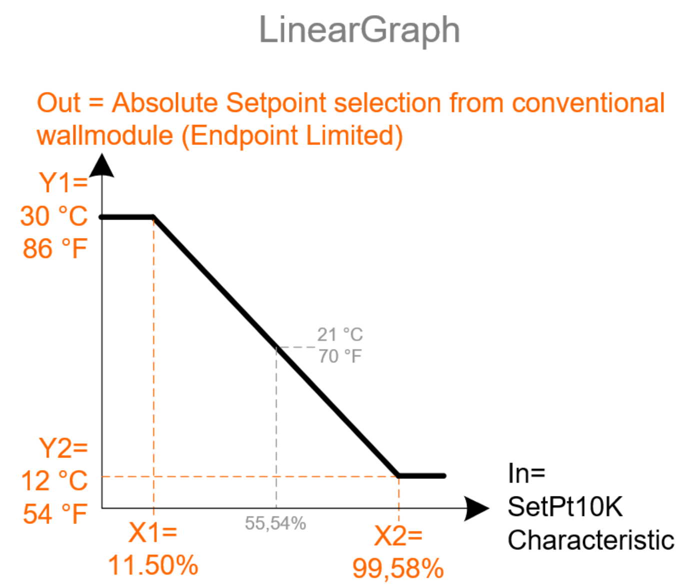
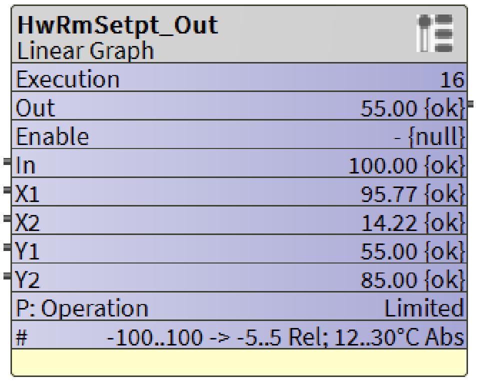
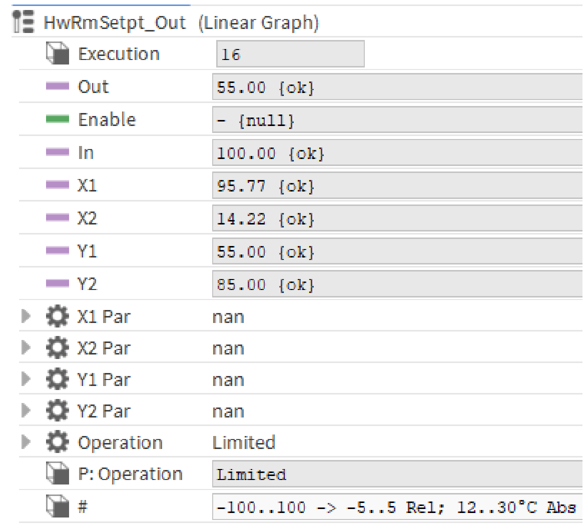


Fig. 92 Linear Graph – Absolute Setpoint from conventional WM in °F (°C)

Example 4: Linear Graph (Absolute Setpoint from conventional WM)

Conventional wired wall module with absolute setpoint °F (°C).





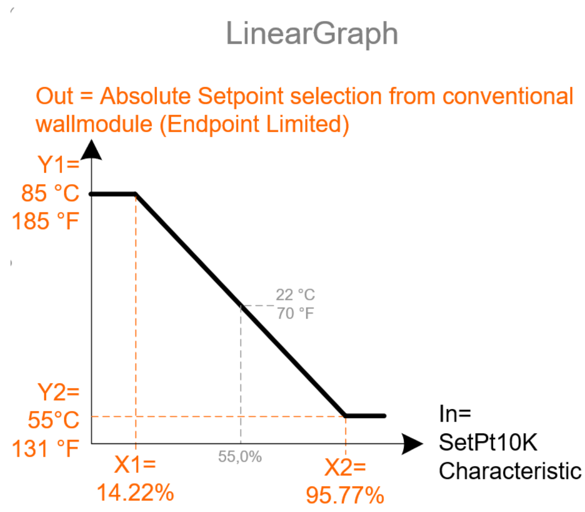


Fig. 93 Linear Graph – Absolute Setpoint from conventional WM in °F (°C)

Math Operation

The Math Operation function block is used to perform various mathematical calculations on a single number.

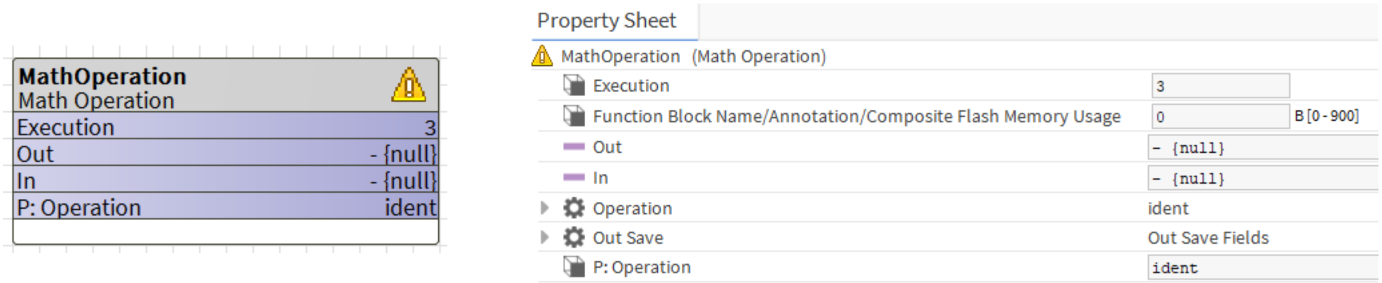


Fig. 94 Math Operation Function Block and Property Sheet

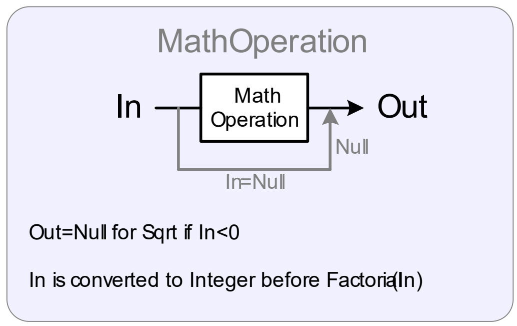


Fig. 95 MathOperation - Logic Diagram

Input

Table 75 Input of Math Operation

Input Name	Description
In	Depending on the operation, In is used to perform a mathematical calculation or it is used to change the representation.

Output

Table 76 Output of Math Operation

Output Name	Description
Out	Result of the calculation or the new representation of the input value.

Parameter

Table 77 Parameters of Math Operation

Parameter Name	Description
Operation	Following operations are possible: **ident:** Out = In **abs:** Absolute value, that is absolute value **int:** Integer value **frac:** Fractional value **arccos:** Inverse trigonometric function arccos() as radian value *1 **- acrsin:** Inverse trigonometric function arcsin() as radian value *1 **cos:** Trigonometric function cos() as radian value *1 **exp:** Exponential e(x) **log10:** Common logarithm with base 10 **ln:** Natural logarithm with base e **sin:** Trigonometric function sin() as radian value *1 **sqrt:** Square root **tan:** Trigonometric function tan() as radian value *1 **round:** Rounding after decimal place **sign:** Result is -1, 0, +1 depending on the sign of Inp *1: 1 rad = 57.29577951° or 1°=0.017453293 rad
Out Save	• **Master Sync Enabled:** If you set it to “TRUE,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the Out feature.

Example 1: Math (Operation set to Round)

- Ident(-24.456) = -24.456; Identical value
- Abs(-3.1) =3.1
- Int(3.25) = 3
- Frac(3.25) = 0.25; Frac(-4.5394) = -0.5394
- Arccos(-1) = Pi = 3.1415927; radian value
- ArcSin(1) = Pi/2 = 1.570796327; radian value
- ArcTan(1.0296) = 0.8
- Cos(3.1415927) = Cos (Pi) = -1; radian value
- Exp(3) = e3 = 2.7182818283 = 20.0855
- Log10(120) = 2.0792
- Ln(15) = loge (15) = 2.7081
- Sin(3.1415927) = Sin(Pi) = 0; Sin(1.570796327) = Sin(Pi/2) = 1; radian value

- Sqrt(16) = 4
- Tan(0.8) = 1.0296
- Round(6.5) = 7; Round(-1.5) =-1; Round(-1.51) = -2
- Sign(-3.4) = -1; Sign(3.4) = 1; Sign(0) = 0

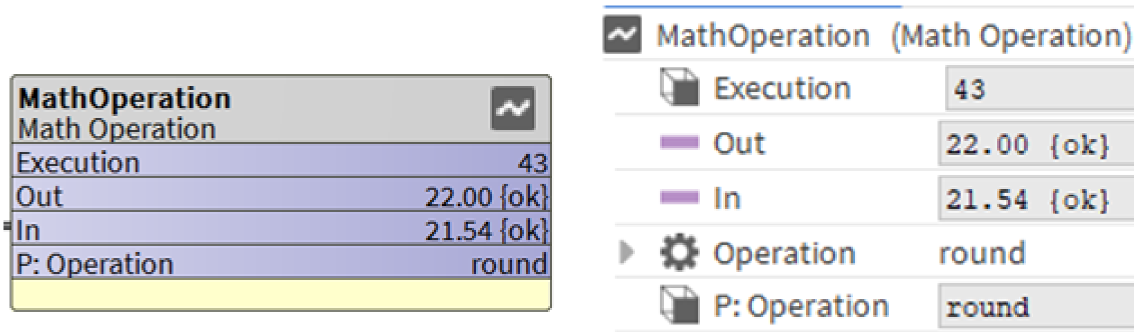


Fig. 96 Math Operation set to Round

Multiply

The Multiply function block multiplies one input with the other. **Out = In A** multiplied by **In B**. If the result overflows the range of a single precision floating point number, the result becomes invalid.

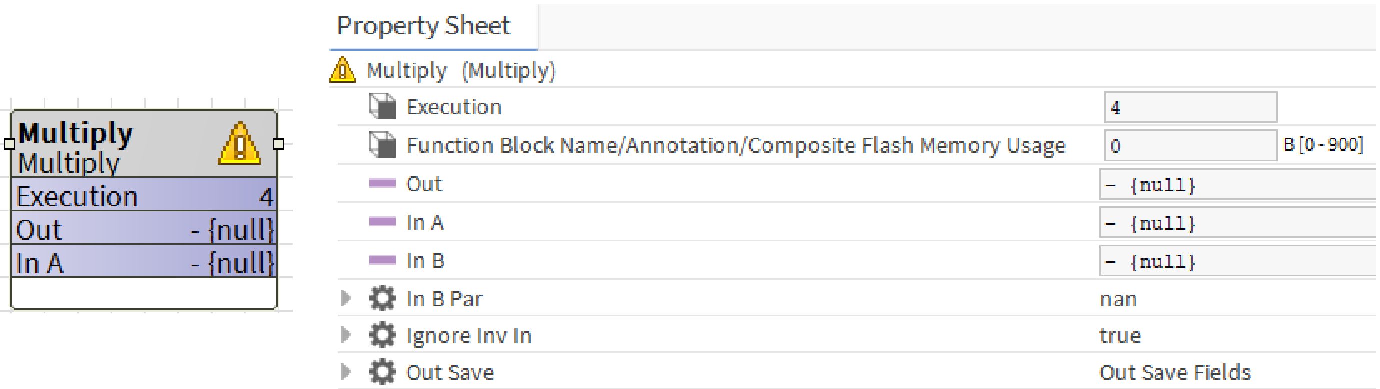


Fig. 97 Multiply Function Block and Property Sheet

The error occurs if there is an overflow.

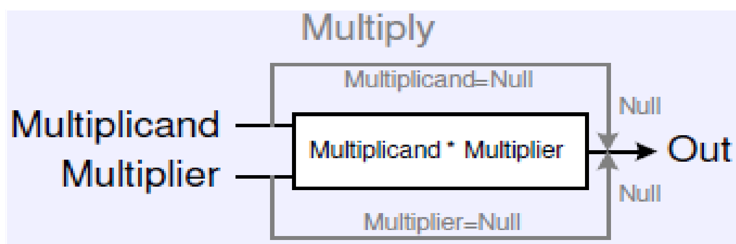


Fig. 98 Multiply Logic Diagram

Input

Table 78 Inputs of Multiply

Input Name	Description
In A	These are 32-bit floating point input slots.
In B (In & Par)	These are 32-bit floating point input slots.

Output

Table 79 Output of Multiply

Output Name	Description
Out	Output is the **In A** multiplied by **In B**.

Parameter

Table 80 Parameters of Multiply

Parameter Name	Description
In B Par (In & Par)	• If **In B** is “null,” **In B** Par is used as a parameter. • If **In B** Par is set to “nan,” it is treated as invalid.
Ignore Inv In	• If this option is set to “true,” the function block considers only valid inputs while determining the addition of the inputs. • If this option is set to “false” and any input becomes “null,” output becomes invalid. Default is "true.”
Out Save	• **Master Sync Enabled:** If you set it to “TRUE,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Example 1: Multiply

Multiply **In A** and **In B** to get the output.

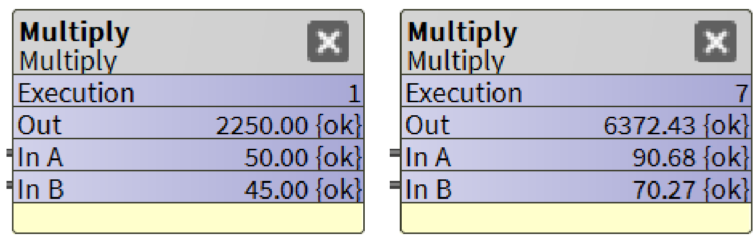


Fig. 99 Multiply Function Block

Negative

In the Negative function block input is inverted, and the output is negated after performing the logic.

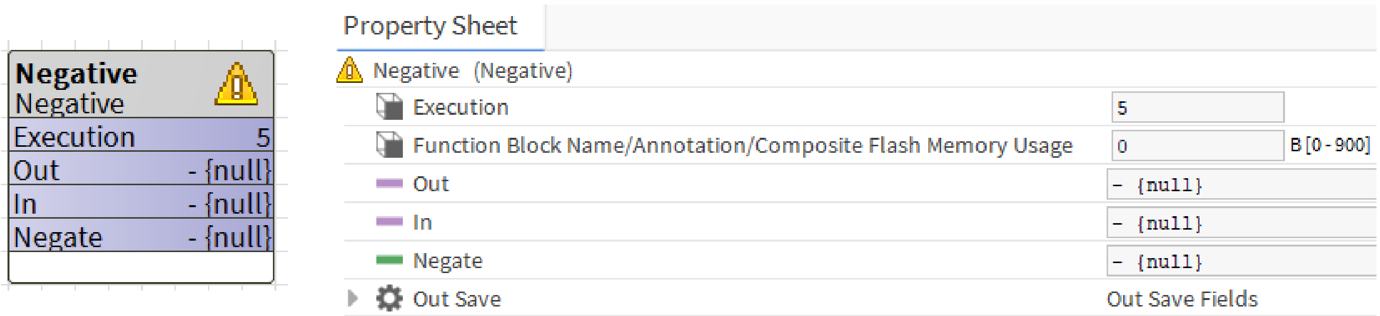


Fig. 100 Negative Function Block and Property Sheet

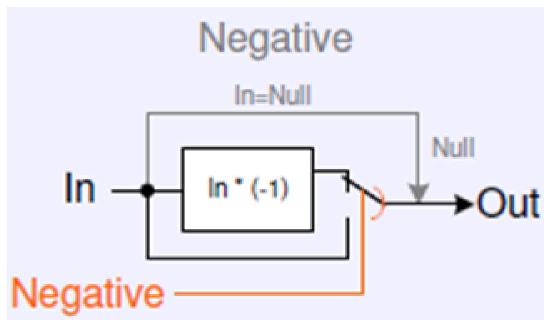


Fig. 101 Negative Logic Diagram

Inputs

Table 81 Inputs of Negative

Input Name	Description
In	This is a 32-bit floating point input slot.
Negate	If this option is selected, it negates the input.

Outputs

Table 82 Output of Negative

Output Name	Description
Out	**Output:** Negation of In.

Parameter

Table 83 Parameter of Negative

Parameter Name	Description
Out Save	• **Master Sync Enabled:** If you set it to “TRUE,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Example 1: Negative

Provides the inverted output for that input.

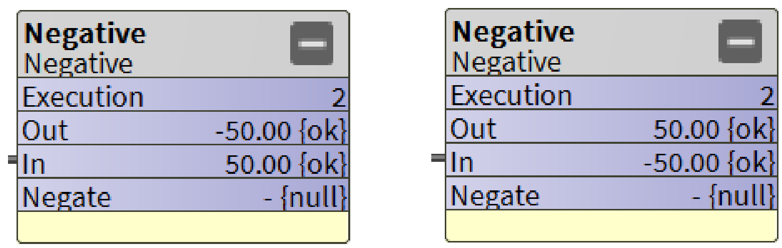


Fig. 102 Negative Function Block

Reset

The Reset function block increases or decreases the value In1 depending on another value In2.

- If In2 is within a specific range corresponding value of In1 is changed. A typical example is increasing a room temperature setpoint (In1) as a function of the outdoor temperature (In2).
- If the outdoor temperature is between 75.2 to 89.6 °F (24 to 32 °C), for example, the room setpoint is raised by a specific value, by a maximum of 39.2 °F (4 °C).
- If In2 is within a specific range corresponding value of In1 is changed. A typical example is increasing a room temperature setpoint (In1) as a function of the outdoor temperature (In2).
- If the outdoor temperature is between 75.2 to 89.6 °F (24 to 32 °C) , for example, the room setpoint is raised by a specific value, by a maximum of 39.2 °F (4 °C).

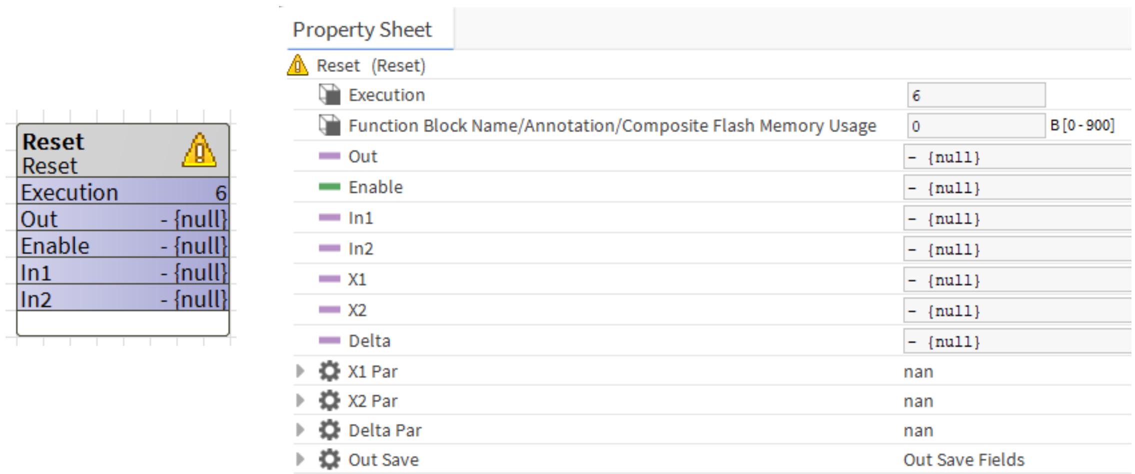


Fig. 103 Reset Function Block and Property Sheet

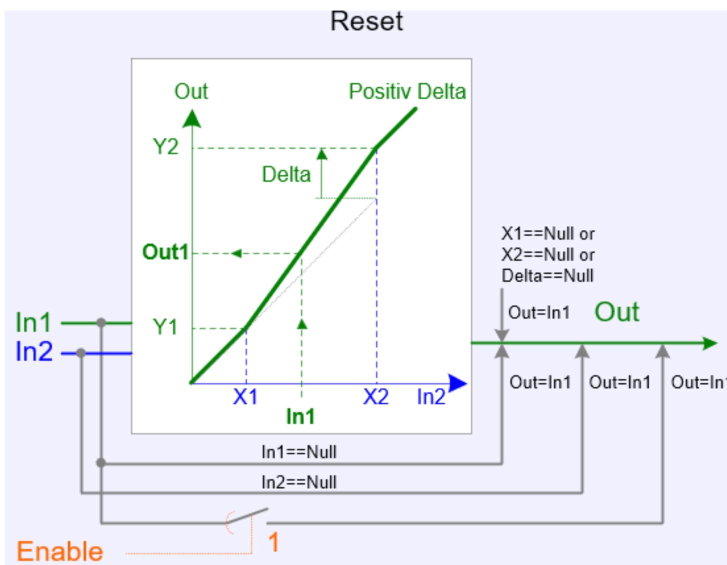


Fig. 104 Reset with Positive Delta

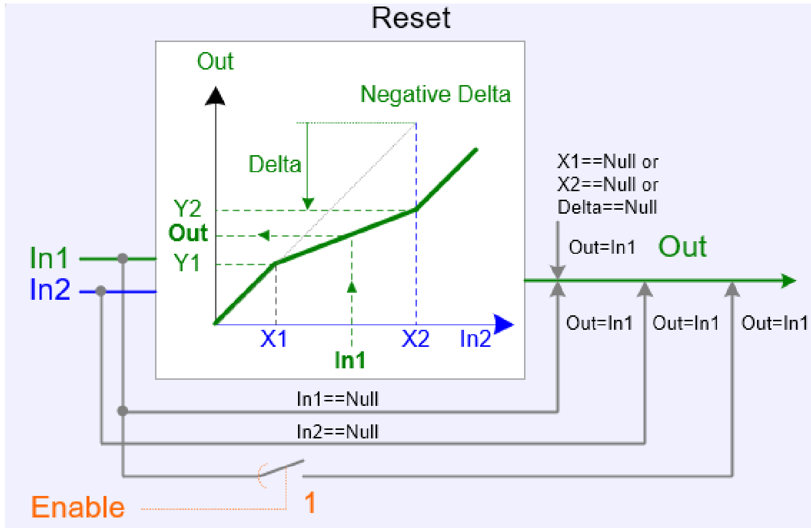


Fig. 105 Reset with Negative Delta

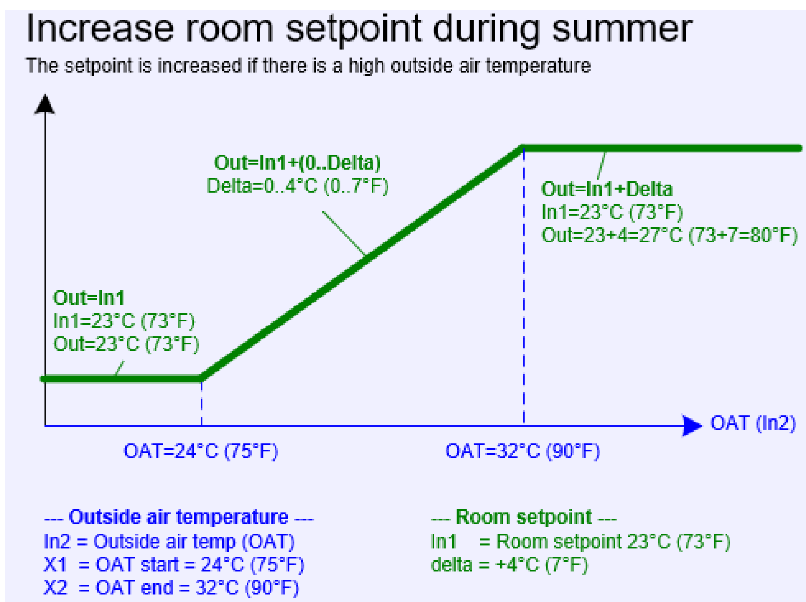


Fig. 106 Reset Function block to increase the cooling setpoint during summer

Note: *If the outside air temperature is more than 75.2 °F (24 °C), the room temperature setpoint of 73.4 °F (23 °C) is increased.*

If the outside air temperature >= 89.6 °F (32 °C), the maximum increase is reached (Fully Delta is added), and the room temperature setpoint is set to 80.6 °F (27 °C).

Input

Table 84 Inputs of Reset

Input Name	Description
Enable	If the **Enable** is set to “false,” **Out = In1**, that means no offset is added.
In1	**In1** serves as the basis for the output value **Out**, which is either passed through or raised or lowered by **Delta**, depending on the amount of the value of **In2.**
In2	In2 is a numeric value that decides whether and to what amount **In1** gets an offset (**Delta**).
X1 (In & Par)	**X1** refers to the value of **In2.** If **In2 <= X1**, **In1** is not corrected, that means **Out = In1** without additional offset.
X2 (In & Par)	**X2** refers to the value of **In2**. • If **In2 >=X2**, the full **Delta** is added to **In1**, that is **Out = In2 + Delta**. • If **In2** is between **X1** and **X2**, a portion of 0 - 100 % of **Delta** is added, that means **Out = In1 + 0 – 100 % * Delta.**
Delta (In & Par)	**Delta** refers to **In1**. It is added to **In1** if In2 is at an equivalent range. • If **In2** is between **X1** and **X2**, a portion of 0 - 100 % of **Delta** is added, that means **Out = In1 + 0 – 100 % * Delta.** • If **In2 > X2**, the entire **Delta** is added.

Output

Table 85 Output of Reset

Output Name	Description
Out	**Output:** Reset (input, sensor, 0%, 100 %, reset amount).

Parameter

Table 86 Parameters of Reset Function Block

Parameter Name	Description
X1 Par (In & Par)	• If **X1** is “null,” **X1 Par** is used instead of **X1**. • If **X1 Par** is set to “nan,” it is treated like “null.”
X2 Par (In & Par)	• If **X2** is “null,” **X2 Par** is used instead of **X2**. • If **X2 Par** is set to “nan,” it is treated like “null.”
Delta Par (In & Par)	• If **Delta** is “null,” **Delta Par** is used instead of **Delta**. • If **Delta Par** is set to “nan,” it is treated like “null.”

Table 86 Parameters of Reset Function Block (Continued)

Parameter Name	Description
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Examples

Reset with °F:

Example 1: Reset (SetptIncrease)

If the outdoor temperature In2 is < 75 °F (**X1 Par**), the cooling setpoint **In1** of 73 °F is not increased. The new cooling value **Out** corresponds to the cooling setpoint **In1**.

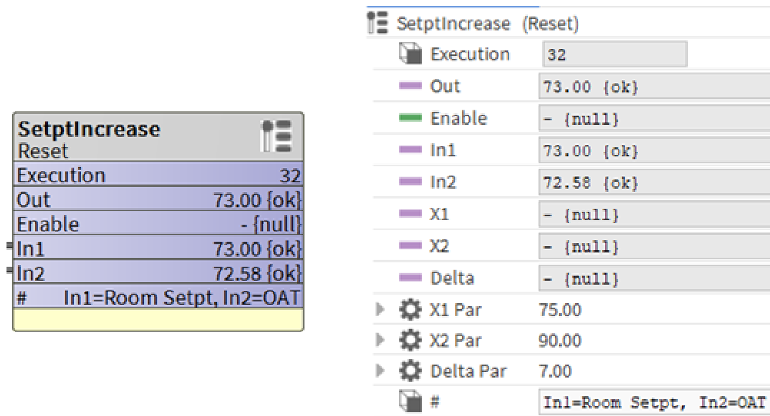


Fig. 107 Reset example 1

Example 2: Reset (SetptIncrease)

If the outside temperature In2 exceeds 75 °F (**X1 Par**), the room cooling setpoint with 73 °F (**In1**) is raised. The new cooling setpoint temperature **Out = In1 + a percentage of Delta = 73.52 °F.**

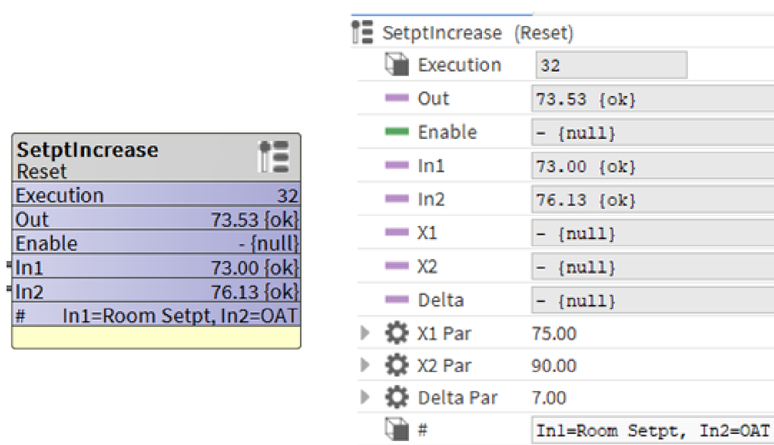


Fig. 108 Reset example 2

Example 3: Reset (SetptIncrease)

At the outdoor temperature of 82.58 °F (In2), the cooling setpoint **In1 = 73 °F** is increased by 3.54 °F, **Out = 76.54 °F.**

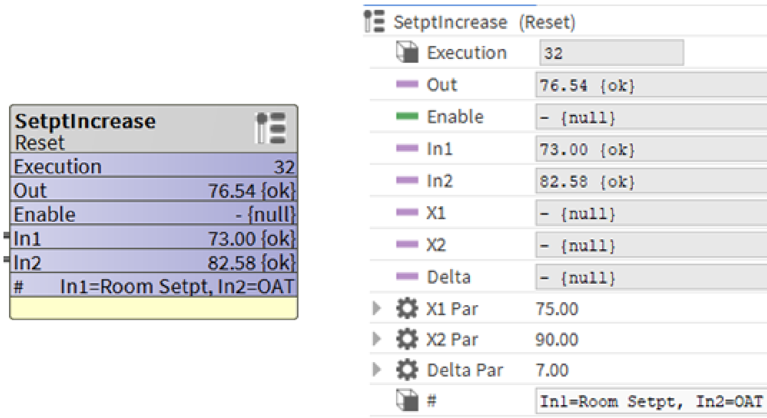


Fig. 109 Reset example 3

Example 4: Reset (SetptIncrease)

At the outdoor temperature of 91.54 °F (In2), the cooling setpoint **In1 = 73 °F** is increased by the fully delta of 7 °F, **Out = 80 °F.**

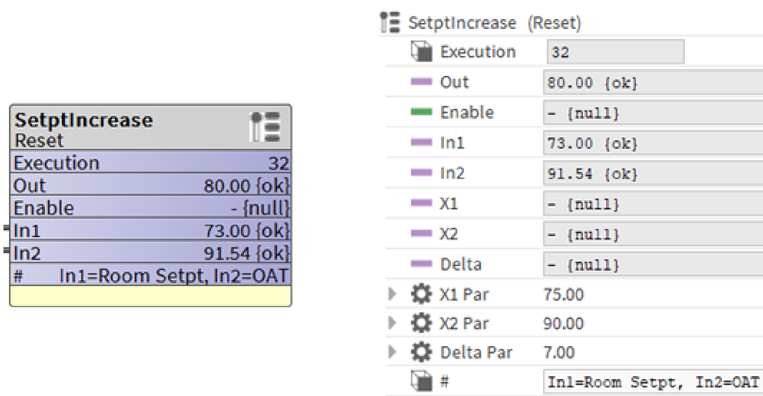


Fig. 110 Reset example 4

Reset with °C:

Example 1: Reset (SetptIncrease)

If the outdoor temperature In2 is < 24 °C (**X1 Par**), the cooling setpoint **In1** of 23 °C is not increased. The new cooling value **Out** corresponds to the cooling setpoint **In1**.

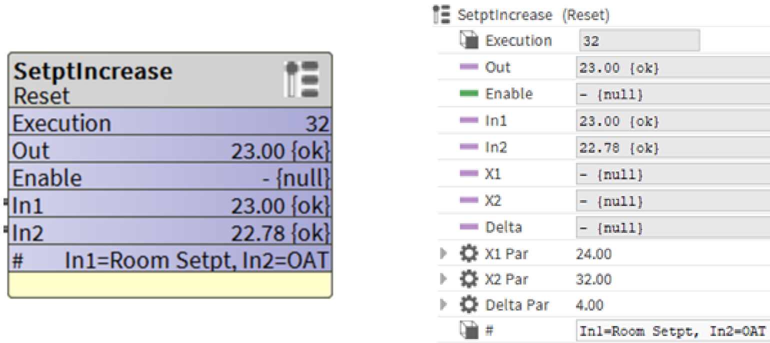


Fig. 111 Reset example 1

Example 2: Reset (SetptIncrease)

If the outside temperature In2 exceeds 24 °C (**X1 Par**), the room cooling setpoint with 23 °C (**In1**) is raised. The new cooling setpoint temperature **Out = In1 + a percentage of Delta = 23.25 °C**.

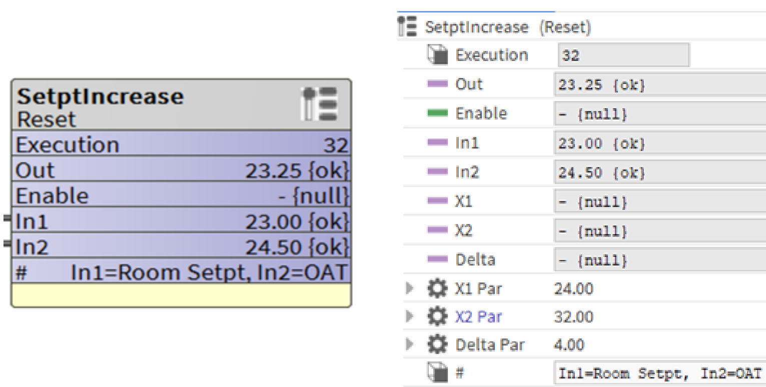


Fig. 112 Reset example 2

Example 3: Reset (SetptIncrease)

At the outdoor temperature of 28.16 °C (In2), the cooling setpoint **In1 = 23 °C** is increased by 2.08 °C, **Out = 25.08 °C**.

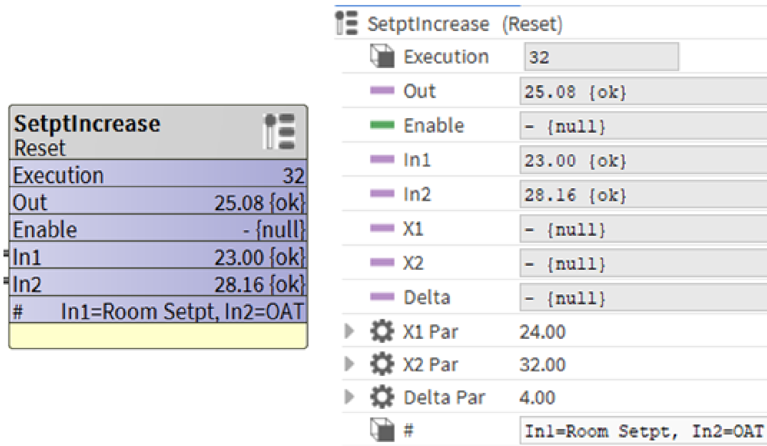


Fig. 113 Reset example 3

Example 4: Reset (SetptIncrease)

At the outdoor temperature of 32.57 °C (In2), the cooling setpoint **In1 = 23 °C** is increased by the full delta of 4 °C, **Out = 27 °C**.

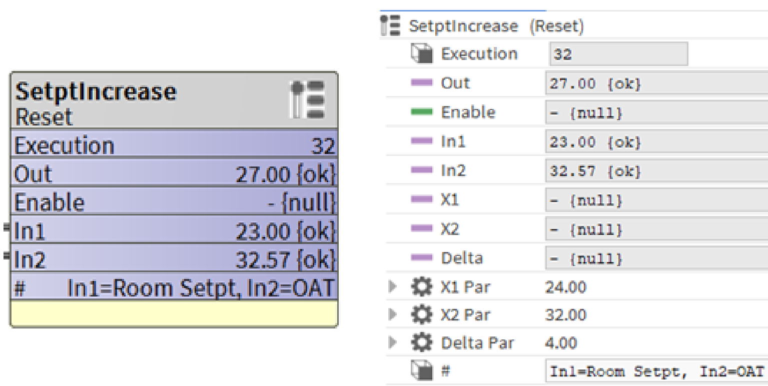


Fig. 114 Reset example 4

Subtract

The Subtract function block subtracts one input from the other. **Out** = **In A** - **In B**.

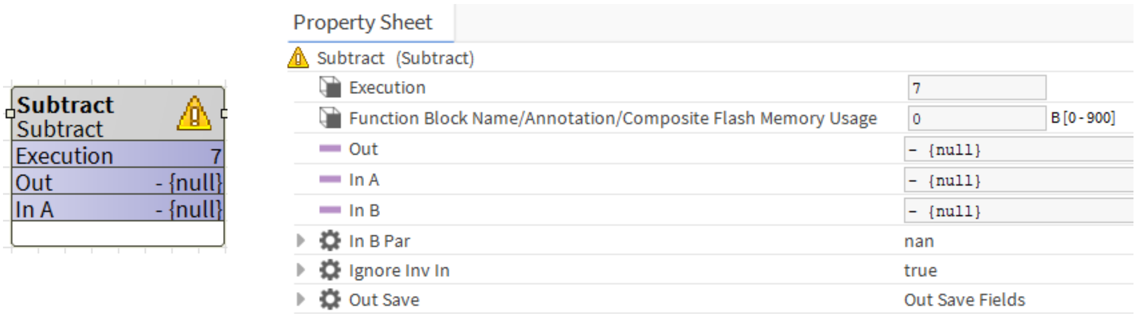


Fig. 115 Subtract Function Block and Property Sheet

If the result overflows the range of a single precision floating point number, the result becomes "null."

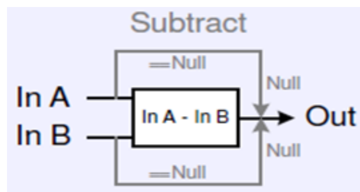


Fig. 116 Subtract Logic Diagram

Input

Table 87 Inputs of Subtract

Input Name	Description
In A	This is a 32-bit floating point input slot.
In B (In & Par)	This is a 32-bit floating point input slot.

Output

Table 88 Output of Subtract

Output Name	Description
Out	**Output: In A - In B**

Parameter

Table 89 Parameters of Subtract

Parameter Name	Description
In B Par (In & Par)	• If **In B** is "null," **In B Par** is used as a parameter. • If **In B Par** is set to "nan," it is treated like "null."
Ignore Inv In	• If this option is set to true, the function block considers only valid inputs while determining the subtraction of the inputs. • If this option is set to false, and any input becomes "null," the output becomes invalid. Default is "true."
Out Save	• **Master Sync Enabled:** If you set it to "TRUE," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Example 1: Subtract

Provide the output of InA - InB.

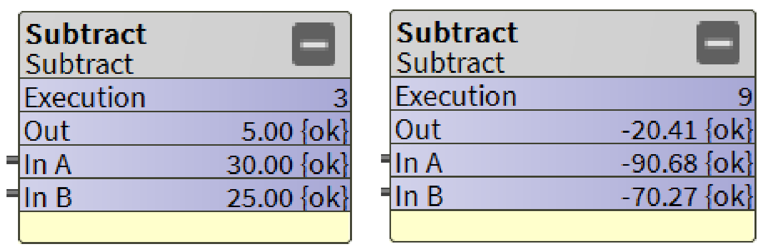


Fig. 117 Subtract Function Block

Digital Filter

The Digital Filter function block is used for smoothing or filtering values. Smoothing or filtering values for outdoor temperature, supply temperature, and exhaust air volume flow or brightness is possible. The filter works for both fast and slow-changing values (duct pressure and outside air temperature). This allows the fluctuations in measured values to be stabilized. The output can be set to zero or the first valid input value read in the first DDC cycle after powering up.

The function block internally remembers the previous **Out** value, which is required for the next DDC cycle's calculation of the new Out value.

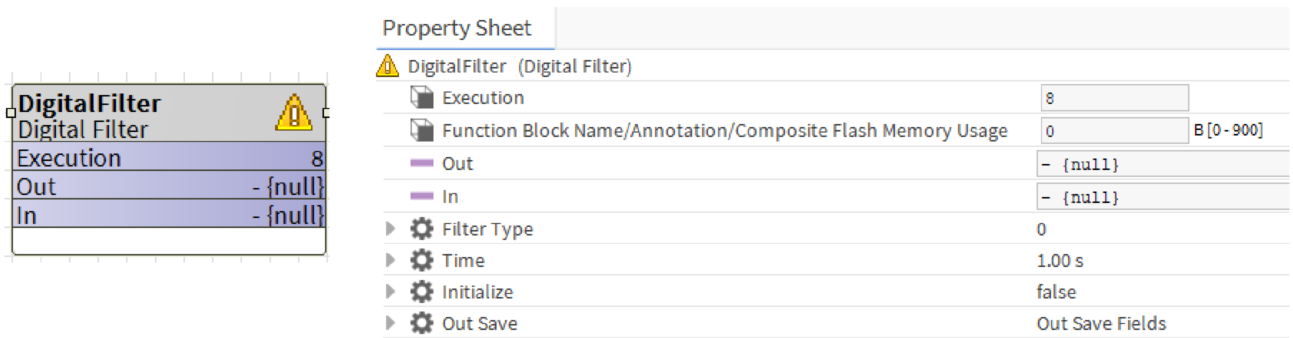


Fig. 118 DigitalFilter Function Block and Property Sheet

Input

Table 90 Input of Digital Filter

Input Name	Description
In	The **In** value is the base for the **Out** value. In terms of the Filter Type and the Time parameter, the **Out** follows the **In**. If **In** has a value of “Null,” “+inf,” or “-inf,” then **Out** is set immediately to the **In** value irrespective of the Filter Type and the Time. Similarly, if **In** changes from “Null” or “+inf” or “-inf” to a valid value, the valid value is immediately set on **Out**. ***Note:*** *If a temperature value is connected at the input In, which is read in via a Ui Terminal, this value can have the value +-inf/-inf for Sensor Open/Short. The +-inf values are immediately passed to the output Out.* **Numeric:** 32-Bit floating Point value.

Output

Table 91 Output of Digital Filter

Output Name	Description
Out	Output filtered value. **Numeric:** 32-Bit floating Point value.

Parameter

Table 92 Parameters of Digital Filter

Parameter Name	Description
Filter Type	**0:** PT1 filter **Numeric:** 32-Bit Numeric value: 0 - 32, Default: 0.
Time	Filter time constant (Sec). **Numeric:** 32-Bit Integer value: 1 - 65535, Default: 1.
Initialize	**False:** Initialize the filter with the first valid input value. **True:** Initialize the filter with value 0. **Default:** False
Out Save	• **Master Sync Enabled:** If you set it to “TRUE,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Examples

Example 1: Digital Filter

In = 15, Filter Type = 0, Time = 10 sec, Initialize = true

The below picture shows the value 11.30 after 7 Secs of an execution.

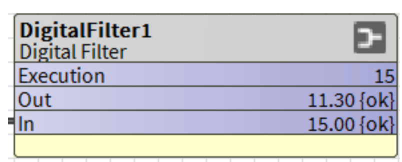


Fig. 119 DigitalFilter Function Block

Example 2: Digital Filter

In = 20, Filter Type = 0, Time = 10 sec, Initialize = true

The below picture shows the value 17.30 after 8.5 Secs of an execution.

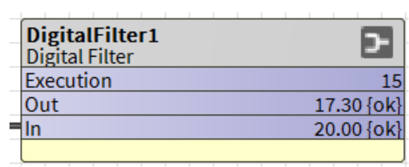


Fig. 120 DigitalFilter Function Block

Flow Velocity

The Flow Velocity function computes the flow and velocity based on the measured pressure and the K Factor. K Factor is often used in terminal unit controls to calculate the actual airflow. From iteration to iteration, the function block keeps track of the last state of the Auto Set Offset. On power-up or reset, this is set to “true” so that Auto Set Offset cannot be executed on a reset or power-up.

$$flow = K\sqrt{\Delta P - offset}$$

and

$$\mathrm{Vel} = \mathrm{flow/area}$$

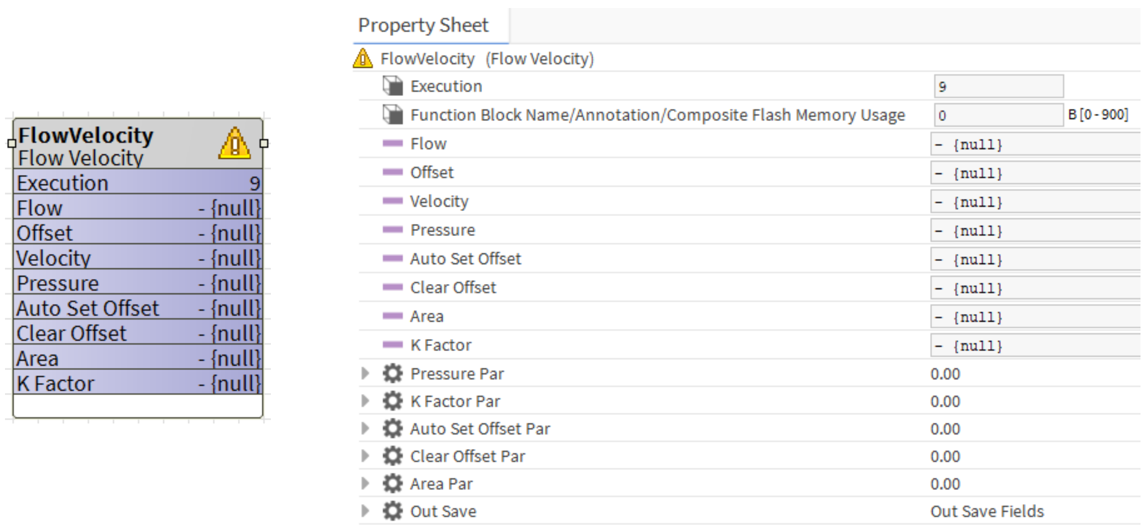


Fig. 121 FlowVelocity Function Block and Property Sheet

Input

Table 93 Inputs of Flow Velocity

Input Name	Description	Engineering Unit Imperial (Metric)
Offset (In & Par)	Input pressure offset correction to adjust the Flow to Zero. **Numeric:** 32-Bit Floating Point value >=-inf-<inf	inW (Pa)
Pressure (In & Par)	Air pressure is measured via a pitot tube. **Numeric:** 32-Bit Floating Point value >=-inf-<inf **Default:** Null	inW (Pa)
Auto Set Offset (In & Par)	Input correction pressure offset **Numeric:** 32-Bit Floating value: >=-inf-<inf **Default:** Null	inW (Pa)

Table 93 Inputs of Flow Velocity (Continued)

Input Name	Description	Engineering Unit Imperial (Metric)
Clear Offset (In & Par)	Reset correction pressure offset to value 0. **Numeric:** 32-Bit Floating value: >=-inf-<inf **Default:** Null	inW (Pa)
Area (In & Par)	Area (ft^2) **Numeric:** 32-Bit Floating value: >=-inf-<inf **Default:** Null	ft^2 (m^2)
K Factor (In & Par)	Actual flow corresponding to a velocity pressure sensor output of 1” w.g. or 249 Pa. **Numeric:** 32-Bit Floating value: >=-inf-<inf **Default:** Null	ft^3/min (m^3/hr)

Output

Table 94 Outputs of Flow Velocity

Output Name	Description	Engineering Unit Imperial (Metric)
Flow	Airflow **Numeric:** 32-Bit Floating point value, >=-inf-<inf	ft^3/min (CFM) or m^3/hr
Velocity	Air flow velocity **Numeric:** 32-Bit Floating point value, >=-inf-<inf	ft/min (m/hr)

Parameter

Table 95 Parameters of Flow Velocity

Parameter Name	Description	Engineering Unit (Metric/Imperial)
Flow Par	Airflow **Numeric:** 32-Bit Floating point value, >=-inf-<inf	ft^3/min (CFM) or m^3/hr
Offset Par	Input press Offset correction **Numeric:** 32-Bit Floating Point value >=-inf-<inf	inW (Pa)
Velocity Par	Air flow Velocity **Numeric:** 32-Bit Floating point value, >=-inf-<inf	ft/min (m/hr)

Table 95 Parameters of Flow Velocity (Continued)

Parameter Name	Description	Engineering Unit (Metric/Imperial)
Pressure Par	Air Pressure **Numeric: 32-Bit Floating Point value >=-inf-<inf** Default: Null	inW (Pa)
AutoSetOffSet Par	Input correction pressure offset **Numeric:** 32-Bit Floating value: >=-inf-<inf, **Default:** Null.	inW (Pa)
ClearOffSet Par	Reset correction pressure offset to value 0. **Numeric:** 32-Bit Floating value: >=-inf-<inf, **Default:** Null	inW (Pa)
Area Par	Area **Numeric:** 32-Bit Floating value: >=-inf-<inf, **Default:** Null	ft^2 (m^2)
KFactor Par	Actual flow in corresponding to a velocity pressure sensor output of 240 Pa or 1" w.g. **Numeric:** 32-Bit Floating value: >=-inf-<inf, **Default:** Null.	ft^3/min (m^3/hr)
Out Save	• **Master Sync Enabled:** If you set it to “TRUE,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Flow:** To enable or disable the **Flow** feature. • **Offset:** To enable or disable the **Offset:** feature. • **Velocity:** To enable or disable the **Velocity** feature.	

Example 1: FlowVelocity

If Pressure is 2.02 in per wc, offset is 0.02 in per wc, K Factor is 1015 cfm, and the area is 0.54 square feet (10 in. diameter), the flow will be 1015 ft per min, and velocity will be 1879 ft per min.

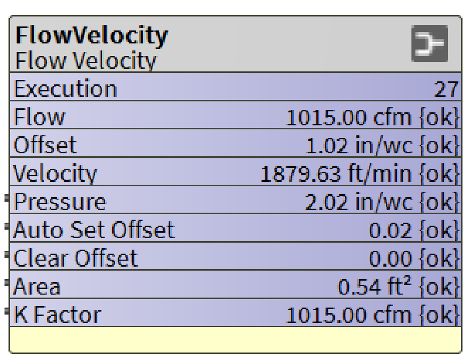


Fig. 122   FlowVelocity Function Block

Encode

The Encode function translates enumerations of a digital value into different enumeration numbers, allowing standard and custom enumerations to be combined and used together.

- If InEnum and one of the In values match, the appropriate output value is calculated, and Fire = “true.”
- If there is no match, Output = InEnum, and Fire = “false.” Disable stops the function and InEnum = Output.

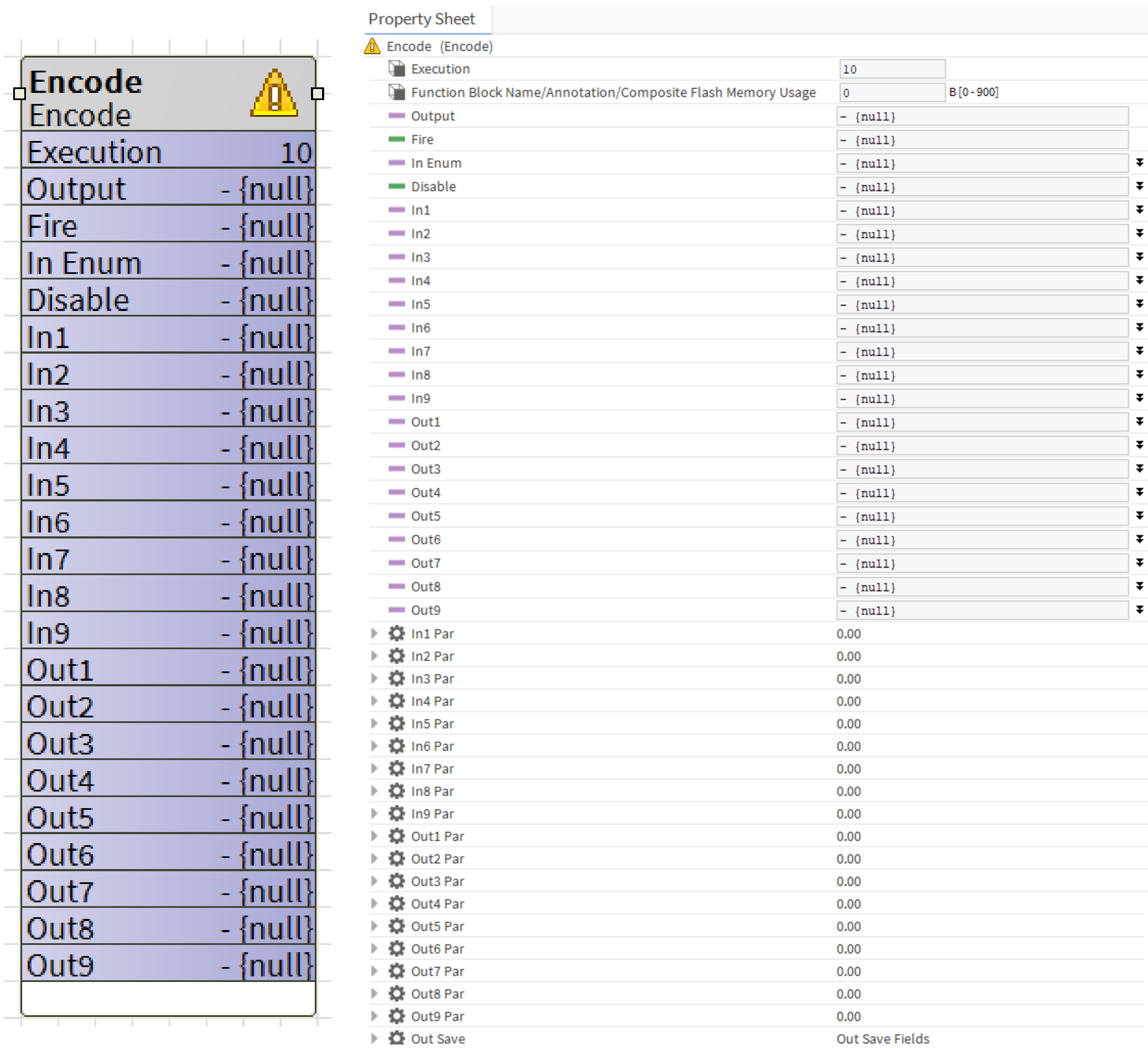


Fig. 123 Encode Function Block and Property Sheet

Input

Table 96 Inputs of Encode

Input Name	Description
Fire	If **InEnum** matches with any input **In1** to **In9**, then output Fire = 1. **Numeric:** Binary value, 0 – 1.
InEnum	If the value of **InEnum** is not matching with any of the **In1** to **In9** values, **Output = InEnum**, and FIRE = “false.” If Disable is “true” the **Output = InEnum**. **Numeric:** 32 Bit Floating value 0-255, **Default:** Null
Disable	Function disable (InEnum = Output). **Numeric:** Binary value 0-1, **Default:** Null
In1	**Numeric:** 32 Bit Floating value 0 – 255, **Default:** Null
In2	**Numeric:** 32 Bit Floating value 0 – 255, **Default:** Null
In3	**Numeric:** 32 Bit Floating value 0 – 255, **Default:** Null
In4	**Numeric:** 32 Bit Floating value 0 – 255, **Default:** Null
In5	**Numeric:** 32 Bit Floating value 0 – 255, **Default:** Null
In6	**Numeric:** 32 Bit Floating value 0 – 255, **Default:** Null
In7	**Numeric:** 32 Bit Floating value 0 – 255, **Default:** Null
In8	**Numeric:** 32 Bit Floating value 0 – 255, **Default:** Null
In9	**Numeric:** 32 Bit Floating value 0 – 255, **Default:** Null

Output

Table 97 Outputs of Encode

Output Name	Description
Out1	If **InEnum = In1**, **Out** = **In1**. **Numeric:** 32 Bit Floating value 0 – 255, **Default:** Null
Out2	If **InEnum = In2**, **Out = In2.** **Numeric:** Binary value 0 – 255, **Default:** Null
Out3	If **InEnum = In3**, **Out = In3.** **Numeric:** 32 Bit Floating value 0 – 255, **Default:** Null
Out4	If **InEnum = In4**, **Out = In4**. **Numeric:** Binary value 0 – 255, **Default:** Null
Out5	If **InEnum = In5**, **Out = In5**. **Numeric:** 32 Bit Floating value 0 – 255, **Default:** Null
Out6	If **InEnum = In6**, **Out = In6**. **Numeric:** Binary value 0 – 255, **Default:** Null

Table 97 Outputs of Encode (Continued)

Output Name	Description
Out7	If **InEnum = In7**, **Out = In7**. **Numeric:** 32 Bit Floating value 0-255, **Default:** Null
Out8	If **InEnum = In8**, **Out = In8**. **Numeric:** Binary value 0 - 255, **Default:** Null
Out9	If **InEnum = In9**, **Out = In9**. **Numeric:** Binary value 0 - 255, **Default:** Null

Parameter

Table 98 Parameters of Encode Function

Parameter Name	Description
In1 Par (In & Par)	**Numeric:** 32 Bit Floating value 0 - 255, **Default:** Null
In2 Par (In & Par)	**Numeric:** Binary value 0-1, **Default**: Null
In3 Par (In & Par)	**Numeric:** 32 Bit Floating value 0 - 255, **Default:** Null
In4 Par (In & Par)	**Numeric:** 32 Bit Floating value 0 - 255, **Default:** Null
In5 Par (In & Par)	**Numeric:** 32 Bit Floating value 0 - 255, **Default:** Null
In6 Par (In & Par)	**Numeric:** 32 Bit Floating value 0 - 255, **Default:** Null
In7 Par (In & Par)	**Numeric:** 32 Bit Floating value 0 - 255, **Default:** Null
In8 Par (In & Par)	**Numeric:** 32 Bit Floating value 0 - 255, **Default:** Null
In9 Par (In & Par)	**Numeric:** 32 Bit Floating value 0 - 255, **Default:** Null
Out1 Par	If **InEnum = In1**, **Out** = **In1**. **Numeric:** 32 Bit Floating value 0 - 255, **Default:** Null
Out2 Par	If **InEnum = In2**, **Out = In2**. **Numeric:** Binary value 0 - 255, **Default:** Null
Out3 Par	If **InEnum = In3**, **Out = In3**. **Numeric:** 32 Bit Floating value 0 - 255, **Default:** Null
Out4 Par	If **InEnum = In4**, **Out = In4**. **Numeric:** Binary value 0 - 255, **Default:** Null
Out5 Par	If **InEnum = In5**, **Out = In5**. **Numeric:** 32 Bit Floating value 0 - 255, **Default:** Null
Out6 Par	If **InEnum = In6**, **Out = In6**. **Numeric:** Binary value 0 - 255, **Default:** Null
Out7 Par	If I**nEnum = In7**, **Out = In7**. **Numeric:** 32 Bit Floating value 0 - 255, **Default:** Null
Out8 Par	If InEnum = In8, **Out** = In8. **Numeric:** Binary value 0 - 255, **Default:** Null
Out9 Par	If **InEnum = In9**, **Out = In9**. **Numeric:** Binary value 0 - 255, **Default:** Null

Table 98 Parameters of Encode Function (Continued)

Parameter Name	Description
Out Save	• **Master Sync Enabled:** If you set it to "TRUE," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Output:** To enable or disable the **Out** feature. • **Fire:** To enable or disable the **Fire** feature.

Examples

Example 1: Encode

If the value of InENum is not matching with any of the **In1** to **In9** values, **Output** = **InEnum** and Fire = "false."

InEnum = 15, Output = 15, Fire = "false."

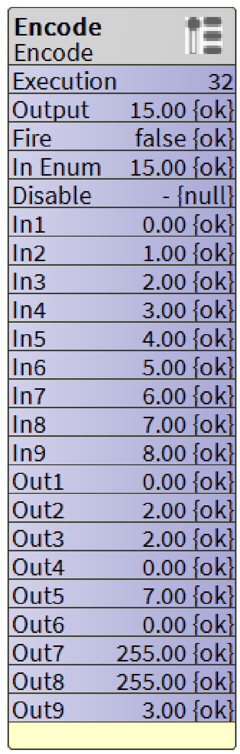


Fig. 124 Encode

Example 2: Encode

If the value of InENum is not matching with any of the **In1** to **In9** values, **Output = InEnum** and If Disable = “true”, Output = InEnum.

InEnum = 15, Output = 15, Disable = “true”, Fire = “true.”

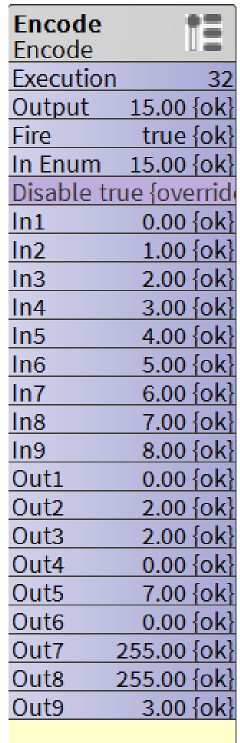


Fig. 125 Encode

To map a standard HVAC enumeration into a custom enumeration, the standard HVAC enumeration and desired mapping is as shown below table.

Table 99 HVAC Enumeration and Desired Mapping

In Parameter	Input Enumeration Configurations		Output Parameter #	Output Enumerations	
in 1	HVAC_AUTO	0	out1	COOL_MODE	0
in 2	HVAC_HEAT	1	out2	HEAT_MODE	2
in 3	HVAC_MORNING-WARM_UP	2	out3	HEAT_MODE	2
in 4	HVAC_COOL	3	out4	COOL_MODE	0
in 5	HVAC_NIGHT_PURGE	4	out5	NIGHT_MODE	7
in 6	HVAC_PRECOOL	5	out6	COOL_MODE	0
in 7	HVAC_OFF	6	out7	OFF_MODE	255
in 8	HVAC_TEST	7	out8	OFF_MODE	255
in 9	HVAC_EMERGENCY_HEAT	8	out9	EMERGE_HEAT	3

Table 99 HVAC Enumeration and Desired Mapping (Continued)

Block 2 passed through	HVAC_FAN_ONLY	9	Block2 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	Block2Out1	REHEAT_MODE	1

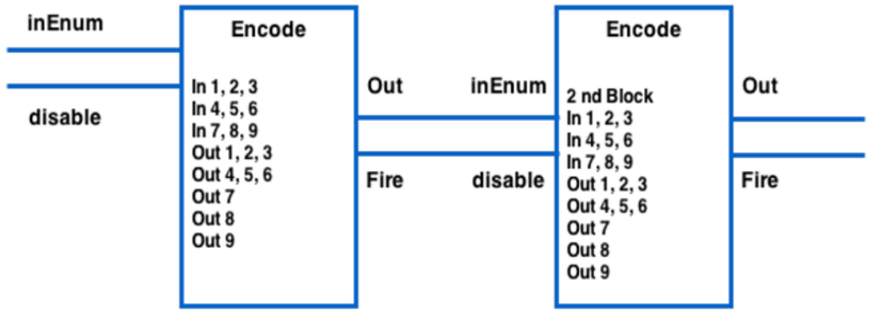


Fig. 126 Encode

Example 3: Encode

InEnum = 4, that is HVAC_NIGHT PURGE, Output is in NIGHTMODE = 7.

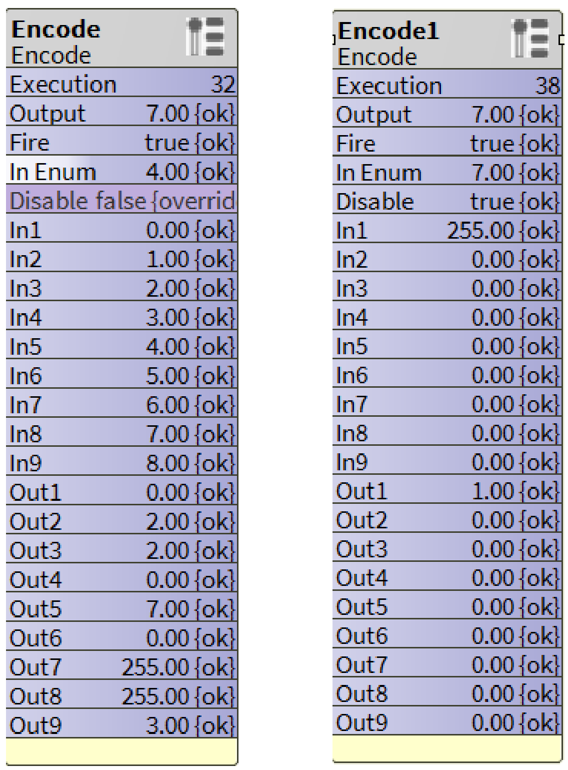


Fig. 127 Encode

CHAPTER

5

BIT FUNCTIONS FUNCTION BLOCKS

The following Bit function blocks are available in the honIrmControl Palette that can be configured and used to create the required application logic:

- Bit And
- Bit Or
- Numeric To Bit

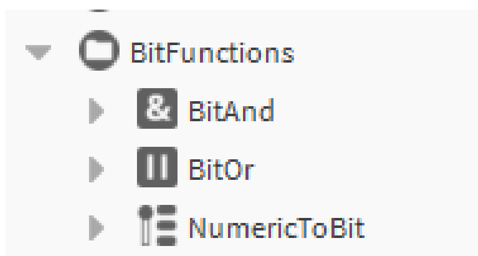


Fig. 128 BitFunctions Function Blocks

Bit And

The Bit And function block provides the bitwise output by performing logical AND operation bitwise between inputs **In** and **Mask** value.

It works on the bits of the input rather than the actual value of the input. Bitwise binary AND does the logical AND of the bits in each position of a number in its binary form. It compares each input bit to the corresponding bit of the mask value. If both bits are 1, the corresponding result bit is set to “1.” If not, the corresponding result bit is set to “0.”

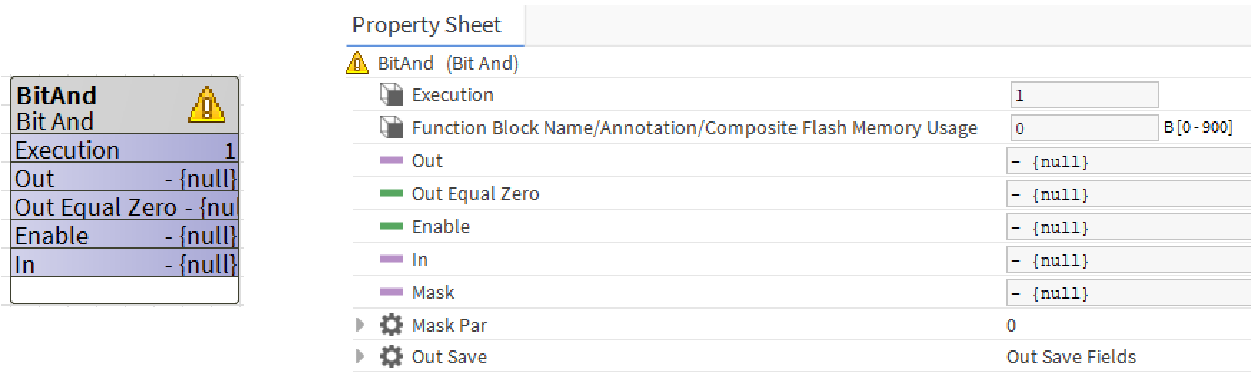


Fig. 129 BitAnd Function Block and Function Block and Property Sheet

If **In** or **Mask** = “null,” **Out** and Out Eq Zero = “null.” If Enable is set to “false.” **Out** = **In** and EqZero = false.

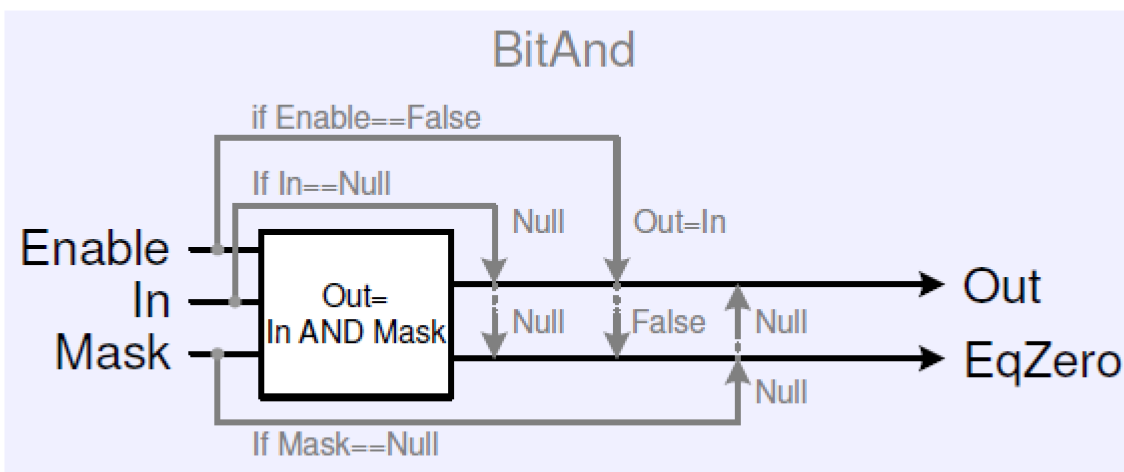


Fig. 130 BitAnd Logic Diagram

Input

Table 100 Inputs of BitAnd

Input Name	Description
Enable	This is a boolean point. If it is set to “true,” it enables the function block.
In	Integer value.
Mask (In & Par)	Integer value.

Output

Table 101 Outputs of BitAnd

Output Name	Description
Out	**Out**: **In** BitAnd **Mask**
Out Equal Zero	If **Out** is zero, then **Out Equal Zero** is “true.”

Parameter

Table 102 Parameters of BitAnd

Parameter Name	Description
Mask Par (In & Par)	If **Mask** is “null” or not connected, **Mask Par** is used as a parameter.
Out Save	• **Master Sync Enabled:** If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out Save** feature. • **Out Equal Zero:** To enable or disable the **Out Equal Zero** feature.

Examples

Example 1: BitAnd (In set to 6 and Mask set to 5)

In is "6" and **Mask** is "5."

- Set **Enable** to "true."
- Convert "6" and "5" to binary value that is "110" and "101."
- Make an AND logic for each bit. 1 AND 1 give 1; anything else gives 0.
- Convert the value from binary to an integer from "100" to "4". Function block passes the value as output.

Table 103 Bit And Example (In is 6 and Mask is 5)

Name of the inputs and output	Integer	Binary		
In	6	1	1	0
Mask	5	1	0	1
Out	4	1	0	0
Output Equal Zero	false			

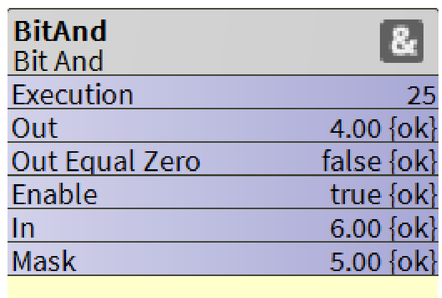


Fig. 131 In "6" and Mask "5"

Example 2: BitAnd (In set to 4 and Mask set to 3)

In is "4" and **Mask** is "3".

- Set **Enable** to "true."
- Convert "4" and "3" to binary value that is "100" and "011".
- Make an AND logic for each bit. 1 AND 1 give 1; anything else gives 0.
- Convert the value from binary to an integer from "000" to "0". Function block passes the value as output.

Table 104 Bit And Example (In is 4 and Mask is 3)

Name of the inputs and output	Float	Binary		
In	4	1	0	0
Mask	3	0	1	1
Out	0	0	0	0
Output Equal Zero	true			

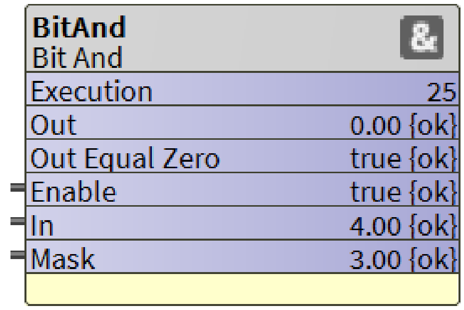


Fig. 132 In “4” and Mask “3”

Example 3: BitAnd (In set to 4 and Mask set to “3)

In is “4” and **Mask** is “3.”

- **Enable** is set to “false.”
- If the **Enable** is “0,” the output value is the same value of input which is 4. However, **Out** = “0” and it disables the calculation, and “false” will be set as output because **Out** is not zero.

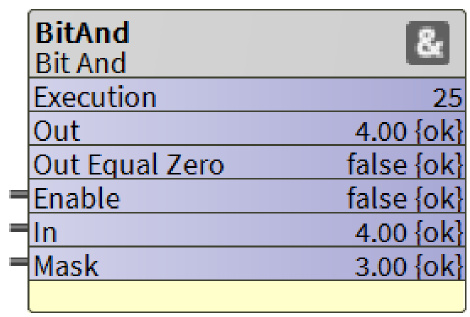


Fig. 133 In “4” and Mask “3”

Bit Or

Bit Or function block provides the bitwise output by performing logical OR operation bitwise between input **In** and **Mask** value.

Operator compares each binary digit across two integers and gives back "1" if either of the inputs is "1." If not, it returns to "0".

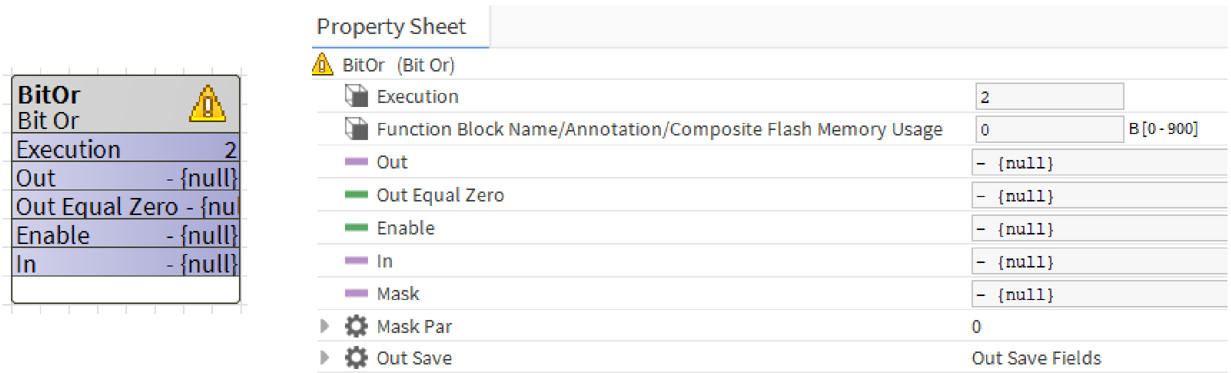


Fig. 134 BitOrFunction Block and Property Sheet

If **In** or **Mask** = "null," **Out** and EqZero = "null". If **Enable** = "false" **Out** = **In** and EqZero = "false".

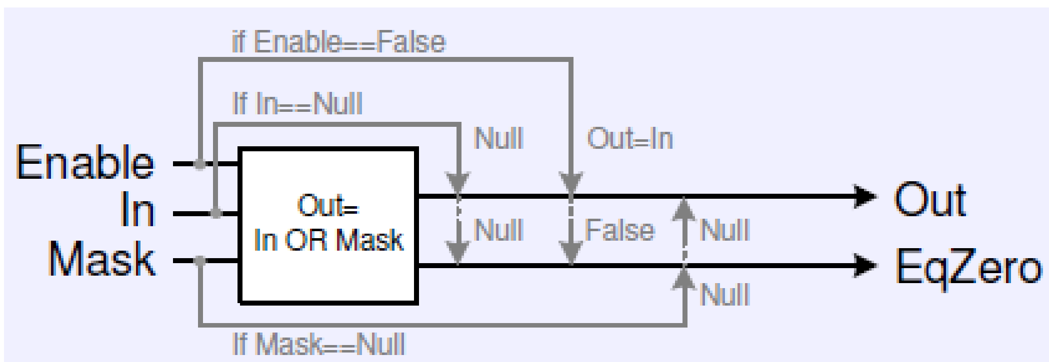


Fig. 135 BitOr Logic Diagram

Input

Table 105 Inputs of Bit Or

Input Name	Description
Enable	This is a Boolean point. If it is set to true, it enables the function block.
In	Integer value.
Mask (In & Par)	Integer value.

Output

Table 106 Outputs of Bit Or

Output Name	Description
Out	**Out**: **In** BitOr **Mask**
Out Equal Zero	If **Out** is zero, **Out Equal Zero** is “true.”

Parameter

Table 107 Parameters of Bit Or

Parameter Name	Description
Mask Par (In & Par)	If **Mask** is “null” or not connected, **Mask Par** is used as a parameter.
Out Save	• **Master Sync Enabled:** If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out Save** feature. • **Out Equal Zero:** To enable or disable the **Out Equal Zero** feature.

Examples

Example 1: Bit Or (In set to “6” and Mask set to “5”)

- Set Enable to “true.”
- Convert “6” and “5” to binary value that is “110” and “101.”
- Make an OR logic for each bit. 0 OR 0 gives 1; anything else gives 1.
- Convert the value from binary to an integer from “111” to “7.” Function block passes the value as output.

Table 108 Bit Or Example (In is 6 and Mask is 5)

Name of the inputs and output	Float	Binary		
In	6	1	1	0
Mask	5	1	0	1
Out	7	1	1	1
Output Equal Zero	false			

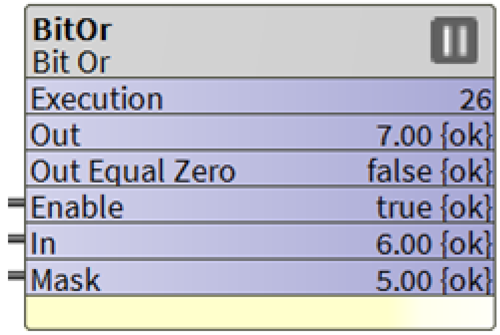


Fig. 136 In “6” and Mask “5”

Example 2: BitOr (In set to 0 and Mask set to 0)

- Set Enable to “true.”
- Convert “0” and “0” to binary value that is “000” and “000”.
- Make an OR logic for each bit. 0 OR 0 gives 1; anything else gives 1.
- Convert the value from binary to an integer from “000” to “0.” Function block passes the value as output.

Table 109 Bit Or Example (In is 0 and Mask is 0)

Name of the inputs and output	Float	Binary		
In	0	0	0	0
Mask	0	0	0	0
Out	0	0	0	0
Output Equal Zero	true			

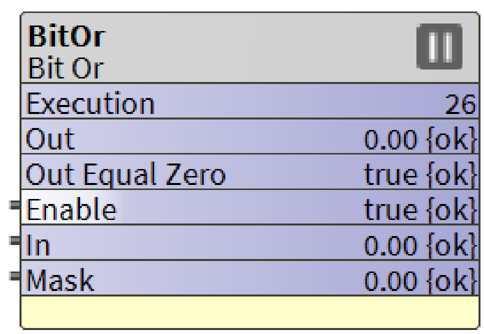


Fig. 137 In “0” and Mask “0”

Example 3: BitOr (In set to 4 and Mask set to 3)

Enable is set to “false.”

If the Enable is “0,” the output value is the same value of input which is 4. However, **Out** = “0,” and it disables the calculation, and “false” will be set as output.

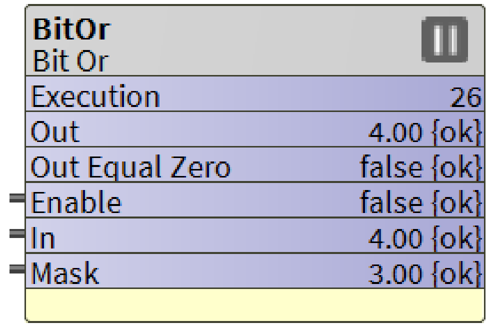


Fig. 138 In “4” and Mask “3”

Numeric To Bit

The Numeric To Bits function block converts an integer value into a maximum of 24 bits. Often, bits are used for configuration since a 3-byte number (24 bits) can be used to handle up to 24 different meanings. Each bit can be set individually to 0 or 1. This function determines each individual bit from one of the 3 bytes of the integer value, whereby each bit can be "0", "1," or "null."

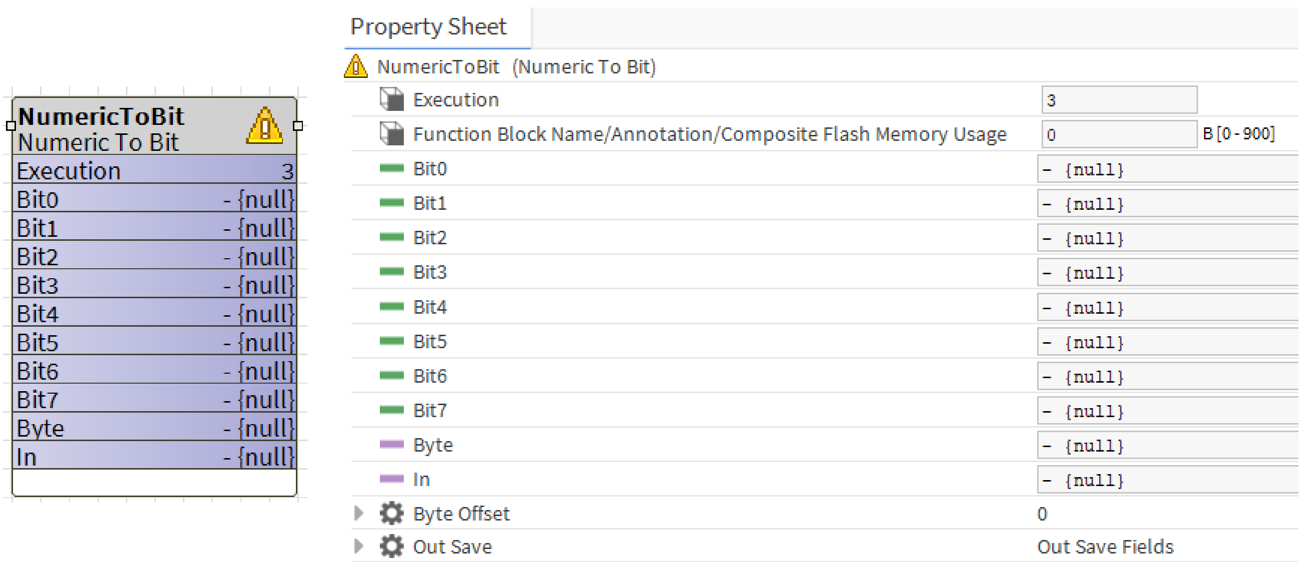


Fig. 139 NumericToBit Function Block and Property Sheet

Input

Table 110 Input of Numeric To Bit

Input Name	Description
In	This input represents the integer number to be divided into bits. The integer number is a maximum of 4 bytes long (32 bits). **Range:** 0 - 16777215 or "null." If **In** is "null," all outputs are "null."

Output

Table 111 Outputs of Numeric To Bit

Output Name	Description
Bit0 – Bit7	Represents the bits, where bit0 represents the least significant bit. The bits are either 0 or 1. If **In** is set to "null," **Bit0 - Bit7** is set to "null."
Byte	The byte value represents an integer value of the **Byte** selected by the offset. The **Byte** has the value 0 - 255, or if **In** is set to "null," **Byte** is set to "null."

Parameter

Table 112 Parameters of Numeric To Bit

Parameter Name	Description
Byte Offset	If the input value **In** consists of several bytes, you can select the byte which is to be divided into bits. 0 = lowest byte (outer right). Valid Range is 0 - 2 (0 is the default).
Out Save	• **Master Sync Enabled:** If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out Save** feature. • **Bit0:** To enable or disable the **Bit0** feature. • **Bit1:** To enable or disable the **Bit1** feature. • **Bit2:** To enable or disable the **Bit2** feature. • **Bit3:** To enable or disable the **Bit3** feature. • **Bit4:** To enable or disable the **Bit4** feature. • **Bit5:** To enable or disable the **Bit5** feature. • **Bit6:** To enable or disable the **Bit6** feature. • **Bit7:** To enable or disable the **Bit7** feature. • **Byte:** To enable or disable the **Byte** feature.

Examples

Example 1: NumericToBits calculation

The value 26434 is to be broken down into bits; the value corresponds to 67 42 in hexadecimal representation, consisting of 2 bytes. The lower byte is on the right (42 hex) and the higher byte on the left (67 hex).

- To split the lower byte, select byte Offset = 0.
- To split the next byte, select Offset = 1.
- With Offset = 0, the lower byte (42 hex) represents the integer value (Output Byte) as value 66 and the binary value 0100 0010.

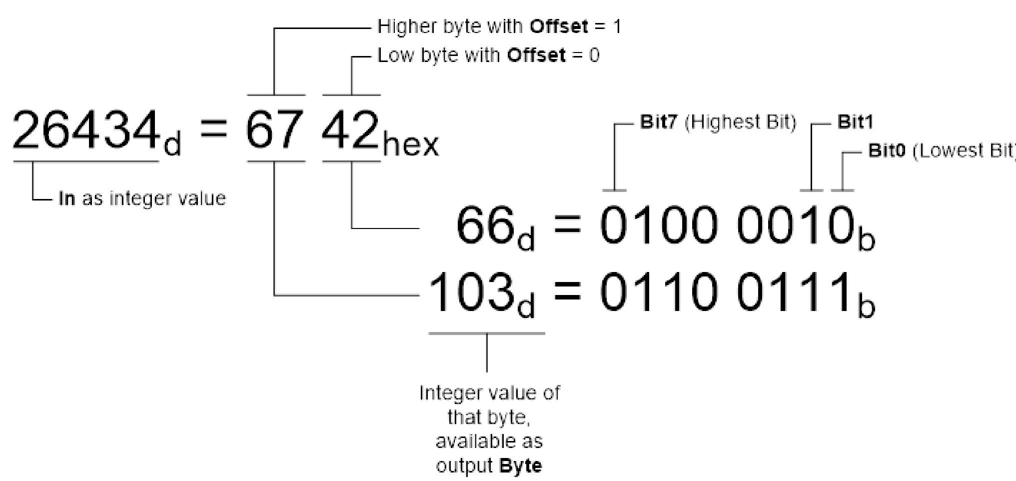


Fig. 140 NumericToBits calculation example

Example 2: NumericToBits

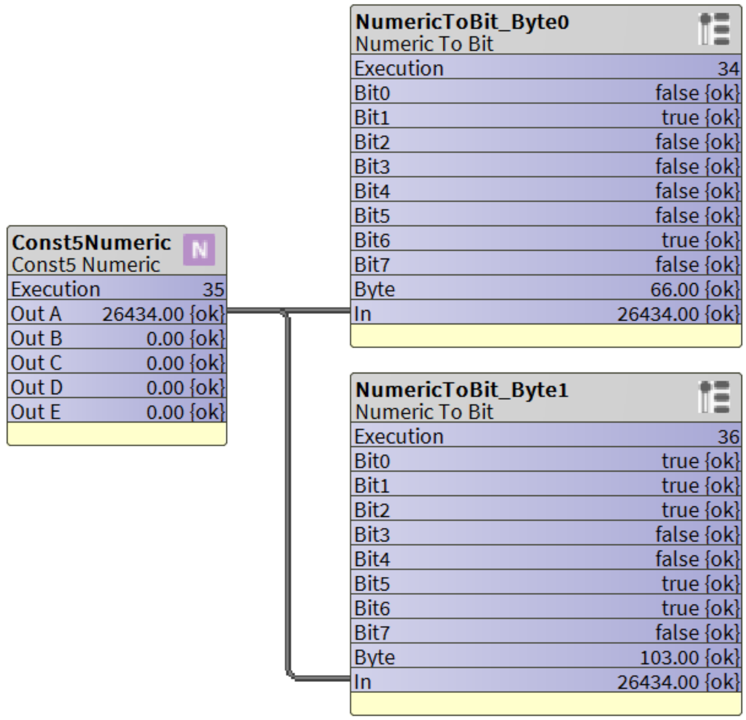


Fig. 141 NumericToBits function block

NumericToBits Property Sheet Byte 0 and Property Sheet Byte 1.

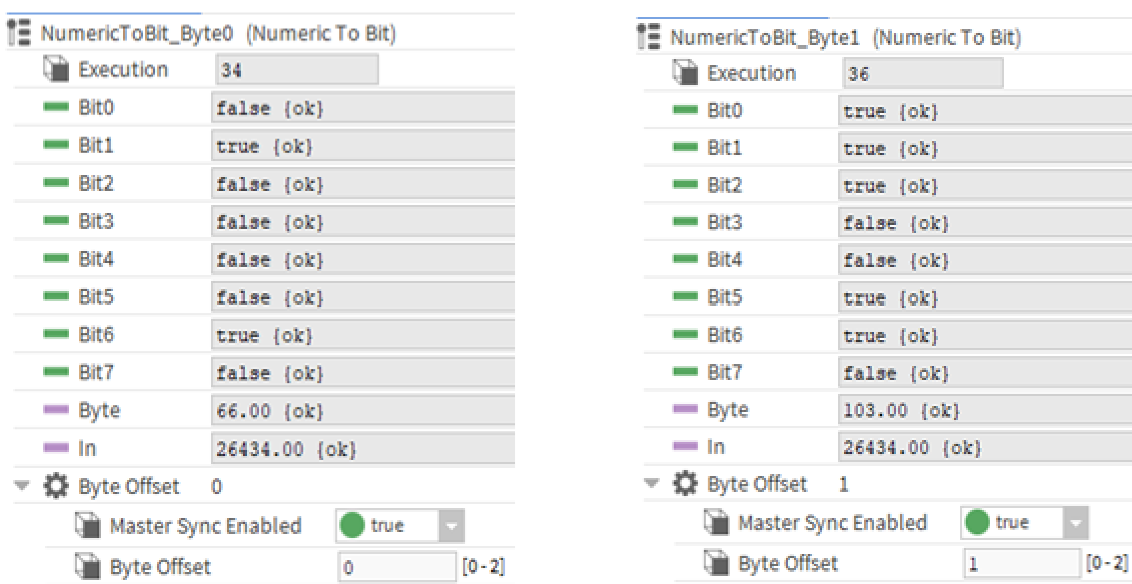


Fig. 142 NumericToBits Property Sheet

CHAPTER 6

CONTROL LOOP FUNCTION BLOCKS

The following Control Loop function blocks are available in the honIrmControl Palette that can be configured and used to create the required application logic:

- Aia
- PidA
- Pid
- Stager
- Stage Driver
- Flow Control
- Cycler

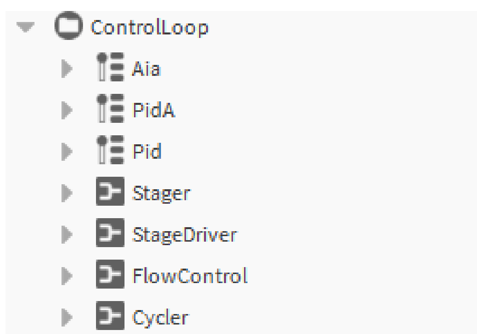


Fig. 143 Control Loop Function Blocks

Aia

The PID could be replaced with an adaptive integral action controller (AIA). To maintain stability even in the presence of parametric interceptions, adaptive integral control utilizes proportional–integral control and dynamic gains calculations. When delays in the controlled process causes integral windup, (either undershooting or overshooting) leads to instability, this control works better than PID.

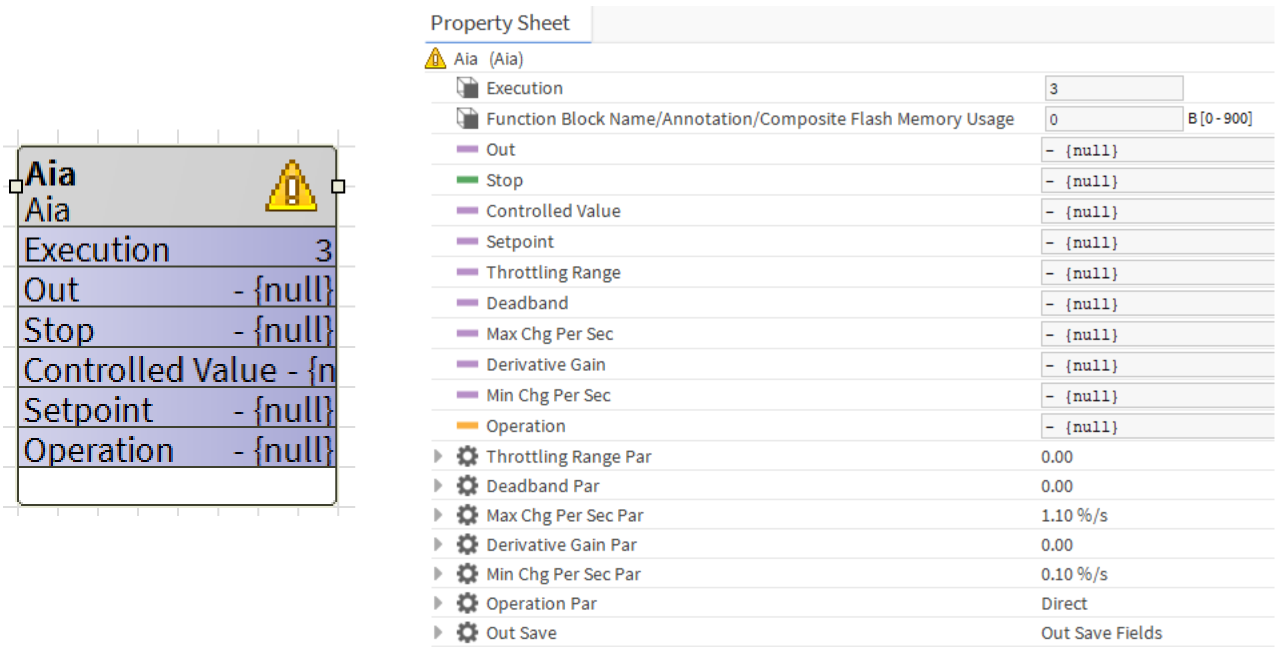


Fig. 144 Aia Function Block and Property Sheet

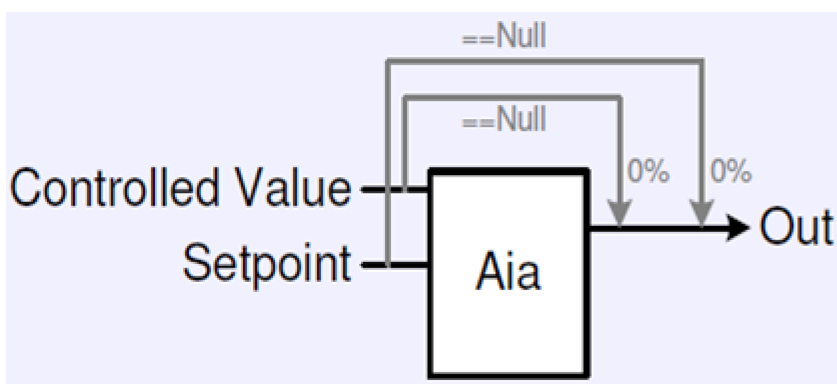


Fig. 145 Aia Logic Diagram

Err = Sensor – SetPoint

The Err term is set to – Err if Direct or Reverse is set to reverse. The function block keeps track of the old proportional error from iteration to iteration. This is cleared when you power up or reset.

Input

Table 113 Inputs of Aia

Input Name	Description
Controlled Value	Input from the sensor present value (temperature, pressure, etc.).
Setpoint	It is a 32-Bit floating set point, which is used to maintain the controlled value.
Throttling Range (In & Par)	It is the same as the proportional band.The Output rate of change in value will be decided based on error and Throttling Range deviation. • If the error value is equal to the throttling value, then the rate of change in output will be the maximum value per sec. • If the error value is 50 % of the throttling value, then the rate of change in output will be the mid-value between the minimum and maximum value per sec.
Deadband (In & Par)	The absolute value of the error must be higher than it was before the output changed. EffErr = Err – deadband. • If Err > 0, ErrSign = 1 else ErrSign = –1. • If |Err| < dead band, then AbsErr = 0. Otherwise (|Err| > deadband), AbsErr = |Err| – deadband. Output = Output + ErrSign * [(maxAOchng – minAO) * (AbsErr / (ThrottlingRange – Deadband)) * 3 + MinAO)].
Max Chg Per Sec (In & Par)	This is the maximum amount (%) that output will change for a single cycle of the control. This is set to 100 % / (Actuator speed (second/full stroke))/ (DDC cycles per second). **Example**: 100 %/150 sec/2=0.333 % per sec.
Derivation Gain (In & Par)	It gives a controller additional control action when the error changes consistently. It also makes the loop more stable (up to a point), which allows using a higher controller gain and a faster integral.
Min Chg Per Sec (In & Par)	This is the minimum amount (%) that output will change for a single control cycle if the setpoint is not reached.
Operation (In & Par)	**Direct:** “0” = Direct If the CV >SP, then the calculated output value will vary from 0-100 %. **Example:** Cooling. **Reverse:** “1” = Reverse If the CV <SP, then the calculated output value will vary from 0-100 %. **Example**: for heating and pressure application. **Off “2”:** operations will stop.

Output

Table 114 Outputs of Aia

Output Name	Description
Out	**Output:** Output + ErrSign * NonLin (AbsErr, ThrottlingRange, MaxAOchange, MinAOchange)
Stop	It is a boolean point. • If it is set to "true," the block will disable the Aia function block. • If it is set to "false," it enables the Aia function block.

Parameter

Table 115 Parameters of Aia

Parameter Name	Description
Throttling Range Par (In & Par)	If **Throttling Range** is "null," **Throttling Range Par** is used as a parameter.
Deadband Par (In & Par)	If **Deadband** is "null," **Deadband Par** is used as a parameter.
Max Chg Per Sec Par (In & Par)	If **Max Chg Per Sec** is "null," **Max Chg Per Sec Par** is used as a parameter.
Derivative Gain Par (In & Par)	If **Derivative Gain** is "null," **Derivative Gain Par** is used as a parameter.
Min Chg Per Sec Par (In & Par)	If **Min Chg Per Sec** is "null," **Min Chg Per Sec Par** is used as a parameter.
Operation Par (In & Par)	The **operation** specifies the direction of the controller. For example, a different direction is necessary for heating mode than in cooling mode. There are three **operations**: • Direct (0) • Reverse (1) • Off (2)
Out Save	• **Master Sync Enabled:** If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Examples

Example 1: Aia (Direct operation)

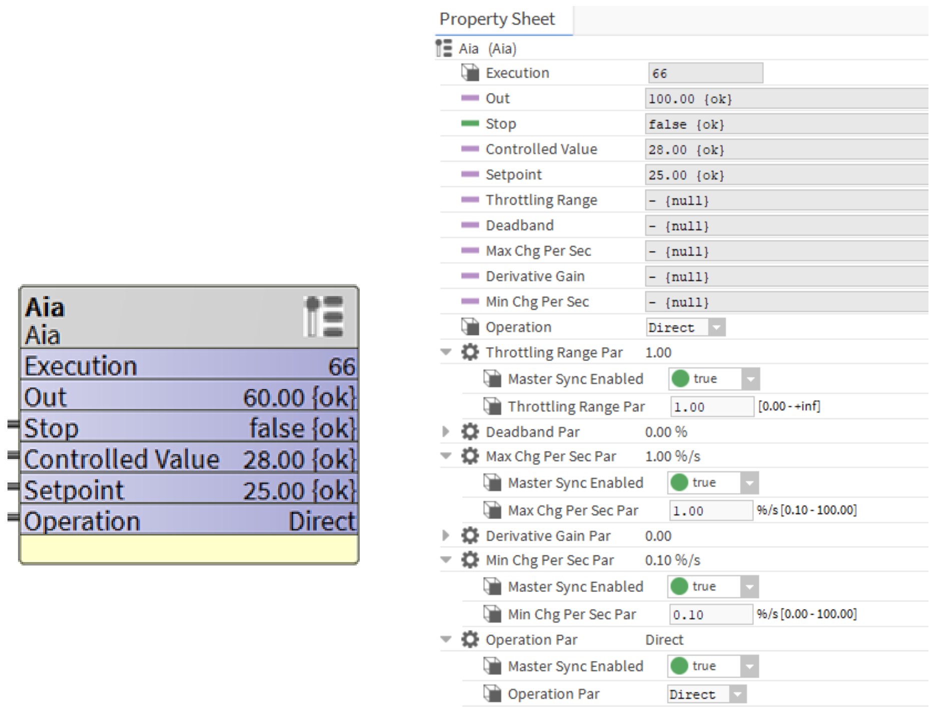


Fig. 146 Aia with Direct Operation – Property Sheet

Example 2: Aia (Reverse operation)

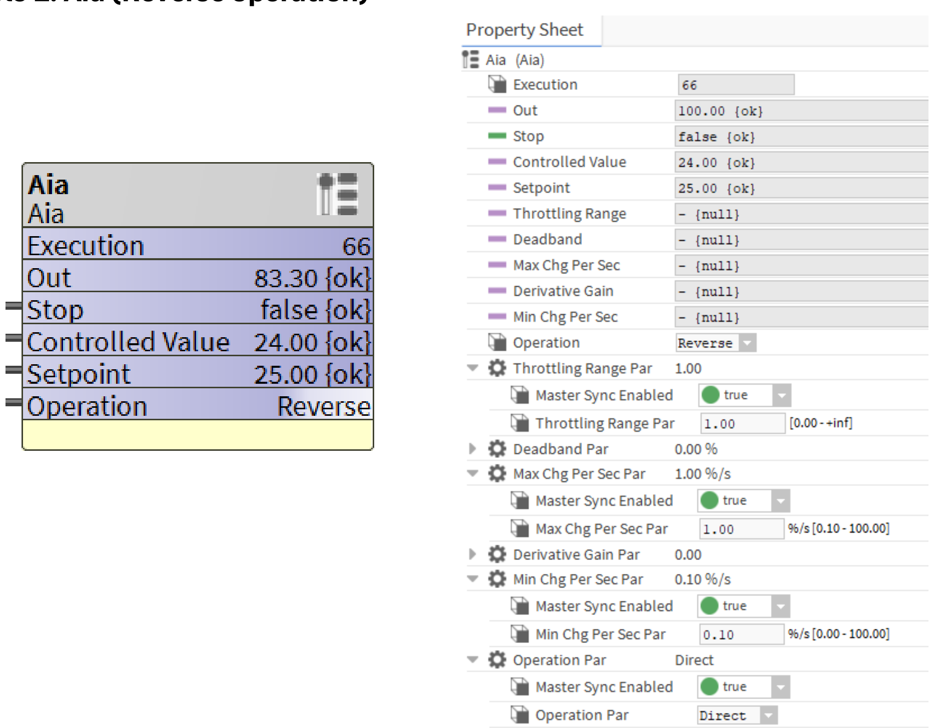


Fig. 147 Aia with Reverse Operation – Property Sheet

PidA

The PidA function block is designed to break away from the backward compatibility of older PID implementations. This function block is used for various applications such as temperature, pressure, humidity, CO2, and flow control. The control output is calculated based on the setpoint and the controlled value. Depending on the configuration, the output will have a value ranging from 0 to 100 % or from -100 to 100 %.

The application can trigger the controller at any time interval using a manual input, and the control output can be set to a manual value, which then serves as the start value for further PID control. The proportional part can be deactivated by setting Proportional Band = 0, allowing for a pure integral or an integral and differential control algorithm. In this case, the formula utilities an internal Proportional Band = 1.

A change of the operation mode resets the Integral Part. If the control deviation is greater deadband, then a new PID output value is calculated after the deadband delay has expired.

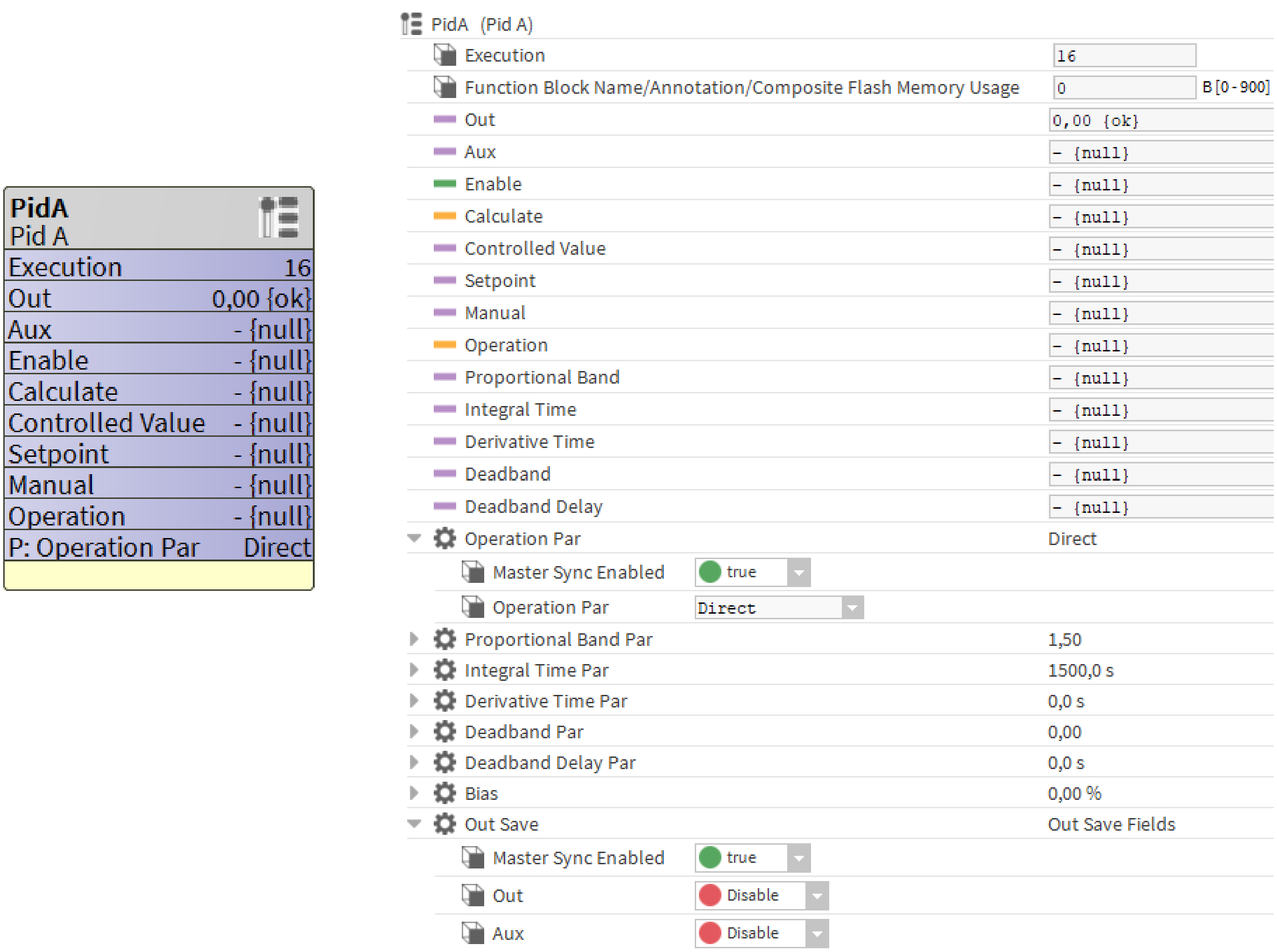


Fig. 148 PidA Function Block and Property Sheet

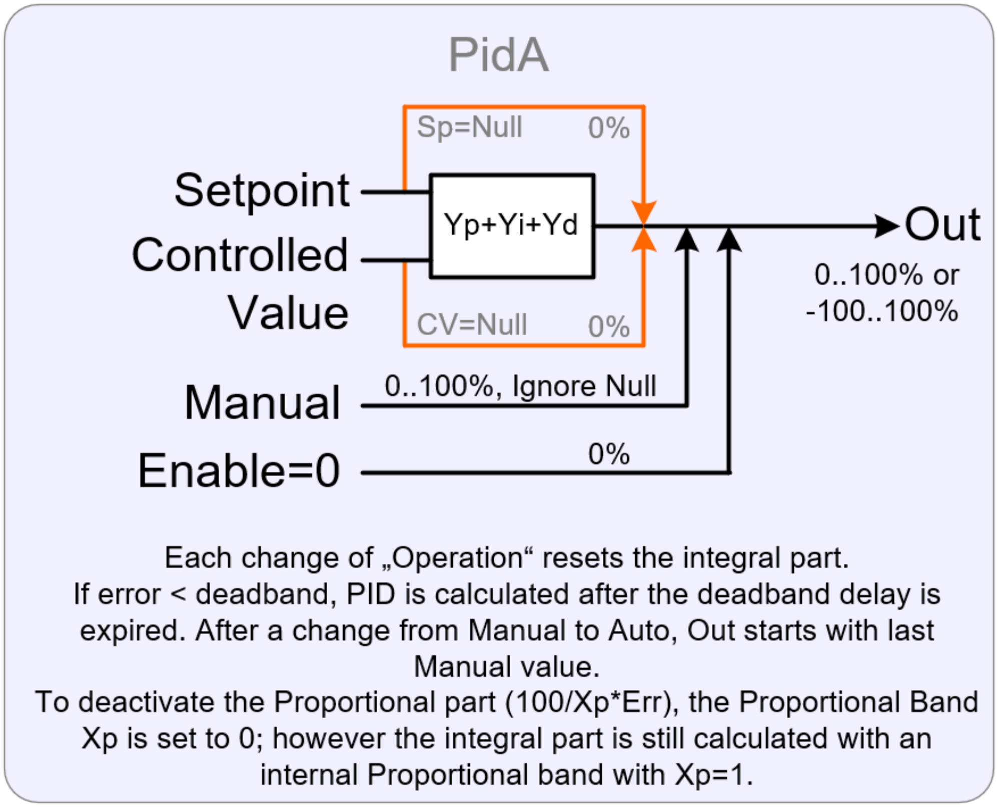


Fig. 149 PidA Logic Diagram

Formula

Error

- **Direct Mode:** Error = Setpoint - Controlled Value
- **Reverse Mode:** Error = Controlled Value – Setpoint

Proportional Part

- P = 100 / Xp * Error
- P = 0 if Xp = 0
- P is limited to 0..100 %

Integral Part

- I = I(old) + 100 / (Xp * Tn) * (Error * TimeDelta)
- I = 0 if if Tn = 0
- If Xp = 0, then use Xp = 1
- I is limited to 0..(100 - Bias - 100/Xp * Error)
- I is limited to be 0..100 %

The integral Part is reset when the Operation mode changes.

Derivative Part

- D = 100 / Xp * Td * Error(Delta) / TimeDelta; Delta(Delta) = Error – OldError.

If Xp = 1, then use if Xp = 0

PID Output

- Out (%) = Bias + 100/Xp * Error + 100/(Xp*Tn) * Integral(Error)*dt + 100/Xp* Td*dError/dt

Out is limited to be 0..100% or -100..100 % depending on the operation mode.

Out is calculated if abs (Setpoint - Controlled Value) >= Deadband or if the Deadband Delay time is expired.

Input

Table 116 Inputs of PidA

Input Name	Description
Enable	The input enables or disables the calculation of the PID algorithm. • **Bool:** 0: Out = 0 %, **1:** Out = PID enabled, **Default:** Null = PID enabled
Calculate	Normally the PID algorithm is calculated according to the DDC interval. With a constantly changing actual value, this leads to a constantly changing output. The PID calculation can be triggered from the application via this input. The PID calculation can be stopped and triggered via this input. If relays are used to control the valve drives, this input can be used to prevent the relays from being switched on and off frequently. • With “Remain old Out“, the I-component remains at its old value and no new I-component is added to the integral. The integration is stopped and the D-component is set to 0 %. • With "Calculate PID", the I-component is integrated again and the D-component is calculated again.The DeltaTime used for the formula is measured in the function block; it is the time since the last calculation of the I and D components. • With "Calculate P only" only the P component is recalculated while the I component remains at its old value. **Enum:** - 0 = Remain old Out, - 1= Calculate PID, 2 = Calculate P only, Default = Null = Calculate PID any DDC cycle
Controlled Value	The input represents the present sensor value (temperature, pressure, etc.). **Numeric:** 32-Bit floating point value (float), Default = Null
Setpoint	The input represents the setpoint controlled by the PID algorithm. **Numeric:** 32-Bit floating point value (float), Default = Null
Manual	The PID output can be set to a specific value via this input. When changing from Manual to Auto, the PID output starts at the previous **Manual** value. This behavior is achieved by correcting the integral part internally as long as a manual value is available. Example: A critical valve is opened to 50 % manually. This input is then switched to Auto. The PID output starts regulation at 50 %. **Numeric:** 32-Bit floating point value (float), Default = Null = Auto

Table 116 Inputs of PidA (Continued)

Input Name	Description
Operation (In & Par)	PidA will decide the operation based on this input. Four modes of operation: **Off:** Means that the P+I+D parts are set to 0 %. The internal integral timer is stopped, however **Manual** and **Bias** are valid. **Direct:** Means that the output increases to get a larger actual value. If a heating valve is opened more, then there is a higher temperature; if an air damper is opened more, then there is more air volume; if the light is dimmed up, then there is more brightness. If **Controlled Value < Setpoint**, the calculated **Out** value varies from 0 to 100 %. **Reverse:** Means that the output increases to get a smaller actual value. If a cooling valve is opened more, then there is a lower temperature; if an air damper is opened more, then there is less CO2 in the room. The output is from 0 to 100 %. If **Controlled Value > Setpoint**, the calculated **Out** value varies from 0 to 100 %. **Direct And Reverse:** Means that the output increases to get a larger actual value. In this operating mode, however, the output is between -100 % and 100 %. A typical application is to achieve a constant outlet temperature with a cooler and heater by outputting negative values to the cooler and positive values to the heater. However, this type of control should not be used to control room temperatures, as it only recognizes a single setpoint. If **Controlled Value < Setpoint**, the calculated Out value varies from -100 to 100 %. **Enum:** **1:** Off **2:** Direct **3:** Reverse **4:** Direct and Reverse **Default:** Null = Use **Operation Par** ***Note:*** *Typically, the operation mode is from the **wall moduleA** function block. In this case, the output **OutSetptMd** (Off, Clg, Htg) is connected to **Operation** via a **NumericSelect** function block.*
Proportional Band (Xp) (In & Par)	**Proportional Band** behavior depends on **Operation**. The Xp value is used to calculate both the P component and the I and D components. Small values lead to fast control behavior, but tend to oscillate. The P component can be deactivated via the value 0, the I and D components are still calculated. A pure integral controller can thus be configured via the value 0. **Recommended settings for different applications** • For room temperature control with FCU: Xp = 34.7 °F (1.5 °C), Tn =1500 sec, Td = 0 sec • For constant light control: Xp = 0 Lux, Tn = 20000, Td = 0 sec (with LightA) or Xp = 600, Tn = 70 sec, Td = 0 sec

Table 116 Inputs of PidA (Continued)

Input Name	Description
	Numeric: 32-Bit floating point value (float), Default = Null = Use **Proportional Band Par**
Integral Time (Tn) (In & Par)	The Tn value is used to calculate the Integral component. The integral component prevents a permanent deviation by increasing or decreasing the integral component after a periodic time interval (DDC time or from **Calculate** Input). 0 means that there is no Integral part calculated. **Numeric:** 32-Bit floating point value (float), Default = Null = Use Integral Time Par
Derivative Time (Td) (In & Par)	The Td value is used to calculate the Derivative component. It corrects the future error based on the current rate of change of the error by increasing or decreasing the derivative component. 0 means that there is no Derivative part calculated. **Numeric:** 32-Bit floating point value (float), Default = 0 sec = Use **Derivative Time Par**
Deadband (In & Par)	If the absolute difference between the setpoint and actual value is smaller than the deadband, an error of 0 is used for the calculation. The I component (and thus also the output "Out") remains at the last calculated value. A Deadband Delay is started. If the control deviation is smaller than the deadband, then the PID is calculated in the time intervals of the deadband delay. As soon as the control deviation is greater than the deadband, the timer is stopped and the PID is calculated in each DDC cycle. **Numeric:** 32-Bit floating point value (float), Default = 0 = Use **Deadband Par**
Deadband Delay (In & Par)	The Deadband Delay is used to slowly adjust even small control deviations without causing oscillation. If the control deviation is smaller than the deadband, then the Deadband Delay is started. If the Deadband Delay is expired then the PID is calculated and the timer is restarted.If the control deviation is higher than the deadband, then the Deadband Delay is stopped. **Numeric:** 32-Bit floating point value (float), Default = 0 sec = Use **Deadband Delay Par**

Output

Table 117 Outputs of PidA

Output Name	Description
Out	This is the calculated controller output, calculated based on the PID formula. This output is typically connected to an output function block. With a cooling/heating control, the output is connected either to a cooling or heating output depending on the Setpoint Mode of the wall module. **Numeric:** 32-Bit floating point value (float), 0..100 % or -100..100 % depending on **Operation**.
Aux	The output is currently not used. It will be used for future enhancements.

Parameter

Table 118 Parameters of PidA

Parameter Name	Description
Operation Par (In & Par)	Since this parameter is also available as an input, please refer to the description of the input **Operation**. **Default:** 2 = Direct
Proportional Band Par (In & Par)	Since this parameter is also available as an input, please refer to the description of the input **Proportional Band**. **Default:** 1.5 K
Integral Time Par (Tn) (In & Par)	Since this parameter is also available as an input, please refer to the description of the input **Integral Time**. **Default:** 1500 sec
Derivative Time Par (Td) (In & Par)	Since this parameter is also available as an input, please refer to the description of the input **Derivative Time**. **Default:** 0 sec = Derivate Part is disabled
Deadband Par (In & Par)	Since this parameter is also available as an input, please refer to the description of the input **Deadband**. **Default:** 0 = No deadband
Deadband Delay Par (In & Par)	Since this parameter is also available as an input, please refer to the description of the input **Deadband Delay**. **Default:** 0sec = No delay
Bias	The bias value is added to the output as an offset after the PID calculation. **Numeric:** 32-Bit floating point value (float), Default = Null = 0 %
Out Save	• **Master Sync Enabled:** If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature. • **Aux:** To enable or disable the **Aux** feature.

Examples

Example 1: PidA (Cooling + Heating)

Cooling out value modulates from 0 to 100 %.

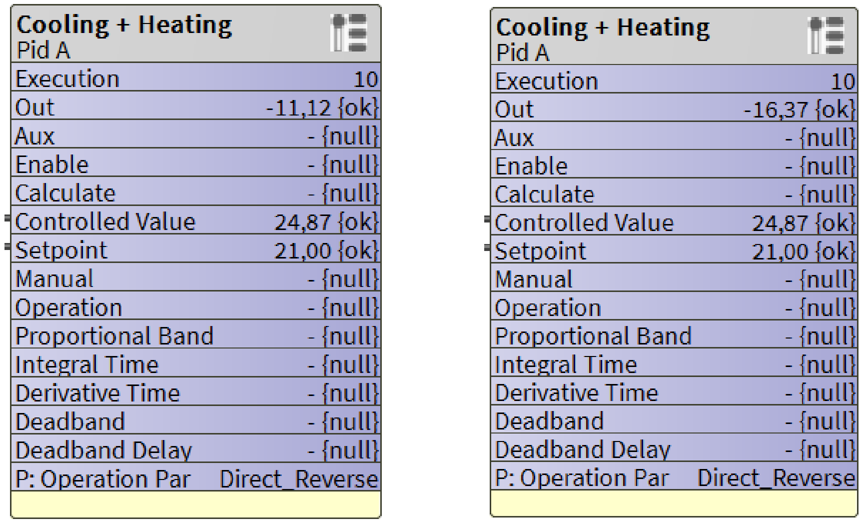


Fig. 150 Cooling: The Out value modulates to 0 to 100 %

Example 2: PidA (Cooling + Heating)

The Out value moves to + 100 %.

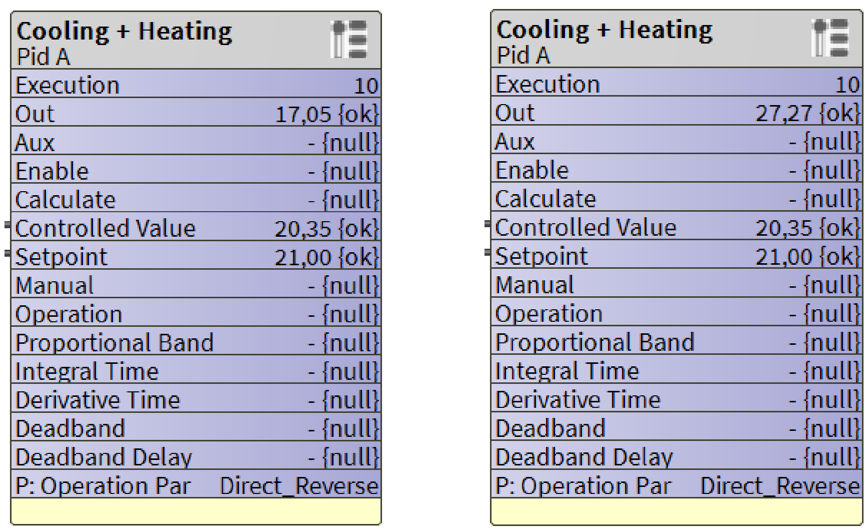


Fig. 151 Heating: The Out value modulates to + 100 %.

Example 3: PidA (Heating)

The Out value moves to +100 %.

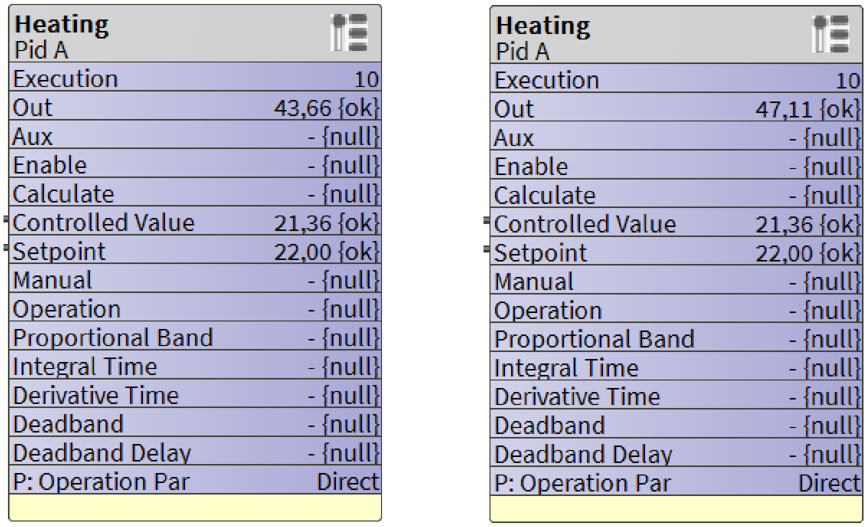


Fig. 152 Heating: The Out value modulates to +100 %

Example 4: PidA (Cooling)

The Out value moves to +100%.

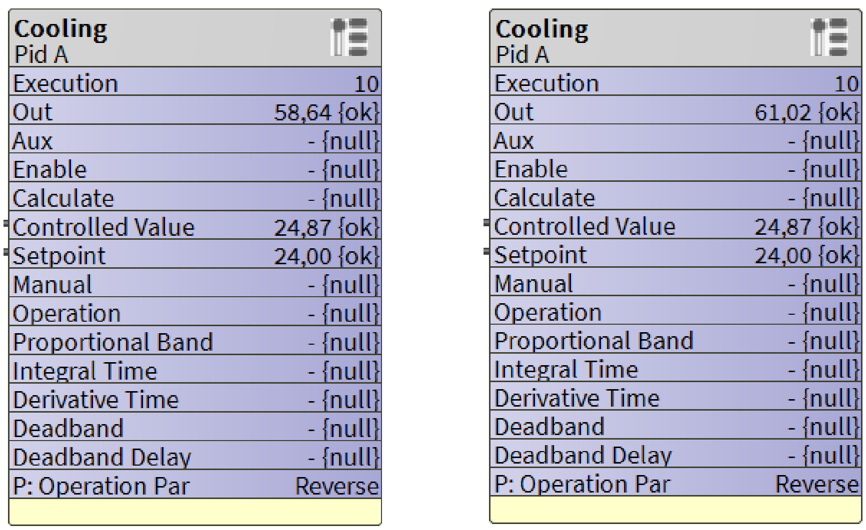


Fig. 153 Cooling: The Out value modulates to +100%.

Pid

The Pid functional block compares a process's measured value to a reference setpoint value. The difference (or error signal) is then used to calculate a new value for a manipulatable process input that returns the measured value to the desired set point. The Pid functional block, unlike simpler control algorithms, can adjust the process outputs based on the error signal's history and rate of change, resulting in more accurate and stable control.

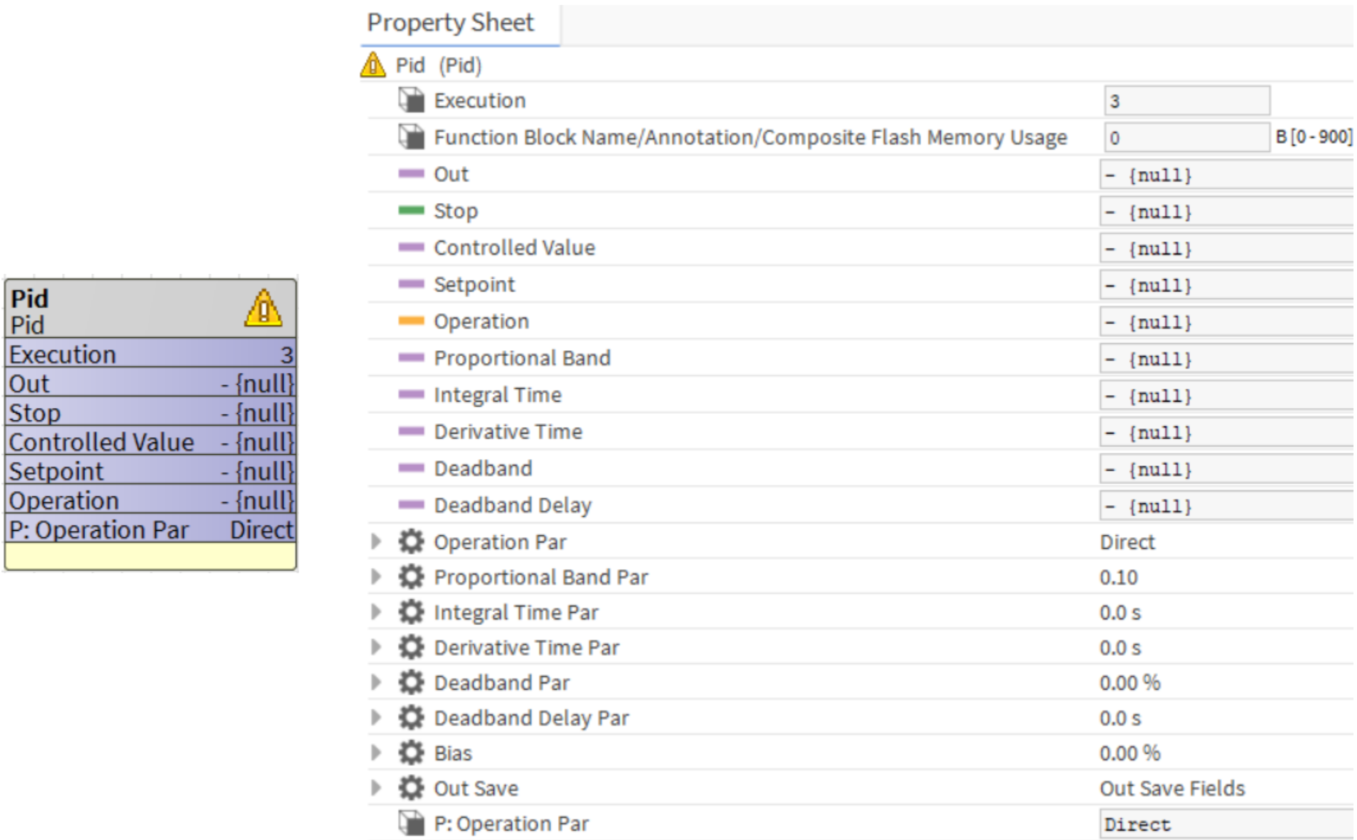


Fig. 154 Pid Function Block and Property Sheet

Each change of **operation** resets an integral part. If error < deadband, Pid is calculated after the **Deadband Delay** is expired. After changing from manual to auto, **Out** starts with the last manual value.

To deactivate the proportional part (100/Xp*Err), the **Proportional Band** Xp is set to 0; however, the integral part is still calculated with an internal **Proportional Band** with Xp=1.

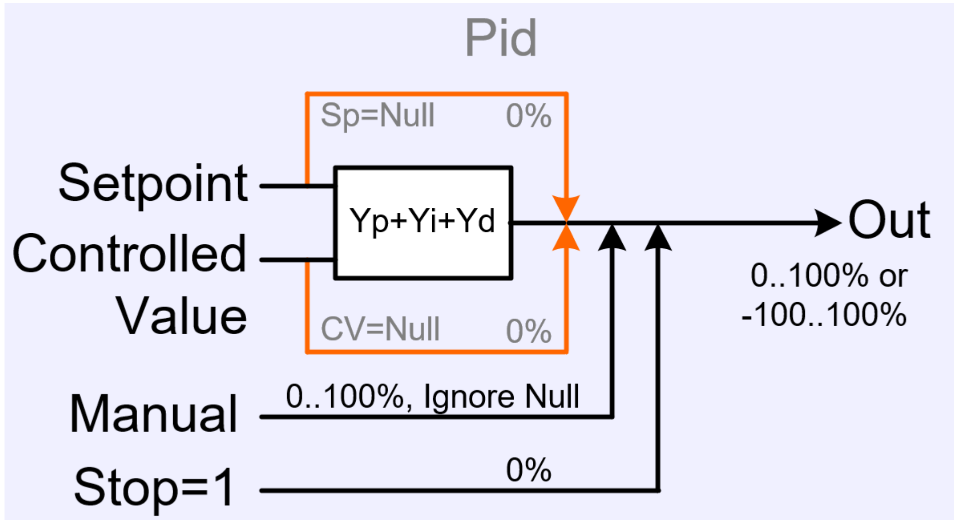


Fig. 155 Pid Logic Diagram

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 (seconds)
Bias: proportional offset (%)

$$\text{Output (\%)} = \text{bias} + Kp*Err + Kp/Ti \int_0^t (Err)\,dt + Kp*Td*dErr/dt$$

Example:

- If Direct or Reverse is set to reverse, Err term is set to Err.
- If 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 deadband timer. On power-up/reset, these are cleared.

Input

Table 119 Inputs of Pid

Input Name	Description
Stop	Used to enable or disable the PID calculation. **Bool:** 0: PID enabled **1:** Out = 0 % **null:** PID enabled
Controlled Value	It represents the actual measured value.
Setpoint	It is a 32-Bit floating setpoint. It represents the setpoint controlled by the PID algorithm.
Operation (In & Par)	This is Enum point. There are three operations: **Direct:** 0 = Direct (Clg - 0 - 200 %)**. Proportional Band** = 0 > Out = 0 %. **Reverse:** 1= Reverse (Htg - 0 -200 %). **Proportional Band** = 0 > **Out** = 0 %. **DependingOnSign:** 2 = Depending on sign of **Throttling Range**. • If throttling range < 0, it is reverse action. • If throttling range >= 0, it is direct action.
Proportional Band (In & Par)	It cancels out the current error directly.
Integral Time (In & Par)	The amount of time the error has continued uncorrected. The **Integral Time** is limited to 0 (**100 –Bias –100/Xp * Err**). If Xp = 0 -> Then use Xp=1 for the integral part.
Derivative Time (In & Par)	It anticipates the future error from the rate of change of the error over time.
Deadband (In & Par)	Deadband is the absolute value that error must be greater than before the output changes.
Deadband Delay (In & Par)	If the **deadband** is activated and the **deadband** timer has expired, a new PID calculation is performed. The **Deadband Timer** is restarted. It ensures that even small control deviations are softly compensated without oscillation.

Output

Table 120 Outputs of Pid

Output Name	Description
Out	**Output (%) :** Bias+ 100/Xp * Err+ 100/(Xp*Tn) * Integral (Err)*dt+ 100/Xp* Td*dErr/dt

Parameter

Table 121 Parameters of Pid

Parameter Name	Description
Operation Par (In & Par)	This is Enum point. There are three operations: **Direct:** 0 = Direct (Clg - 0 - 200 %). **Proportional Band** = 0 > Out = 0 %. **Reverse:** 1= Reverse (Htg - 0 - 200 %). **Proportional Band** = 0 > Out = 0 %. **DependingOnSign:** 2 = Depending on sign of **Throttling Range**. • If **throttling range** < 0, it is reverse action. • if **throttling range** >= 0, it is direct action.
Proportional Band Par (In & Par)	If **Proportional Band** is "null," **Proportional Band Par** is used as a parameter.
Integral Time Par (In & Par)	If **Integral Time** is "null," **Integral Time Par** is used as a parameter.
Derivative Time par (In & Par)	If **Derivative Time** is "null," **Derivative Time Par** is used as a parameter.
Deadband Par	If **Deadband** is "null," **Deadband Par** is used as a parameter.
Deadband Delay Par (In & Par)	If **Deadband Delay** is "null," **Deadband Delay Par** is used as a parameter.
Bias	Defines the amount added to the output for pure P-Control.
Out Save	• **Master Sync Enabled:** If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Examples

Example 1: Pid

Application with PI with Direct and Reverse operations.

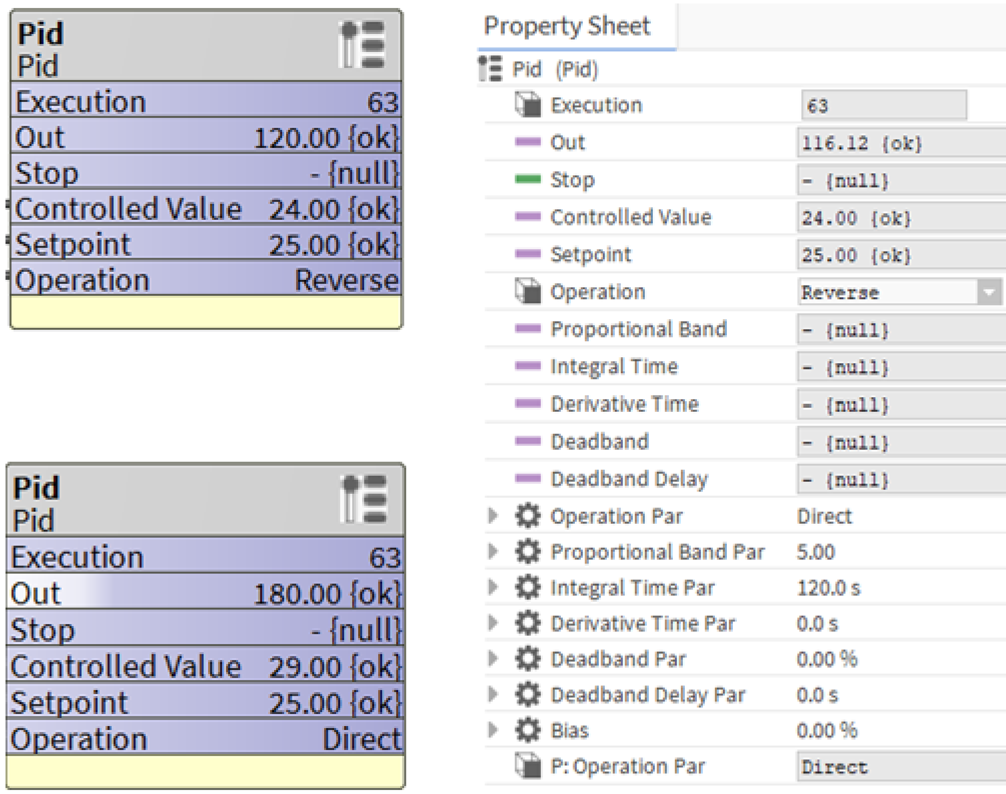


Fig. 156 Pid with Direct and Reverse operations

Example 2: Pid

Application with P with Direct and Reverse operations

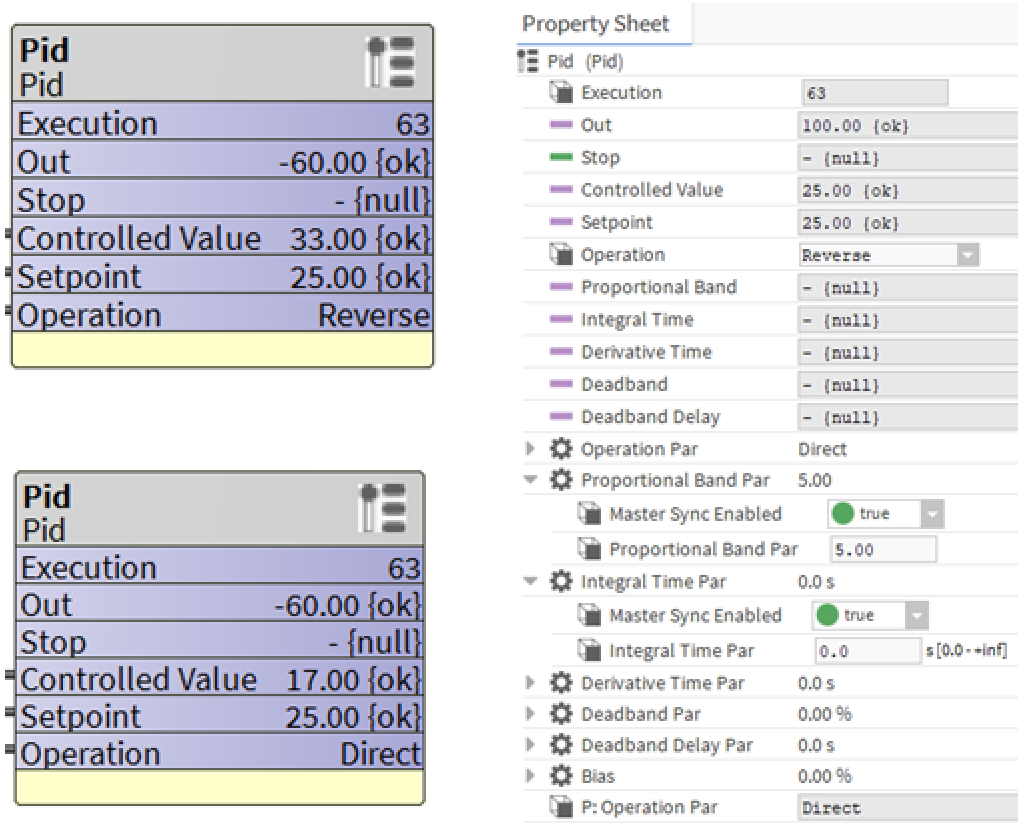


Fig. 157 Pid with Direct and Reverse operations

Stager

The Stager function block determines how many stages of turning on from a 0 to 100 percent (typically Pid error) input. The input range of 0 to 100 percent is divided evenly among the number of stages configured in MaxStages.

The first stage is activated when CmdPercent > 0 and deactivated when CmdPercent - Hyst. The general criterion for turning on stage N is: CmdPercent > (N 1)*100 % /MaxStages, as shown in the diagram below.

The following is the criterion for turning off stage N: CmdPercent (N -1) *100 %/ MaxStages –Hyst.

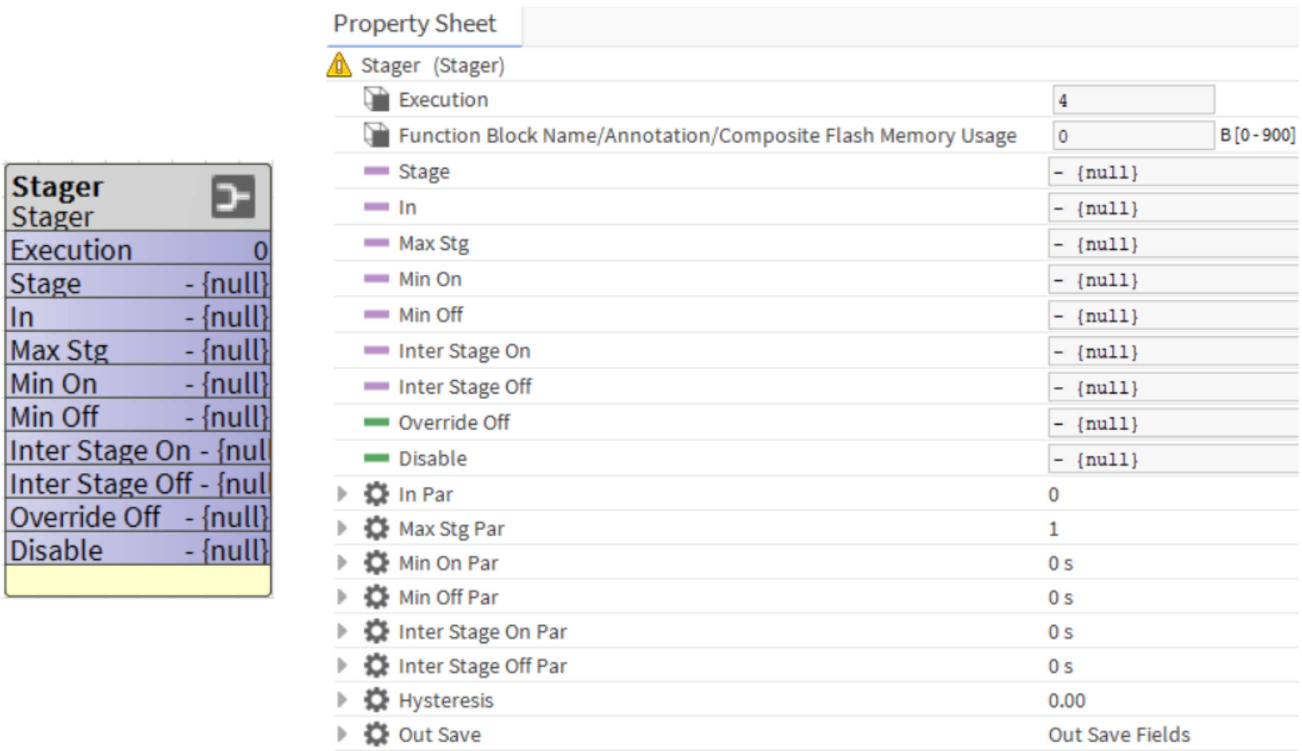


Fig. 158 Stager Function Block and Property Sheet

Input

Table 122 Inputs of Stager

Input Name	Description
In	Input load demand in percentage. **Numeric:** 32-Bit Floating value -100 - 100. **Default:** Null
Max Stg	Maximum stages available to be turned on. **Numeric:** 32-Bit Integer value 1 - 255. **Default:** Null
Min On	Minimum time a stage should be turned on in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null

Table 122 Inputs of Stager (Continued)

Input Name	Description
Min Off	Minimum time a stage should be turned off in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null
Inter Stage On	Minimum time before the next stage can be turned on in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null
Inter Stage Off	Minimum time before the next stage can be turned off in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null.
Override Off	Override to turn off stages. **Numeric:** Binary value 0, 1. **Default:** Null.
Disable	**Disable** function. **Numeric:** Binary value 0, 1. **Default:** Null.

Output

Table 123 Output of Stager

Output Name	Description
Stages	The number of stages active.

Parameter

Table 124 Parameters of Stager

Parameter Name	Description
In Par	Input load demand in percentage. **Numeric:** 32-Bit Floating value -100 - 100. **Default:** Null.
Max Stg Par	Maximum stages available to be turned on. **Numeric:** 32-Bit Integer value 1 - 255. **Default:** Null.
Min On Par	Minimum time a stage should be turned on in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null.
Min Off Par	Minimum time a stage should be turned off in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null.
Inter Stage On Par	Minimum time before the next stage can be turned on in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null.
Inter Stage Off Par	Minimum time before the next stage can be turned off in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null.

Table 124 Parameters of Stager (Continued)

Parameter Name	Description
Hysteresis	Stage off hysteresis. **Numeric:** 32-Bit Floating value 0 - 100. **Default:** 0
Out Save	• **Master Sync Enabled:** If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

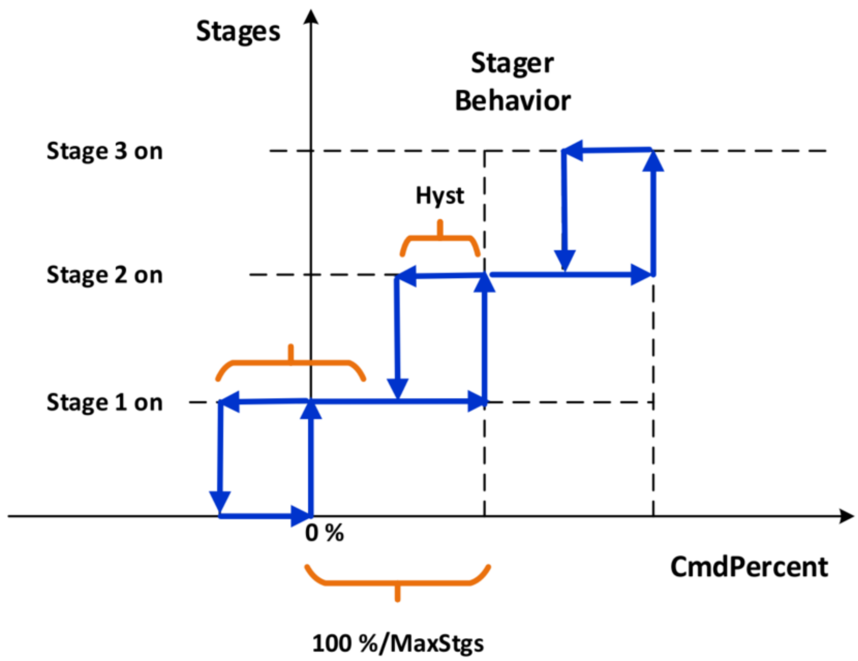


Fig. 159 Stager Behavior

Examples

If In > 0, stage-1 = On. If In < 0 (0 - Hysteresis), which is -11 stage-1 = Off.

The stage-1 is ON when input is greater than 0 and goes OFF when input is less than (0-Hysteresis), which is -11.

- Stage 1 On = 1 %
- Stage 1 Off = -11 %

Example 1: Stager (Stager_Fan)

Input is 0, Stage-1 is OFF Input is 0.

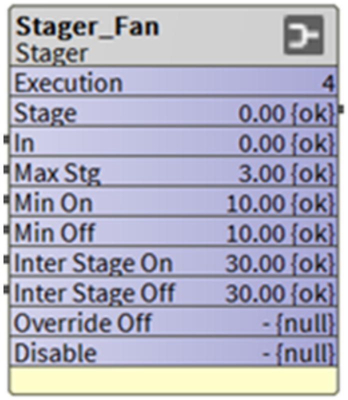


Fig. 160 Stager Function Block

Example 2: Stager (Stager_Fan)

In > 0, Stage-1 = On.

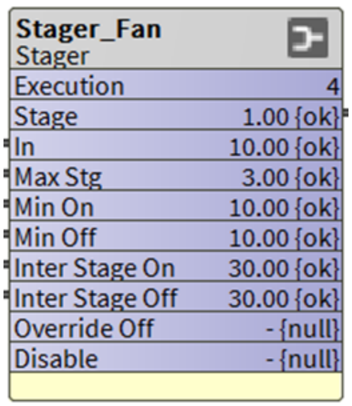


Fig. 161 Stager Function Block

Example 3: Stager (Stager_Fan)

In < 0, Stage-1 = On. In= below stage-1 hysteresis level. Stage-1 = OFF.

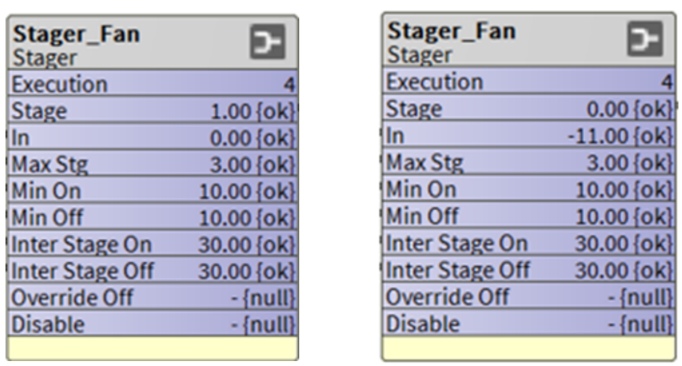


Fig. 162 Stager Function Block

Example 4: Stager (Stager_Fan)

Interstage ON delay, in rises from 10 to 100, stages 1, 2, and 3 = ON. The stager, on the other hand, turns on only stage-1 and waits for the InterStage ON delay, which is set to 30 seconds, before turning on stage-2 and then stage-3 (This lets the next stage turn ON until the set time after the previous stage is ON).

Stage-2 = ON, when interstage ON delay has expired and then stage-3.

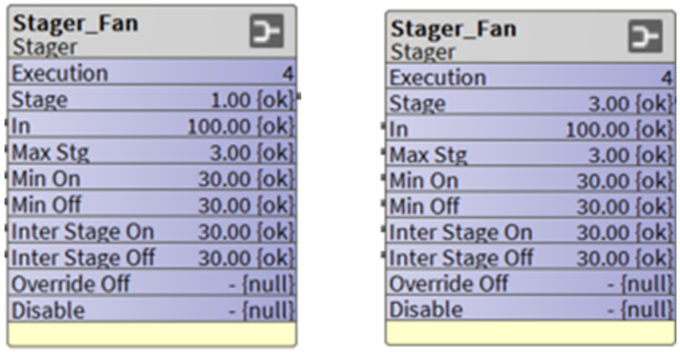


Fig. 163 Stager Function Block

Example 5: Stager (Stager_Fan)

Interstage ON delay, In = 100. If In suddenly increases to 10, stages 2 and 3 turn off. Stage-3 stager turns off immediately and waits for the InterStage OFF delay, which is set to 10 seconds, before turning off stage-2 (This lets the next stage turn OFF until the set time after the previous stage is OFF). Stage 2 = ON with In = 10.

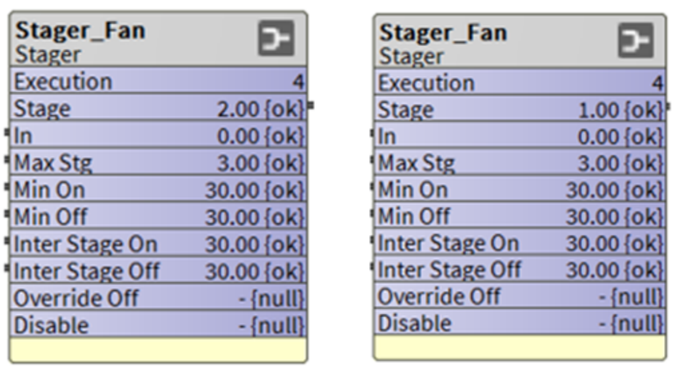


Fig. 164 Example Stager Function Block

Example 6: Stager (Stager_Fan)

Stage-2 = OFF with In=10, after interstage OFF delay.

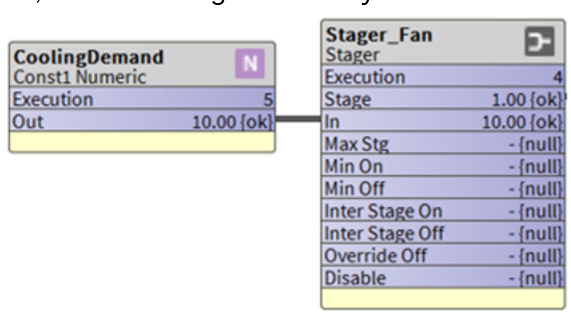


Fig. 165 Example Stager Function Block

Stage Driver

The **StageDriver** function block is used to control several devices such as pumps, Fans, boiler or chillers in an intelligent sequence. The function block can be configured with different strategies that allow the devices to be controlled in alternation or in the way that all devices will get a similar switch-on time. The function block determines which devices are switched on and off based on the active stages. If required, the function block will switch on several outputs at the same time. For example, if there are 5 pumps and 3 pumps are to run, then the function block determines which 3 of the 5 pumps will be switched on.

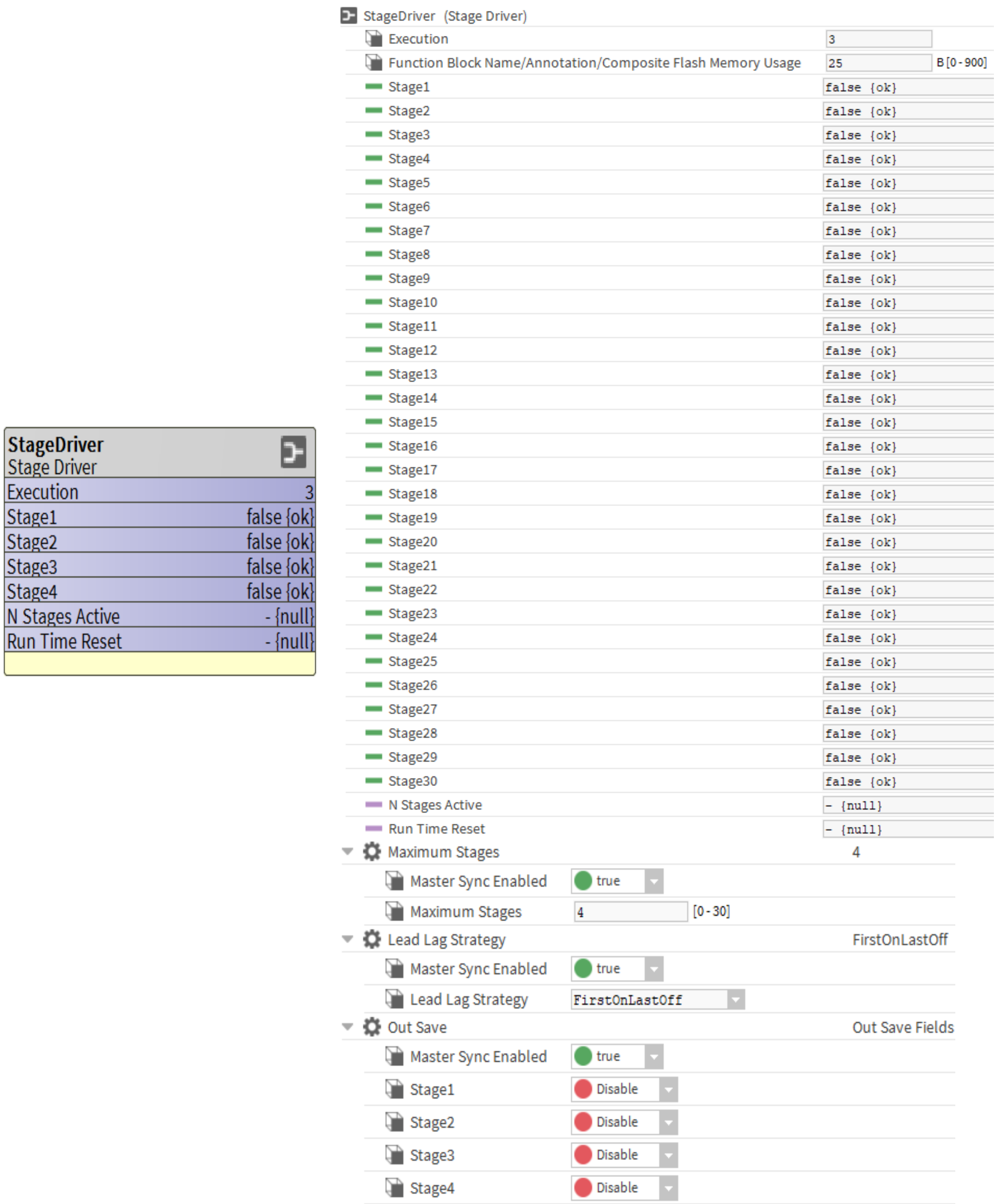


Fig. 166 StageDriver Function Block and Property Sheet

Input

Table 125 Inputs of StageDriver

Input Name	Description
N Stages Active	The input indicates how many individual outputs **Stage1** - **Stage30** are to be distributed to On/Off values. **N Stages Active** needs to be <= **Maximum Stages**. If, for example, N Stages Active = 5, then 5 outputs are also switched on; which ones these are, is decided by this function block. **Numeric:** 32-Bit integer; Default = Null = All Outputs are Off, 0 = All Outputs are Off, 1 - 30 to switch **Stage 1 – 30** according to the configured **Lead Lag Strategy**.
Run Time Reset	This input is only used if the configuration **Lead Lag Strategy** = Runtime Equalization for the lowest runtime. In the function block, the switch-on time in minutes is logged internally for each output as a floating point number. The maximum switch-on time is 16777216 min (31.92 years). The switch-on time is stored in a flash and reused when the controller is restarted. This input resets the runtime of one output. The output number to be reset is written here. For example, to reset the runtime of output **Stage 5**, the value 5 is entered here. As long as a value of 1 - 30 is present, the runtime remains at 0 min; It means to activate the timer, the input must be changed to 0 = No reset. **Numeric:** 32-Bit integer, Default = Null = No reset; 1 - 30 to reset the runtime of Output 1 - 30. 0 = No reset.

Output

Table 126 Outputs of StageDriver

Output Name	Description
Stage1	This output is typically connected via a Binary Output (relay, triac, ...) to the first device (pump 1 or fan 1). The output is turned on depending on the **Lead Lag Strategy**. By default, only Stage1 is shown on the wiresheet as output. More Stage outputs are visible by configuring **MaximumStages** to a higher value. **Binary:** Value 0=Off, 1=On
Stage2	This output is connected to the second device (pump 2 or fan 2). The slot is visible only if the parameter **MaximumStage** is >= 2.
Stage3 - Stage30	Same as the Stage2 description.

Parameter

Table 127 Parameters of StageDriver

Parameter Name	Description
Maximum Stages	This parameter configures the number of outputs **Stage 1** - **Stage30** = number of devices in the system (number of pumps or fans) controlled via this function block. By selecting a higher number than 1, more outputs are shown on the wiresheet. **Numeric:** 32-Bit integer; 0 - 30 while 0 = All outputs are OFF, Default = 1
Lead Lag Strategy	The parameter selects the Lead Lag Strategy, which determines how the outputs are switched. For more details, refer to the below information. 0 = First On - Last Off 1 = First On – First Off (Rotation) 2 = Runtime Equalization for lowest runtime. Next On Stage has the lowest runtime, and next Off has the highest runtime. With Runtime Equalization, when **N Stages Active** is changed, a decision is made as to which output is switched on or off depending on the runtime. If **N Stages Active** does not change, the outputs do not change either. **Numeric:** 32-Bit integer; 0..2, Default = 0 = First On Last Off
Out Save	**Stage1 - Stage30:** For each stage output, the **Out Save** feature can be Enabled or Disabled. Enabled means that the last value at the stage output **Stage1 - Stage30** is saved in case of a power loss and is set back to the saved value for one DDC cycle when the power returns.

Examples

0 on a white background means the stage is OFF and 1 on yellow background means the stage is ON.

First On - Last Off with Maximum Stages = 3

N Stages Active	Stage 1	Stage 2	Stage 3
0	0	0	0
1	1	0	0
2	1	1	0
3	1	1	1
2	1	1	0
1	1	0	0
0	1	0	0
Null	0	0	0

Fig. 167 First On - Last Off with Maximum Stages = 3

0 on a white background means the stage is OFF and 1 on yellow background means the stage is ON.

First On – First Off with Maximum Stages = 4

N Stages Active	Stage 1	Stage 2	Stage 3	Stage 4
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
2	0	1	1	0
1	0	0	1	0
2	0	0	1	1
3	1	0	1	1
Null	0	0	0	0

Fig. 168 First On – First Off with Maximum Stages = 4

0 on a white background means the stage is OFF and 1 on yellow background means the stage is ON.

Runtime Equalization with Maximum Stages = 4

Time	N Stages Active	Stage 1 Runtime	Stage 2 Runtime	Stage 3 Runtime	Stage 4 Runtime
0min Start a timer after full teach	4	1 0min	1 0min	1 0min	1 0min
2min (120sec)	3	1 2min	1 2min	1 2min	0 2min
4min (240sec)	2	1 4min	1 4min	0 4min	0 2min
6min (360sec)	3	1 6min	1 6min	0 4min	1 2min
8min (480sec) RunTimeReset =1, then 0	3	1 0min	1 0min	0 0min	1 0min
10min (600sec)	2	1 2min	0 2min	0 0min	1 2min
12min (720sec)	1	1 4min	0 2min	0 0min	0 4min
14min (840sec)	0	0 6min	0 2min	0 0min	0 4min
16min (960sec)	1	0 6min	0 2min	1 0min	0 4min
18min (1080sec)	2	1 6min	0 2min	1 2min	0 4min

Fig. 169 Runtime Equalization with Maximum Stages = 4

Flow Control

The Flow Control function is a Damper Flow Controller with Variable Air Volume (VAV). This is typically the second half of a pressure independent VAV box cascade control strategy, with the input typically coming from the output of a PID block that controls space temperature.

The flow control function determines an effective flow control set point (effFlowSetPt) and generates a 0 - 100 percent command to control a VAV box damper. The effective flow setpoint is mapped to the commanded flow setpoint (in percent) from a temperature control signal, with 0 % mapping to the min flow setpoint and 100 % mapping to the max flow setpoint.

If the sensedFlowVol input is invalid (the sensor fails), the damper will be driven in a "pressure dependant" mode, with the output equal to 100 percent *(minSP/maxSP) + (1-minSP/maxSP)*cmd%. The output = 20 % + .8*cmdPercent if either flow MinSP or MaxSP is invalid..

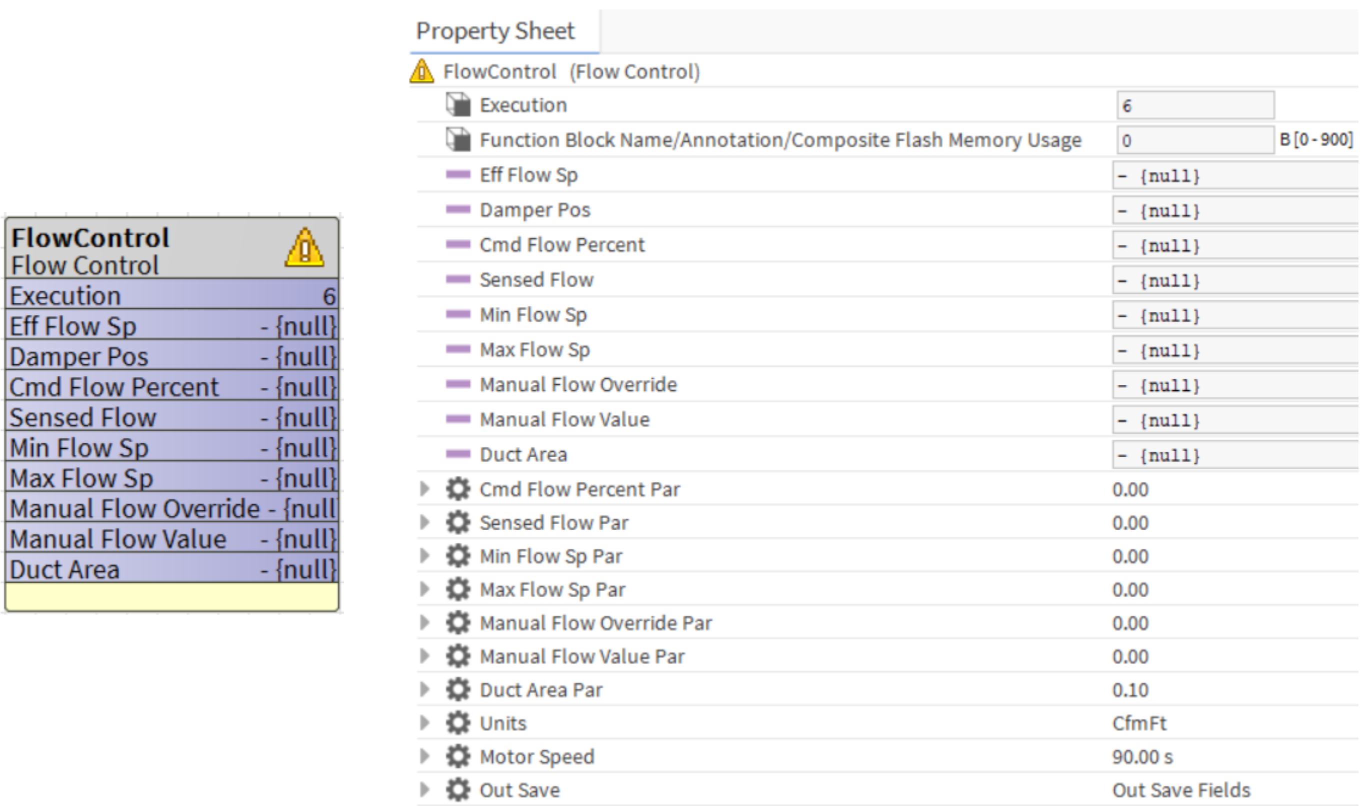


Fig. 170 FlowControl Function Block and Property Sheet

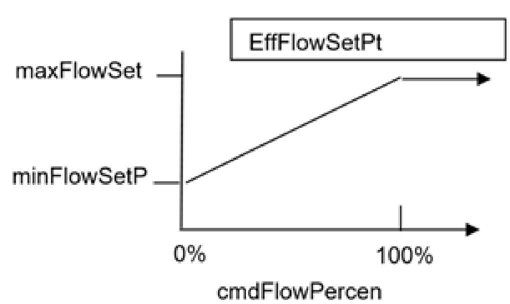


Fig. 171 Flow Control Behaviour

Input

Table 128 Inputs of FlowControl

Input Name	Description
Cmd Flow Percent	Command flow percent. **Numeric:** 32 Bit Floating value, 0 - 100, **Default:** Null
Sensed Flow	Sensed flow. **Numeric:** 32 Bit Floating value, >=-inf-<inf, **Default:** Null
Min Flow Sp	Minimum flow set point. **Numeric:** 32 Bit Floating value, >=-inf-<inf, **Default:** Null
Max Flow Sp	Maximum flow set point. **Numeric:** 32 Bit Floating value, >=-inf-<inf, **Default:** Null
Manual Flow Override	The manual Flow Override input allows the flow setpoint to be selectively overridden based on the following codes: **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 **Numeric:** 32 Bit Floating value, >=-inf-<inf, **Default:** Null
Manual Flow Value	Manual flow value. **Numeric:** 32 Bit Floating value, 0-<inf, **Default:** Null
Duct Area	If **Units** = 0 then **duct area** is in ft2. If **Units** = 1, 2 then the **duct area** is in m2. **Numeric:** 32 Bit Floating value, 0-<inf, **Default:** Null

Output

Table 129 Outputs of FlowControl

Output Name	Description
Eff Flow Sp	Effective air flow setpoint. **Numeric:** 32 Bit Floating value, 0 - inf
Damper Pos	Damper position in percent. **Numeric**: 32 Bit Floating value, 0 - 100

Parameter

Table 130 Parameters of FlowControl

Parameter Name	Description
Cmd Flow Percent (In & Par)	Command flow percent. **Numeric:** 32 Bit Floating value, 0 - 100, **Default:** 0
Sensed Flow (In & Par)	Sensed flow. **Numeric:** 32 Bit Floating value, >=-inf-<inf, **Default:** 0

Table 130 Parameters of FlowControl (Continued)

Parameter Name	Description
Min Flow Sp (In & Par)	Minimum flow setpoint. **Numeric:** 32 Bit Floating value, >=-inf-<inf, **Default**: 0
Max Flow Sp (In & Par)	Maximum flow setpoint. **Numeric:** 32 Bit Floating value, >=-inf-<inf, **Default:** 0
Manual Flow Override (In & Par)	Manual flow override. **0** or any other value not listed = No override **2:** ManFlowValue will be set to EffFlowSp **6:** MinFlowSp will be set to EffFlowSp **7:** MaxFlowSp will be set to EffFlowSp **Numeric:** 32 Bit Floating value, >=-inf-<inf, **Default:** 0
Manual Flow Value (In & Par)	Manual flow value. **Numeric:** 32 Bit Floating value, 0-<inf, **Default:** 0
Duct Area (In & Par)	DuctArea. **Numeric:** 32 Bit Floating value, 0-<inf, **Default:** 0.1
Units	Units **0:** cfm(flow) and ft2 (area), **1:** L/s(flow) and m2(area), **2:** m3/hr(flow) and m2(area) **Numeric:** 32 Bit Integer value, 0-2, **Default:** 0
Motor Speed	Motor Speed measued in seconds. **Numeric:** 32 Bit Floating value, 1 - 255, **Default:** 90 Sec
Out Save	• **Master Sync Enabled:** If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Eff Flow Sp:** To enable or disable the **Effective Flow Setpoint** feature. • **Damper Pos:** To enable or disable the **Damper Position** feature.

Examples

Example 1: FlowControl (No manual override)

When there is no manual override.

The minimum flow setpoint is 100 CFM, the maximum flow setpoint is 200 CFM, and the Duct Area is 0.5 sq.ft., so the effective flow setpoint is 121 CFM when the temperature control output is 21 and fed as Command flow percent. The damper will be set to a position determined by the flow feedback sensed.

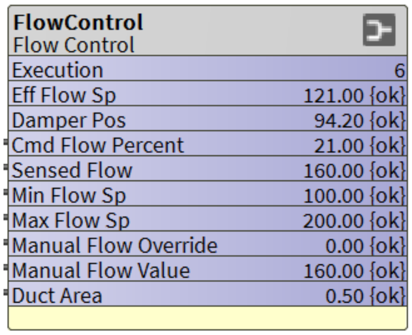


Fig. 172 No manual override

Example 2: FlowControl (Manual override)

When there is a manual override value set to 2 = ManFlowValue input.

The minimum flow setpoint is 100 CFM when the temperature control output is 21 and fed as Command flow percent. With a ManualFlowValue of 160 CFM, the maximum flow setpoint is 200 CFM.

- ManFlowValue input with ManualFlowOverride set to 2.
- The effective flow setpoint will be 160 CFM, and the damper position will modulate accordingly.
- The duct area is 0.5 sq.ft., the effective flow setpoint will be 160 CFM, and the damper position will modulate accordingly.

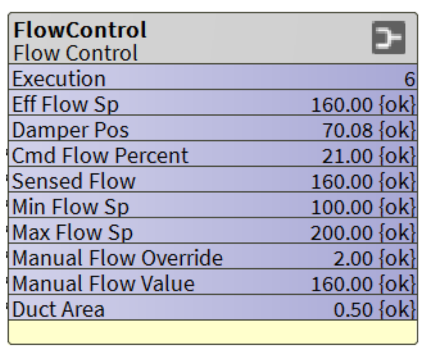


Fig. 173 Manual override

Example 3: FlowControl (Manual override)

When there is a manual override value set to 6 = MinFlowSP

When the manual override is set to 6, the effective flow setpoint will be 100 CFM, and the damper position will modulate accordingly.

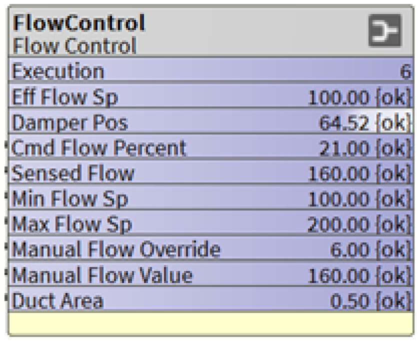


Fig. 174 Manual override

Example 4: FlowControl (Manual override)

When there is a manual override value set to 7 = MaxFlowSP

When the manual override is set to 7, the effective flow setpoint will be 160 CFM, and the damper position will modulate accordingly.

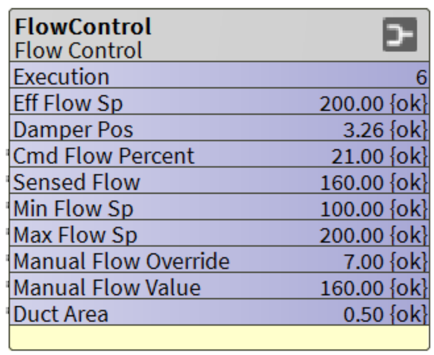


Fig. 175 Manual override

Cycler

The Cycler function is a generic stage driver or a Thermostat Stage Cycler dependant on the value of the CPH parameter (CPH = 0 means stager functionality, and CPH = 1 - 60 gives thermostat cycler functionality).

The Cycler function is a Honeywell thermostat's traditional anticipator cycling algorithm.

P or PI space temperature error in percent is the input (0 - 100). The following are the standard (recommended) settings:

- CPH = 3 for cooling,
- CPH = 6 for heating,
- anticAuth = 100 %,
- hyst = 100 %/maxstages/1.2.

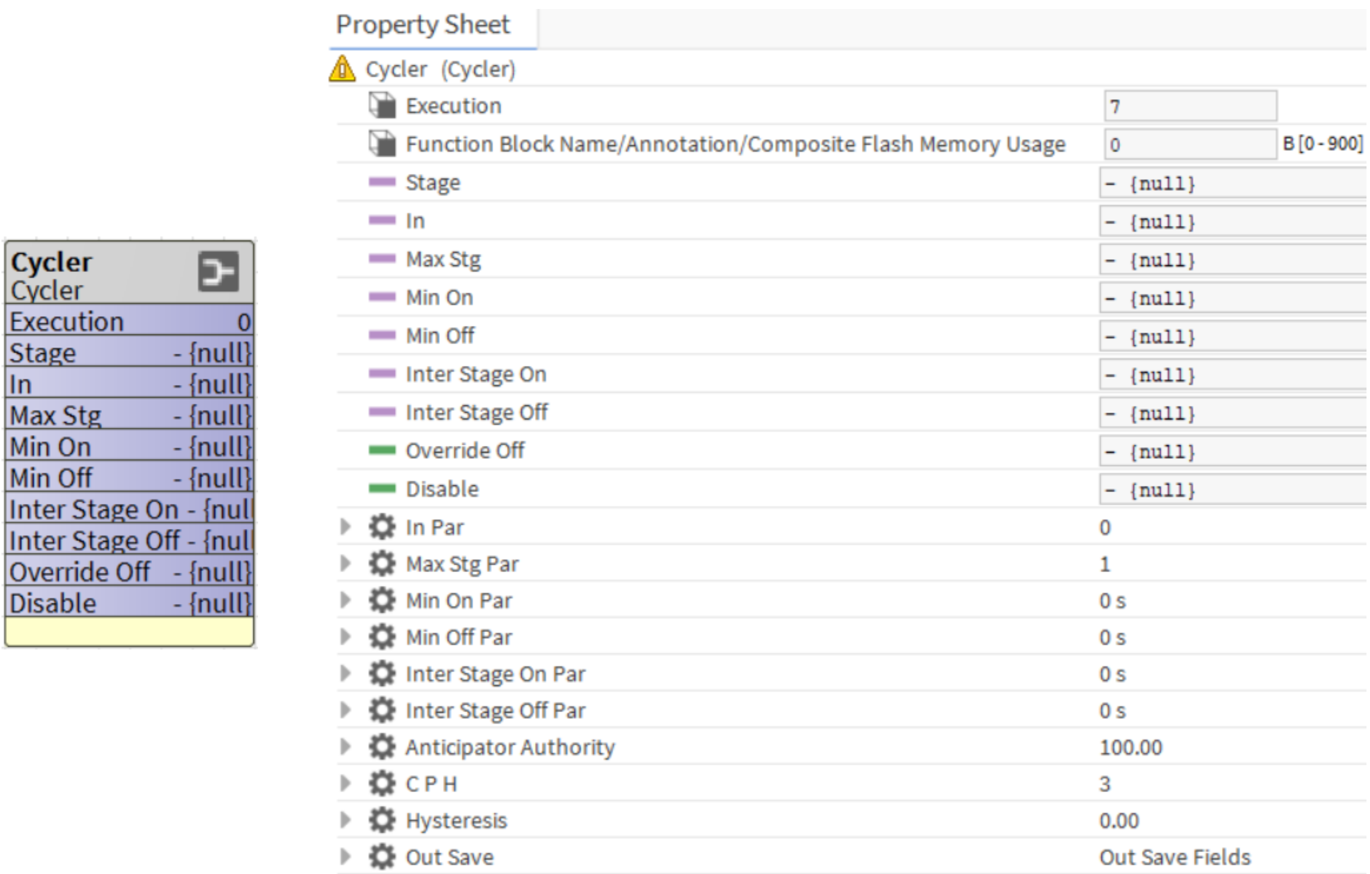


Fig. 176 Cycler Function Block and Property Sheet

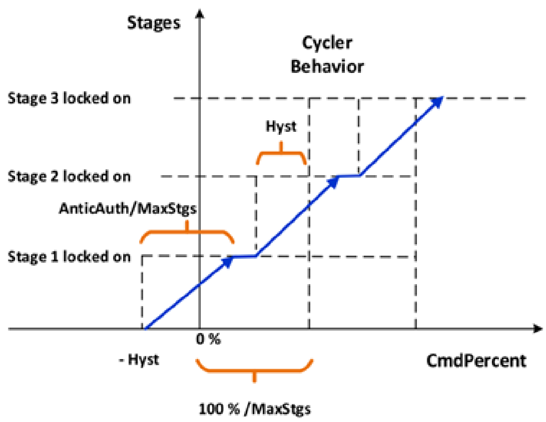


Fig. 177 Cycler Behavior Diagram

Input

Table 131 Inputs of Cycler

Input Name	Description
In	Input load demand in percentage. **Numeric:** 32-Bit Floating value -100 - 100. **Default:** Null.
Max Stg	Maximum stages available to be turned on. **Numeric:** 32-Bit Integer value 1 - 25. **Default:** Null.
Min On	Minimum time a stage should be turned on in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null.
Min Off	Minimum time a stage should be turned off in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null.
Inter Stage On	Minimum time before the next stage can be turned on in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null.
Inter Stage Off	Minimum time before the next stage can be turned off in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null.
Override Off	Override to turn off stages. **Numeric:** Binary value 0, 1. **Default:** Null.
Disable	Disable function. **Numeric:** Binary value 0, 1. **Default**: Null.

Output

Table 132 Output of Cycler

Output Name	Description
Stages	The number of stages active.

Parameter

Table 133 Parameters of Cycler

Parameter Name	Description
In Par	Input load demand in percentage. **Numeric:** 32-Bit Floating value -100 - 100. **Default:** Null
Max Stg Par	Maximum stages available to be turned on. **Numeric:** 32-Bit Integer value 1 - 25. **Default:** Null
Min On Par	Minimum time a stage should be turned on in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null
Min Off Par	Minimum time a stage should be turned off in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null
Inter Stage On Par	Minimum time before the next stage can be turned on in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null
Inter Stage Off Par	Minimum time before the next stage can be turned off in Sec. **Numeric:** 32-Bit Integer value 0 - 64799. **Default:** Null
Anticipator Authority	Helps in adjusting cycler behavior in percentage. **Numeric:** 32-Bit Floating value 0-200. **Default:** 100
CPH	Cycles per hour. **CPH** (non-zero) is the max cycle rate in **CPH** when input is halfway between the stages available, and AnticAuth is at the default value (100 %). **CPH** = 0 means the Stager logic is performed and has no other effect. **Numeric:** 32-Bit Integer value 0 - 60. **Default:** 3
Hysteresis	Stage off hysteresis. **Numeric:** 32-Bit Floating value 0 - 100. **Default:** 0
Out Save	• **Master Sync Enabled:** If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Stage:** To enable or disable the **Stage** feature.

Examples

The below figure explains how the stages become ON when MaxStg = 1, interstage delay On or Off = 10 seconds, **Hysteresis** = 10, AnticAuth = 100, CPH = 0. CPH = 0 means the Stager logic is performed.

Example 1: Cycler (1-Stage On)

The below figure explains how the single-stage becomes ON when MaxStg = 1, interstage delay On or Off = 5 sec, Hysteresis = 5, AnticAuth = 100, CPH = 3 that is Cooling.

- Stage 1 On = 25 %, which is (AnticAuth/MaxStgs) - Hysteresis.
  CmdPercent > (N –1) * 100 % / MaxStages
- For turning off stage N the criterion is:
  CmdPercent < (N –1) * 100 % / MaxStages – Hysteresis

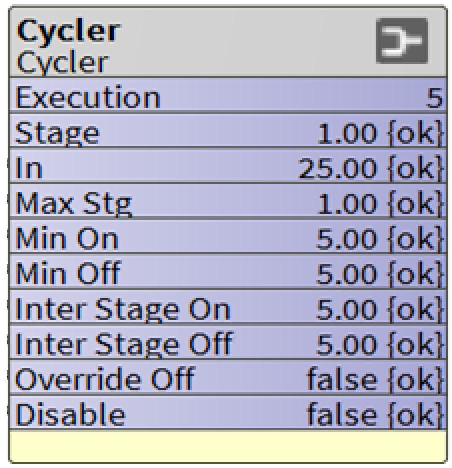


Fig. 178 1-Stage On

The below figure explains how the single stage becomes OFF when MaxStg = 1, interstage delay On or Off = 5 sec, Hysteresis = 5, AnticAuth = 100, CPH = 3 that is Cooling.

Example 2: Cycler (1-Stage Off)

Stage 1 Off = 15 %, which is (AnticAuth/MaxStgs) - Hysteresis.

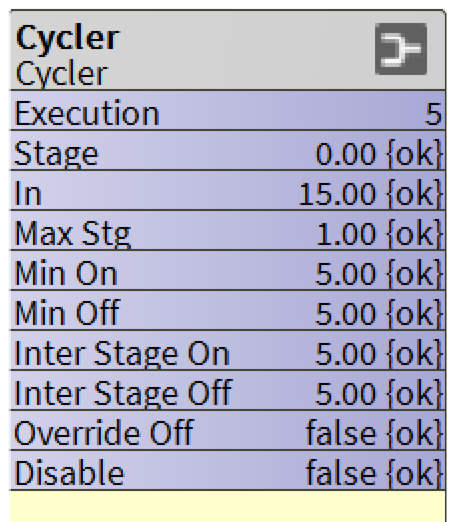


Fig. 179 1-Stage Off

Example 3: Cycler (2-Stage On)

The below figure explains how the 2 stage becomes ON when MaxStg = 2, interstage delay On/Off = 5 sec, Hysteresis = 5, AnticAuth = 100, CPH = 3 that is Cooling.

- Stage 1 On = 30 %, which is (AnticAuth/MaxStgs) - Hysteresis.

- Stage 2 On = 35 % which is (AnticAuth/MaxStgs) - Hysteresis.

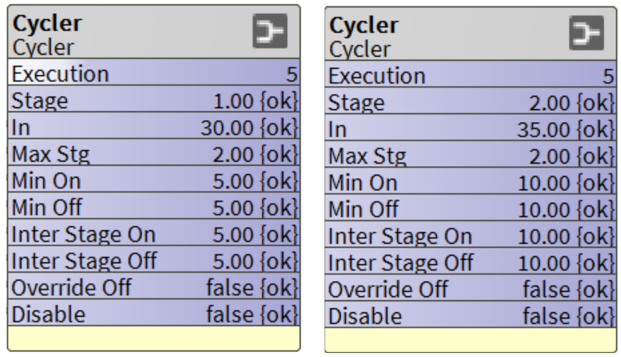


Fig. 180 2-Stage On

Example 4: Cycler (2-Stage Off)

The below figure explains how the dual stages becomes OFF when MaxStg = 2, interstage delay On/Off = 5 sec, Hysteresis = 5, AnticAuth = 100, CPH = 3 that is Cooling.

- Stage 2 Off = 32 %, which is (AnticAuth/MaxStgs) - Hysteresis.
- Stage 1 Off = 30 %, which is (AnticAuth/MaxStgs) - Hysteresis.

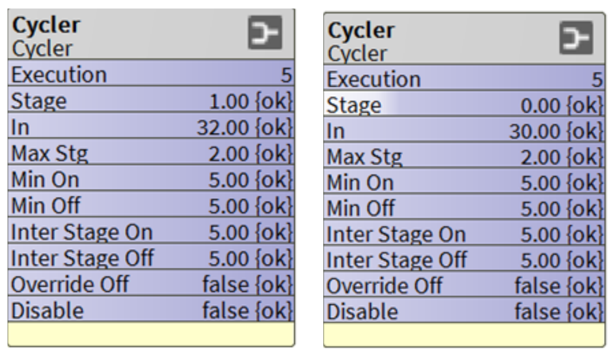


Fig. 181 2-Stage OFF

This behavior can be visualized with two stages. For example below:

If the load slowly increases, the first stage starts with a low cycle rate and low average on-time, and as the load increases, the cycle rate increases to the max at 50 % average on-time, and decreases again as the average on-time approaches 100 % until the first stage is now locked on. The second stage then follows the same pattern.

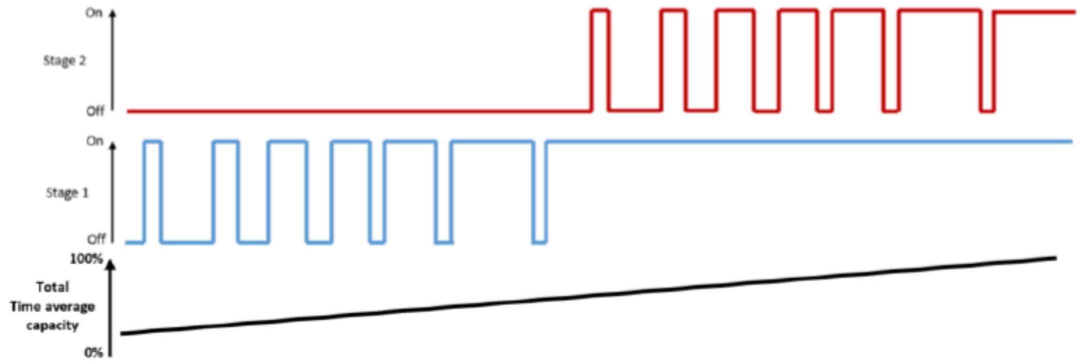


Fig. 182 2-Stage

Example 5: Cycler (4-Stage Override Off)

The below figure explains how the four stages are overridden by the Override Off Pushbutton.

When MaxStg = 4, interstage delay On and Off = 3 sec, MinOn and MinOff = 5, Hysteresis = 10, AnticAuth = 100, CPH = 3 that is Cooling, OverrideOff = “true”.

When override is “true,” active stages are shed (turned off) based on MinOn and interstage timers regardless of the CmdPercent input. Output is the number of stages active (0 - MaxStgs) that can send to the StageDriver function block.

- Stage 1,2,3, and 4 are OFF = 0 %, which is (AnticAuth and MaxStgs) - Hysteresis.

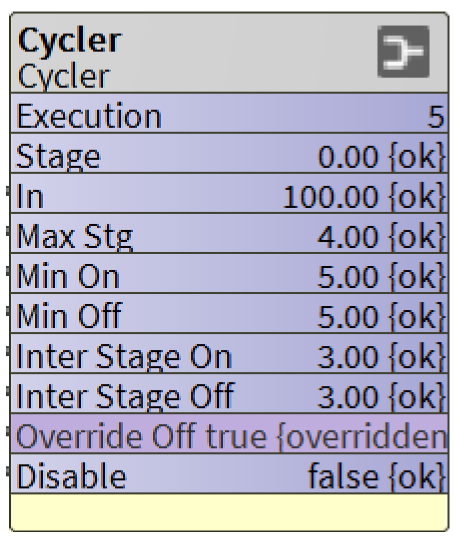


Fig. 183 4-Stages Override Off

Example 6: Cycler (4-Stage)

The below figure explains how the 4 stages are increasing based on the command percent.

When MaxStg = 4, interstage delay On or Off = 3 sec, MinOn or MinOff = 2 sec, Hysteresis = 10, AnticAuth = 100, CPH = 60, OverrideOff = “false”.

- Stage 1, 2, 3, and 4 are ON = 100 %, which is (AnticAuth or MaxStgs) - Hysteresis.

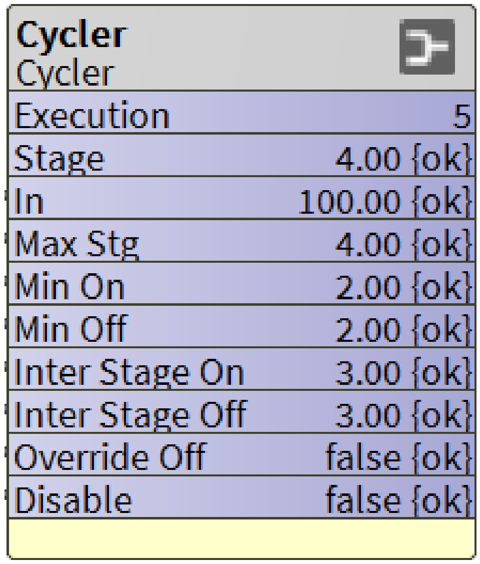


Fig. 184 4-Stage

CHAPTER 7

CONVERSION FUNCTION BLOCK

The following Conversion function block is available in the honIrmControl Palette that can be configured and used to create the required application logic:

- Binary To Numeric

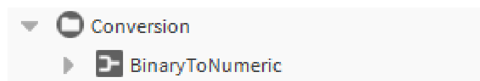


Fig. 185 Conversion Function Block

Binary To Numeric

The Binary To Numeric function block is used to assign three different numeric values to a numeric output based on a binary input that can be "true," "false," or "null."

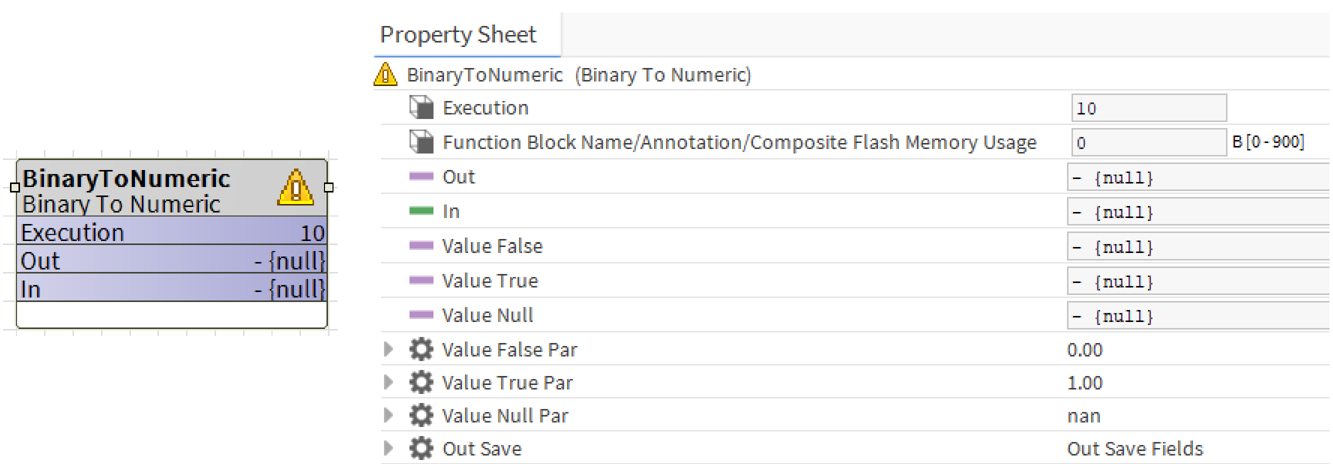


Fig. 186 BinaryToNumeric Function Block and Property Sheet

Depending on the boolean input **In**, the output **Out** is assigned to one of the three numeric input or parameter values **Value False**, **Value True**, or **Value Null Par**.

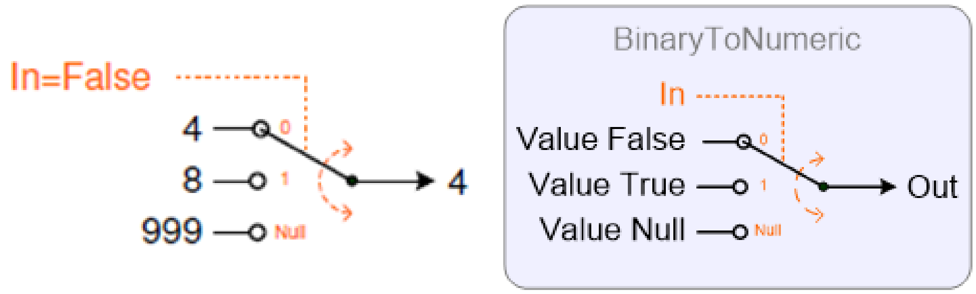


Fig. 187 BinaryToNumeric Logic Diagram

Input

Table 134 Inputs of Binary To Numeric

Input Name	Description
In	This binary input determines the value of the output **Out**. • If **In** is set to “false,” the value of the input/parameter **Value False** is output **Out**. • If **In** is set to “true,” the value of the input/parameter **Value True** is output **Out**. • If **In** is set to “null,” the value of the input/parameter **Value Null** is output **Out**. • **Valid values:** “false,” “true,” “null.”
Value False (In & Par)	This value is assigned to the output **Out** if the input **In** is “false”.
Value True (In & Par)	This value is assigned to the output **Out** if the input **In** is “true”.
Value Null (In & Par)	This value is assigned to the output **Out** if the input **In** is “null”.

Output

Table 135 Outputs of Binary To Numeric

Output Name	Description
Out	• If **In** is set to “false,” the value of the input or parameter **Value False** is output **Out**. • If **In** is set to “true,” the value of the input or parameter **Value True** is output **Out**. • If **In** is set to “null,” the value of the input or parameter **Value Null** is output **Out**.

Parameter

Table 136 Parameters of Binary To Numeric

Parameter Name	Description
Value False Par (In & Par)	If **Value False** is “null,” **Value False Par** is used as a parameter (default = 0).
Value True Par (In & Par)	If **Value True** is “null,” **Value True Par** is used as a parameter (default = 1).
Value Null Par (In & Par)	If **Value Null** is “null,” **Value Null Par** is used as a parameter (default = null).
Out Save	• **Master Sync Enabled:** If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Note: *For Conversion from a numeric to a binary value, please select the Switch – NumericSelect functional block.*

Example 1: BinaryToNumeric

An occupancy sensor is to be sent to BACnet with the values 1 = NotUsed, 2 = Unoccupied, 3 = Occupied. The occupancy sensor is a binary value with the values null = Not connected, 0 = Unoccupied, 1 = Occupied.

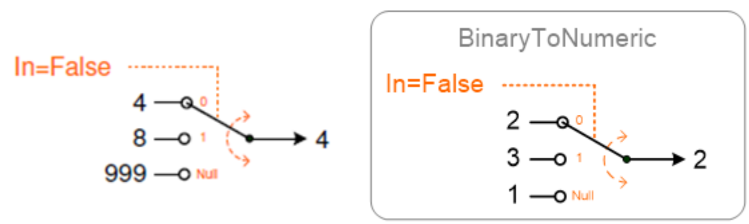


Fig. 188 BinaryToNumeric Example Logic Diagram

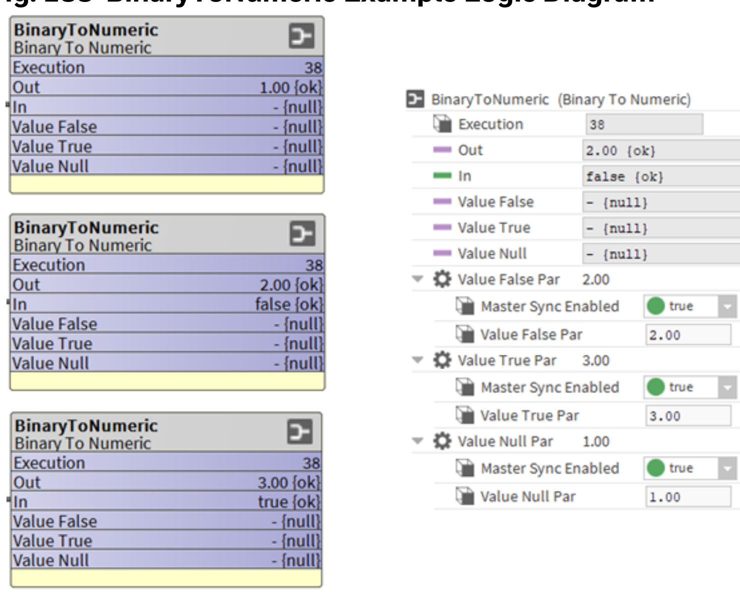


Fig. 189 BinaryToNumeric Property Sheet

CHAPTER 8 COMPARISON FUNCTION BLOCKS

The following Bit function blocks are available in the honIrmControl Palette that can be configured and used to build an application that performs compare operations between two inputs:

- Compare
- Equal Null
- Greater Than
- Greater Than Equal
- Less Than
- Less Than Equal

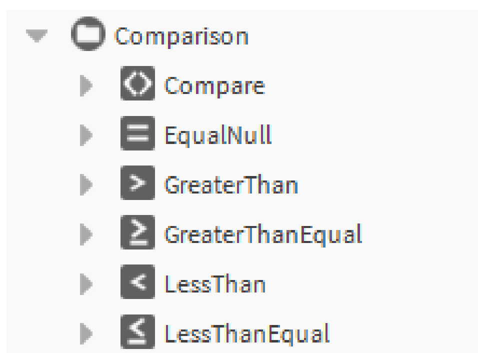


Fig. 190 Comparison Function Blocks

Compare

The compare function block compares two inputs between In A, In B, Hyst On, Hyst Off, and Start Up value.

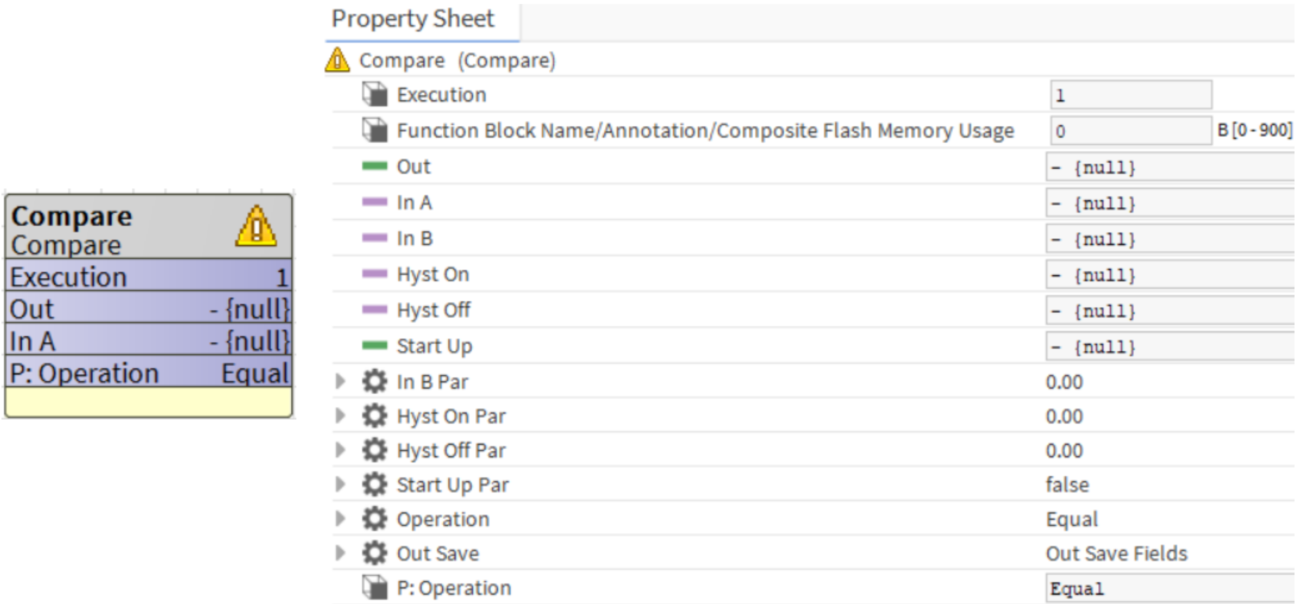


Fig. 191 Compare Function Block and Property Sheet

If no inputs are connected or "null," the output returns to "false."

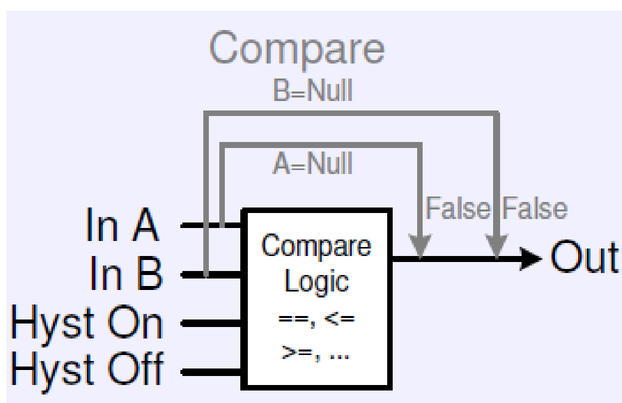


Fig. 192 Compare Logic Diagram 1

The following comparison calculations can be made using the Compare function block:

- **In A** less than **In B**
- **In A** greater than **In B**
- **In A** equal to **In B**

Input

Table 137 Inputs of Compare

Input Name	Description
In A	If **In A** = **Nul**l, then **Out** = **False**. But if **In A** = **Null** and **In B** = **Null** and **Operation** = **Equal**, then **Out** = **True**. These are 32-bit floating point input slot.
In B (In & Par)	If **In B** = **Null**, then **Out** = **False.** But if **In A** = **Null** and **In B** = **Null** and **Operation** = **Equal**, then **Out** = **True**. These are 32-bit floating point input slot.
Hyst On (In & Par)	This is a hysteresis value related to **In B**. See figure Compare Schematic.
Hyst Off (In & Par)	This is a hysteresis value related to value **In B**. See figure Compare Schematic.
Start Up (In & Par)	If the value lies within the hysteresis at the first call after the start-up, value is used as the result of the hysteresis.

Output

Table 138 Outputs of Compare

Output Name	Description
Out	Comparison of inputs. Result is “false” or “true.”

Parameter

Table 139 Parameters of Compare

Parameter Name	Description
In B par (In & Par)	If **In B** is “null,” **In B Par** is used as a parameter.
Hyst On Par (In & Par)	This is an (OR: the) hysteresis analog input that is used to perform compare calculations.
Hyst Off Par (In & Par)	This is an (OR: the) hysteresis analog input that is used to perform compare calculations.
Start Up Par (In & Par)	If **Start Up** is “null,” **Start Up Par** is used as a parameter.
Operation	There are four operations: • Equal • Less Than • Greater Than • Between

Table 139 Parameters of Compare (Continued)

Parameter Name	Description
Out Save	• **Master Sync Enabled:** If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

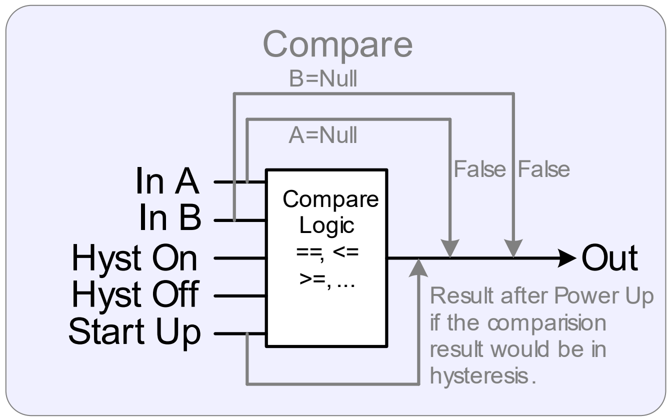


Fig. 193 Compare Schematic

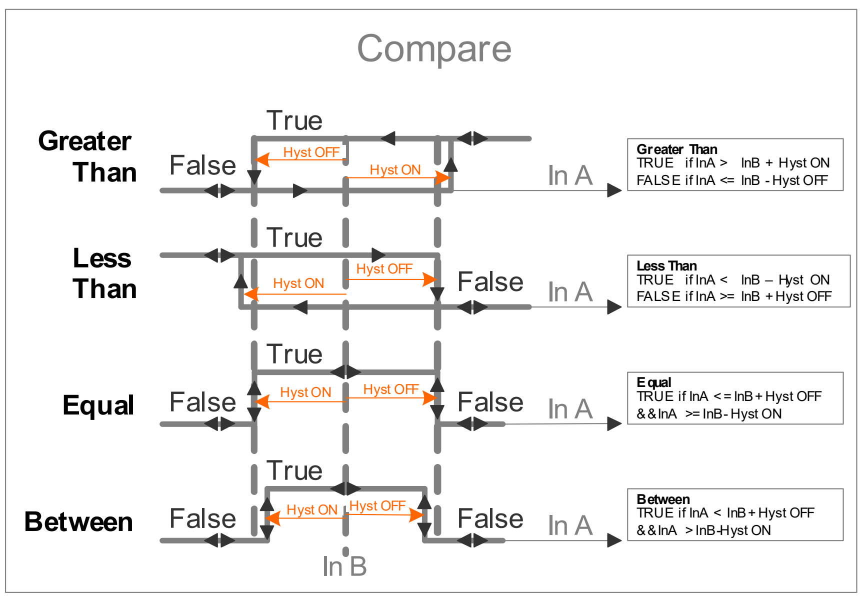


Fig. 194 Compare Parameters

Equal

If (In B – On Hyst) <= In A <= (Input B + Off Hyst), output is set to “true.”

Less Than

- If In A < (In B – on Hyst), the output is set to “true.”
- If (In B – on Hyst) <= input 1 less than (In B +off Hyst), output does not change.

- If In A >= (In B + off Hyst), the output is set to “false.”

Greater Than

- If In A > (input 2 + on Hyst), the output is set to “true.”
- If (In B – off Hyst) < In A <= (In B + on Hyst), the output does not change.
- If In A <= (In B - off Hyst), the output is set to “false.”

Between

The output is set to true, if (In B – On Hyst) < In A < (Input B + Off Hyst).

Examples

Example 1: Brightness with Greater Than

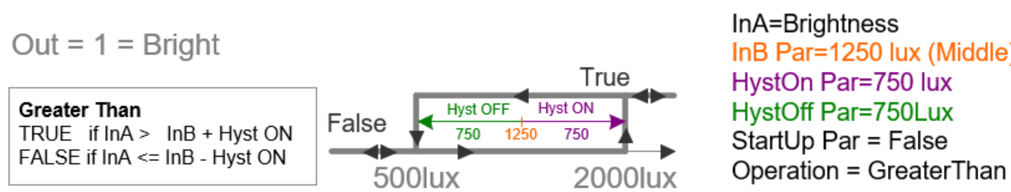


Fig. 195 Example of Brightness with Greater Than

Example 1: Compare (Operation set to Equal)

If **In A** = **In B,** the output is set to “True. On or Off Hystersis may be defined.

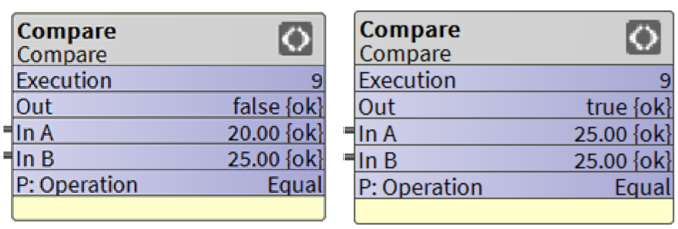


Fig. 196 Example of Equal Function

Example 2: Compare (Operation set to Greater Than)

If **In A** >**In B,** the output is set to “True. On or Off Hysteresis may be defined.

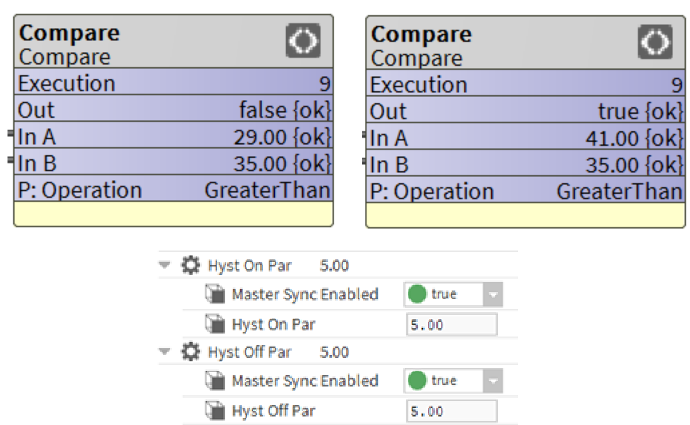


Fig. 197 Greater Than Function (+ hysteresis)

Example 3: Compare (Operation set to Less Than)

If **In A** < **In B**, the output is set to “True. On or Off Hysteresis may be defined.

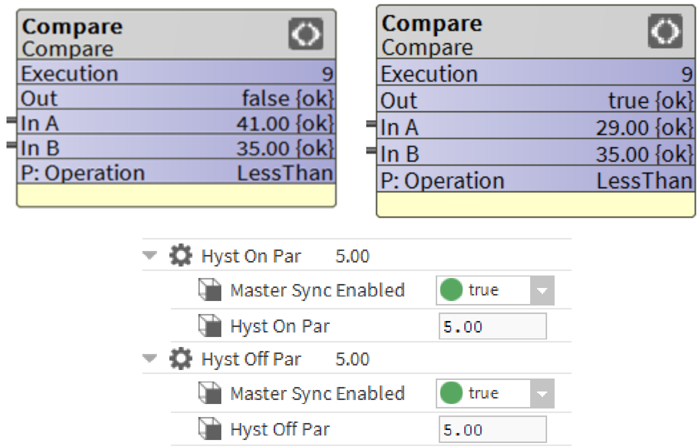


Fig. 198 Less Than Function block (+ hysteresis)

Example 4: Compare (Operation set to Between)

If In A is between In B+ Hysteresis & In B– Hysteresis, then the output is “True.” otherwise “False.”

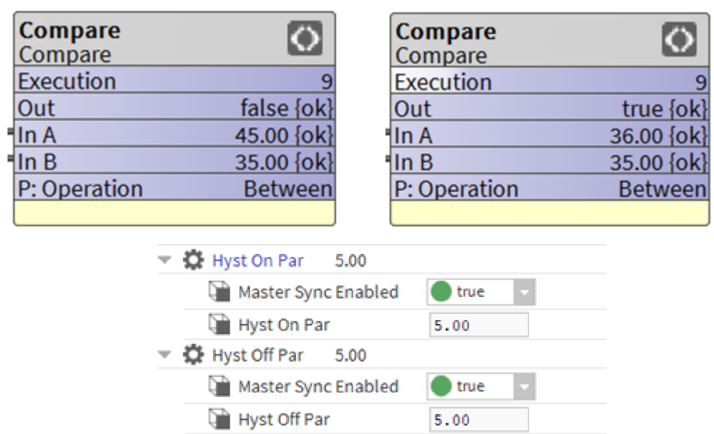


Fig. 199 Between Function block (+ hysteresis)

Equal Null

The Equal Null function block detects if the output is null.

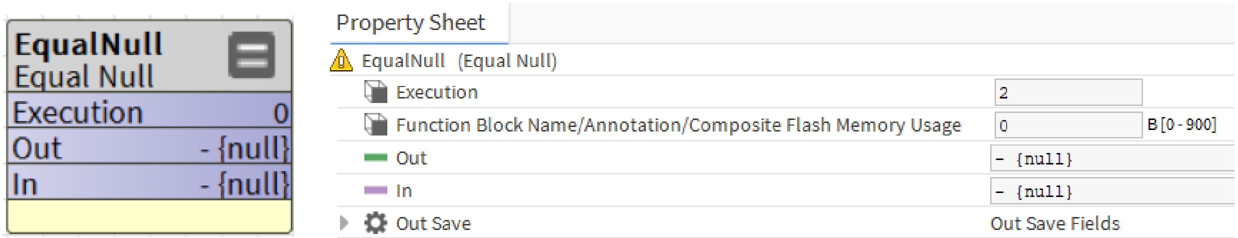


Fig. 200 Equal Null Function Block and Property Sheet

Output comes null when In is null.

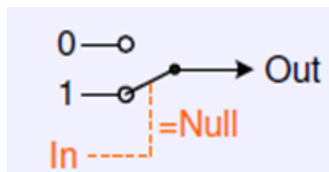


Fig. 201 EqualNull Logic Diagram

Input

Table 140 Input of EqualNull

Input Name	Description
In	This is 32-bit floating point.

Output

Table 141 Output of EqualNull

Output Name	Description
Out	**Output:** “null”.

Parameter

Table 142 Parameter of EqualNull

Parameter Name	Description
Out Save	• **Master Sync Enabled:** If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Example 1: Equal null

It can be used, for example, as a frost protection level while 0 - 100 % is the level and Null means that there is no frost condition. It can be used to indicate that the current sensor value is not valid due to a broken sensor or missing communication over BACnet.

If the Input is equal to “NULL,” Output is set to “true.”

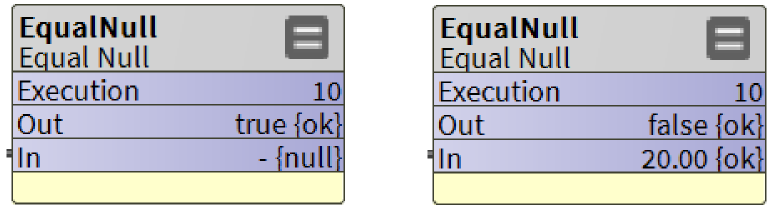


Fig. 202 Equal null Function Block

Greater Than

Greater Than function compares two inputs with each other. The output is set to “true,” if **In A > In B**.

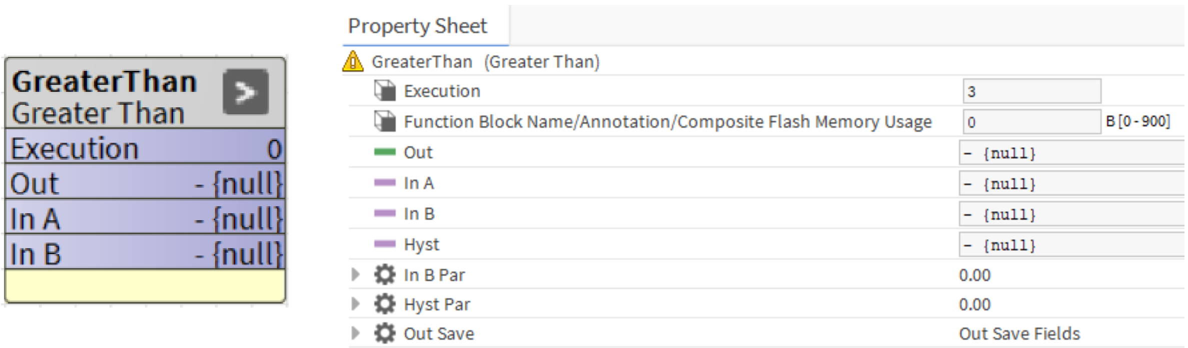


Fig. 203 Greater Than Function Block and Property Sheet

The output is set to “true”, if **In A > In B.**

The output does not change or “false”, if **In A** < (**In B** – **Hyst**).

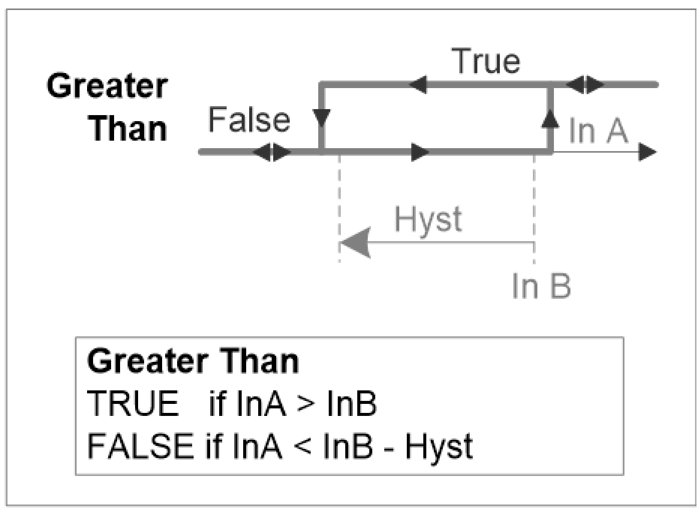


Fig. 204 Greater Than Logic Diagram

Input

Table 143 Inputs of Greater Than

Input Name	Description
In A	This value is compared against **In B**. These are 32-bit floating point input slots.
In B (In & Par)	This value is compared against **In A**. These are 32-bit floating point input slots.
Hyst (In & Par)	This hysteresis value is defined to avoid cycling output.

Output

Table 144 Ouput of Greater Than

Output Name	Description
Out	If **In A** > **In B**, **Out** is “true.” If **In A** < (**In B** – **Hyst**), **Out** is “false.”

Parameter

Table 145 Parameters of Greater Than

Parameter Name	Description
In B Par (In & Par)	If **In B** is “null,” then **In B Par** is used.
Hyst Par (In & Par)	If **Hyst** is “null,” then **Hyst Par** is used.
Out Save	• **Master Sync Enabled:** If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Example 1: GreaterThan

If **In A** > **In B**, **Out** is set to “true.” Hysteresis may be defined.

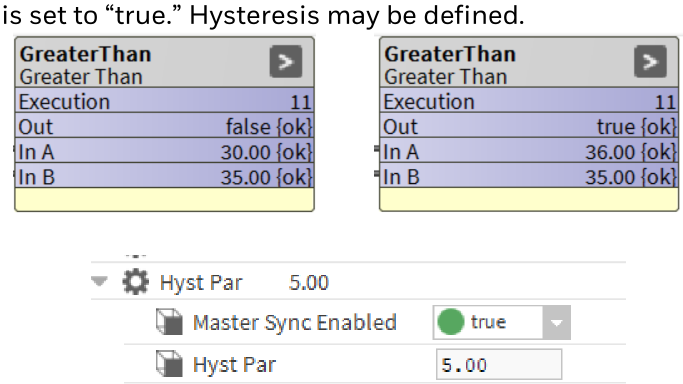


Fig. 205 GreaterThan Function Block

Greater Than Equal

The Greater Than Equal function compares two inputs. The block outputs "true" when the In A is greater than or equal to the In B; otherwise, it outputs "false."

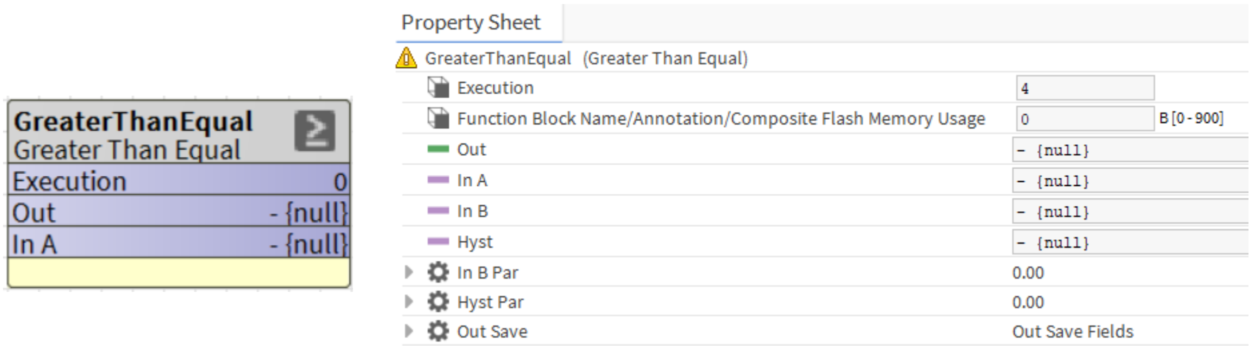


Fig. 206 Greater Than Equal Function Block and Property Sheet

The output is set to “true,” if **In A > = In B**.

The output does not change or “false”, if **In A = < (In B – Hyst**).

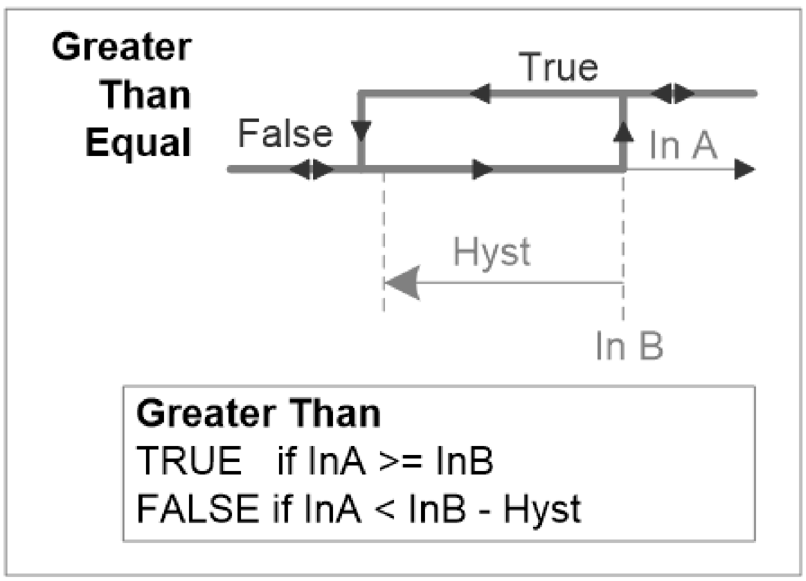


Fig. 207 Greater Than Equal Logic Diagram

Input

Table 146 Inputs of Greater Than Equal

Input Name	Description
In A	This is a 32-bit floating point input slot.
In B (In & Par)	This is a 32-bit floating point input slots.
Hyst	This hysteresis value is defined to avoid cycling output.

Output

Table 147 Output of Greater Than Equal

Output Name	Description
Out	**Output:** "true" or "false".

Parameter

Table 148 Parameters of Greater Than Equal

Parameter Name	Description
In B Par (In & Par)	If **In B** is "null," then **In B Par** is used.
Hyst Par (In & Par)	If **Hyst** is "null," then **Hyst Par** is used.
Out Save	• **Master Sync Enabled:** If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Example 1: GreaterThanEqual

If **In A** >= **In B**, **Out** is set to "true." Hysteresis may be defined.

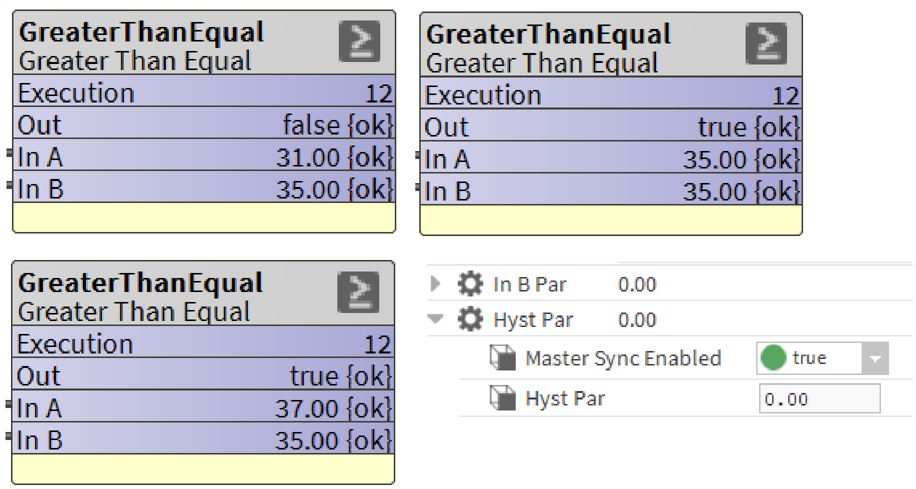


Fig. 208 GreaterThanEqual Function Block

Less Than

The Less Than function compares two inputs. When the first **In A** is less than the **In B**, the block outputs “true”; otherwise, it outputs “false.”

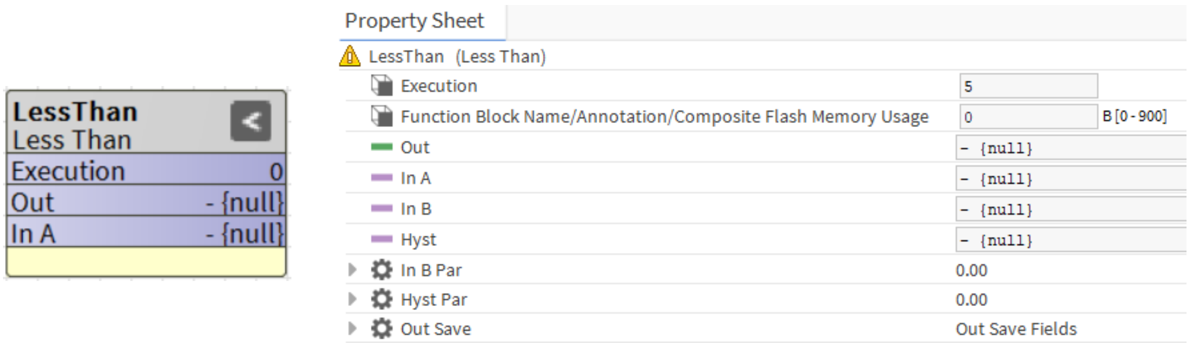


Fig. 209 Less Than Function Block and Property Sheet

The output is set to “true” if **In A < In B.**

The output does not change or “false,” if **In A > (In B + Hyst)**.

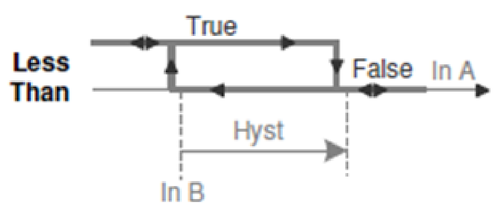


Fig. 210 Less Than Logic Diagram

Input

Table 149 Inputs of Less Than

Input Name	Description
In A	These are 32-bit floating point input slots.
In B (In & Par)	These are 32-bit floating point input slots.
Hyst (In & Par)	This hysteresis value is defined to avoid cycling output.

Output

Table 150 Outputs of Less Than

Output Name	Description
Out	**Output:** “true” or “false”.

Parameter

Table 151 Parameters of Less Than

Parameter Name	Description
In B Par (In & Par)	If **In B** is "null," **In B Par** is used.
Hyst Par (In & Par)	If **Hyst** is "null," **Hyst Par** is used.
Out Save	• **Master Sync Enabled:** If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Example 1: LessThan

If **In A** < **In B**, the output is set to "true." Hysteresis may be defined.

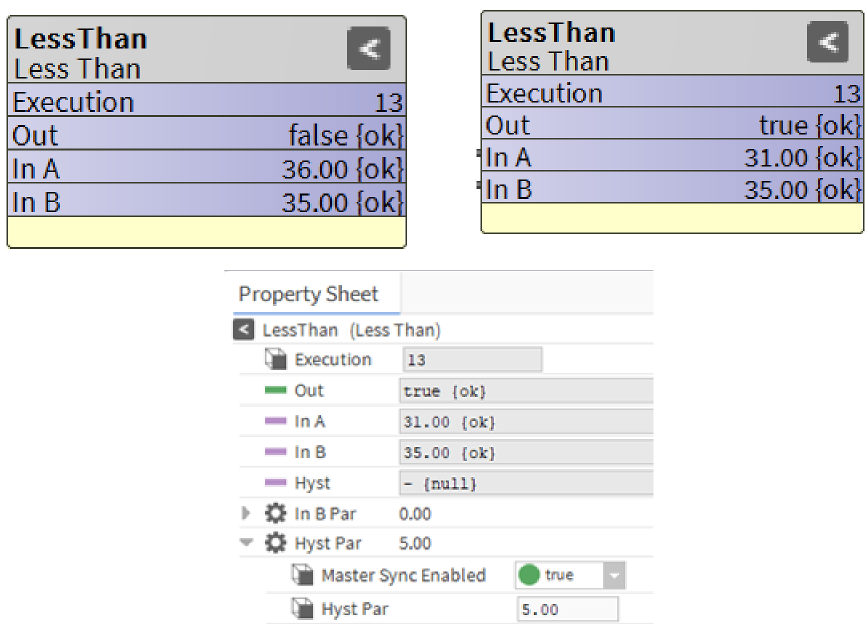


Fig. 211 Less Than Function Block and Property Sheet

Less Than Equal

The Less Than Equal function compares two inputs with each other. When the first In A is less than or equal to the In B, the block outputs “true”; otherwise, it outputs “false.”

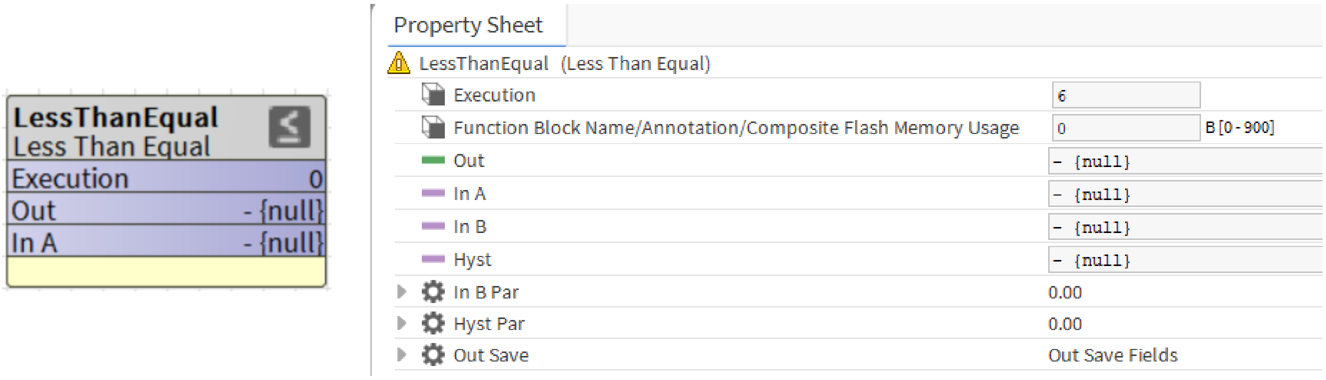


Fig. 212 Less Than Equal Function Block and Property Sheet

The output is set to “true,” if **In A <= In B.**

The output does not change or “false,” if **In A >= (In B + Hyst**).

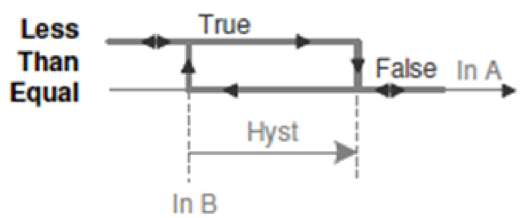


Fig. 213 Less Than Equal Logic Diagram

Input

Table 152 Inputs of Less Than Equal

Input Name	Description
In A	These are 32-bit floating point input slots.
In B (In & Par)	These are 32-bit floating point input slots.
Hyst (In & Par)	This hysteresis value is defined to avoid cycling output.

Output

Table 153 Output of Less Than Equal

Output Name	Description
Out	**Output:** “true” or “false.”

Parameter

Table 154 Parameters of Less Than Equal

Parameter Name	Description
In B Par (In & Par)	If **In B** is "null," then **In B Par** is used.
Hyst Par (In & Par)	If **Hyst** is "null," then **Hyst Par** is used.
Out Save	• **Master Sync Enabled:** If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Example 1: LessThanEqual

If **In A** <= **In B**, the output is set to "true." Hystersis may be defined.

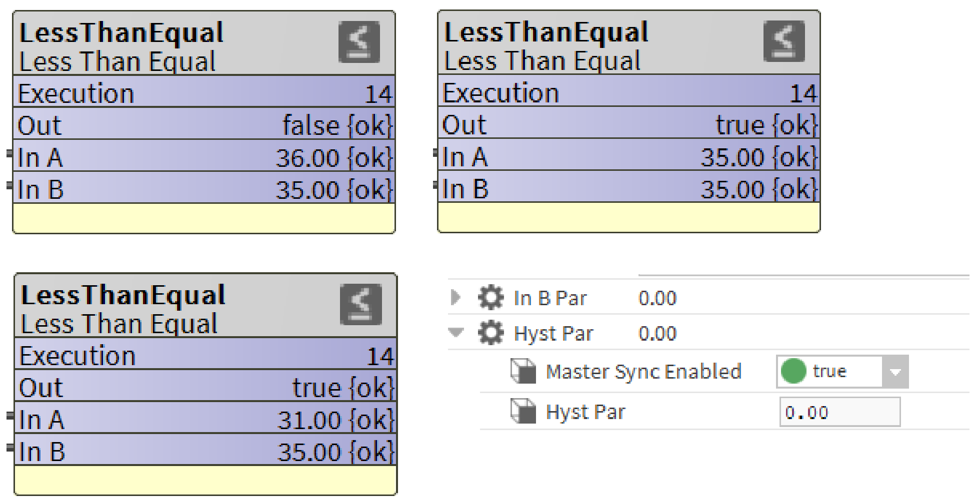


Fig. 214 LessThanEqual Function Block and Property Sheet

CHAPTER

9 SELECT SWITCH FUNCTION BLOCKS

The honIrmControl Palette provides the following Select Switch function blocks that can be configured and used to build the required application logic:

- Binary Select
- Binary Select Prio
- Binary Select Multi
- Binary Select Multi Prio
- Change Select
- Max Select Multi
- Min Select Multi
- Numeric Select
- Valid Select Prio
- Valid Select Multi Prio
- Numeric Switch

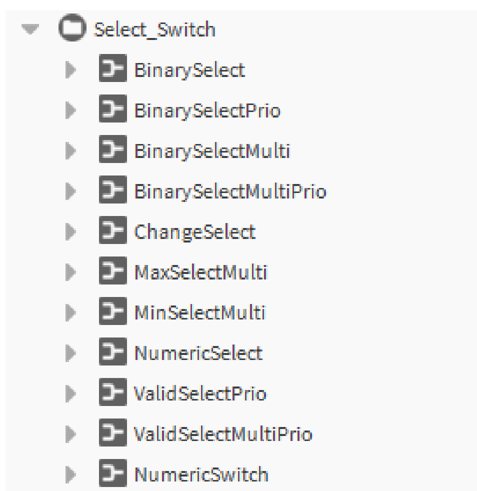


Fig. 215 Select Switch Function Blocks

Binary Select

The Binary Select function allows one to two inputs to be individually enabled in any combination. The input is converted into the output.

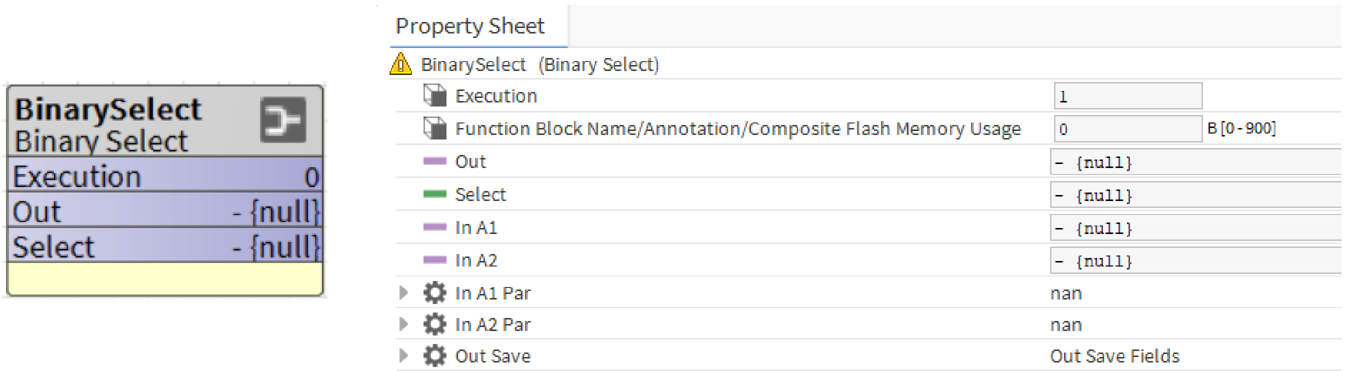


Fig. 216 BinarySelect Function Block and Property Sheet

- If **Select** is set to “false” (0) or “null,” **In A1** is equal to output.
- If **Select** is set to “true” (1), it switches **In A1** to **In A2**, and **In A2** has priority.

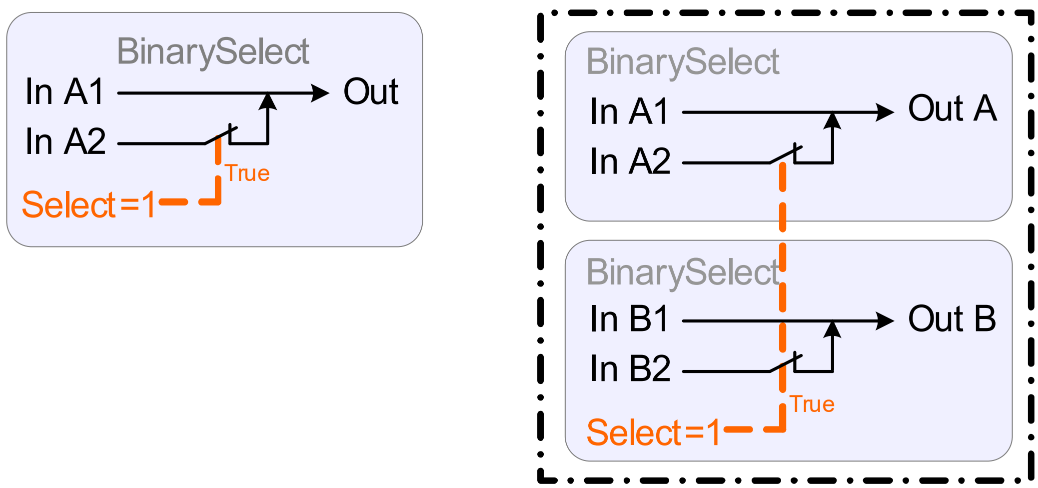


Fig. 217 BinarySelect Logic Diagram

Input

Table 155 Inputs of Binary Select

Input Name	Description
Select	This input decides whether **In A1** or **In A2** is used as output.
In A1 (In & Par)	If **Select** equals to “**0**,” then **Out** = **In A1**. It is used as a default value
In A2 (In & Par)	If **Select** equals to “**1**,” then **Out** = **In A2**. It is used as an override value.

Output

Table 156 Output of Binary Select

Output Name	Description
Out	• If **Select** equals to "**0**," then **Out** = **In A1**. • If **Select** equals to "**1**," then **Out** = **In A2**.

Parameter

Table 157 Parameters of Binary Select

Parameter Name	Description
In A1 Par (In & Par)	If **In A1** is "null," **In A1 Par** is used as a parameter. **In A1 Par** is set to "nan." In this case, it is treated like an invalid.
In A2 Par (In & P ar)	If **In A2** is "null," **In A2 Par** is used as a parameter. **In A2 Par** is set to "nan." In this case, it is treated like an invalid.
Out Save	• **Master Sync Enabled:** If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Example 1: BinarySelect

If the input is "true," **In A1** is set as output. If the input is "false," **In A2** is set as output.

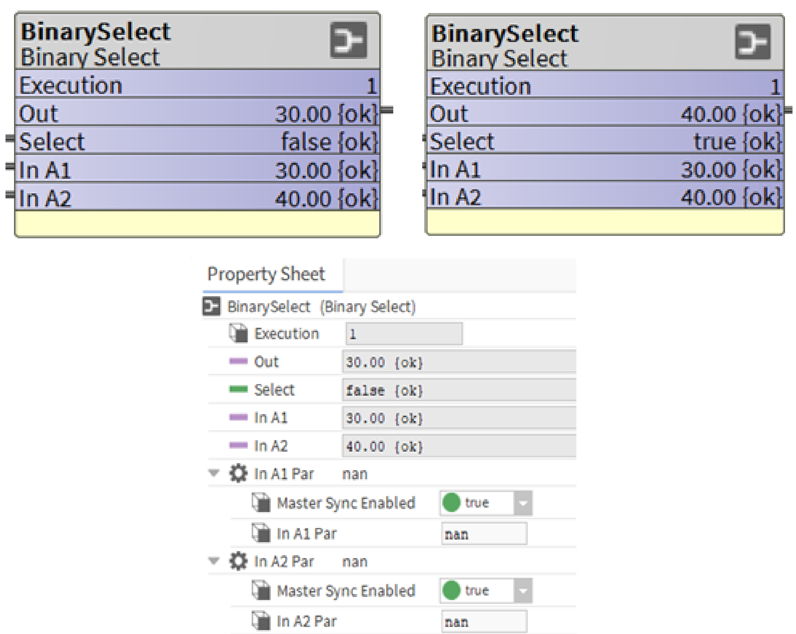


Fig. 218 BinarySelect Function Block and Property Sheet

Binary Select Prio

The Binary Select Prio function allows for input selection based on a select value. Each of the four numeric inputs is output is determined by the four binary inputs. If more than one binary input is "true" at the same time, the upper input takes priority.

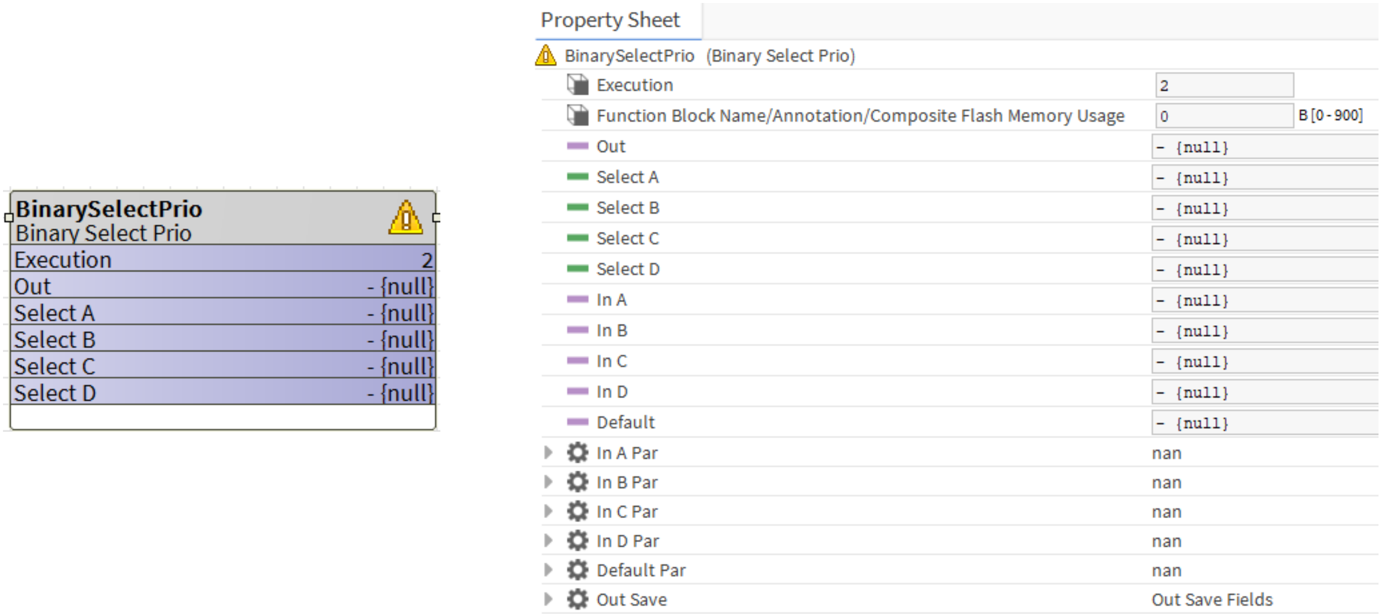


Fig. 219 BinarySelectPrio Function Block and Property Sheet

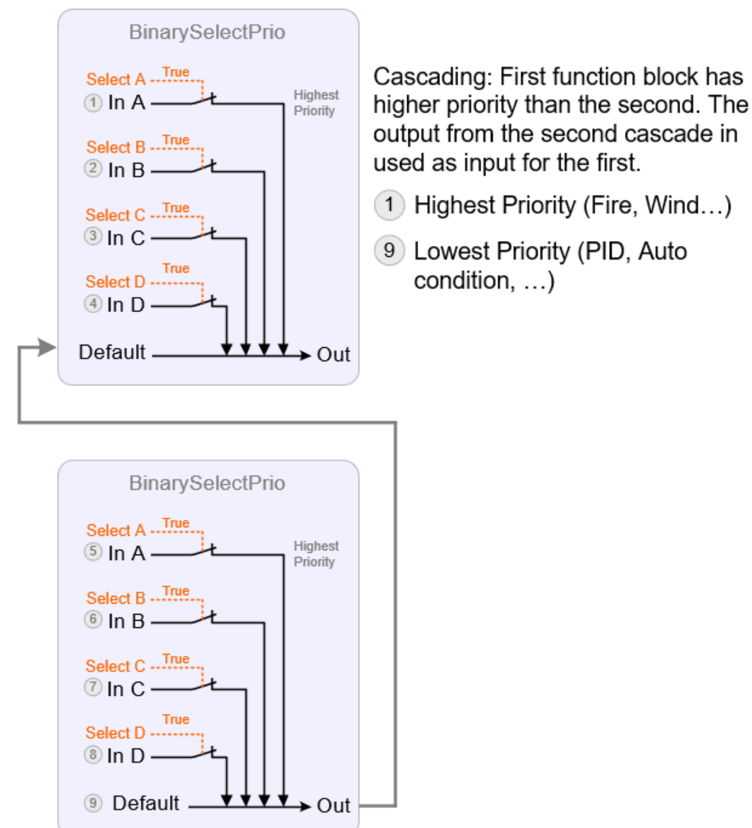


Fig. 220 BinarySelectPrio Logic Diagram

Input

Table 158 Inputs of Binary Select Prio

Input Name	Description
Select A – Select D	This input decides whether **In A**, **In B**, **In C**, or **In D** is used as output.
In A (In & Par)	If **Select A** equals to “1”, then **Out** = **In A**. It has the highest priority.
In B (In & Par)	If **Select B** equals to “1”, then **Out** = **In B**. **In B** has lower priority than **In A**.
In C (In & Par)	If **Select A** equals to “1”, then **Out** = **In A. In C** has lower priority than **In A** and **In B**.
In D (In & Par)	If **Select D** equals to “1”, then **Out** = **In D**. **In D** has lower priority than **In A**, **In B**, and **In C**.
Default (In & Par)	If all **Select** inputs are set to “false,” **Out** = **Default.**

Output

Table 159 Output of Binary Select Prio

Output Name	Description
Out	**Out** = **In A/In B/In C/In D**. • If **Select A** equals to “1”, then **Out** = **In A**. • If **Select B** equals to “1”, then **Out** = **In B**. • If **Select C** equals to “1”, then **Out** = **In C.** • If **Select D** equals to “1”, then **Out** = **In D.**

Parameter

Table 160 Parameters of Binary Select Prio

Parameter Name	Description
In A Par (In & Par)	If **In A** is “null,” **In A Par** is used as a parameter. **In A Par** is set to “nan.” In this case, it is treated like an invalid.
In B Par (In & Par)	If **In B** is “null,” **In B Par** is used as a parameter. **In B Par** is set to “nan.” In this case, it is treated like an invalid.
In C Par (In & Par)	If **In C** is “null,” **In C Par** is used as a parameter. **In C Par** is set to “nan.” In this case, it is treated like an invalid.
In D Par (In & Par)	If **In D** is “null,” **In D Par** is used as a parameter. **In D Par** is set to “nan.” In this case, it is treated like an invalid.
Default Par (In & Par)	If **Default** is “null,” **Default Par** is used as a parameter. In **Default Par** is set to “nan.” In this case, it is treated like an invalid.

Table 160 Parameters of Binary Select Prio (Continued)

Parameter Name	Description
Out Save	• **Master Sync Enabled:** If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Examples

Example 1: BinarySelectPrio

If any one of the inputs is "true," the corresponding input is set as output.

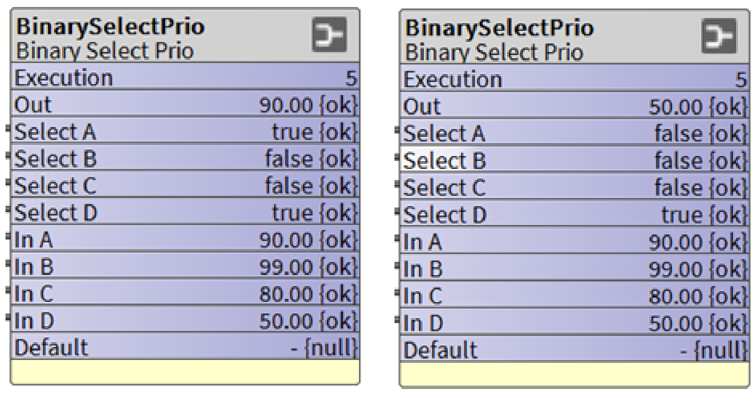


Fig. 221 BinarySelectPrio Function block

Example 2: BinarySelectPrio

If more than one input is "true," **In A** has the highest priority, **In D** has low priority. Accordingly, the output will be set.

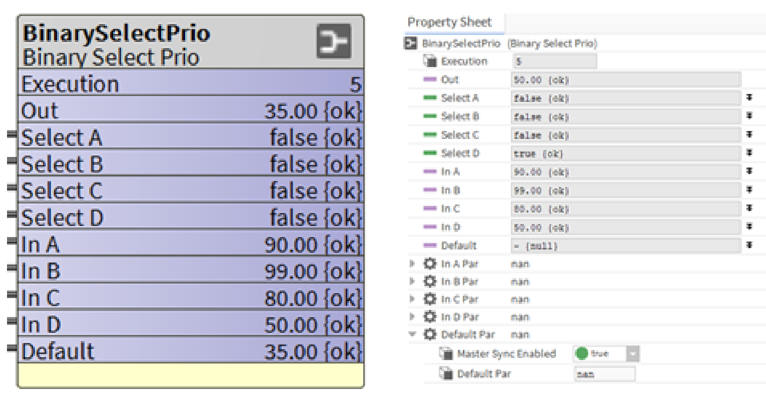


Fig. 222 BinarySelectPrio Function block and Property Sheet

Binary Select Multi

A binary input determines which of the two numeric inputs is output. The term "multi" refers to the switching of two independent inputs, similar to a relay with two changeover contacts.

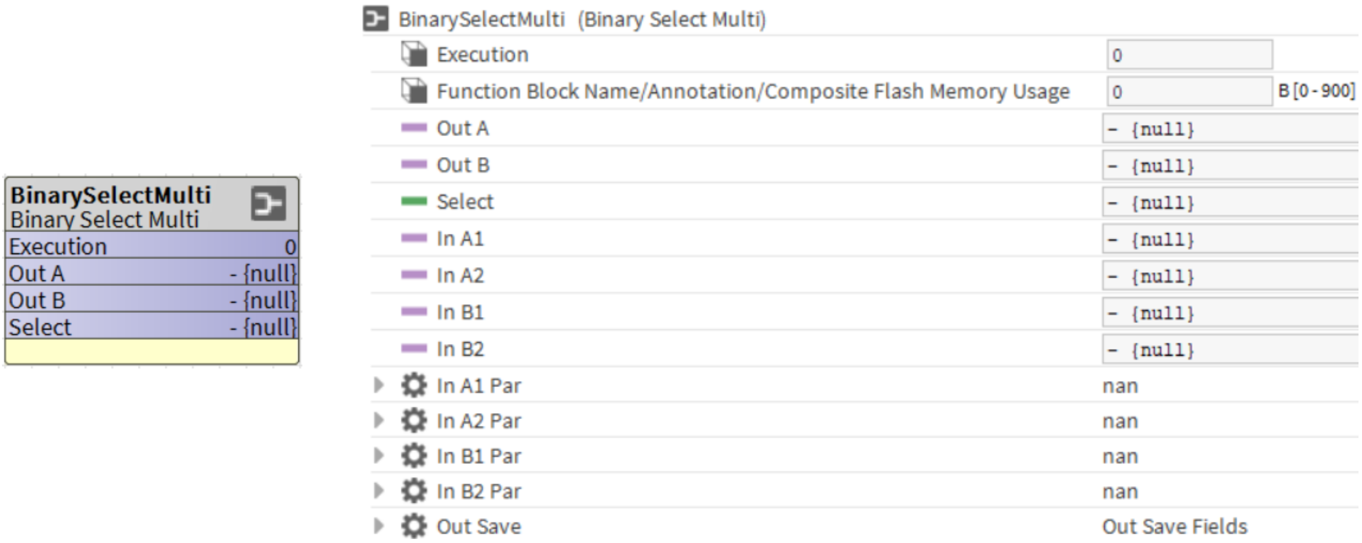


Fig. 223 BinarySelectMulti Function Block and Property Sheet

Instead of BinarySelectMulti functional block, the BinarySelect can be used two times. But this block is clear and avoids programming errors.

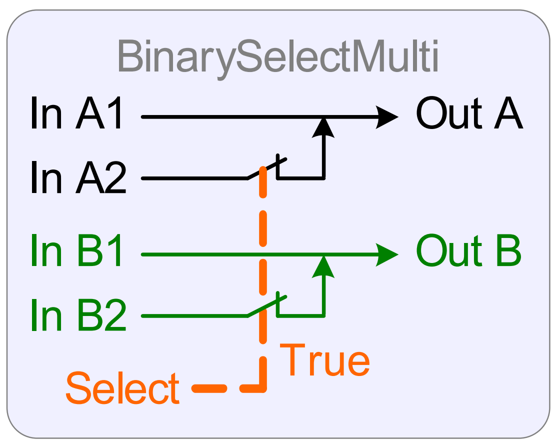


Fig. 224 BinarySelectMulti Logic Diagram

Input

Table 161 Inputs of Binary Select Multi

Input Name	Description
Select	This input decides whether **In A1**/**In B1** or **In A2**/**In B2** are used as outputs.
In A1 (In & Par)	If **Select** equals to “0,” then **Out** = **In A1**. It is used as a default value.

Table 161 Inputs of Binary Select Multi (Continued)

Input Name	Description
In A2 (In & Par)	If **Select** equals to “1,” then **Out** = **In A2**. It is used as an override value.
In B1 (In & Par)	If **Select** equals to “0,” then **Out** = **In B1**. It is used as a default value.
In B2 (In & Par)	If **Select** equals to “1,” then **Out** = **In B2**. It is used as an override value.

Output

Table 162 Outputs of Binary Select Multi

Output Name	Description
Out A	• If **Select** equals to “0,” then **Out** = **In A1** or **In B1**. • If **Select** equals to “1,” then **Out** = **In A2** or **In B2**.
Out B	• If **Select** equals to “0”, then **Out** = **In A1** or **In B1**. • If **Select** equals to “1,” then **Out** = **In A2** or **In B2**.

Parameter

Table 163 Parameters of Binary Select Multi

Parameter Name	Description
In A1 Par (In & Par)	If **In A1** is “null,” **In A1 Par** is used as a parameter. **In A1 Par** is set to “nan.” In this case, it is treated like an invalid.
In A2 Par (In & Par)	If **In A2** is “null,” **In A2 Par** is used as a parameter. **In A2 Par** is set to “nan.” In this case, it is treated like an invalid.
In B1 Par (In & Par)	If **In B1** is “null,” **In B1 Par** is used as a parameter. **In B1 Par** is set to “nan.” In this case, it is treated like an invalid.
In B2 Par (In & Par)	If **In B2** is “null,” **In B2 Par** is used as a parameter. **In B2 Par** is set to “nan.” In this case, it is treated like an invalid.
Out Save	• **Master Sync Enabled:** If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out A:** To enable or disable the **Out A** feature. • **Out B:** To enable or disable the **Out B** feature.

Examples

Example 1: BinarySelectMulti

If **Select** = “false,” **In A1** is set to **Out A** , and **In B1** is set to **Out B**.

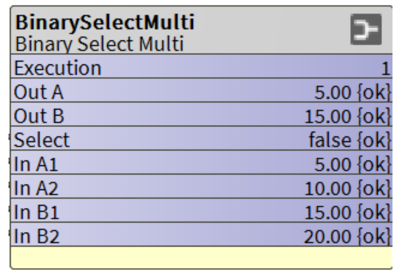


Fig. 225 BinarySelectMulti Function Block

Example 2: BinarySelectMulti

If **Select** = “true,” **In A2** is set to **Out A**, and **In B2** is set to **Out B**.

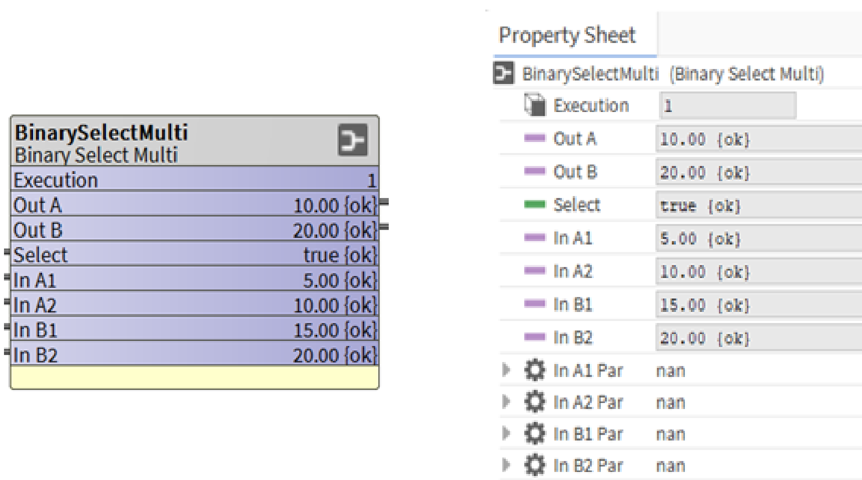


Fig. 226 BinarySelectMulti Function Block

Binary Select Multi Prio

The output of four numeric inputs is determined by the four binary inputs. When more than one binary input is true at the same time, the higher input takes priority. The term "multi" refers to the switching of two independent inputs, similar to a relay with two changeover contacts.

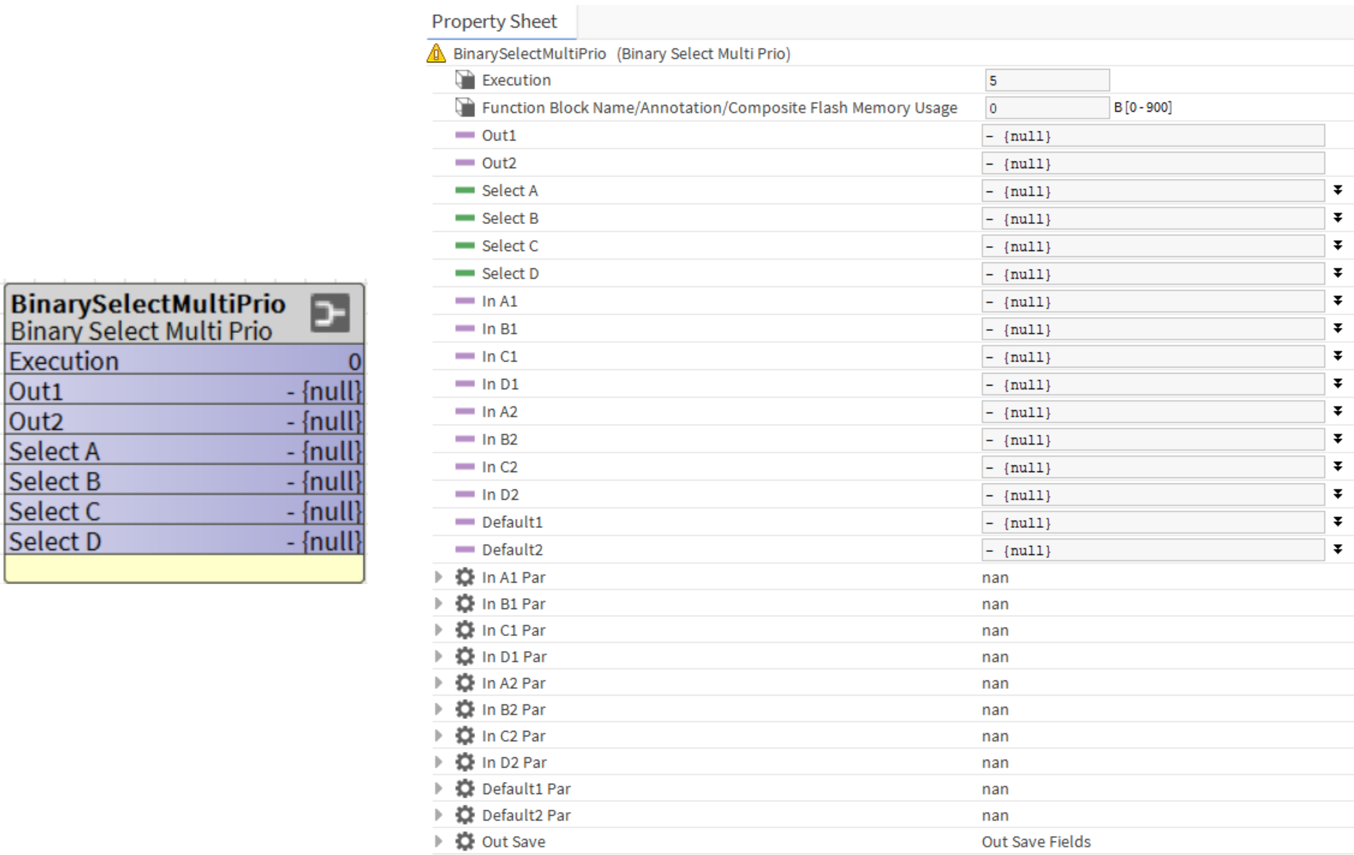


Fig. 227 BinarySelectMultiPrio Function Block and Property Sheet

The first function block has a higher priority than the second. The output from the second cascade is used as input for the first. In the figure BinarySelectMultiPrio Logic Diagram , **In A1** has the highest priority (fire and wind) than default value 9 (PID and auto condition.

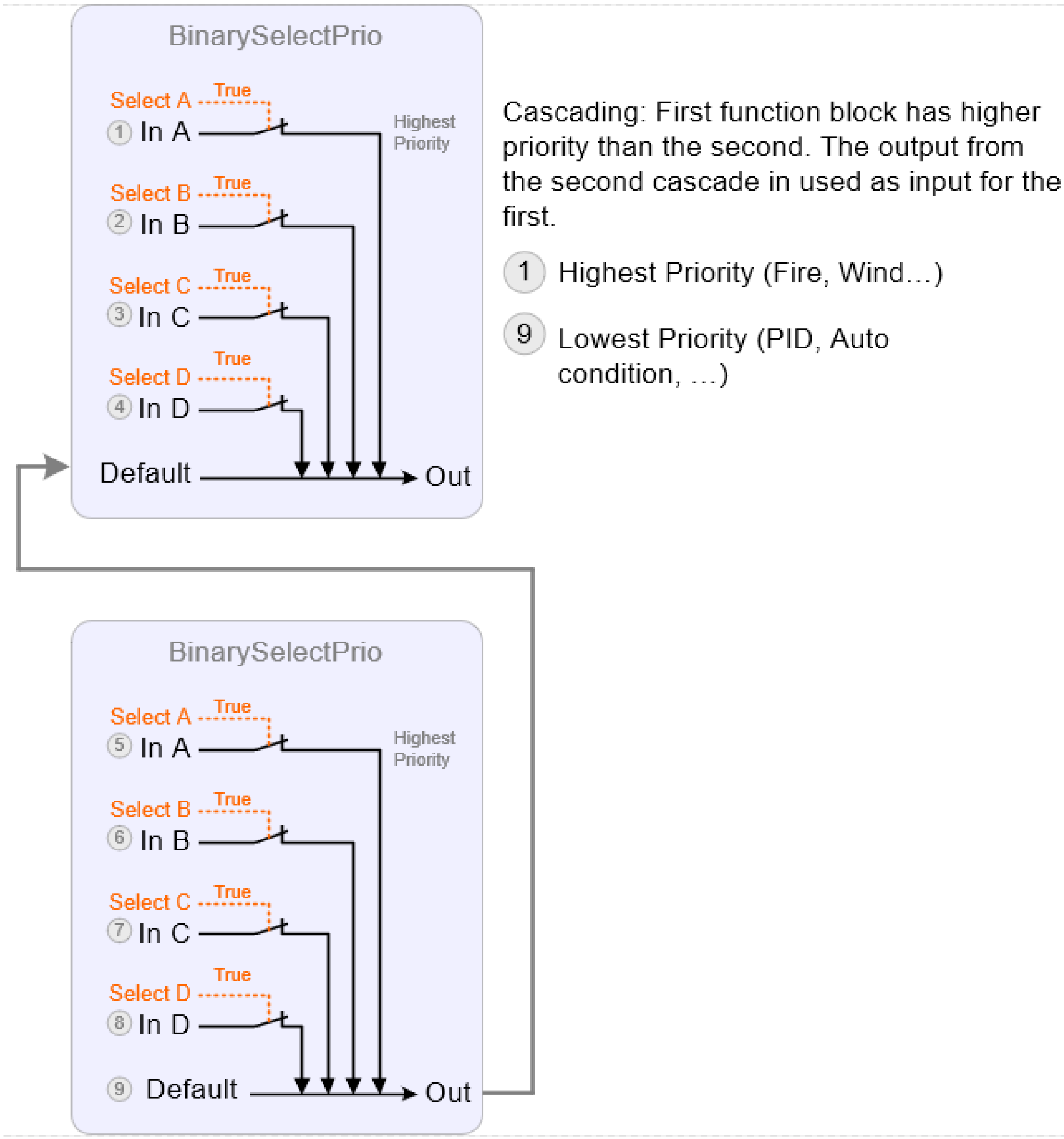


Fig. 228 BinarySelectMultiPrio Logic Diagram

Input

Table 164 Inputs of Binary Select Multi Prio

Input Name	Description
Select A – Select D	It's a priority selector with a Boolean value. Input will be chosen for output based on the true status value. If there is more than one true status, a priority value will be selected. “A” means high priority “D” means low priority.

Table 164 Inputs of Binary Select Multi Prio (Continued)

Input Name	Description
In A1 – In A2 In B1 – In B2 In C1 – In C2 In D1 – In D2 (In & Par)	These slots are 32-bit floating point input slots. This input is selected as an output based on priority selector “true” status.
Default 1 – Default 2 (In & Par)	This slot is a 32-bit floating point input slot. If none of the priorities is active default value will be set as an output.

Output

Table 165 Outputs of Binary Select Multi Prio

Output Name	Description
Out 1	**Output: In A1/In B1/In C1/In D1**.
Out 2	**Output: In A2/In B2/In C2/In D2**.

Parameter

Table 166 Parameters of Binary Select Multi Prio

Parameter Name	Description
In A1 Par (In & Par)	If **In A1** is “null,” **In A1 Par** is used as a parameter. **In A1 Par** is set to “nan.” In this case, it is treated like an invalid.
In B1 Par (In & Par)	If **In B1** is “null,” **In B1 Par** is used as a parameter. **In B1 Par** is set to “nan.” In this case, it is treated like an invalid.
In C1 Par (In & Par)	If **In C1** is “null,” **In C1 Par** is used as a parameter. **In C1 Par** is set to “nan.” In this case, it is treated like an invalid.
In D1 Par (In & Par)	If **In D1** is “null,” **In D1 Par** is used as a parameter. **In D1 Par** is set to “nan.” In this case, it is treated like an invalid.
In A2 Par (In & Par)	If **In A2** is “null,” **In A2 Par** is used as a parameter. **In A2 Par** is set to “nan.” In this case, it is treated like an invalid.
In B2 Par (In & Par)	If **In B2** is “null,” **In B2 Par** is used as a parameter. **In B2 Par** is set to “nan.” In this case, it is treated like an invalid.
In C2 Par (In & Par)	If **In C2** is “null,” **In C2 Par** is used as a parameter. **In C2 Par** is set to “nan.” In this case, it is treated like an invalid.
In D2 Par (In & Par)	If **In D2** is “null,” **In D2 Par** is used as a parameter. **In D2 Par** is set to “nan.” In this case, it is treated like an invalid.
Default1 Par (In & Par)	If **Default1** is “null,” **Default1 Par** is used as a parameter. In **Default1 Par** is set to “nan.” In this case, it is treated like an invalid.

Table 166 Parameters of Binary Select Multi Prio (Continued)

Parameter Name	Description
Default2 Par (In & Par)	If **Default2** is “null,” **Default2 Par** is used as a parameter. In **Default2 Par** is set to “nan.” In this case, it is treated like an invalid.
Out Save	• **Master Sync Enabled:** If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Examples

Example 1: BinarySelectMultiPrio

For priority selection, the highest priority selector “true” status value is selected as output.

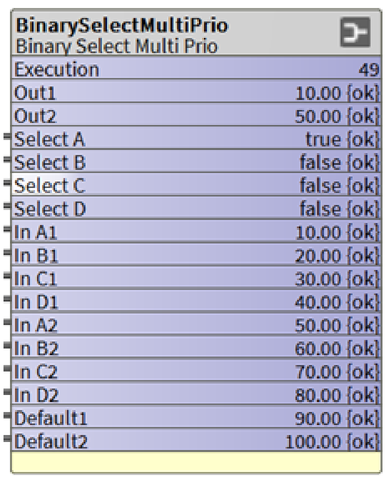


Fig. 229 BinarySelectMultiPrio – Priority Selection

Example 2: BinarySelectMultiPrio

For default selection, if all the priority selector values are "false.".

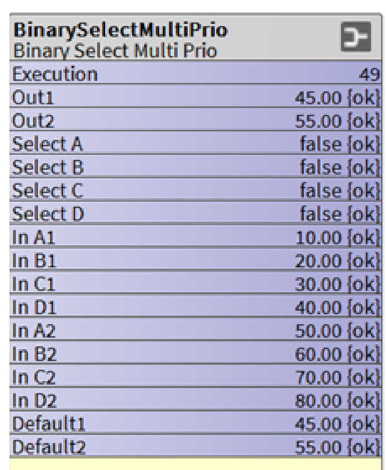


Fig. 230 BinarySelectMultiPrio – Default Selection

Change Select

The Change Select function block checks whether the input values have changed compared to the last DDC cycle. If the value has changed, it sets the corresponding Boolean output to “true” for 1 DDC cycle. More than one output can be “true” if numerous changes are detected.

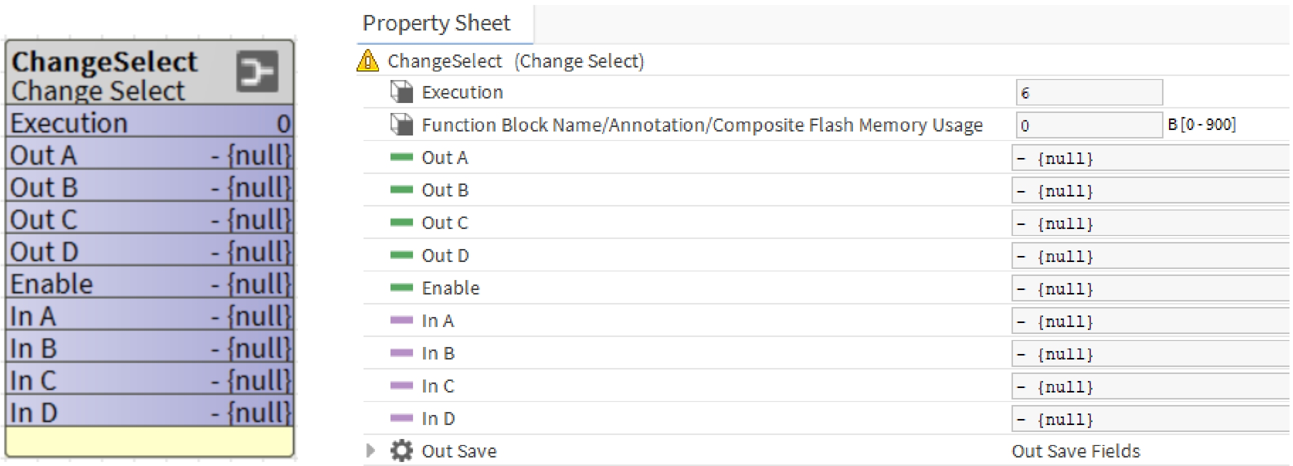


Fig. 231 ChangeSelect Function Block and Property Sheet

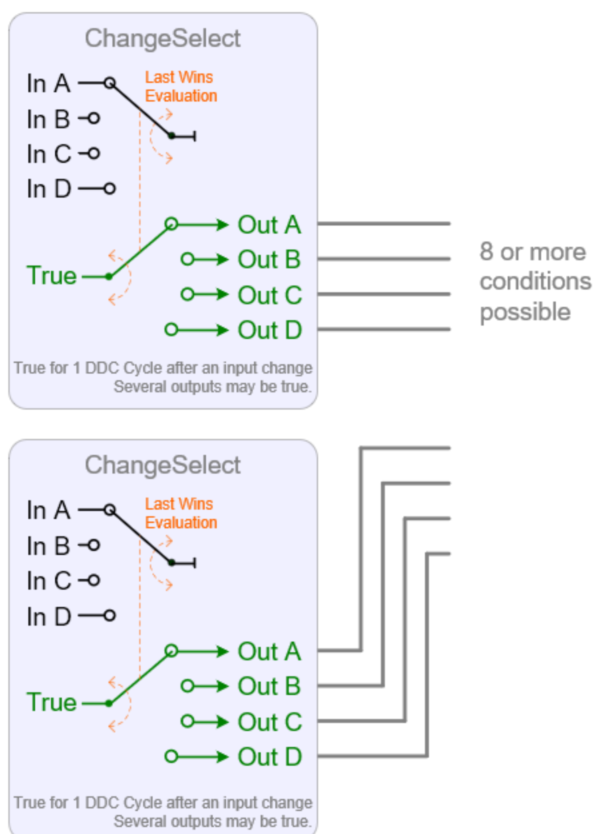


Fig. 232 ChangeSelect Logic Diagram

Input

Table 167 Inputs of Change Select

Input Name	Description
Enable	It is a Boolean point that is used to enable the block. • If it is set to “false,” there is no change, and all outputs remain “false.” • If it is set to “true,” it activates the change detection.
In A	If this input changes from the previous DDC cycle, **Out A** becomes “true.”
In B	If this input changes from the previous DDC cycle, **Out B** becomes “true.”
In C	If this input changes from the previous DDC cycle, **Out C** becomes “true.”
In D	If this input changes from the previous DDC cycle, **Out A** becomes “true.”

Output

Table 168 Outputs of Change Select

Output Name	Description
Out A	If **In A** changes from the previous DDC cycle, **Out A** becomes “true.” • The output is “true” only in the DDC cycle where the change is detected. • A change to “null” or from “null” is a valid change.
Out B	If **In B** changes from the previous DDC cycle, **Out B** becomes “true.” • The output is “true” only in the DDC cycle where the change is detected. • A change to “null” or from “null” is a valid change.
Out C	If **In C** changes from the previous DDC cycle, **Out C** becomes “true.” • The output is “true” only in the DDC cycle where the change is detected. • A change to “null” or from “null” is a valid change.
Out D	If **In D** changes from the previous DDC cycle, **Out D** becomes “true.” • The output is “true” only in the DDC cycle where the change is detected. • A change to “null” or from “null” is a valid change.

Parameter

Table 169 Parameters of Change Select

Parameter Name	Description
Out Save	• **Master Sync Enabled:** If you set it to “TRUE,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **OutA:** To enable or disable the **OutA** feature. • **OutB:** To enable or disable the **OutB** feature. • **OutC:** To enable or disable the **OutC** feature. • **OutD:** To enable or disable the **OutD** feature.

Example1: ChangeSelect

If Block **Enable** is “true,” change in any of the input values, the concerned output is set to “true” for a short while.

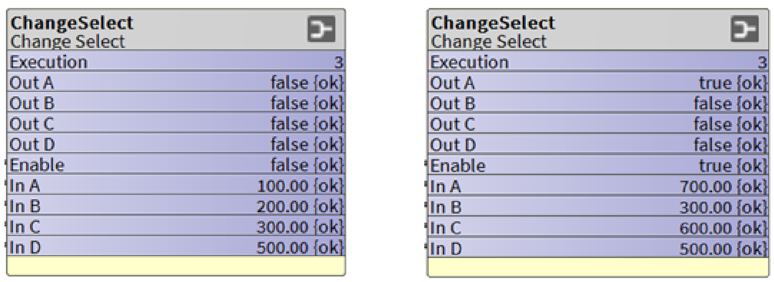


Fig. 233 ChangeSelect Function Block

Max Select Multi

The Max Select Multi function block makes a maximum selection of the inputs A1, B1, C1, and D1. Multi means that two independent inputs are switched, like a relay with two changeover contacts. “In A2 – In D2” are not used to determine the maximum; instead, they are looped through to the corresponding output with the same switching position as the maximum position.

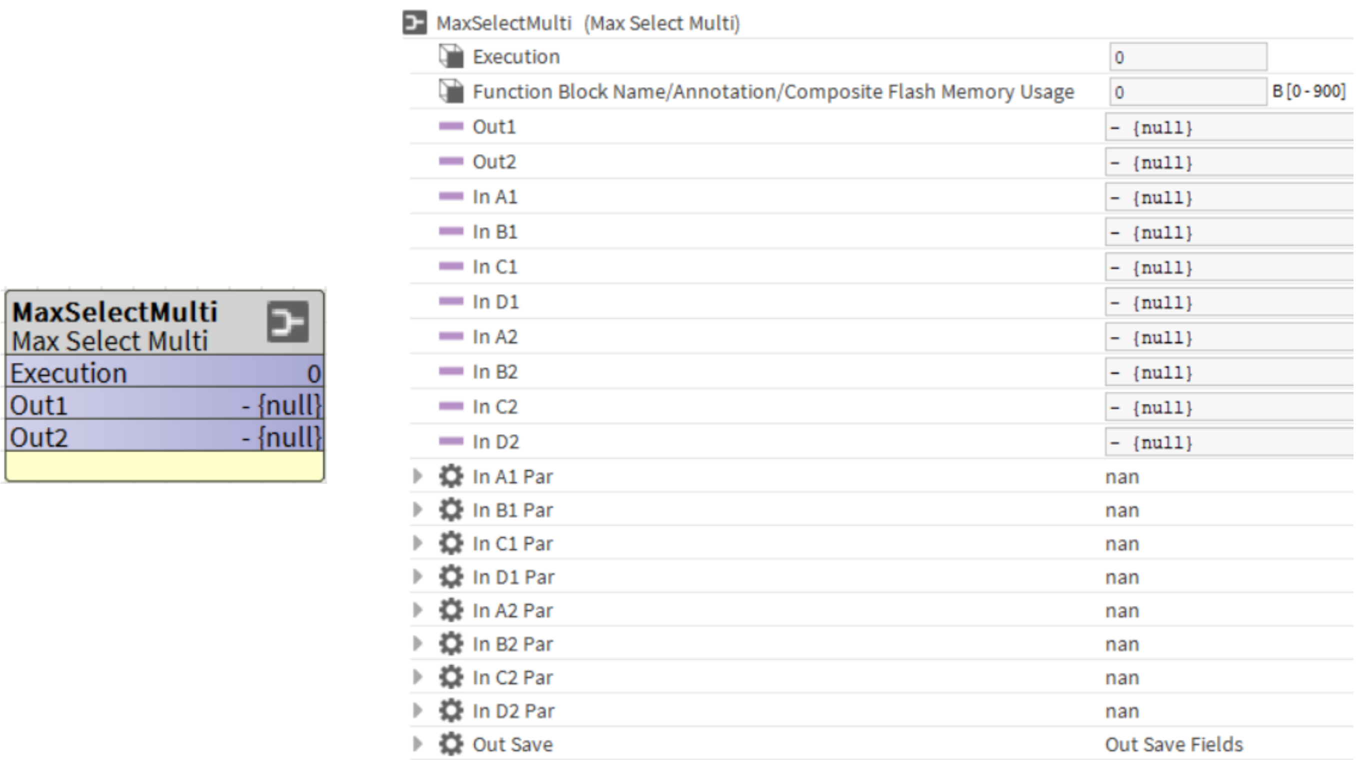


Fig. 234 MaxSelectMulti Function Block and Property Sheet

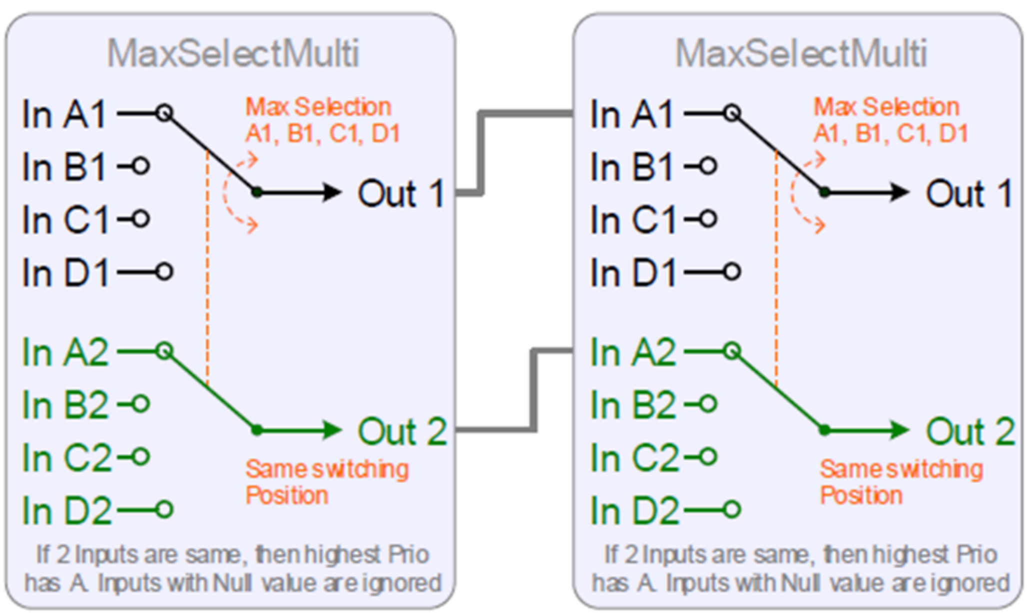


Fig. 235 MaxSelectMulti Logic Diagram

The actuator is commanded to 85% from the PID, though frost protection is active.

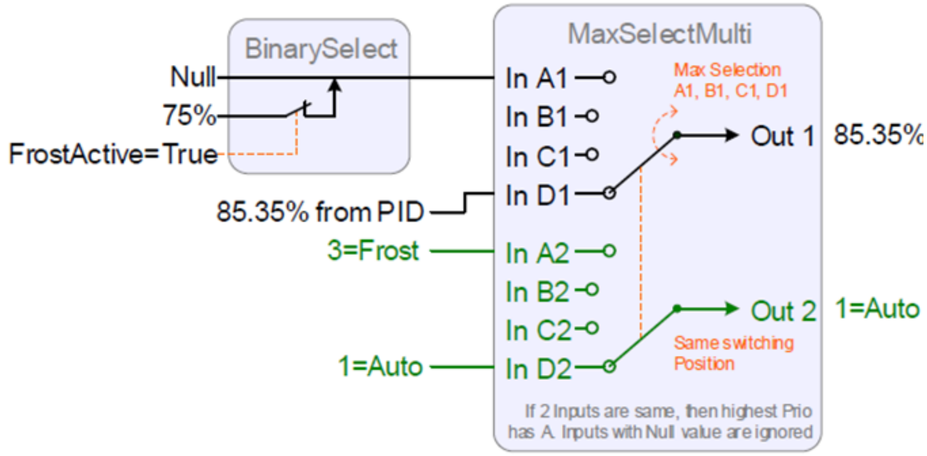


Fig. 236 MaxSelectMulti Logic Diagram with BinarySelect

Input

Table 170 Inputs of Max Select Multi

Input Name	Description
In A 1 (In & Par)	This input is used for maximum selection. If another input has the same value, this input is taken with higher priority. The priority is only important for **In A2** and **In D2**.
In B1 (In & Par)	This input is used for maximum selection. The input has higher priority than **In C1** and **In D1**.
In C1 (In & Par)	This input is used for maximum selection. The input has higher priority than **In D1**.
In D1 (In & Par)	This input is used for maximum selection.
In A2 (In & Par)	If **In A1** is the maximum input between **In A1** – **In D1**, **Out 2** = **In A2**.
In B2 (In & Par)	If **In B1** is the maximum input between **In A1** – **In D1**, **Out 2** = **In B2**.
In C2 (In & Par)	If **In C1** is the maximum input between **In A1** – **In D1**, **Out 2** = **In C2**.
In D2 (In & Par)	If **In D1** is the maximum input between **In A1** – **In D1**, **Out 2** = **In D2**.

Output

Table 171 Outputs of Max Select Multi

Output Name	Description
Out1	**Output:** The maximum value of inputs A1, B1, C1 and D1.

Table 171 Outputs of Max Select Multi (Continued)

Output Name	Description
Out2	This is used to output one of the input values of A2, B2, C2, and D2. The same switch position is used that comes from the maximum selection between A1, B1, C1, and D2. **Example:** If **In B1** is the maximum from inputs A1, B1, C1, and D1, **Out 2** = **In B**.

Parameter

Table 172 Parameters of Max Select Multi

Parameter Name	Description
In A1 Par (In & Par)	If **In A1** is "null," **In A1 Par** is used as a parameter. **In A1 Par** is set to "nan." In this case, it is treated like an invalid.
In B1 Par (In & Par)	If **In B1** is "null," **In B1 Par** is used as a parameter. **In B1 Par** is set to "nan." In this case, it is treated like an invalid.
In C1 Par (In & Par)	If **In C1** is "null," **In C1 Par** is used as a parameter. **In C1 Par** is set to "nan." In this case, it is treated like an invalid.
In D1 Par (In & Par)	If **In D1** is "null," **In D1 Par** is used as a parameter. **In D1 Par** is set to "nan." In this case, it is treated like an invalid.
In A2 Par (In & Par)	If **In A2** is "null," **In A2 Par** is used as a parameter. **In A2 Par** is set to "nan." In this case, it is treated like an invalid.
In B2 Par (In & Par)	If **In B2** is "null," **In B2 Par** is used as a parameter. **In B2 Par** is set to "nan." In this case, it is treated like an invalid.
In C2 Par (In & Par)	If **In C2** is "null," **In C2 Par** is used as a parameter. **In C2 Par** is set to "nan." In this case, it is treated like an invalid.
In D2 Par (In & Par)	If **In D2** is "null," **In D2 Par** is used as a parameter. **In D2 Par** is set to "nan." In this case, it is treated like an invalid.
Out Save	• **Master Sync Enabled:** If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Example 1: MaxSelectMulti

The maximum value from Input A1, B1, C1, D1 is set to **Out1**. And same input value from A2, B2, C2, D2 is set to **Out 2**.

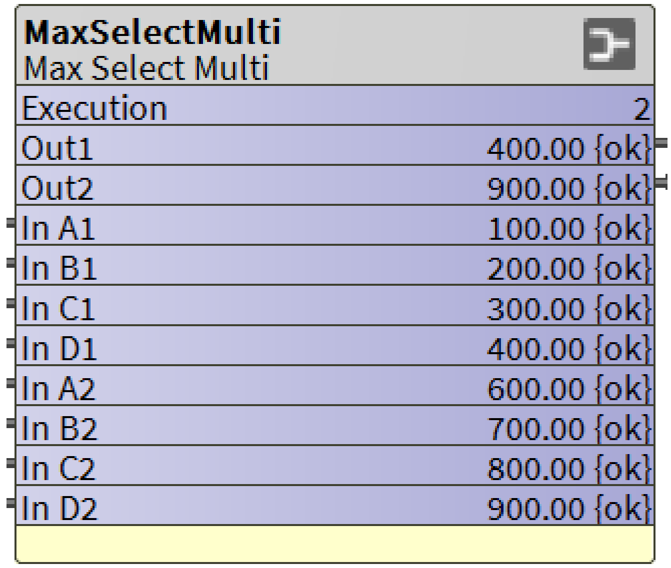


Fig. 237 MaxSelectMulti Function Block

Min Select Multi

The Min Select Multi function block makes a minimum selection of the inputs A1, B1, C1, and D1. Multi means that two independent inputs are switched, like a relay with two changeover contacts. “In A2 – In D2” are not used to determine the minimum; instead, they are looped through to the corresponding output with the same switching position as the minimum position.

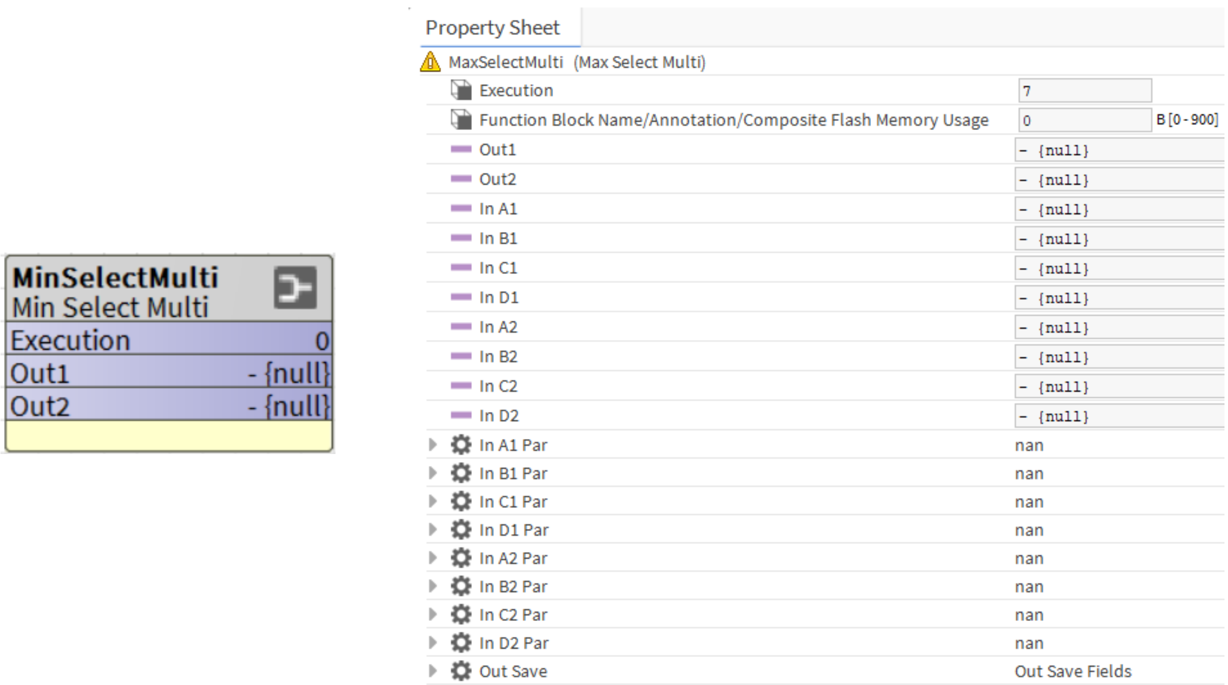


Fig. 238 MinSelectMulti Function Block and Property Sheet

If two inputs are the same, then A is considered as the highest priority. Inputs with null are ignored.

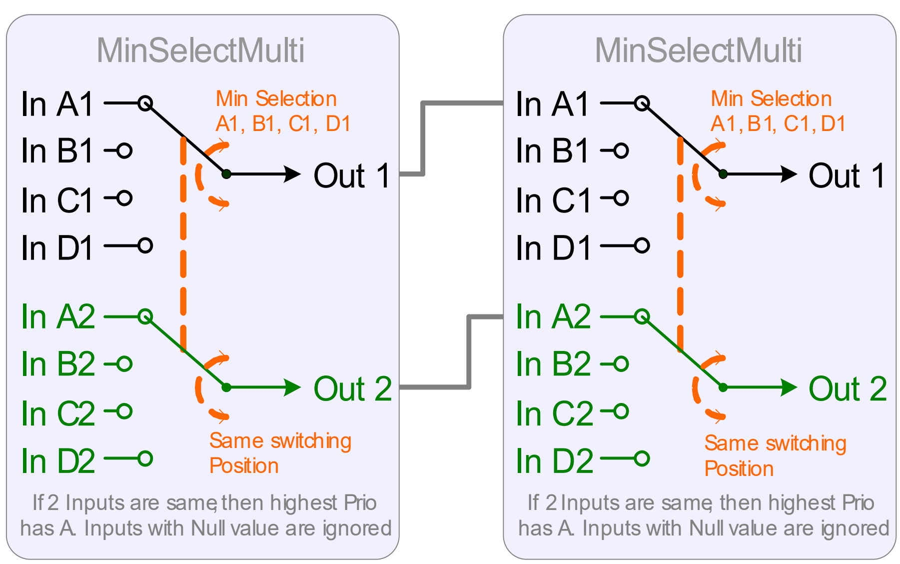


Fig. 239 MinSelectMulti Logic Diagram

The Actuator has 0% because of missing Airflow.

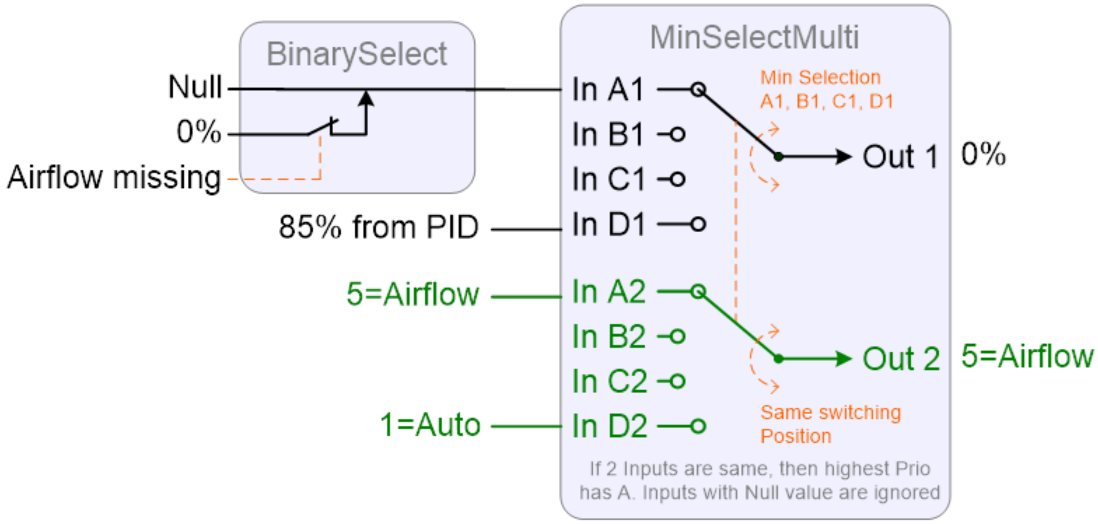


Fig. 240 MinSelectMulti Logic Diagram with BinarySelect

Input

Table 173 Inputs of Min Select Multi

Input Name	Description
In A 1 (In & Par)	This input is used for minimum selection. If another input has the same value, this input is taken with higher priority. The priority is only important for **In A2** and **In D2**.
In B1 (In & Par)	This input is used for minimum selection. The input has a higher priority than **In C1** and **In D1**.
In C1 (In & Par)	This input is used for minimum selection. The input has a higher priority than **In D1**.
In D1 (In & Par)	This input is used for maximum selection.
In A2 (In & Par)	If **In A1** is the minimum input between **In A1** – **In D1**, **Out 2** = **In A2**.
In B2 (In & Par)	If **In B1** is the minimum input between **In A1** – **In D1**, **Out 2** = **In B2**.
In C2 (In & Par)	If **In C1** is the minimum input between **In A1** – **In D1**, **Out 2** = **In C2**.
In D2 (In & Par)	If **In D1** is the minimum input between **In A1** – **In D1**, **Out 2** = **In D2**.

Output

Table 174 Outputs of Min Select Multi

Output Name	Description
Out1	**Output:** The minimum value of inputs A1, B1, C1, and D1.

Table 174 Outputs of Min Select Multi (Continued)

Output Name	Description
Out2	This is used to output one of the input values of A2, B2, C2, and D2. The same switch position is used that comes from the minimum selection between A1, B1, C1, and D2. **Example:** If **In B1** is the minimum from inputs A1, B1, C1, and D1, **Out 2** = **In B**.

Parameter

Table 175 Parameters of Min Select Multi

Parameter Name	Description
In A1 Par (In & Par)	If **In A1** is “null,” **In A1 Par** is used as a parameter. **In A1 Par** is set to “nan.” In this case, it is treated like an invalid.
In B1 Par (In & Par)	If **In B1** is “null,” **In B1 Par** is used as a parameter. **In B1 Par** is set to “nan.” In this case, it is treated like an invalid.
In C1 Par (In & Par)	If **In C1** is “null,” **In C1 Par** is used as a parameter. **In C1 Par** is set to “nan.” In this case, it is treated like an invalid.
In D1 Par (In & Par)	If **In D1** is “null,” **In D1 Par** is used as a parameter. **In D1 Par** is set to “nan.” In this case, it is treated like an invalid.
In A2 Par (In & Par)	If **In A2** is “null,” **In A2 Par** is used as a parameter. **In A2 Par** is set to “nan.” In this case, it is treated like an invalid.
In B2 Par (In & Par)	If **In B2** is “null,” **In B2 Par** is used as a parameter. **In B2 Par** is set to “nan.” In this case, it is treated like an invalid.
In C2 Par (In & Par)	If **In C2** is “null,” **In C2 Par** is used as a parameter. **In C2 Par** is set to “nan.” In this case, it is treated like an invalid.
In D2 Par (In & Par)	If **In D2** is “null,” **In D2 Par** is used as a parameter. **In D2 Par** is set to “nan.” In this case, it is treated like an invalid.
Out Save	**Master Sync Enabled:** If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. **Out1:** To enable or disable **Out1** feature. **Out2:** To enable or disable **Out2** feature.

Example 1: MinSelectMulti

The minimum value from Input A1, B1, C1, and D1 is set to **Out1**. And same input value from A2, B2, C2, and D2 is set to **Out 2**.

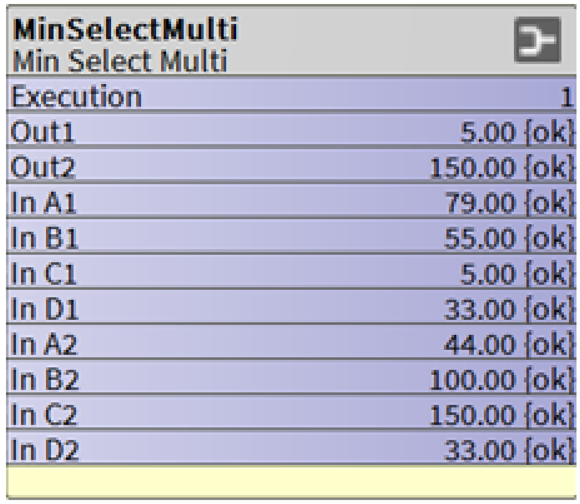


Fig. 241 MinSelectMulti Function block

Numeric Select

The Numeric Select function block chooses one of the six output values from the six input values. If the value in the selection is incorrect, the default input is used. If the six inputs are insufficient, the function can be extended to more inputs using an offset.

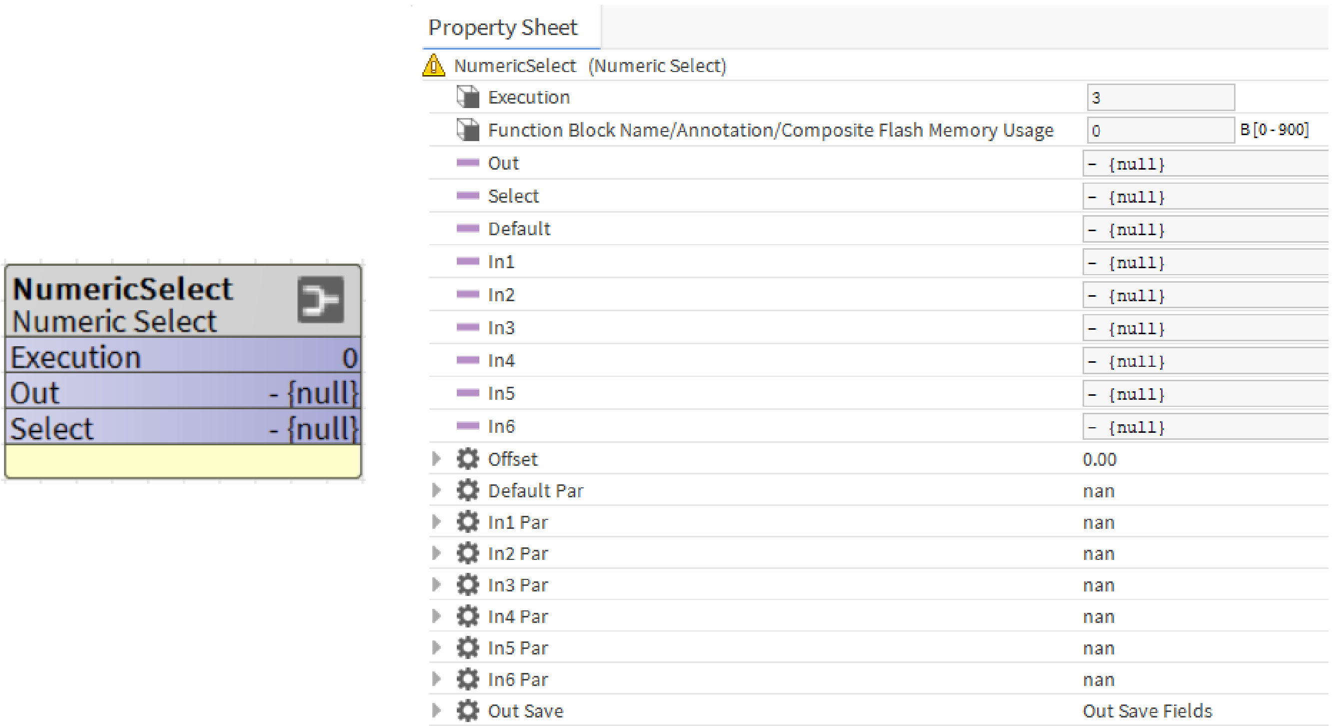


Fig. 242 NumericSelect Function Block and Property Sheet

When (Offset) selects one of the six inputs, the output equals the value on the input (Offset). Otherwise, the output equals the value on the default input.

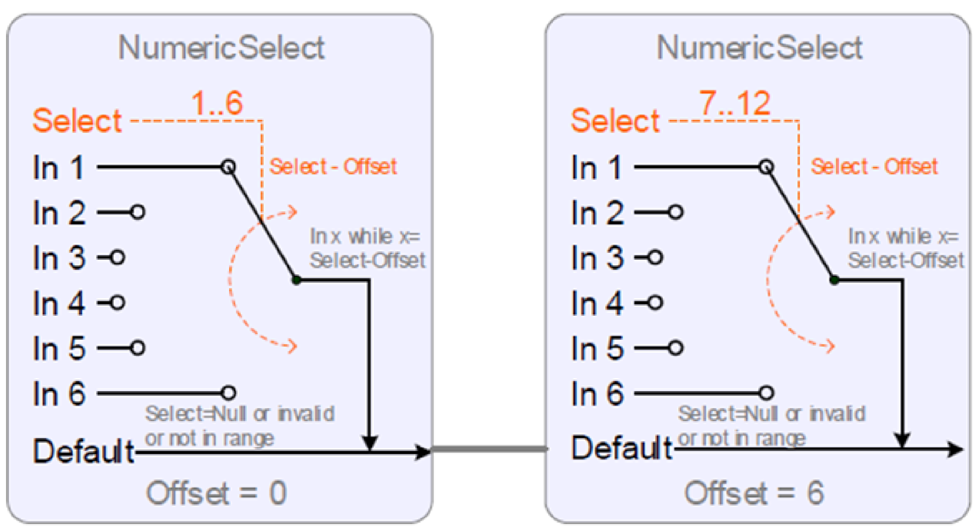


Fig. 243 NumericSelect Logic Diagram

Example: Wall moduleA - OutSetptMd > PidA - Operation3, 4, 5 > 1, 3, 2.

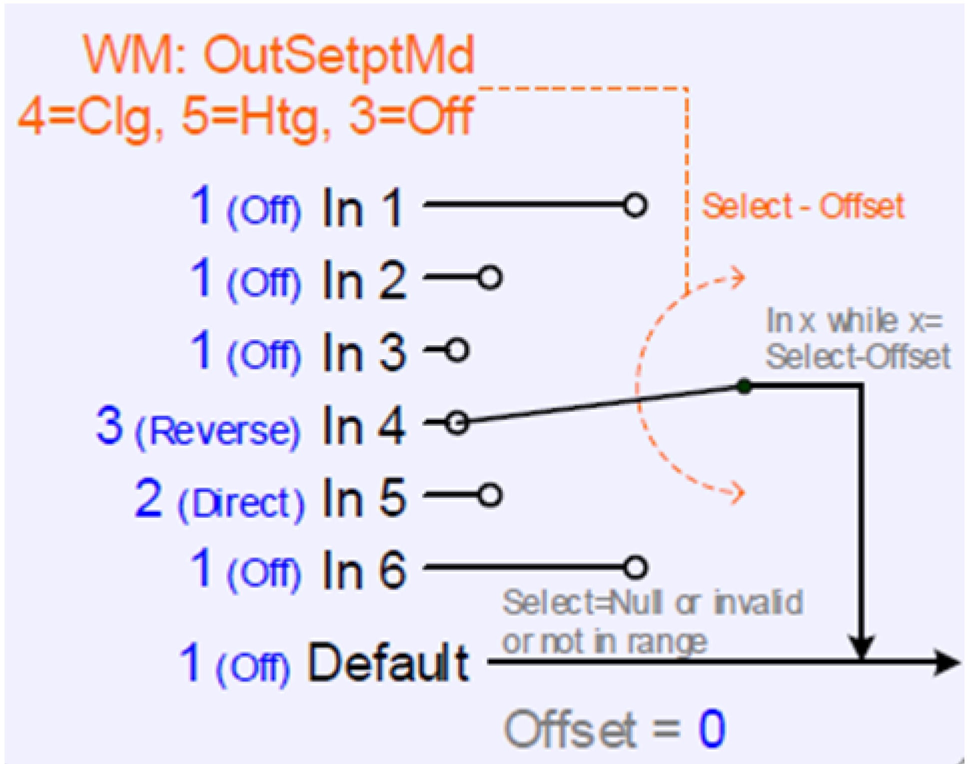


Fig. 244 Example for Enum Conversion

Output = Position determined by the value (X - **Offset**).

Input

Table 176 Inputs of Numeric Select

Input Name	Description
Select	This input is used to select which of the inputs In 1 - In 6 is used as output. If the value is < 1+ **Offset** or > 6 + **Offset**, the input Default is assigned to the output.
Default (In & Par)	If the **Select** < 1 + **Offset** or > 6 + **Offset**, **Out** = **Default**.
In 1 (In & Par)	This input is used as a output value, if **Select** = 1 + **Offset**.
In 2 (In & Par)	This input is used as a output value, if **Select** = 2 + **Offset**.
In 3 (In & Par)	This input is used as a output value, if **Select** = 3 + **Offset**.
In 4 (In & Par)	This input is used as a output value, if **Select** = 4 + **Offset**.
In 5 (In & Par)	This input is used as a output value, if **Select** = 5 + **Offset**.
In 6 (In & Par)	This input is used as a output value, if **Select** = 6 + **Offset**.

Output

Table 177 Output of Numeric Select

Output Name	Description
Out	If the **Select** < 1 + **Offset** or > 6 + **Offset**, **Out** = **Default**. If **Select** equals to 1 + **Offset**, then **Out** = **In 1**. If **Select** equals to 2 + **Offset**, then **Out** = **In 2**. If **Select** equals to 3 + **Offset**, then **Out** = **In 3**. If **Select** equals to 4 + **Offset**, then **Out** = **In 4**. If **Select** equals to 5 + **Offset**, then **Out** = **In 5**. If **Select** equals to 6 + **Offset**, then **Out** = **In 6**.

Parameter

Table 178 Parameters of Numeric Select

Parameter Name	Description
Offset	The **offset** is used for an extension if the number of inputs 1 - 6 is not sufficient.
Default Par (In & Par)	If **Default** is "null," **Default Par** is used as a parameter. In **Default Par** is set to "nan." In this case, it is treated like an invalid.
In1 Par (In & Par)	If **In1** is "null," **In1 Par** is used as a parameter. **In1 Par** is set to "nan." In this case, it is treated like an invalid.
In2 Par (In & Par)	If **In2** is "null," **In2 Par** is used as a parameter. **In2 Par** is set to "nan." In this case, it is treated like an invalid.
In3 Par (In & Par)	If **In3** is "null," **In3 Par** is used as a parameter. **In3 Par** is set to "nan." In this case, it is treated like an invalid.
In4 Par (In & Par)	If **In4** is "null," **In4 Par** is used as a parameter. **In4 Par** is set to "nan." In this case, it is treated like an invalid.
In5 Par (In & Par)	If **In5** is "null," **In5 Par** is used as a parameter. **In5 Pa**r is set to "nan." In this case, it is treated like an invalid.
In6 Par (In & Par)	If **In6** is "null," **In6 Par** is used as a parameter. **In6 Par** is set to "nan." In this case, it is treated like an invalid.
Out Save	• **Master Sync Enabled:** If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Examples

Example 1: NumericSelect

If **Select** value = In1 to In6, In1 to In6 is set to output.

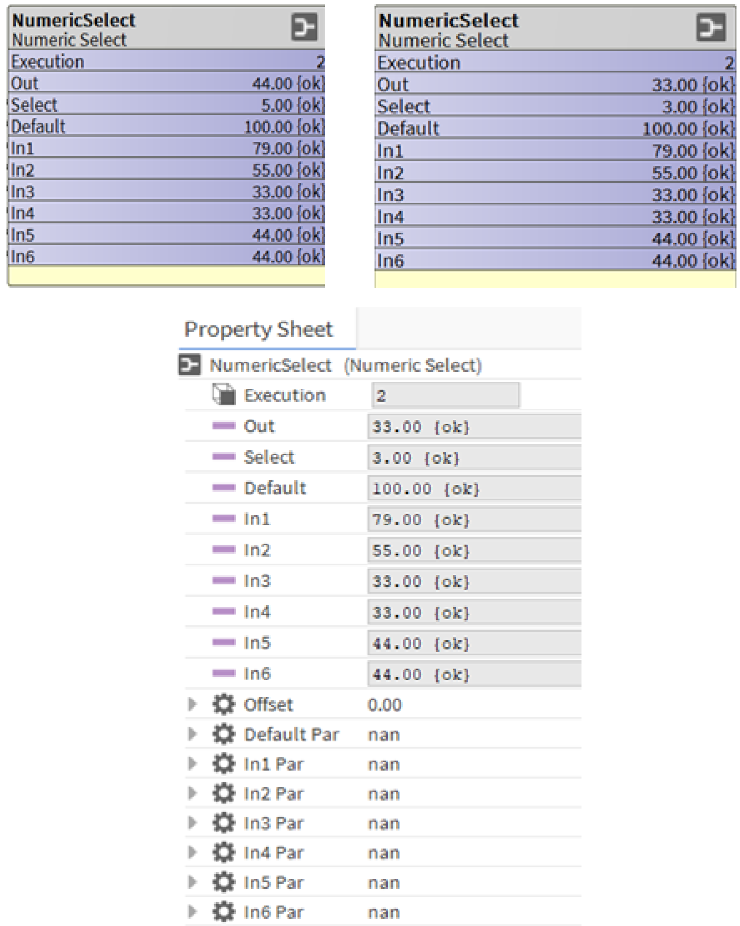


Fig. 245 NumericSelect Function Block

Example 2: NumericSelect

If the **Select** value is “invalid,” the default value is set to output.

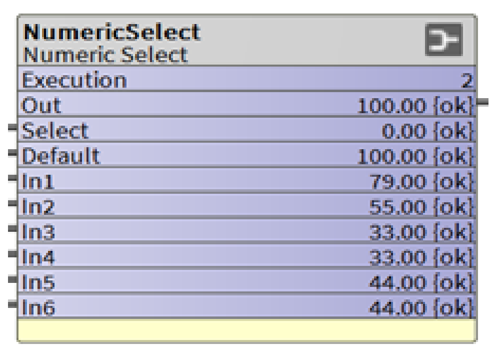


Fig. 246 NumericSelect Function Block

Valid Select Prio

The Valid Select Prio function block looks for a valid value in four inputs. If an input contains a valid value (less than or greater than zero), the value is output on the "Out" line. If any of the numerous inputs has a valid value, the value from the input with the highest priority is output on "Out" (A has the highest priority and D has the lowest priority).

If no input has a valid value, the value from the Default input is output.

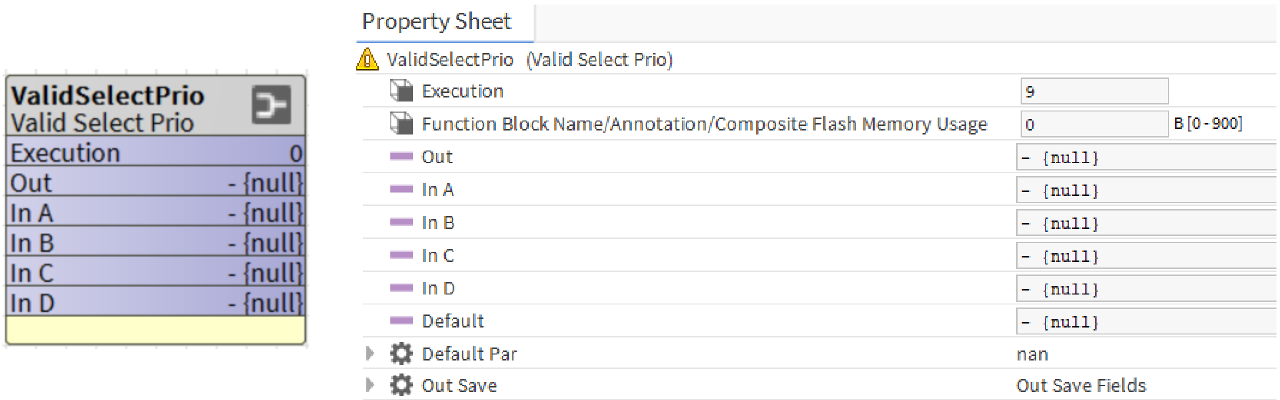


Fig. 247 ValidSelectPrio Function Block and Property Sheet

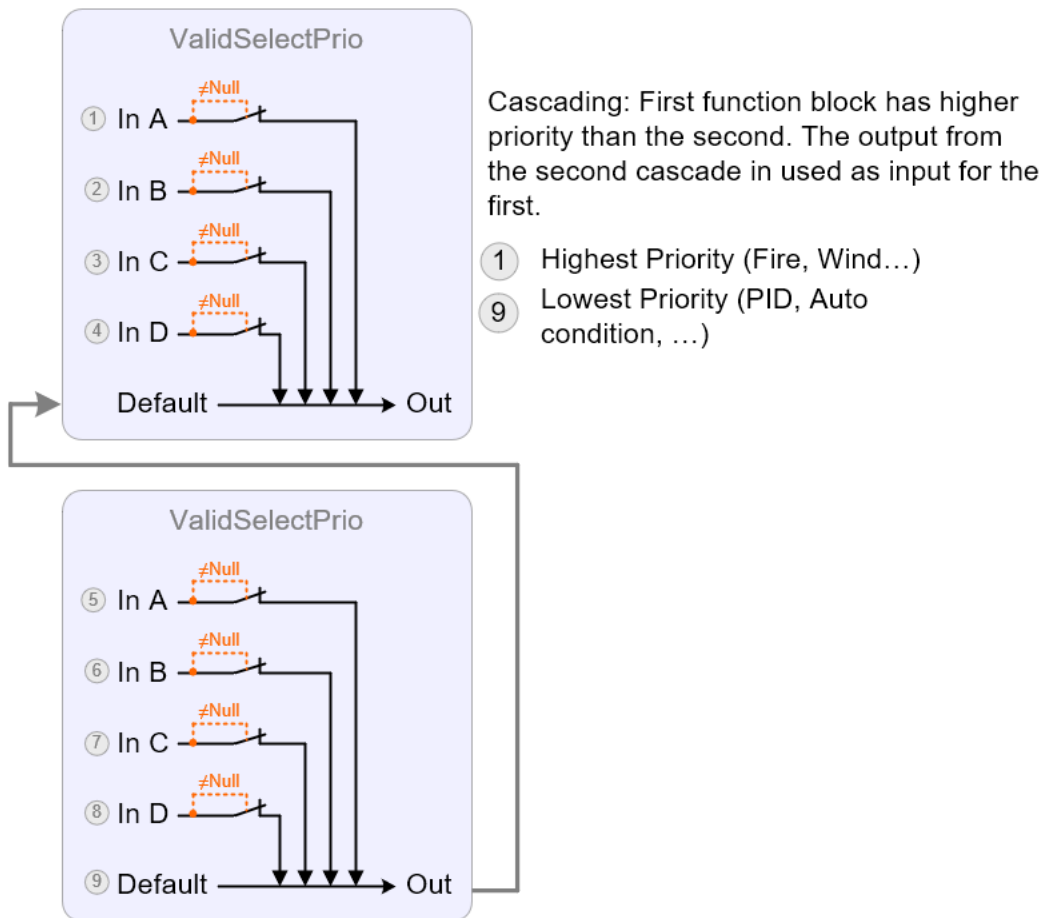


Fig. 248 ValidSelectPrio Logic Diagram

Input

Table 179 Inputs of Valid Select Prio

Input Name	Description
In A	If **In A** has a valid value (less or greater than "null"), **Out** = **In A**. **In A** has the highest priority.
In B	If **In A** has an invalid value ("null"), but **In B** has a valid value (less or greater than "null"), **Out** = **In B**.
In C	If **In A** and **In B** have an invalid value ("null"), but **In C** has a valid value (less or greater than "null"), **Out** = **In C**.
In D	If **In A**, **In B** and **In C** have an invalid value ("null"), but **In D** has a valid value (less or greater than "null"), **Out** = **In D**.
Default (In & Par)	If all Inputs have an invalid value ("null"), **Out** = **Default**.

Output

Fig. 249 Output of Valid Select Prio

Output Name	Description
Out	**Output:** Highest priority input that is valid.

Parameter

Table 180 Parameters of Valid Select Prio

Parameter Name	Description
Default Par (In & Par)	If **Default** is "null," **Default Par** is used as a parameter. In **Default Par** is set to "nan". In this case, it is treated as invalid.
Out Save	• **Master Sync Enabled:** If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Examples

Example 1: ValidSelectPrio

If **In A** to **In D** = valid, the corresponding input is set to output.

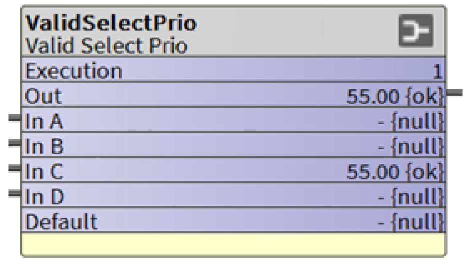


Fig. 250 Valid Select Prio Function Block

Example 2: ValidSelectPrio

If more than one input is valid, **In A** has high priority, and **In D** will have low priority. Accordingly, the output is set.

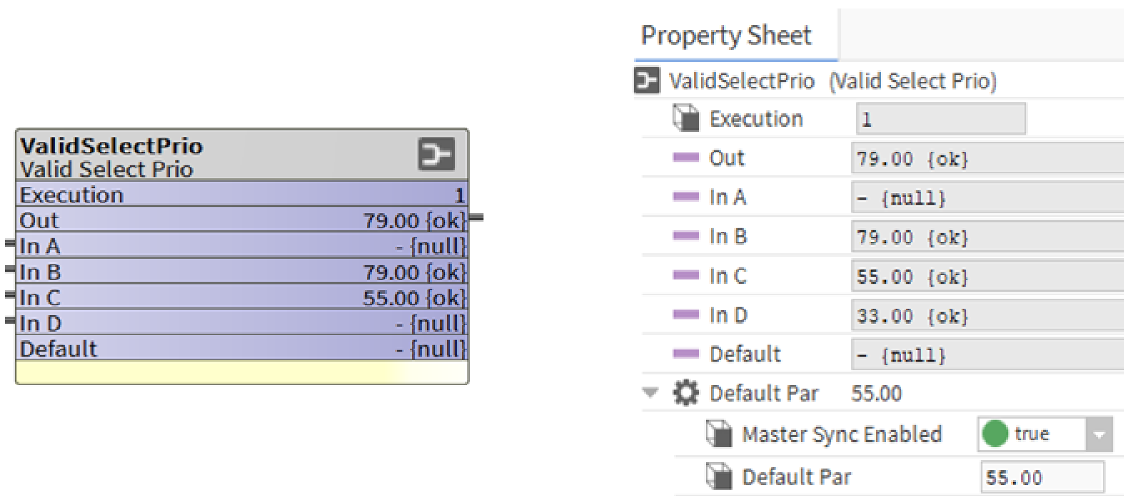


Fig. 251 Valid Select Prio

Example 3: ValidSelectPrio

If all the values are "invalid," **Default** (input or parameter) is set to output.

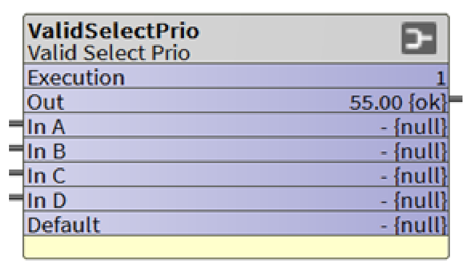


Fig. 252 ValidSelectPrio Function Block

Valid Select Multi Prio

The Valid Select Multi Prio function returns the highest priority input that is not invalid. This block is similar to ValidSelectPrio, but it has two numerical lines.

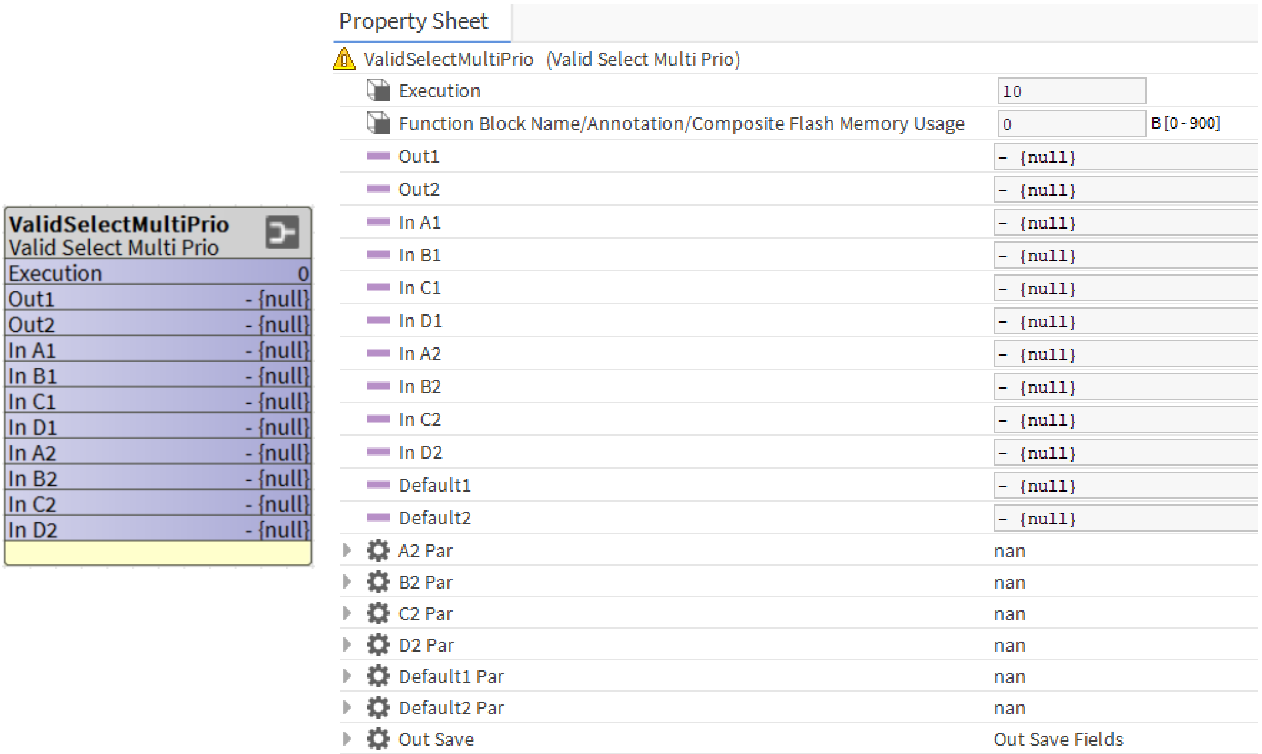


Fig. 253 ValidSelectMultiPrio Function Block and Property Sheet

The first function block has a higher priority than the second. The output from the second cascade is used as input for the first. In the below Figure, **In A1** has the highest priority than the default value 9 (PID and auto condition).

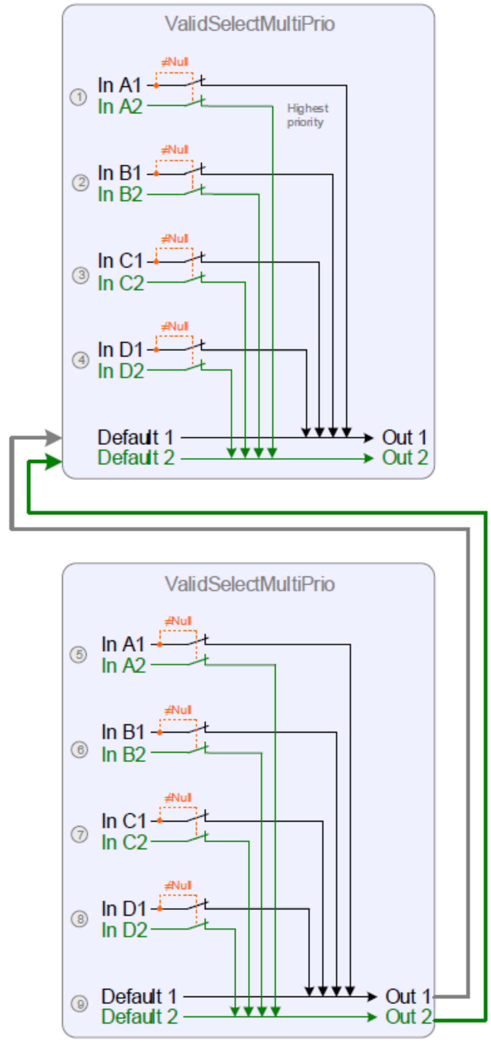


Fig. 254 ValidSelectMultiPrio Logic Diagram

Input

Table 181 Inputs of Valid Select Multi Prio

Input Name	Description
In A1 (In & Par)	If **In A1** has a valid value (less or greater than zero), **Out 1** = **In A1**. **In A1** has the highest priority.
In B1 (In & Par)	If **In A1** has an invalid value (“null”), but **In B1** has a valid value (less or greater than “null”), **Out** = **In B1**.
In C1 (In & Par)	If **In A1** and **In B1** have an invalid value (“null”), but **In C1** has a valid value (less or greater than “null”), **Out1** = **In C1**.

Table 181 Inputs of Valid Select Multi Prio (Continued)

Input Name	Description
In D1 (In & Par)	If **In A1**, **In B1** and **In C1** have an invalid value (“null”), but **In D1** has a valid value (less or greater than “null”), **Out** = **In D1**.
In A2 (In & Par)	If **In A1** is output on **Out 1**, **In A2** is output on **Out 2**.
In B2 (In & Par)	If **In B1** is output on **Out 1**, **In B2** is output on **Out 2**.
In C2 (In & Par)	If **In C1** is output on **Out 1**, **In C2** is output on **Out 2**.
In D2 (In & Par)	If **In D1** is output on **Out 1**, **In D2** is output on **Out 2**.
Default1 (In & Par)	If all Inputs **In A1** - **In D1** have invalid values (“null”), **Out 1** = **Default1**.
Default2 (In & Par)	If all Inputs **In A2** - **In D2** have invalid values (“null”), **Out 2** = **Default2**.

Output

Table 182 Outputs of Valid Select Multi Prio

Output Name	Description
Out 1	**Output:** Highest priority input that is not invalid (**In A1**/**In B1**/**In C1**/**In D1**/**Default1**).
Out 2	**Output:** Highest priority input that is not invalid (**In A2**/**In B2**/**In C2**/**In D2**/**Default2**).

Parameter

Table 183 Parameters of Valid Select Multi Prio

Parameter Name	Description
In A2 Par (In & Par)	If **In A2** is “null,” **In A2 Par** is used as a parameter. **In A2 Par** is set to “nan.” In this case, it is treated like an invalid.
In B2 Par (In & Par)	If **In B2** is “null,” **In B2 Par** is used as a parameter. **In B2 Par** is set to “nan.” In this case, it is treated like an invalid.
In C2 Par (In & Par)	If **In C2** is “null,” **In C2 Par** is used as a parameter. **In C2 Par** is set to “nan.” In this case, it is treated like an invalid.
In D2 Par (In & Par)	If **In D2** is “null,” **In D2 Par** is used as a parameter. **In D2 Par** is set to “nan.” In this case, it is treated like an invalid.
Default1 Par (In & Par)	If **Default1** is “null,” **Default1 Par** is used as a parameter. In **Default1 Par** is set to “nan,” In this case, it is treated like an invalid.
Default2 Par (In & Par)	If **Default2** is “null,” **Default2 Par** is used as a parameter. In **Default2 Par** is set to “nan.” In this case, it is treated like an invalid.

Table 183 Parameters of Valid Select Multi Prio (Continued)

Parameter Name	Description
Out Save	• **Master Sync Enabled:** If you set it to true, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out1:** To enable or disable the **Out1** feature. • **Out2:** To enable or disable the **Out2** feature.

Examples

Example 1: ValidSelectMultiPrio

For priority selection, the highest priority selector value is the same valid value that is selected as output.

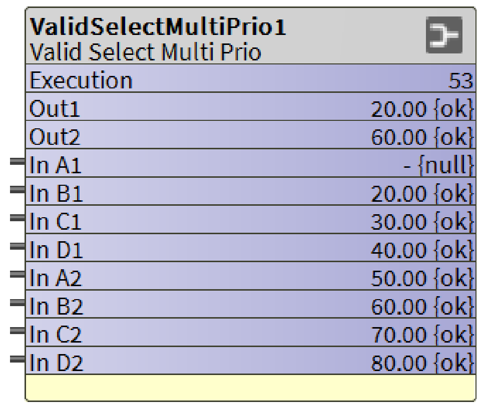


Fig. 255 ValidSelectMultiPrio – Priority Selection

Example 2: ValidSelectMultiPrio

Default selection, if all the priority selector values are “nan.”

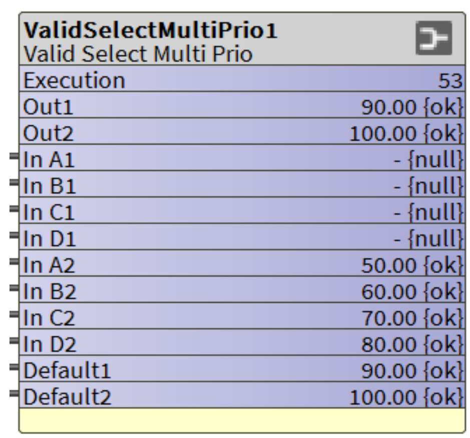


Fig. 256 ValidSelectMultiPrio – Default Selection

Numeric Switch

The Numeric Switch function block selects one of the eight boolean outputs that is set to "true." If the numeric input uses a wrong value, all boolean outputs are "false." The function can be extended to more inputs via an offset if the eight outputs are not sufficient.

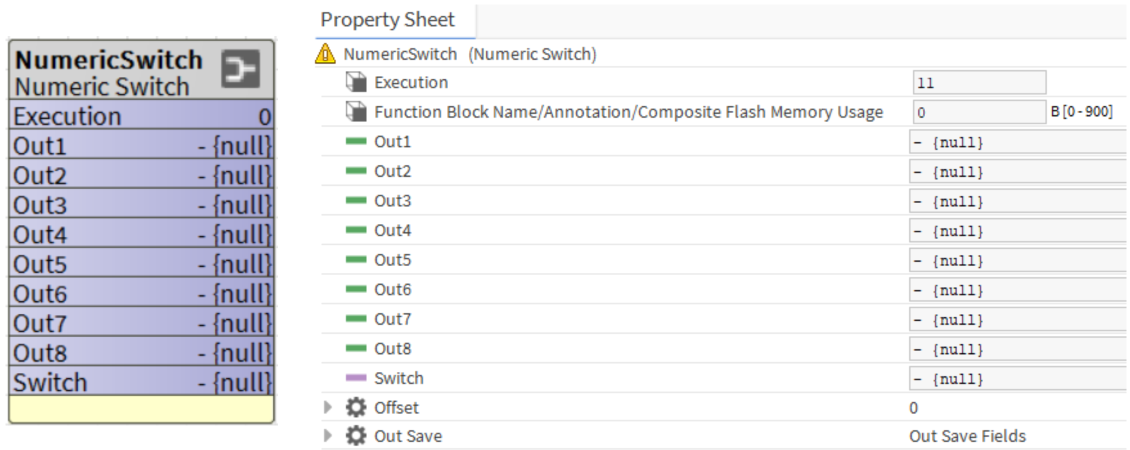


Fig. 257 NumericSwitch Function Block and Property Sheet

The valid range of the input minus the offset is 1 through 8. The output X (1 through 8) is true. If **Switch – Offset** = **X**, else, it is "false."

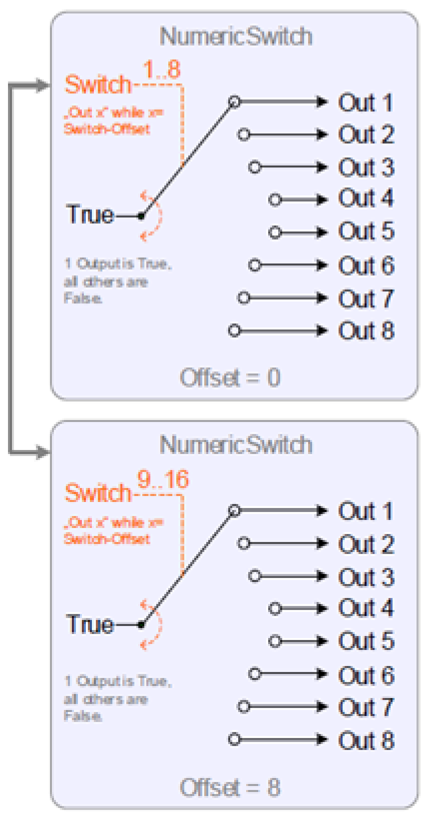


Fig. 258 NumericSwitch Logic Diagram

Input

Table 184 Input of Numeric Switch

Input Name	Description
Switch	This input selects the output that is set to “true” based on the value. A switch can have any value, even negative values, because **Offset** can bring the **Switch** value into a valid range. **Example:** With **Offset** = 0, if **Switch** = 8, **Out 8** = “true” and all other outputs are “false.”

Output

Table 185 Outputs of Numeric Switch

Output Name	Description
Out1 – Out8	The output 1 through 8 is "true", if (input +offset) = X, otherwise it is "false". If the output is negated, then it is also negated from the value determined by the function block logic.

Parameter

Table 186 Parameters of Numeric Switch

Parameter Name	Description
Offset	It is used for an extension, if the number of outputs 1 - 8 is not sufficient or if the range of **Switch** is not from 1 - 8. **Example:** The **Switch** range is from 16 -24 -> **Offset** = 15.
Out Save	• **Master Sync Enabled:** If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out1:** To enable or disable the **Out1** feature. • **Out2:** To enable or disable the **Out2** feature. • **Out3:** To enable or disable the **Out3** feature. • **Out4:** To enable or disable the **Out4** feature. • **Out5:** To enable or disable the **Out5** feature. • **Out6:** To enable or disable the **Out6** feature. • **Out7:** To enable or disable the **Out7** feature. • **Out8:** To enable or disable the **Out8** feature.

Examples

Example 1: NumericSwitch

Numeric switch selector.

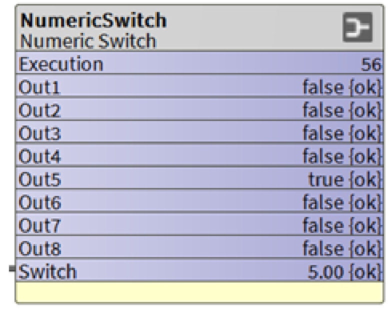


Fig. 259 Numeric Switch Selector

Example 2: NumericSwitch

Numeric switch selector along with offset.

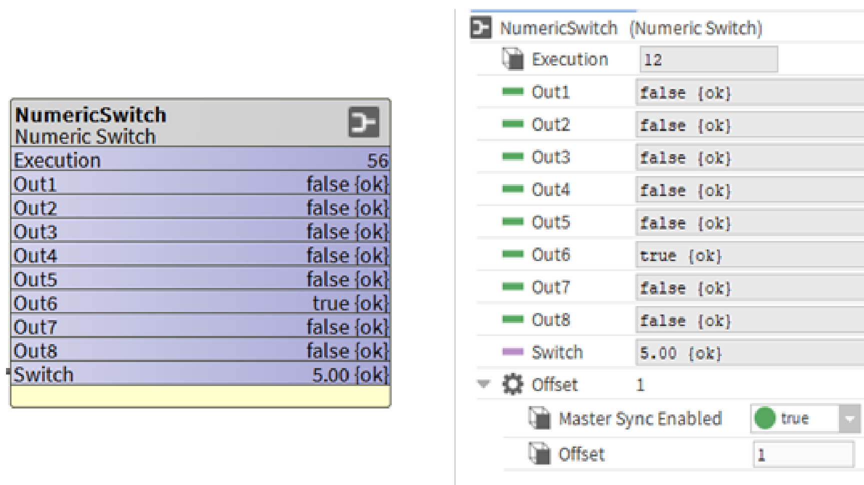


Fig. 260 Numeric Switch Selector with Offset along with the property sheet

CHAPTER

10 LOGIC FUNCTION BLOCKS

The following Logic function blocks are available in the honIrmControl Palette and can be configured and used to create the required application logic:

- And
- Or
- Xor
- Not
- Rs Flip Flop
- Sr Flip Flop
- Trigger

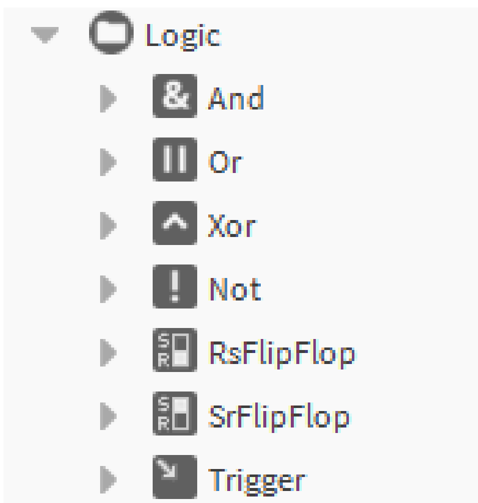


Fig. 261 Logic Function Blocks

And

The AND function block has six-inputs AND function . Each input may be individually inverted (NOT). The AND output becomes "true" if all inputs are "true."

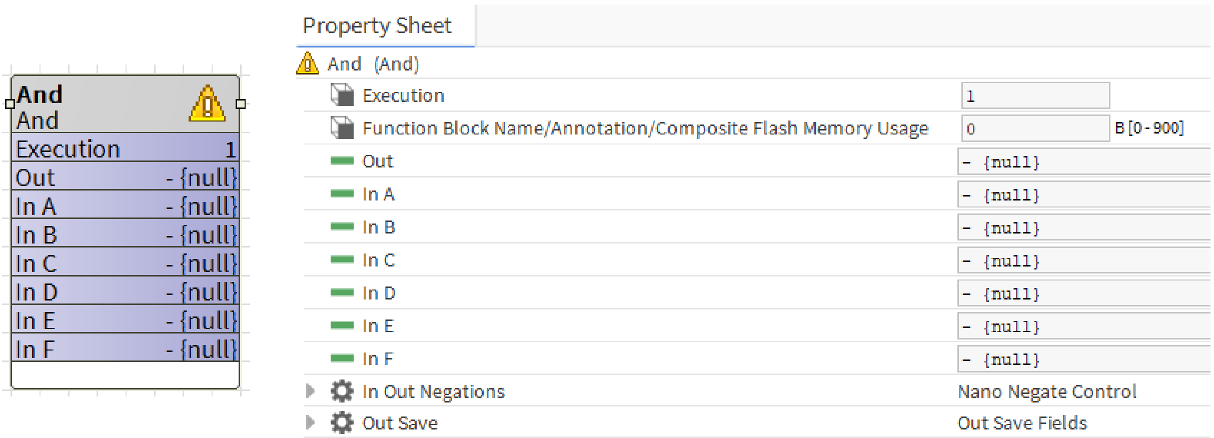


Fig. 262 AND Function Block and Property Sheet

In AND function block, inputs with "null" are ignored, which means if either of the inputs is not equal to 0.00, the output is equal to the input.

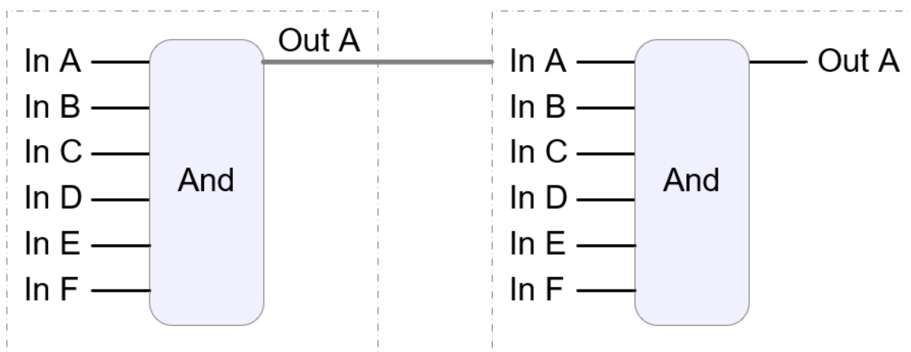


Fig. 263 AND Logic Diagram

Unconnected or invalid inputs default to "true," without negation, not affecting the result.

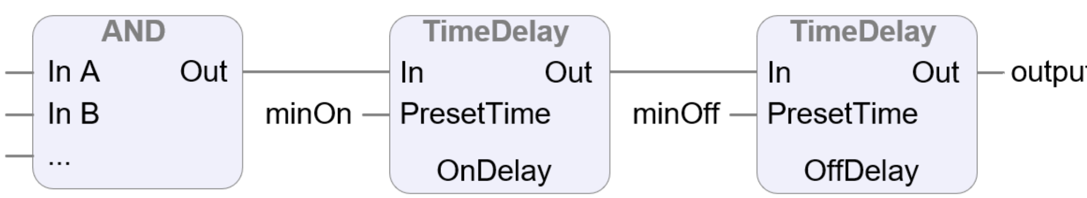


Fig. 264 AND with trueDelay and falseDelay

Input

Table 187 Input of And

Input Name	Description
In A - In F	These are 32-bit floating point 6 input slots.

Output

Table 188 Output of And

Output Name	Description
Out	**Output:** AND or NAND (inputs). When the output is negative, the AND function block behaves like a NAND function block.

Parameter

Table 189 Parameters of And

Parameter Name	Description
In Out Negations	• If this option is set to “true,” it acts as a NAND function block. • If it is “false,” it acts as an AND function block.
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out**: To enable or disable the **Out** feature.

The Below table shows the basic operation of AND (“Out” negation is false).

Table 190 Basic Operation of AND

Input (x)	Input (y)	Output
0 (false)	0 (false)	0 (false)
0 (false)	1 (true)	0 (false)
1 (true)	0 (false)	0 (false)
1 (true)	1 (true)	1 (true)

Below table shows the basic operation of NAND (“Out negation” is true).

Table 191 Basic Operation of NAND

Input (x)	Input (y)	Output
0 (false)	0 (false)	1 (true)
0 (false)	1 (true)	1 (true)
1 (true)	0 (false)	1 (true)
1 (true)	1 (true)	0 (false)

Examples

Example 1: And

If any one of **In A** to **In F** = “false,” the output is “false.”

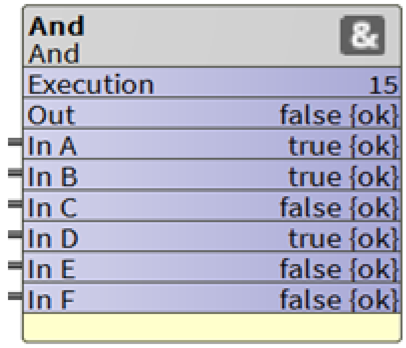


Fig. 265 And Function Block

Example 2: And

If all **In A** to **In F** are “true,” the output is “true.”

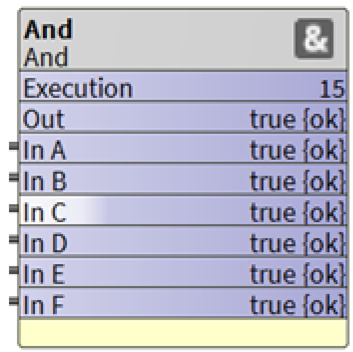


Fig. 266 AND Function Block

Example 3: And

Not enable for **In A** and **In C**.

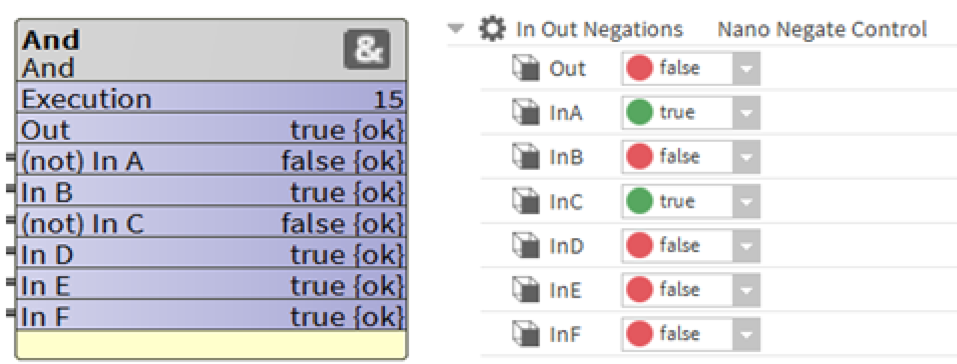


Fig. 267 AND Function Block and Property Sheet

Or

The OR function block has six inputs OR functions. Each input can be inverted separately (NOT). If at least one of the inputs is "true," the OR output is "true."

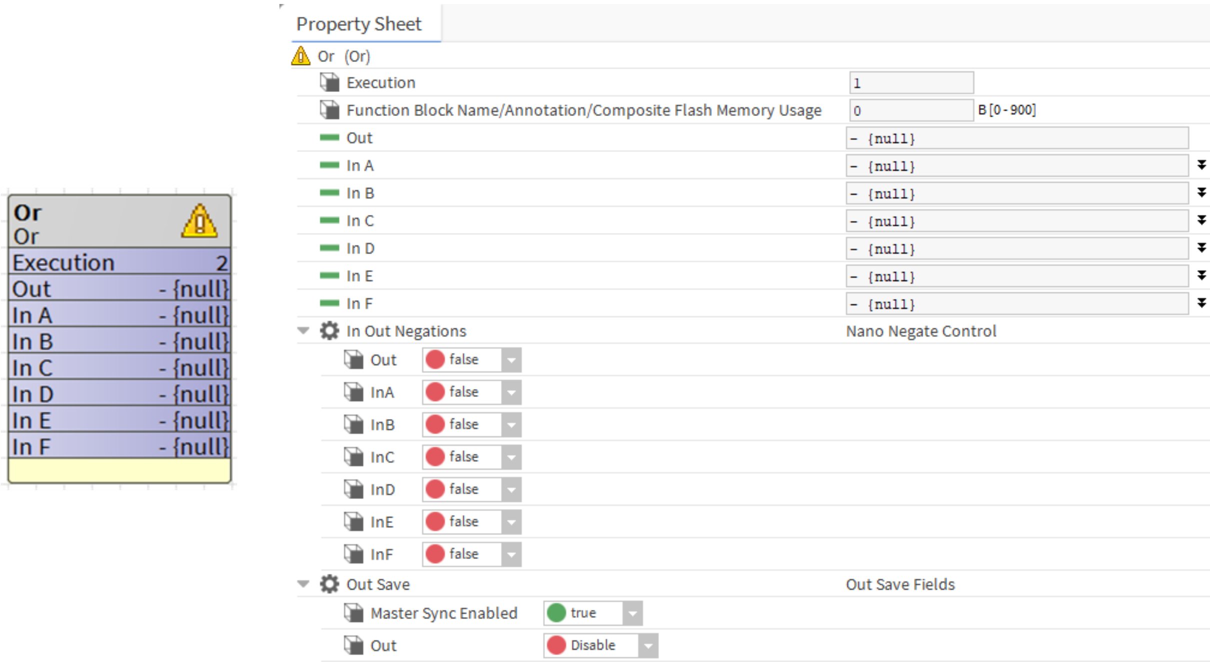


Fig. 268 OR Function Block and Property Sheet

In this function block, inputs with "null" are ignored, which means if either of the inputs is not equal to 0.00, the output is equal to the input.

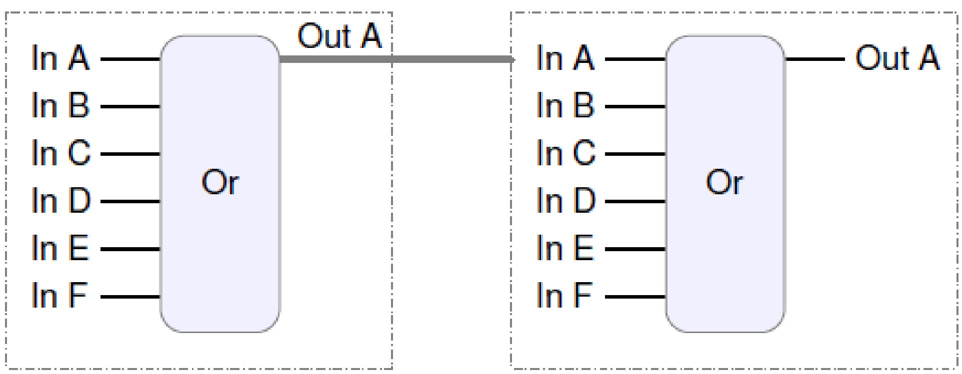


Fig. 269 OR Logic Diagram

Unconnected or invalid inputs default to "true," without negation, not affecting the result.

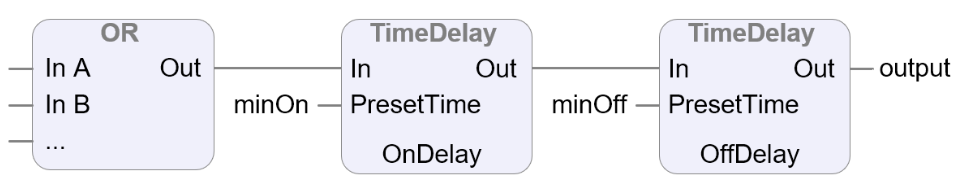


Fig. 270 OR with trueDelay and falseDelay

Input

Table 192 Input of OR

Input Name	Description
In A - In F	These are boolean point 6 input slots.

Output

Table 193 Output of OR

Output Name	Description
Out	Output is the **In A** multiplied by **In B**.

Parameter

Table 194 Parameters of OR

Parameter Name	Description
In Out Negations	• If this option is set to "true," it acts as an AND function block. It negates inputs and outputs. • If it is "false," it acts as an Or function block.
Out Save	• **Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out**: To enable or disable the **Out** feature.

Below table shows the basic operation of OR ("In Out Negations" is "true").

Table 195 Basic Operation of OR

Input (x)	Input (y)	Output
0 (false)	0 (false)	0 (false)
0 (false)	1 (true)	1 (true)
1 (true)	0 (false)	1 (true)
1 (true)	1 (true)	1 (true)

When the "In Out Negations" is set to "true," OR block turns into an AND block as it negates inputs and outputs.

Examples

Example 1: Or

If any one of **In A** to **In F** - “true,” the output is “true.” If all **In A** to **In F** are “false,” the output is “false.”

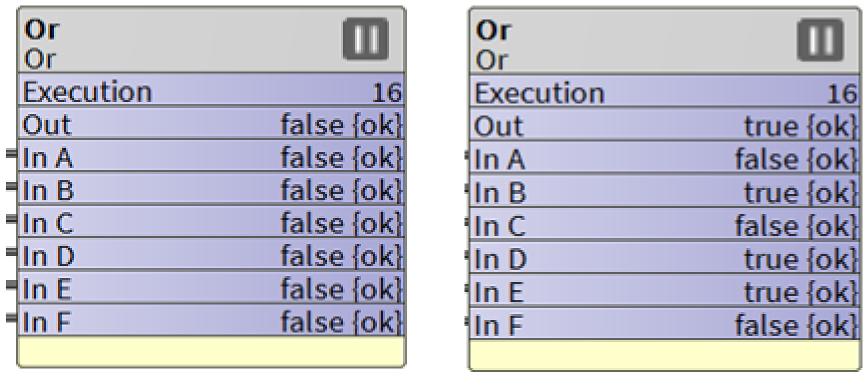


Fig. 271 Or Function Block

Example 2: Or

Negate IN/OUT slots.

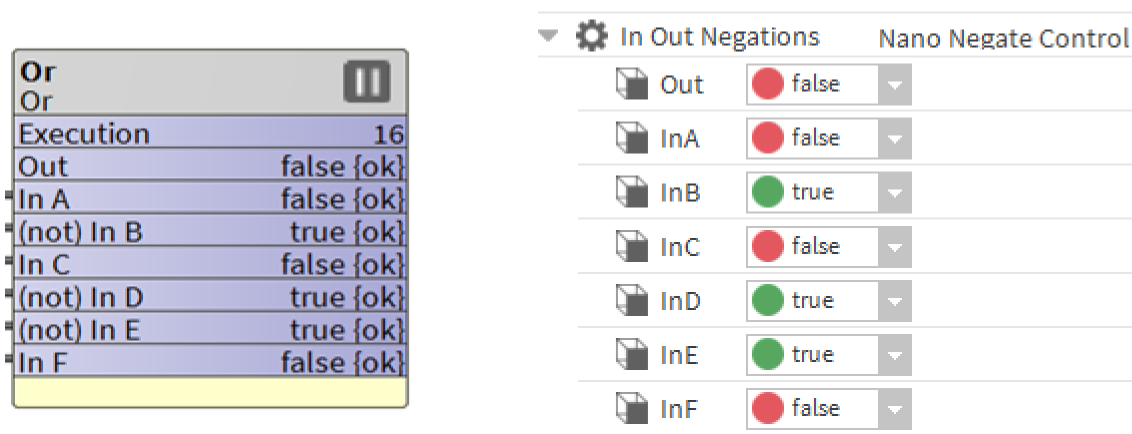


Fig. 272 Or Function Block and Property Sheet

Xor

This XOR function block has six input XOR function. Each input may be individually inverted (NOT). The XOR output becomes "true" if exactly one input is "true."

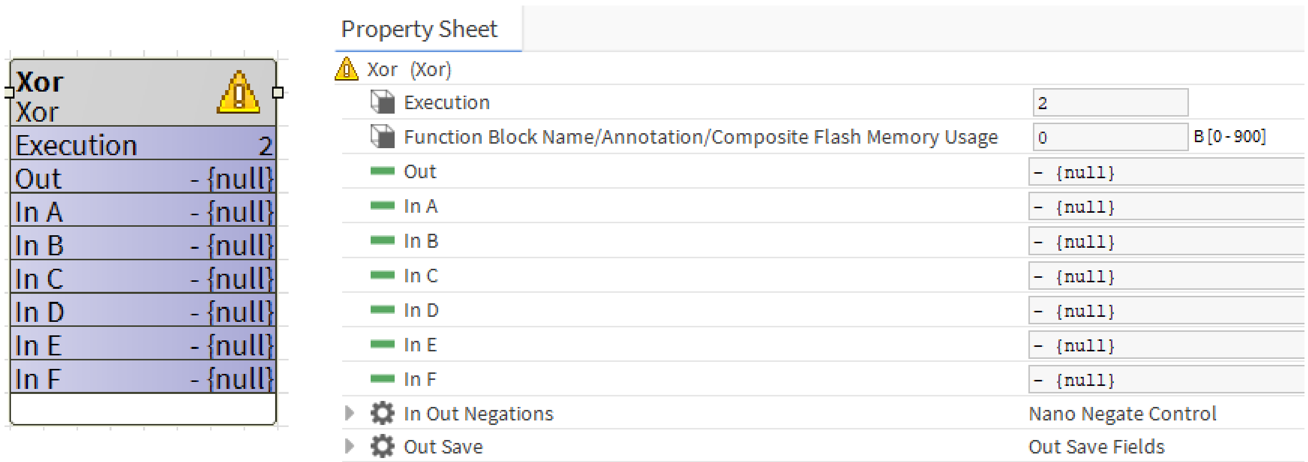


Fig. 273 XOR Function Block and Property Sheet

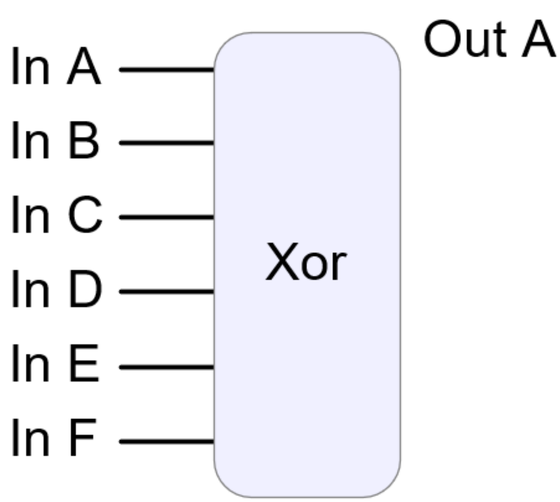


Fig. 274 XOR Logic Diagram

Unconnected or invalid inputs default to “true,” without negation, not affecting the result.

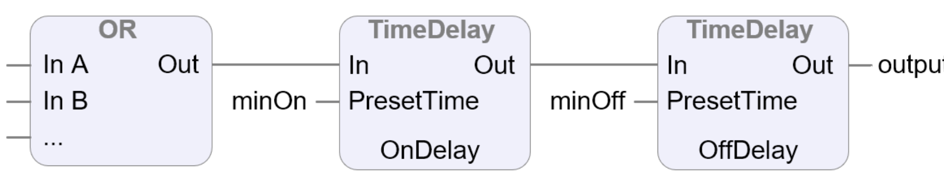


Fig. 275 XOR with trueDelay and falseDelay

Input

Table 196 Input of XOR

Input Name	Description
In A - In F	These are Boolean point 6 input slots.

Output

Table 197 Output of XOR

Output Name	Description
Out	**Out:** 1 if an odd number of inputs is 1.

Parameter

Table 198 Parameters of XOR

Parameter Name	Description
In Out Negations	• If this option (Nano Negate Control) is set to “true,” it acts as a NOT function block input or output. It negates inputs and outputs. • If it is “false,” it acts as an XOR function block.
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out**: To enable or disable the **Out** feature.

The below table shows the basic operation of XOR.

Table 199 Basic Operation of XOR

Input (A)	Input (B)	Input (C)	Output
0 (false)	0 (false)	0 (false)	0 (false)
0 (false)	0 (false)	1 (true)	1 (true)
0 (false)	1 (true)	0 (false)	1 (true)
0 (false)	1 (true)	1 (true)	0 (false)
1 (true)	0 (false)	0 (false)	1 (true)
1 (true)	0 (false)	1 (true)	0 (false)
1 (true)	1 (true)	0 (false)	0 (false)
1 (true)	1 (true)	1 (true)	0 (false)

Examples

Example 1: Xor

If only one of **In A** - **In F** is “true,” the output is set to “true.” If not, “false.”

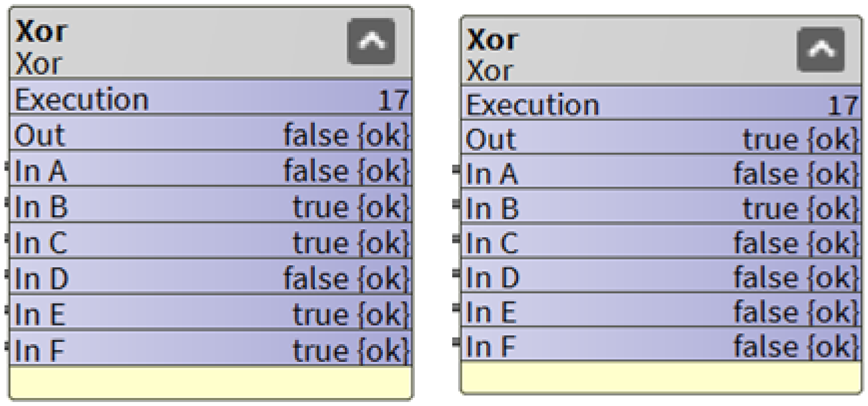


Fig. 276 Xor Function Block

Example 2: Xor

Negate IN/OUT slots.

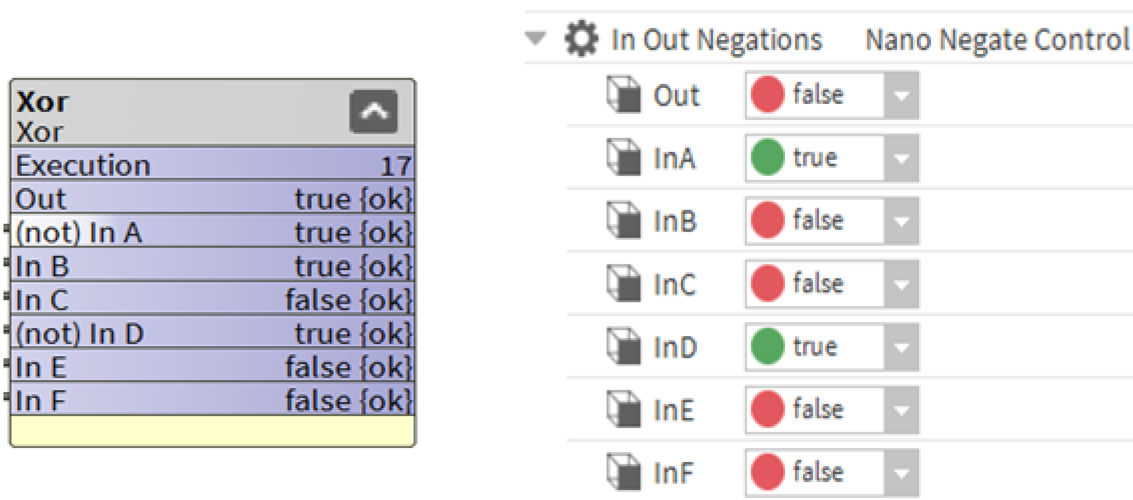


Fig. 277 Xor Function Block and Property Sheet

Not

In the NOT function block, “true” becomes “false” and vice versa after performing the logic.

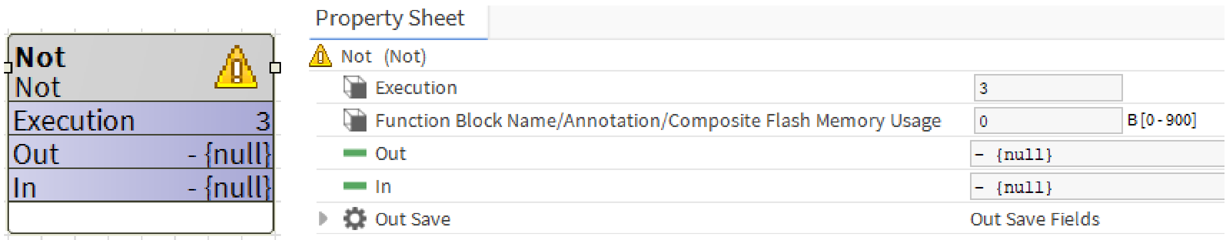


Fig. 278 Not Function Block and Property Sheet

Input

Table 200 Input of NOT

Input Name	Description
In	This is a Boolean point input slot.

Output

Table 201 Output of NOT

Output Name	Description
Out	**Output:** “true” or “false” (opposite value of In).

Parameter

Table 202 Parameter of NOT

Parameter Name	Description
Out Save	**Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. **Out**: To enable or disable the **Out** feature.

Examples

Example 1: Not

If **In** = “true”, output = “false.”

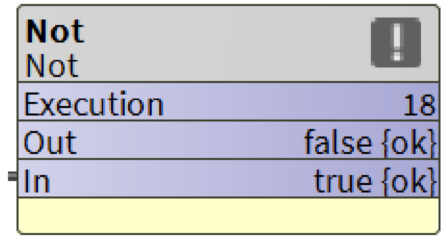


Fig. 279 Not Function Block

Example 2: Not

If **In** = “false”, output = “true.”

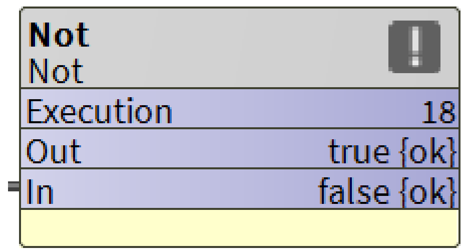


Fig. 280 Not Function Block

Rs Flip Flop

The Rs Flip Flop function block prioritizes the reset input if both the inputs are “true.” Rs Flip Flop has a one-bit functional block.

Rs Flip Flop function block has two inputs:

- **Set**: Sets the device **Out** = “1,” labeled as “S.”
- **Reset**: Resets the device **Out** = “0,” labeled as “R.”

The RS abbreviation stands for Set or Reset. The flip-flop is reset to its original state using the Reset input, and the output is either logic level "1" or logic level "0." It is determined by the flip-Set flop's or Reset condition. The term "flip flop" refers to the ability to be "FLIPPED" into one logic state and "FLOPPED" back into another.

The basic NOR gate and RS Flip Flop circuit are used to store the data and thus provide feedback from both of its outputs again back to its inputs.

The RSFlipFlop has three inputs Set, Reset, and Out (current output) that relate to its current state.

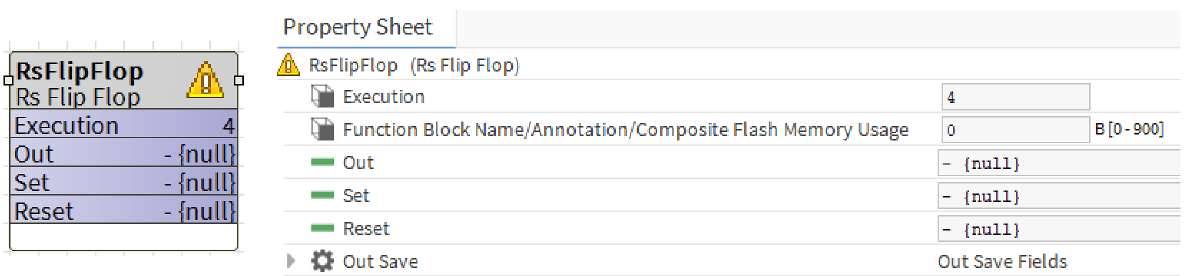


Fig. 281 RsFlipFlop Function Block and Property Sheet

Inputs with the value “false” or “null” are not evaluated. The output keeps on the last value.

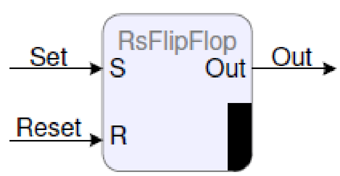


Fig. 282 RsFlipFlop Logic Diagram

Input

Table 203 Inputs of Rs Flip Flop

Input Name	Description
Set	This is a boolean point input slot.
Reset	This is a boolean point input slot. If both inputs are “true,” it has a priority.

Output

Table 204 Output of Rs Flip Flop

Output Name	Description
Out	**Output:** “true” or “false” based on the input and **reset** value.

Truth table

Table 205 Truth Table

Set	Reset	Output
false	false	No change or remains in the previous state
false	true	false
true	false	true
true	true	false

Parameter

Table 206 Parameter of Rs Flip Flop

Parameter Name	Description
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out**: To enable or disable the **Out** feature.

Examples

The five conditions of Rs FlipFlop are as follows:

Example 1: RsFlipFlop

If **Set** and **Reset** are “true,” **Out** is “false.”

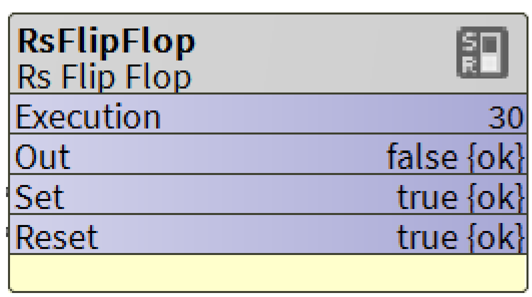


Fig. 283 Set and Reset “true,” Out “false”

Example 2: RsFlipFlop

If **Set** and **Reset** are set to “false,” **Out** will retain the last state.

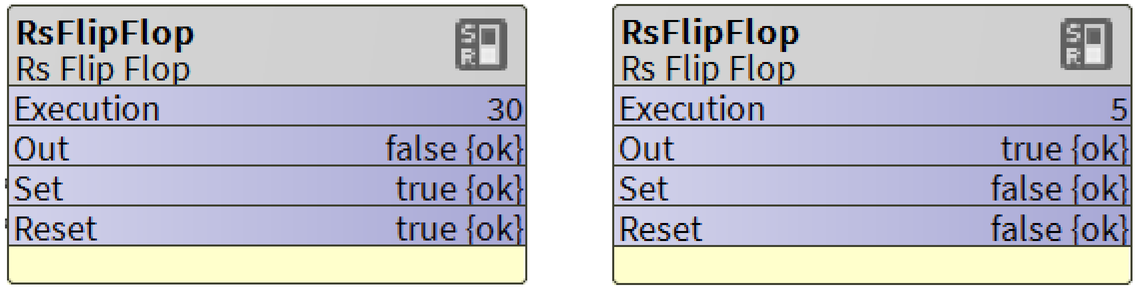


Fig. 284 Set and Reset “false,” Out last state

Example 3: RsFlipFlop

If **Set** = “true,” **Reset** = “false,” **Out** = ”true.”

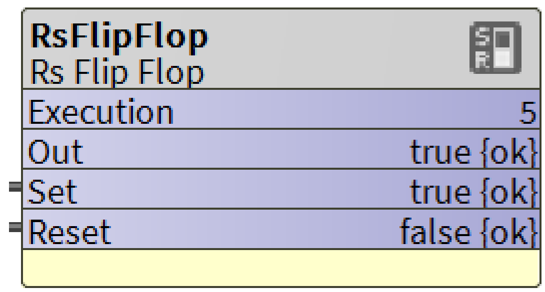


Fig. 285 Set “true,” Reset “false” and Out “false”

Example 4: RsFlipFlop

If **Set** = “false” and **Reset** = “true,” **Out** = “false.”

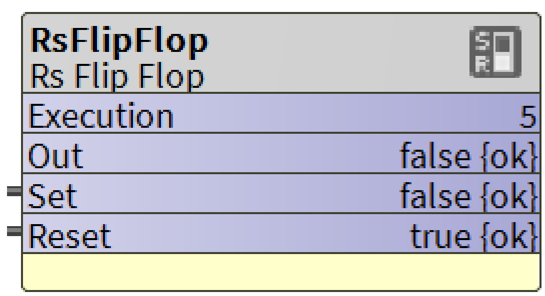


Fig. 286 Set “false,” Reset “true” and Out “false”

Sr Flip Flop

The Sr Flip Flop function block prioritizes the set input if both the inputs are “true.” Sr Flip Flop is a one-bit functional block.

Sr Flip Flop function block has two inputs:

- **Set**: Sets the device **Out** = “1,” labeled as “S.”
- **Reset**: Resets the device **Out** = “0,” labeled as “R.”

The RS stands for Set or Reset. With the help of the Reset input, the flip-flop is reset to its original state, and the output is either logic level "1" or logic level "0." It is determined by the flip-Set flop's or Reset state. It can be "FLIPPED" into one logic state or "FLOPPED" back into another.

The basic NAND gate SR Flip Flop circuit is used to store the data and thus provides feedback from both of its outputs back to its inputs. The Sr Flip Flop has three inputs that relate to its current state: Set, Reset, and Out (current output).

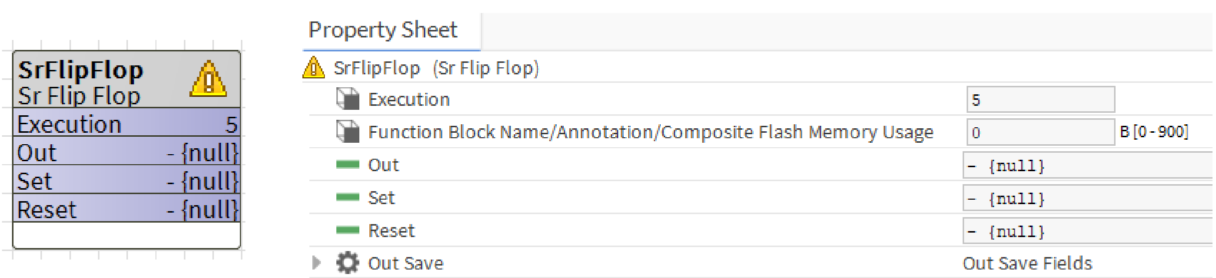


Fig. 287 SrFlipFlop Function Block and Property Sheet

Inputs with the value “true” or “null” are not evaluated. The output keeps on the last value.

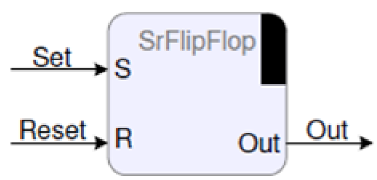


Fig. 288 SrFlipFlop Logic Diagram

Truth table

Table 207 Truth Table

Set	Reset	Output
false	false	No change or remains in the previous state
false	true	false
true	false	true
true	true	true

Input

Table 208 Inputs of SrFlipFlop

Input Name	Description
Set	This is a Boolean point input slot. It has a priority if both inputs are true.
Reset	This is a Boolean point input slot.

Output

Table 209 Output of SrFlipFlop

Output Name	Description
Out	**Output:** “true” or “false” depending on the input and reset value.

Parameter

Table 210 Parameter of SrFlipFlop

Parameter Name	Description
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out**: To enable or disable the **Out** feature.

Examples

Example 1: SrFlipFlop

If **Set** and **Reset** are set to “true,” **Out** is set to “true.”

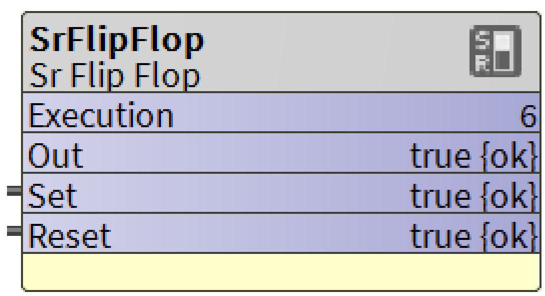


Fig. 289 Set and Reset “true,” Out “false”

Example 2: SrFlipFlop

If **Set** and **Reset** are set to “false,” **Out** will retain the last state.

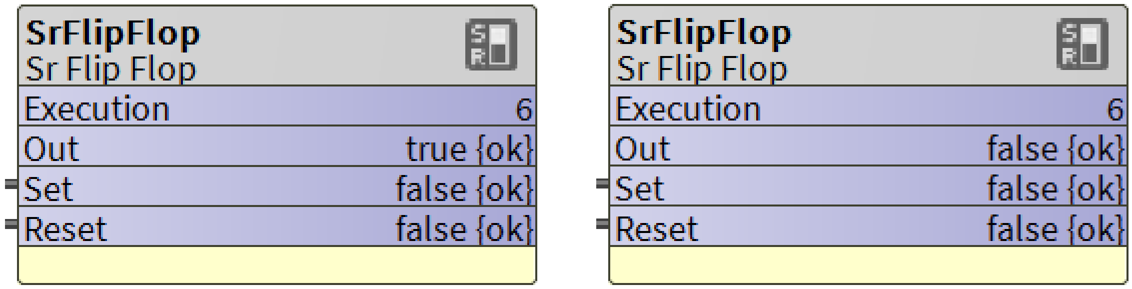


Fig. 290 Set and Reset “false,” Out last state

Example 3: SrFlipFlop

If **Set** = “true” and **Reset** = “false,” **Out** = “true.”

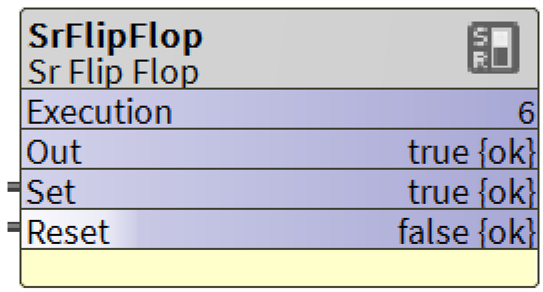


Fig. 291 Set “true” and Reset “false,” Out “true”

Example 4: SrFlipFlop

If **Set** = “false” and **Reset** = “true,” **Out** = “false.”

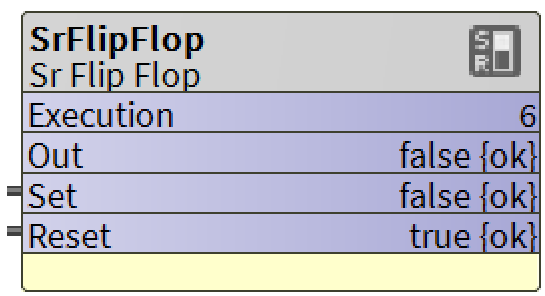


Fig. 292 Set “false” and Reset “true,” Out “false”

Trigger

The function block checks whether the input “In” has a change between True and False. If there is a change, a True for one DDC cycle occurs at the output “Out”.

A typical example is the detection, if a presence detector goes to occupied, the wind speed changes to high, the brightness changes to dark, the room temperature changes below the frost protection. Then you can trigger the logic in the application with the pulse at the output “Out”.

Out can be directly connected to a FlipFlop function block to Set or Reset the FlipFlop..

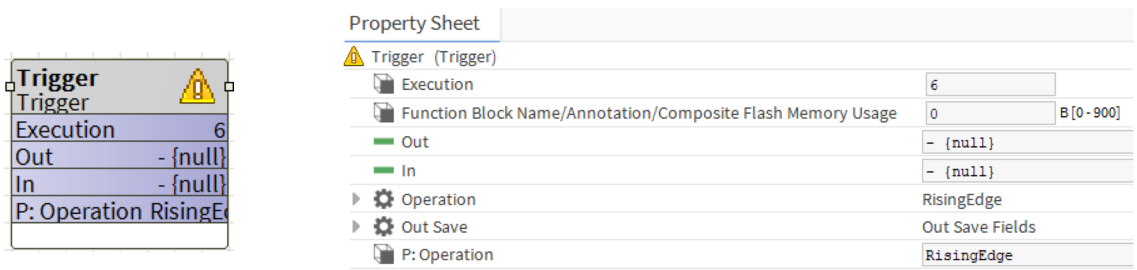


Fig. 293 Trigger Function Block and Property Sheet

Input

Table 211 Inputs of Trigger

Input Name	Description
In	This is a Boolean point input slot. • If In is connected the very first time, there is a change from “null” = “true” or “false.” That leaves **Out** as “false.” • If In is “null,” Operation = RisingEdge or FallingEdge or RisingOrFallingEdge, **Out** = “null.” • If In is “null,” Operation = Off, **Out** = Off.

Output

Table 212 Outputs of Trigger

Output Name	Description
Out	**Output:** “true” or “false” (depends on selected operation).

Parameter

Table 213 Parameters of Trigger

Parameter Name	Description
Operation	There are four operations: • **Rising Edge:** When In changes from "false" to "true," **Out** is set to "true" for 1 DDC cycle. • **Falling Edge:** When In changes from "true" to "false," **Out** is set to "true" for 1 DDC cycle. • **Rising or Falling Edge:** When In changes from "false" to "true" or "true" to "false," **Out** is set to "true" for 1 DDC cycle. • **Off: Out** is set as "false" no matter what In is.
Out Save	• **Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out**: To enable or disable the **Out** feature.

RisingEdge

Input value is rising from "false" to "true," block triggers "true" pulse on the output for 1 DDC cycle.

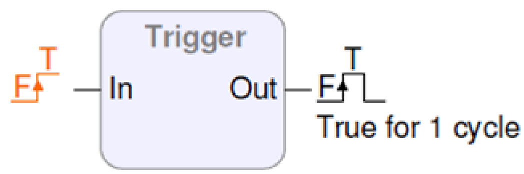


Fig. 294 RisingEdge

FallingEdge

Input value is falling from "true" to "false," block triggers "true" pulse on the output for 1 DDC cycle.

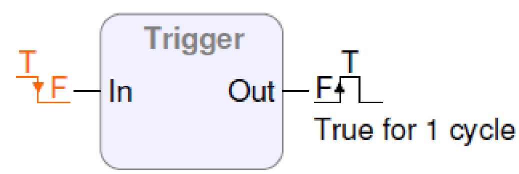


Fig. 295 FallingEdge

RisingAndFallingEdge

Input value is rising from "false" to "true" or from "true" to "false," block triggers "true" pulse on the output for 1 DDC cycle.

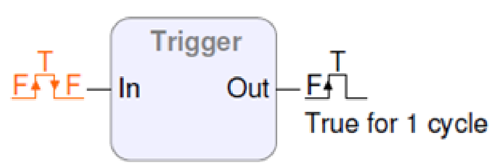


Fig. 296 RisingAndFallingEdge

Examples

Example 1: Trigger (Operation set to RisingEdge)

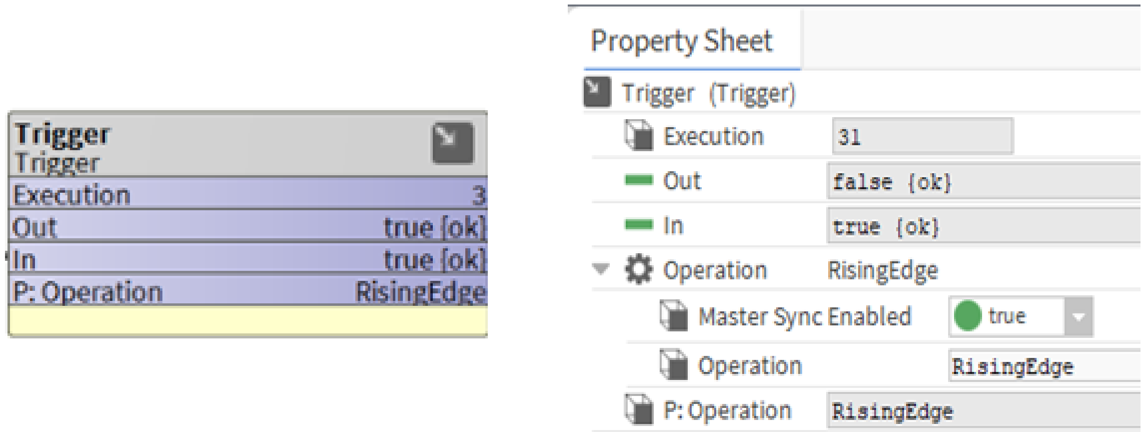


Fig. 297 Rising Edge Functional Block and Property Sheet

Example 2: Trigger (Operation set to FallingEdge)

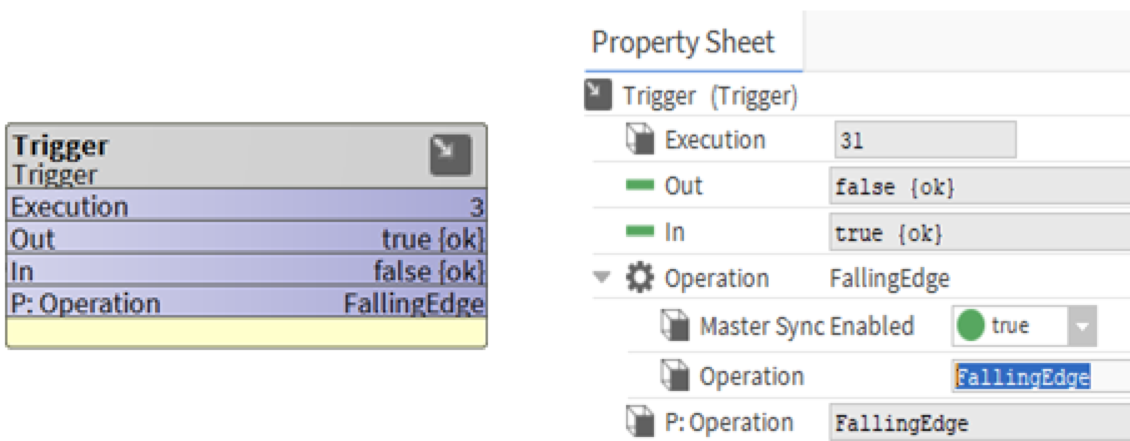


Fig. 298 Falling Edge Functional Block and Property Sheet

Example 3: Trigger (Operation set to RisingOrFallingEdge and In is True)

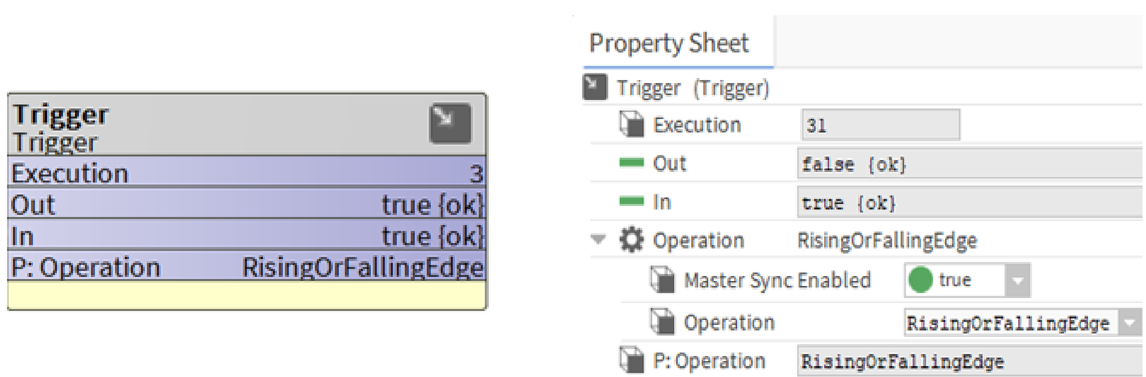


Fig. 299 Falling Edge Functional Block and Property Sheet

Example 3: Trigger (Operation set to RisingOrFallingEdge and In is False)

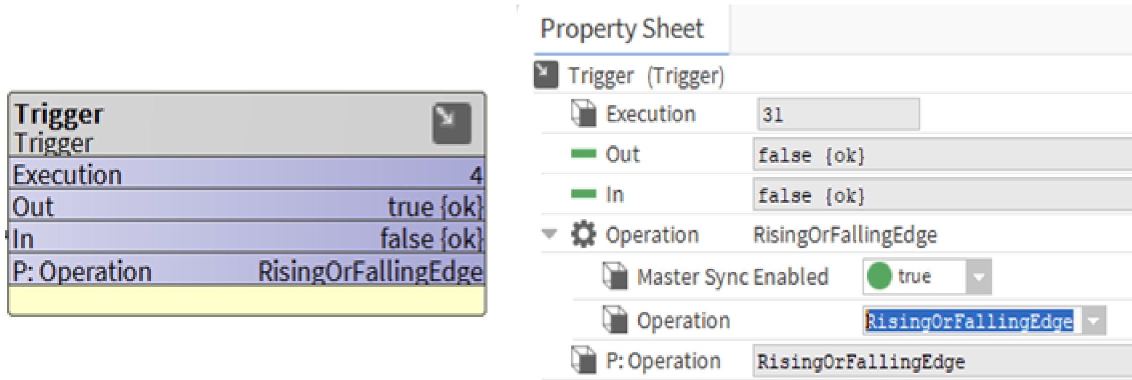


Fig. 300 Rising or Falling Edge Functional Block and Property sheet

CHAPTER

11 TIMER FUNCTION BLOCKS

The honIrmControl palette provides the following Timer function blocks that can be configured and used to build the required application logic:

- One Shot
- Rate Limit
- Time Ramp
- Time Delay
- Multi function Timer
- Runtime Accumulate

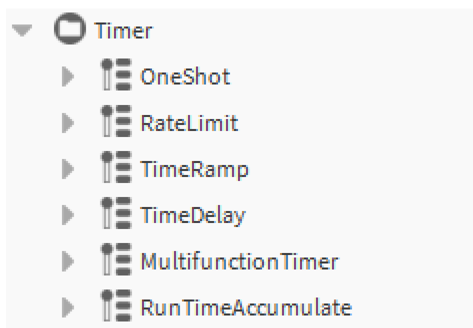


Fig. 301 Timer Function Blocks

One Shot

In the One Shot function block, if the **In** becomes “true” from “false,” **Out** is set to “true” for a given duration.

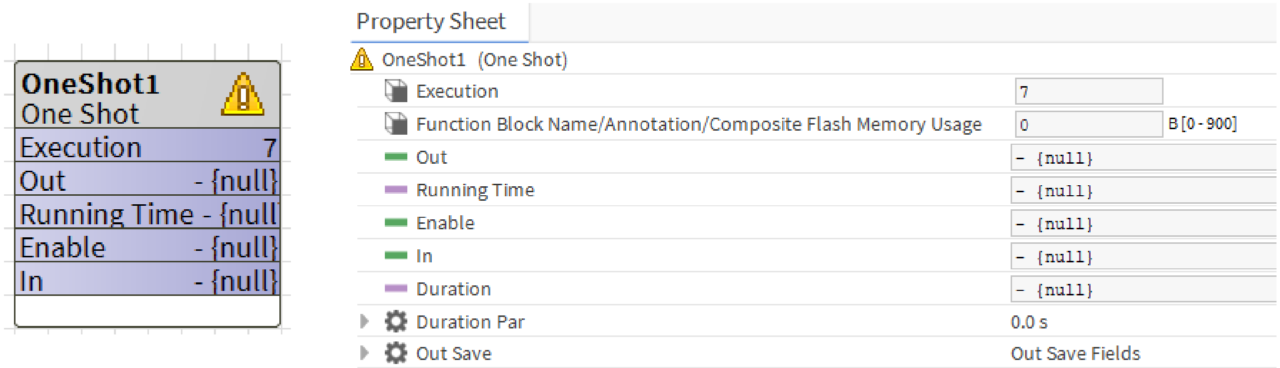


Fig. 302 OneShot Function Block and Property Sheet

If **In** = null, **Out** = 0.

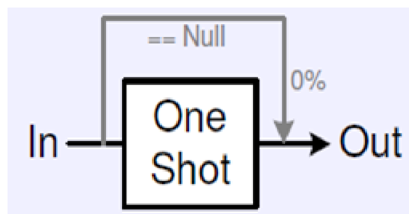


Fig. 303 OneShot Logic Diagram

Example

The **Input** is a square wave of 2 sec amplitude. The time transition diagram of the output for different running times of 0 and 5 seconds is illustrated.

If **Enable** is set to “false,” output is set to “false.”

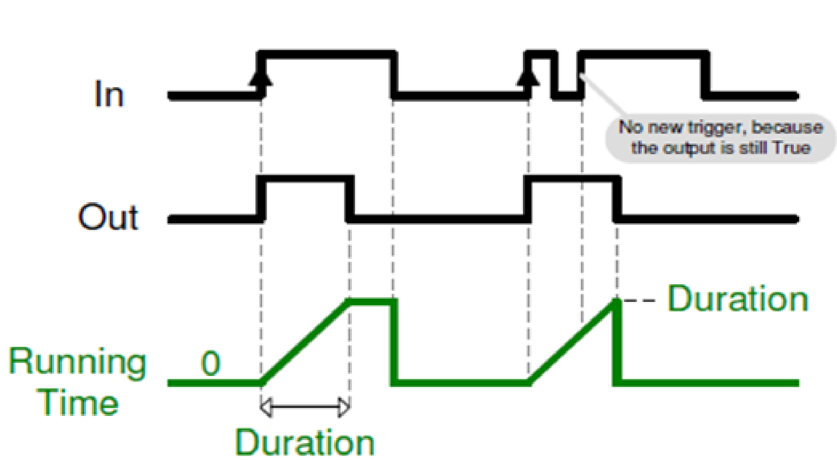


Fig. 304 Time Transition Diagram

Input

Table 214 Inputs of Oneshot

Input Name	Description
Enable	Set this property to true to enable the block.
In	This is a Boolean point.
Duration (In & Par)	Range 0 - + inf sec. Zero keeps the output off and independent of changes at the **In**.

Output

Table 215 Outputs of Oneshot

Output Name	Description
Out	**Output:** When **In** transitions from “false” to “true,” **Out** will be set to “true” for a given duration (seconds).
Running Time	It keeps track of the time that is passed in the given duration.

Parameter

Table 216 Paramaters of Oneshot

Parameter Name	Description
Duration Par (In & Par)	If input duration is “null,” **Duration Par** is used as the parameter.
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out**: To enable or disable the **Out** feature.

Example 1: One Shot (Fan Minimum On Time)

Below is an example of the OneShot function and the minimum on-time requirement for Fan Command.

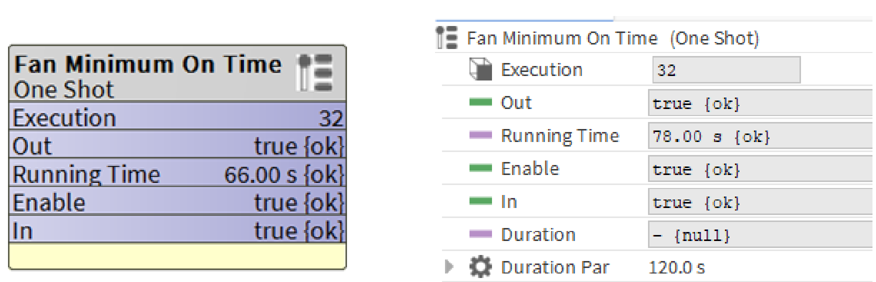


Fig. 305 Oneshot Function Block and Property Sheet

Rate Limit

The Rate Limit function creates an output that follows the input but prevents the output from changing faster than the specified rates. The function block can be used, for example, to stabilize a sensor value. The output moves at a maximum allowed rate towards the new input value each second.

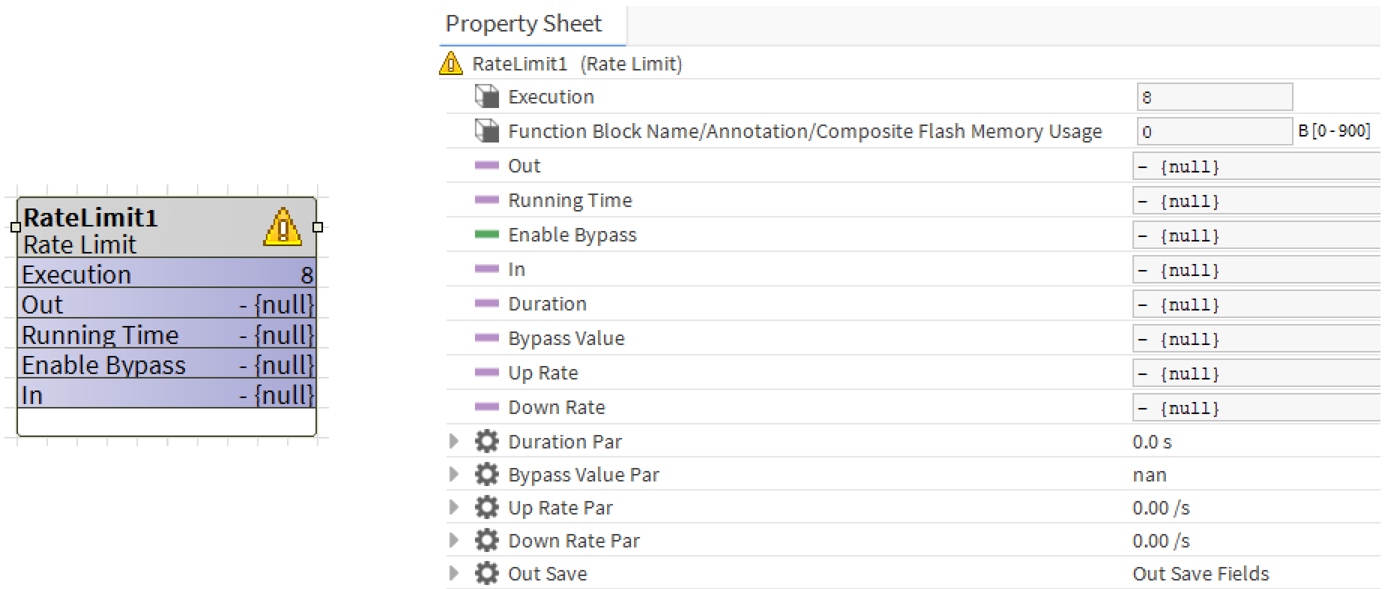


Fig. 306 RateLimit Function Block and Property Sheet

When the Rate limit is configured, the output will move at a maximum allowed rate towards the new input value each second.

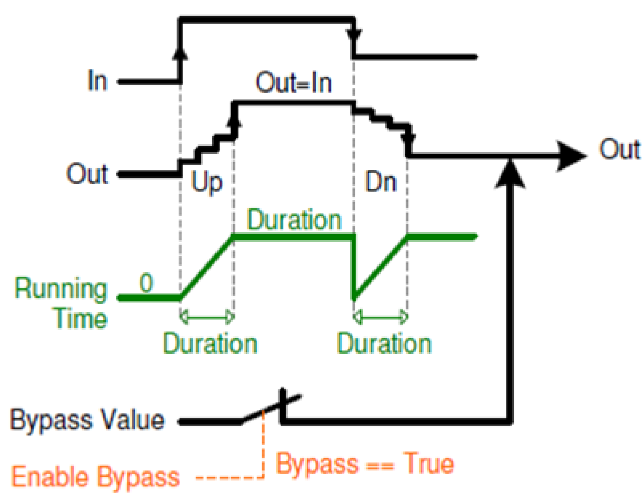


Fig. 307 RateLimit Logic Diagram

Input

Table 217 Inputs of Rate Limit

Input Name	Description
Enable Bypass	If it is set to “true,” the output **Out** is set to the **Bypass Value** independent of **Duration** and **Up** or **Down Rate**. Range: “**false**”**:** Bypass disabled “**true**”**:** Bypass enabled “**null**”**:** Bypass disabled
In	The **In** value is the base for the output **Out**. The output **Out** follows the **In** considering the **Up** or **Down Rate** and the optional Duration timer. If **In** has a value of “Null,” “+inf,” or “-inf,” then **Out** is set immediately to the **In** value irrespective of the **Up** or **Down Rate** and the Duration timer. Similarly, if **In** changes from “Null” or “+inf” or “-inf” to a valid value, the valid value is immediately set to **Out**. ***Note:*** *If a temperature value is connected at the input In, which is read in via a UI terminal, this value can have +inf/-inf for sensor open/short. The +-inf values are immediately passed to the output Out.*
Duration (In & Par)	This value defines a time when the **Out** jumps to the **In** value. • If the Duration timer is running (**Running Time** is increasing), the output follows the defined **Up** or **Down Rates** input. When the duration time has expired, the output receives the value from the input, which will result in a jump. The running Duration timer is shown on **Running Time**. • If **Duration** = 0, **Out** follows the **In** value with the specified Up/ Down rate without the jump. Range: 0 – 259200 sec (48h)
Bypass Value (In & Par)	If the **Enable Bypass** input is “true,” the **Out** is set to this value independent of **Duration** and **Up** or **Down Rate**.
Up Rate (In & Par)	• If **In** value is greater than the **Out** value, **Out** value follows the **In** value according to **Up Rate**. • If **Up Rate** is “0,” **Out** = **In**, if **In** > **Out**.
Down Rate (In & Par)	• If **In** value is less than the **Out** value, **Out** value follows the **In** value according to **Down Rate**. • If the **Down Rate** is 0, then **Out** = **In**, if **In** < **Out**.

Output

Table 218 Outputs of Rate Limit

Output Name	Description
Out	The **Out** follows the **In** considering the **Up** or **Down Rate** and the optional Duration timer.
Running Time	The timer starts from 0 sec with each new **In** value, and the timer value remains if **Out** has attained the **In** value.

Parameter

Table 219 Parameters of Rate Limit

Parameter Name	Description
Duration Par (In & Par)	If the input Duration is “null,” **Duration Par** is used as the parameter. **Default:** 0 sec.
Bypass Value Par (In & Par)	If the input **Bypass Value** is “null,” **Bypass Value Par** is used as the parameter. **Default:** “null” (-> **Out** = **In**)
Up Rate Par (In & Par)	If the **Up Rate** is “null,” **Up Rate Par** is used as the parameter. **Default:** 0 changes / sec
Down Rate Par (In & Par)	If the **Down Rate** is “null,” **Down Rate Par** is used as the parameter. **Default:** 0 changes / sec
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out**: To enable or disable **Out** feature.

Examples

Example 1: RateLimit (Up or Down Rate)

After the complete cycle of **Up Rate** or **Down Rate** and the duration, input will pass into output without any delay.

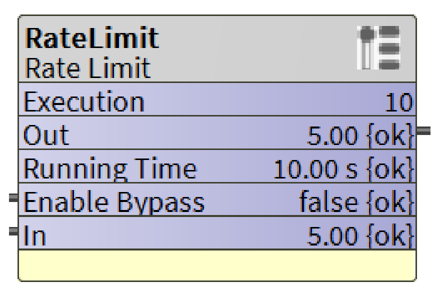


Fig. 308 RateLimit (Up or Down Rate)

Example 2: RateLimit (Ramp Up Cycle)

While the ramp up cycle ramps down based on the **Up Rate Par** value.

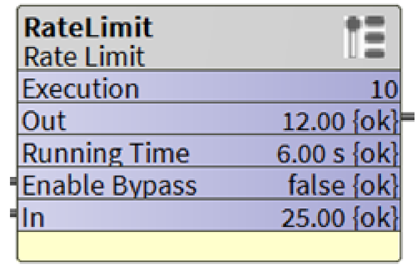


Fig. 309 RateLimit (Ramp Up Cycle)

Example 3: RateLimit (Ramp Up Cycle)

While the ramp up cycle ramps down based on the “down rate parameter value.”

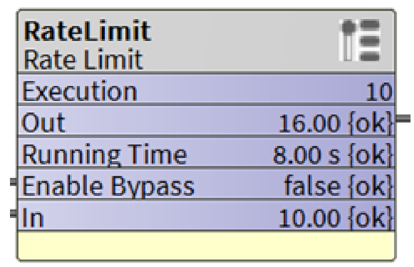


Fig. 310 RateLimit (Ramp Up Cycle)

Example 4: RateLimit (Bypass Override)

While the Bypass Override, the output will change without any delay.

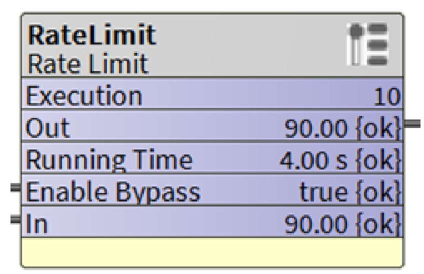


Fig. 311 RateLimit (Bypass Override)

Time Ramp

The Time Ramp functional block keeps decrementing the output till the output is equal to input during the duration of time provided when the block is enabled.

- If the block is disabled (**Enable** = “false”), **In** = **Out**.
- If the block is enabled (**Enable** = “true”), the **Out** will be the summation of **In** and **Delta Par,** starting to decrease the **Out** value towards the **In** value by defined duration time.

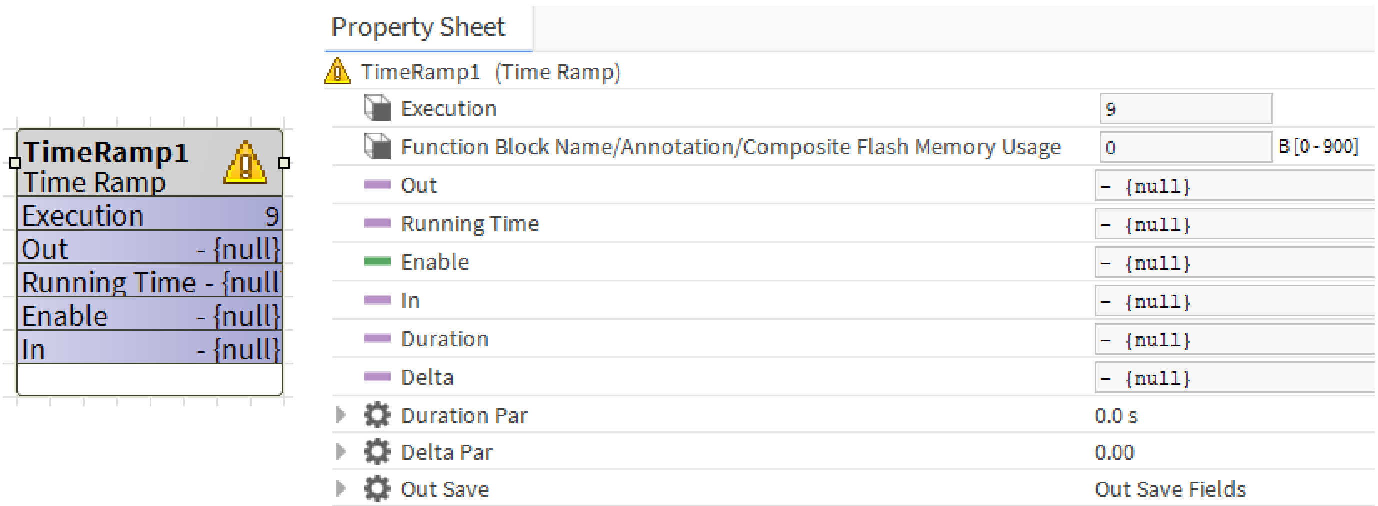


Fig. 312 TimeRamp Function Block and Property Sheet

- If you set **Enable** to “true,” it starts the timer.

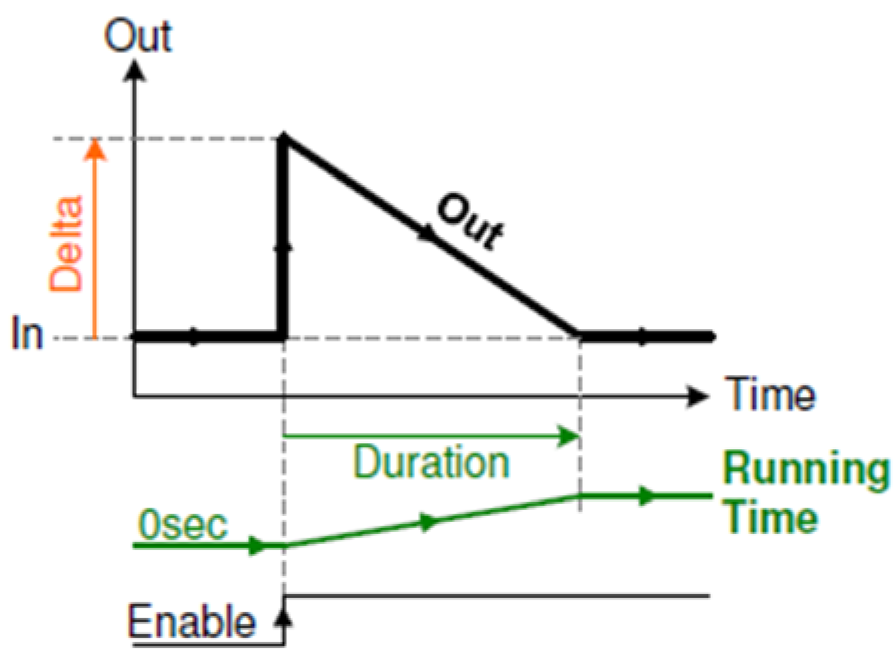


Fig. 313 TimeRamp Start Timer Diagram

- If you set **Enable** to "false," it stops the timer.

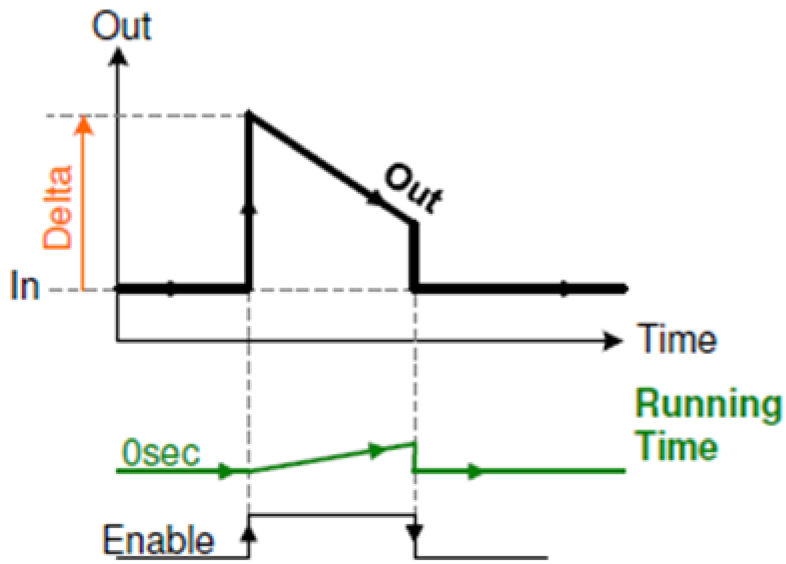


Fig. 314 TimeRamp Stop Timer Diagram

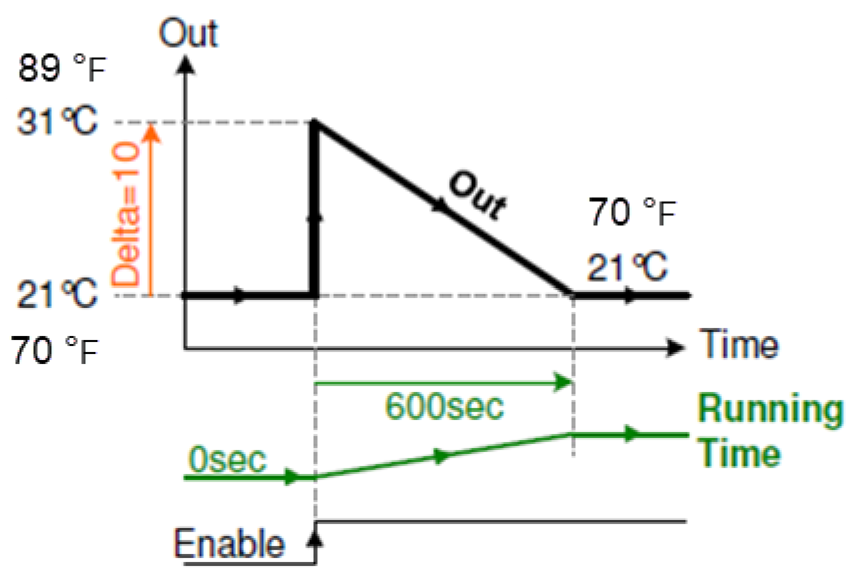


Fig. 315 TimeRamp Setpoint Example

Input

Table 220 Inputs of Time Ramp

Input Name	Description
Enable	This enables or disables the block.
In	This is a 32-bit floating point.

Table 220 Inputs of Time Ramp (Continued)

Input Name	Description
Duration (In & Par)	**Range:** 0 - + inf sec. Zero keeps the output off and independent of changes at the **In**.
Delta (In & Par)	It is the summation parameter with **In** during the block is enabled.

Output

Table 221 Outputs of Time Ramp

Output Name	Description
Out	This is a 32-bit floating point. • If disabled, **In** = **Out**. • If enabled, **Out** = **In** + **Delta Par,** start decrementing towards **In** during the defined duration.
Running Time	It keeps track of the time that is passed in the given duration.

Parameter

Table 222 Parameters of Time Ramp

Parameter Name	Description
Duration Par (In & Par)	If input Duration is “null,” **Duration Par** is used as the parameter.
Delta Par (In & Par)	It is the summation parameter with **In** during the block is enabled.
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out**: To enable or disable the **Out** feature.

Examples

Example 1: TimeRamp

If **Enable** is set to “false”.

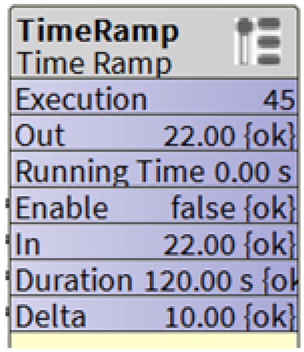


Fig. 316 Enable “false”

Example 2: TimeRamp

If **Enable** is set to “true,” **Out** starts decrementing.

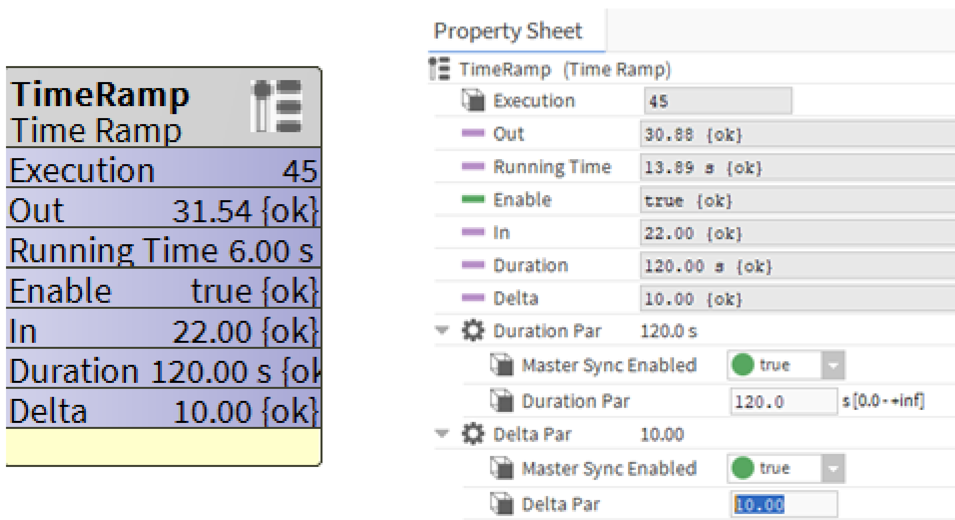


Fig. 317 Enable “true” (decrementing)

Example 3: TimeRamp

If **Enable** is set to “true,” after duration.

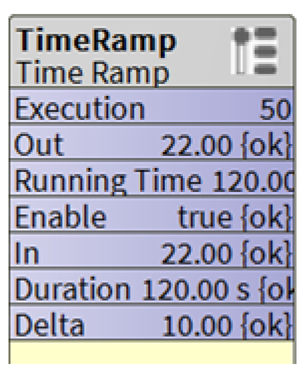


Fig. 318 Enable “true”

Time Delay

The Time Delay function block is used to delay the input value appearing at the output for a delay time provided in the block.

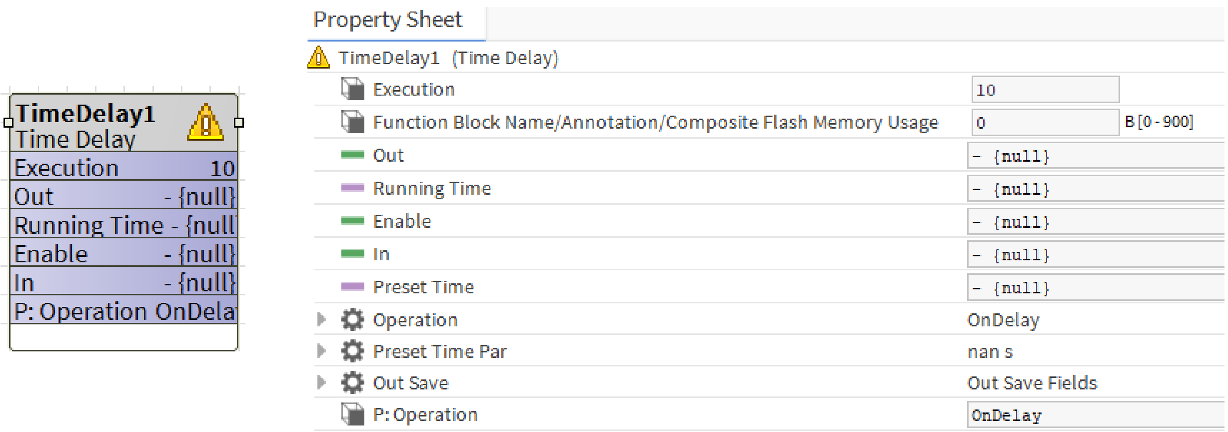


Fig. 319 TimeDelay Function Block and Property Sheet

Input

Table 223 Inputs of Time Delay

Input Name	Description
Enable	This enables the time delay. If the **Enable** is set to “false,” **In** will be set to **Out** simultaneously without any delay.
In	This is a Boolean point.
Preset Time (In & Par)	It is a pre-defined delay duration.

Output

Table 224 Outputs of Time Delay

Output Name	Description
Out	**Output:** Time delay in seconds.
Running Time	It gives information on the time passed during the block execution.

Parameter

Table 225 Parameters of Time Delay

Parameter Name	Description
Operation	There are five operations: • Off • OnDelay • OffDelay • OnOffDelay • Pass
Preset Time Par (In & Par)	It is a pre-defined delay duration.
Out Save	**Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. **Out**: To enable or disable the **Out** feature.

- **OnDelay**: If you select this operation, it delays driving output value for preset.

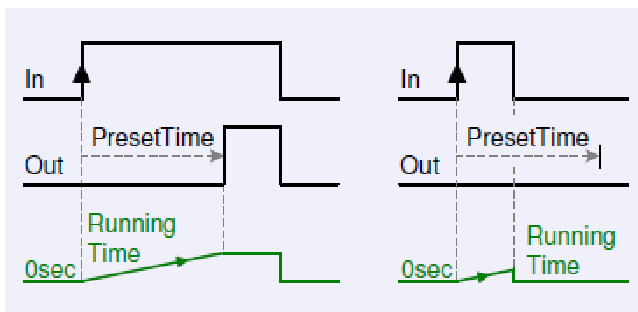


Fig. 320   TimeDelay – On Delay

- **OffDelay**: If you select this operation, it delays the Off (false) value after the given time duration is completed and the input will push the output.

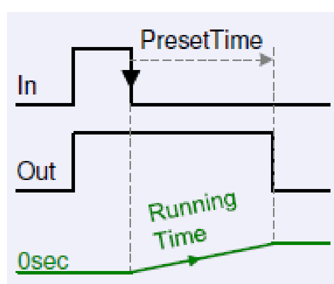


Fig. 321   TimeDelay – Off Delay

- **OnOffDelay**: Every change in the input, output keeps on delaying.

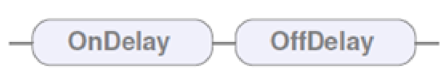


Fig. 322   TimeDelay – On and Off Delay

Examples

Example 1: TimeDelay

If the Enable and In are set to “true,” starts the delay running time until the preset time is reached.

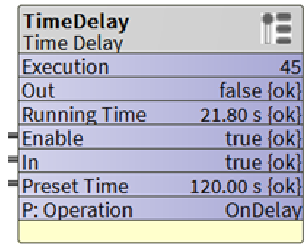


Fig. 323   Enable and In “true”

Example 2: TimeDelay

If the running time is elapsed, **In** = **Out**.

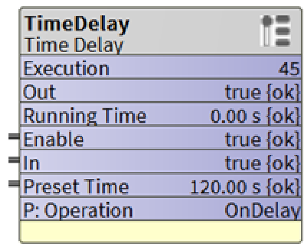


Fig. 324   In = Out, after running time elapsed

Example 3: TimeDelay

If the **Enable** is set to “true” and **In** is set to “false,” starts the delay running time until the preset time is reached.

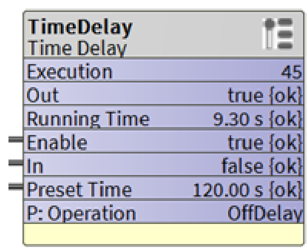


Fig. 325   Enable “true,” In “false”

Example 4: TimeDelay

If the running time is elapsed, **In** = **Out**.

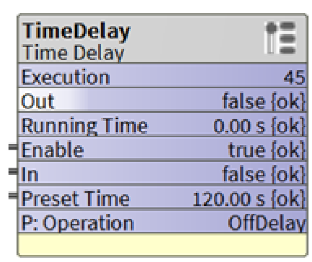


Fig. 326   In=Out after running time elapsed

Multi function Timer

The Multifunction Timer function block is used to generate the various time-based outputs. Two timers are provided for this purpose.

For example, you can generate a flashing output for fault indication, an alarm buzzer, or a part of the application (for example, the PID controller is to be executed in another cycle (30 second cycle)). The function module is defined in a general way to add new functions easily.

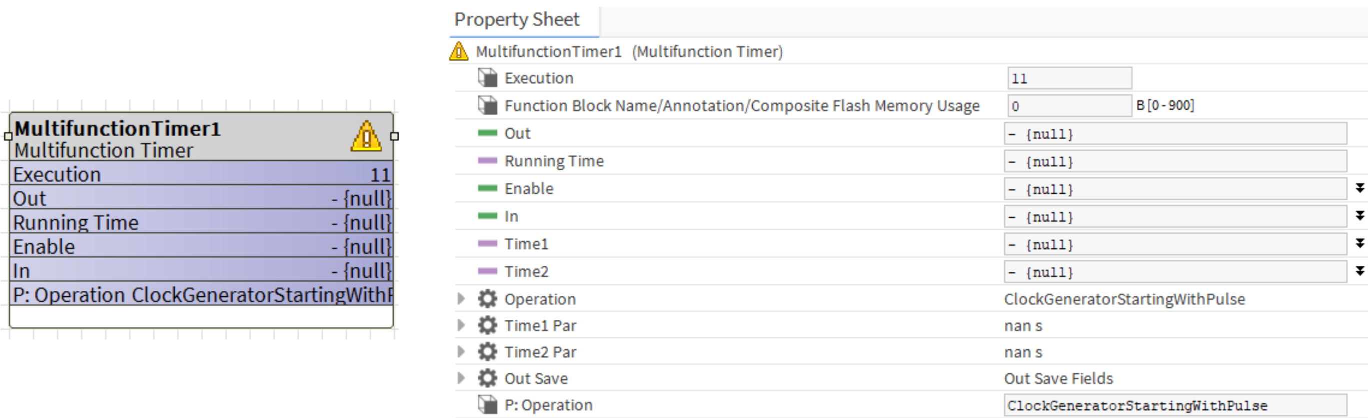


Fig. 327 MultifunctionTimer Function Block and Property Sheet

Input

Table 226 Inputs of Multifunction Timer

Input Name	Description
Enable	This enables the timer function, which is configured via **Operation**. Range: **false:** No timer function, **Out** = “false” **true:** Timer function active as per the Operation mode. **null:** Timer function active
In	The input has different values depending on the Operation. • Clock generator starting with pulse: **0: Out** remains “false.” **1: Out** generates the clock generator pulses. **null: Out** remains “false.” • Clock generator starting with pause: **0: Out** remains “false.” **1: Out** generates the clock generator pulses. **null: Out** remains “false.”

Table 226 Inputs of Multifunction Timer (Continued)

Input Name	Description
Time1 (In & Par)	The **Time1** has different values depending on the Operation. • Clock generator starting with pulse: **Time1** is the Switch-ON time of the pulse signal. • Clock generator starting with pause: **Time1** is the Switch-OFF time of the pulse signal. If time < DDC cycle time, **Out** is ON or OFF at least for 1 DDC cycle.
Time2 (In & Par)	The **Time2** has different values depending on the Operation. • Clock generator starting with pulse: **Time1** is the Switch-OFF time of the pulse signal. • Clock generator starting with pause: **Time1** is the Switch-ON time of the pulse signal. If time < DDC cycle time, **Out** is ON or OFF at least for 1 DDC cycle.

Output

Table 227 Outputs of Multifunction Timer

Output Name	Description
Out	**Output:** “true” or “false” (depend on the Operation).
Running Time	This output shows the current timer value of the timer **Time1** or **Time2**. See the Operation mode.

Parameter

Table 228 Parameters of Multifunction Timer

Parameter Name	Description
Operation	This is used to set the various time functions: • **Off**: **Out** is always “false.” • **ClockGeneratorStartingWithPulse**: **Out** according to the figures below. • **ClockGeneratorStartingWithPause**: **Out** according to the figures below. • **Pass**: **Out** = **In**.
Time1 Par (In & Par)	If **Time1** is “null,” **Time1 Par** is used as the parameter. **Time1 Par** needs a valid timer value between 1 – 259200 secs (3 days).

Table 228 Parameters of Multifunction Timer (Continued)

Parameter Name	Description
Time2 Par (In & Par)	If **Time2** is “null,” **Time2 Par** is used as the parameter. **Time2 Par** needs a valid timer value between 1 – 259200 secs (3 days).
Out Save	**Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. **Out**: To enable or disable the **Out** feature.

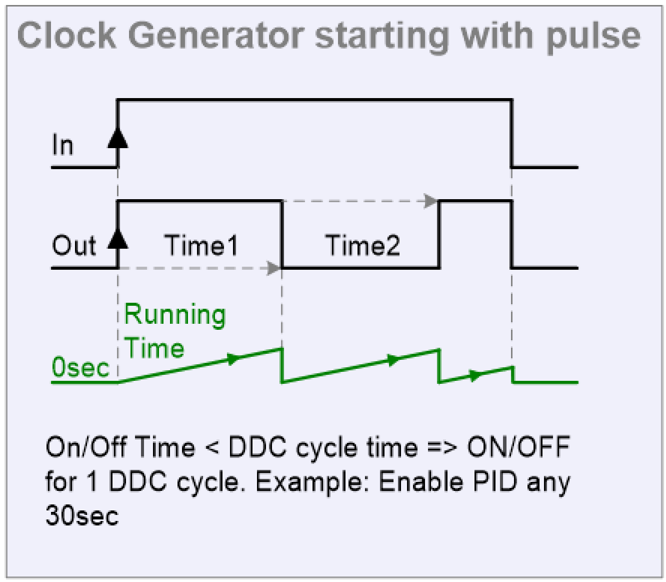


Fig. 328 Operation = Clock generator starting with pulse

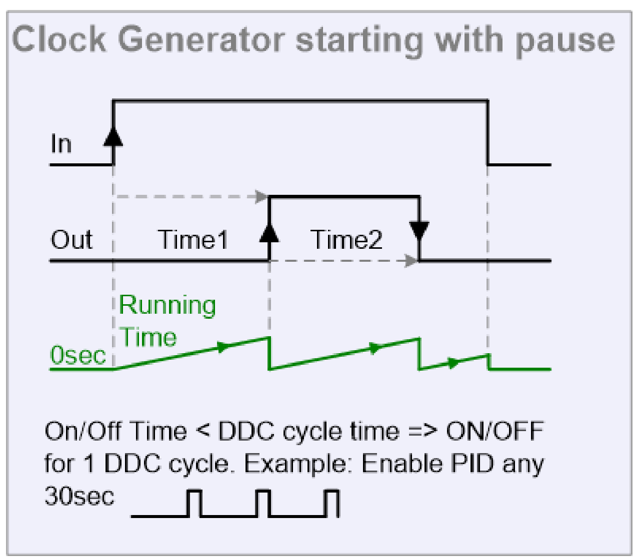


Fig. 329 Operation = Clock generator starting with pause

Example 1: MultifunctionTimer

Following output signal is required: When 10 secs is “true,” 20 secs is “false” and endless repetition.

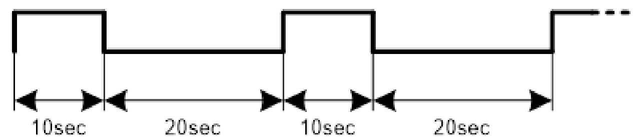


Fig. 330 MultifunctionTimer Function Block – Schematic

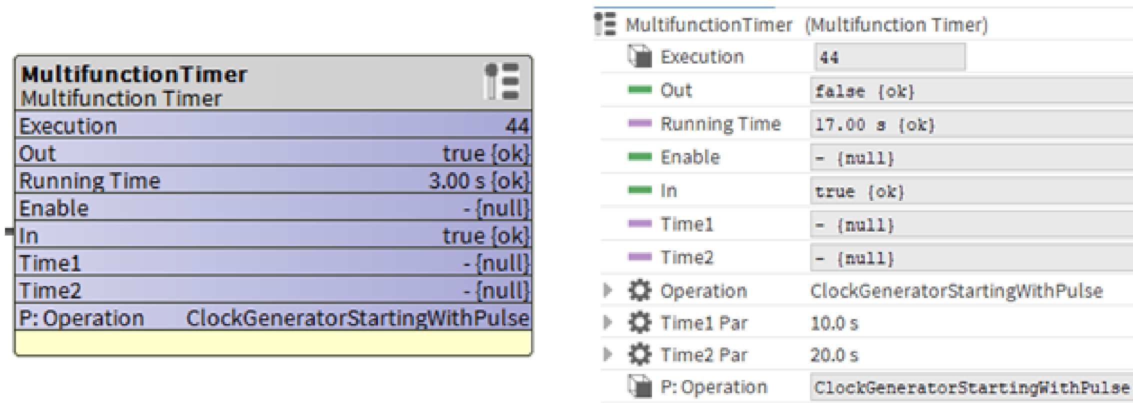


Fig. 331 Multifunction Timer Function Block and Property Sheet

Runtime Accumulate

This function accumulates runtime whenever the **Input** = “true” (non-zero) and the **Enable** = “true.” If **Preset** is “true,” then runtime is set equal to the **Preset** Value. Runtime is provided in 4 outputs of seconds, minutes, hours, and days. From iteration to iteration, the Function Block keeps track of the Runtime Sec.

Depending on the controller type, the Runtime Min is saved over a power outage. The other times like Runtime Hours, Runtime Days are not lost. They are restored from Run Time Min. The Runtime Sec starts at 0 after a power failure, which means up to one minute will be lost then. Only the controllers that can detect a power failure support saving to the flash, which is the Compact VAV controller with MSTP.

The other controllers additionally require a Save Permanent function block to store the time in the flash. If not, the function block starts at 0.

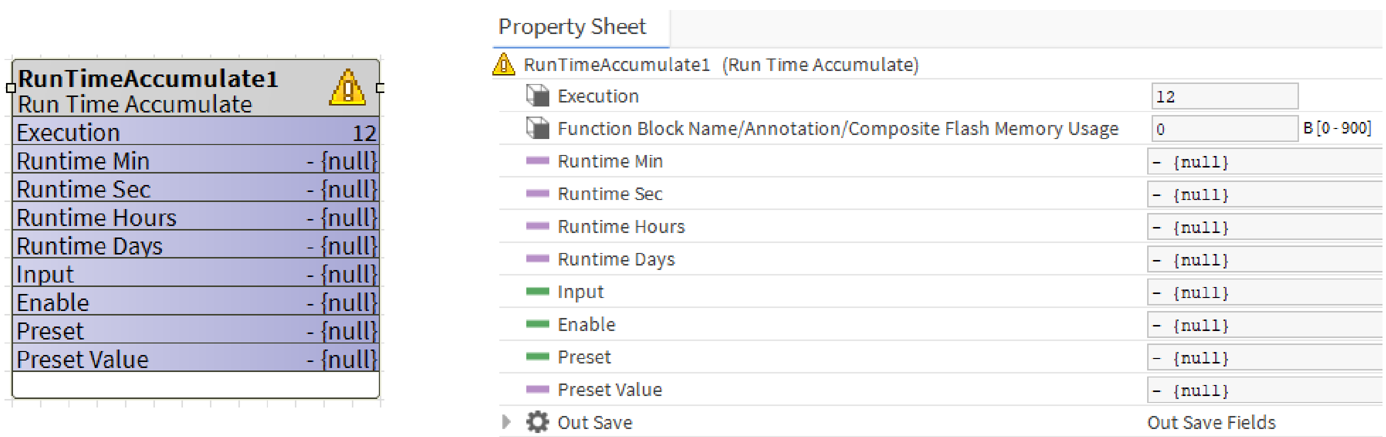


Fig. 332 Runtime Accumulate Function Block and Property Sheet

Input

Table 229 Inputs of Runtime Accumulate

Input Name	Description
Input	**Input** Value **Numeric:** Binary value 0 - 1, **Default:** Null.
Enable	Run Time accumulate **Enable**. **Numeric:** Binary value: 0 - 1, **Default:** Null.
Preset	**Preset** input sets the accumulator output to Presetvalue. **Numeric:** Binary value: 0 - 1, **Default:** Null.
PresetValue	Run Time accumulate **PresetValue** (min). **Numeric:** 32 Bit Floating value: 0 - 167772216 (>30 years), **Default:** Null.

Output

Table 230 Outputs of Runtime Accumulate

Output Name	Description
RunTimeMin	Run Time output in minutes. Depending on the controller, this value is saved in the Flash, which means after a new Power-Up of the controller, this value is restored as it was before. The value is the base to restore all the other outputs **RunTimeSec**, **RunTimeHrs**, **RunTimeDays**. The value stops at 16777216. **RunTimeMin** starts at 0 for the controllers that do not support saving to flash. Please see the general description from above of Runtime Accumulate for the corresponding controllers. **Numeric:** 32-Bit floating point value, 0 - 16777216 min (>30 years).
RunTimeSec	This value is restored from the **RunTimeMin** after the power is lost or after a new **Preset** value. For example, max. 59sec is lost after the power is lost for the controllers, which saves **RunTimeMin** in the Flash. The value stops at 1006632960 (>30 years). **Numeric:** 32-Bit floating point value, 0 - 1006632960.
RunTimeHrs	This value is restored from **RunTimeMin** after power is lost or after a new **Preset** value. The value stops at 279620.28 hr. **Numeric:** 32-Bit floating point value, 0 - 279620.28 hrs.
RunTimeDays	This value is restored from **RunTimeMin** after power is lost or after a new **Preset** value. The value stops at 11650.84 days. **Numeric:** 32-Bit floating point value, 0 - 11650.84 days.

Parameter

Table 231 Parameters of Runtime Accumulate

Parameters	Description
InputPar (In & Par)	Since this parameter is also available as an input, please refer to the description of the **Input**. **Default Value:** Null
EnablePar (In & Par)	Since this parameter is also available as an input, please refer to the description of the input **Enable**. **Default Value:** Null
PresetPar (In & Par)	Since this parameter is also available as an input, please refer to the description of the input **Preset**. **Default Value:** Null
PresetValuePar (In & Par)	Since this parameter is also available as an input, please refer to the description of the input **PresetValue**. **Default Value:** Null

Table 231 Parameters of Runtime Accumulate (Continued)

Parameters	Description
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Runtime Min**: To enable or disable the **Runtime Min** feature. • **Runtime Sec**: To enable or disable the **Runtime Sec** feature. • **Runtime Hours**: To enable or disable the **Runtime Hours** feature. • **Runtime Days**: To enable or disable the **Runtime Days** feature.

Examples

Example 1: Runtime Accumulate

If **Input** and **Enable** are “true” for 123 minutes, the RunTimeAccumulate FB shows the following result:

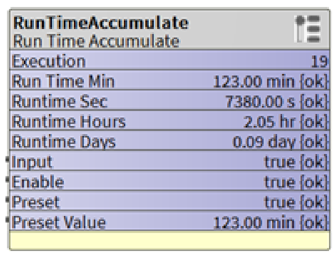


Fig. 333 Runtime Accumulate Function Block

Example 2: Runtime Accumulate

Configuration of IRM CVAV (FCU) with IRM VAV (MSTP)

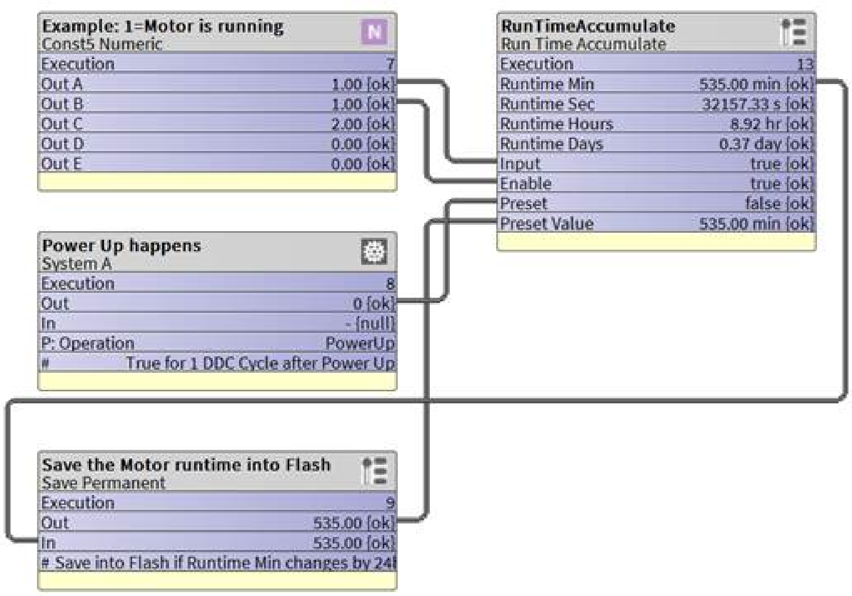


Fig. 334 Runtime Accumulate Function Block

Note: *The execution order “SavePermanent” must be run before “RunTimeAccumulate.”*

CHAPTER

12 DATE AND TIME FUNCTION BLOCKS

The following Date and Time function blocks are available in the honIrmControl Palette that can be configured and used to create the required application logic:

- Enum Schedule
- Calendar
- Current Date Time
- Date And Time Operation

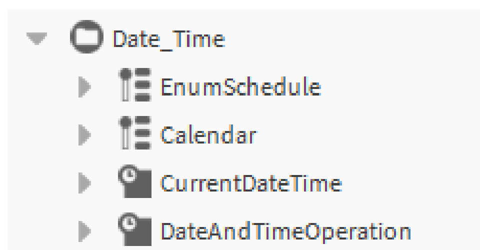


Fig. 335 Date and Time Function Blocks

Enum Schedule

The schedule functional block can be configured and linked to control the scheduling of other components. Each schedule functional block includes a "Schedule Config" view for defining events.

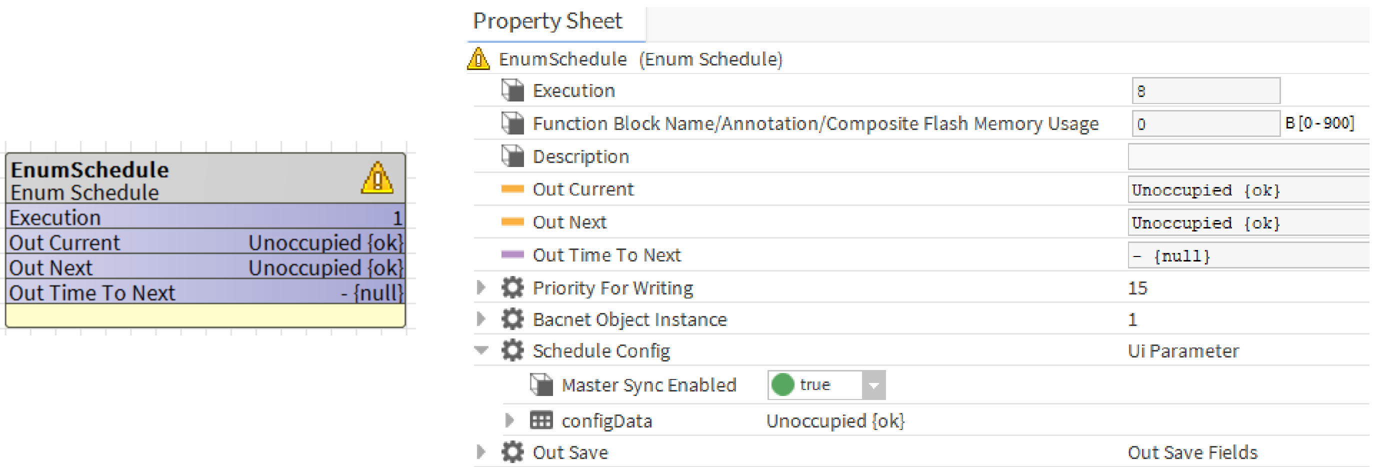


Fig. 336 Schedule Function Block and Property Sheet

Output

Table 232 Outputs of Schedule Function

Output Name	Description
Out Current	Current schedule default state **0:** Occupied **1:** Unoccupied **3:** Standby **255:** Null **Numeric:** 32-Bit Floating value 0, 1, 3.
Out Next	Next schedule default state **0:** Occupied **1:** Unoccupied **3:** Standby **255:** Null **Numeric:** 32-Bit Floating value 0, 1, 3.
Out Time To Next	Time until the next state change (in minutes). **Numeric:** 32-Bit Integer value 1-11520.

Parameter

Table 233 Parameters of Schedule Function

Parameter Name	Description
Priority For Writing	It is used to pass the value with a defined priority because the BACnet output type receives the value in an array of 15.
Bacnet Object Instance	It displays the BACnet object function block's instance number.
Schedule Config	To enable and configure configData, double-click scheduleConfig in the EnumSchedule view. **0:** Occupied **1:** Unoccupied **3:** Standby **255:** Null
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out Current:** To enable or disable the **Out Current** option. • **Out Next:** To enable or disable the **Out Next** option. • **Out Time To Next:** To enable or disable the **Out Time To Next** option.

Schedule Configuration

Step 1. Double-click **Schedule Config** on the EnumSchedule property sheet.

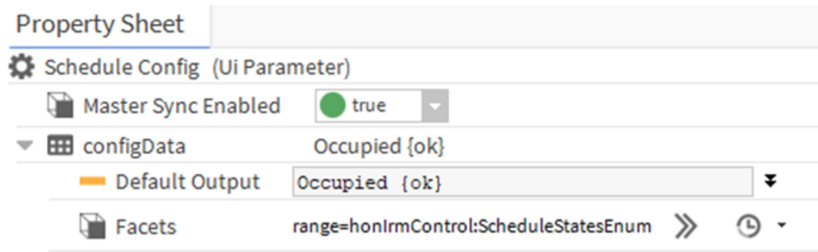


Fig. 337 Schedule Config Property

Step 2. Select **Master Sync Enable** to true.

Step 3. Click **ConfigData** to view the AX Scheduler.

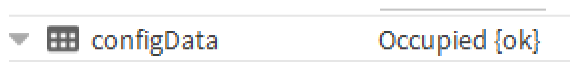


Fig. 338 Schedule Configure Option

Note: *Add supported devices to the wiresheet before configuring Config Data scheduler properties.*

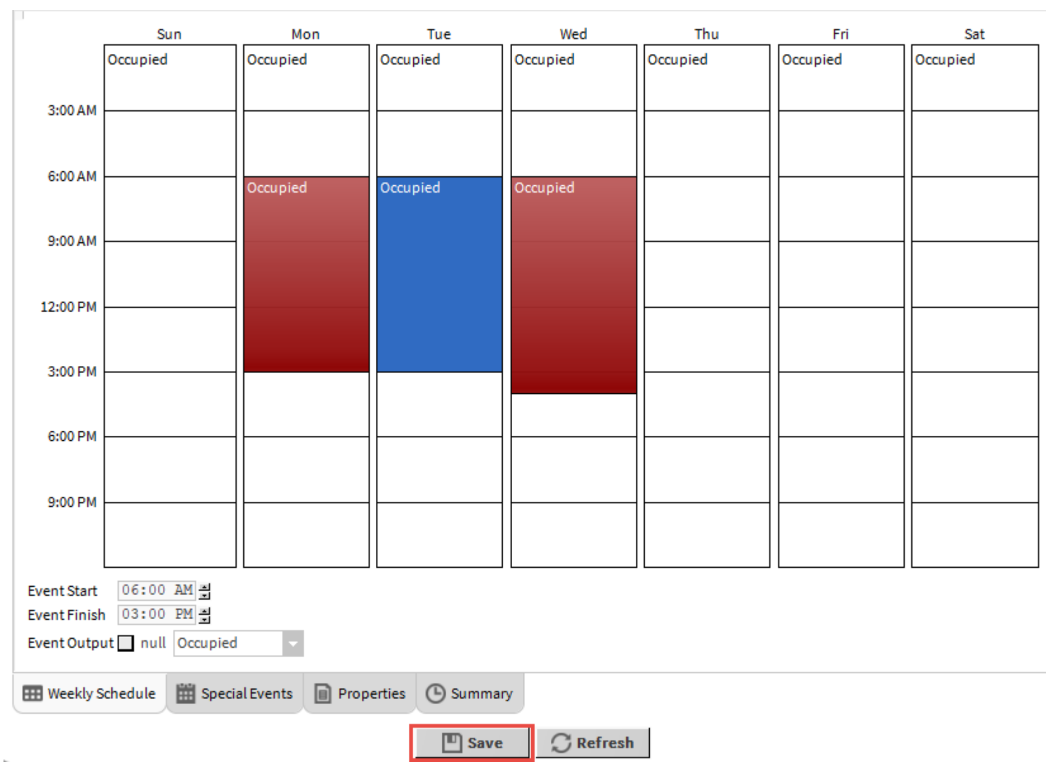


Fig. 339 AX Scheduler

Step 4. Click **Save** to save the changes made.

Or

Click **Refresh** to cancel the changes.

Configuring Schedule Event with Supported Device

Step 1. Open the property sheet of the supported device model (Example: TR75 and TR120).

Step 2. Set the **Schedule Editable From Zio** field to **Yes**.

Step 3. Change **Param Permissions** (Contractor Only, Tenant Read only and Tenant Write only) as per requirement.

Step 4. Change the **Schedule Type** as per requirement (8 day schedule/ 7 day schedule/ 5-2-1 day schedule/ 5-2 day schedule).

Step 5. Enter the schedule block location in the **Schedule Block Location** field.

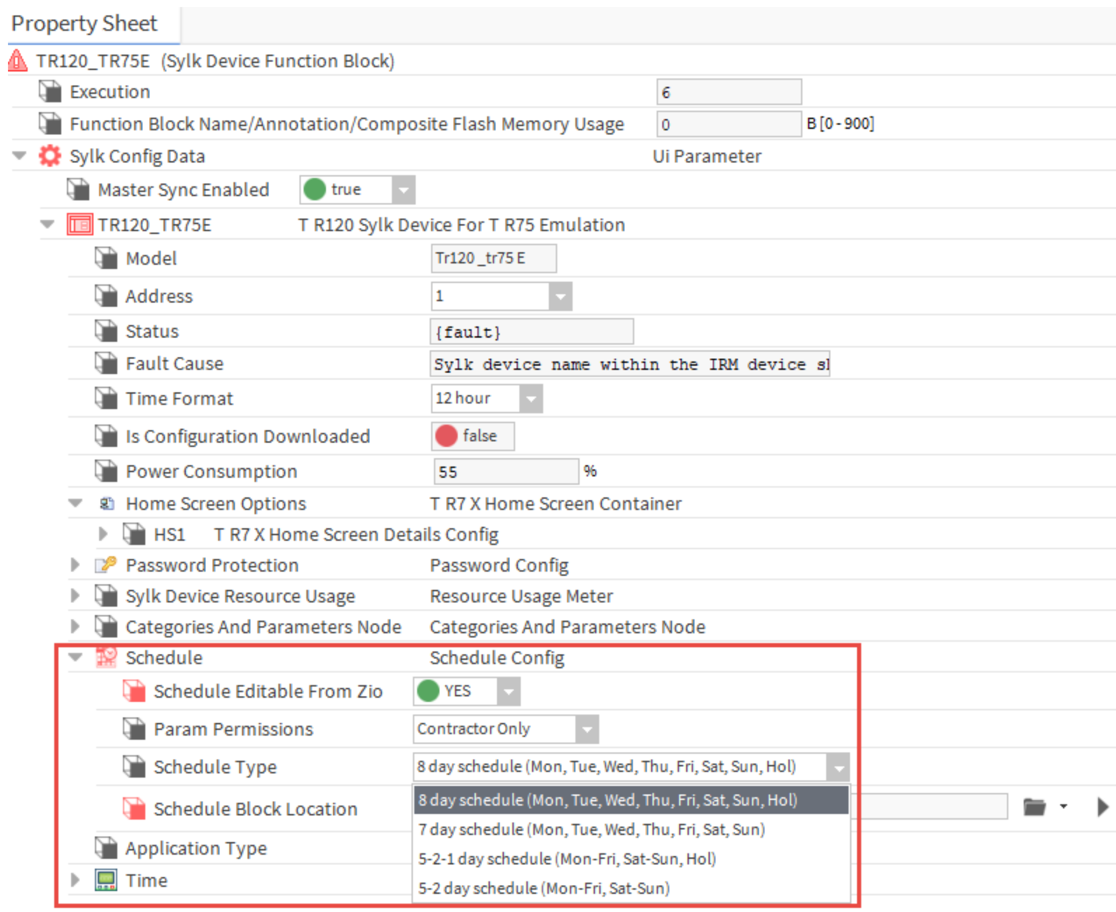


Fig. 340 Sylk Schedule Configuration

Configuring Schedule Block Location

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

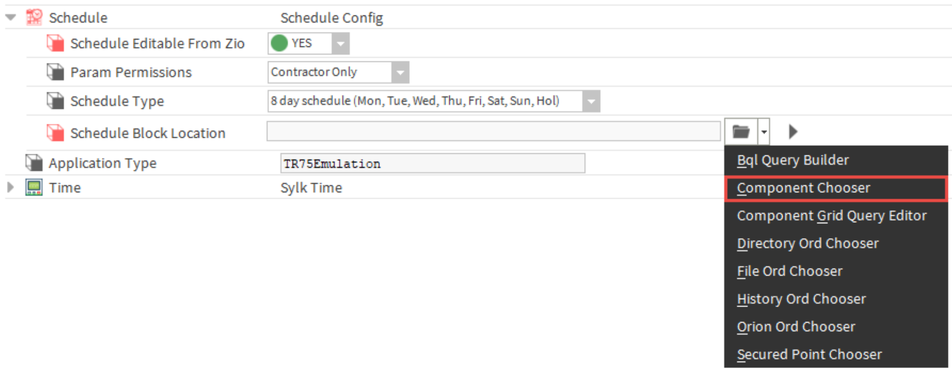


Fig. 341 Component Chooser

Step 2. On the **Select Ord** window, click **Drivers** > **BacnetNetwork** > **IrmBacnetDevices** > **IRM Program** > **Event Program** > **EnumSchedule**, and click **OK.**

This action fetches the scheduler location to the **EnumSchedule Block Location** filed.

Step 3. Click **Save** Or click **Refresh** to cancel the changes.

Or

Go to Nav Window, navigate to the **Station** > **Config** > **Drivers** > **BacnetNetwork** > **IrmBacnetDevices** > **IRM Program** > **Event Program** > **Select EnumSchedule,** and press Ctrl + L.

This action opens the Ord window, copy the location,

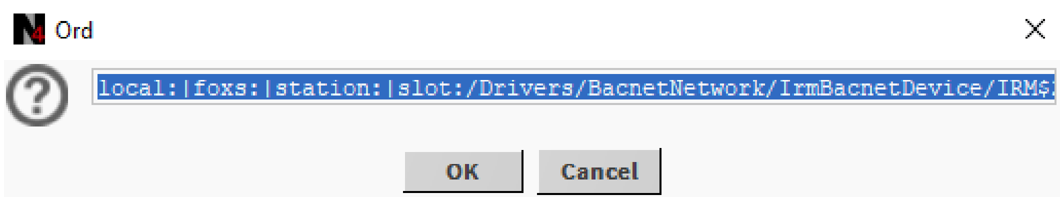


Fig. 342 Ord Window

Navigate to the **Schedule Config** option and paste the link in the **Scheduler Block Location** field.

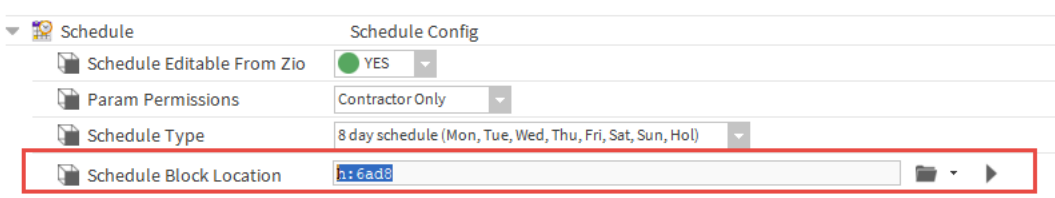


Fig. 343 Scheduler Block Location

Step 4. Click **Save** to save the changes.

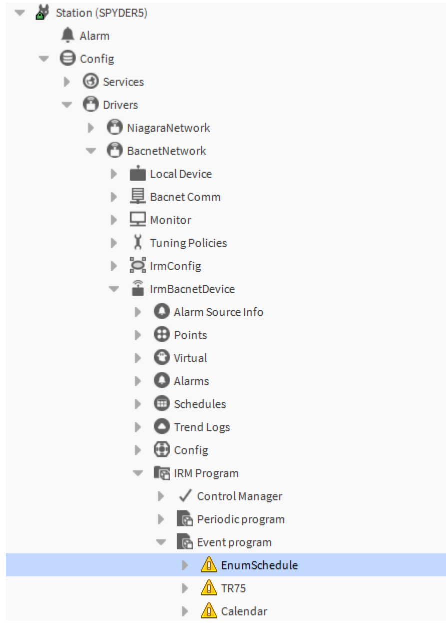


Fig. 344 Enum Schedule

Configure Enum Range

Define Enum range

Steps to define Enum range:

Step 1. Double-click on **ConfigData** > **Properties** > **Facets option,** select **default range** (honIrmControl: ScheduleStatesEnum), and click (>>) icon.

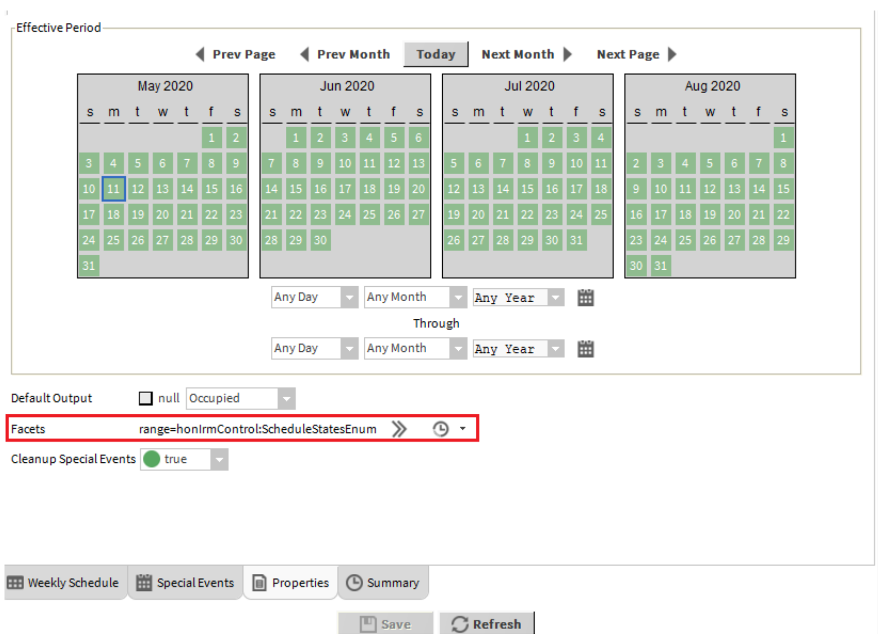


Fig. 345 Defining Enum range

Step 2. Select the default **Enum** range, click the (...) icon, and click the (>>) icon.

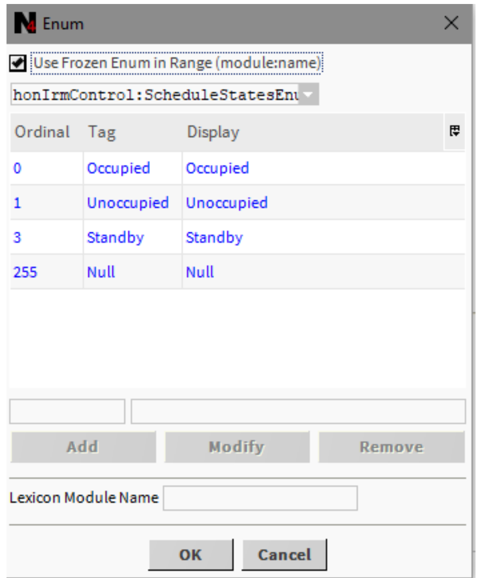


Fig. 346 Default Enum Range

Add Entry in Enum library

Steps to add an entry in the Enum library:

Step 1. Select the blank fields.

Step 2. Enter the **Ordinal** value and display information in the respective field and click **Add**.

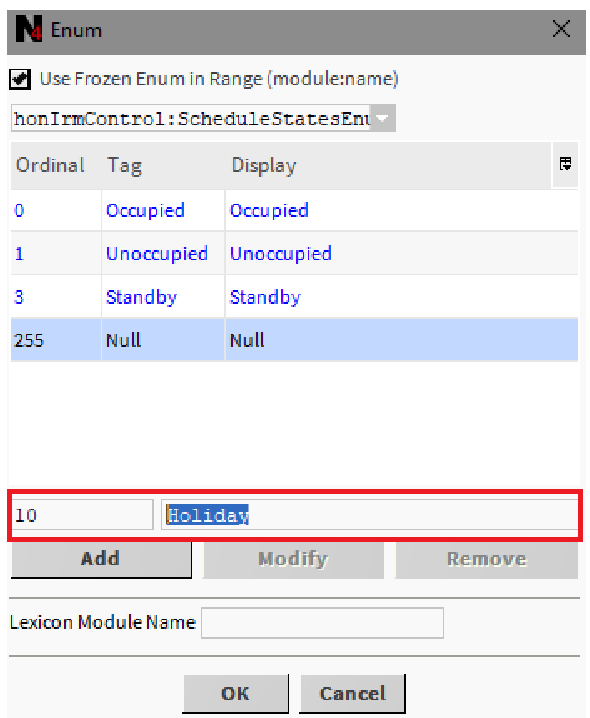


Fig. 347 To add an Entry in Enum Library

Note: *When the schedule is linked to the Sylk Device, it is recommended that you use the default facets configuration. If the user changes the facets value from the default, the Sylk module may not function properly.*

Modify Enum library

Steps to modify the entry in the Enum library:

Step 1. Select the entry.

Step 2. Modify the **Ordinal** value**, Display** field**,** and **Tag** field and click **Modify.**

Or

Step 3. Change the **Ordinal** value**, Display** field**,** and **Tag** field and click **Add**.

Additional Configuration

The following are the additional configuration available on the Scheduler:

- Event time tuning
- Event Output
- Right-click menus

Event Time Tuning: You can set the start and finish time for any selected event by selecting the hours or minutes from the calendar (see below screenshot) or directly entering the event start and event finish values in the respective fields.

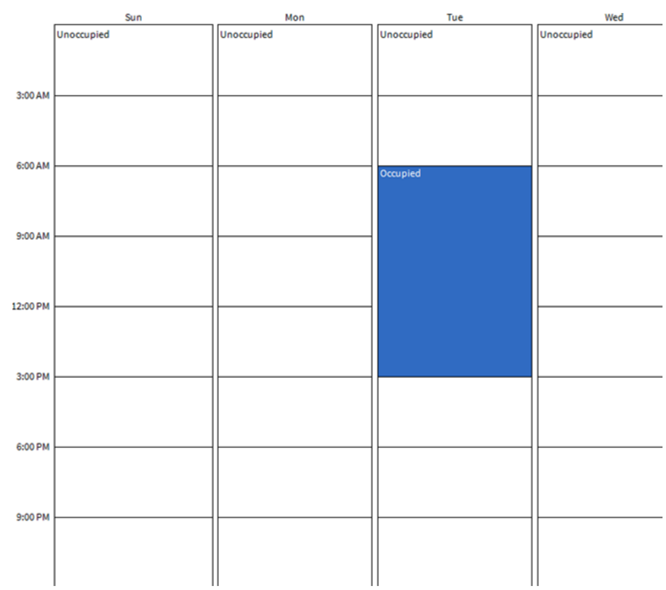


Fig. 348 Schedule Event Time Selection

Event Output – You can select the required output value from the drop-down list.

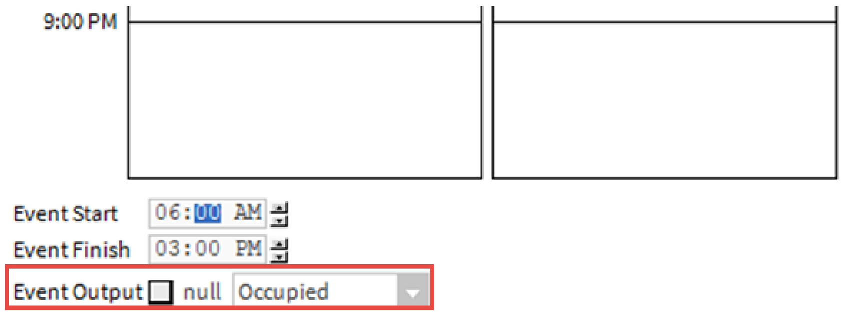


Fig. 349 Schedule Output Value Selection

Right-click menus – The below table describes the available right-click menus (in the schedule area) and their descriptions.

Table 234 Right-click menus

Options	Descriptions
Delete Event	Deletes the selected event.
Paste Day	Appears only if the copy day option was used first. Copies all events into the selected day.
All Day Event	Makes currently selected (or last entered) events extend to an entire day.
Apply M-F	Copies all events in the selected day to Mon, Tue, Wed, Thu, and Fri (and overwrites any existing events on those days).
Copy Day	Copies all events in the selected day to use with the paste day option.
Clear Day	Clears all events for the selected day.
Clear Week	Clears all events in the entire weekly schedule.

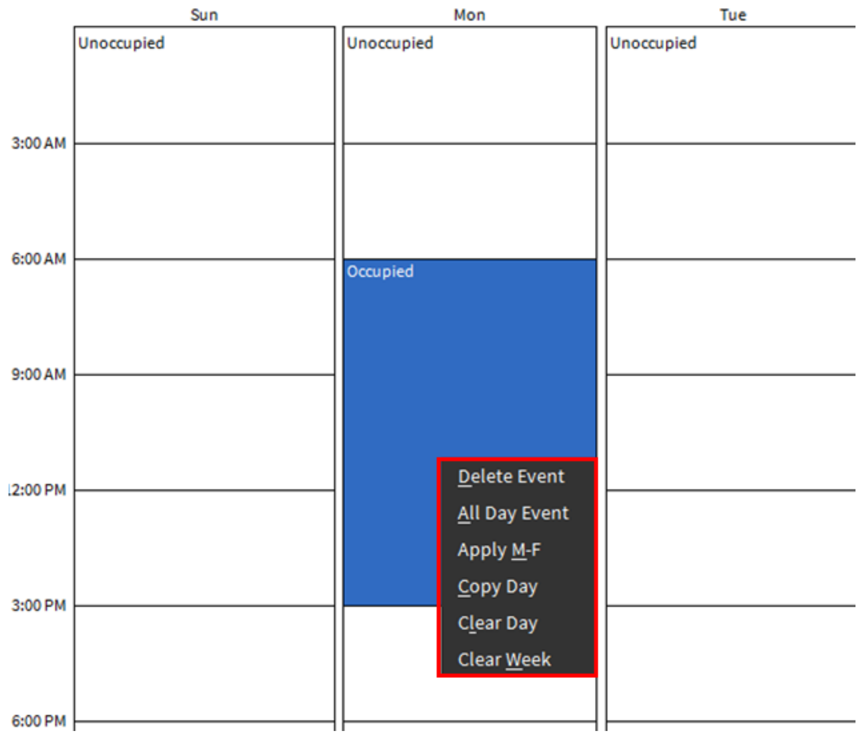


Fig. 350 Right-click menus

Special Events

In the schedule view, each schedule object's special events are configured on a special events tab. The event times and values entered for any special event are only valid for that schedule.

The days of the special event's occurrence should be specified as a reference in the Calendar Schedule functional block if it is configured as a "reference" type. It allows you to change the dates of special events that happened while editing one or more referenced CalendarSchedules on a global scale.

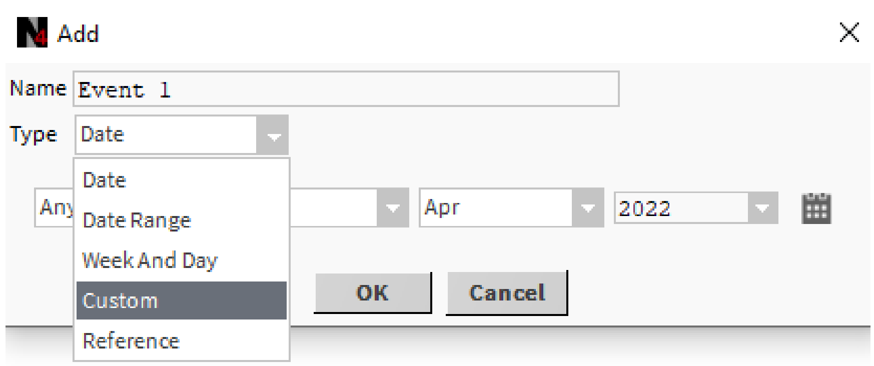


Fig. 351 Schedule - Special events

Table 235 Special Events

Options	Descriptions
Date (default option)	Various combinations of weekday, numerical date, month or month combinations, and year.
Date Range	Start and end range, using for each a combination of day, month, and year.
Week and Day	Combination of day of the week, week in the month. ***Note:*** *Calendar Week type will not support.*
Custom	By combination of any day, month, weekdays, any week, and year ***Note:*** *Custom type events for Special events not supported by IRM CVAV.*
Reference	Refer to a specific calendar schedule component in the station

Table 236 Special Event Priority

Options	Descriptions
Add	Add a new special event.
Edit	Edit day(s) selection criteria (but not changing special event type)
Rename	Rename selected special event
Priority (up)	Move special events up in the priority list.
Priority (down)	Move special events down in the priority list.
Delete	Removes selected special events from the schedule component.

Sylk wall module supports the following exception:

- One exception schedule to configure for a holiday.
- BACnet schedule supports more than one exception schedule.

You can edit one of the first exception schedules in the wall module, but the second and subsequent exception schedules cannot be edited.

To work around this limitation, configure the calendar object with multiple holidays and use calendar references to create an exception schedule in the controller schedule.

Note: *You can change the schedule for a single holiday using the TR120x and TR75x wall modules. Because the controller supports a separate schedule for each holiday, when you integrate TR120x and TR75x, you can only edit the first holiday created in the schedule.*

Use the calendar object to create multiple holidays and assign them to a single schedule when working with TR120x and TR75x wall modules. When you change the holiday schedule, it affects all holidays.

Creating Event for Multiple Days

You can create one event and apply it to multiple days.

Steps to create an event for multiple days:

Step 1. Add a **Calendar** function block in the wiresheet, define all the holiday dates, and date ranges on the calendar.

Note: *When using the Schedule and Calendar function blocks in the application, the controller time should be synchronized with the system time (such as PC time); otherwise, the Schedule function block will have incorrect references and may not function properly. The best way to synchronize time in the controller is to use the TSYNC feature.*

Step 3. Navigate to **EnumSchedule** > **ConfigData** > **Special Events,** create one **Exception Schedule**, select **calendar Type** as **Reference**, and click **OK.**

Now define the time events for this Exception Schedule. The schedule will execute these time events for all the dates defined in the Calendar.

Step 5. Wall module displays this exception schedule time events (as this is the first one) as Holiday time events on display.

Step 6. You can modify the holiday time events. These modified time events get written back to the first Exception Schedule which is configured as Calendar Type as Reference. Hence same time events get applied to all the defined dates on the Calendar.

Default Output

When a scheduled event (special or weekly) is not defined or effective, the schedule block output (Out slot) will be the default value.

In listed events, the white area indicates where the default value is used and displays the current default value.

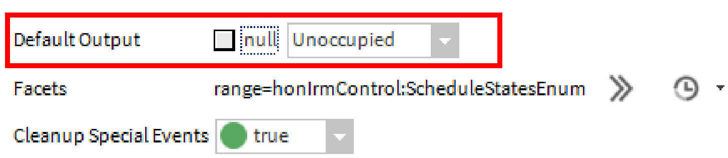


Fig. 352 Schedule event default value

Note: *Custom-type events for Special events are not supported by the IRM CVAV.*

Note: *Sylk Commissioning fails if the number of configured events per day on the scheduled holidays exceeds four.*

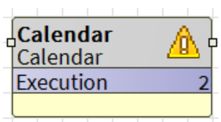
Note: *Sylk commission should be performed after changing any schedule changes in the tool.*

Note: *A maximum of 255 entries can be added to a Zio enum library.*

Calendar

The Calendar function block contains a list of holidays that are used in the exception schedule of a controller. You can create multiple calendars in the same controller, each with its own set of dates.

Calendar entries can refer to a specific date, a date range, or a recurring event. Instead of linking CalendarSchedules, you can "reference" them from the "special events" configuration of one or more weekly schedules. Each CalendarSchedule defines the "day portion" of a special event as a reference. Then, in each special event, you configure time-of-day events as needed.



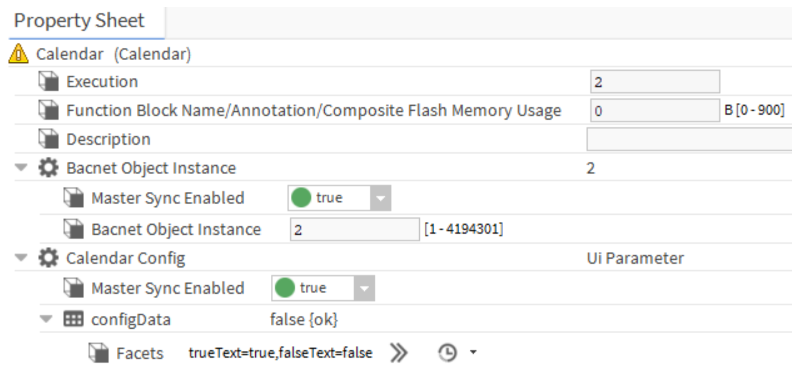


Fig. 353 Calendar Function Block and Property sheet

Parameters

Table 237 Parameters of Calendar Function

Parameter Name	Description
Calender Type	• **Date:** Various combinations of weekday, numerical date, month or month combinations, and year. • **Date Range:** Start and end range, using for each a combination of day, month, year. • **Week and Day:** Combination of day of week, week in month, month.
Right-click menus	• **Add:** Add a new calendar event (same as using Add button). • **Edit:** Edit day(s) selection criteria (but not changing calendar type). Same as Edit button. • **Rename:** Rename selected calendar event (same as using Rename button). • **Priority (up):** Move the calendar event up in the display list (same as using Priority button). • **Priority (down):** Move the calendar event down in the display list (same as using Priority button). • **Delete:** Removes the selected calendar event from the schedule component.

Example 1: Calendar Events

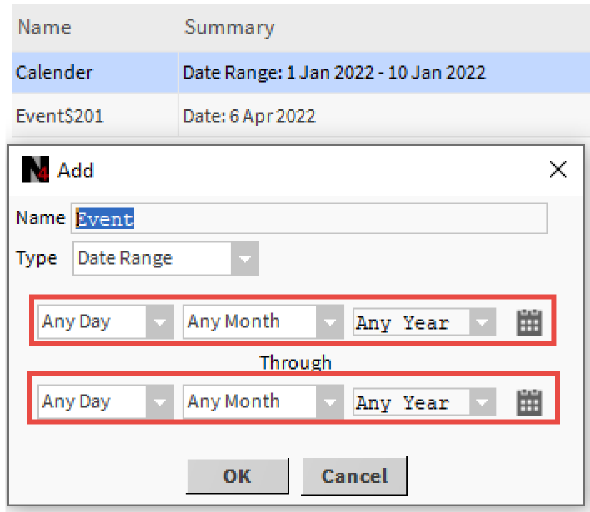


Fig. 354 Calendar Events

Current Date Time

The Current Date Time function block determines the current date, time, and some other useful values that can be configured for various outputs. This function block ensured that the outputs were completely correct for a DDC cycle, even if time synchronization occurred and the synchronization lasted longer than one DDC cycle.

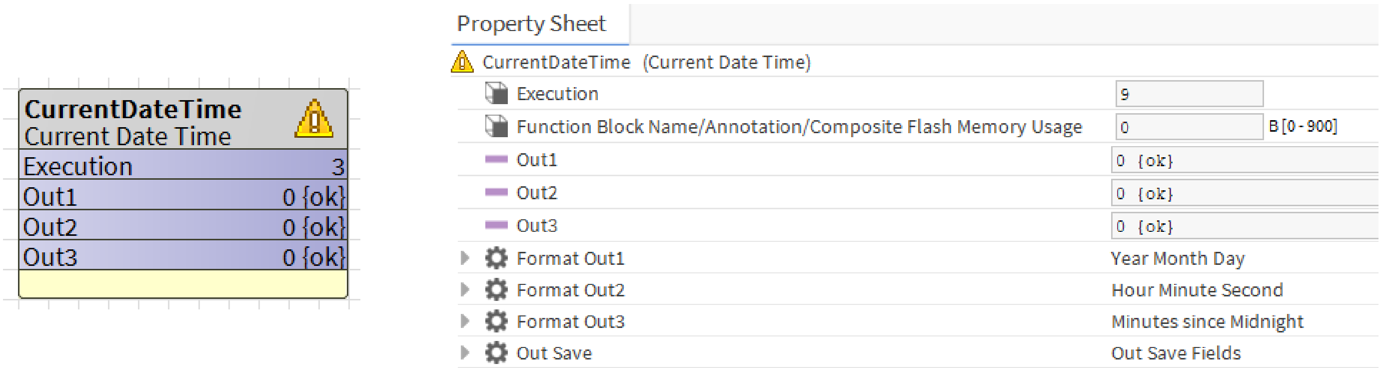


Fig. 355 CurrentDateTime Function Block and Property Sheet

Input

Table 238 Input of Current Date Time

Input Name	Description
NA	NA

Output

Table 239 Outputs of Current Date Time

Output Name	Description
Out1	The value depends on the **Format Out 1** parameter. **Numeric:** unassigned int value
Out2	The value depends on the **Format Out 2** parameter. **Numeric:** unassigned int value
Out3	The value depends on the **Format Out 3** parameter. **Numeric:** unassigned int value

Parameters

Table 240 Parameter of Current Date Time

Parameter Name	Description
Format Out 1	**Format: ss:** Seconds, **mm:** Minutes, **hh:** Hours, **MM:** Month, **YYYY:** Year **Default:** YYYYMMDD **Enums:** **1:** Always 0 :0 **2:** Second ss: 0-59 (1) **3:** Minutes mm: 0-59 (0) **4:** Minute Second mmss: 0-5959(1) **5:** Hour hh: 0-23 (0) **6:** Hour Minute hhmm: 0-2359 (0) **7:** Hour Minute Second hhmmss: 0-235959 (1) **8:** Day DD: 1-31 (1) **9:** Month MM: 1-12 (1) **10:** Month Day MMDD: 101-1231 (101) **11:** Year YYYY: 0-9999 (1970) **12:** Year Month YYYYMM: 0-999912 (197001) **13:** Year Month Day YYYYMMDD: 101-99991231 (19700101) **14:** Day of the Year: 1-366 (1) **15:** Minutes since Midnight: 0-1440 (0) **16:** Week of the Year ISO 8601: 1-53 (1) **17:** Week of the Year US: 1-53 (1) **18:** Day of the Week ISO 8601: 1-7 (4) (1=Monday) **19:** Day of the Week US: 1-7 (5) (1=Sunday) **20:** Daylight Saving Active: 0=Not active, 1=Active. (Not active) (*1) ***Note:*** *For implementation, use a Floating Output without decimal.*
Format Out 2	Same Enumeration as Format Out 1. **Default:** hhmmss
Format Out 3	Same Enumeration as Format Out 1. **Default:** Minutes since midnight
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out1:** To enable or disable the Out1 option. • **Out2:** To enable or disable the Out2 option. • **Out3:** To enable or disable the Out3 option.

Date And Time Operation

The Date And Time Operation function block can carry out a variety of operations with two dates and times. Three outputs that indicate the operation's outcome.

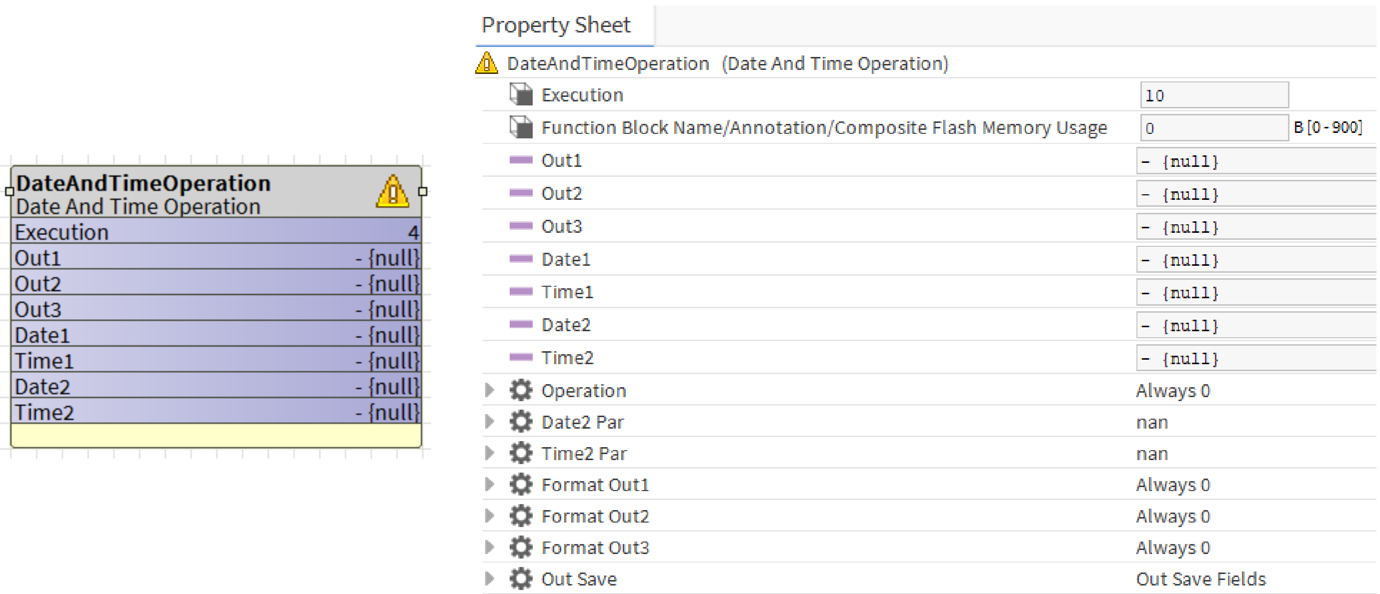


Fig. 356 DateAndTimeOperation Function Block and Property Sheet

Input

Table 241 Inputs of Date and Time Operation

Input Name	Description
Date1	The input should be in YYMMDD format, where YY - Year, MM - Month, and DD - Day. **Range:** **Day:** 1 - 31 **Month:** 1 - 12 **Year:** 0 - 9999 **Null:** Ignore the Date
Time1	The input Should be in the format hhmmss, where hh - hours, mm - minutes, and ss - seconds. **Range:** **Seconds:** 0 - 59 **Minutes:** 0 - 59 **Hours:** 0 - 23 **Null:** ignore the Time
Date2 (In & Par)	The input should be in YYMMDD format, where YY - Year, MM - Month, and DD - Day. **Range:** **Day:** 1 - 31 **Month:** 1 - 12 **Year:** 0 - 9999 **Null:** Ignore the Date

Table 241 Inputs of Date and Time Operation (Continued)

Input Name	Description
Time2 (In & Par)	The input Should be in the format hhmmss, where hh - hours, mm - minutes, and ss - seconds. **Range:** **Seconds:** 0 - 59 **Minutes:** 0 - 59 **Hours:** 0 - 23 **Null:** ignore the Time

Output

Table 242 Output of Date and Time Operation

Output Name	Description
Out1	The value depends on the **Format Out 1** parameter. Output is the floating value of 5 digits
Out2	The value depends on the **Format Out 2** parameter. Output is the floating value of 5 digits
Out3	The value depends on the **Format Out 3** parameter. Output is the floating value of 5 digits

Parameters

Table 243 Parameters of Date and Time Operation

Parameter Name	Description
Operations	If something is missing, either only the date or only the time are considered as operations. **Enum:** **1:** Always 0, means Out1, Out2, Out3 = 0 **2:** Subtract (Date2 Time2 - Date 1Time1). The result can be negative.
Date 2 Par	**Default:** Null = Ignore the Date.
Time 2 Par	**Default:** Null = Ignore the Time.

Table 243 Parameters of Date and Time Operation (Continued)

Parameter Name	Description
Format Out 1	The Output 1 is formatted and meaning is configured differently. • For integer values the decimals are truncated.(2.24324->2). • For Floating values, the value is shown with 5 digits. **Enum:** **00:** Always 0(Default) **01:** Second as integer value **02:** Minutes as integer value **04:** Hours as integer value **08:** Days as integer value **16:** Month as integer value **32:** Year as integer value **65:** Seconds as floating value **66:** Minutes as floating value **68:** Hours as floating value **72:** Days as floating value **80:** Months as floating value **96:** Year as floating value
Format Out 2	The output Out 2 is configured with different meanings and formatting. Same as **Enum** as **Format Out 1**. **Default:** 0 = Always 0
Format Out 3	The output Out 3 is configured with different meanings and formatting. Same as **Enum** as **Format Out 1**. **Default:** 0 = Always 0
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out1:** To enable or disable the Out1 option. • **Out2:** To enable or disable the Out2 option. • **Out3:** To enable or disable the Out3 option.

Example 1: DateAndTimeOperation (Time until next Filter Replacement)

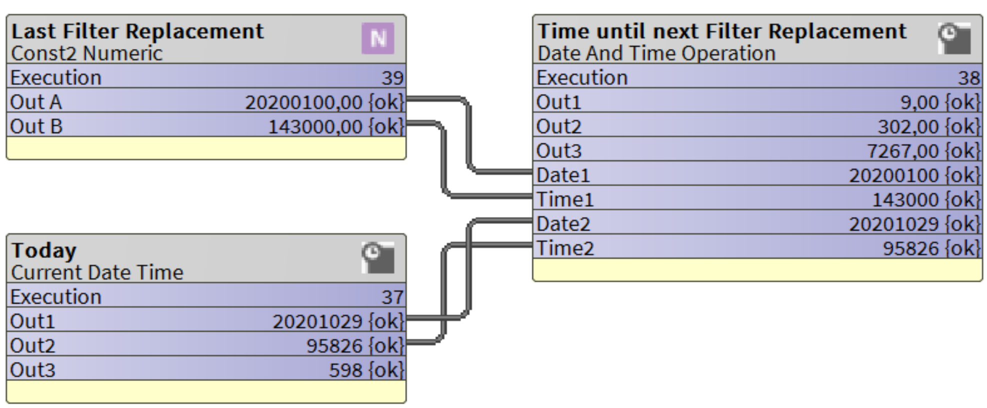


Fig. 357 DateAndTimeOperation Function Block

CHAPTER

13 UTIL FUNCTION BLOCKS

The honIrmControl Palette provides the following Util function blocks that can be configured and used to build the required application logic:

- Error
- Pass Thru
- Prev Value
- Const1 Numeric
- Const2 Numeric
- Const5 Numeric
- Const1 Boolean
- Const2 Numeric
- Const5 Numeric
- Save Permanent
- Evaluate Bacnet Status Flags
- Text A
- System A

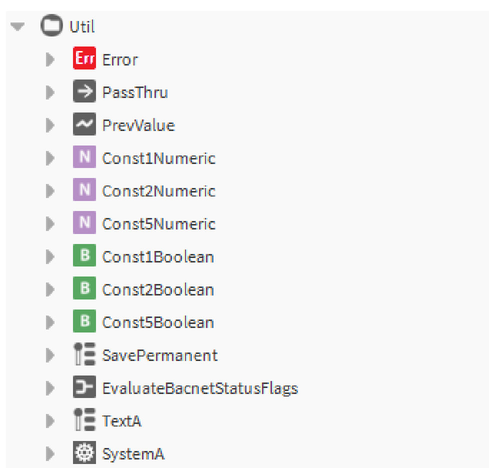


Fig. 358 Util Function Blocks

Error

The Error function block provides the error code to find an error in the system.

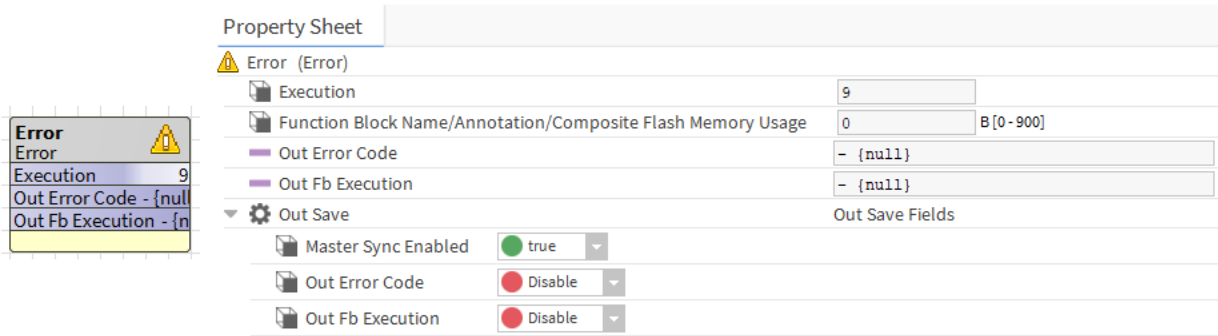


Fig. 359 Error Function Block and Property Sheet

Input

Table 244 Input of Error

Input Name	Description
Error Code	This is a 32-bit floating point input slot.
Fb Breadcrumb	This is a 32-bit floating point input slot.

Error Code:

- **fbExecOK:** 0
- **fbExecDivByZero:** 1, Division by zero, or some other asymptotically infinite result (pole error).
- **fbExecOverflow:** 2, The result is too large in magnitude to be represented as a value of the return type.
- **fbExecUnderflow:** 3, The result is too small in magnitude to be represented as a value of the return type.
- **fbExecRangeError:** 4, An input exceeded the permitted range (For example an input selector is too small or too big).
- **fbExecNoValidInputs:** 5, All inputs are null.
- **fbExecCannotCalc:** 6, It cannot provide a result because not all necessary inputs are valid.

Parameter

Table 245 Parameter of Error

Parameter Name	Description
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out Error Code**: To enable or disable the **Out Error Code** feature. • **Out Fb Execution**: To enable or disable the **Out Fb Execution** feature.

Examples

Example 1: Error

Error code 1 - Division by zero, or some other asymptotically infinite result.

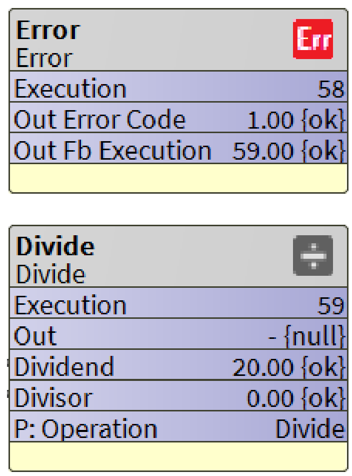


Fig. 360   Error code 1 - Division by zero or some other asymptotically infinite result

Example 2: Error

Error code 5 - Cannot provide a result because not all necessary inputs are valid.

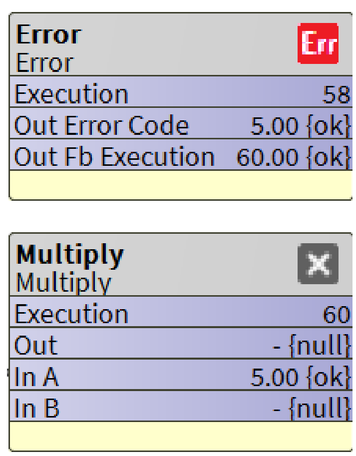


Fig. 361   Error code 5 - Cannot provide a result because not all necessary inputs are valid

Pass Thru

The Pass Thru function block provides an input and output slot to the application block, allowing it to connect inputs and outputs.

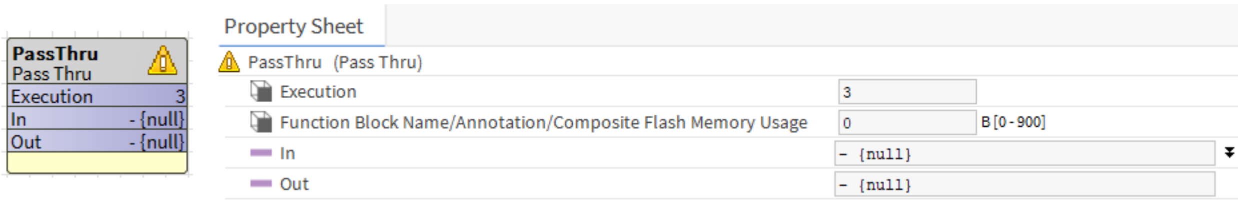


Fig. 362 PassThru Function Block and Property Sheet

Input

Table 246 Input of Pass Thru

Input Name	Description
In	This is a 32-bit floating point input slot.

Output

Table 247 Output of Pass Thru

Output Name	Description
Out	Average value

Example 1: PassThru

It takes the average of two temperature inputs and transfers the average value to the network outputs. Slots do not exist in the application block. The PassThru block help in the creation of slots, and the user can connect the application block's inputs and outputs to the slots that are created.

It passes the input from physical IOsPre or different logics.

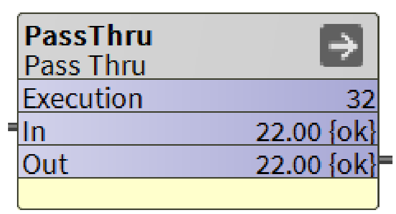


Fig. 363 PassThru Function Block

Prev Value

The Prev Value function block is used to determine the change from the previous DDC cycle.

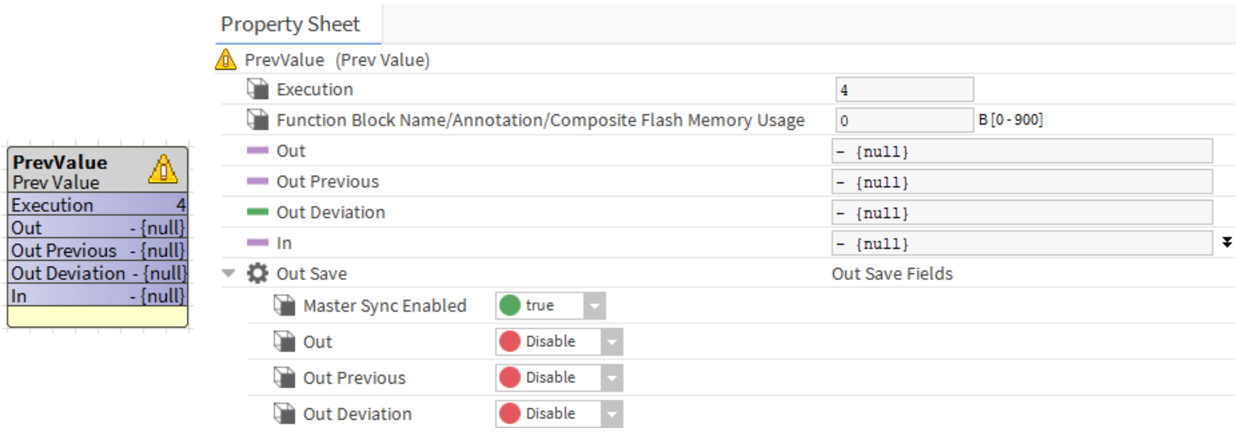


Fig. 364 PrevValue Function Block and Property Sheet

Input

Table 248 Input of Prev Value

Input Name	Description
In	This is a 32-bit floating point input slot.

Output

Table 249 Output of Prev Value

Output Name	Description
Out	In
Out Previous	Value from the previous DDC cycle from **In**.
Out Deviation	Value from the deviated DDC cycle from **In**.

Parameter

Table 250 Parameter of Prev Value

Parameter Name	Description
Out Save	• **Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out**: To enable or disable the **Out** feature. • **Out Previous**: To enable or disable the **Out Previous** feature. • **Out Deviation**: To enable or disable the **Out Deviation** feature.

Example 1: PrevValue

If any change in **In**, the **Out Previous** is set to the previous input value, and the **Out Deviation** is set to "true."

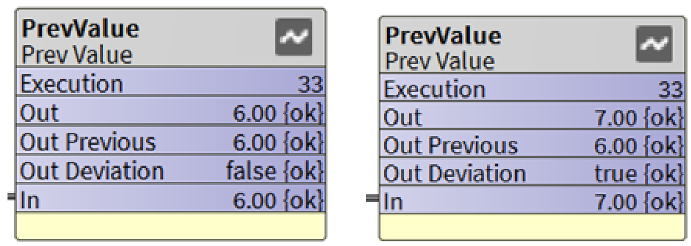


Fig. 365 PrevValue Function Block

Const1 Numeric

The Const1Numeric function block is a non-physical input that can be used to create the application logic in the wiresheet.

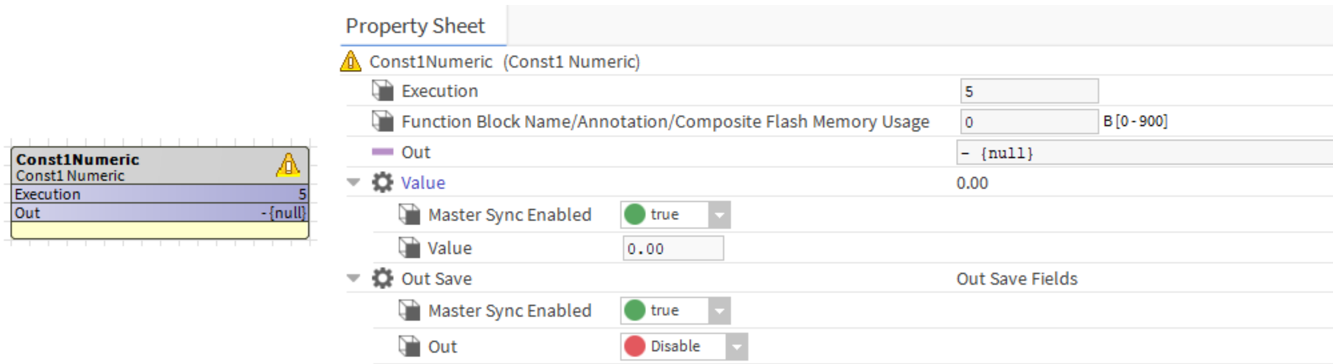


Fig. 366 Const1Numeric Function Block and Property Sheet

Output

Table 251 Output of Const1 Numeric

Output Name	Description
Out	This is a 32-bit floating point input slot.

Parameter

Table 252 Parameter of Const1 Numeric

Parameter Name	Description
Value	This is the input value.
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature.

Example 1: Const1Numeric

Provide a single numeric value as output.

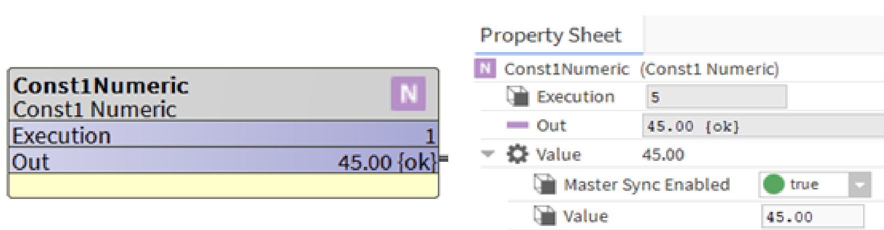


Fig. 367 Const1Numeric Function Block and Property Sheet

Const2 Numeric

The Const2Numeric function block is a non-physical input that can be used to create the application logic in the wiresheet. It has two outputs.

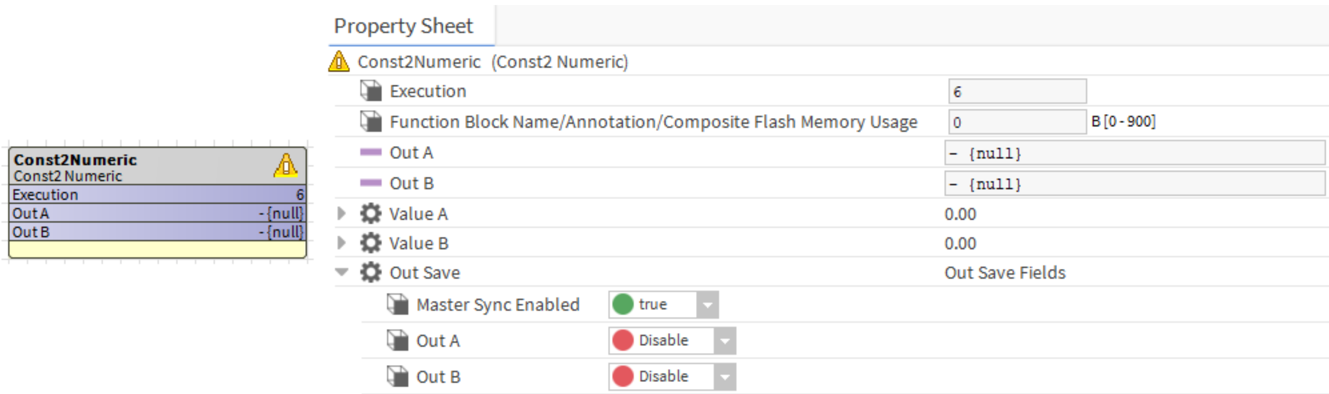


Fig. 368 Const2Numeric Function Block and Property Sheet

Output

Table 253 Output of Const2 Numeric

Output Name	Description
Out A	This is a 32-bit floating point input slot.
Out B	This is a 32-bit floating point input slot.

Parameter

Table 254 Parameter of Const2 Numeric

Parameter Name	Description
Value A	This is the input value.
Value B	This is the input value.
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out A**: To enable or disable the **Out A** feature. • **Out B**: To enable or disable the **Out B** feature.

Example 1: Const2Numeric

Provide two defined numeric values as output.

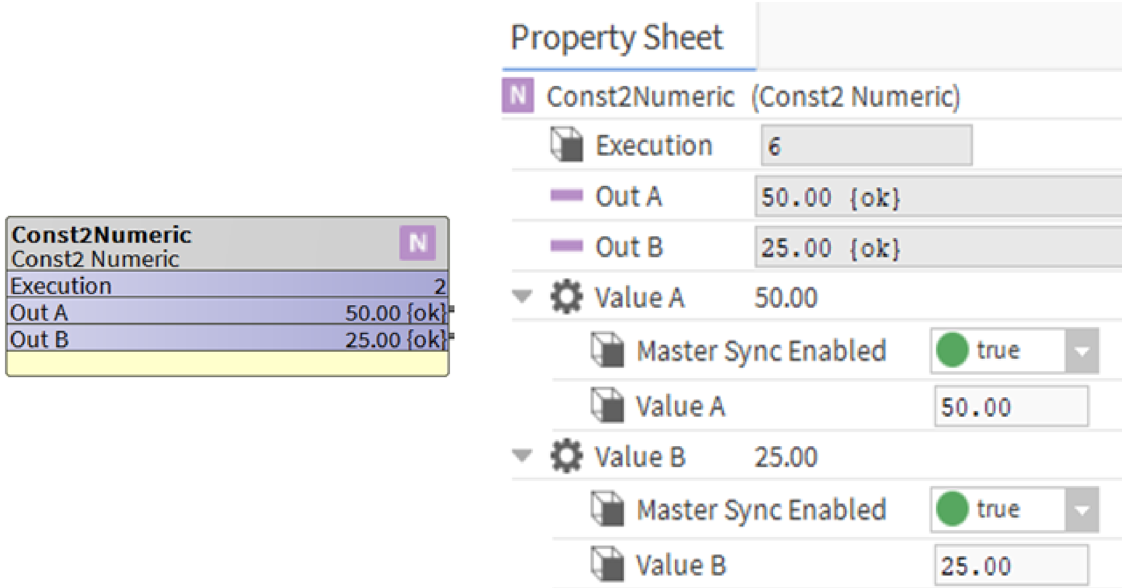


Fig. 369 Const2Numeric Function Block and Property Sheet

Const5 Numeric

The Const5 Numeric function block is a non-physical input that is not visible to the network. The Const5 Numeric can be configured and used while creating the application logic in the BACnet. It has five output slots.

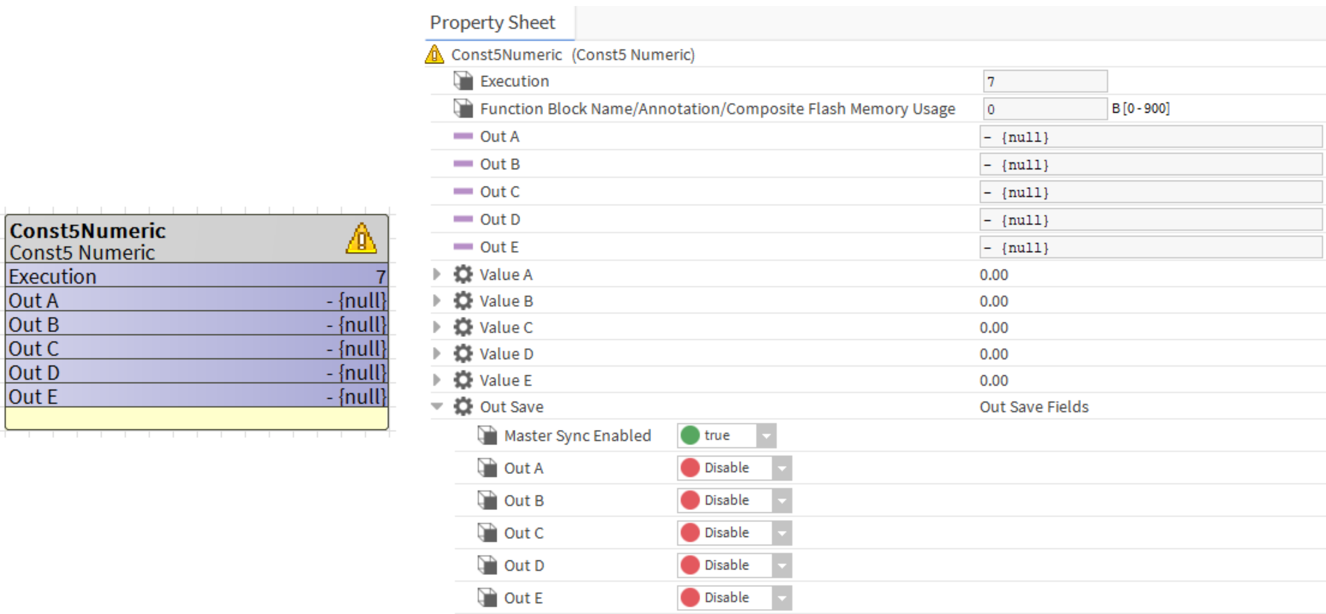


Fig. 370 Const5Numeric Function Block and Property Sheet

Output

Table 255 Output of Const5 Numeric

Output Name	Description
Out A	This is a 32-bit floating point input slot.
Out B	This is a 32-bit floating point input slot.
Out C	This is a 32-bit floating point input slot.
Out D	This is a 32-bit floating point input slot.
Out E	This is a 32-bit floating point input slot.

Parameter

Table 256 Parameter of Const5 Numeric

Parameter Name	Description
Value A	This is the input value.
Value B	This is the input value.

Table 256 Parameter of Const5 Numeric (Continued)

Parameter Name	Description
Value C	This is the input value.
Value D	This is the input value.
Value E	This is the input value.
Out Save	• **Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out A**: To enable or disable the **Out A** feature. • **Out B**: To enable or disable the **Out B** feature. • **Out C**: To enable or disable the **Out C** feature. • **Out D**: To enable or disable the **Out D** feature. • **Out E**: To enable or disable the **Out E** feature.

Example 1: Const5Numeric

Provide five defined numeric values as output.

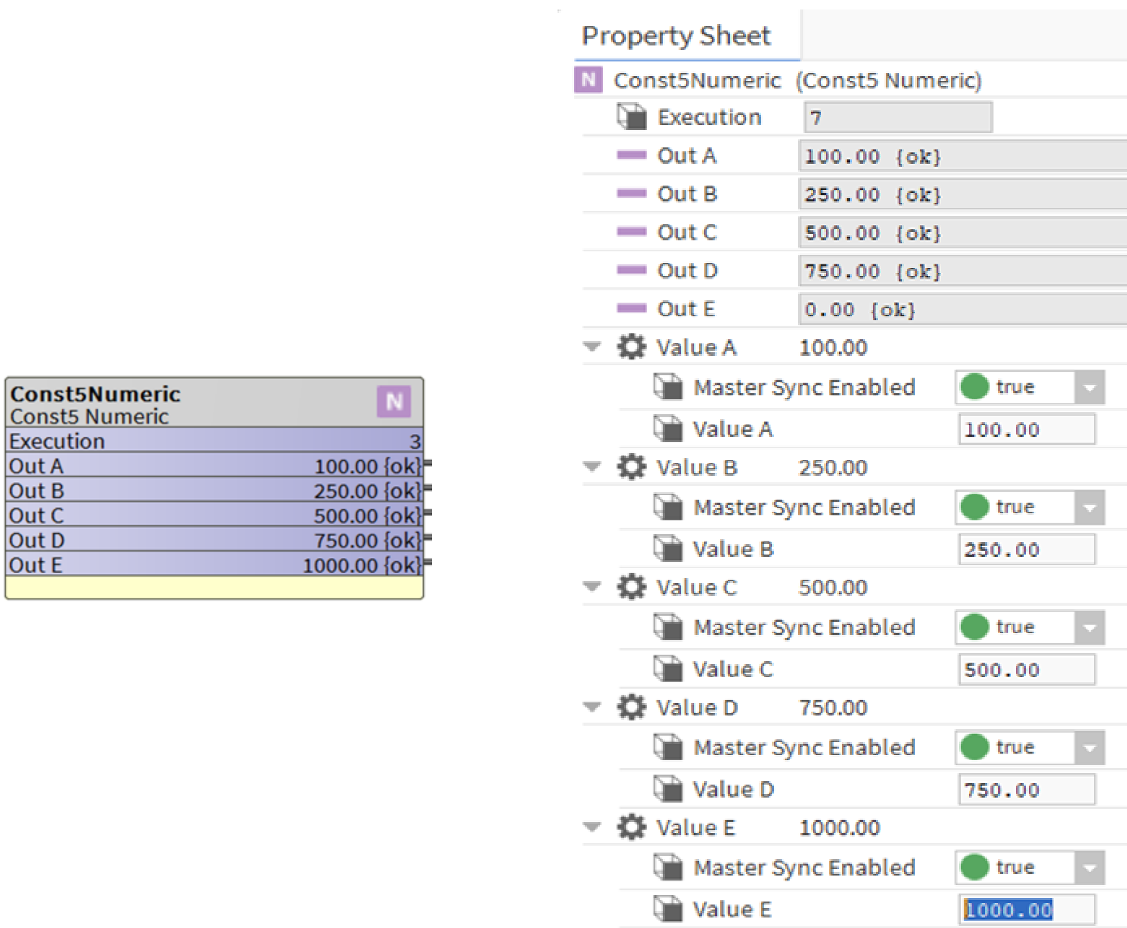


Fig. 371 Const5Numeric Function Block and Property Sheet

Const1 Boolean

The Const1 Boolean function block is a non-physical input that is not visible to the network. The Const1 Boolean can be configured and used while creating the application logic in the BACnet. It accepts only the Boolean data.

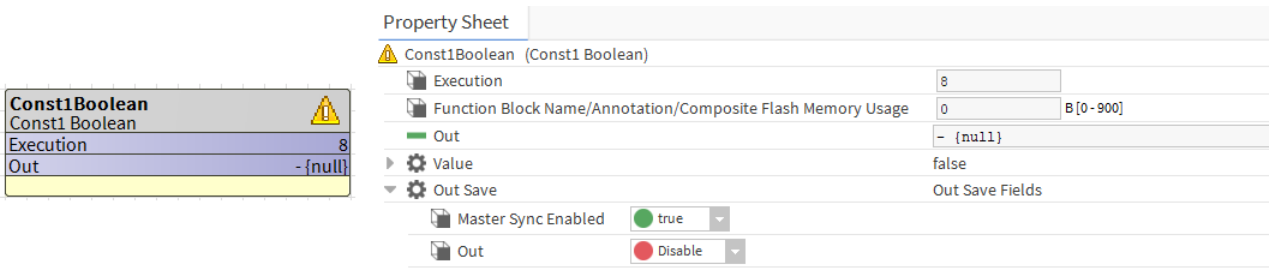


Fig. 372 Const1Boolean Function Block and Property Sheet

Output

Table 257 Output of Const1 Boolean

Output Name	Description
Out	This is a 32-bit floating point input slot.

Parameter

Table 258 Parameter of Const1 Boolean

Parameter Name	Description
Value	This is the input value.
Out Save	• **Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out**: To enable or disable the **Out** feature.

Example 1: Const1Boolean

Provide one defined boolean value as output.

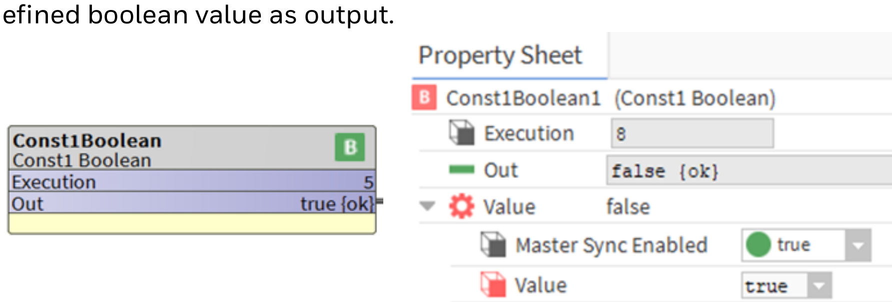


Fig. 373 Const1Boolean Function Block and Property Sheet

Const2 Boolean

The Const2 Boolean function block is a non-physical input that is not visible to the network. The Const2 Boolean can be configured and used while creating the application logic in the BACnet.

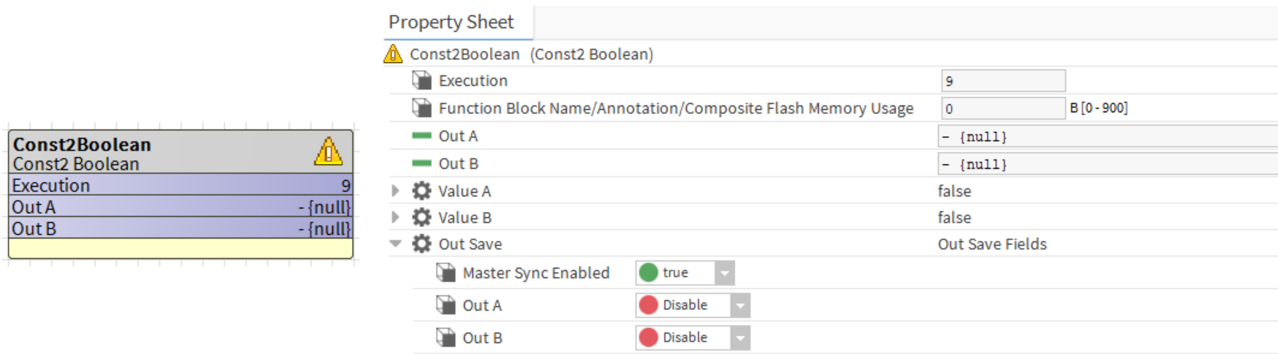


Fig. 374 Const2Boolean Function Block and Property Sheet

Output

Table 259 Output of Const2 Boolean

Output Name	Description
Out A	This is a 32-bit floating point input slot.
Out B	This is a 32-bit floating point input slot.

Parameter

Table 260 Parameter of Const2 Boolean

Parameter Name	Description
Value A	This is the input value.
Value B	This is the input value.
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out A**: To enable or disable the **Out A** feature. • **Out B**: To enable or disable the **Out B** feature.

Example 1: Const2Boolean

Provide two defined boolean values as output.

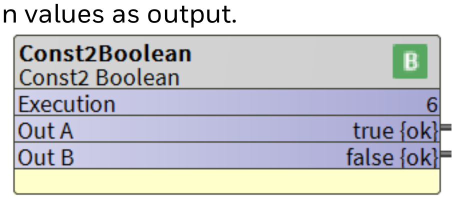


Fig. 375 Const2Boolean Function Block

Const5 Boolean

The Const5 Boolean function block is a non-physical input that is not visible to the network. The Const5 Boolean can be configured and used while creating application logic in BACnet. It has five output slots.

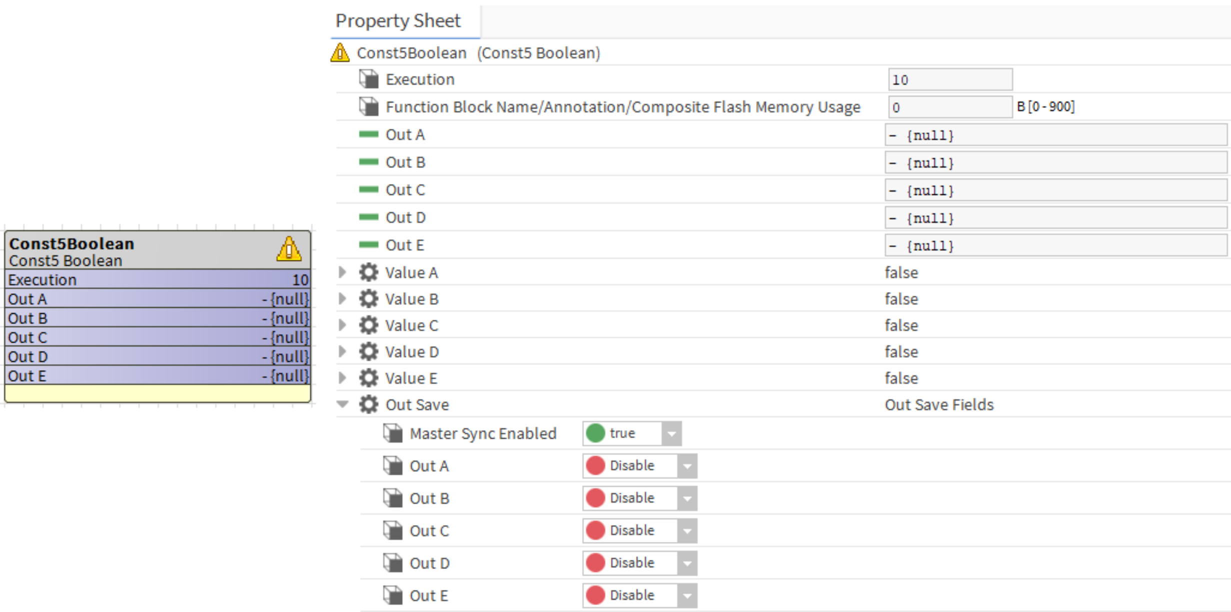


Fig. 376 Const5Boolean Function Block and Property Sheet

Outputs

Table 261 Outputs of Const5 Boolean

Output Name	Description
Out A	This is a 32-bit floating point input slot.
Out B	This is a 32-bit floating point input slot.
Out C	This is a 32-bit floating point input slot.
Out D	This is a 32-bit floating point input slot.
Out E	This is a 32-bit floating point input slot.

Parameter

Table 262 Parameter of Const5Boolean

Parameter Name	Description
Value A	This is the input value.
Value B	This is the input value.
Value C	This is the input value.

Table 262 Parameter of Const5Boolean (Continued)

Parameter Name	Description
Value D	This is the input value.
Value E	This is the input value.
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out A**: To enable or disable the **Out A** feature. • **Out B**: To enable or disable the **Out B** feature. • **Out C**: To enable or disable the **Out C** feature. • **Out D**: To enable or disable the **Out D** feature. • **Out E**: To enable or disable the **Out E** feature.

Example 1: Const5Boolean

Provide five defined boolean values as output.

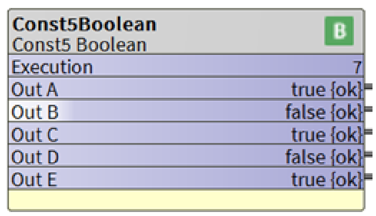


Fig. 377 Const5Boolean Function Block

Save Permanent

The Save Permanent function block is used to save values in non-volatile memory (for example, flash). The value is not lost when the supply voltage is turned off. When the power is turned back on, the last saved value is restored in the first DDC cycle; from the second DDC loop on, SavePermanent can save the value into non-volatile memory.

Save Permanent function block can be used to save a measured value (energy consumption value, motor running time) or a parameter written via BACnet.

The controller has no battery to store the data. Instead, it stores the data in the flash memory. This data is never lost, no matter how long the controller was turned off. But, flash memory has a limited number of write cycles. Therefore, an intelligent solution is needed to store the data that changes permanently in a flash.

The Save Permanent function block behaves differently in controllers that detect voltage interruptions and controllers that do not detect voltage interruptions. Because of this distinction, the same parameters in SavePermanent can be used for both controller types, requiring no parameter changes.

The following applies to the controllers that can detect a voltage interruption:

- If the supply voltage is interrupted, there is still enough time to save the current value in a flash.
- There is no save to the flash in case of a significant change. The parameter Delta For Writing and Max Writes Per Week are ignored.
- A periodic save can be configured with the parameter Forced Write Period so that the value is not completely lost in the event of a software crash.

The following applies to the controllers that cannot detect a voltage interruption:

- If the input value has changed from the saved value by more than the value configured via Delta For Writing, then the value is saved again.
- A periodic save can be configured with the parameter Forced Write Period so that the small changes are not lost.
- Max Writes Per Week ensures that not too many writes occur over Delta For Writing and Forced Write Period to protect the flash memory.

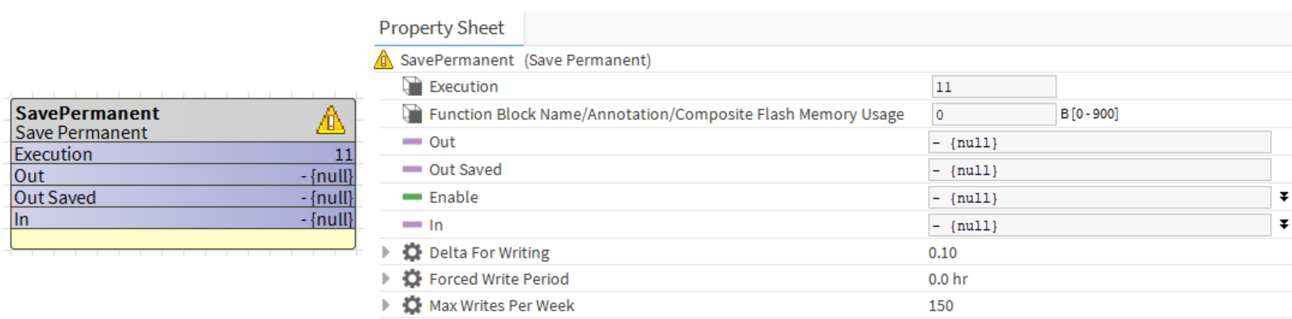


Fig. 378 SavePermanent Function Block and Property Sheet

Example for a BACnet Parameter.

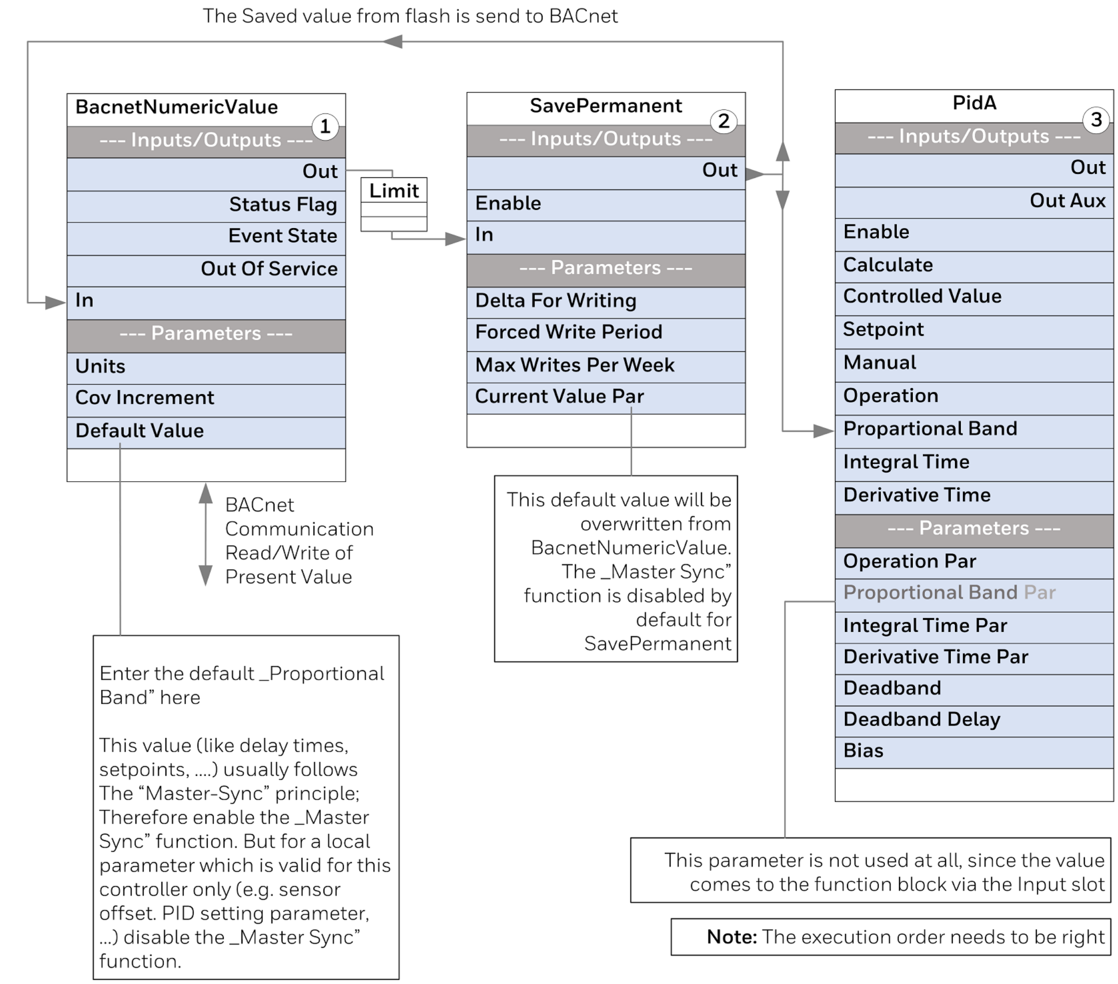


Fig. 379 Example for a BACnet Parameter

Input

Table 263 Input of Save Permanent

Input Name	Description
Enable	This enables or disables the complete saving. The automatic saving in the event of a controller power supply failure is disabled. **0:** Do not save **1:** null = Saving is active
In	This is a 32-bit floating point. The value of this input is saved in the non-volatile memory under the described conditions. The value is saved permanently, even if the controller is not supplied with power.

Output

Table 264 Output of Save Permanent

Output Name	Description
Out	In the first DDC cycle after power-up, **Out** displays the value of the last stored In value. From the 2nd DDC cycle, the output shows the input value, independent of whether the value is saved in a flash. **Numeric:** 32-Bit (float, integer), Null
Out Saved	The output shows the last stored value after power-up and during the normal DDC loop. This makes it easy to see if the save into the flash was successful. The behavior is the same for controllers that support the automatic saving in the event of a controller power supply failure. **Numeric:** 32-Bit (float, integer), Null

Parameter

Table 265 Parameter of Save Permanent

Parameter Name	Description
Delta For Writing	For controllers that support automatic saving in the event of a controller power supply failure, **Delta For Writing** is ignored. For controllers that cannot detect a voltage interruption: The value entered through In is saved if it differs from the last saved value by this delta value. The value 0 means that every change is saved. A “null” value means **Save Permanent** function block won’t save any value. ***Note:*** *There are memories with a limited lifetime where the number of write cycles is limited (IRM: 100000). If the function block is used to save a BACnet parameter, use 0.01 to save every change.* The default value is “0.1.”

Table 265 Parameter of Save Permanent (Continued)

Parameter Name	Description
Forced Write Period	The value entered through **In** is saved if the time set elapses after the last saving or power-up. It does not matter whether the saving was caused by **Delta For Writing** or by the elapsed time; the time is restarted with each saving. It ensures that even small value changes are saved periodically. The value 0 deactivates the periodic saving. ***Note:*** *There are memories with a limited lifetime where the number of write cycles is limited (IRM: 100000). If the function block is used to save a BACnet parameter, use 0.0 hours (no periodic saving).* For controllers that support automatic saving in the event of a controller power supply failure, **Forced Write Period** could be disabled (0 hours). However, to ensure that the value is not completely lost in the event of a software crash. **Forced Write Period** is configurable. **Range:** 0-8760 hours, **Default:** 0.0 hours
Max Writes Per Week	For controllers that support automatic saving in the event of a controller power supply failure, **Max Writes Per Week** is ignored. For controllers that cannot detect a voltage interruption: There are memories with a limited lifetime where the number of saving cycles is limited. With the IRM controllers, the maximum number of saving cycles over the device's lifetime is 100000. Due to the data's blockwise storage of data, it is recommended that the controller use one write cycle per day (7/week) for continuously changing values, such as counters, measured values, etc. If the function block is used to save a BACnet parameter, then it is assumed that the value changes during commissioning but not in daily operation. In this case, significantly higher values can be entered. The internal counter for **Max Writes Per Week** starts with 0 after power-up. **Default:** 150
Current Value Par	This value represents the saved value. When saving measured values (energy consumption, motor runtime, etc.), disable the **Master Sync** function to prevent the measured value of the synchronized devices from being overwritten by the master measured value. If the function block is used to save a BACnet parameter, then there is no need to write any default value because the default value comes from the BACnet function block. Also, the **Master Sync** function needs to be disabled. The parameter default value is entered in the BACnet function block.

Example 1: Save Permanent

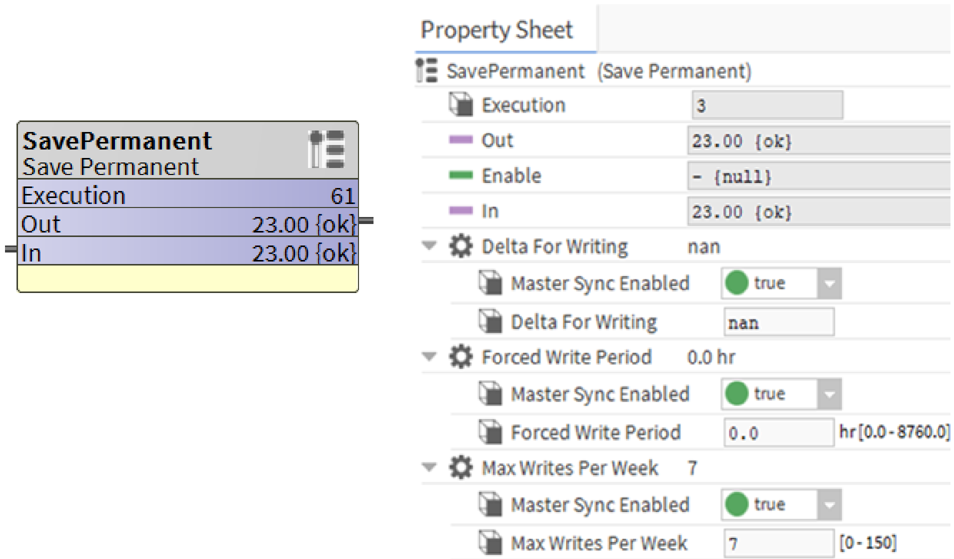


Fig. 380 Save Permanent Function Block and Property Sheet

Evaluate Bacnet Status Flags

The Evaluate Bacnet Status Flags function block is used to read the attribute of the BACnet points.

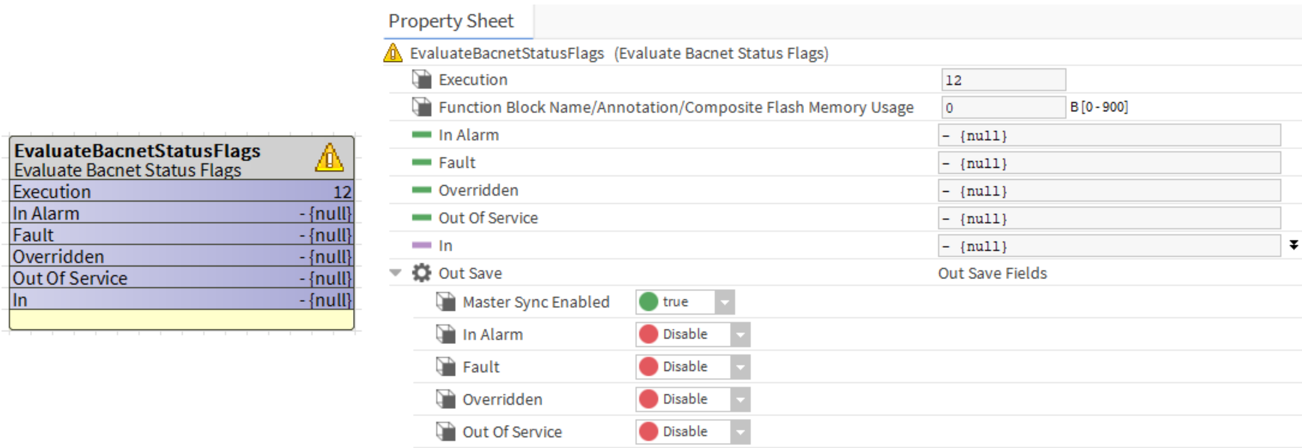


Fig. 381 EvaluateBacnetStatusFlag Function Block and Property Sheet

Input

Table 266 Input of Evaluate Bacnet Status Flag

Input Name	Description
In	Inputs will be any Bacnet input or output points status flag. • Bacnet Numeric input • Bacnet Boolean Input • Bacnet Numeric Output • Bacnet Boolean Output • Bacnet Enum Output • Bacnet Numeric Value • Bacnet Boolean Value • Bacnet Enum Value MV (Multi State)

Output

Table 267 Output of Evaluate Bacnet Status Flag

Output Name	Description
In Alarm	If the point is in alarm, the attribute will read an alarm as “true.” The alarm will be High limit, Low limit, and Open Loop.

Table 267 Output of Evaluate Bacnet Status Flag (Continued)

Output Name	Description
Fault	If the point is in invalid value or zero, the attribute will read a fault as "true."
Overridden	If the point is overridden, the attribute will read an overridden as "true."
Out of Service	If the point is Out of Service, the attribute will read an Out of Service as "true."

Parameter

Table 268 Parameter of Evaluate Bacnet Status Flag

Parameter Name	Description
Out Save	• **Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **In Alarm**: To enable or disable the **In Alarm** feature. • **Fault**: To enable or disable the **Fault** feature. • **Overridden**: To enable or disable the **Overridden** feature. • **Out of Service**: To enable or disable the **Out of Service** feature.

Examples

Example 1: EvaluateBacnetStatusFlag (Normal status)

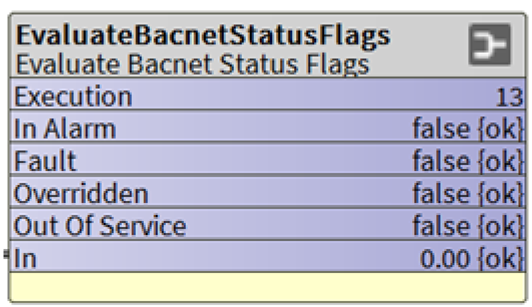


Fig. 382   EvaluateBacnetStatusFlag – Normal status

Example 2: EvaluateBacnetStatusFlag (Alarm status)

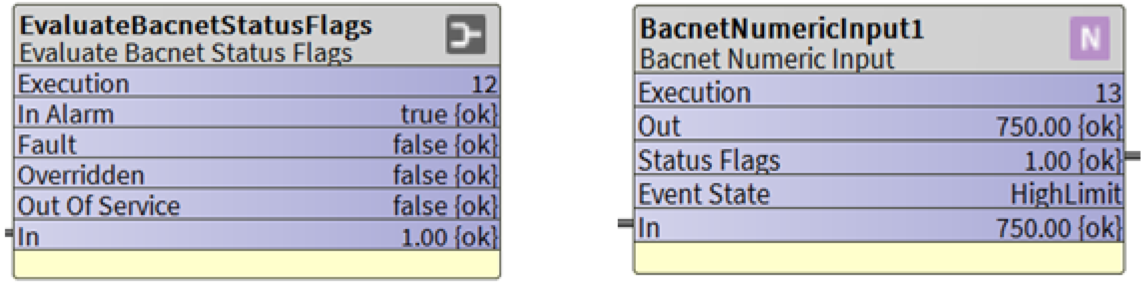


Fig. 383 EvaluateBacnetStatusFlag - Alarm status

Example 3: EvaluateBacnetStatusFlag (Fault status)

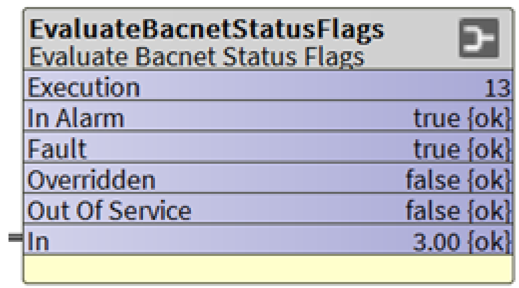


Fig. 384 EvaluateBacnetStatusFlag - Fault status

Example 4: EvaluateBacnetStatusFlag (Out of Service status)

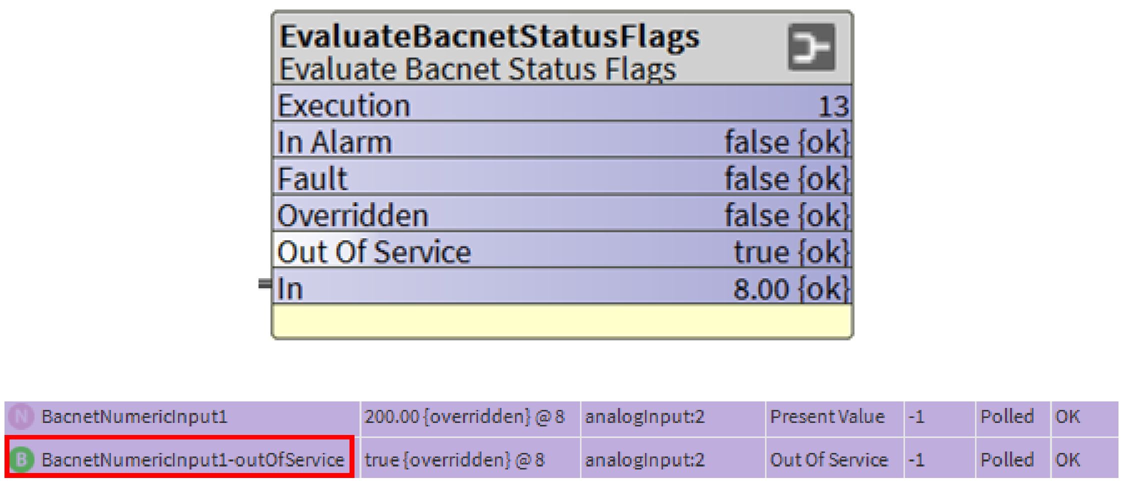


Fig. 385 EvaluateBacnetStatusFlag - Out Of Service status

Text A

The Text A function block is used to add the text about the program or application.

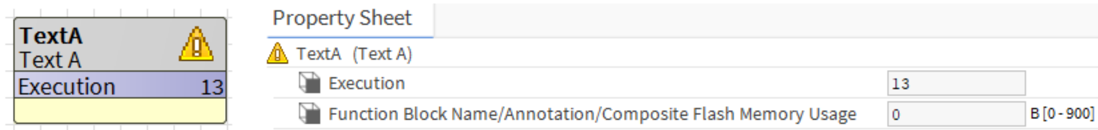


Fig. 386 TextA Function Block

Steps to enter the text:

Step 1. Right-click on the Text A and select **Actions > Annotation.**

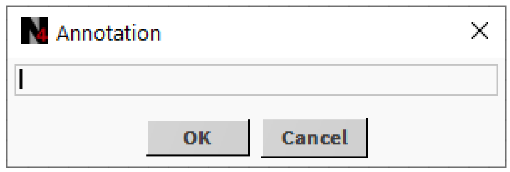


Fig. 387 Annotation box

Step 2. Enter the text and click **OK**.

Example 1: TextA

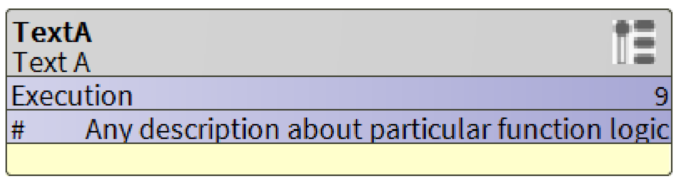


Fig. 388 TextA Function Block

System A

The System A function block is used to provide system relevant information for the application.

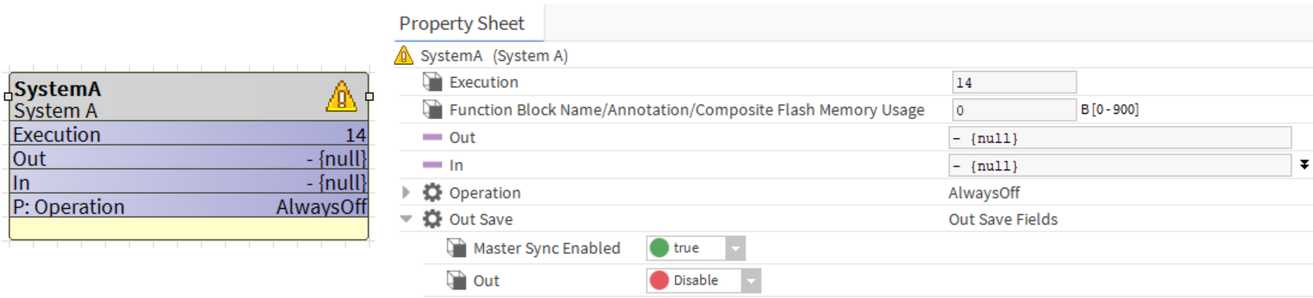


Fig. 389 SystemA Function Block and Property Sheet

Input

Table 269 Input of System A

Input Name	Description
In	This is a 32-bit floating point input slot. It can be used in the future to trigger a system event (sending serial).

Output

Table 270 Output of System A

Output Name	Description
Out	The numeric value depends on the operation.

Parameter

Table 271 Parameter of System A

Parameter Name	Description
Operation	Always Off: Output is ''false.'' Always On: Output is ''true.'' **EngUnitTemperature**: It measures the unit type (SI-Metric or Imperial). • If the measurement type is SI-Metric in the control manager, the value is "0." • If the measurement type is Imperial, the value is "1." **Powerup**: If the controller is powered up, it gives one cycle input.

Table 271 Parameter of System A (Continued)

Parameter Name	Description
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out**: To enable or disable the **Out** feature.

Examples

Example 1: SystemA (Operation set to AlwaysOff)

System A operation is AlwaysOff.

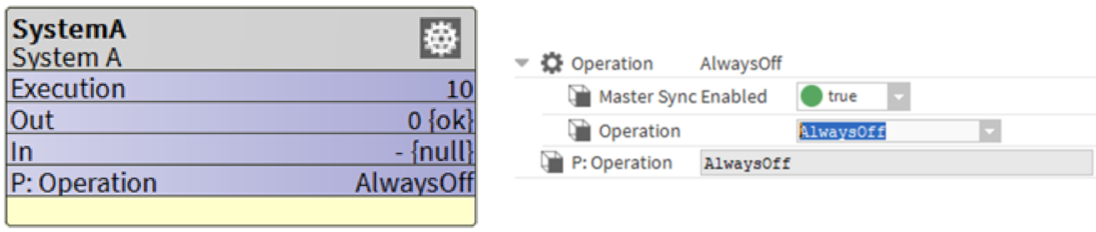


Fig. 390 SystemA Function Block and Property sheet (Always Off)

Example 2: SystemA (Operation set to AlwaysOff)

System A operation is AlwaysOn.

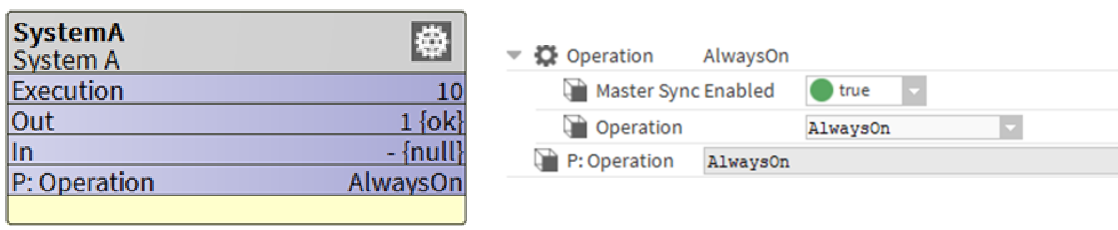


Fig. 391 SystemA Function Block and Property sheet (Always On)

Example 3: SystemA (Operation set to EngUnitTemperature)

System A operation is EngUnitTemperature set to Imperial.

When the measurement unit is set as Imperial, the output of systemA block is “1.”

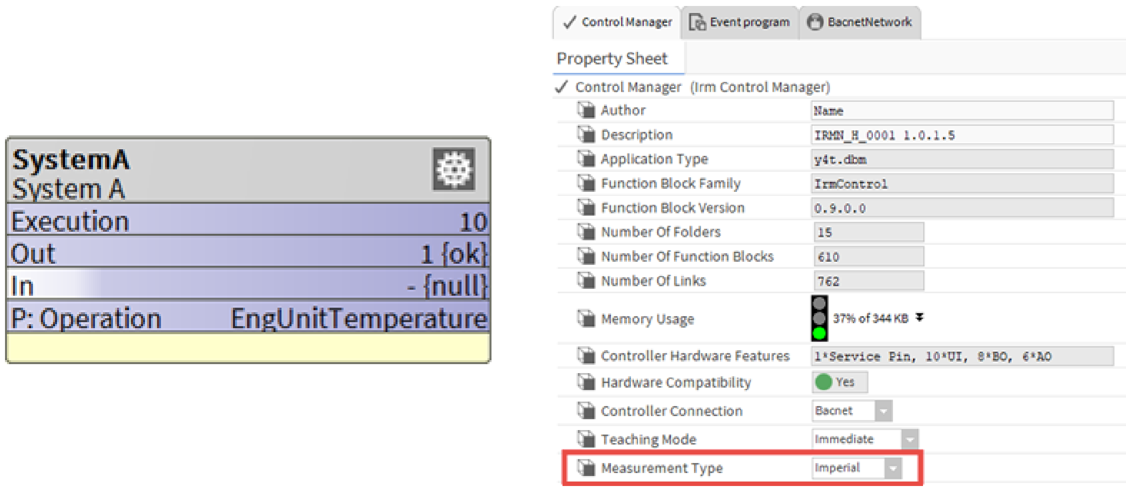


Fig. 392 SystemA Function Block and Property sheet (EngUnitTemperature)

Example 4: SystemA (Operation set to EngUnitTemperature)

System A operation is EngUnitTemperature set to SI-Metric.

When the measurement unit is set as SI-Metric, the output of systemA block is “0.”

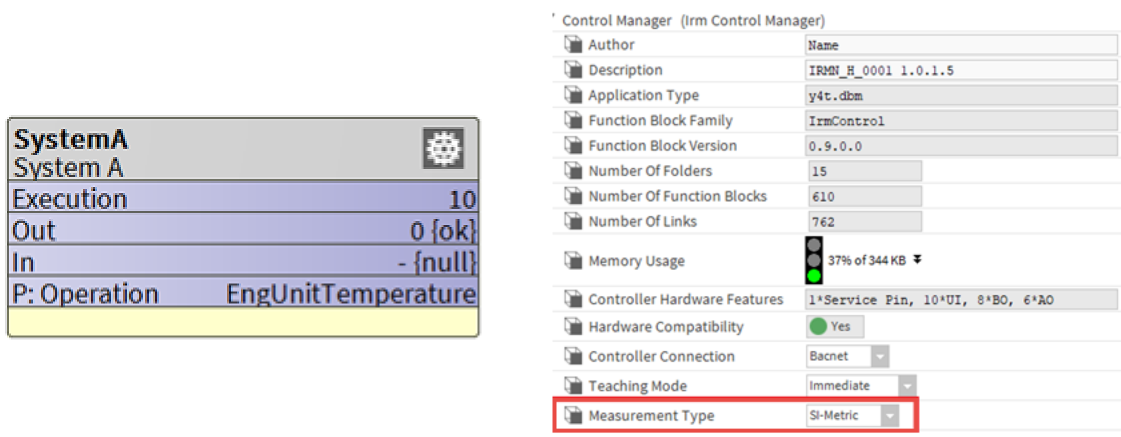


Fig. 393 SystemA Function Block and Property sheet (EngUnitTemperature)

CHAPTER

14

IRMWSTEXTBLOCK FUNCTION BLOCK

IrmWsTextBlock

This function block is a component that you can drop onto a wire sheet and position to add text notes. It allows to customize the graphics to add better text in the application.

Note: *Irm engineering tools support Niagara's WsTextBlock under name IRMWsTextBlock.*

For more details, refer to Niagara help guide"local:|module://docDeveloper/doc/baja-rt/javax/baja/util/BWsTextBlock.bajadoc".

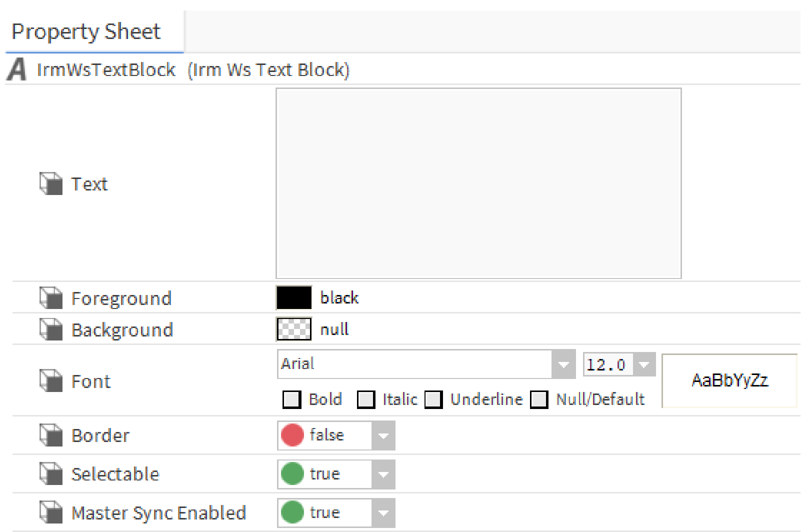


Fig. 394 IrmWsTextBlock Property Sheet

Input

Table 272 Input of IrmWsTextBlock

Input Name	Description
Text	Specifies the text on the label.
Foreground	Specifies foreground fill. Null indicates no foreground fill.

Table 272 Input of IrmWsTextBlock

Input Name	Description
Background	Specifies background fill. Null indicates no foreground fill.
Font	Configures the additional parameters for the font on the label.
Border	Displays the border design.
Selectable	If true allows to select the textbox on wiresheet. If false disallows to select the text box on wiresheet.
Master Sync Enabled	If true allows all the properties to sync from master to target. If false disallows all the properties to sync from master to target.

CHAPTER

15 OUTPUTS FUNCTION BLOCKS

The following Outputs function blocks are available in the honIrmControl Palette that can be configured and used to create the required application logic:

- Floating
- Pwm
- Stg123 Outp

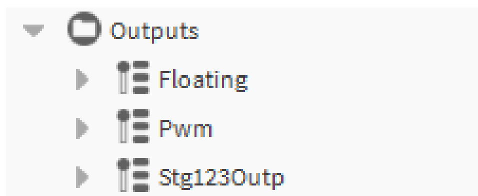


Fig. 395 Outputs Function Blocks

Floating

The Floating function block converts a 0 - 100 % position into OPEN and CLOSE running times, which are output on the digital outputs. A floating actuator is controlled via two digital outputs.

- The actuator moves the valve or damper to the OPEN position via the first output.
- The other output moves the valve or damper to the CLOSED position.

If neither of the two outputs is controlled, the actuator remains in the current position. Maximum eight floating function blocks can be used in an application.

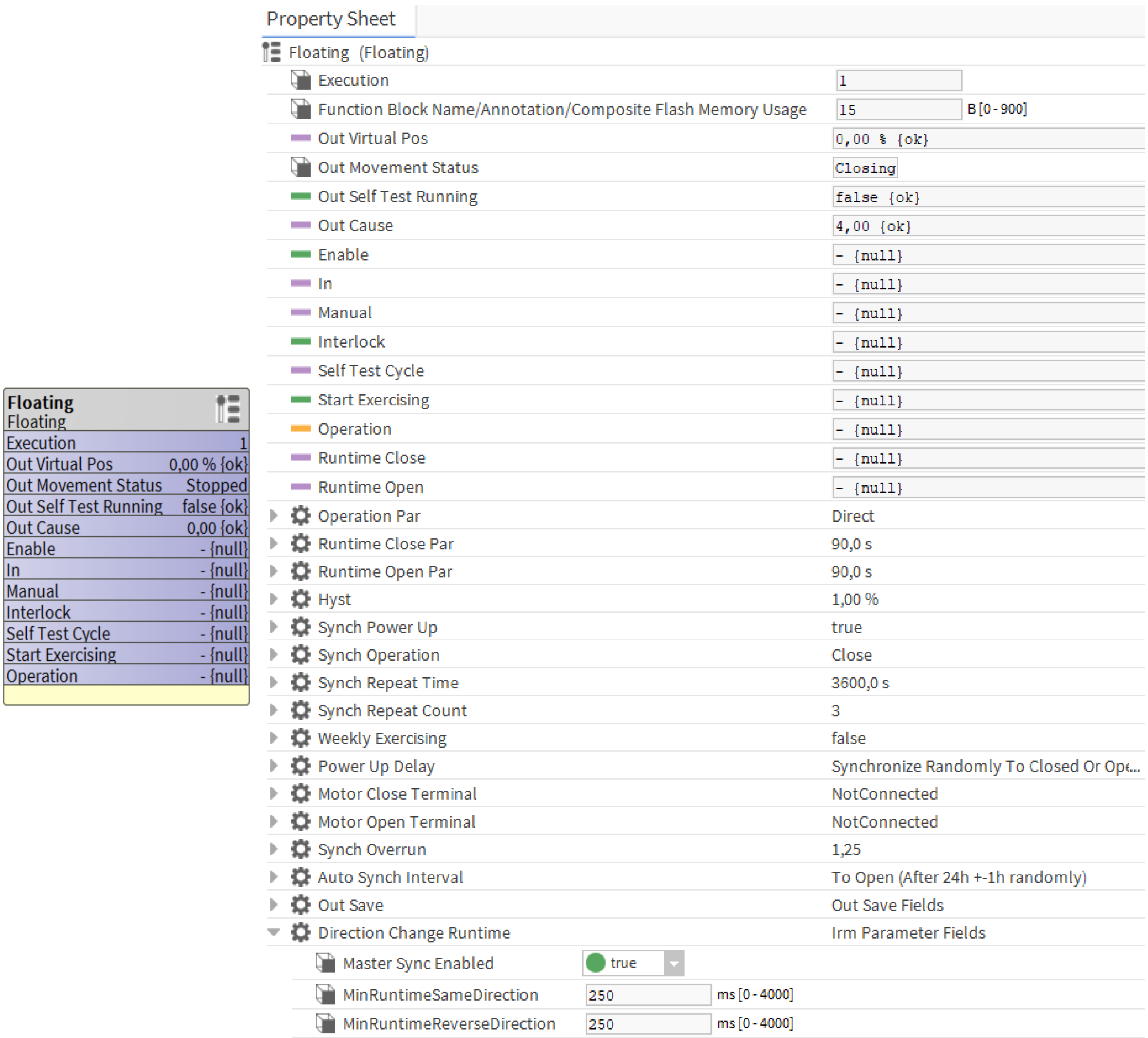


Fig. 396   Floating Function Block and Property Sheet

Input

Table 273 Input of Floating

Input Name	Description
Enable	This is a safety feature to switch a running motor to OFF immediately. • If **Enable** = true, then the power-up synchronization (see Synch Power Up) is performed to bring the motor model together with the valve / damper position. • If **Enable** = false, it works like an "emergency stop switch": The motor stops immediately, Out Virtual Pos = Null, and the motor model is no longer calculated. **Enable** has the highest priority. False = Outputs are switched Off True or Null = Outputs according to Floating Model
In	This input is used to move the valve or damper to position 0 - 100 %. This input slot is typically connected to the PID output. The range is 0 - 100 %.
Manual	This input is used to override the **In** input from the PID controller manually. **Manual** has higher priority than **In**. Under certain conditions (Frost Protection Switch, On-site overwrite), the actuator can apply for a certain position. The Range is 0 - 100 %, Null = No Overwrite
Interlock	This input can be used to perform an interlock operation, in which the valve or damper is only opened if a True is present; otherwise, the valve or damper is closed. **Example:** Only when a fan is running or an airflow contact detects flow can you open the output. Until **Interlock** is True, the output remains closed. Even if **Manual** is greater than 0 %, the output remains closed until I**nterlock** is True. If the **Interlock** is False, the actuator is closed, according to the **Runtime Close**.

Table 273 Input of Floating (Continued)

Input Name	Description
Self Test Cycle	The self-test determines whether the drive is properly wired. This input is used to set a time. The self-test first opens the actuator for the specified time, then closes it again for the same time. The procedure is then repeated indefinitely. This allows the wiring to be quickly tested during commissioning. If the **Interlock** is connected, it only works if the **Interlock** = true. During Self Test, there is no motor model calculation, and therefore OutVirtualPos = “Null“; the motor runtime (**Runtime Close**, **Runtime Open**) are ignored. The inputs **In** and **Manual** are ignored during the self-test. **Enable** is still evaluated. During Self Test, it is not possible to start a synchronization over the input **Operation**. After the self-test input is set to null, the floating function block starts the power-up behavior (for example, a Power Up Synch). **Example:** If **Self Test Cycle** = 300sec, then the actuator opens for 300sec. After 300sec, it closes for 300sec. It opens again for 300sec. **Default:** Null = No Self Test
Start Exercising	Start Exercising keeps the value from becoming stuck due to water deposits when the valve is closed for an extended period of time. • If the current valve position **Out Virtual Pos** is <= 50 %, the valve is opened to 100 % and returned to the then valid position. • If the current valve position is > 50 %, the valve is moved to 0 % and then back to the current position. This input is used to start exercising at a 0->1 edge over an external logic, such as weekly depending on date or time. See also **Weekly Exercising**. • If **Synch Repeated** is active, then exercising starts. • If all repeat counts are finished, that means after the whole synchronization is finished. **False or Null:** No Exercising **True:** Start Exercising

Table 273 Input of Floating (Continued)

Input Name	Description
Operation (In & Par)	Operation refers to the process of determining the operating direction of an actuator. Direct refers to the opening of the actuator to supply cooling or heating energy or air into the room. However, in order for energy to be supplied, the actuator must be closed before selecting "Reverse." The drive can be turned off by selecting "Off." **Example:** To avoid shortcuts from unconnected cables, if the drive has not yet been installed during commissioning. During "Off," all outputs are turned off, the motor model is not calculated, and Out Virtual Pos is null. The self-test cannot be launched. If the Operation value changes, values 4-7 initiate a one-time synchronization (with the time to fully open or close + RuntimeOpen/Close * SynchOverrun). A new In value does not halt the synchronization. This enables synchronization from a separate logic, for example, every noon. If the synchronization is complete, the operation direction remains valid. Example to start a Synch to Open: **Operation:** Null (use parameter **Operation Par**) **Operation:** 4 = Direct + Start Synch to Open **Operation:** Null (use parameter **Operation Par**) **Values:** **1:** OFF **2:** Direct **3:** Reverse **4:** Direct + Start a one-time Synch to Open **5:** Direct + Start a one-time Synch to Close **6:** Reverse + Start a one-time Synch to Open **7:** Reverse + Start a one-time Synch to Close Null (Default, See parameter **Operation Par**)
Runtime Close (In & Par)	Runtime Close is the time taken by the actuator to move from the fully open position to the fully close position. In the Close position, no water or air flows through the tube or air duct. The output **Out Virtual Pos** is then 0 %. With a non-linear valve, you can also limit the running time to the linear range. In this case, the synchronization to OPEN (see parameter **Synch Power Up**) must be omitted.
Runtime Open (In & Par)	Runtime Open is the time taken by the actuator to move from the fully close position to the fully open position. In the Open position, all the water or air flows through the tube or air duct. The output **Out Virtual Pos** is then 100 %. With a non-linear valve, you can also limit the running time to the linear range. In this case, the synchronization to OPEN (see parameter **Synch Power Up**) must be omitted.
Position Feedback	The Position Feedback input is reserved for future usage.

Output

Table 274 Output of Floating

Output Name	Description
Out Virtual Position	The Out Virtual Position shows the current position of the calculated position of the floating model. The model follows the **In** value depending on the runtimes. See also **Out cause** for a better understanding. If the position is unknown and during power-up synchronizing, the value is "null." It is not "null" during a normal Open/Closing Synch. **Out Virtual Position:** 0-100 %, Null
Out Movement Status	The Out Movement Status is for information about what the actuator is currently doing. • Closing means that there will be less water flow through the register or less airflow through the air duct. • Opening means that there will be more water flow or more airflow. The output can be used for diagnosis and monitoring if this output is used to control further relays / triacs. ***Note:*** *It is impossible to achieve exact runtime due to the DDC cycle.* **Out Movement Status.** **1:** Stopped, **2:** Opening, **3:** Closing
Out Self Test Running	The Out Self Test Running output is "true" if the self-test is running. Refer to the input **Self Test Cycle** for more information.
Out Cause	The Out Cause is used to understand what the function block is doing right now. The output is bit-coded, which means that multiple bits can be active at the same time. Values greater than 0 indicate an exceptional situation. If the PositionFeedback input is connected and the feedback does not reflect the correct position after fully closed or open synchronization, the bit is 1024 = PositionFeedbackIsInvalid is set. ***Note:*** *The PositionFeedback is not yet implemented* **1:** PowerUpSynch **2:** SynchToOpen **4:** SynchToClose **8:** SynchRepeat active **16:** Disabled **32:** Manual **64:** Interlock false **128:** Self Test active **256:** Exercising active **512:** OperationIsOff **1024:** PositionFeedbackInvalid (Not yet implemented).

Parameter

Table 275 Parameter of Floating

Parameter Name	Description
Operation Par (In & Par)	This parameter is also available as an input, please refer to the description of the input **Operation Par**. ***Note:*** *The operation modes 4-7 (Direct/Reverse + start a one-time synch) are not supported by that parameter.* **Default:** Direct.
Runtime Close Par (In & Par)	This parameter is also available as an input, please refer to the description of the input **Runtime Close**. **Default:** 90 sec.
Runtime Open Par (In & Par)	See **Input of Floating** table for Input **Runtime Open** details. **Default:** 90 sec.
Hyst	An actuator cannot be moved to a small position. • If the PID controller sends very small changes to the Floating FB, a threshold can be set here. • If the difference between the new position to be approached and the motor model is greater than this threshold value, a new positioning is performed. If not, the input is ignored. The percentage value, like the **In** value, ranges from 0 to 100 percent. To avoid back and forth movements, the firmware limits the value to 0.05 * motor run time (100 % * 50msec cycle time = 0.05). **Range:** 0...100 %, **Default:** 1 %
Synch Power Up	Synch Power Up is used to set whether the actuator should synchronize after power-up. During power-up synchronization, the **Motor Close Terminal** is switched ON for the **Runtime Clos**e multiplied by the **Synch Overrun** factor (that is 2). Then the actuator closes the valve / damper, and the motor model is set to **Out** **Virtual Pos** = 0 %. If **Synch Power Up** is disabled, then the **Motor Close Terminal** keeps OFF, and the motor model starts internally with **Out Virtual Pos** = 50 %; this means that the model is matched if In = 0 % or 100 % and a Synch to Open or Close is active (see **Synch Operation**). **False:** Disabled **True:** Enabled (Default)

Table 275 Parameter of Floating (Continued)

Parameter Name	Description
Synch Operation	Due to many back and forth movements of the actuator, there will be a deviation between the calculated position of the motor model and the actual position of the valve or damper over time. As a result, whenever the drive is moved to the 0 percent or 100 percent position (to the mechanical stop of the valve / damper), it is useful to compare the model with the real position. Synch Operation specifies whether and in which direction the adjustment (Synchronization) should be performed. **1:** None: No Synchronization to Close or Open direction. **2 (Default):** Close: If the **In** value is < 2 %, the actuator is not closed with the runtime calculated by the model but with a longer runtime (**Synch Overrun * Runtime Close**). **3:** Open: If the **In** value is > 98 %, the actuator is not opened with the runtime calculated by the model but with a longer runtime (**Synch Overrun * Runtime Open**). **4:** Open and Close: This enables the Open and Close Synchronization as described above. ***Note:*** *The Synchronization is terminated immediately when the input "In" reaches a normal range between 2 and 98 % value. Thus, the PID control can act at any time if a control deviation occurs due to synchronization or if other positions must be approached due to different reasons.*
Synch Repeat Time	General description of Synch Repeat: If a synchronization after Close or Open has been performed (see **Synch Operation**), valves may not be completely closed after a certain time. **Example:** If the cold medium is missing, a cooling valve may heat up and expand after closing, becoming leaky. Synch Repeat allows you to repeat the close and open synchronization several times after an adjustable time to tighten leaking valves. • Synch Repeat to Close is only executed if **In** is unchanged by less than 2 %. • Synch Repeat to Open is only executed if **In** is unchanged by more than 98 %. • No further Synch Repeats are performed once the In value is between 2..98 %. ***Note:*** *This repeated synchronization should not be confused with a periodic 24h synchronization (see* ***Auto Synch Interval*** *below).*

Table 275 Parameter of Floating (Continued)

Parameter Name	Description
	Synch Repeat Time is used to set the time after the close or open synchronization is repeated. The Synch Repeat Timer starts when the previous synchronization (0 % or 100 % or the previous Synch Repeat) is finished. **Synch Repeat Count** (see below) sets several synchronizations repeats. **Example:** Synch Repeat Time = 3600 sec, Synch Repeat Count = 2. When **In** = 0 % -> Synch to Close with **Runtime Close * Synch Overrun** is started because **Synch Operation** = Close or Open and Close. **In** keeps on 0 %. 3600 sec (1 hour later): Synch to Close with **Runtime Close * Synch Overrun** is started as the first repeat. **In** keeps still on 0 %. 10800 sec (2 hours later): Synch to Close with **Runtime Close * Synch Overrun** is started as the 2nd repeat. **0 sec:** No repeated synch **Default:** 3600 sec (1h)
Synch Repeat Count	Specifies the number of repeated counts (See **Synch Repeat Time** above). **0:** No repeated Synch. **Default:** 3
Weekly Exercising	Instead of starting the exercise with the Start Exercising input, you can set up an automatic weekly exercise here. If the actuator (which is primarily used for valves that are not used during the winter or summer) has been in the same position for one week, an exercise will be initiated. The week-timer is restarted with every actuator movement. See also **Start Exercising**. **False:** No weekly exercising (Default). **True:** Weekly exercising.

Table 275 Parameter of Floating (Continued)

Parameter Name	Description
Power Up Delay	If a building has a large number of air damper actuators, it may cause problems with the main air handling system after a power outage and restart if all dampers are closed at the same time. There are several options for avoiding this: • No Power Up Delay • Wait according to the selected Power Up Delay The power-on synchronization (see **Synch Power Up**) begins with a time delay specified in Power Up Delay Time. After clicking the SAVE button, a new field Power Up Delay Time appears. The actuator remains in the last position during the power-on delay. The Power On Delay Time is between 1 and 4095s. • Move to closed after power-up without waiting To ensure adequate airflow of a central air conditioning system after power-up, it makes sense for the building's VAV dampers to be synchronized to CLOSE and the other to OPEN. When the central fan is turned on, this prevents an overpressure shutdown of the central air conditioning system. With this setting, the synchronization is performed to the CLOSE position after Power Up, regardless of the settings of **Synch Power Up** and **Synch Operation**, and without a delay time, with a runtime of **Runtime Close * Synch Overrun**. • Move to open after power-up without waiting Like “Move To Closed After Power Up Without Waiting,” the synchronization is performed to the OPEN position with Runtime Open * Synch Overrun runtime. • (Default): Synchronize randomly to closed Or open without waiting. After power-up, randomness decides whether to synchronize to the Closed or Open position. There is no wait time. The randomness differs from controller to controller after power-up. ***Note:*** *If the power-up synchronization is not finished or until fully open and fully synch is not reached or until there was a Synch to Close or a Synch to Open, OutVirtualPos = Null.*

Table 275 Parameter of Floating (Continued)

Parameter Name	Description
Motor Close Terminal	Motor Close Terminal defines the terminal at which the motor is wired for closing. Closing means that Out Virtual Pos changes from 100 -> 0 %. See **Operation and Operation Par,** where you can select between “Direct” and “Reverse.” The Motor Close Terminal can be a Relay or Triac. There are also AC1 and AC2 for controllers with a built-in actuator. **ROx**: Relay (for example, IRM) **TOx**: Triac (for example, IRM) **ACx**: Built-in actuator (for example, IRM). When voltage is applied to output AC1, the actuator moves clockwise. When voltage is applied to output AC2, the actuator moves counterclockwise. ***Note:*** *Use AC1 and AC2 only with a controller equipped with an integrated actuator (for example, Compact VAV). Otherwise, the hardware compatibility check will return an error.*
Motor Open Terminal	Motor Open Terminal defines the terminal at which the motor is wired for opening. Opening means that Out Virtual Pos changes from 0 -> 100 %. See also **Operation and Operation Par**, where you can select between “Direct” and “Reverse.” The Motor Open Terminal can be a Relay or Triac. There are also AC1 and AC2 for controllers with a built-in actuator. **ROx**: Relay (for example, IRM) **TOx**: Triac (for example, IRM) **ACx**: Built-in actuator (for example, IRM). When voltage is applied to output AC1, the actuator moves clockwise. When voltage is applied to output AC2, the actuator moves counterclockwise. ***Note:*** *Use AC1 and AC2 only with a controller equipped with an integrated actuator (for example, Compact VAV). Otherwise, the hardware compatibility check will return an error.*
Synch Overrun	The Synch Overrun specifies by how much the Runtime Close and Runtime Open are extended during the synchronization. • After Power Up – see **Synch Power Up**, • To Close and Open Direction – see S**ynch Operation**, • As Synch Repeat to Close and Open Direction – see **Synch Repeat Time** and **Synch Repeat Count** **Example:** The value 2.0 means that the synchronization runtime is 2* Runtime Close and 2* Runtime Open. • If the Runtime Close = 150 sec, then the Synchronization after Power Up or any other Synchronization to the Close position (0 %) takes 2 * 150 sec = 300 sec. • If the Runtime Open = 90 sec, then any Synchronization to the Open position (100 %) takes 2 * 90 sec = 180 sec. **Default:** 1.25

Table 275 Parameter of Floating (Continued)

Parameter Name	Description
Auto Synch Interval	If a valve or damper runs continuously without a break, that is, 24 hours a day, the calculated position Out Virtual Pos becomes increasingly inaccurate due to the numerous drives. As a result, it makes sense to begin a synchronization several hours after the most recent synchronization (that is after 24 hours). There are several possibilities for this: • No time-based repeated synchronization • To Closed or Open depending on SynchOperation Synchronization takes place after an adjustable time interval, which is specified in **Auto Synch Interval**. After clicking the SAVE button, a new field **Auto Synch Interval** appears. The time begins after the most recent synchronization. The direction of synchronization is determined by **Synch Operation**. • If Synch Operation is Close, it synchronizes to Close. • If Synch Operation is Open, it synchronizes to Open. • If Synch Operation is Open and Close, it synchronizes to the closer position based on Out Virtual Pos. Other Synch Operation settings do not initiate periodic synchronization. The Auto Synch Interval is between 1-255 hours. • To Closed (Independent of Synch Operation) Like above with an adjustable **Auto Synch Interval**, the synchronization always works in the Close Position. • To Open (Independent of Synch Operation) Like above with an adjustable **Auto Synch Interval**, the synchronization always works in the Open Position. • To Closed (After 24 h +- 1 h randomly) Synchronizes after 24 hours - 60 to + 60 min randomly to the Closed position if the actuator has not been driven fully closed within this interval. • Default: To Open (After 24 h +- 1 h randomly) Synchronizes randomly to the Open position after 24 hours -60 to +60 min if the actuator has not been driven fully open within this interval. ***Note:*** *While the synchronization is running, Out Virtual Pos shows the calculated position. The random value is calculated new after each periodic synchronization.*

Table 275 Parameter of Floating (Continued)

Parameter Name	Description
Out Save	• **Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out Virtual Pos**: To enable or disable the **Out Virtual Pos** feature. • **Out Movement Status**: To enable or disable the **Out Movement Status** feature. • **Out Self Test Running**: To enable or disable the **Out Self Test Running** feature. • **Out Cause**: To enable or disable the **Out Cause** feature.
Direction Change Runtime	**Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. **Min Runtime Same Direction:** Short running times frequently result in no movement. If the motor is restarted in the same direction after stopping, the **Min Runtime Same Direction** ensures that it is controlled for at least this amount of time. The motor remains turned off for shorter calculated running times. This means that **Min Runtime Same Direction** is the minimum switch-on time for the actuator when it is stopped, and the next target position can be reached by rotating the motor in the same direction (that means without reversing the direction). **Range:** 0-4095 msec; **Default:** 250 msec **Min Runtime Reverse Direction:** Due to mechanical play, short running times frequently result in no movement (backlash). If the motor is restarted in the reverse direction after a stop, the **Min Runtime Reverse Direction** ensures that it is controlled for at least this amount of time. For shorter calculated running times, the motor remains turned off. This means that **Min Runtime Reverse Direction** is the minimum switch-on time for the actuator when the actuator is stopped, and the next target position can be reached by rotating the motor in the opposite direction. **Range:** 0-4095 msec; **Default:** 250 msec ***Note:*** *The minimum runtime is 50msec because of the DDC cycle of the Floating function block.*

Example:

When using compact actuators with zone valves, the following parameters are recommended:

Floating Parameter	Value
Runtime Open Par	Check the stroke of the valve and the speed of the actuator. For old Honeywell valves and actuator it is 6.5mm stroke, which means: • 150sec for 50Hz line frequency • 125sec for 60Hz line frequency
Hyst	1%
Synch Power Up	True
Synch Operation	Close
Synch Repeat Time	3600sec for Cooling valves 0sec for Heating valves
Synch Repeat Count	3 for Cooling valves 0 for Heating valves
Weekly Exercising	true
Power Up Delay	No Power Up Delay
Synch Overrun	1,25
Auto Synch interval	false
Direction Change Runtime – MinRuntimeSameDirection	100msec
Direction Change Runtime – MinRuntimeReverseDirection	250msec

Pwm

The Pwm function provides a fixed cycle time with a variable duty cycle time. When the pulse width equals the cycle time, the duty cycle is 100 %, and the output is turned on continuously.

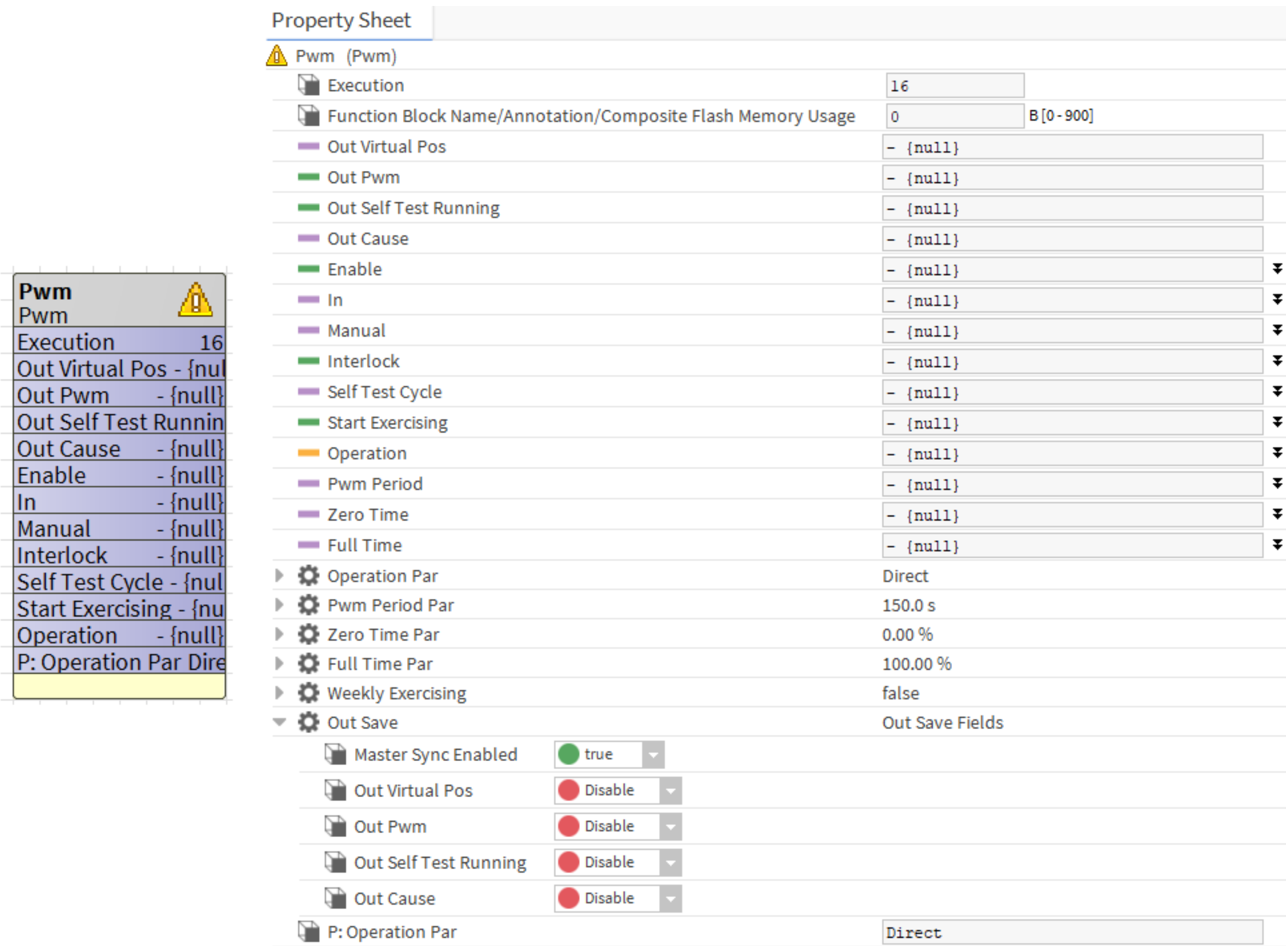


Fig. 397 Pwm Function Block and Property Sheet

If parameter Zero Time = 0 % and parameter Full Time = 100 % and input In = 0 %, then Pwm period interval is stop and restart with the 0 % (Off).

Note: *Restarting means that input In value > 0 % can switch On after the Pwm Period is expired. This enables fast switch-off (Fire, Drip Pan, Condensation) for safety reasons.*

Special case, if Zero Time = 0 % and Full Time = 100 %: As soon as In = 0 %, the Pwm period interval is stopped and restarted with the 0 % (Off). Restarting means that a value > 0 % can switch ON after an expired Pwm Period. This enables fast switch-off (Fire, Drip Pan, Condensation) for safety reasons.

Input

Table 276 Inputs of Pwm

Input Name	Description
Enable	This is a safety feature to switch a running Pwm output immediately to off. If **Enable** is set to true “1”, the Pwm cycle is started. For Reverse operation, the output is on in case of **Enable** set to false “0”.
In	The "**In**" input slot is generally connected to the PID output. For a cooling and heating control, a NumericSelect FB must be used between the PID and this output FB, which only transmits the cooling or heating PID output value to this output FB.
Manual	The **Manual** input is used to override the "**In**" input of the PID controller (automatic).
Interlock	Interlock is used if the output may open only if a fan is running or an airflow contact says flow. The output remains OFF until the interlock is "true." Even if Manual Override > 0 %, the output keeps closed until interlock is true "0". The output Out Pwm is switched off immediately. The Pwm cycle is terminated immediately. 1 or 0, Out Pwm is controlled as per the input value.
Self Test Cycle	It is a time after the output toggles between 0 % and 100 %. Whenever the time has elapsed, the output toggles. "0" or "null" disables the self-test.
Start Exercising	Start Exercising keeps the value from becoming stuck due to water deposits when the valve is closed for an extended period of time.

Output

Table 277 Output of Pwm

Output Name	Description
Out Virtual Position	Out Virtual Position shows the current position of the Pwm output as a result of all input conditions.
Out Pwm	The Out Pwm output must be connected to the physical output like a relay or a triac.
Out Self Test Running	The Out Self Test Running output is true if the self-test is running.
Out Cause	The output serves to understand what the function block is currently doing. **Disabled:** 16, **Manual:** 32, **Interlock false:** 64, **Self Test active:** 128, **Exercising active:** 256, **OperationIsOff:** 512.

Parameter

Table 278 Parameter of Pwm

Parameter Name	Description
Operation	Type the value to operate the actuator in direct or reverse action. **Off:** 1, **Direct:** 1, **Reverse:** 2
Pwm period	Type the pulse width period in seconds. The Pwm period is the time divided into between 0 and 100 %. If the Pwm period is 100 sec and In = 30 %, the output would be ON for 30 sec and then OFF for 70 sec. Then the cycle is repeated, ON for 30 sec and OFF for 70 sec.
Zero Time	Type the Zero Time value in percentage. When the 0 % command is given, output Out Pwm will be set to 1 for the duration specified in this parameter.
Full Time	Type the FullTime value in percentage. When a 100 % command is given, output Out Pwm will be set to 1 for the duration specified in this parameter.
Weekly Exercising	If the actuator is not moving for more than one week, exercise will be performed. **Disable:** 0, **Enable:** 1
Out Save	**Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. **Out Virtual Pos**: To enable or disable the **Out Virtual Pos** feature. **Out Pwm**: To enable or disable the **Out Pwm** feature. **Out Self Test Running**: To enable or disable the **Out Self Test Running** feature. **Out Cause**: To enable or disable the **Out Cause** feature.

Examples

The below diagram explains the Pwm output behavior.

Example 1: Pwm (Operation set to Direct)

Pwm output behavior

- **Operation:** Direct
- **Pwm period:** 100 Sec
- **Zero Time:** 0 %
- **Full Time:** 100 %

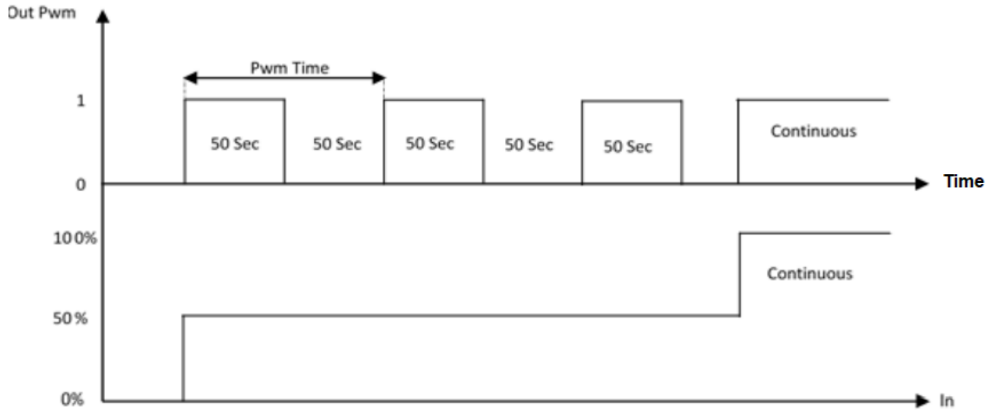


Fig. 398 Pwm output behavior

Example 2: **Pwm (Operation set to Direct)**

Pwm output behavior

- **Operation:** Direct
- **Zero Time Par:** 50 %
- **Full Time Par:** 100 %
- **In:** 50 %

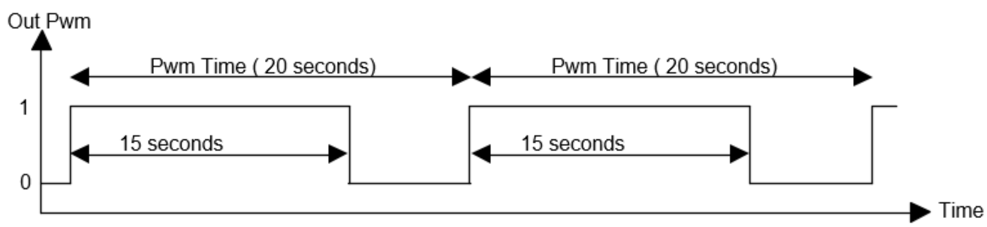


Fig. 399 Pwm output behavior

Example 3: **Pwm (Operation set to Direct)**

Pwm output behavior

- **Operation:** Direct
- **Zero Time Par:** 25 %
- **Full Time Par:** 100 %
- **In:** 20 %

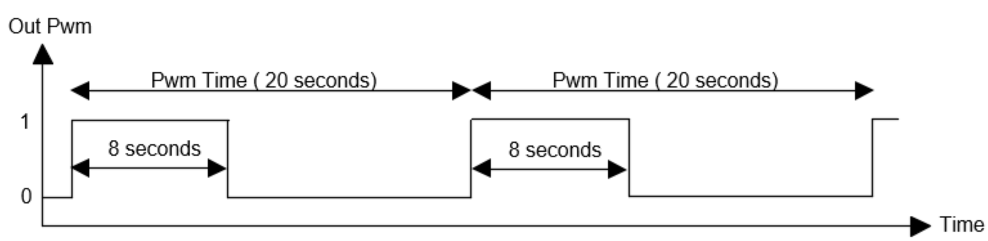


Fig. 400 Pwm output behavior

Stg123 Outp

The Stg123 Output function block is used to calculate the number of stages to be switched on based on 0 - 100 % input control signal. It supports up to a maximum of 3 stages. The function block also drives the individual binary outputs based on property configuration in a serial or parallel fashion.

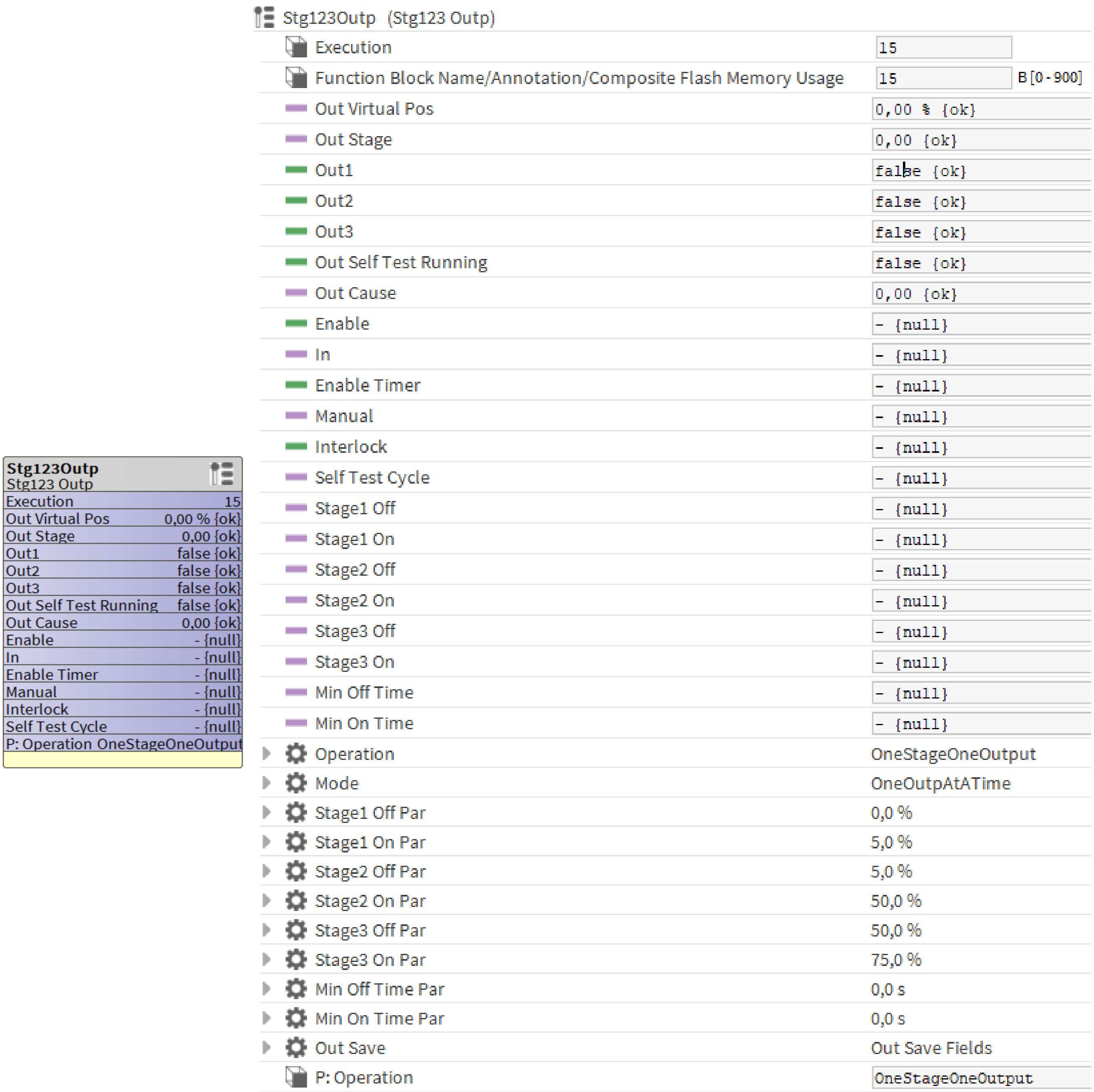


Fig. 401 Stg123Outp Function Block and Property Sheet

Stg123Outp: 1-Stage (1BO)

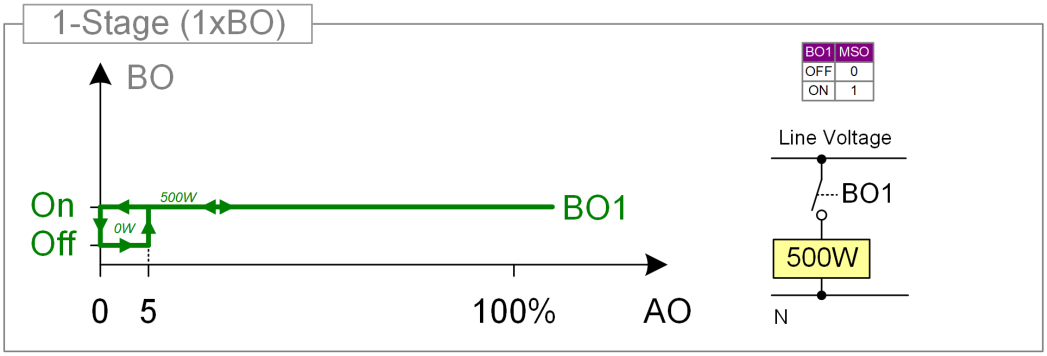


Fig. 402 Stg123Outp 1-Stage (1BO)

Stg123Outp: 2-Stage (2BO Parallel)

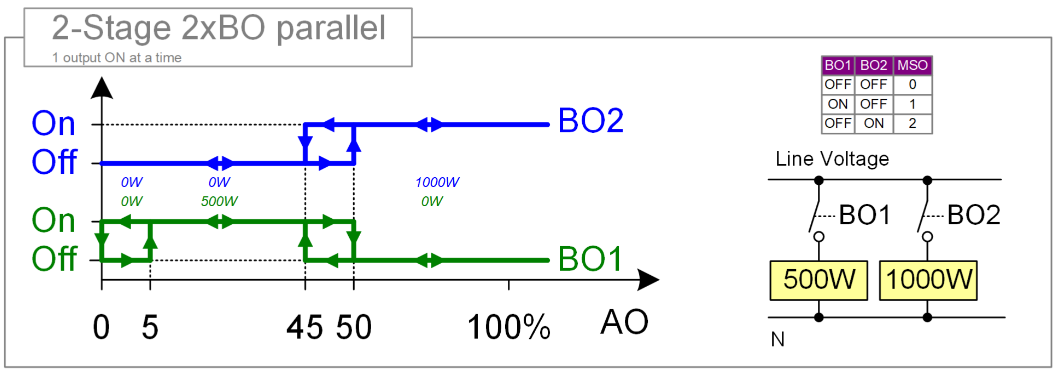


Fig. 403 Stg123Outp 2-Stage (2BO Parallel)

Stg123Outp: 2-Stage (2BO Serial)

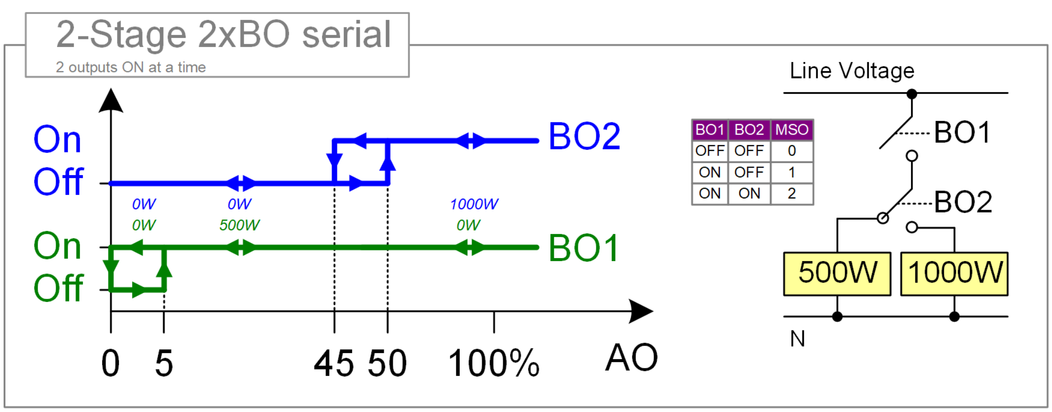


Fig. 404 Stg123Outp 2-Stage (2BO Serial)

Stg123Outp: 3-Stage (2BO Parallel)

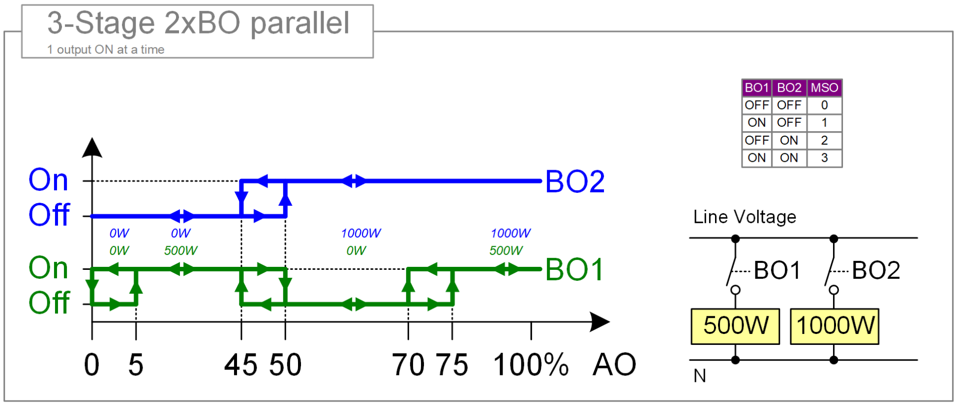


Fig. 405 Stg123Outp 3-Stage (2BO Parallel)

Stg123Outp: 3-Stage (3BO Parallel)

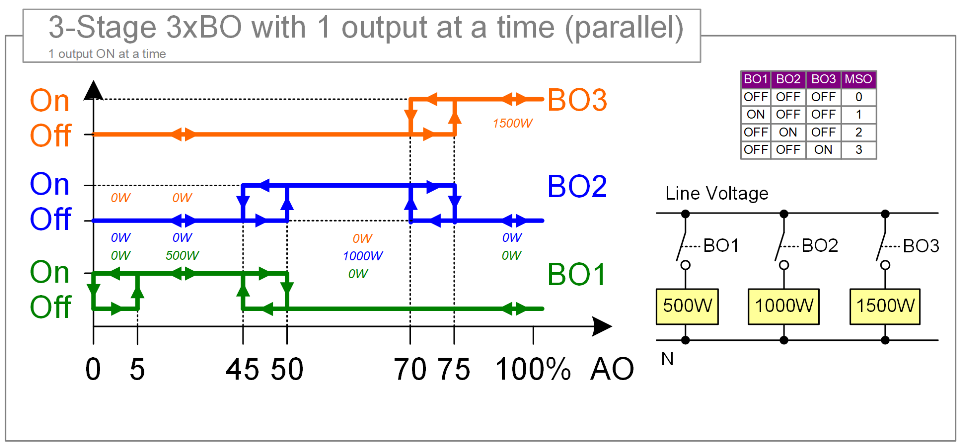


Fig. 406 Stg123Outp 3-Stage (3BO Parallel)

Stg123Outp: 3-Stage 3BO Serial

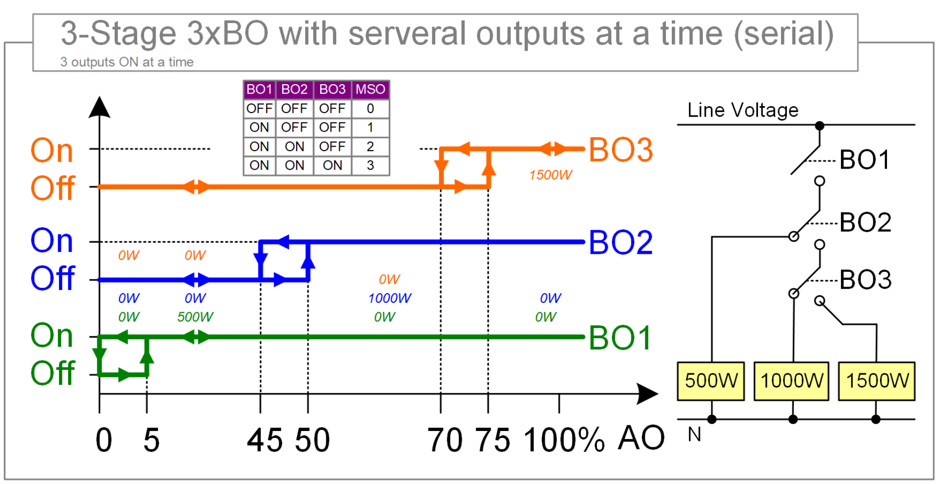


Fig. 407 Stg123Outp 3-Stage (3BO Serial)

Stg123Outp: MinOnOffTime

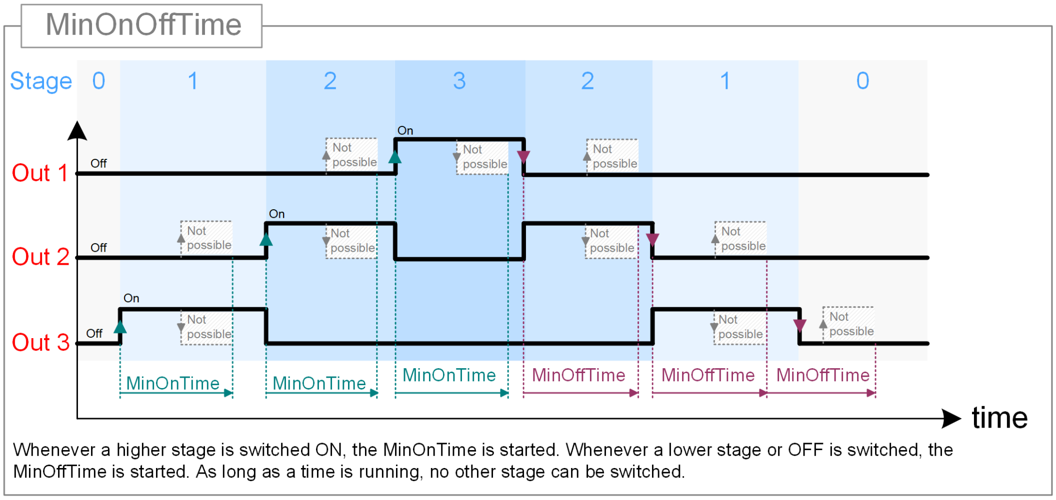


Fig. 408 Stg123Outp MinOnOffTime

Input

Table 279 Inputs of Stg123 Outp

Input Name	Description
Enable	Function block enable **Disable:** 0, **Enable:** 1
In	Input control signal 0 - 100 %.
Enable Timer	Enables Stage, On/Off times. If disabled, a running timer is stopped immediately. **Disable:** 0, **Enable:** 1
Manual	The Manual input is used for manual override of the "In" input of the PID controller (automatic). Under certain conditions (Frost Protection Switch), a certain position can be applied to the actuator. Manual has higher priority than the Input "In."
Interlock	Interlock is used where output opens only if a fan is running or an airflow contact detects flow. The output remains closed until Interlock is true. Even if Manual Override > 0 %, the output keeps closed until Interlock is “true.”
Self Test Cycle	Self Test Cycle is a time after the output toggles between 0 % and 100 %. Whenever the time has elapsed, the output toggles. 0 or null disables the self-test.
Min Off Time (In & Par)	It starts when any lower stage is switched or to OFF. No other stage can be switched if the Min Off Time is running.
Min On Time (In & Par)	It starts when any higher stage is switched. No other stage can be switched if the Min On Time is running.

Output

Table 280 Outputs of Stg123 Outp

Output Name	Description
Out Virtual Position	The value indicates the current stage as a percentage value. The possible values depend on the number of configured stages.
Out Stage	The value indicates the current stage.
Out1	This output must be connected to the physical output like a relay or a triac.
Out2	This output must be connected to the physical output like a relay or a triac.
Out3	This output must be connected to the physical output like a relay or a triac.
Out Self Test Running	The output is true if the self-test is running. Self test not running = 0, Self test running = 1
Out Cause	The output serves to understand what the function block is currently doing. **Disabled:** 16, **Manual:** 32, **Interlock false:** 64, **Self Test active:** 128, **Min Off Timer running:** 512, **Min On Timer running:** 1024, **OperationIsOff:** 2048

Parameter

Table 281 Parameters of Stg123 Outp

Parameter Name	Description
Operation	There are four operations as follows: • OneStageOneOutput • TwoStagesTwoOutputs • TwoStagesThreeOutputs • ThreeStagesThreeOutputs This is used to define the number of stages and the number of outputs used. For a 3-stage fan 3 outputs are necessary, for a 3-stage reheater 2 outputs are also sufficient (for example, output 1 with 500 W, output 2 with 1000 W, together this results in stage with 1500 W). The output can be deactivated via Off. For example, if the output has not yet been installed during commissioning to avoid shortcut from the unconnected cables. During Off, the output switch immediately OFF, OutVirtualPos = 0, OutStage = 0, Self-test cannot be started, the Staging Model is no longer executed.

Table 281 Parameters of Stg123 Outp (Continued)

Parameter Name	Description
Mode	There are two modes: • OneOutputAtATime • SeveralOutputAtATime This can be used to set whether only one individual output is to be switched on at a time or whether the outputs are to be switched on in parallel, that is several at the same time. The answer depends on the wiring.
Stage1 Off	Type the **Stage1 Off** value in percentage. An input value below this value switches to OFF.
Stage1 On	Type the **Stage1 On** value in percentage. An input value above this value switches to Stage1.
Stage2 Off	Type the **Stage2 Off** value in percentage. An input value below this value switches to Stage1.
Stage2 On	Type the **Stage2 On** value in percentage. An input value above this value switches to Stage2.
Stage3 Off	Type the **Stage3 Off** value in percentage. An input value below this value switches to Stage2.
Stage3 On	Type the **Stage3 On** value in percentage. An input value above this value switches to Stage3.
Min Off Time Par (In & Par)	Type the **Min Off Time Par** in seconds. It starts when any lower stage is switched or to OFF. If Min Off Time is running, no other stage can be switched.
Min On Time Par (In & Par)	Type the **Min On-Time Par** in seconds. It starts when the higher stage is switched. If Min On Time is running, no other stage can be switched.
Out Save	• **Master Sync Enabled**: If you set it to “true,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out Virtual Pos**: To enable or disable the **Out Virtual Pos** feature. • **Out Stage**: To enable or disable the **Out Stage** feature. • **Out 1**: To enable or disable the **Out 1** feature. • **Out 2**: To enable or disable the **Out 2** feature. • **Out 3**: To enable or disable the **Out 3** feature. • **Out Self Test Running**: To enable or disable the **Out Self Test Running** feature. • **Out Cause**: To enable or disable the **Out Cause** feature.

Example 1: Stg123

The below diagram explains the stage output behavior.

Stage 1 On: 10 %

Stage 1 Off: 0 %

Stage 2 On: 50 %

Stage 2 Off: 10 %

Stage 3 On: 75 %

Stage 3 Off: 50 %

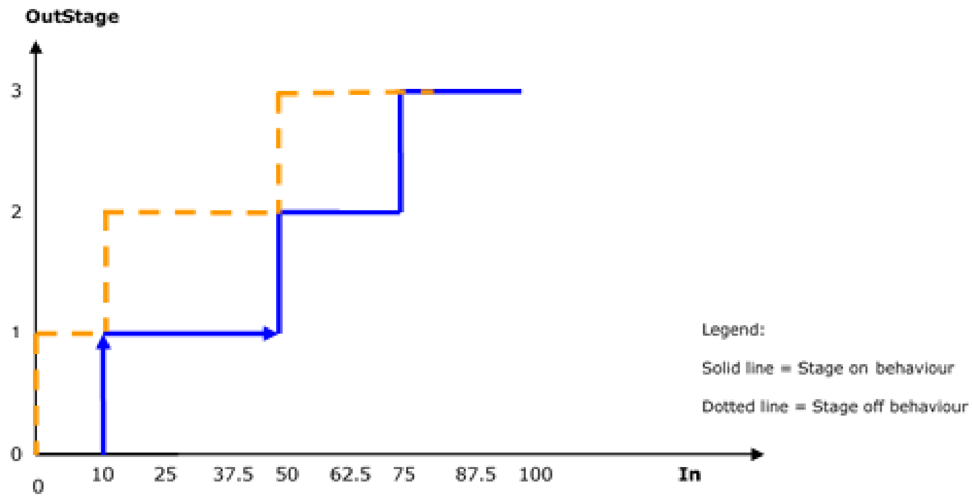


Fig. 409 Stage output behavior

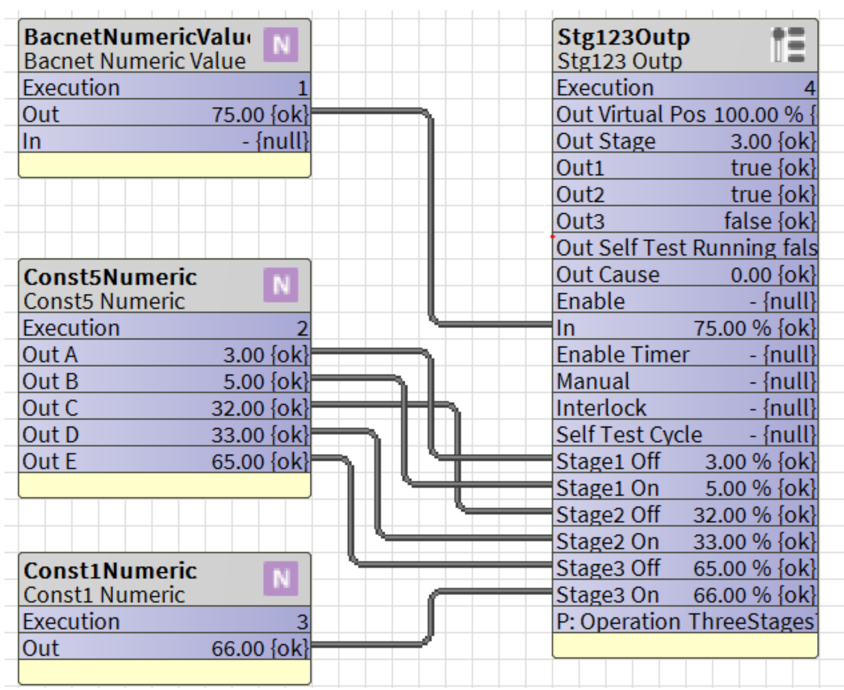


Fig. 410 Stg123 Out function block

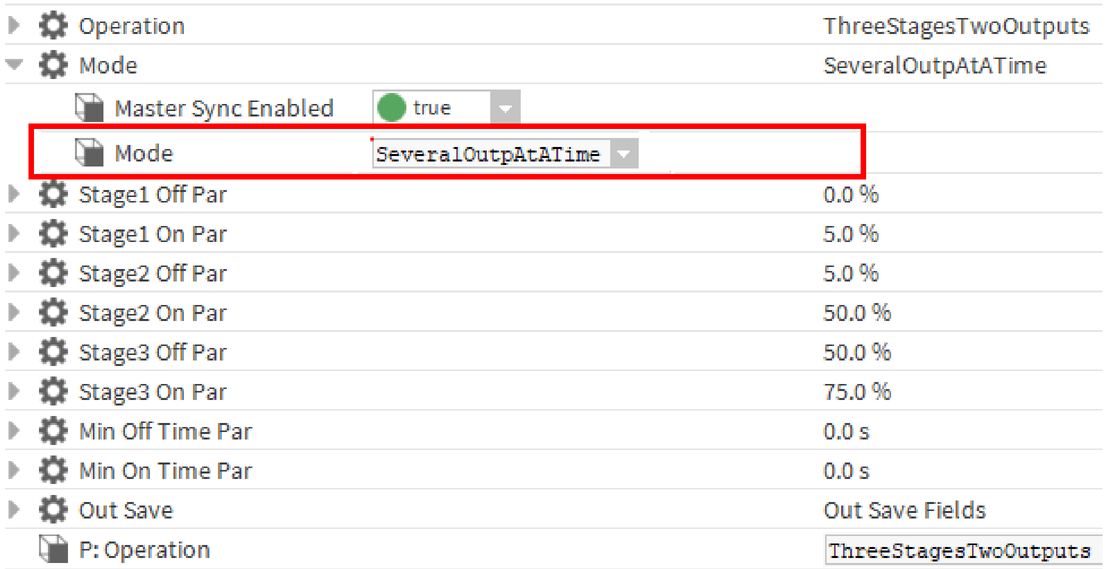


Fig. 411 Stg123 Out property sheet

CHAPTER

16 LIGHT FUNCTION BLOCK

The following Light function blocks are available in the honIrmControl Palette and can be configured and used to create the required application logic:

- Light A

Fig. 412 Light Function Block

Refer to General Description on page 23 for general information about Function Blocks.

Light A

Light A function block is used to control light for On/Off, dimming, or a stairwell application.

To reliably detect a short push-button action, directly wired push buttons are read in directly within the function block (do not enter the function block via an input slot). The function block also includes wired outputs (not via output slots).

The function block should be able to recognize a short keystroke with >= 50 msec.

If the light is dimmed with the push button, the light output level will respond quickly. A maximum of 8 LightA function blocks can be used in a single application.

Note: *To achieve the best performance for the operation of light, the light function block must be put into the Event program folder. As soon as a wired button is pressed or a Modbus message is received (the Modbus function block must also be in the Event Program Folder for this to happen), the Event Program Folder is executed immediately.*

Light A has input slots with different priorities:

- **Emergency**
- **Motion Brightness Sensor**
- **Manual**
- **Application**

- **Emergency:** The emergency input has the highest priority. If the emergency command is active, motion brightness sensor command, manual override command and application command inputs are deactivated. It is used during fire alarm, where all buttons and motion sensors are deactivated. After termination of the Emergency condition the Emergency Input must be set to "101= DoNothing".
- **Motion Brightness Sensor:** The Motion Brightness sensor enables the light to be turned on and the logic to prevent it from being turned off if motion is detected during the night. It can also be used only for to turn off. The FunctionBits can be configured to perform specific operations. It is intended to keep people from standing in the dark. If the auto off timer is enabled via function bits, the light will turn off when the auto off timer expires and no motion is detected..
- **Manual:** The manual input is used to enable manual operation. For example, via external BACnet pushbuttons or via Niagara graphics. The manual input and the directly wired push buttons work as per the "Last Wins" principle. Commands via the manual input are executed once when the input is changed. The manual command has higher priority than the application input. As soon as the manual input receives a command, a LockTime can be started. This ensures that the manual command (e.g. from the button) is valid for a certain time and is not immediately overwritten by the application. Normally, the Manual Ovrd Cmd input is always set to "Do Nothing". Only when the manual command is triggered by pressing an external button or by an external wall module, the command is converted into a valid command (i.e. 50 %) for a DDC cycle. Afterwards the input at the function block must be set to “Do Nothing” again.
- **Application:** The application input serves as input of the self-programmed "automatic" logic. The input has the lowest priority. Depending on brightness, room occupancy, different Logic can be programmed which can be overridden by a manual operation (directly wired or over manual inputs). While the optional LockTime is running, various automatic commands can be set on the application input, e.g. Light Off because it is bright outside. All these commands are not executed if the LockTime is running. As soon

as the LockTime has expired, the last valid Application command is executed. Here, too, the principle applies that normally a "Do Nothing" is present at the input and only when a change occurs, e.g. a change from dark to bright, a valid light command is set for one DDC cycle. Afterwards the application input at the function block must be set to “DoNothing” again.

An important feature in this cause. The programmer can optionally add a cause to the function block. Depending on which input is currently being used (whether via one of the 2 input slots or directly wired), the correct cause is given to the output. This makes it easy to see why the function block behaves like this.

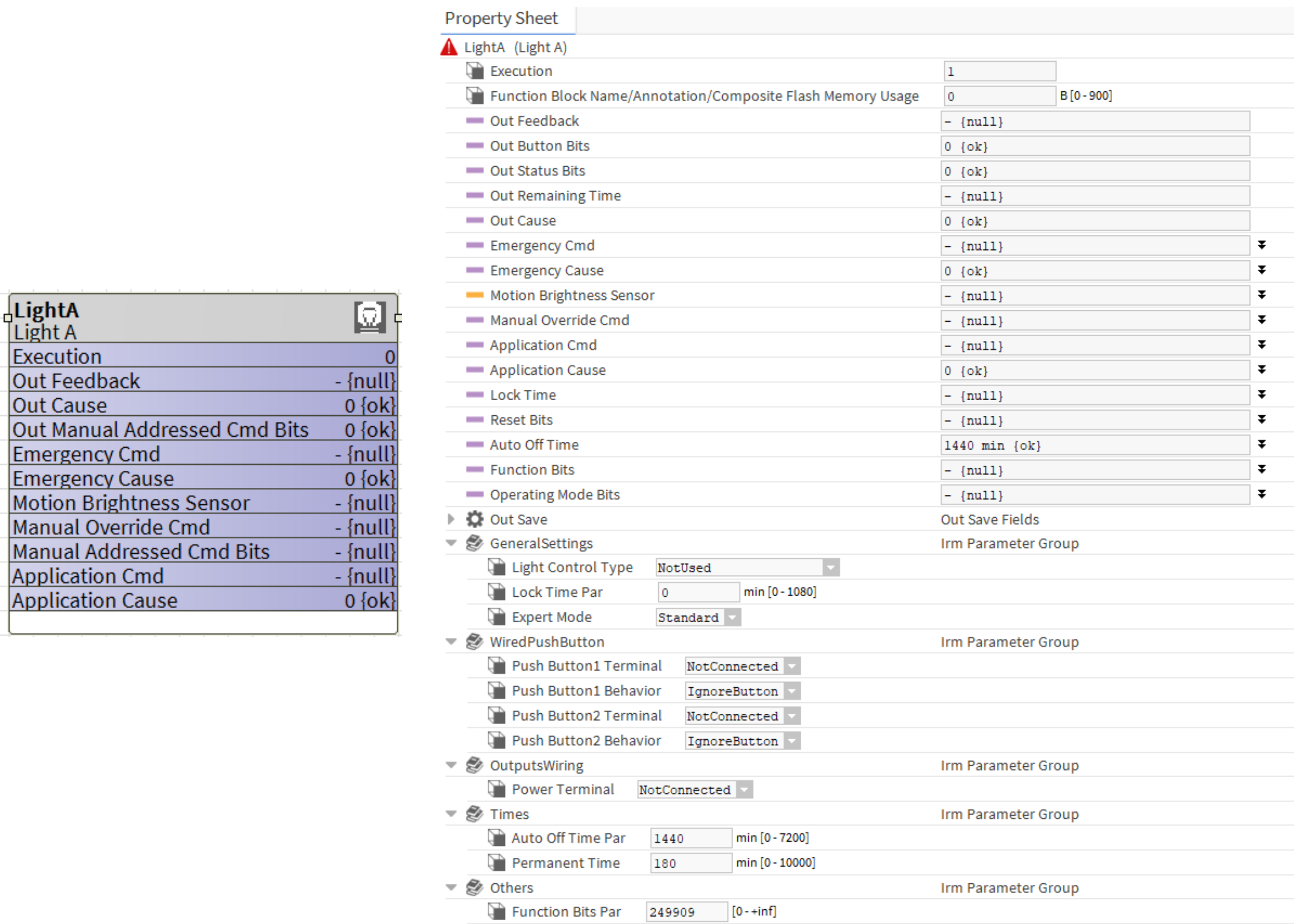


Fig. 413 LightA Function Block and Property Sheet

Input

Table 282 Inputs of Light A

<table>
<tr><th>Input Name</th><th>Description</th></tr>
<tr><td>Emergency Cmd</td><td>The emergency input has the highest priority. The manual operation (wired or via manual) as well as the application cmd are ignored.
Active value = 0 - 100% (or 200-300, *1)
Do Nothing = 101 (or 301, *1)
Last Valid Application Command + Do Nothing= 102 (or 302, *1). The Emergency Mode is terminated, and the last valid application Command is set. After switching to the last valid application command, the command works like a “Do Nothing” command.
Null: Same meaning as “Do Nothing”.</td></tr>
<tr><td>Emergency Cause</td><td>For debug reasons, it is good to have a cause to understand why the light is in this state. This input is looped through to the Out Cause when the EmergencyCmd is executed.
Since many causes can also be used for the BlindA function block, it is recommended to use the same values for both function blocks. User defined causes start with the value >= 100.
Examples:
151=Wind Low
152=Wind High
152=Service
153=No Service
153=Fire
155=No Fire
154=Rain
155=NoRain
156=Ice (Outside Air temp)
157=No Ice (Outside Air temp)</td></tr>
<tr><td>Motion Brightness Sensor</td><td>This is used to connect a motion detector including a brightness sensor, as it is typically offered in DIY stores. Depending on the FunctionBits, the light can, for example, only be switched On or only switched Off or both On and Off. 1 means that the light can be Off because there is no occupancy or because it is bright. 2 - 4 means that there is Occupancy detected during darkness and depending on the Function Bits, the light behaves accordingly for switching ON or it ignores some Off commands.
Note: If an ON-command (2-5) occurs while the light is already ON, then there is no need to switch the light ON; the light keeps on the current light level.
No Need for Light = 1 or Null
On With Maximum Light = 2
On With Minimum Light = 3
On with Memory Light = 4
On with SwitchOnOutput = 5</td></tr>
</table>

Table 282 Inputs of Light A (Continued)

<table>
<tr><th>Input Name</th><th>Description</th></tr>
<tr><td>Manual Override Cmd</td><td>This input serves as manual operation via an external wall module or external push-buttons (BACnet, Modbus etc) or via Niagara graphic. The input is only evaluated in the DDC cycle, where this input receives a new value e.g. from “Do Nothing” -> “80%” or from “ManOn” -> “ManOff”. To repeat a command, a " DoNothing (101)" must be set in between.

Please also look for the "3 upper bits" description in the command list. (See *1).

• Active value = 0 - 100% (or 200-300, *1)
• DoNothing = 101 (or 301, *1)
• Last Valid Application Command = 102 (or 302, *1)
• ManOnOff = 103 (i.e. after short press) (or 303, *1)
• ManOnOffDimming = 104 (i.e. after long press) (or 304, *1) To stop the dimming, the command 110 is required.
• ManOn = 105 (i.e. after short press) (or 305, see *1)
• ManOff = 106 (i.e. after short press) (or 306, see *1)
• ManOnDimmingUp = 107 (i.e. after long press) (or 307, see *1)
• ManOffDimmingDn = 108 (i.e. after long press) (or 308, see *1)
• ManStaircase = 109 (or 309, see *1)

The command does exactly what the wired button with a short press would do. Between the command and the wired button there is a Last Wins mechanism. It’s not possible to enable the Extension or Permanent time feature with this command. The command does also not dim the light. If the (Modbus) Button is released, then DoNothing is expected. If after 109 a 309 command is received, then the AutoOffTime should be restarted.

• ManRelase = 110 (or 310, see *1) (after releasing a button to stop dimming from command 104, 107 or 108.

Note: If the Command 104 (Start Dimming) is given, but Command 110 (ManRelease) is not given, then the light dims max. 10x and then it sets to light to the Switch On value and the Cause to 28.</td></tr>
<tr><td>Manual Addressed Cmd Bits</td><td>This input is used if a pushbutton connected to another light function block should operate with this function block. The value is bitcoded and contains the light address and the light value. This input has the same priority as Manual Override Cmd and the hardwired Push Buttons.

The value “null “deactivates the Addressing Feature</td></tr>
</table>

Table 282 Inputs of Light A (Continued)

Input Name	Description
Application Cmd	The application input has the lowest priority. The manual operation (wired or via manual) is still working. The input is used to set the automatic control to fullfill i.e. EN15232 programmed in the wiresheet. The last command before DoNothing is stored in RAM so that it can be executed when the Locktime has expired or if the command is Last Valid Application Cmd i.e. from EmergencyCmd or Manual Override Cmd. ***Note:*** *If the last command is a 0% command to turn off the light, the command will be ignored if the MotionBrightness sensor is configured accordingly and detecting Occupied+Dark.* • Automatic value: 0-100% (or 200-300, see *1) • DoNothing = 101 (or 301, *1) • Last Valid Application Command = 102 (or 302, *1)
Application Cause	For debug reasons, it is good to have a cause to understand why the light is in this state. This input is looped through to the Out Cause when the Application Cmd is executed. Since many causes can also be used for the BlindA function block, it is recommended to use the same values for both function blocks. User defined causes start with the value >= 100. Examples: 100=Bright 101=Dark 102=OccOccupied 103=OccOff 104=OccHoliday 105=OccUnoccupied 106=Standby 107=Bypass 108=OccSensorOcc 109=OccSensorUnocc 110=Time 111=TimeWithPresenceSimulation 112=Sunrise 113=Sunset 114=Morning dawn 115=Evening dawn

Table 282 Inputs of Light A (Continued)

Input Name	Description
	116=GlareProtection (Not fully negative angle) 117=SightProtection (Fully negative angle) 118=InsulationSummer (Room Temp) 119=InsulationWinter (Room Temp) 120=DaylightRegulation (Date, Time, Location) (Max Ligth w/o glare) 121=WindowOpen 122=WindowClosed 123=DoorOpen (DisableApplication, Disable closing) 124=DoorClosed 125=DifferentialTempHigh (RoomTemp <-> Outside Air Temp) 126=DifferentialTempLow (RoomTemp <-> Outside Air Temp) 127=AstroMorning (Date, Time, Location) 128=AstroEvening (Date, Time, Location) 129=ShockSensorActive 130=ShockSensorDeactive 131=RoomTempHigh (Allow Open; Start Auto) 132=RoomTempLow 133=RoomHumidityHigh (Open Window) 134=RoomHumidityLow (Close Window) 135=AutomaticOn 136=AutomaticOff
Lock Time (In & Par)	After a manual command (wired or via Manual Override), a Lock Time is activated to ignore the application cmd for this time. This prevents the application from immediately overriding / canceling the manual command. 0 min and null mean No Lock. Range: 0-1080min, default=Null
Reset Bits	This can be used to reset various states. 1=Reset Auto off Timer (including permanent and extension time). If a SwitchOffWarning is configured, then the switch off warning is started. 2=ResetSwitchOffDelay 4=ResetLockTime 8=ResetLastValidApplicationCmd

Table 282 Inputs of Light A (Continued)

Input Name	Description
Min Output (In & Par)	Dimming is performed between **Min Output** and **Max Output** if the corresponding FunctionBits are set. 0-100%, default=Null
Max Output (In & Par)	Dimming is performed between **Min Output** and **Max Output** if the corresponding Function Bits are set. 0-100%, default=Null
Switch On Output (In & Par)	This defines the Switch-on Output for a dimmable lamp. The lamp is switched on with this value. If the Operation Mode Bit "Memory" is activated, the SwitchOnValue is used until a "Memory" value has been learned after manual dimming. 0-100%, default=Null
Auto Off Time	The AutoOffTime is used to automatically switch off a lamp after a time that can be set here. This function can be used in a staircase with a short time, but also for an office with a long time (e.g. 12 or 24 hours). Switching Off is prevented via the Motion Brightness Sensor input. In addition, an OFF warning can be activated so that the room user is warned before switching off. In this case, the switch-off warning is issued after the Auto Off Time has expired. The Operation Mode Bit and Auto Off Time needs to be set. In the staircase, the time can be extended up to 4 times by pressing the button. 0-7200.0 min, Default=Null
Switch Off Delay Time	The Switch Off Time can be used to delay the automatic switch-off by a few seconds. If, for example, several lamps in an office are switched off by the same Application Logic, you can achieve that they are switched off in a time sequence and not simultaneously. This prevents all lamps from performing the OffWarning at the same time or that someone is in the dark. 0-86400sec, Default=Null

Table 282 Inputs of Light A (Continued)

Input Name	Description
Function Bits (In & Par)	This configures the behaviour of the lamp under various conditions. • On_After_PowerUp = 1 • Off_After_PowerUp = 2 • Off_If_AutoOffTime_is_expired = 4 • On_If_Occ_and_Dark = 8 • On_and_Restart_AutoOffTimer_If_Occ_and_Dark = 16 • Off_If_Unocc_or_Bright = 32 • Enable_Off_If_AutoOffTimer_is_expired_during_Occ_and_Dark = 64 • Enable_Off_From_ApplicationCmd_If_Occ_and_Dark = 128 • Enable_On_With_Memory_Value = 256 • Enable_SoftOn_DimUp = 512 • Enable_SoftOn_DimDown = 1024 • Enable_SoftOff = 2048 • Enable_SwitchOffWarning = 4096 • Enable_SwitchOff30secDelay = 8192 • Enable_LimitToMin = 16384 • Enable_LimitToMax = 32768 • Enable_ExtensionTime = 65536 • Enable_PermanentTime = 131072 • Null • Default: Null
Operating Mode Bits	Sometimes the given priority of the emergency, manual and application command inputs does not fit. Certain operating modes or enabling and disabling of some extra features are necessary to implement the desired function. • Always100PctOn = 1 • AlwaysOff = 2 • DisableWiredPushButton1OffCmd = 4 • DisableWiredPushButton1OnCmd = 8 • DisableWiredPushButton2OffCmd = 16 • DisableWiredPushButton2OnCmd = 32 • EnableWiredPushButton1OffCmd = 64 • EnableWiredPushButton1OnCmd = 128 • EnableWiredPushButton2OffCmd = 256 • EnableWiredPushButton2OnCmd = 512 • EnableManualCmdDuringEmergency = 1024 • EnableMotionBrightnessSensorDuringEmergency = 2048 • Null

**1 These values can also be used alternatively. If you want to repeat the same command, you can alternately use one and the other value.*

Function Bits for Automatic switching On or Off within the Function block

- Off_After_PowerUp means that the light keeps Off after power up of the controller.
- On_After_PowerUp switches the light On to the SwitchOnOutput even if Enable_On_With_Memory_Value is enabled, because after Power up there is no Memory value available.
- With Off_If_AutoOffTime_is_expired a AutoOffTimer is started when the light is switched On manually or from ManualOverrideCmd or from ApplicationCmd. The Functionblock switches the light Off if the AutoOffTimer is expired. The AutoOffTimer is restarted if the Off_If_AutoOffTime_is_expired bit changes from 0 -> 1. This can be used, for example, to ensure that no timer is running when the room is occupied, but that the timer is running with the start value when the room is not occupied. If the long press switches the light from Off to On, then the AutoOffTimer starts. If the light is already ON and the long press is used for dimming, then the AutoOffTimer is still running without any correction. If the light is OFF, the AutoOffTimer is stopped. If the AutoOffTimer is expired, the Light is switched Off (With a warning if warning is enabled). If the AutoOffTimer is expired, but the MotionBrightnessSensor detects Occ, then the light is switched Off only in case Enable_Off_If_AutoOffTimer_is_expired_during_Occ_and_Dark is enabled.
- On_If_Occ_and_Dark means that the lamp is switched on if the input MotionBrightnessSensor changes to ON.
- On_and_Restart_AutoOffTimer_If_Occ_and_Dark means that the lamp is switched on if the input MotionBrightnessSensor changes to ON and then the AutoOffTimer is restarted (if Off_If_AutoOffTime_is_expired is enabled).
- Off_If_Unocc_or_Bright means that the lamp is switched off when MotionBrightnessSensor changes to Unocc. If there is an AutoOffTimer running and Enable_Off_If_AutoOffTimer_is_expired_during_Occ_and_Dark is enabled, then the light would switch Off, even if the AutoOffTimer is still running.

Function Bits to enable additional Features

- Enable_Off_If_AutoOffTimer_is_expired_during_Occ_and_Dark means that the light is switched Off if the input MotionBrightnessSensor changes to Unocc and the AutoOffTimer is expired at that time. If the timer is not expired, then the change to Unocc is ignored.
- Enable_Off_From_ApplicationCmd_If_Occ_and_Dark means, that 0 % from ApplicationCmd switches the Light Off in case that MotionBrightnessSensor is Occ.
- Enable_On_With_Memory_Value means that if the light is switched On manually, it is switched ON with the latest manually dimmed percentage level (from button long press or from the ManualOvrdCmd input slot with the commands 0.01 - 100 %, ManOnOffDimming, ManOnDimmingUp or ManOffDimmingDn). The memory value is the last value reached during dimming. If the lamp was switched off by dimming, i.e. dimming was stopped at 0 %, the old memory value remains. The memory value is always > 0 %. After the first PowerUp the memory value is equal to the parameter "SwitchOnOutput“. This mode is not recommended for fluorescent lamps as they should always be switched on with full power.
- With Enable_SoftOn_DimUp the brightness is dimmed up when switched on; this should be deactivated for fluorescent lamps.
- With Enable_SoftOn_DimDown the brightness is slowly dimmed down when switched on (usable for fluorescent lamps).
- With Enable_SoftOff, the light is dimmed down for switching Off.

- With Enable_SwitchOffWarning, the On/Off light switches Off 3 times for 1 sec (at 30/18/8 sec) before switching Off; for LightDimming: The light is dimmed down to the Min value within 30 sec before switching Off; this is used to prevent someone from being in the dark without a warning.
- Enable_SwitchOff30 sec Delay delays the Off command by 30 sec. If 2 lamps are installed in the office, a time offset between 2 lamps can be achieved so that both lamps are not switched off at the same time.
- With Enable_LimitToMin, the light output is always >= MinOutput.
- With Enable_LimitToMax, the light output is always <= MaxOutput.

For Staircase only

- With Enable_ExtensionTime the time in the staircase can be extended up to 4 times.
- With Enable_PermanentTime the time in the staircase can be set to a long timer value.

Output

Table 283 Outputs of Light A

Output Name	Description
Out Feedback	The output indicates the current controlled light value, which is output at the terminals. • 0 % = Light is Off • 0 % -100% = Light is ON • Null = If the LightControlType = NotUsed or after power up
Out Button Bits	Often it is important to know whether the button has been pressed, e.g. to restart timers in the application logic, • 0 means that no button is pressed. • 1 means button 1 is pressed • 2 means button 1 was pressed shortly • 4 means button 1 was pressed for a long time • 8 means button 2 is pressed • 16 means button 2 was pressed shortly • 32 means button 2 was pressed for a long time The detection of a button press happens after releasing the button or after 2sec. The output is set only for 1 DDC cycle. Example: After releasing a long button press, Out Button Bits = 5.
Out Status Bits	Various states can be read from the function block via this output. See also Out Remaining Time. 0 means no timer is running • 1 means the AutoOffTime is running • 2 means the SwitchOffDelay is running • 4 means the SwitchOffWarningTimer is running • 8 means the LockTime is running

Table 283 Outputs of Light A (Continued)

Output Name	Description
Out Remaining Time	There are different timers, but only 1 timer is running at a certain time: • Auto Off Timer • Warning Timer • Off Delay Timer Whenever one of these timers is running, the remaining time in seconds is shown here. The Lock Time is not shown. See also Out Status Bits.
Out Cause	The Out Cause provides different states for diagnosis to see at a glance what is currently happening in the FB. Since the Emergency and Application Logic is programmed by the user, the Cause inputs of Emergency and Application are looped through to Out Cause when the corresponding Cmd has been executed (and the Light is controlled). If the Cause inputs have the value null and an Emergency Cmd or Application Cmd was executed, then Out cause is set to the values Emergency Cmd or Application Cmd. • 001 = NoCause (initial value) • 002 = Emergency (EmergencyCmd ! = Null and In Cause = Null) • 003 = WiredManualOvrdButton1 • 004 = WiredManualOvrdButton2 • 005 = WiredManualOvrdButton1WithLockTime • 006 = WiredManualOvrdButton2WithLockTime • 009 = ManualOvrdCmd (Cmd 0-102) • 014 = ManualOvrdCmdWithLockTime (Cmd 0-102) • 015 = ApplicationCmd (ApplicationCmd ! = Null and In Cause = Null) • 017= OperatingModeAlwaysOn • 018 = OperatingModeAlwaysOff • 019 = MotionBrightnessSensor_OnCondition • 020 = MotionBrightnessSensor_OffCondition • 021 = ManualAddrCmd • 022 = ManualAddrCmdWithLockTime • 023 = PowerUpOn • 024 = PowerUpOff.

Table 283 Outputs of Light A (Continued)

Output Name	Description
	• 025 = ManualOvrdButtonCmd (Cmd 103-110) • 026 = ManualOvrdButtonCmdWithLockTime (Cmd 103-110) • 027 = AutoOffTimer expired • 028 = DimmingUpDnTimerExpired Reserved until 99 for firmware enhancements
Out Manual Addressed Cmd Bits	The connected push button can be used for other lamps represented by other function blocks. The value is bit-coded and serves as input signal for other light function blocks. The value contains the light address and the light value. The meaning of the individual bits is explained separately.

Parameter

Table 284 Parameter of Light A

Parameter Name	Description
Light Control Type	Set the light control type. • Not Used • Light On Off • Light Dimming • Light On Off Timer • Light Dimming Timer • Dimming On Off Staircase • Light Dimming Staircase ***Note:*** *If the light contol type is set to Not used, then the function block will be disabled.*
Lock Time Par (In & Par)	Type the manual command Lock Time in minutes. After a manual command (wired or via Manual Override), a lock time is activated to ignore the application cmd for this time. This prevents the application from immediately overriding / canceling the manual command. 0 min and "null"means No Lock. Default=0min
Expert Mode	Set the mode Standard / Expert The expert mode will enable more light features.

Table 284 Parameter of Light A (Continued)

Parameter Name	Description
Push Button 1 Terminal	Basically, 2 inputs are supported to switch or dim the light. According to the configured button behavior, only 1 input or 2 inputs are necessary. The inputs are typically connected with pushbuttons, but it is also possible to connect the inputs to another controller to switch the light via pulses. 1 = UI1, 2 = UI2, 255 = NotConnected
Push Button 1 Behavior	The configuration is intended to adapt the button operation to different needs. If the same functionality can be performed from both inputs via the same configuration, then the inputs are ored. Enum: • 1 = IgnoreButtonForLight, • 2 = OnOff, • 3 = OnOffDimming, • 4 = On, • 5 = Off, • 6 = OnDimmingUp, • 7 = OffDimmungDn, • 8 = Staircase
Push Button 2 Terminal	See description of **Push Button 1 Terminal**.
Push Button 2 Behavior	See description of **PushButton 1 Behavior**.
Light Address	Set the light address decimal value. If the push button input is shared from one LightA function block to another LightA function block, then light address needs to be set. A maximum of 16 lights can be controlled from a single push button. NotUsed = 0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768.
Push Button1 Address Bits	Normally there are 1 or 2 push buttons for each light function block, which can only switch the light which is controlled by the function block. However, if a pushbutton is to switch a light in another function block, then address of the other light must be selected here. Example: This push button should work with this light (i.e. LightAddress = 01) and with Light 02 and Light 08 => Enter 1 + 2 + 8 = 11
Push Button2 Address Bits	See description of Push Button1 Address Bits.

Table 284 Parameter of Light A (Continued)

Parameter Name	Description
Push Button Dimming Send On Delta	This setting is only used, if one of the button is used to switch a light from another function block. When dimming, the current value is sent to the other lamp when a certain change occurs. If the other lamp is in the same controller, the offset is only 1 DDC cycle (500 msec). Default=1%
Power Terminal	This defines the terminal that supplies the lamp with power to drive the lamp. In the case of an on/off lamp, this is used to switch the lamp on and off. In the case of a dimmable lamp, this is used for the power supply while the Light Dimming Terminal is used to adjust the brightness. A dimming lamp dim between 0.1 - 100%. RO1, RO2, RO3, RO4, TO1, TO2, TO3, TO4, Default = NotConnected
Dimming Terminal	This terminal is used only with a dimmable lamp. AO1, AO2, AO3, AO4, AO5, AO6, Default = NotConnected
Dimming Terminal Characteristics	Set the characteristics. See also **Min Output** and **Max Output**. 1 = means 0-10V 2 =means 1-10V (Default)
Min Output Par (In & Par)	Dimming is performed between Min Output and **Max Output**, if the corresponding Function Bits are set. Default = 10 %
Max Output Par (In & Par)	Dimming is performed between **Min Output** and **Max Output** if the corresponding Function Bits are set. Default = 100 %
Switch On Output Par (In & Par)	Set the **Switch On Output** value in percentage. This defines the **Switch On Output** for a dimmable lamp. The lamp is switched on with this value. If the Operation Mode Bit "Memory" is activated, the Switch On Value is used until a "Memory" value has been learned after manual dimming. ***Note:*** *The memory value is lost after Power Up.* The **Switch On Output** value should be between **Min Output** and **Max Output**. If “On_After_PowerUp “is enabled, then the light is switched on to this value too. Default = 100%

Table 284 Parameter of Light A (Continued)

Parameter Name	Description
Dimming Speed	The "Dimming Speed" is only relevant when dimming via the push buttons. According to the PushButtonBehavior, dimming only takes place when the Push Buttons are pressed for a long time, i.e. only then will the dimming take place between 0.1 - 100 % or between MinOutput and MaxOutput according the dimming speed. For all other commands via EmergencyCmd, MotionBrightnessSensor, ManualOverrideCmd, ApplicationCmd the **DimmingSpeed** is ignored, but the SoftSpeed is relevant. Example: A short press switches the light On or Off immediately according the SoftSpeed, but without the **DimmingSpeed**. A long press dims the light between MinOutput and MaxOutput according to the **DimmingSpeed** but without the SoftSpeed. The **DimmingSpeed** always refers to a dimming range of 0.1-100 % or 100-0 %, independent of MinOutput and MaxOutput. Example: **DimmingSpeed** = 10 sec, Min = 10%, Max = 90% => Dimming between 10 % and 90 % requires 8 sec. 1-300.0 sec, 10.0 sec
Soft Speed	If the operation mode bits Enable_SoftOn_DimUp, Enable_SoftOn_DimDown or Enable_Soft_Off are activated, the light is not switched on or off abruptly, but the light is softly dimmed up or down. The time describes the speed of dimming from 0.1-100 % and vice versa. 1-300.0 sec, 2.0 sec
Auto Off Time Par (In & Par)	See the Input Auto Off Time for description. Default= 1440 min.
Permanent Time	The **Permanent Time** is activated and deactivated in the staircase via a long press. This allows the staircase light to be switched on for a long period of time if required. See OperationModeBit Enable_PermanentTime. 0-10000 min, 180 min
Switch Off Delay Time Par	See the Input **Switch Off Delay Time** for description. Default = 30 sec.

Table 284 Parameter of Light A (Continued)

Parameter Name	Description
Function Bits Par (In & Par)	This configures the behavior of the lamp under various conditions. • On_After_PowerUp = 1 • Off_After_PowerUp= 2 • Off_If_AutoOffTime_is_expired = 4 • On_If_Occ_and_Dark = 8 • On_and_Restart_AutoOffTimer_If_Occ_and_Dark = 16 • Off_If_Unocc_or_Bright = 32 • Enable_Off_If_AutoOffTimer_is_expired_during_Occ_and_Dark = 64 • Enable_Off_From_ApplicationCmd_If_Occ_and_Dark= 128 • Enable_On_With_Memory_Value = 256 • Enable_SoftOn_DimUp = 512 • Enable_SoftOn_DimDown = 1024 • Enable_SoftOff = 2048 • Enable_SwitchOffWarning = 4096 • Enable_SwitchOff30secDelay = 8192 • Enable_LimitToMin = 16384 • Enable_LimitToMax = 32768 • Enable_ExtensionTime = 65536 • Enable_PermanentTime = 131072 • Null • Default=249909
Operating Mode Bits Par	See the Input Operating Mode Bits for description. Default = 0.

The below diagram explains the light addressing.

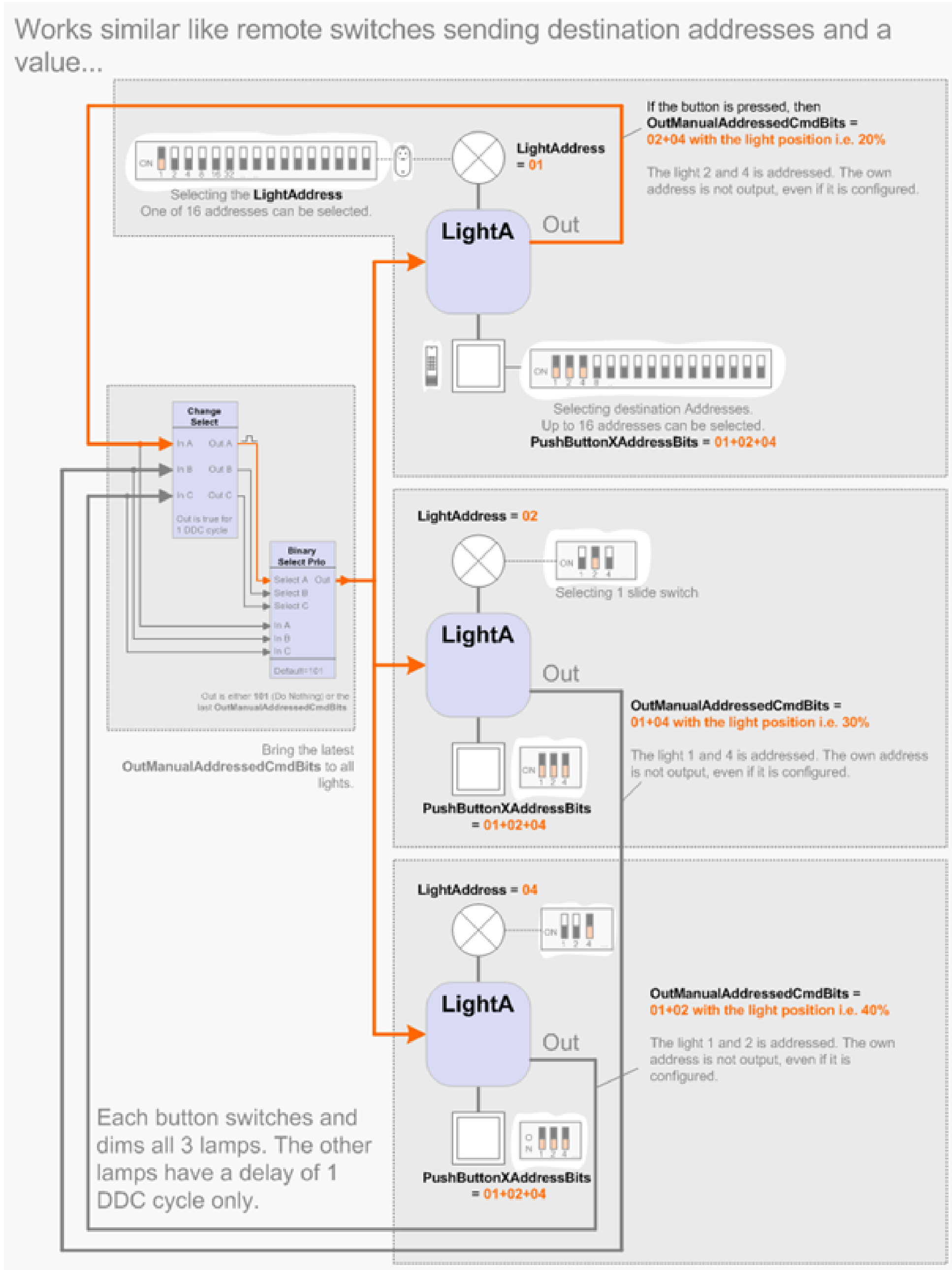
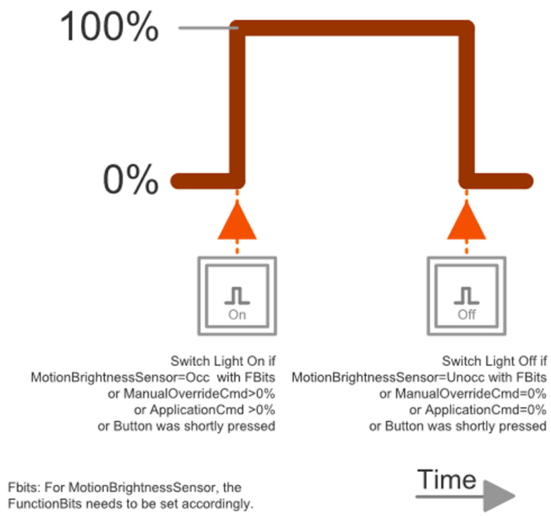


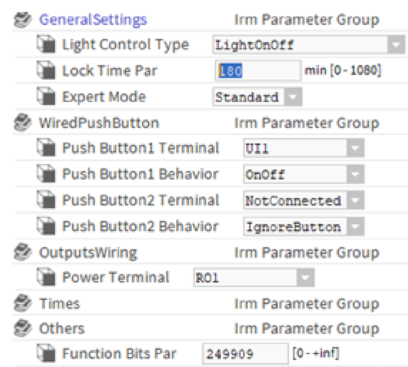
Fig. 414 LightA Function Block-Light Addressing

Examples

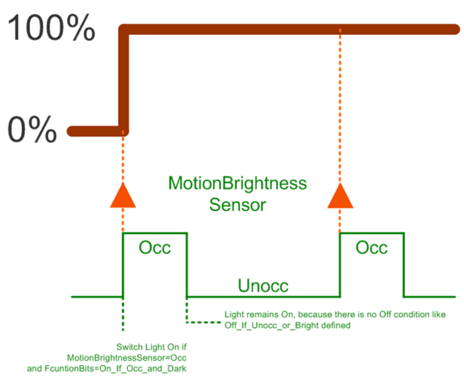
Lights Simple On/Off

Below is the example for simple light on/Off. Any input terminal will be configured as push button Eg: UI1 is configured as an OnOff push button. First press switches the light “On”. Second press “Off”. If a Lock Time is entered, then the light remains for the Lock Time in the manually switched position (if no Emergency Command comes in). An Application Cmd on the input is ignored then. If the lock time is expired, then the light switches to the last application command. Any output terminal will be configured as output Eg: “RO1”.

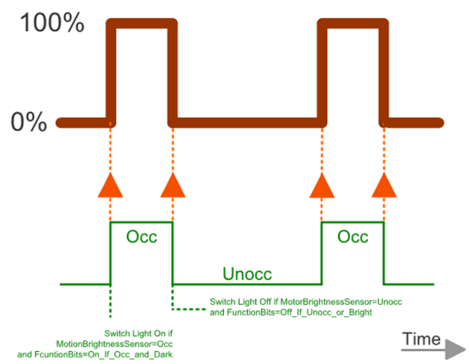




LightOnOff (Light Ref 01)



LightOnOffTimer / LightDimmingTimer (Light Ref. 05)
+On_If_Occ_and_Dark



LightOnOffTimer / LightDimmingTimer (Light Ref. 06)
+Off_If_Unocc_or_Bright
+On_If_Occ_and_Dark

Fig. 415 Example: Light Simple On / OFF

Below is an example for an application command. An Application command is on based on the combination of conditions like Occupancy schedule, brightness, darkness, time, etc.

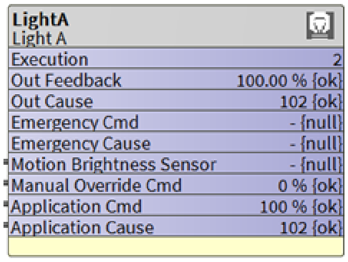


Fig. 416 Light A function Block- Application Command

Light OnOffTimer / LightDimmingTimer

The below example is for light OnOffTimer LightDimmingTimer + Off_If_AutoOffTime_is_expired.

Any input terminal will be configured as push button i.e. UI1 configured as "On dimming Up" UI2 configured as "Off dimming down" and once the PB is pressed or application or manual command is on then the light will be ON until the auto off time par is expired then it will back to application command.

Any output terminal will be configured as output i.e. "RO1".

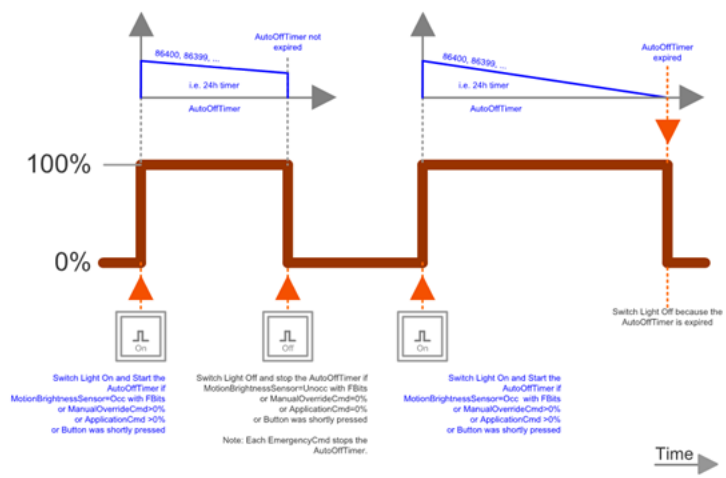


Fig. 417 light OnOffTimer/LightDimmingTimer+Off_If_AutoOffTime_is_expired

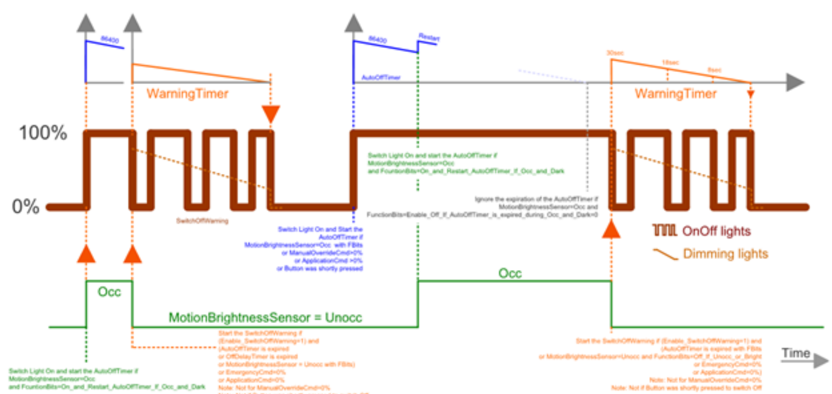


Fig. 418 LightOnOffTimer / LightDimmingTimer (Light Ref 07)

- +Off_If_AutoOffTime_is_expired
- +Enable_SwitchOffWarning
- +On_and_Restart_AutoOffTimer_If_Occ_and_Dark
- +Off_If_Unocc_or_Bright

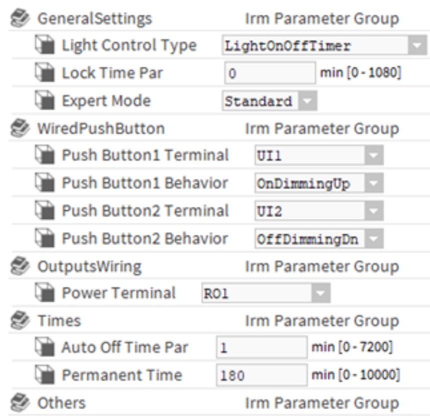


Fig. 419 Example- Light OnOffTimer / LightDimmingTimer

Light OnOffTimer / LightDimmingTimer with off delay timer

Example: The below example is for Light OnOffTimer/ Light.

DimmingTimer = Off_If_AutoOfftime_is_expired with swith of delay timer. Any input terminal will be configured as push button i.e. UI1. will be On dimming Up UI2 will be Off dimming down and once the PB is pressed or application or manual command is on then the light will be ON until the auto off time par is expired then it will back to application command. Any output terminal will be configured as output i.e. "RO1 and AO1 is configured as dimming terminal. Switch off warning will three time turn off and turn on the lights with default of 30 secs delay.

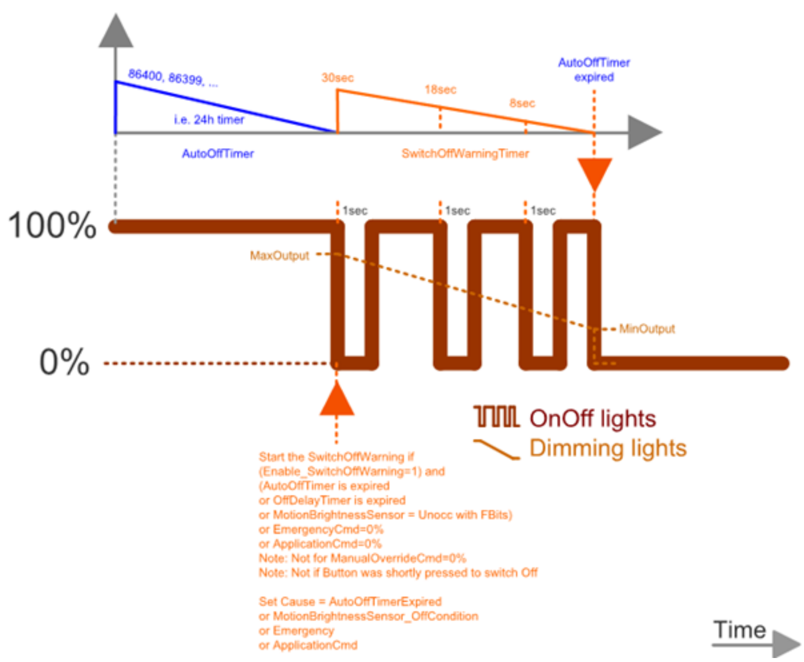


Fig. 420 LightOnOffTimer / LightDimmingTimer (Light Ref. 03)

- +Off_If_AutoOffTime_is_expired
- +Enable_SwitchOffWarning

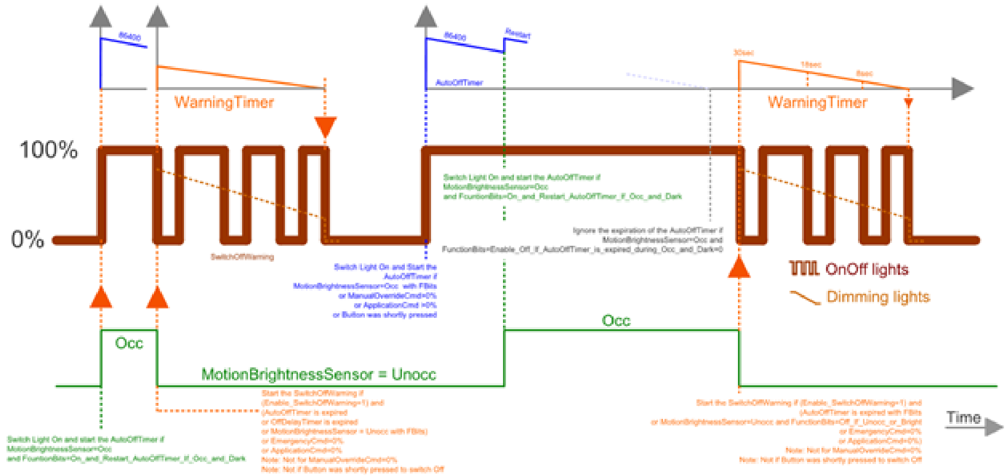


Fig. 421 LightOnOffTimer / LightDimmingTimer (Light Ref. 07)

- +Off_If_AutoOffTime_is_expired
- +Enable_SwitchOffWarning
- +On_and_Restart_AutoOffTimer_If_Occ_and_Dark
- +Off_If_Unocc_or_Bright

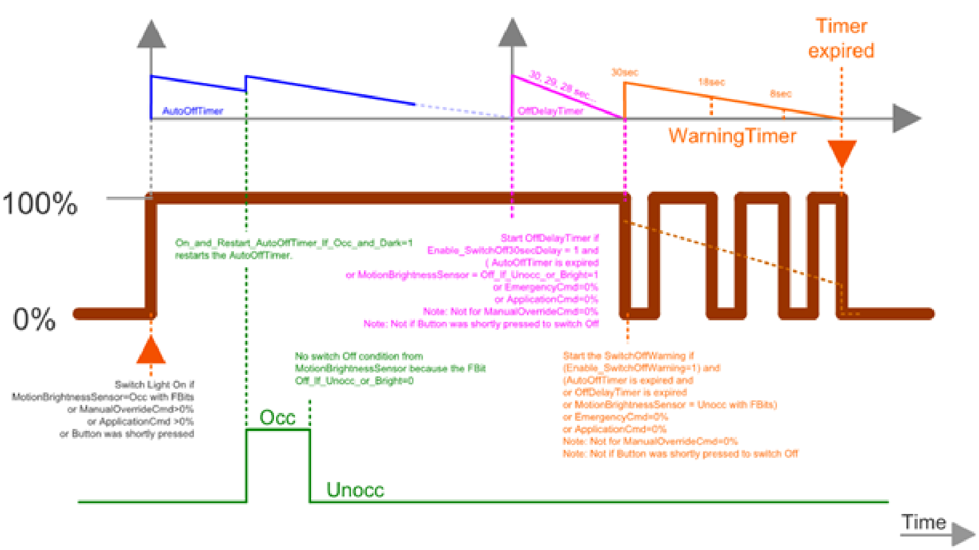


Fig. 422 Light OnOffTimer / LightDimmingTimer (Light Ref. 12)

- + Off_If_AutoOffTime_is_expired = 1
- + On_and_Restart_AutoOffTimer_If_Occ_and_Dark = 1
- + Off_If_Unocc_or_Bright = 1
- + Enable_Off_If_AutoOffTimer_is_expired_during_Occ_and_Dark = 0
- + Enable_SwitchOff30secDelay = 1
- + Enable_SwitchOffWarning = 1

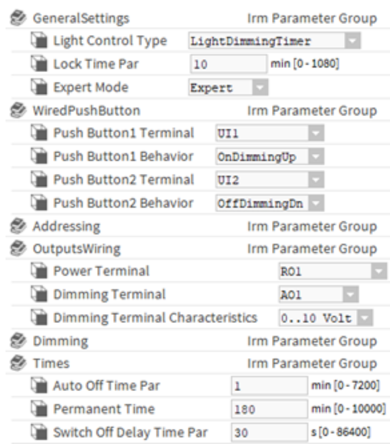


Fig. 423 Example- Light OnOffTimer / LightDimmingTimer with off delay timer

The below example is for Light OnOff staircase. Any input terminal will be configured as push button i.e. UI1 will be configured as on switch or manual command or emergency command is on then the light will be ON until the auto off time par is expired. Any output terminal will be configured as output i.e. "R01 is configured as Light On after the auto off time expired Switch off warning will three time turn off and turn on the lights with default of 30 secs delay.

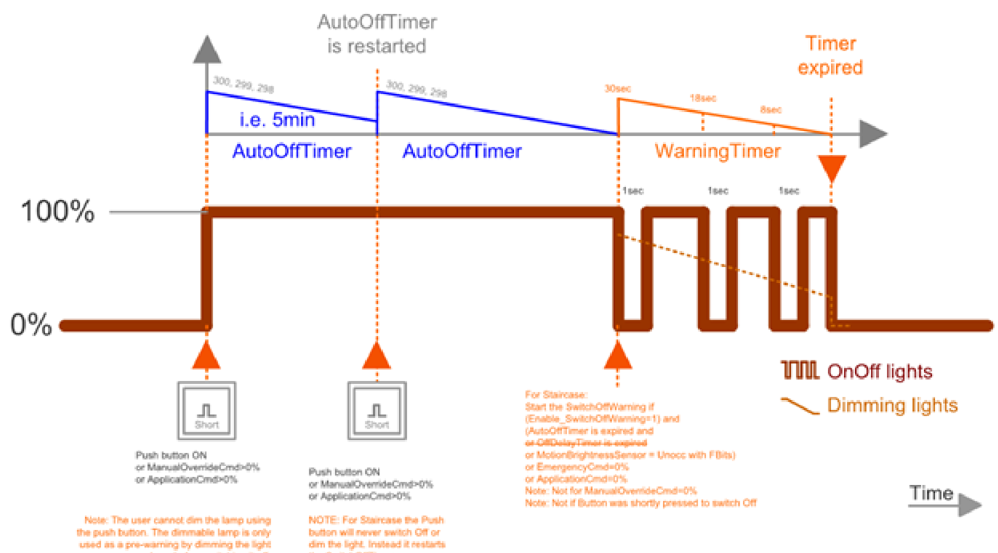


Fig. 424 LightOnOffStaircase / LightDimmingStaircase (Light Ref. 13)

- + Off_If_AutoOffTime_is_expired
- + Enable_SwitchOffWarning

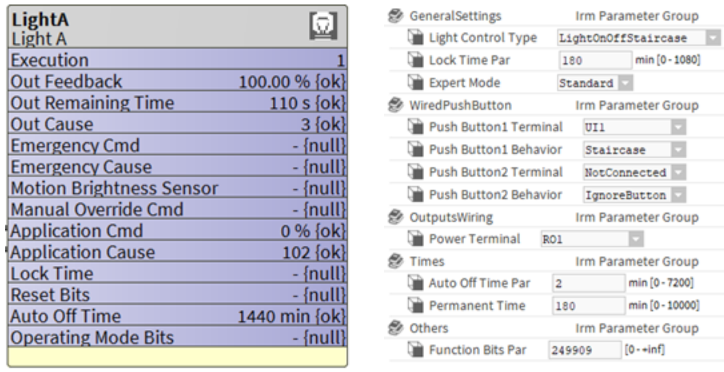


Fig. 425 Example – LightA OnOff staircase

LightOnOffStaircase / LightDimmingStaircase

The below is the example for Light OnOff staircase / LightDimming staircase. Any input terminal will be configured as push button i.e. UI1 will be configured as stair case and UI2 is configured as lights Off or manual command or emergency command is on then the light will be ON until the auto off time par is expired. Any output terminal will be configured as output i.e. “RO1 is configured as Light On, AO1 is configured as Dimming. After the auto off time expired Switch off warning will three time turn off and turn on the lights with default of 30 secs delay.

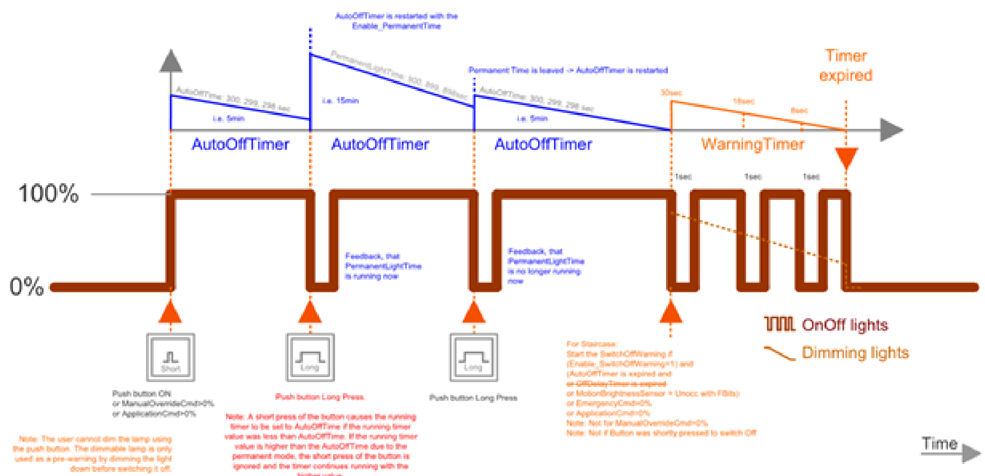


Fig. 426 LightOnOffStaircase / LightDimmingStaircase (Light Ref. 14)

- + Off_If_AutoOffTime_is_expired
- + Enable_SwitchOffWarning
- + Enable_PermanentLight

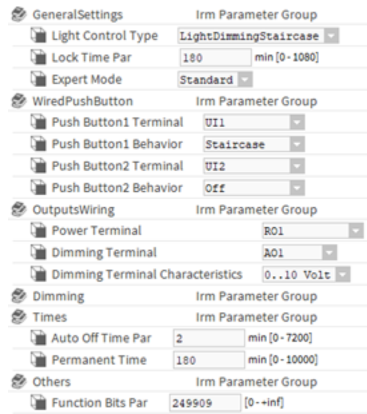


Fig. 427 Example: LightOnOffStaircase / LightDimmingStaircase

Staircase Light On /Off Auto off Timer extension

After each press of stair case push button the auto off timer value will be multiplied.

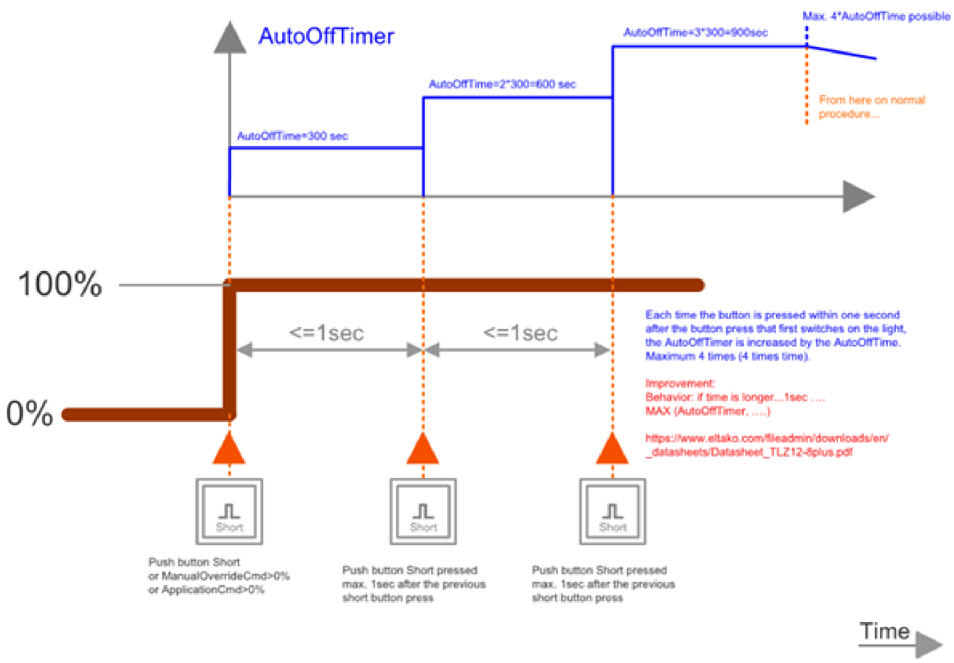


Fig. 428 LightOnOffStaircase / LightDimmingStaircase (Light Ref. 15)

- +Off_If_AutoOffTime_is_expired
- +Enable_SwitchOffWarning
- + ExtensionLight

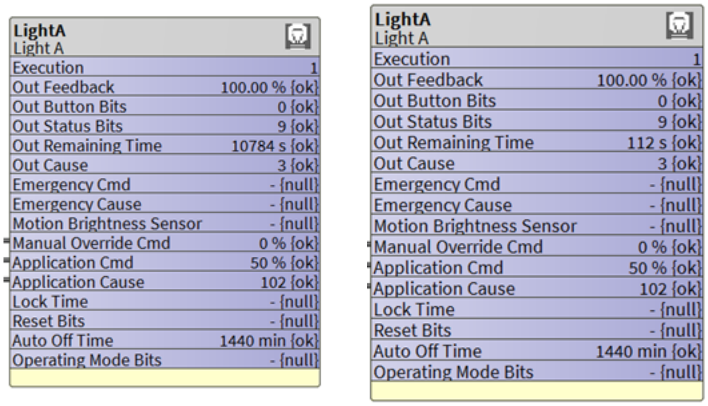


Fig. 429 Example- LightA OnOff staircase

Push Button Behavior

Table 285 Push Button Behaviors

Push Button Behavior	Short Press (< 2sec)	Long Press (>2sec-5sec)
IgnoreButton	This input is not used.	This input is not used.
OnOff	Toggle the lamp value between Off and On.	Toggle the lamp value between Off and On.
OnOffDimming	Toggle the lamp value between Off and the SwitchOnOutput or the Memory value if the OperationModeBit “Enable_On_With_Menmory_value” is enabled. After power up the Memory value = SwitchOnOutput until is is dimmed manually.	Dim the lamp between 0.100 % or between Min and Max if theFunctionBits “Enable_LimitToMin”/ “Enable_LimitToMax’’ are enabled.
On	A rising edge from a pulse switches On the light.	A rising edge from a pulse switches On the light.
Off	A rising edge from a pulse switches On the light.	A rising edge from a pulse switches Off the light.
OnDimmingUp	If the light is Off: Switches On the light to the SwitchOnOutput or the Memory value if the OperationModeBit “Enable_On_With_Memory_Value” is enabled.	• If the light is Off: Switches On the light to 1 % or to the Min value if the OperationModeBit “Enable_LimitToMin” is enabled (Do not switch On to the SwitchOnOutput or the Memory value). • If the light is On: The light is dimmed up to 100 % or to the Max value if the OperationModeBit “Enable_LimitToMax” is enabled.

Table 285 Push Button Behaviors (Continued)

OnDimmingDn	• If the light is Off: Ignore the button press. • If the light is On: Switches Off the light.	• If the light is Off: Ignore the button press. • If the light is On: Dims down the light until 1% or until the Min value if the OperationModeBit “Enable_LimitToMin” is enabled.
Staircase	• If the light is Off, then the light is switched On with 1 short button press. If the OperationModeBit “Extension” is enabled, the light is switched on in addition with a double click, triple click or quad click. Double click > 2xAutoOffTime; tripe-> 3xAutoOffTime; quad -> 4xAutoOffTime). Double/Triple/Quad click means that the button is pressed within a shorter time frame than 1 sec after the light is switched on with the first short button press. • If the light is On for > 1sec and the running Off timer is < 1xAutoOffTime, then a short button press sets the running Off timer to 1xAutoOffTime. • If the PermanentMode is active, the short button press is ignored. See description on the right column.In principle the light can never be switched off via the push-button!	If the OperationModeBit “Enable_PermanentTime” is enabled, then the Timer is started with the PermanentTime. This time is usually considerably longer (several hours), so that, for example, greater work can be carried out in the staircase without the light being switched off periodically. If the PermanentTime is already running, then a short press is ignored while a long press exits the permanent mode and restarts the timer with the AutoOffTime.

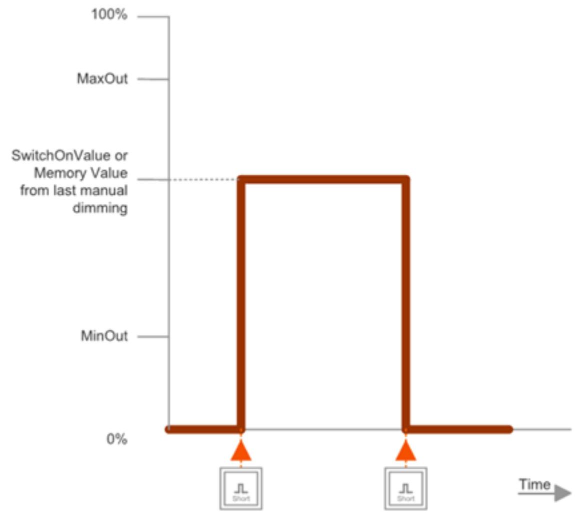


Fig. 430 Light Dimming (Light Ref 17)

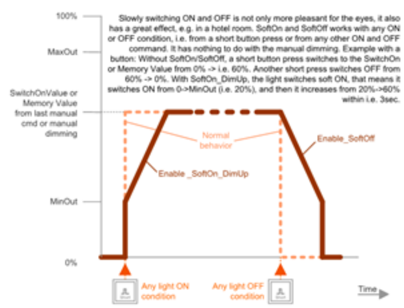


Fig. 431 LightDimming (Light Ref. 19)

- +Enable_SoftOn_DimUp
- +Enable_Soft_Off

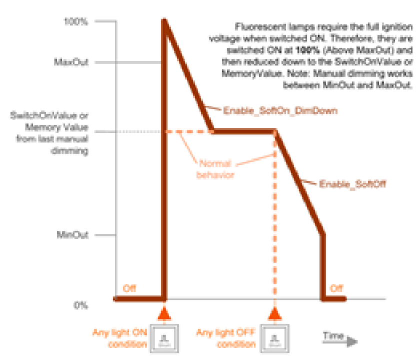


Fig. 432 LightDimming (Light Ref. 20)

- +SoftOnDimDn (Tubes)
- +Enable_Soft_Off
- +Enable_LimitToMin
- +Enable_LimitToMax

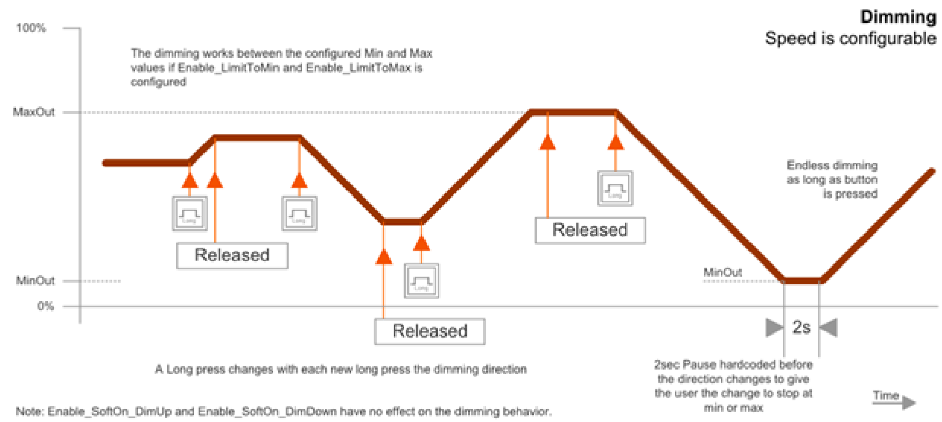


Fig. 433 Dimming with 1 Push Button (Light Ref. 21)

- + Enable_LimitToMin
- + Enable_LimitToMax

CHAPTER

17 SUNBLIND FUNCTION BLOCK

The following Blind A function blocks are available in the honIrmControl Palette and can be configured and used to create the required application logic:

- Blind A

Fig. 434 Sunblind Function Block

Refer to General Description on page 23 for general information about Function Blocks.

Blind A

The Blind A function block is used to control a sunblind, roller shutter or an electrically driven window. The control includes open/ close, stop, push button action, maintain some specific position from 0 - 100% and slat angle position between -180 degrees to +180 degrees along with various priorities (emergency, manual commands and application). To reliably detect a short button press, buttons are read in directly within the function block without a connection from the On Board IO. The wired outputs are also provided within the function block without a connection to the On Board IO.

Note: *A maximum of 4 Blind A function blocks can be used in an application.*

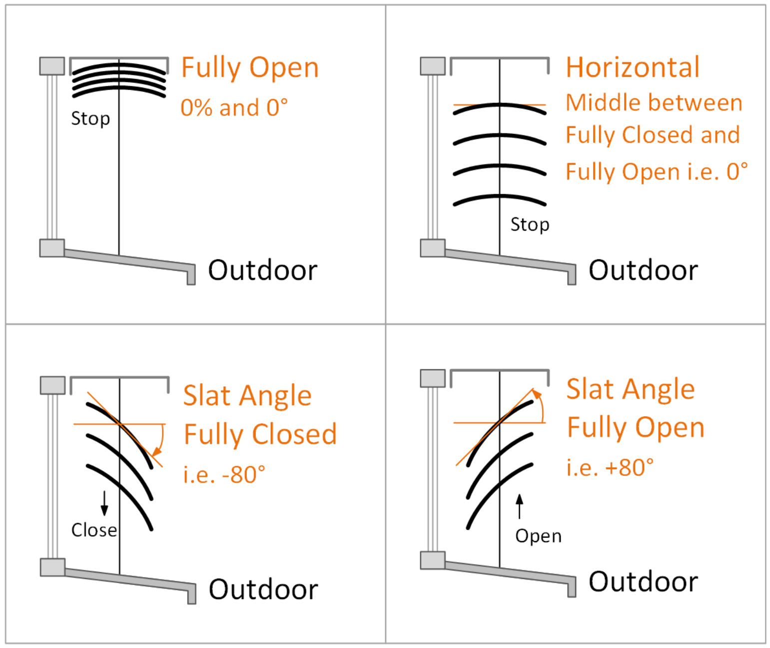
Note: *To achieve the best performance for the operation of blind, the blind function block must be put into the Event program folder. As soon as a wired button is pressed or a Modbus message is received (the Modbus function block must also be in the Event Program Folder for this to happen), the Event Program Folder is executed immediately.*

The Blind A supports various modes of operations along with three different input priorities:

- **Emergency**
- **Manual**
- **Application**

- **Emergency:** The emergency input has the highest priority. If the emergency mode is activated, the manual operation and the application input will be overridden. Emergency mode examples are Fire, Wind, Rain, Ice etc.
- **Manual:** The manual input is used to enable manual override operation. Example: Via BACnet or wired pushbuttons and operator commands from a Niagara frontend. The manual command via the input **Manual Override Cmd** and the directly wired buttons work according to the "Last Wins" principle. Commands via the **Manual Override Cmd** input are executed once when the input is changed. The manual commands will be the second higher priority. As soon as the manual command is set (via wired push button or via **Manual Override Cmd**), a **Lock Time** count down will be initatied. **Lock Time** is the time which the manual command will sustain. If the **Lock Time** count down is elapsed, then manual command will be overriden by application command.
- **Application:** The application command input serves as input of the self-programmed "automatic" logic. The input has the lowest priority. Depending on brightness, position of the sun, room occupancy, cooling or heating mode, different logic can be programmed which can, however, be overridden by a manual operation (directly wired or over manual inputs). While the **Lock Time** is running, various automatic commands can be set to the **Application Cmd** input, e.g. ToFullyOpenPosition, ToFullyClosePosition, ... All these commands are not executed if the **Lock Time** is running. As soon as the **Lock Time** has expired, the last valid Application command is executed (The command "Do Nothing" does not apply).

An important feature is the cause. The programmer can optionally connect a cause to the function block. Depending on which input is currently being used (whether via one of the 3 input slots or wired), the correct cause is given to the output. This makes it easy to see why the function block behaves like this.



Depending on the point of view, i.e. whether the outside is on the right or left side, the configured slat angles are positive or negative.

Below mentioned are the blind commands and types of action which will be performed by BlindA.

Table 286 Blind Commands

Command	Description
01 = Stop	A moving blind or shutter is stopped.
02 = Man Open Used for manual Ovrd Cmd	A stopped blind or shutter (depending on **Position Logic**) is moved to the open position (-> 0 %) with an Extended Time; a Window is moved to the open position (-> 100 %). A manually started motor that is still running is stopped. Extended Time for the blind (**Position Logic** = Normal): With a target position to 0 % and an angle >= Slat Angle Fully Open, the motor runtime is calculated from the time to drive to 0 % + 50 % of the **Time To Open**. The Extended Time is terminated immediately on receipt of a new command or when a pushbutton is pressed; the 0 % 0° position is assumed as the current position.

Table 286 Blind Commands (Continued)

Command	Description
03 = Man Close Used for manual Ovrd Cmd.	A stopped blind or shutter (depending on PositionLogic) is moved to the closed position (->100 %); a Window is moved to the close position (-> 0 %) with an Extended Time. A manually started motor that is still running is stopped. After reaching the close position of a blind, the motor is either switched off and the slat remains closed or the slat is opened to bring light into the room depending on Slat Alignment (*1). Extended Time for the window (**Position Logic** = Inverse): With a target position to 0 %, the motor runtime is calculated from the time to drive to 0 % + 50 % of the **Time To Close**. The Extended Time is terminated immediately on receipt of a new command or when a pushbutton is pressed; the 0 % 0° position is assumed as the current position.
04 = ToPositionAndAngle	The command moves the blind to any position. The slots Manual Override Position and Manual Override Angle are used. Only valid values are evaluated, i.e. only the position or only the angle can be moved to the desired position. The angle is only evaluated for the DeviceType “Blind”. Extended Time: With a target position to 0 % and an angle >= Slat Angle Fully Open, the motor runtime is calculated from the time to drive to 0 % + 50 % of the **Time To Open** (**Position Logic** = Normal) or **Time To Close** (**Position Logic** = Inverse). The Extended Time is terminated immediately on receipt of a new command or when a pushbutton is pressed; the 0 % 0° position is assumed as the current position.
05 = ToFullyOpenPosition Used for Emergency Cmd and Application Cmd	The blind, shutter is moved to the open position (-> 0 %) with an Extended Time; the window is moved to the Open position (-> 100 %). The command is immediately terminated by another command, but not by repeating the same command in the next DDC cycle. Extended Time for the blind (**Position Logic** = Normal): With a target position to 0 % and an angle >= Slat Angle Fully Open, the motor runtime is calculated from the time to drive to 0 % + 5 0% of the **Time To Open**. The Extended Time is terminated immediately on receipt of a new command or when a pushbutton is pressed; the 0 % 0° position is assumed as the current position. ***Note:*** *If the **Emergency Cmd** is active, the push button and the **Application Cmd** are ignored.*

Table 286 Blind Commands (Continued)

Command	Description
06 = ToFullyClosedPosition Used for Emergency Cmd and Application Cmd	The blind, shutter is moved to the close position (-> 100 %); the window is moved to the Closed position with an Extended Time. After the blind has reached the close position, the motor is either switched off and the slat remains closed or the slat is opened to bring light into the room depending on Slat Alignment (*1). The command is immediately terminated by another command, but not by repeating the same command in the next DDC cycle. If a second "To Fully Closed Position" is received, then the command is executed again; this means after the SlatAlignment, the angle is moving to fully closed angle and then slat alignment is done again. So it is very important to follow the principle, that the application sends in normal case "Do Nothing" and only if the logic has changed the „To Fully Closed Position" command. Extended Time for the window (**Position Logic** = Inverse): With a target position to 0%, the motor runtime is calculated from the time to drive to 0% + 50% of the **Time To Close**. The Extended Time is terminated immediately on receipt of a new command or when a pushbutton is pressed; the 0 % 0° position is assumed as the current position. ***Note:*** *If the **Emergency Cmd** is active, the push button and the **Application Cmd** are ignored.*
07 = Slats Horizontal	This moves the slat to the horizontal position. Typically, this command is issued by a wall panel or a Scene. The command is only evaluated for a DeviceType = Blind.
08 = Man Angle Step Positive Used for Manual Ovrd Cmd	The slat angle is controlled step by step. Each time the button is shortly pressed or with each "ManAngleStepPositive" command, the slat angle is moved one step in the positive direction. See parameter Slat Steps. The command cannot be stopped by another manual command.
09 = Man Angle Step Negative Used for Manual Ovrd Cmd	The slat angle is controlled step by step. Each time the button is shortly pressed or with each "ManAngleStepPositive" command, the slat angle is moved one step in the negative direction. See parameter Slat Steps. The command cannot be stopped by another manual command.

Table 286 Blind Commands (Continued)

Command	Description
10 = DoNothing	The command is functionally ignored. The principle of programming is, that basically always only "Do Nothing" commands are permanently present at the inputs (except for **Emergency Cmd**) and only in case of a real change for 1 DDC cycle a command is given to the inputs. This can be easily achieved with a chain of “Prev Value” and “Binary Select” functionblocks. With the Out Cause you can see what happened at the function block. ***Note:*** *Leave the **Emergency Cmd** at the input slot as long as Emergency is valid.*
11 = Last Valid Application Cmd	The last **Application Cmd** is saved in RAM (not DoNothing). This command is used after the **Lock Time** has expired (Taking into accound of the priority). In addition it can also be used as an **Emergency Cmd** before the emergency input is set to „DoNothing“. Last Valid **Application Cmd** is evaluated event triggered. • If Last Valid **Application Cmd** is set, at this point of time the Last Valid **Application Cmd** is set one time. Then the command is ignored. If used for **Emergency Cmd**, then the Emergency Mode is then terminated (like “Do Nothing”). • If used for **Manual Override Cmd**, then the Lock time is stopped/reset and the Last valid application cmd is set immediately.
12 = Start Reference Drive Used for Emergency Cmd and Application Cmd	A **Reference Drive** is started (Independent of the **Reference Drive** parameter) and the 24h timer is reset. When the **Reference Drive** is already running and the “**Start Reference Drive**” command is triggered, then another Reference Drive is started. The reference Drive timer starts if the position is not 0 % 0°. The **Start Reference Drive** command as **Application Cmd** is ignored if an **Emergency Cmd** is active. “**Start Reference Drive**” can also come as an “**Emergency Cmd**”. After the “**Start Reference Drive**”, the last valid command from **Application Cmd** or **Emergency Cmd** is used as new position while Push Buttons and Manual Ovrd Cmd are ignored during the reference drive.

**1 The slat alignment means that after fully closing a blind (DeviceType = Blind), the slats are moved into a "Slat Alignment"position to get light into the room.*

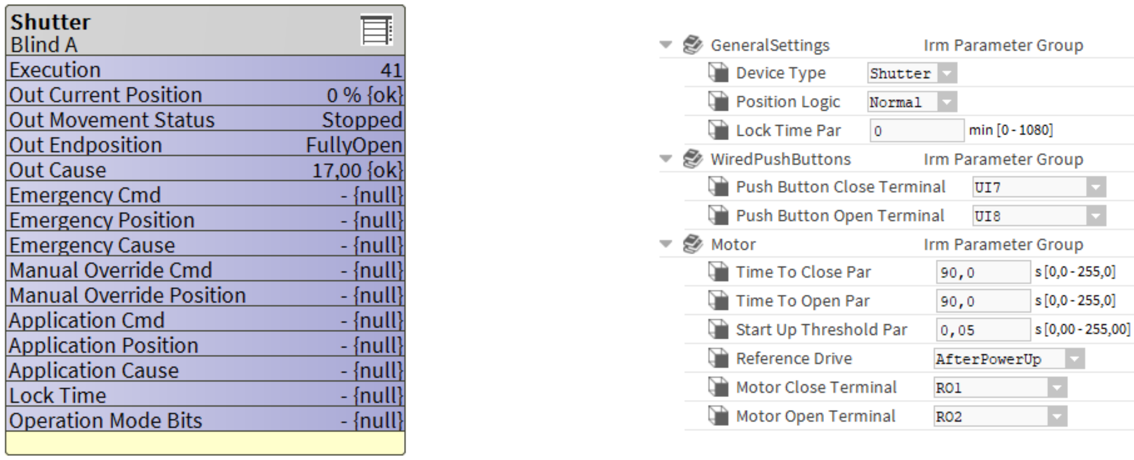


Fig 464. BlindA Function Block and Property sheet for Shutter Application

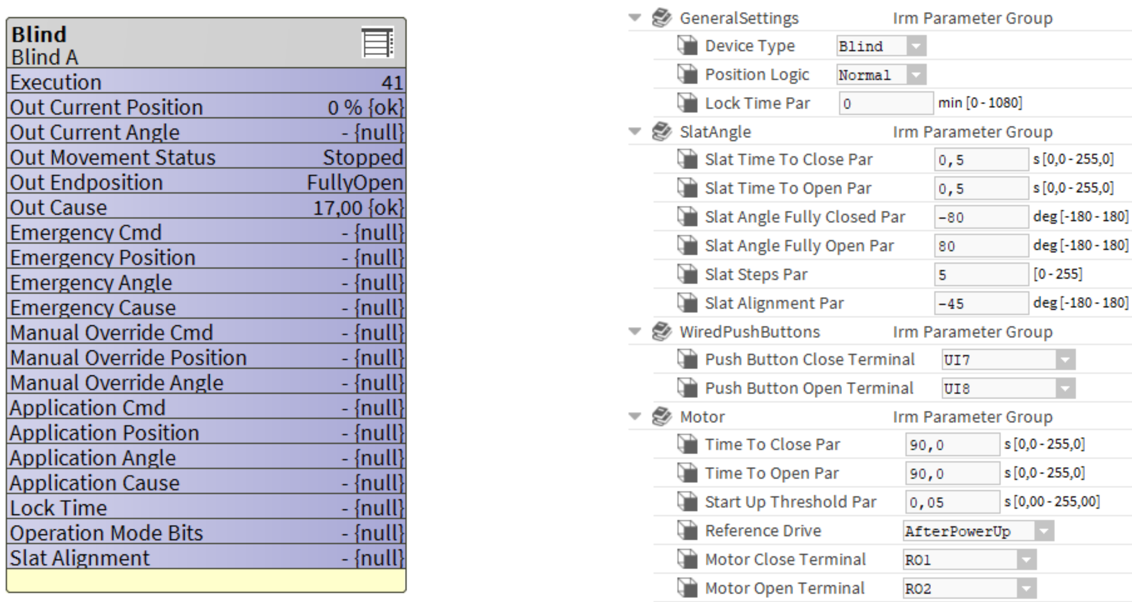


Fig. 465 BlindA Function Block and Property Sheet for Sun Blind

Input

Table 287 Inputs of Blind A

Input Name	Description
Emergency Cmd	This input has the highest priority. The manual operation from wired push buttons or via **Manual Override Cmd** as well as the **Application Cmd** are ignored. The Emergency mode is terminated with the command "Do Nothing" or with "LastValidApplicationCmd”.
Emergency Position	This position is approached if the **Emergency Cmd** obtains the command "ToPositionAndAngle". Each value change is immediately approached in a new position if the "ToPositionAndAngle" command is given. Numeric: 32 bit integer value 0 - 100%, NULL = Keep Position
Emergency Angle	The input is only evaluated if the **Device Type** = Blind has been configured. This angle is approached if the **Emergency Cmd** obtains the command "ToPositionAndAngle". Each value change is immediately approached in a new angle if the "ToPositionAndAngle" command is given Numeric: 32 bit integer value -180 deg to +180 Deg; NULL = Keep Angle
Emergency Cause	For debug reasons, it is good to have a cause to understand why the blind is in this state. This input is looped through to the Out Cause when the **Emergency Cmd** is executed. These values are referenced more often • 151 = WindLow • 152 = WindHigh • 152 = Service • 153 = NoService • 153 = Fire • 155 = NoFire • 154 = Rain • 155 = NoRain • 156 = Ice (Outside Air temp) • 157 = NoIce (Outside Air temp)

Table 287 Inputs of Blind A (Continued)

Input Name	Description
Manual Override Cmd	This input serves as manual operation via an external wall module or external push-buttons (BACnet, Modbus etc) or via a Niagara graphic. The input is set in that DDC cycle with the corresponding positioning command when the manual action has been executed, e.g. when a key is pressed. Otherwise the input is written with "Do Nothing". To repeat a command, a “Do Nothing” must be set in between. Please also look for the "3 upper bits" description in the command list.
Manual Override Position	In that DDC cycle, where **Manual Override Cmd** has the command "ToPositionAndAngle", this input is evaluated and controlled. Any following value changes at this input are ignored. Numeric: 32 bit integer value 0-100%, NULL = Keep Position
Manual Override Angle	The input is only evaluated if the **Device Type** = Blind has been configured. In that DDC cycle, where **Manual Override Cmd** receives the value "ToPositionAndAngle", this input is evaluated and controlled. Any following value changes at this input are ignored. Numeric: 32 bit integer value -180 to +180 deg NULL = Keep Angle
Application Cmd	The input has the lowest priority. The manual operation via wired push buttons or via the Manual Ovrd Cmd is still working. The input is used to set the automatic control to fullfill i.e. EN15232 programmed in the wiresheet (Depending on Brightness, occupancy...). The input is set in that DDC cycle with the corresponding positioning command when the application condition has changed, e.g. if the brightness changes between dark and bright or vice versa or the occupancy mode changes. Otherwise the input is written with "Do Nothing". The last command before “Do Nothing” is stored inside the function block so that it can be executed when the Emergency is finished or the Locktime has expired or with the command “LastValidApplicationCmd” from wallmodule / panel.
Application Position	In that DDC cycle, where **Application Cmd** has the command "ToPositionAndAngle", this input is evaluated and controlled. Any following value changes at this input are ignored. Numeric: 32 bit integer value 0-100%, NULL = Keep Position

Table 287 Inputs of Blind A (Continued)

Input Name	Description
Application Angle	The input is only evaluated if the **Device Type** = Blind has been configured. In that DDC cycle, where **Application Cmd** receives the value "ToPositionAndAngle", this input is evaluated and controlled. Any following value changes at this input are ignored. Numeric: 32 bit integer value -180 to +180 Deg NULL = Keep Angle
Application Cause	For Debug reason it is good to have a cause to understand why the blind is in this state. This input is looped through to the Out cause when the Application command is executed. These values are referenced more often • 100 = Bright • 101 = Dark • 102 = OccOccupied • 103 = OccOff • 104 = OccHoliday • 105 = OccUnoccupied • 106 = Standby • 107 = Bypass • 108 = OccSensorOcc • 109 = OccSensorUnocc • 110 = Time • 111 = TimeWithPresenceSimulation • 112 = Sunrise • 113 = Sunset • 114 = Morning dawn • 115 = Evening dawn • 116 = GlareProtection (Not fully negative angle) • 117 = SightProtection (Fully negative angle) • 118 = InsulationSummer (Room Temp)

Table 287 Inputs of Blind A (Continued)

Input Name	Description
	• 119 = InsulationWinter (Room Temp) • 120 = DaylightRegulation (Date, Time, Location) (Max Ligth w/o glare) • 121 = WindowOpen • 122 = WindowClosed • 123 = DoorOpen (DisableApplication, Disable closing) • 124 = DoorClosed • 125 = DifferentialTempHigh (RoomTemp <-> Outside Air Temp) • 126 = DifferentialTempLow (RoomTemp <-> Outside Air Temp) • 127 = AstroMorning (Date, Time, Location) • 128 = AstroEvening (Date, Time, Location) • 129 = ShockSensorActive • 130 = ShockSensorDeactive • 131 = RoomTempHigh (Allow Open; Start Auto) • 132 = RoomTempLow • 133 = RoomHumidityHigh (Open Window) • 134 = RoomHumidityLow (Close Window) • 135 = AutomaticOn • 136 = AutomaticOff
Lock Time	After a manual command via wired push button or via **Manual Override Cmd**, a lock time is activated to ignore the Application cmd for this time. This prevents the application from immediately overriding / canceling the manual command. Example: A User set the Blind manually via the push button to the desired position. 1 min later, the brightness changes. To avoid any change on the Blind, the **Lock Time** ensures that the Blind remains on the manual position. Note: An Emergency Command will have more priority than a manual command with a **Lock Time**. 0min and null means No Lock. Numeric: 32 bit floating Value, 0 – 1080 Min, NULL

Table 287 Inputs of Blind A (Continued)

Input Name	Description
Operation Mode Bits	Normally the Operation Mode Bits are not needed. However, during commissioning they can be quite useful or if a very special case needs to be covered in order to implement the desired function. • 1 = Drive to 0% position • 2 = Drive to 100% position • 4 = Drive angle to fully open position • 8 = Drive angle to fully closed position • 16 = Lock wired push button close • 32 = Set motor to power off State • 64 = Lock wired push button open • 128 = Lock wired push button close • 256 = Enable push button close • 512 = Enable push button open • 1024 = Enable manual command during emergency • 2048 = Balcony door Open • 4096 = Motor must no longer be switched on in the 0 % position, also not during emergency.
Slat Alignment (In & Par)	After the command "ManClose" or "ToFullyClosedPosition", the slat is moved to a defined position to bring light into the room. Since this Slat Alignment position depends on the position of the sun (summer/winter), the configuration is also available as an input slot. Numeric: 32 bit integer value -180 deg to +180 Deg NULL = Keep Angle
Time To Close (In & Par)	The time is used to calculate the required running time for absolute positioning. Measured is the time from position fully open to fully closed including the Slat **Time To Close** runtime. For the blinds and shutters (PositionLogic = Normal) this is the time to move down, for the window (PositionLogic = Invers) this is the time to close the window. Measured from Switching ON the Close - Output until Motor stops automatically in the fully open position. Numeric: 32 bit floating Value;0 - 255.00 secs, NULL
Time To Open (In & Par)	The time is used to calculate the required running time for absolute positioning. Measured is the time from position fully closed to fully open including the Slat **Time To Open** runtime. For the blinds and shutters (PositionLogic = Normal) this is the time to move up, for the window (PositionLogic = Invers) the time to open the window. Measured from Switching ON the Open - Output until Motor stops automatically in the fully closed position. Numeric: 32 bit floating Value, 0 - 255.00 secs, NULL

Table 287 Inputs of Blind A (Continued)

Input Name	Description
Start Up Threshold (In & Par)	This is the time it takes for the drive to start moving. The motor must not be switched on for less than this time. If the calculated switch-on time is shorter, the new position/angle is not addressed. The time is delivered by the supplier of the drive. For example, it makes no sense to switch ON the actuator for 1 msec. Numeric: 32 bit floating Value, 0 - 255.00 secs, NULL
Slat Time To Close (In & Par)	Running time in seconds (one decimal place) of the blind to rotate from the open slat angle end position to the close end position. This running time must be measured manually and then entered. If there are no slats, e.g. for a shutter, the value is 0 sec. The value is measured after a close command (Down). Numeric: 32 bit floating Value, 0 - 255.00 Secs, NULL
Slat Time To Open (In & Par)	Running time in seconds of the blind to rotate from the close slat angle end position to the open-end position. This running time must be measured manually and then entered. If there are no slats, e.g. for a shutter, the value is 0 sec. The value is measured after an open command (Up). Numeric: 32 bit floating Value;0 - 255.00 secs, NULL
Slat Angle Fully Closed (In & Par)	This parameter specifies the angle when the blinds are fully closed (when running down). When the blinds are closed, the slats close mechanically to the "completely closed" position. This is the angle in degrees that defines this completely closed position. The value is measured from the horizontal position. Numeric: 32 bit integer value -180 deg to +180 Deg NULL = Keep Angle
Slat Angle Fully Open (In & Par)	This parameter specifies the angle when the blinds are fully open (when running up). When the blinds are open, the slats open mechanically to the "completely open" position. This is the angle in degrees that defines this completely open position. The value is measured from the horizontal position. Numeric: 32 bit integer value -180 deg to +180 Deg NULL = Keep Angle
Slat Steps (In & Par)	This defines the number of steps for slat adjustment that the operator can reach by pushing the Open (up) and Close (down) buttons. The manual angle adjustment is carried out via short key presses. This setting divides the total angle, i.e. the angle between **Slat Time To Close Par** and **Slat Time To Open Par** into steps. One step is executed after each short press of the button. Example: Total angle = abs (- 80 - 80) = 160° divided by 5 steps = 32° per step. The operator can set the angles in 32° steps with 5 times button press. Numeric: 32 bit floating value 0 - 255.00, Null

Output

Table 288 Outputs of Blind A

Output Name	Description
Out Current Position	This output shows the current calculated position of the blind. Null is output if the position is unknown (After power up and if no reference drive has been completed). For a blind or shutter (**Position Logic** = Normal) 0 % means that the blind or shutter is at the very upper point, for a window (**Position Logic** = Inverse) 0 % means that the window is closed. Numeric: 32 - Bit integer value, 0 - 100 %, NULL = Not Used
Out Current Angle	The output shows the current calculated angle of the blind. Null is output if the angle is unknown (after power up and if no reference drive has been completed) or in case there is no slat control (**Device Type** = Shutter). 0° means slats are horizontal if Abs (Slat Angle Fully Closed) = Abs (Slat Angle Fully Open). Numeric: 32-Bit integer value, -180 - 180 %, NULL = Not Used
Out Movement Status	The output is for information about what the blind is currently doing. 1 = Stopped, 2 = Opening (blind/ Shutter moving Up, Window is opened) 3 = closing (Blind/ Shutter moving down, Window Closing)
Out Endposition	The output gives information about the end position of the blind as soon as the blind has stopped. 1 = No end position reached, 2 = Fully Open, 3 = Fully Closed, 4 = UnknownEndPos
Out Cause	The Out Cause provides different states for diagnosis to see at a glance what is currently happening in the FB. Since the Emergency and Application Logic is programmed by the user, the Cause inputs of Emergency and Application are looped through to Out Cause when the corresponding Cmd has been executed (and the blind is controlled). If the Cause inputs have the value null and an Emergency or **Application Cmd** was executed, then Out cause is set to the values “**Emergency Cmd**” or “**Application Cmd**”. These values are referenced more often • 001 = NoCause (i.e. after power up) • 002 = Emergency • 003 = WiredManualOvrdStop • 004 = WiredManualOvrdClose • 005 = WiredManualOvrdOpen • 006 = WiredManualOvrdAngleStepPos • 007 = WiredManualOvrdAngleStepNeg • 009 = ManualOvrdStop

Table 288 Outputs of Blind A (Continued)

Output Name	Description
	• 010 = ManualOvrdClose • 011 = ManualOvrdOpen • 012 = ManualOvrdAngleStepPos • 013 = ManualOvrdAngleStepNeg • 015 = Application (ApplicationCmd! = Null, Cause = Null) • 017 = ReferenceDriveRunning ***Note:*** *Reserved until 99.* All other values can be freely defined. However it is recommended to use these given causes for standardization. This also provides suggestions for implementation. • 151 = Wind Low • 152 = Wind High • 153 = Service • 154 = No Service • 155 = Fire • 156 = No Fire • 157 = Rain • 158 = No Rain • 159 = Ice 160 = No Ice

Parameters

Table 289 Parameter of Blind A

Parameter Name	Description
Device Type	This parameter enables to differ the behavior between a shutter and a window without having an angle and a blind having an angle. 1 = Shutter or electric driven window without angle, 2= Blind with angle
Position Logic (In & Par)	• "Off": The blind is disabled. This is usefull during maintenance. • "Normal": A position of 0% means the shutter or blind is fully up (open), while a position of 100% means fully down (closed). Downwards closing / slat downwards closed). • "Inverse": A position of 100% means the electric driven window is fully open, while a position of 0% means fully closed. Upwards opening / slat upwards closed)
Lock Time Par (In & Par)	Refer the input table as shown. Default = 0 min.

Table 289 Parameter of Blind A (Continued)

Parameter Name	Description
Slat Time To Close Par (In & Par)	Refer the input table as shown. Default = 0 sec.
Slat Time To Open Par (In & Par)	Refer the input table as shown. Default = 0 sec.
Slat Angle Fully Closed Par (In & Par)	Refer the input table as shown. Default = - 80º.
Slat Angle Fully Open Par (In & Par)	Refer the input table as shown. Default = + 80º.
Slat Steps Par (In & Par)	Refer the input table as shown. Default = 5.
Slat Alignment Par (In & Par)	Refer the input table as shown. Default = - 45º.
Push Button Close Terminal (In & Par)	This defines the terminal at which the button is wired for closing (Blind/shutter driving down, window is closing).
Push Button Open Terminal (In & Par)	This defines the terminal at which the button is wired for opening (Blind/shutter driving up, window is opening).
Time To Close Par (In & Par)	Refer the input table as shown.
Time To Open Par (In & Par)	Refer the input table as shown.
Start Up Threshold Par (In & Par)	Refer the input table as shown.

Table 289 Parameter of Blind A (Continued)

Parameter Name	Description
Reference Drive (In & Par)	After PowerUp, the blind model does not know where the blind is in reality. Therefore, a ReferenceDrive to 0 % is necessary, which means to the upper position for blind and shutter (**Position Logic** = Normal) and to the window closed position for the window (**Position Logic** = Inverse). After the reference drive, the upper blind position or the closed window position are assumed to be 0 % with 0º angle as the reference position. The reference drive lasts 1.5 x as long as the **Time To Open** for a blind and shutter (PositionLogic = “Normal“) or **Time To Close** for a window (PositionLogic = “Inverse“) time. Commands coming in via **Emergency Cmd** and **Application Cmd** during the **Reference Drive** are executed after the ReferenceDrive (according to the Priority Logic). Following settings are available: • **AfterPowerUp:** Immediately after powering up of the controller, a **Reference Drive** is performed (Recommended default). This first **Reference Drive** after Power up cannot be interrupted by a new command or the push buttons. Also an **Emergency Cmd** will not stop the first **Reference Drive** after power up. If this is required, then use the **Operating Mode Bits** and set the motor to power off state. • **RefAfterNewCmd:** There is no **Reference Drive** after power up. The shutter or blind remains in its current position and waits until the first command comes (from **Emergency Cmd**, **Manual Override Cmd**, Push Button or **Application Cmd**). When the first blind command is received, the reference drive is performed and then the blind command is executed.If the first command is a **Manual Override Cmd** or a push button, then the user has to wait until the **Reference Drive** is finished. This first **Reference Drive** after Power Up cannot be interrupted by a new command or the push buttons. Also, an **Emergency Cmd** will not stop the first **Reference Drive** after power up. If this is required, then use the **Operating Mode Bits** and set the motor to power off state. If the blind is in the reference position, i.e. in the upper position (0%, 0º) or the window is closed with 0%, the 24h counter remains stopped. As soon as the reference position is left, the 24h counter starts. See also the explanation of the Extended Drive.

Table 289 Parameter of Blind A (Continued)

Parameter Name	Description
	Once the 24 hour timer have elapsed, the **Reference Drive** starts with the next application command (The FB is waiting for an application command; with the next command, the reference drive is performed and then the command is executed). This usually prevents the blinds from being moved at night (making noise). The **Reference Drive** drive is never done if an **Emergency Cmd** is active. A running **Reference Drive** which is started by the 24hour timer is stopped if the push button is pressed or a manualOverrideCmd or an **Emergency Cmd** is set. Background: If, for example, for service purposes the blind via **Emergency Cmd** has been closed to 100% for several days, then no 24h reference drive may take place after 24 hours. Enum: 1=RefAfterNewCmd, 2=AfterPowerUp
Motor Close Terminal (In & Par)	This defines the terminal at which the motor is wired for closing (Blind, shutter driving Down, Window driving to close). When the direction of rotation is changed (Open <-> Close), there is a pause of >= 600 msec. If this time is not observed, the motor may be damaged. When the same direction of rotation is switched on again, there is a pause of 300 msec. The minimum switch-on time is defined as **Start Up Threshold**. 1 = RO1, 2 = RO1, 3 = RO1, 4 = RO1, 101 = TO1, 102 = TO1, 103 = TO1, 104 = TO1, 255 - Not Connected
Motor Open Terminal (In & Par)	This defines the terminal at which the motor is wired for opening (Blind, shutter driving Up, Window driving to open). When the direction of rotation is changed (Open <-> Close), there is a pause of >= 600 msec. If this time is not observed, the motor may be damaged. When the same direction of rotation is switched on again, there is a pause of 300 msec. The minimum switch-on time is defined as **Start Up Threshold**. 1 = RO1, 2 = RO1, 3 = RO1, 4 = RO1, 101 = TO1, 102 = TO1, 103 = TO1, 104 = TO1, 255 - Not Connected

Push Button Behavior

The configuration is intended to adapt the button operation to different needs. This may be necessary when replacing external systems. The operation of a roller shutter, a motor-driven window and slats can also be desired differently. Operation via one push button could come.

Table 290 Push Button Behavior for Blinds

Wired Push Button	Input Slot, Manual cmd	Motor for blinds * 1
Short press when the blind is not running *2.	Man, angle step positive Man angle step positive.	Adjust the slat angle for one step.
Long Press	Man Open - Man Close	Drive to fully open or fully closed position.

Table 291 Push Button Behavior for Shutter / Window

Wired Push Button	Input Slot, Manual cmd	Motor for shutter or window * 1
Short press when the shutter is not running * 2.	Man Open - Man Close	Drive to fully open or fully closed position.
Short press when the shutter is running * 1.	Immediate Stop at current position.	Immediate Stop at current position

1 Based on* ***Device Type*

**2 A running shutter of blind is stopped immediately at the current position by the “Stop”, “ManOpen”, “ManClose” command or by shortly pressing one of the buttons.*

Example

Emergency override for Rain application

1. Priority will be emergency.
2. Emergency Override cause is Rain and out cause is 154.
3. Action type is 04 which is ToPositionAndAngle.

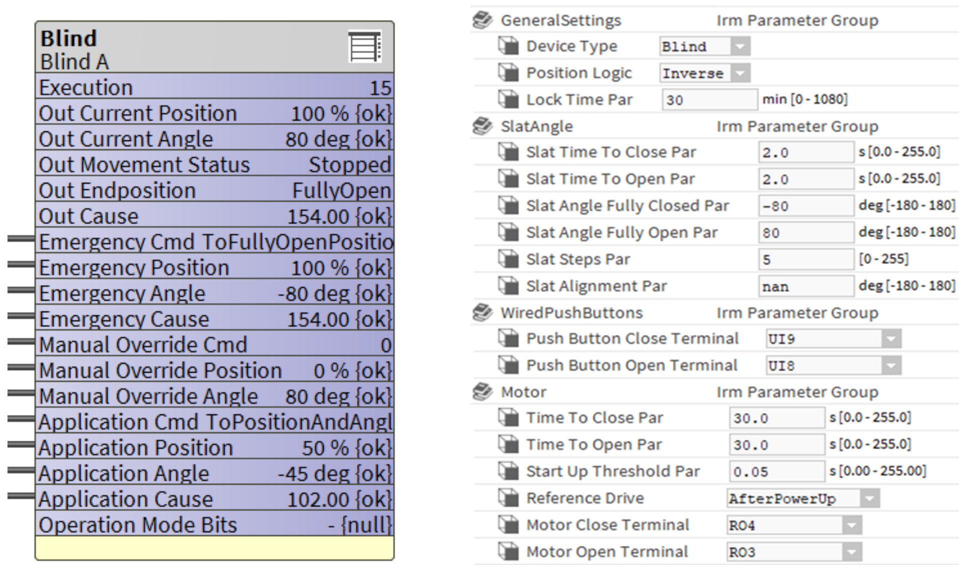


Fig. 466 Example - Emergency Override for Rain Application function block and Property Sheet

Manual override from operator command

1. Priority will be Manual Override.
2. Operator Manual Override Open and out cause is 11.
3. Action type is 02 which is Manual Open.
4. Lock time is 30 minutes. After 30 minutes. Manual command will be override by application command.

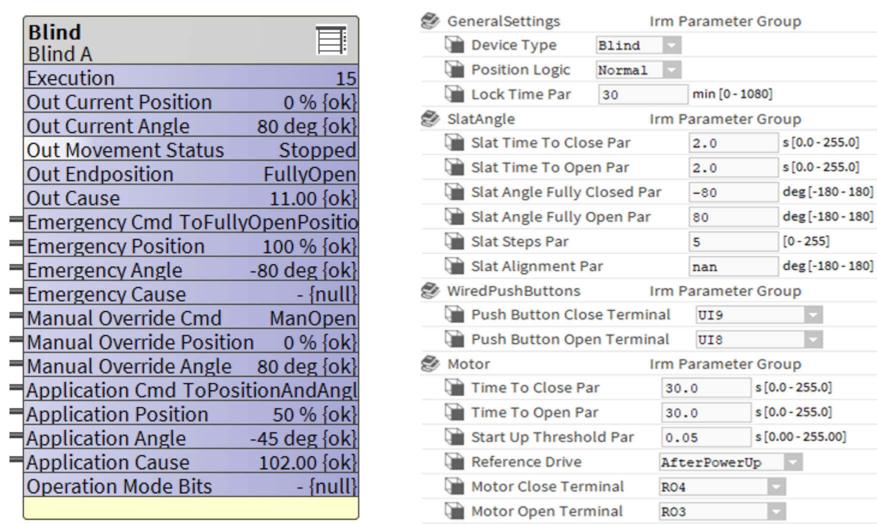


Fig. 467 Example - Manual Override from Operator Command Niagara function block and Property Sheet

Application for occupied Mode

1. Priority will be application override.
2. Operator application Override and out cause is 102.
3. Action type is 04 which is ToPositionAndAngle.

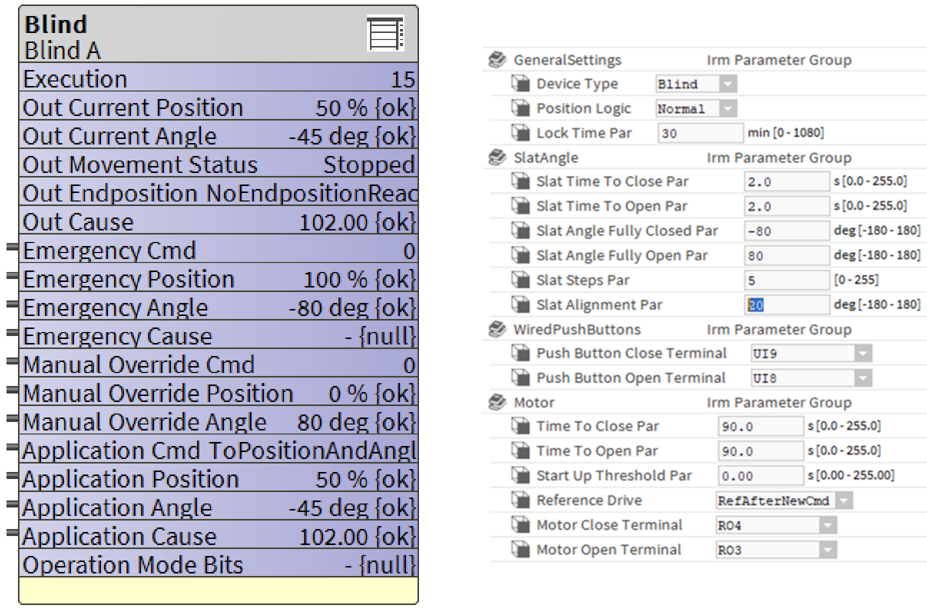


Fig. 468 Example – Application for Occupied Mode function block and Property Sheet

CHAPTER

18 WALL MODULE FUNCTION BLOCK

The honIrmControl Palette provides the following wall module function block that can be configured and used to build the required application logic that supports various wall operating devices.

- Wm Config Hvac A

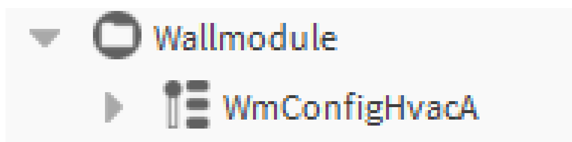


Fig. 469 Wall module Function Block

Wm Config Hvac A

The function block supports various wall modules like wired, Sylk TR40 or TR42, and an external wall modules. The external wall module communicates via BACnet to the room controller.

Note: *A wall module that communicates via BACnet or Modbus is called an external wall module. It can be a Honeywell wall module, such as the Modbus wall module TR80, or a third-party vendor wall module.*

The name **Config** means that the operator sequence is not freely programmable but configurable. **HVAC** means that only HVAC related things are supported.

The object provides the measured sensor values such as room temperature, humidity, and air quality. In addition, the desired setpoint and fan stage or level will be provided.

The object determines the effective occupancy mode, the setpoint mode (Off, Cooling, Heating) with the corresponding setpoint value, and the HVAC mode. The WmConfigHvacA function block is a fundamental component of a room application and should be used in every room, even if no wall modules are present.

The reason for determining these values in the function block is that the user can manipulate all these points via the wall modules. For example, the user can temporarily switch to Occupied Mode, or in the future, the operator could say about the wall module that he is the next 4 hours "Unoccupied," or he enters his vacation days into the wall module.

See the options below which the user can select. Depending on the selection of the wall module and other settings, the I/O slots and the parameters are shown or hidden accordingly.

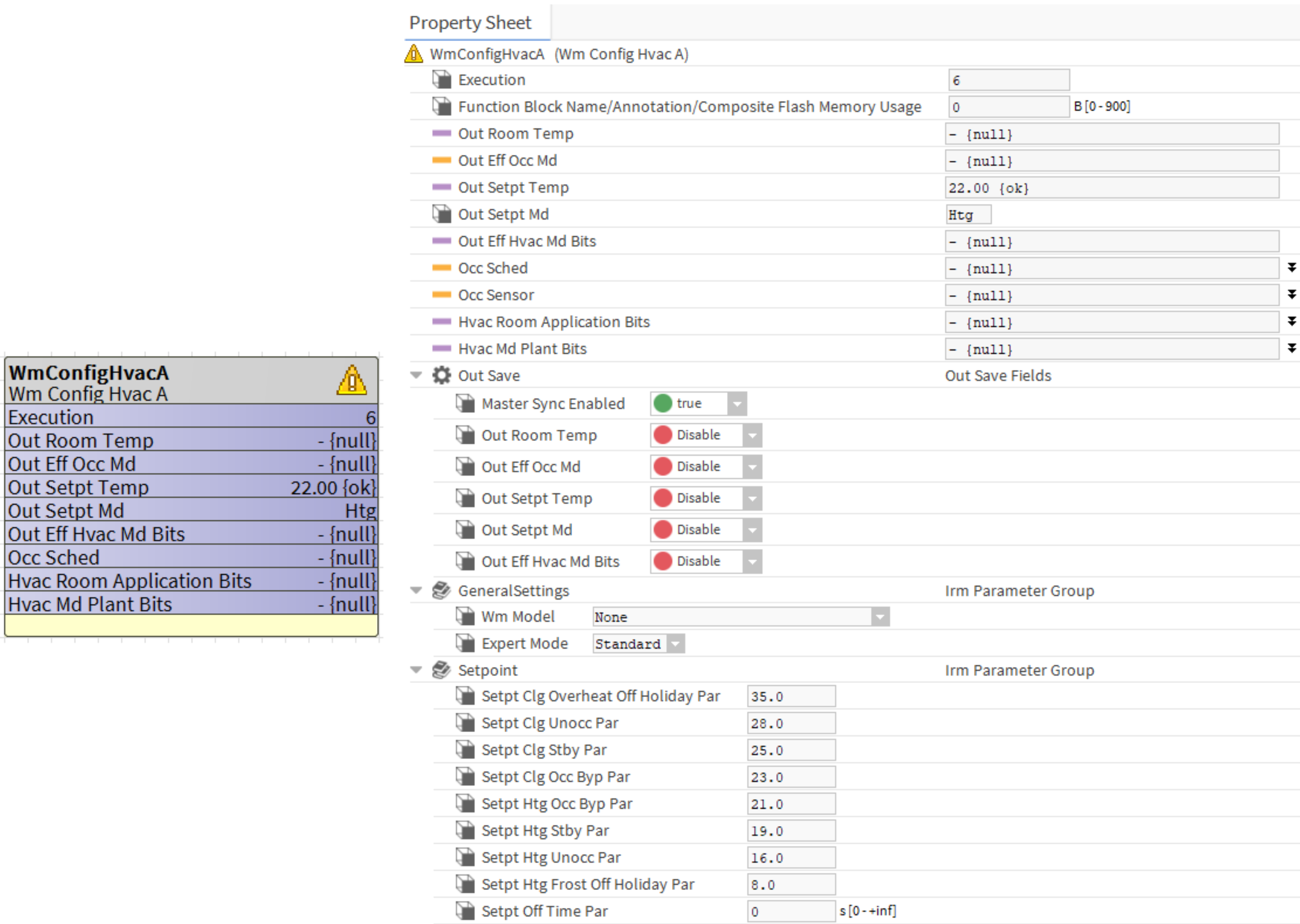


Fig. 470 WmConfigHvacA Function Block and Property Sheet

Once the object is on the wiresheet, the user should choose the wall module in **Wm Model** under **General Settings.**

Note: *After selecting the Wm Model, press the* ***Save*** *button to show or hide other parameters and I/O slots.*

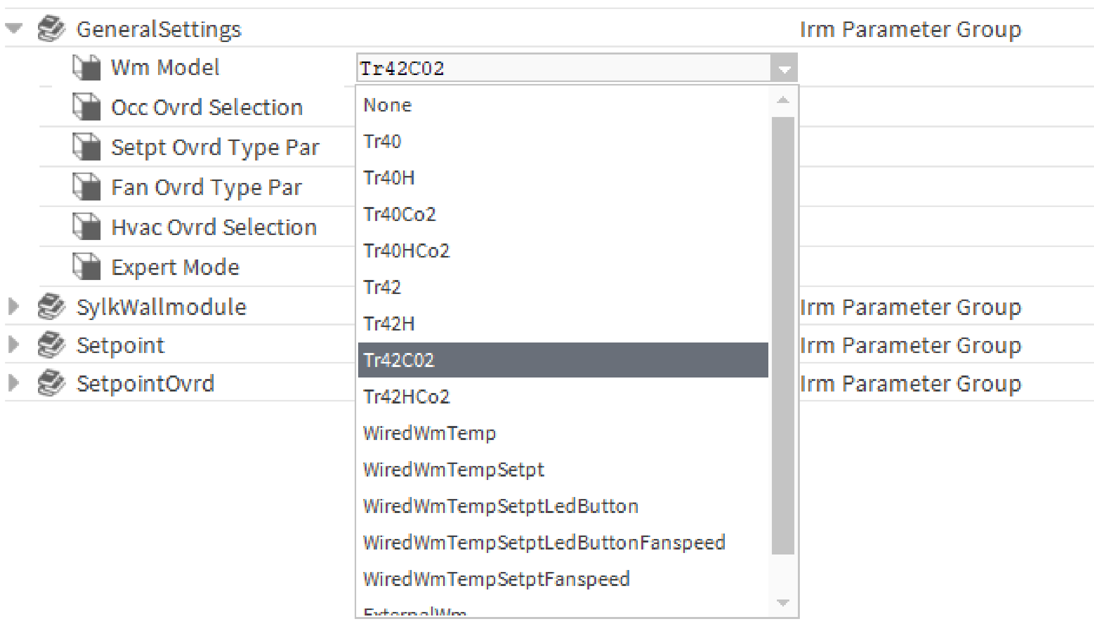


Fig. 471 WmConfigHvacA Property Sheet

Input

Table 292 Inputs of Wm Config Hvac A

Input Name	Description
Room Temp External Wm	The room temperature received by an external WM, for example, via BACnet or Modbus, is connected to this input. This input has higher priority than the input **Room Temp Wired Wm** and higher than the room temperature from the Sylk wall module. See **Sylk Sensor Arbitration Bits**, **Room Temp Calib Offs**, **Room Temp Calib Offs Par**, and **Out Room Temp**. **Range:** -58 °F to 302 °F (-50 °C to 150 °C)
Room Temp Wired Wm	The room temperature read by a conventional wired wall module or by a wired sensor via a UI Terminal function block is connected to this input. This input has lower priority than the input **Room Temp External Wm** but higher priority than the room temperature from the Sylk wall module. See **Sylk Sensor Arbitration Bits**, **Room Temp Calib Offs**, **Room Temp Calib Offs Par**, and **Out Room Temp**. **Range:** -58 °F to 302 °F (-50 °C to 150 °C)
Room Temp Calib Offs (In & Par)	The Calibration Offset is added to the room temperature, whether the room temperature comes from **Room Temp External Wm**, from **Room Temp Wired Wm**, or the Sylk wall module. The offset can be positive or negative. Thus, a measuring error can be corrected, for example, by an inappropriate positioning of the WM. **Range:** -58 °F to 302 °F (-50 °C to 150 °C)

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Room Temp Frost	If Room Temp Frost input is **true**, it means that the room is in frost protection condition. With Frost, the HVAC mode (cooling or heating) selected by the room user from the Sylk wall module or via **Hvac Md External Wm Bits** is ignored. **Out Eff Hvac Md Bits** is set if the room user has selected the HVAC mode to AUTO. The same happens with the fan speed; the fan stage/speed selected by the room user from the Sylk wall module or via **Fan Stg External Wm** or **Fan Variable Spd External Wm** is ignored, and **Out Fan Stage Ovrd** and **Out Variable Fan Speed Ovrd** provide AUTO. This ensures that the user can prevent frost protection via the wall module. As soon as the frost condition is finished, the initial HVAC mode and Fan speed selected by the room user will apply again as it was before the frost condition. Frost is also shown on a corresponding display (***Note****: Sylk-TR42 does not show Frost*). **Values:** **False:** No Room Frost **True:** Room Frost is active
Room Temp Overheat	If Room Temp Overheat is **true**, it means that the room is in overheat protection condition. With Overheat, the HVAC mode (cooling or heating) selected by the room user from the Sylk wall module or via **Hvac Md External Wm Bits** is ignored. **Out Eff Hvac Md Bits** is set like if the room user has selected the HVAC mode to AUTO. The same happens with the fan speed; the fan stage/speed selected by the room user from the Sylk wall module or via **Fan Stg External Wm** or **Fan Variable Spd External Wm** is ignored, and **Out Fan Stage Ovrd** and **Out Variable Fan Speed Ovrd** provide AUTO. This ensures that the user can prevent overheating protection via the wall module. As soon as the overheat condition is finished, the initial HVAC mode and fan speed selected by the room user will apply again as it was before the overheat condition. Overheat is also shown on a corresponding display (Note: Sylk-Tr42 does not show Overheat). **Values:** **False:** No Room Frost **True:** Room Frost is active

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Room Humidity External Wm	This input is linked to the room humidity, which is received by an external wall module via BACnet or Modbus. This input has a higher priority than the input **Room Humidity Wired Wm** and the room humidity from the Sylk wall module. See **Sylk Sensor Arbitration Bits**, **Room Humidity Calib Offs**, **Room Humidity Calib Offs Par**, and **Out Room Humidity**. **Range:** 0 - 100 %
Room Humidity Wired Wm	A conventional wired wall module or a wired sensor reads the room humidity via a UI Terminal function block connected to this input. This input is less important than the input **Room Humidity External Wm**, but it is more important than the room humidity from the Sylk wall module. See **Sylk Sensor Arbitration Bits**, **Room Humidity Calib Offs**, **Room Humidity Calib Offs Par**, and **Out Room Humidity**. **Range:** 0 - 100 %
Room Humidity Calib Offs (In & Par)	The Calibration Offset is added to the room humidity, whether the room humidity comes from **Room Humidity External Wm**, **Room Humidity Wired Wm**, or the Sylk wall module. The offset can be positive or negative. Thus, a measuring error can be corrected, for example, by an inappropriate positioning of the WM. **Range:** -100 to 100 %
Room Air Quality External Wm	The room quality received by an external WM via BACnet or Modbus is connected to this input. This input has higher priority than the **Room Air Quality Wired Wm** and higher than the room air quality from the Sylk wall module. See (see Parameter of Wm Config Hvac A - Sylk wall module on page 459) for **Sylk Sensor Arbitration Bits** and **Out Room Air Quality** details. **Range:** 0 - 5000 ppm CO2
Room Air Quality Wired Wm	The room air quality, which is read by a conventional wired wall module or by a wired sensor via a UI Terminal function block, is connected to this input. This input has lower priority than the input **Room Air Quality External Wm** but higher than the room air quality from the Sylk wall module. See (see Parameter of Wm Config Hvac A - Sylk wall module on page 459) for **Sylk Sensor Arbitration Bits**, **Room Air,** and **Out Room Air Quality** details. **Range:** 0 - 5000 ppm CO2

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Occ Sched	This input connects the base occupancy mode coming from a scheduler or time program. Occ Sched defines the occupancy modes **Occupied**, **Standby**, and **Unoccupied** at different times. Typically, the time program is received via BACnet. **Example:** Occupied from 6 am – 7 pm (6:00 - 19:00) Unoccupied from 7:01 pm – 5:59 am (19:01 – 5:59) Of course, the base occupancy mode Occ Sched can be overridden by other conditions. Refer to: • **Occ Sensor** Input • **Occ Md Cmd External Wm** Input • **Occ Ovrd Type Bits** Input or **Occ Ovrd Type Bits Par** defining the occupancy overwrite via the wired or Sylk wall module • **Occ Md Prio** Input ignoring OccSched, OccSensor and Occ Ovrd from wall module The result of all Occupancy Modes are the outputs: • **Out Eff Out Occ Md** showing the effective occupancy mode • **Out Occ Md Ovrd Md** showing an occupancy override • **Out Occ Md Ovrd Rem Time** showing an occupancy override timer **Enums:** **2:** Occupied: Room is occupied. Use comfort setpoint **5:** Unoccupied: The Room is unoccupied (Night, Weekend). The room temperature is reduced. **6:** Standby: The room is not occupied right now. The setpoint is not so convenient. **null:** Same as Occupied
Occ Sensor	The Occ Sensor can change the occupancy mode of the time program. If the time program says OCCUPIED for a room in which no movement is detected (employee is at lunch break or has already gone home), the effective occupancy mode switches to STANDBY to save energy. The behavior of the occupancy sensor is configured via the parameter **Occ Sensor Behav Bits**. **Enums:** **1:** No Occupancy Sensor available. Ignore this Input. **2:** Occupancy Sensor is Occupied. Movement detected. **5:** Occupancy Sensor is Unoccupied. No movement is detected. **null:** Ignore this Input.

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Occ Md Cmd External Wm	The occupancy mode determined by Occ Sched and Occ Sensor can be changed at any time by using the wall module (External BACnet or Modbus wall module, wired, or Sylk wall module). • If the room is used differently than specified in the time program and no occupancy sensor is installed, or • If the occupancy sensor is configured so that it does not change the effective occupancy mode, this is required. In this case, we're referring to the concept of "Occupancy Override." **Example**: Someone comes to the office to work at the weekend. In addition to the effective occupancy output **Out Eff Occ Md**, an Out Occ Md Ovrd shows the occupancy override and output **Out Occ Md Ovrd Rem Time** showing the remaining time of an occupancy Override. **Occ Md Cmd External Wm** typically comes as an occupancy overwrite command from an external wall module via BACnet or Modbus. Instead of selecting the overwrite via the push button from the wired wall module or via the Sylk wall module, the external wall module sends the overwrite command. Thus, the time program **Occ Sched** can be overwritten by the operator of the external WM. **Occ Md Cmd External Wm** is handled as an event, that is the command is only executed if there is a change, like pressing the push button on the wired wall module or selecting the "Overwrite" command on the Sylk wall module. **null** = Not used The firmware performs a Last-Wins between the conventional wall module, the Sylk wall module, and **Occ Md Cmd External Wm**. **Example:** During **Occ Sched** = Unoccupied, the push button of the wired wall module is pressed. **Out Eff Occ Md** changes to BYPASS, and the bypass timer runs for a temporary occupancy override (seen by **Out Occ Md Ovrd Rem Time**). Then **Occ Md Cmd External Wm** is set to UNOCCUPIED. **Out Eff Occ Md** changes to UNOCCUPIED, stopping the bypass timer. The occupancy overwrite can be canceled from the wall module by the user. With the wired wall module the button is pressed; with the Sylk wall module, the canceling is done over the menu, and with the input **Occ Md Cmd External Wm**, the canceling works. With the value selection "ResetOccMdCmd." The occupancy overwrite can also be done automatically, depending on Occ Ovrd Behaviour or if the **Out Occ Md Ovrd Rem Time** is expired (that is bypass timer is expired).

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
	Occ Md Cmd External Wm is not checked against the parameter **Occ Ovrd Type Bits** / **Occ Ovrd Type Bits Par**, which means every command is valid (like other external Wm Inputs). **Enums:** **1:** Not used **2:** Override to Occupied **4:** Override to Holiday (Saved into Flash, do not periodically write!) **5:** Override to Unocc **6:** Override to Stby **7:** Override to Byp **8:** Reset Occupancy Override (Effective occupancy logic determined by **Occ Sched** and **Occ Sensor**)
Occ Ovrd Type Bits (In & Par)	The occupancy mode determined from **Occ Sched** and **Occ Sensor** can be changed at any time via the wall module (External BACnet/Modbus, wired, or the Sylk wall module). The Occ Ovrd Type Bits parameter defines which occupancy overrides from wired and Sylk wall modules should be possible for this application. If the wall module user does not change the occupancy mode, enter zero here. Otherwise, add the bit values of the occupancy modes which are to be selected by the operator. The supported occupancy override modes depend on the wall module. See **Wm Model**. Bitvalues for **Wm Model** Wired WM with LED Button. **0:** NoManOvrd **2:** Holiday **4:** Unocc **32:** Bypass until event **Occ Sched** = Occ, see also **Occ Ovrd Time** **null:** NoManOvrd. Bitvalues for **Wm Model** Tr42. **0:** NoManOvrd **32:** Bypass, see also **Occ Ovrd Time** **null:** NoManOvrd. ***Note:*** *This input or parameter is not relevant for the external wall module coming via* ***Occ Md Cmd External Wm****.* ***Note:*** *The overwrite menu allowing “Bypass” is displayed on the Sylk-WM display only if the “Bypass” bit is set AND if the* ***Occ Ovrd Time*** *> 0. A running bypass override is stopped if the* ***Occ Ovrd Type Bits*** *are not set off the* ***Occ Ovrd Time*** *= 0.*

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Occ Ovrd Time (In & Par)	If no occupancy sensor is installed in the room or if the occupancy sensor does not switch from the UNOCCUPIED to the OCCUPIED mode, the wall module user can select the occupancy override mode BYPASS. The user can select an occupancy override to BYPASS from the external wall module, the wired, and the Sylk wall module. For the wired and the Sylk wall module, **Occ Ovrd Type Bits** need to be set accordingly. The occupancy mode BYPASS uses the same setpoint as the occupied mode, but only for a certain period. This duration is entered here over **Occ Ovrd Time**. When the time has elapsed, the occupancy mode returns to the value determined by **Occ Sched** and **Occ Sensor**. The BYPASS occupancy mode can also be terminated earlier by the user via the wall module. **Example:** The occupancy mode on the weekend is Unoccupied. Suppose you come to the office and choose the mode BYPASS as occupancy override at the wall module. The BYPASS mode is activated for an **Occ Ovrd Time** of 480 min. After 300 minutes, if you leave the office and forget to reset the BYPASS mode on the wall module. After 480 min, the occupancy override mode BYPASS is terminated, and the effective occupancy mode changes back from BYPASS to UNOCCUPIED. **Range:** 0 – 1440 min, "null" means 0 min
Occ Md Prio	The occupancy mode on this input bypasses the entire Occupancy Logic, consisting of **Occ Sched**, **Occ Sensor**, **Occ Md Cmd External Wm**, **Occ Ovrd Type Bits**, and **Occ Ovrd Time**. This also means that the occupancy sensor is no longer evaluated and that the occupancy mode can no longer be set or corrected by the wall module. The entire occupancy logic takes place outside the function block, and the result is given into the function block via this input. **Enums:** **1:** Not used. Use internal occupancy logic. **2: Out Eff Occ Md** Occupied as effective Occupancy Mode **3: Out Eff Occ Md** Off as effective Occupancy Mode **4: Out Eff Occ Md** Holiday as effective Occupancy Mode **5: Out Eff Occ Md** Unoccupied as effective Occupancy Mode **6: Out Eff Occ Md** Standby as effective Occupancy Mode **null:** Not used. Use internal occupancy logic.

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Setpt Clg Overheat Off Holiday (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. There is a cooling and heating setpoint for each occupancy mode (OCCUPIED, STANDBY, UNOCCUPIED, OFF / HOLIDAY). See the separate setpoint description below. This input or parameter represents the room temperature cooling setpoint when the room control is switched off (such as Fanspeed overwrite to OFF from wall module), or the occupancy mode is in holiday mode (such as Occupancy overwrite to HOLIDAY from wall module). In this case, the output **Out Eff Occ Md** is set to OFF or HOLIDAY. The setpoint is higher than the **Setpt Clg Unocc** setpoint to save as much energy as possible while the room is not used for a longer time. The setpoint is activated when a cooling application is installed (See input **HVAC Room Application Bits**), and a cooling medium is present (See input **Hvac Md Plant Bits**), and the room user enabled cooling via the wall module (See input **Hvac Md External Wm Bits** and Hvac Md Internal Wm Bits). In this case, the output Eff Hvac Md Bits represents a cooling HVAC mode. If this setpoint is active, then the output **Out Setpt Md** shows cooling. Irrespective of this setpoint, the **Room Temp Overheat** input protects the building from overheating damage by simply overwriting the cooling and heating outputs independent of the HVAC, setpoint, and PID control. **Range:** -58 °F to 302 °F (-50 °C to 150 °C)
Setpt Clg Unocc (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. This input or parameter represents the room temperature cooling setpoint when the effective Occupancy mode is UNOCCUPIED. In this case, the output **Out Eff Occ Md** is set to UNOCCUPIED. **Range:** -58 °F to 302 °F (-50 °C to 150 °C)
Setpt Clg Stby (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. This input or parameter represents the room temperature cooling setpoint when the effective Occupancy mode is STANDBY. In this case, the output **Out Eff Occ Md** is set to STANDBY. **Range:** -58 °F to 302 °F (-50 °C to 150 °C)

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Setpt Clg Occ Byp (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. This input or parameter represents the room temperature cooling setpoint when the effective Occupancy mode is OCCUPIED or BYPASS (temporary Occupied). In this case, the output **Out Eff Occ Md** is set to OCCUPIED or BYPASS. **Range:** -58 °F to 302 °F (-50 °C to 150 °C)
Setpt Htg Occ Byp (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. This input or parameter represents the room temperature heating setpoint when the effective Occupancy mode is OCCUPIED or BYPASS (temporary Occupied). In this case, the output **Out Eff Occ Md** is set to OCCUPIED or BYPASS. **Range:** -58 °F to 302 °F (-50 °C to 150 °C)
Setpt Htg Stby (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. This input or parameter represents the room temperature heating setpoint when the effective Occupancy mode is STANDBY. In this case, the output **Out Eff Occ Md** is set to STANDBY. **Range:** -58 °F to 302 °F (-50 °C to 150 °C)
Setpt Htg Unocc (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. This input or parameter represents the room temperature heating setpoint when the effective Occupancy mode is UNOCCUPIED. In this case, the output **Out Eff Occ Md** is set to UNOCCUPIED. **Range:** -58 °F to 302 °F (-50 °C to 150 °C)
Setpt Htg Frost Off Holiday (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. This input or parameter represents the room temperature heating setpoint when the room control is switched off (such as Fanspeed overwrite to Off from wall module), or the occupancy mode is in HOLIDAY mode (such as Occupancy overwrite to HOLIDAY from wall module). In this case, the output **Out Eff Occ Md** is set to OFF or HOLIDAY. **Range:** -58 °F to 302 °F (-50 °C to 150 °C)

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Setpt Off Time (In & Par)	When a change is determined from a cooling setpoint to a heating setpoint or vice versa, shown on **Out Setpt Md**, the setpoint mode always first changes to the OFF state for the **Setpt Off Time** before the new setpoint mode is set. 0 or "null" deactivates the Off Mode. This ensures that a heating and cooling valve is not opened simultaneously. **Example:** If there is a setpoint mode change from cooling to heating when the cooling valve is fully open, for example, because the setpoint on the wall module has been changed, the cooling valve will close while the heating valve opens simultaneously. With a 4-pipe system, the cold water return flow could be increased depending on the position of the registers; with a 4-pipe system using a common register for cooling and heating (that is ceiling), even mixing between cold and warm water would take place, since both valves are open at the same time. During the Off-Mode, the cooling valve closes and remains closed until the **Setpt Off Time** has elapsed. **Range:** 0 - 28800 sec
Setpt Shift Demand Limit	This input can be used to increase or decrease the setpoint and the basic setpoint or the setpoint selected by the operator to save more energy. This is typically used to optimize the electrical load profile. Demand limiting is a control strategy designed to prevent the demand of electrical energy (measured as a quarter-hour power output in MW or KW) of a facility from going over a predetermined threshold by altering the heating and cooling setpoints. The application could vary the shift value depending on the Heat/Cool Mode or Occupancy Mode. **Range:** 0 - 45 °F delta (0 - 25 °C delta)**,** **null:** No offset
Setpt Ovrd Type (In & Par)	The input or parameter defines whether the operator obtains a relative setpoint or an absolute setpoint selection via the wall module. At the same time, the setpoint selection can also be deactivated. Depending on the selected **Setpt Ovrd Type**, the limits **Min Clg Setpt Selection**, **Max Clg Setpt Selection**, **Min Htg Setpt Selection**, and Max Htg Setpt Selection must be adapted. **Enum:** **1:** No Setpoint Override **2:** Relative **3:** Absolute **null:** No Setpoint Override

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Setpt Adjust External Wm	The value serves as the base setpoint of an external wall module. The value is evaluated either as a relative offset or as an absolute setpoint, depending on **Setpt Ovrd Type**. The value is not limited to the **Min Clg Setpt Selection**-**Max Clg Setpt Selection** range, **Min Htg Setpt Selection**-Max Htg Setpt Selection. If SetptOvrdType is set to NoSetptOvrd, then this input is ignored. This input has higher priority than **Setpt Adjust Wired Wm** and higher priority than the setpoint from the Sylk wall module. The value cannot be reset via **Reset Wm** to a default value. **Range:** -58 °F to 302 °F or -90 to 270 °F delta (-50 °C to 150 °C or °C delta)
Setpt Adjust Wired Wm (In & Par)	The value serves as the base setpoint of a wired wall module. The value is evaluated either as a relative offset or as an absolute setpoint, depending on **Setpt Ovrd Type**. Typically, the wired setpoint is read out by a connected potentiometer via a UI terminal function block and then converted into a relative or absolute value. A LinearGraph FB with the following values are used for the conversion from the UI Terminal using the Setpt10KCharacteristic: **Rel °C:** X1=95.74; X2=14.26; Y1=-5; Y2=5 **Abs °C:** X1=99.58; X2=11.5; Y1=12; Y2=30 **Rel °F:** X1=95.77; X2=14.23; Y1=-9; Y2=9 **Abs °F:** X1=95.77; X2=14.22; Y1=55; Y2=85 The value is limited to the range of **Min Clg Setpt Selection**- **Max Clg Setpt Selection**, **Min Htg Setpt Selection**- Max Htg Setpt Selection. If SetptOvrdType is set to NoSetptOvrd, then this input is ignored. The value cannot be reset via **Reset Wm** to a default value because the wired WM evaluates a potentiometer position. **Range:** -58 °F to 302 °F or -90 °F delta to 90 °F delta (-50 °C to 150 °C or -50 °C delta to 50 °C delta).

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Min Clg Setpt Selection (In & Par)	If the user selects the setpoint of the wired WM **Setpt Adjust Wired Wm**, the Sylk wall module (not the value of the external wall module **Setpt Adjust External Wm**) is limited by this input or parameter. The limitation can be done via the application depending on for example, the effective Occupancy mode **Out Eff Occ Md**. **null** means that the setpoint of the wall module cannot be changed in the cooling mode (**Out Setpt Md** is Cooling). Typical Values: • If **Setpt Ovrd Type** is Relative: -9 °F delta (-5 °C delta) • If **Setpt Ovrd Type** is Absolute: 53.6 °F (12 °C) • If a different engineering unit is displayed on the wall module than is specified in the engineering tool Control Manager, then this range is also converted; this means this input or parameter does not need to be recalculated in the wiresheet.
Max Clg Setpt Selection (In & Par)	If the user selects the setpoint of the wired WM **Setpt Adjust Wired Wm**, the Sylk wall module (not the value of the external wall module **Setpt Adjust External Wm**) is limited by this input or parameter. The limitation can be done via the application depending on for example, the effective Occupancy mode **Out Eff Occ Md**. "null" means that the setpoint of the wall module cannot be changed in the cooling mode (**Out Setpt Md** is Cooling). Typical Values: • If **Setpt Ovrd Type** is Relative: 9 °F delta (5 °C delta) • If **Setpt Ovrd Type** is Absolute: 86 °F (30 °C) • If a different engineering unit is displayed on the wall module than is specified in the engineering tool Control Manager, then this range is also converted; this means this input or parameter does not need to be recalculated in the wiresheet. • If **Max Clg Setpt Selection** < **Min Clg Setpt Selection**, then **Max Clg Setpt Selection** = **Min Clg Setpt Selection**.

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Min Htg Setpt Selection (In & Par)	If the user selects the setpoint of the wired WM **Setpt Adjust Wired Wm**, the Sylk wall module (not the value of the external wall module **Setpt Adjust External Wm**) is limited by this input or parameter. The limitation can be done via the application depending on the effective Occupancy mode **Out Eff Occ Md**. "null" means that the setpoint of the wall module cannot be changed in the heating mode (**Out Setpt Md** is Heating). Typical Values: • If **Setpt Ovrd Type** is Relative: -9 °F delta (-5 °C delta) • If **Setpt Ovrd Type** is Absolute: 53.6 °F (12 °C) • If a different engineering unit is displayed on the wall module than is specified in the engineering tool Control Manager, then this range is also converted; this means this input or parameter does not need to be recalculated in the wiresheet.
Max Htg Sept Selection (In & Par)	If the user selects the setpoint of the wired WM **Setpt Adjust Wired Wm**, the Sylk wall module (not the value of the external wall module **Setpt Adjust External Wm**) is limited by this input or parameter. The limitation can be done via the application depending on the effective Occupancy mode **Out Eff Occ Md**. **null** means that the setpoint of the wall module cannot be changed in the heating mode (**Out Setpt Md** is Heating). Typical Values: • If **Setpt Ovrd Type** is Relative: 9 °F delta (5 °C delta) • If **Setpt Ovrd Type** is Absolute: 86 °F (30 °C) • If a different engineering unit is displayed on the wall module than is specified in the engineering tool Control Manager, this range is also converted; this means this input or parameter does not need to be recalculated in the Wiresheet. • If Max Htg Setpt Selection < **Min Htg Setpt Selection**, then Max Htg Setpt Selection = **Min Htg Setpt Selection**.

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Setpt Prio	Via **Setpt Prio**, an absolute setpoint can be temporarily specified by the building management system. Use the input **Setpt Adjust External Wm** if you have an external wall module. The value is always absolute, even if the wall module is configured for a relative setpoint via **Setpt Ovrd Type**. **Setpt Prio** has a higher priority than the setpoint of the wall module. **Setpt Prio** is not displayed on the wall module. This value is used to determine the setpoint mode **Out Setpt Md** with the room temperature. The effective setpoint is +- 1/2 * ZEB (Difference between the configured cooling and heating setpoint). See also **Out Cause**. **Range:** -58 °F to 302 °F (-50 °C to 150 °C)
Fan Ovrd Type (In & Par)	The function block supports a 1 – 3 speed fan and a variable speed fan. Normally the application determines the fan speed or position. If the room user does not like that fan speed, he can override it via the wall module. This input or parameter **Fan Ovrd Type** is used to configure the desired behavior of the fanspeed selection from the wall module. The programmer uses this to determine which the operator can select fan stages or speed from the wall module. The Staged Fan uses the output **Out Fan Stage Ovrd**. The variable speed fan the output **Out Variable Fan Speed Ovrd**. **Fan Ovrd Type** is not valid for an external wall module (BACnet, Modbus) coming via **Fan Stg External Wm** and **Fan Variable Spd External Wm**. "No Selection" means that the user has no possibility to change the fanspeed. In this case, the output for the staged fan is **Out Fan Stage Ovrd** = Auto, and the output for the variable speed fan is **Out Variable Fan Speed Ovrd** = Null (Auto).

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
	Enums: **1:** No Selection: No fanspeed selection possible from wall module **2:** Off / Auto (*3): User can select between Off and Auto **3:** Off / Auto / On: User can select between Off, Auto and On **4:** Off / Auto / 1 / 2 (*1): User can select between Off, Auto, 1 and 2 **5:** Off / Auto / 1 / 2 / 3 (*1): User can select between Off, Auto, 1, 2 and 3 **6:** Off / Auto / 0-100 % (*2) (*3): User can select between Auto and 0-100 % **Null:** No fanspeed selection possible from wall module (*1) This configuration is not supported for a variable speed fan. The result **Out Variable Fan Speed Ovrd** is always **null**. (*2) Configuration is not supported for a Staged Fan. The result is always **Out Fan Stage Ovrd** = Auto. (*3) Not supported by TR42. Below is a table showing the detailed outputs of **Out Variable Fan Speed Ovrd** and **Out Fan Stage Ovrd** depending on Fan Ovrd Type. See the table "Detailed **Out Fan Stage Ovrd** / **Out Variable Fan Speed Ovrd** depending on Fan Ovrd Type." The fan selection can be reset to AUTO at any time, no matter where the selection came from (either from Sylk or from the external WM; not the wired fanspeed selection because there is a switch available). The reset to AUTO is carried out via • **Fan Stg External Wm:** Ignore or **null** • **Fan Variable Spd External Wm: null** • **Reset Wm:** FanSel or All The logic that applies in **Fan Ovrd Behave**.

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Fan Stg External Wm	Usually, the application determines the fan speed or position. If the operator does not like the fan speed, he can override it via the wall module. If the fan selection is made from an external wall module, there are two inputs available: • **Fan Stg External Wm** for usage with a 1 – 3 staged fan. • **Fan Variable Spd External Wm** for usage with a variable speed fan. This input works independently from Fan Ovrd Type. The fan stage selection from an external wall module is transferred into the function block via this input. The fan selection can be reset to AUTO at any time, no matter where the selection came from (either from Sylk or the external WM; not the wired fanspeed selection because there is a switch available). The reset to AUTO is carried out via **Fan Stg External Wm:** Ignore or **null** **Fan Variable Spd External Wm: null** **Reset Wm:** FanSel or All The logic that applies in Fan Ovrd Behave. **Enums:** **1:** Ignore **2:** Auto **3:** Off **4:** Speed 1 (On) **5:** Speed 2 **6:** Speed 3 **null:** Ignore
Fan Variable Spd External Wm	Usually, the application determines the fan speed or position. If the operator does not like the fan speed, he can override it via the wall module. If the fan selection is made from an external wall module, there are two inputs available: • **Fan Stg External Wm** for usage with a 1 – 3 -staged fan • **Fan Variable Spd External Wm** for usage with a variable speed fan. This input works independently from Fan Ovrd Type. The fan speed selection from an external wall module is transferred into the function block via this input.

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
	The fan selection can be reset to AUTO at any time, no matter where the selection came from (either from Sylk or from the external WM; not the wired fanspeed selection because there is a switch available). The reset to AUTO is carried out via • **Fan Stg External Wm:** Ignore or **null** • **Fan Variable Spd External Wm: null** • **Reset Wm:** FanSel or All The logic that applies in Fan Ovrd Behav. • **Range:** 0-100 % • **null:** Auto
HVAC Room Application Bits	The **HVAC Room Application Bits** describe the heating and cooling system installed in the room, for example, that a water cooling coil, a water heating coil, and an electric heater are present. With a 2-pipe changeover system, the application sets the bits at runtime according to the medium available. The values are simply added according to availability. Null = 65535. Bitvalues for **Cooling** **1:** Water Clg **2:** Electric Clg **4:** Clg Fan Only **8:** Clg with Pretreated air Bitvalues for **Heating** **256:** Water Htg **512:** Electric Htg **1024:** Htg Fan Only **2048:** Htg Pretreated air **Example 1:** Installed is a 4-pipe system with water for cooling and water for heating **HVAC Room Application Bits** = 1 + 256 = 257. **Example 2:** Installed is a 2-pipe changeover system with water for cooling and water for heating and with electric heating **HVAC Room Application Bits** = 512. Add either 1 or 256 depending on the water, medium cold, or hot water availability with small logic.

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Hvac Md Plant Bits	The room controller receives information from the plant controller via BACnet on which medium is currently available or which part of the system is to be operated. In summer only cooling should be possible, in winter only heating and in the transition period both. The enum of BACnet must be converted into the bits by the application using the Numeric Select function block. Since the bits distinguish between water and electrical energy, it is possible to define that. For example, now only electrical heating is enabled (for example, for cost reasons). Bitvalues for **Cooling** (Low Byte) **1:** Water Clg **2:** Electric Clg **4:** Clg Fan Only **8:** Clg with Pretreated air Bitvalues for **Heating** (High Byte) **256:** Water Htg **512:** Electric Htg **1024:** Htg Fan Only **2048:** Htg Pretreated air Simple approach • Use 255 to enable all Cooling Outputs for opening • Use 65280 to enable all Heating Outputs for opening • Use 65535 to enable all Outputs for opening **Example:** • Only cooling is available or allowed: 1 + 2 + 4 + 8 = 15 (or 255) • Only electric cooling is available or allowed: 2 • Water Cooling and electric heating is possible or allowed: 1 + 512 = 513 • Ony Heating is available or allowed: 256 + 512 + 1024 + 2048 = 3840 (or 65280)

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Hvac Md External Wm Bits	In principle, a room user can select from a wall module which cooling or heating is to be used. From the TR42 wall module, you can currently choose between cooling or heating or Auto, while all options are available from the external wall module. From the wired-wall module, there is no possibility of cooling or heating selection. Bitvalues **Cooling** (Low Byte) **1:** Water Clg **2:** Electric Clg **4:** Clg Fan Only **8:** Clg with Pretreated air Bitvalues for **Heating** (High Byte) **256:** Water Htg **512:** Electric Htg **1024:** Htg Fan Only **2048:** Htg Pretreated air Simple approach • Use 255 to enable all Cooling Outputs for opening • Use 65280 to enable all Heating Outputs for opening • Use 65535 to enable all Outputs for opening **Examples:** • User wants to select on the external wall module that only Cooling may work now: 1 + 2 + 4 + 8 = 15 (or 255) • User wants to select on the external wall module that only Electric Cooling may work now: 2 • User selects on external wall module that only Water Clg and Electric Htg may work now: 1 + 512 = 513 • User wants to select on external wall module that only Heating may work now: 256 + 512 + 1024 + 2048 = 3840 (or 65280) Typically, the command from the external wall module is sent over BACnet and then converted in these values and then connected to the wall module input **Hvac Md External Wm Bits**.
Hvac Md Internal Wm Config Bits (In & Par)	Keep the number on 65535. Currently, this input is not supported by the TR42-wall module and not by the wired wall module.

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Reset Wm	With a wall module, the room user can make many settings, for example, change the setpoint, switch the fan to a certain speed, select another occupancy mode, or change the Cooling or Heating mode. Often these settings make no sense for the next day, and they should be reset for the next day. Also, it makes sense to reset these settings in a hotel if a new guest checks in. There are two methods to perform the reset. Either over the Input **Reset Wm** which is coming over BACnet or from a self-programmed logic or by a logic predefined in the wall module function block, see Setpt Ord Behav Bits, **Fan Ovrd Behav Bits**, **Occ Ovrd Behav Bits**, and **Hvac Ovrd Behav Bits**. The bit values inform the function block what is to be reset. Only the individual bit values must be added. The reset occurs as soon as the input **Reset Wm** changes from 0 to another value. With Bit32, the wall module is done powerless for 30sec; then, the wall module is reloaded with the proxy file. This can be done every midnight, for example, if the wall module has a problem. During that time, the output slots remain on the current value. ***Note:*** *Via this input, it is possible to reset all room user settings even after power-up by writing a value! = null to the input in the first DDC cycle.* **Bit-Values:** **0:** No Reset **1:** Reset Setpoint (Relative and Absolute) **2:** Reset Fan Selection **4:** Reset Occupancy Override Selection, but not a Holiday mode **8:** Reset HVAC Selection (Cooling, Heating etc) **16:** Reset Occupancy Override Selection to Holiday Mode **32:** Reload the WM **255:** Reset All **Example:** To Reset the Setpoint and the Fan Selection **Reset Wm** = 1 + 2 = 3

Table 292 Inputs of Wm Config Hvac A (Continued)

Input Name	Description
Screen Layout (In & Par)	**Screen Layout** is used to select the appearance of the display. For the TR42 wall module, user can choose between the English text and standardized symbols. **Enums:** **1:** Show standardized symbols **2:** Show English text **3:** Symbols and Degree F (*1) **4:** Symbols and Degree C (*1) **5:** English text and Degree F (*1) **6:** English Text with Degree C (*1) (*1) The value on the display will be either F or C, as desired. The input and parameter values remain in the unit defined in the engineering tool.
Home Screen (In & Par)	The TR42 wall module displays a home screen after power- up or a few seconds after an operation. A value can be displayed in large font on the home screen, which can be selected here. **Enums:** **1:** Black **2:** Scroll all sensor values **3:** Room Temp **4:** Setpoint **5:** Humidity **6:** Air Quality ***Note:*** *The power supply of the Spyder Model 5 or Spyder Model 7 controller needs to be AC. Otherwise, the Humidity and Air Quality cannot be measured and not shown.*

Table 293 Details of Out Fan Stage Ovrd and Out Variable Fan Speed Ovrd

Fan Ovrd Type	Out Fan Stage Ovrd wall module selection = Off / Auto / 1 / 2 / 3	Out Variable Fan Speed Ovrd wall module selection = Off / Auto / 1 / 2 / 3
No Selection or Null	A / A / A / A / A	Null / Null / Null / Null / Null
Off / Auto	Off / A / A / A / A	0 % / Null
Off / Auto / On	Off / A / 1/ 1 / 1	0 % / Null / 100 %
Off / Auto / 1 / 2 (*1)	Off / A / 1 / 2 / 2	Null / Null / Null / Null / Null
Off / Auto / 1 / 2 / 3 (*1)	Off / A / 1 / 2 / 3	Null / Null / Null / Null / Null
Off / Auto / 0..100 % (*2) (*3)	A / A / A / A / A	0 % / Null / 1..100 %

(*1) This configuration is not supported for a variable speed fan. The result **Out Variable Fan Speed Ovrd** is always NULL.

(*2) This configuration is not supported for a Staged Fan. The result is always **Out Fan Stage Ovrd** = Auto.

(*3) Not supported by Sylk-TR42.

Output

Table 294 Output of Wm Config Hvac A Function

Output Name	Description
Out Room Temp	This output indicates the room temperature, measured and received from the Sylk wall module or coming from the inputs **Room Temp External Wm** or **Room Temp Wired Wm**. The room temperature can be calibrated via **Room Temperature Calib Offs**. Usually, the external wall module has the highest priority, followed by the wired wall module and the Sylk wall module with the lowest priority. However, the Parameter **Sylk Sensor Arbitration Bits**, it's possible to calculate an average or use the minimum or maximum value. **Range:** -58 °F to 302 °F (-50 °C to 150 °C)
Out Room Humidity	This output indicates the relative room humidity, measured and received from the Sylk wall module or the inputs **Room Humidity External Wm** or **Room Humidity Wired Wm**. The relative room humidity can be calibrated via **Room Humidity Calib Offs**. Usually, the external wall module has the highest priority, followed by the wired wall module and the Sylk wall module with the lowest priority. However, the Parameter **Sylk Sensor Arbitration Bits**, it's possible to calculate an average or See notes above to use the minimum or maximum value. **Range:** 0 - 100 % relative Humidity
Out Room Air CO2	This output gives out the air quality (CO2), which is measured and received from the Sylk wall module or coming from the inputs **Room Air Quality External Wm** or **Room Air Quality Wired Wm**. The room air quality cannot be calibrated. Usually, the external wall module has the highest priority, followed by the wired wall module and the Sylk wall module with the lowest priority. However, the Parameter **Sylk Sensor Arbitration Bits**, it's possible to calculate an average or use the minimum or maximum value. **Range:** 0 - 5000 ppm CO2

Table 294 Output of Wm Config Hvac A Function (Continued)

Output Name	Description
Out Eff Occ Md	The output shows the effective occupancy mode. The effective occupancy mode is determined from the time program via the input **Occ Sched**, the Occupancy Sensor via the input **Occ Sensor** or the occupancy override command from the room user. The user gives the occupancy override command from the Sylk wall module, wired wall module, or external wall module via the input **Occ Md Cmd External Wm**. In addition, the entire Occupancy Logic can be bypassed via the **Occ Md Prio** input. For diagnostics, see also the other outputs such as **Out Cause** (Bit 6, Value 32 = Occupancy Override active, Bit 7, Value 64 = OccSensor is occupied, Bit 9, Value 256 = OccupancyPrio is active), **Out Occ Md Ovrd**, **Out Occ Md Ovrd Rem Time** and **Out Button Counter.** **Enums:** **2:** Occupied (Default) **3:** Off **4:** Holiday **5:** Unoccupied **6:** Standby **7:** Bypass
Out Occ Md Ovrd	The output indicates whether the room user has overwritten the occupancy mode. The effective occupancy logic results from the inputs **Occ Sched** and **Occ Sensor** without overwriting. The occupancy overwrite is carried out from the user via the wall module (Wired, TR42 or from the external wall module via **Occ Md Cmd External Wm**). If there is no overwrite from the user, then NO OCC OVRD is output. Otherwise, the occupancy mode selected by the user is given. In addition, the entire occupancy Logic can be bypassed via the **Occ Md Prio** input. In this case, an overwrite is possible, but it has no effect on the effective Occupancy mode **Out Eff Occ Md**. If the overwrite mode is only valid for a limited time, the remaining time is displayed in **Out Occ Md Ovrd Rem Time**. For diagnostics, see the other outputs such as **Out Cause** (Bit 6, Value 32 = Occupancy Override active, Bit 9, Value 256 = Occupancy Prio is active), **Out Eff Occ Md**, **Out Occ Md Ovrd Rem Time**, and **Out Button Counter**.

Table 294 Output of Wm Config Hvac A Function (Continued)

Output Name	Description
	Enums: **2:** Overwrite to Occupied active **3:** Overwrite to Off active **4:** Overwrite to Holiday active **5:** Overwrite to Unoccupied active **6:** Overwrite to Standby active **7:** Overwrite to Bypass active (Temporary Overwrite to Occupied, see parameter **Occ Ovrd Time**) **null:** No Occupancy Overwrite active (Default)
Out Occ Md Ovrd Rem Time	The output shows the remaining time of a temporary occupancy overwrite in minutes. If 0 min is reached, then the temporary overwrite is finished, and then the timer changes to **null**. The room user has overwritten the occupancy mode from the wall module (From TR42, wired or external wall module). Without overwriting, the effective occupancy logic would result from **Occ Sched** and **Occ Sensor** inputs. **Example**: If the user selects the occupancy overwrite mode BYPASS from the wall module. In this case, the BYPASS mode is active as defined by the input or parameter **Occ Ovrd Time**. **Out Occ Md Ovrd Rem Time** shows the remaining time of the BYPASS mode, counting from **Occ Ovrd Time** to 0min. If the timer is expired, the occupancy overwrite mode to BYPASS will be finished, and **Out Occ Md Ovrd** changes from BYPASS to Null. For diagnostics, see the other outputs such as **Out Cause** (Bit 6, Value 32 = Occupancy Override active, Bit 9, Value 256 = Occupancy Prio is active), **Out Eff Occ Md**, **Out Occ Md Ovrd Rem Time**, and **Out Button Counter**. **Range:** 0 – 1440 min, showing the remaining time. **null:** No occupancy overwrite timer is running (default)
Out Setpt Temp	This output presents the effective room temperature setpoint as an absolute value in °F or °C (Eng. Unit depending on the setting in workbench). The effective setpoint **Out Setpt Temp** is determined by: • The effective Occupancy mode **Out Eff Occ Md** reduces selection from the eight basic setpoints to 2 (Cooling + Heating Setpoint).

Table 294 Output of Wm Config Hvac A Function (Continued)

Output Name	Description
	• The setpoint adjuster of the wall module from Sylk or the external wall module via the input **Setpt Adjust External Wm** or from the wired wall module via the input **Setpt Adjust Wired Wm**. • The Input SetptPrio which bypasses the internal setpoint logic • The effective HVAC mode **Out Eff Hvac Md Bits**. If the HVAC mode contains only bits for cooling, the effective setpoint corresponds to a cooling setpoint; the same applies to heating. If **Out Eff Hvac Md Bits** = 0, which means neither cooling nor heating is possible, then the effective setpoint **Out Setpt Temp** is the value between cooling and heating (that is 71.6 °F (22 °C)) • Indirectly by the fan selection to OFF from the wall module, indicated by **Out Fan Stage Ovrd** set to OFF or **Out Variable Fan Speed Ovrd** set to 0 %, which is also reflected in the effective Occupancy mode **Out Eff Occ Md**. • The effective room temperature **Out Room Temp**. If both cooling and heating are possible (see **Out Eff Hvac Md Bits**), the room temperature determines whether a cooling or heating setpoint is selected. When switching between cooling and heating and vice versa, there are two timers involved, see **Setpt Md Delay Time** and **Setpt Off Time**. ***Note:*** *The setpoint adjustment of the Sylk-WM is retained after switching the supply voltage off and on (Power resistant or non-volatile memory).* **Range:** -58 °F to 302 °F (-50 °C to 150 °C)
Out Setpt Md	This output informs whether the effective room temperature setpoint **Out Setpt Temp** is a cooling or heating setpoint. See the description above. When **Out Setpt Temp** changes between a cooling and heating setpoint or vice versa, the output **Out Setpt Md** changes to OFF for the adjustable time **Setpt Off Time**, which is available as input and a parameter. This ensures that the change between cooling and heating is delayed, that the cooling and heating outputs are opened simultaneously, or that a mixture of cold and hot water is prevented. **Out Setpt Md** is OFF if **Out Eff Hvac Md Bits** = 0 is, which is the case if **Hvac Md Plant Bits** = 0 is. If **Out Room Temp** = Null, then **Out Setpt Md** = Htg if the HVAC mode allows heating. **Enums:** **3:** Off **4:** Clg **5:** Htg

Table 294 Output of Wm Config Hvac A Function (Continued)

Output Name	Description
Out Mid Occ Setpt Temp	For some applications, it is useful to know how the setpoint would be in Occupied mode, while the current occupancy mode is not OCCUPIED, for example, during night purge. **Out Mid Occ Setpt Temp** shows the value between occupied cooling and occupied heating considering the setpoint selection from the wall module. **Example:** **Setpt Clg Occ Byp:** 73.4 °F (23 °C) **Setpt Htg Occ Byp:** 69.8 °F (21°C) **Setpoint adjustment from wall module:** +5 °F delta (+3 °C delta) **Out Mid Occ Setpt Temp:** 70 + (73 – 70) / 2 + 5 = 76.5 °F **Out Mid Occ Setpt Temp:** 21 + (23 – 21) / 2 + 3 = 25 °C A Night purge can reduce the room temperature to 77 °F (25 °C) at night via unconditioned outside air. When the occupancy mode changes to occupied in the morning, the room is already at the right temperature considering the setpoint of the wall control unit. **Range:** -58 °F to 302 °F (-50 °C to 150 °C)
Out Setpt Wm Display	**Out Setpt Wm Display** returns the setpoint value shown on the display of the wall module (such as TR42). Depending on the Setpt Ovrd Type, the value can be a relative value or an absolute value. This is used for diagnosis or for a graphic of the wall control unit. ***Note:*** *The setpoint adjustment of the Sylk wall module is retained after switching the supply voltage off and on.* **Range:** -58 °F to 302 °F or °F delta (-50 °C to 150 °C or °C delta)

Table 294 Output of Wm Config Hvac A Function (Continued)

Output Name	Description
Out Fan Stage Ovrd	This output is valid only for a staged fan, not a variable speed fan. See **Out Variable Fan Speed Ovrd** for a variable speed fan. Normally, the application logic on the wiresheet determines the fan stage. However, the operator can switch the fan from the wall module to any desired position via **Out Fan Stage Ovrd**. When switching OFF via the wall module, the frost and overheating protection remain active, which means that the fan stage selection from the wall module will be ignored in this case. This is also the reason for connecting the fan stage via the input **Fan Stg External Wm** to the wall module function block when using an external wall module. For diagnostics, see also the other outputs such as **Out Cause** (Bit 11, Value 1024 = Fan Override is active). ***Note:*** *The fan adjustment of the Sylk wall module is retained after switching the supply voltage OFF and ON.* **Enums:** **2:** Auto (Default) **3:** Off **4:** Speed 1 (On) **5:** Speed 2 **6:** Speed 3
Out Variable Fan Speed Ovrd	This output is valid only for a variable speed fan and not for a staged fan. See Out Variable Fan Stage Ovrd for a staged fan. Normally, the application logic on the wiresheet determines the fan speed. However, the operator can switch the fan from the wall module to any desired position via **Out Variable Fan Speed Ovrd** (From external wall module only, not wired Sylk-TR42 wall module). When switching OFF via the wall module, the frost and overheating protection remain active, which means that the fan speed selection from the wall module will be ignored in this case. This is also the reason for connecting the fan speed via the input **Fan Variable Spd External Wm** to the wall module function block when using an external wall module. For diagnostics, see also the other outputs such as **Out Cause** (Bit 11, Value 1024 = Fan Override is active). **Range:** 0 - 100 % while 0 = Off **null:** Auto (Default)

Table 294 Output of Wm Config Hvac A Function (Continued)

Output Name	Description
Out Eff Hvac Md Bits	In order to understand the logic, it is necessary to know what is installed in the room, whether a cooling unit, a heating unit, or both are installed, or if a 2-pipe changeover system is installed. This information is passed to the function block via the **HVAC Room Application Bits** input. In addition, the primary plant controller must provide information about the available energy and whether the heating system, cooling system, or both systems are currently operational. This information is given to the function block via the **HVAC Md Plant Bits** input. Finally, the operator can set a specific HVAC mode like cooling, heating, or auto via the wall module. The HVAC mode distinguishes between cooling and heating and can also distinguish between water, electrical, and air energy. The flexibility is realized by using bits. The bits 1 to 8 stand for cooling and 9 to 16 for heating, whereby the bits have a fixed assignment. For example, bit 1 stand for water energy, bit 2 for electrical energy, bit 3 for fan only, and bit 4 for pretreated air energy. ***Note:*** *If **Out Eff Hvac Md Bits** is 0, then neither cooling nor heating is possible. In this case, one of the inputs **HVAC Room Application Bits**, **Hvac Md Plant Bits**, or **Hvac Md External Wm Bits**, probably has the value 0, or a non-available medium was selected as HVAC mode via the wall module.* ***Note:*** *The HVAC adjustment of the Sylk wall module is retained after switching the supply voltage OFF and ON.* **Bits** The HVAC mode output is a decimal value, although it represents single bits. Decimal values can be converted to binary values with a decimal to binary converter. If no bit is set, neither cooling nor heating is possible. **Bit 1:** Cooling with cold water **Bit 2:** Cooling with electrical energy **Bit 3:** Cooling with fan only **Bit 4:** Cooling with pretreated cold air **Bit 5:** Cooling Reserved **Bit 6:** Cooling Reserved **Bit 7:** Cooling for User definable **Bit 8:** Cooling for User definable

Table 294 Output of Wm Config Hvac A Function (Continued)

Output Name	Description
	Bit 9: Heating with hot water **Bit 10:** Heating with electrical energy **Bit 11:** Heating with fan only **Bit 12:** Heating with pretreated warm air **Bit 13:** Heating Reserved **Bit 14:** Heating Reserved **Bit 15:** Heating for User definable **Bit 16:** Heating for User definable A simple approach is to use the complete byte for cooling or heating. • **Cooling**: 255 means Bit 1 - Bit 8 is set to true. • **Heating**: 65280 means Bit 9 – Bit 16 is set to true. • **Auto**: 65535 means all Cooling and Heating Bit 1 – Bit 16 are true. **Example 1:** **Out Eff Hvac Md Bits** = 1538. Convert 1539 to binary results in 11000000010. Bit 11 is true: Heating with fan only. **Example 2:** • If **Out Eff Hvac Md Bits** = 0, then no cooling and no heating is possible. • If **Out Eff Hvac Md Bits** = 255, then cooling is possible. • If **Out Eff Hvac Md Bits** = 65280, then heating is possible. • If **Out Eff Hvac Md Bits** = 65535, then cooling and heating are possible.

Table 294 Output of Wm Config Hvac A Function (Continued)

Output Name	Description
Out Cause	The output **Out Cause** provides different states for diagnosis to see the current happening condition in the object. The output shows as a decimal value, although it represents single bits. With a decimal to the binary converter, you can convert the decimal value into bits (for example, enter in the browser: convert decimal value to binary). • **Bit 1, Value 1:** Frost or Overheat is active • **Bit 2 , Value 2:** Setpoint Adjustment from wired, Sylk, or external wall module is set. • **Bit 3, Value 4:** The input **Setpt Shift Demand Limit** is not 0. • **Bit 4 , Value 8:** The **Setpt Md Delay Time** is active. This is the time before switching between cooling and heating and vice versa. • **Bit 5, Value 16:** The **Setpt Off Time** is active. This is the time for a change between cooling and heating and vice versa in which the output **Out Setpt Md** is set to OFF. • **Bit 6, Value 32:** An occupancy override is active. • **Bit 7, Value 64:** The occupancy sensor via the input **Occ Sensor** is OCCUPIED. • **Bit 8, Value 128:** This bit is set for 1 DDC cycle after the conventional WM button is pressed for a short time. • **Bit 9, Value 256:** The input **Occ Md Prio** is active, which overwrites the internal occupancy logic. • **Bit 10, Value 512:** An Hvac Ovrd is active. • **Bit 11, Value 1024:** A Fan Ovrd is active, which means the selected fan speed isn't on AUTO. • **Bit 12, Value 2048:** The bit is set for 1 DDC cycle after the conventional WM button is pressed for a medium time. • **Bit 13, Value 4096:** The bit is set for 1 DDC cycle after the conventional WM button is pressed for a long time • **Bit 14, Value 8192:** The input **Setpt Prio** is active, overwriting the internal setpoint logic. • **Bit 15 , Value 16384:** There is a communication failure with the Sylk wall module. • **Bit 32, Value 32768:** The Sylk wall module is currently loaded (See also **Reset Wm**, Bit 32).

Table 294 Output of Wm Config Hvac A Function (Continued)

Output Name	Description
	Example: **Out Cause** = 1090 Convert 1090 to binary results in 10001000010. Starting from the right side (Bit 1): **Bit 2 is true:** A setpoint from the wall module is set **Bit 7 is true:** The occupancy sensor detects OCCUPIED. **Bit 11 is true:** The fan speed is selected manually from wall module.
Out Button Counter	The wired wall module button is connected to the terminal configured via **Byp Fan Terminal**. The **Out Button Counter** is incremented whenever the push button is pressed on the wired wall module. This allows the button to be tested or used for other functions within free programming. **Range:** 0..65535 - 0..65535

Tr40 Model

When you select the Tr40 Model as inputs, outputs, and parameters, more options are added to the wall module function block like Sylk wall module parameter group.

Parameter - General

Table 295 Parameter of Wm Config Hvac A

Parameter Name	Description
Wm Model	The function block supports various wall operating devices like wired, Sylk device or an external wall module. ***Note:*** *A wall module that communicates via BACnet or Modbus is called an external wall module. It can be a Honeywell wall module, such as the Modbus wall module TR80, or a third-party vendor wall module.* Even without a wall module, the function block determines the Occupancy mode, the Setpoint, the Heating or Cooling mode, and other things. Visibility rules determine whether input or output slots are shown or hidden. The same is applicable for parameters. **1:** No WM connected **2:** Tr40 (Temperature) **3:** Tr40-H (Temperature, Humidity) **4:** Tr40-CO2 (Temperature, Air Quality CO2) **5:** Tr40-H-CO2 (Temperature, Humidity, Air Quality CO2) **6:** Tr42 (Temperature) **7:** Tr42-H (Temperature, Humidity) **8:** Tr42-CO2 (Temperature, Air Quality CO2) **9:** Tr42-H-CO2 (Temperature, Humidity, Air Quality CO2) **20:** External WM (BACnet, Modbus, etc)
Occ Ovrd Selection	It is used to determine whether or not the room user has the ability to overwrite the occupancy mode from the wall module. As a result, I/O slots and parameters are displayed or hidden based on this parameter. Overwrites can be performed from a wired wall module with an overwrite button, a Sylk wall module via the display, or an external wall module. **Values:** **1:** No Occupancy Overwrite **2:** Occupancy Overwrite

Table 295 Parameter of Wm Config Hvac A (Continued)

Parameter Name	Description
Setpt Ovrd Type Par (In & Par)	The input or parameter defines whether the operator obtains a relative setpoint or an absolute setpoint selection via the wall module. At the same time, the setpoint selection can also be deactivated. Depending on the selected **Setpt Ovrd Type**, the limits **Min Clg Setpt Selection**, **Max Clg Setpt Selection**, **Min Htg Setpt Selection**, and Max Htg Setpt Selection must be adapted. **Enum:** **1:** No Setpoint Override **2:** Relative **3:** Absolute **null:** No Setpoint Override **Default Value:** **1:** No Setpoint Overwrite
Fan Ovrd Type Par (In & Par)	The function block supports a 1 – 3 speed fan and a variable speed fan. Normally the application determines the fan speed or position. If the room user does not like that fan speed, he can override it via the wall module. This input or parameter **Fan Ovrd Type** is used to configure the desired behavior of the fanspeed selection from the wall module. The programmer uses this to determine which the operator can select fan stages or speed from the wall module. The Staged Fan uses the output **Out Fan Stage Ovrd**. The variable speed fan the output **Out Variable Fan Speed Ovrd**. **Fan Ovrd Type** is not valid for an external wall module (BACnet, Modbus) coming via **Fan Stg External Wm** and **Fan Variable Spd** **External Wm**. "No Selection" means that the user has no possibility to change the fanspeed. In this case, the output for the staged fan is **Out Fan Stage Ovrd** = Auto, and the output for the variable speed fan is **Out Variable Fan Speed Ovrd** = Null (Auto). **Default Value:** **1:** No Selection: No fanspeed selection possible from wall module

Table 295 Parameter of Wm Config Hvac A (Continued)

Parameter Name	Description
Hvac Ovrd Selection	Hvac Ovrd Selection is used to select whether the room user may or may not overwrite the wall module's HVAC mode (Cooling or Heating). For example, the room user could switch to HEATING in the winter to avoid COOLING if the sun shines into the room. The overwrite can be done from a TR42 wall module or an external WM (Not from a wired wall module). With "No HVAC overwrite," the parameter Sylk wall module – Show Item Bits should also be adjusted. **Values:** **1:** No HVAC overwrite **2:** HVAC overwrite allowed
Expert Mode	In **Expert Mode**, more I/O slots and parameters are displayed for advanced programmers, providing more functionality. **Values:** **0:** Standard (Default) **1:** Expert
Out Save	• **Master Sync Enabled**: If you set it to "true," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out Room Temp**: To enable or disable the **Out Room Temp** feature. • **Out Eff Occ Md**: To enable or disable the **Out Eff Occ Md** feature. • **Out Setpt Tepm** To enable or disable the **Out Setpt Tepm** feature. • **Out Setpt Md** To enable or disable the **Out Setpt Md** feature. • **Out Eff Hvac Md Bits**: To enable or disable the **Out Eff Hvac Md Bits** feature.

Table 296 Parameter of Wm Config Hvac A – wired wall module

Wired wall module Parameters	Description
Byp Fan Terminal	This parameter is used to configure the connection terminal for the Occupancy override button and fan overwrite selection on the wired wall module. The details on the range and their respective supported controller's information are given below: **Range:** **UI1** – RS4N, RS5N, RL4N, RL6N, VA423B24N. **UI2** – RS4N, RS5N, RL4N, RL6N, VA423B24N. **UI3** – RS4N, RS5N, RL4N, RL6N, VA423B24N. **UI4** – RS4N, RS5N, RL4N, RL6N, VA423B24N. **UI5** – RL4N, RL6N, RL8N* **UI6** – RL4N, RL6N, RL8N* **UI7** – Not supported **UI8** – Not supported **UI9** – Not supported **UI10** – Not supported **UIO1** – VA75I24NM, VA75IB24NM, VA75M24NM, VA75MB24NM. **UIO2** – VA75I24NM, VA75IB24NM, VA75M24NM, VA75MB24NM. **UIO3** – VA75I24NM, VA75IB24NM, VA75M24NM, VA75MB24NM. **UIO4** – VA75I24NM, VA75IB24NM, VA75M24NM, VA75MB24NM. **UIO5** – VA75I24NM, VA75IB24NM, VA75M24NM, VA75MB24NM. **UIO6** – VA75I24NM, VA75IB24NM, VA75M24NM, VA75MB24NM. **UIO7** – VA75I24NM, VA75IB24NM, VA75M24NM, VA75MB24NM. **UIO8** – Not supported **UIO9** – Not supported **UIO10** – Not supported **UIO11** – Not supported **UIO12** – Not supported **UIO13** – Not supported **UIO14** – Not supported **UIO15** – Not supported **UIO16** – Not supported

Table 296 Parameter of Wm Config Hvac A – wired wall module (Continued)

Wired wall module Parameters	Description
Led Terminal	Some wired wall modules have a LED to indicate the effective occupancy mode or an occupancy overwrite. Depending on the hardware used (Spyder Model 5 and Spyder Model 7), a separate LED output is available, or the LED is operated on an AO with 0 V = LED OFF or 5 V = LED ON. The details on the range and their respective supported controller's information are given below: **Range:** **AO1** – RS4N, RS5N, RL4N, RL6N, RL8N*, VA423B24N. **AO2** – RL4N, RL6N, RL8N*, VA423B24N. **AO3** – Not supported **AO4** – Not supported **AO5** – Not supported **AO6** – Not supported **Led Output** – RL8N* **UIO1** – VA75I24NM, VA75IB24NM, VA75M24NM, VA75MB24NM. **UIO2** – VA75I24NM, VA75IB24NM, VA75M24NM, VA75MB24NM. **UIO3** – VA75I24NM, VA75IB24NM, VA75M24NM, VA75MB24NM. **UIO4** – VA75I24NM, VA75IB24NM, VA75M24NM, VA75MB24NM. **UIO5** – VA75I24NM, VA75IB24NM, VA75M24NM, VA75MB24NM. **UIO6** – VA75I24NM, VA75IB24NM, VA75M24NM, VA75MB24NM. **UIO7** – VA75I24NM, VA75IB24NM, VA75M24NM, VA75MB24NM. **UIO8** – Not supported **UIO9** – Not supported **UIO10** – Not supported **UIO11** – Not supported **UIO12** – Not supported **UIO13** – Not supported **UIO14** – Not supported **UIO15** – Not supported **UIO16** – Not supported
Led Mode	The wired WM can display the Effective occupancy mode (recommended) or occupancy overwrite via the LED. **Range:** **0:** LED is not used. **1:** Show the Occupancy override. **2:** Show the effective Occupancy Mode (Default).

Note: *RL8N* – Totally, there are only two physical analog inputs available to connect a wired wall module with an RL8N controller, but three physical analog inputs are required to connect a wired wall module to read the room temperature, the setpoint, and the Fan speed switch and occupancy selection.*

Table 297 Parameter of Wm Config Hvac A - Sylk wall module

Parameters – Sylk wall module	Description
Wm Address	This parameter is used to configure the Sylk wall module address. **Range:** 0-15 (1 = Default = Factory default from TR42 wall module)
Screen Layout Par (In & Par)	**Screen Layout** is used to select the appearance of the display. For the Tr42 wall module, there is the possibility to choose between the English text and standardized symbols. **Enums:** **1:** Show standardized symbols **2:** Show English text **3:** Symbols and Degree F (*1) **4:** Symbols and Degree C (*1) **5:** English text and Degree F (*1) **6:** English Text with Degree C (*1) (*1) Displayed on Display in selected C or F, but the Input and Output Slots and parameter remain in the unit defined in the engineering tool. **Default Value:** **1:** Show standardized Symbols
Home Screen Par (In & Par)	The Tr42 wall module displays a home screen after powerup or a few seconds after an operation. A value can be displayed in large font on the home screen, which can be selected here. **Enums:** **1:** Black **2:** Scroll all sensor values **3:** Room Temp **4:** Setpoint **5:** Humidity **6:** Air Quality **Default Value:** **3:** Room Temperature
Show Items Bits	In addition to the home screen, the TR42 display shows various other values or menus for operation, which are defined here. Add the bit values of the items you want the operator to see. **Bit-Values:** **1:** Show Room Temperature **2:** Show Setpoint **4:** Show the effective occupancy mode **8:** Show and allow the change of the occupancy overwrite **16:** Show and allow the change of the effective HVAC mode **32:** Show the fan stage and allow the change of the fan stage **64:** Show the humidity **128:** Show the air quality **Default:** 255 (All Bits true, show all)

Table 297 Parameter of Wm Config Hvac A - Sylk wall module (Continued)

Parameters – Sylk wall module	Description
Sylk Sensor Arbitration Bits	In some cases, in addition to the Sylk wall module, there can be an external sensor (such as from BACnet) or a wired sensor or both. For example, because the Sylk wall module is poorly placed, calculate an average value. Inputs with the value NULL are ignored for that calculation. The external sensor is usually given the highest priority, followed by the wired and Sylk sensors. Bit values can be used to combine various sensors. Add the bit values of the desired operations. **Example:** The average room temperature of the Sylk wall module and a wired room sensor is to be calculated as the effective room temperature **Out Room Temp**. The maximum humidity of the Sylk wall module and a wired humidity sensor are to be used as **Out Room Humidity**. **Sylk Sensor Arbitration Bits** = 1 + 128 = 129. **Bit-Values:** **0:** Default (Highest priority has the external sensor, then wired, then Sylk) **1:** Average room temperature from all valid values **2:** Reserved. Do not use. **4:** Minimum value of all valid room temperatures **8:** Maximum value of all valid room temperatures **16:** Average room humidity from all valid values **32:** Reserved. Do not use. **64:** Minimum value of all valid room humidity **128:** Maximum value of all valid room humidity **256:** Average room air quality from all valid values **512:** Reserved. Do not use. **1024:** Minimum value of all valid room air quality **2048:** Maximum value of all valid room air quality

Table 298 Parameter of Wm Config Hvac A - Roomtemp

Parameters – Roomtemp	Description
Room Temp Calib Offs Par (In & Par)	The Calibration Offset is added to the room temperature, whether the room temperature comes from **Room Temp External Wm**, from **Room Temp Wired Wm**, or the Sylk wall module. The offset can be positive or negative. Thus, a measuring error can be corrected, for example, by an inappropriate positioning of the WM. **Range:** -58 °F to 302 °F (-50 °C to 150 °C) **Default Value:** 0 Delta °F (0 Delta °C).

Table 299 Parameter of Wm Config Hvac A - Humidity

Parameters – Humidity	Description
Room Humidity Calib Offs Par (In & Par)	The Calibration Offset is added to the room humidity, whether the room humidity comes from **Room Humidity External Wm**, **Room Humidity Wired Wm**, or the Sylk wall module. The offset can be positive or negative. Thus, a measuring error can be corrected, for example, by an inappropriate positioning of the WM. **Range:** -100 to 100 % **Default Value:** 0 Delta % r.H.

Table 300 Parameter of Wm Config Hvac A - Occupancy

Parameters – Occupancy	Description
Occ Sensor Behav Bits	If an occupancy sensor is present, it can be used to influence the occupancy mode. The different occupancy behaviors can be combined via these bits. Add the bit values of the operations you want. With this, you can realize two concepts: 1. With OCCUPIED, the occupancy sensor increases the comfort 2. With UNOCCUPIED, the occupancy sensor reduces the comfort (recommended) ***Note:*** *Do not mix the concepts.* **Example**: The scheduler = OCCUPIED but the Occupancy Sensor = UNOCCUPIED. The effective occupancy mode **Out Eff Occ Md** should change to Standby. Use the value 1 = Occ to Stby. **Bit-Values for concept 1** **4:** Stby to Occ **16:** Unocc to Occ **32:** Unocc to Stby **Bit-Values for concept 2** **1:** Occ to Stby **2:** Occ to Unocc **8:** Stby to Unocc

Table 301 Parameter of Wm Config Hvac A – Occupancy Ovrd

Parameters – Occupancy Ovrd	Description
Occ Ovrd Type Bits Par (In & Par)	The occupancy mode determined from **Occ Sched** and **Occ Sensor** can be changed at any time via the wall module (External BACnet/Modbus, wired, or the Sylk wall module). The Occ Ovrd Type Bits parameter defines which occupancy overrides from wired and Sylk wall modules should be possible for this application. If the wall module user does not change the occupancy mode, enter zero here. Otherwise, add the bit values of the occupancy modes which are to be selected by the operator. The supported occupancy override modes depend on the wall module. See **Wm Model**. Bitvalues for **Wm Model** Wired WM with LED Button. **0:** NoManOvrd **2:** Holiday **4:** Unocc **32:** Bypass until event **Occ Sched** = Occ, see also **Occ Ovrd Time** **null:** NoManOvrd. Bitvalues for **Wm Model** Tr42. **0:** NoManOvrd **32:** Bypass, see also **Occ Ovrd Time** **null:** NoManOvrd. **Default Value:** **0:** No manual occupancy overwrites

Table 301 Parameter of Wm Config Hvac A – Occupancy Ovrd (Continued)

Parameters – Occupancy Ovrd	Description
Occ Ovrd Behav Bits	A room user can overwrite the occupancy mode via the wall module. The overwrite can be manually reset on the wall module via Wm Reset input or by an automatic condition defined here. **Example**: The room user adjusts the occupancy mode via the wall module at 17:00 (5:00 pm) to leave the office. The next day at 7:00 (7:00 am), the office should be in an OCCUPIED state. Solution: At 17:00 (5 pm), the user overwrites the occupancy mode to UNOCCUPIED. **Occ Ovrd Behav Bits** get the value 8, which means that next morning when **Occ Sched** receives an OCCUPANCY mode, the occupancy override to UNOCCUPIED is reset, and the **Occ Sched** mode becomes active as **Out Eff Occ Md**. **Bit-Values:** **1:** Overwrite to Unoccupied until **Occ Sched** changes to Occupied **2:** Overwrite to Unoccupied until **Occ Sched** changes to Stby **4:** Overwrite to Unoccupied until **Occ Sched** changes to Unocc **8:** Overwrite to Unoccupied until **Occ Sensor** changes to Occ **16:** Overwrite to Unoccupied until **Occ Sensor** changes to Unocc **32:** Bypass until **Occ Sched** changes to Occ. The Bypass is canceled then, **Out Eff Occ Md** is set to Occ, and Occ Md Ovrd Rem Time is set to Null.

Table 301 Parameter of Wm Config Hvac A – Occupancy Ovrd (Continued)

Parameters – Occupancy Ovrd	Description
Occ Ovrd Time Par (In & Par)	If no occupancy sensor is installed in the room or if the occupancy sensor does not switch from the UNOCCUPIED to the OCCUPIED mode, the wall module user can select the occupancy override mode BYPASS. The user can select an occupancy override to BYPASS from the external wall module, the wired, and the Sylk wall module. For the wired and the Sylk wall module, **Occ Ovrd Type Bits** need to be set accordingly. The occupancy mode BYPASS uses the same setpoint as the occupied mode, but only for a certain period. This duration is entered here over **Occ Ovrd Time**. When the time has elapsed, the occupancy mode returns to the value determined by **Occ Sched** and **Occ Sensor**. The BYPASS occupancy mode can also be terminated earlier by the user via the wall module. **Example:** The occupancy mode on the weekend is Unoccupied. Suppose you come to the office and choose the mode BYPASS as occupancy override at the wall module. The BYPASS mode is activated for an **Occ Ovrd Time** of 480 min. After 300 minutes, if you leave the office and forget to reset the BYPASS mode on the wall module. After 480 min, the occupancy override mode BYPASS is terminated, and the effective occupancy mode changes back from BYPASS to UNOCCUPIED. **Range:** 0 – 1440 min, **null** means 0 min **Default Value:** 180 min

Table 302 Function – Setpoint

Setpoint Parameters	Description
Setpt Clg Overheat Off Holiday Par (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. There is a cooling and heating setpoint for each occupancy mode (OCCUPIED, STANDBY, UNOCCUPIED, OFF / HOLIDAY). See the separate setpoint description below. This input or parameter represents the room temperature cooling setpoint when the room control is switched off (such as Fanspeed overwrite to OFF from wall module), or the occupancy mode is in holiday mode (such as Occupancy overwrite to HOLIDAY from wall module). In this case, the output **Out Eff Occ Md** is set to OFF or HOLIDAY. The setpoint is higher than the **Setpt Clg Unocc** setpoint to save as much energy as possible while the room is not used for a longer time. The setpoint is activated when a cooling application is installed (See input **HVAC Room Application Bits**), and a cooling medium is present (See input **Hvac Md Plant Bits**), and the room user enabled cooling via the wall module (See input **Hvac Md External Wm Bits** and Hvac Md Internal Wm Bits). In this case, the output Eff Hvac Md Bits represents a cooling HVAC mode. If this setpoint is active, then the output **Out Setpt Md** shows cooling. Irrespective of this setpoint, the **Room Temp Overheat** input protects the building from overheating damage by simply overwriting the cooling and heating outputs independent of the HVAC, setpoint, and PID control. **Range:** -58 °F to 302 °F (-50 °C to 150 °C) **Default Value:** 95 °F (35 °C)
Setpt Clg Unocc Par (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. This input or parameter represents the room temperature cooling setpoint when the effective Occupancy mode is UNOCCUPIED. In this case, the output **Out Eff Occ Md** is set to UNOCCUPIED. **Range:** -58 °F to 302 °F (-50 °C to 150 °C) **Default Value:** 82.4 °F (28 °C)
Setpt Clg Stby Par (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. This input or parameter represents the room temperature cooling setpoint when the effective Occupancy mode is STANDBY. In this case, the output **Out Eff Occ Md** is set to STANDBY. **Range:** -58 °F to 302 °F (-50 °C to 150 °C) **Default Value:** 77 °F (25 °C)

Table 302 Function – Setpoint (Continued)

Setpoint Parameters	Description
Setpt Clg Occ Byp Par (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. This input or parameter represents the room temperature cooling setpoint when the effective Occupancy mode is OCCUPIED or BYPASS (temporary Occupied). In this case, the output **Out Eff Occ Md** is set to OCCUPIED or BYPASS. **Range:** -58 °F to 302 °F (-50 °C to 150 °C) **Default Value:** 73.4 °F (23 °C)
Setpt Htg Occ Byp Par (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. This input or parameter represents the room temperature heating setpoint when the effective Occupancy mode is OCCUPIED or BYPASS (temporary Occupied). In this case, the output **Out Eff Occ Md** is set to OCCUPIED or BYPASS. **Range:** -58 °F to 302 °F (-50 °C to 150 °C) **Default Value:** 69.8 °F (21°C)
Setpt Htg Stby Par (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. This input or parameter represents the room temperature heating setpoint when the effective Occupancy mode is STANDBY. In this case, the output **Out Eff Occ Md** is set to STANDBY. **Range:** -58 °F to 302 °F (-50 °C to 150 °C) **Default Value:** 66.2 °F (19 °C)
Setpt Htg Unocc Par (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. This input or parameter represents the room temperature heating setpoint when the effective Occupancy mode is UNOCCUPIED. In this case, the output **Out Eff Occ Md** is set to UNOCCUPIED. **Range:** -58 °F to 302 °F (-50 °C to 150 °C) **Default Value:** 60.8 °F (16 °C)
Setpt Htg Frost Off Holiday Par (In & Par)	The room temperature setpoint is determined by the effective occupancy mode **Out Eff Occ Md**. This input or parameter represents the room temperature heating setpoint when the room control is switched off (such as Fanspeed overwrite to Off from wall module), or the occupancy mode is in HOLIDAY mode (such as Occupancy overwrite to HOLIDAY from wall module). In this case, the output **Out Eff Occ Md** is set to OFF or HOLIDAY. **Range:** -58 °F to 302 °F (-50 °C to 150 °C) **Default Value:** 46.4 °F (8 °C)

Table 302 Function – Setpoint (Continued)

Setpoint Parameters	Description
Setpt Md Delay Time	This time prevents frequent changes between cooling and heating (Setpt Md). If the room temperature falls below the heating setpoint in cooling mode or if the room temperature exceeds the cooling setpoint in heating mode, the time is started. If the condition is stable for the entire time, the setpoint mode change between cooling and heating and vice versa is performed. If the condition is no longer given, the time is stopped and starts again from the beginning when the condition is again fulfilled. **Range:** 0-14400 sec
Setpt Off Time Par	When a change is determined from a cooling setpoint to a heating setpoint or vice versa, shown on **Out Setpt Md**, the setpoint mode always first changes to the OFF state for the **Setpt Off Time** before the new setpoint mode is set. 0 or "null" deactivates the Off Mode. This ensures that a heating and cooling valve is not opened simultaneously. **Example:** If there is a setpoint mode change from cooling to heating when the cooling valve is fully open, for example, because the setpoint on the wall module has been changed, the cooling valve will close while the heating valve opens simultaneously. With a 4-pipe system, the cold water return flow could be increased depending on the position of the registers; with a 4-pipe system using a common register for cooling and heating (that is ceiling), even mixing between cold and warm water would take place, since both valves are open at the same time. During the Off-Mode, the cooling valve closes and remains closed until the **Setpt Off Time** has elapsed. **Range:** 0 - 28800 sec **Default Value:** 0 sec

Table 303 Parameter of Wm Config Hvac A Function – Occupancy Ovrd

Setpoint Override Parameters	Description
Min Clg Setpt Selection Par (In & Par)	If the user selects the setpoint of the wired WM **Setpt Adjust Wired Wm**, the Sylk wall module (not the value of the external wall module **Setpt Adjust External Wm**) is limited by this input or parameter. The limitation can be done via the application depending on for example, the effective Occupancy mode **Out Eff Occ Md**. **null** means that the setpoint of the wall module cannot be changed in the cooling mode (**Out Setpt Md** is Cooling). Typical Values: • If **Setpt Ovrd Type** is Relative: -9 °F delta (-5 °C delta) • If **Setpt Ovrd Type** is Absolute: 53.6 °F (12 °C) • If a different engineering unit is displayed on the wall module than is specified in the engineering tool Control Manager, then this range is also converted; this means this input or parameter does not need to be recalculated in the wiresheet. **Default Value:** 53.6 °F (12 °C) or -9 °F (-5 °C)
Max Clg Setpt Selection Par (In & Par)	If the user selects the setpoint of the wired WM **Setpt Adjust Wired Wm**, the Sylk wall module (not the value of the external wall module **Setpt Adjust External Wm**) is limited by this input or parameter. The limitation can be done via the application depending on for example, the effective Occupancy mode **Out Eff Occ Md**. "null" means that the setpoint of the wall module cannot be changed in the cooling mode (**Out Setpt Md** is Cooling). Typical Values: • If **Setpt Ovrd Type** is Relative: 9 °F delta (5 °C delta) • If **Setpt Ovrd Type** is Absolute: 86 °F (30 °C) • If a different engineering unit is displayed on the wall module than is specified in the engineering tool Control Manager, then this range is also converted; this means this input or parameter does not need to be recalculated in the wiresheet. • If **Max Clg Setpt Selection** < **Min Clg Setpt Selection**, then **Max Clg Setpt Selection** = **Min Clg Setpt Selection**. **Default Value:** 86 °F (30 °C) or +9 delta °F (+5 delta °C)

Table 303 Parameter of Wm Config Hvac A Function – Occupancy Ovrd (Continued)

Setpoint Override Parameters	Description
Min Htg Setpt Selection Par (In & Par)	If the user selects the setpoint of the wired WM **Setpt Adjust Wired Wm**, the Sylk wall module (not the value of the external wall module **Setpt Adjust External Wm**) is limited by this input or parameter. The limitation can be done via the application depending on the effective Occupancy mode **Out Eff Occ Md**. "null" means that the setpoint of the wall module cannot be changed in the heating mode (**Out Setpt Md** is Heating). Typical Values: • If **Setpt Ovrd Type** is Relative: -9 °F delta (-5 °C delta) • If **Setpt Ovrd Type** is Absolute: 53.6 °F (12 °C) • If a different engineering unit is displayed on the wall module than is specified in the engineering tool Control Manager, then this range is also converted; this means this input or parameter does not need to be recalculated in the wiresheet. **Default Value:** 53.6 °F (12 °C) or -9 delta °F (-5 delta °C)
Max Htg Setpt Selection Par (In & Par)	If the user selects the setpoint of the wired WM **Setpt Adjust Wired Wm**, the Sylk wall module (not the value of the external wall module **Setpt Adjust External Wm**) is limited by this input or parameter. The limitation can be done via the application depending on the effective Occupancy mode **Out Eff Occ Md**. **null** means that the setpoint of the wall module cannot be changed in the heating mode (**Out Setpt Md** is Heating). Typical Values: • If **Setpt Ovrd Type** is Relative: 9 °F delta (5 °C delta) • If **Setpt Ovrd Type** is Absolute: 86 °F (30 °C) • If a different engineering unit is displayed on the wall module than is specified in the engineering tool Control Manager, this range is also converted; this means this input or parameter does not need to be recalculated in the Wiresheet. • If Max Htg Setpt Selection < **Min Htg Setpt Selection**, then Max Htg Setpt Selection = **Min Htg Setpt Selection**. **Default Value:** 86 °F (30 °C) or +9 delta °F (+5 delta °C)

Table 303 Parameter of Wm Config Hvac A Function – Occupancy Ovrd (Continued)

Setpoint Override Parameters	Description
Setpt Ovrd Behav Bits	A room user can overwrite the setpoint via wall module. An overwrite can be performed manually by entering the Wm Reset input or automatically by a defined condition. **Example**: The room user overwrites the setpoint to +9 °F delta (+5 °C delta) on the TR42 wall module, and then he leaves the office. The next day the setpoint should not have any setpoint offset. **Setpt Ovrd Behav Bits** are set to 4, which means that when Occ Sched receives a UNOCCUPIED mode in the evening, the setpoint overwrites with +9 °F delta (+5 °C delta) are reset to 0 delta °F or delta C°. With the value 1 + 2 + 4 = 7, any **Occ Sched** change would reset the setpoint overwrite. **Bit-Values:** **1:** The Setpt overwrite is valid until **Occ Sched** changes to Occupied. **2:** The Setpt overwrite is valid until **Occ Sched** changes to Stby. **4:** The Setpt overwrite is valid until **Occ Sched** changes to Unocc. **8:** The Setpt overwrite is valid until **Occ Sensor** changes to Occ. **16:** The Setpt overwrite is valid until **Occ Sensor** changes to Unocc.

Table 304 Parameter of Wm Config Hvac A Function – Fan Ovrd

Fan Ovrd Parameters	Description
Fan Ovrd Behav Bits	A room user can overwrite the fan speed via the wall module. An overwrite can be performed manually by entering the Wm Reset input or automatically by a defined condition. **Example**: In the afternoon, the room user changes the fan stage to stage 3 before leaving the office. The fan should run in AUTO mode the next day without any fan overwrite. The value of **Fan Ovrd Behav Bits** is 1+2+7, which means that when Occ Sched receives an UNOCCUPIED mode in the evening, the fan stage overwrite to stage 3 is reset to AUTO. This saves energy until the next day. The fan overwrites reset to AUTO with every Occ Sched change, which is seven. **Bit-Values:** **1:** The Setpt overwrite is valid until **Occ Sched** changes to Occupied **2:** The Setpt overwrite is valid until **Occ Sched** changes to Stby **4:** The Setpt overwrite is valid until **Occ Sched** changes to Unocc **8:** The Setpt overwrite is valid until **Occ Sensor** changes to Occ **16:** The Setpt overwrite is valid until **Occ Sensor** changes to Unocc **32:** If **Out Fan Stage Ovrd** = Off or **Out Variable Fan Speed Ovrd** = 0 %, then set Eff Occ Md = Off. A running bypass time is stopped and, an occupancy overwrite is ignored and reset. **Out Setpt Temp** changes to **Setpt Clg Overheat Off Holiday** or **Setpt Htg Frost Off Holiday** with an **Out Setpt Md** = Clg or Htg or Off depending on the Room Temperature. Off if HVAC mode is missing.

Table 305 Parameter of Wm Config Hvac A Function – Hvac Mode Ovrd

Hvac Mode Ovrd Parameters	Description
Hvac Md Internal Wm Config Bits Par	Keep the number on 65535. Currently, this input is not supported by the Sylk Tr42-wall module and not by the wired wall module. **Default Value:** **Null:** 65535

Table 305 Parameter of Wm Config Hvac A Function - Hvac Mode Ovrd (Continued)

Hvac Mode Ovrd Parameters	Description
Hvac Ovrd Behav Bits	Users of the room can override the HVAC Mode (Cooling or Heating) via the wall module. An overwrite can be performed manually by entering the Wm Reset input or automatically by a defined condition. **Example**: In the afternoon, the room user changes the HVAC mode to Cooling, and then he leaves. The next day the HVAC mode should be AUTO. Otherwise, the heating will not work in the morning, and the room will keep cold. **HVAC Ovrd Behav Bits** receive the value 1, which means that the next day in the morning, when OCC Sched receives an OCCUPIED mode, the HVAC mode overwrite is reverted to AUTO. **Bit-Values:** **1:** The HVAC overwrite is valid until **Occ Sched** changes to Occupied **2:** The HVAC overwrite is valid until **Occ Sched** changes to Stby **4:** The HVAC overwrite is valid until **Occ Sched** changes to Unocc **8:** The HVAC overwrite is valid until **Occ Sensor** changes to Occ **16:** The HVAC overwrite is valid until **Occ Sensor** changes to Unocc

Relative Setpoint

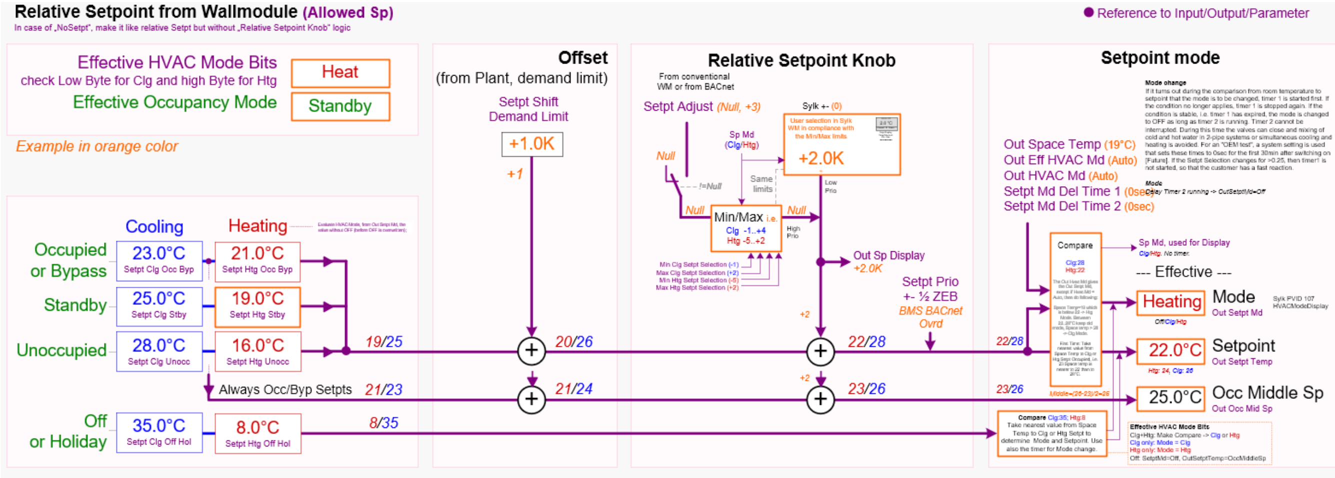


Fig. 472 WmConfigHvacA Function – Relative Setpoint

Absolute Setpoint

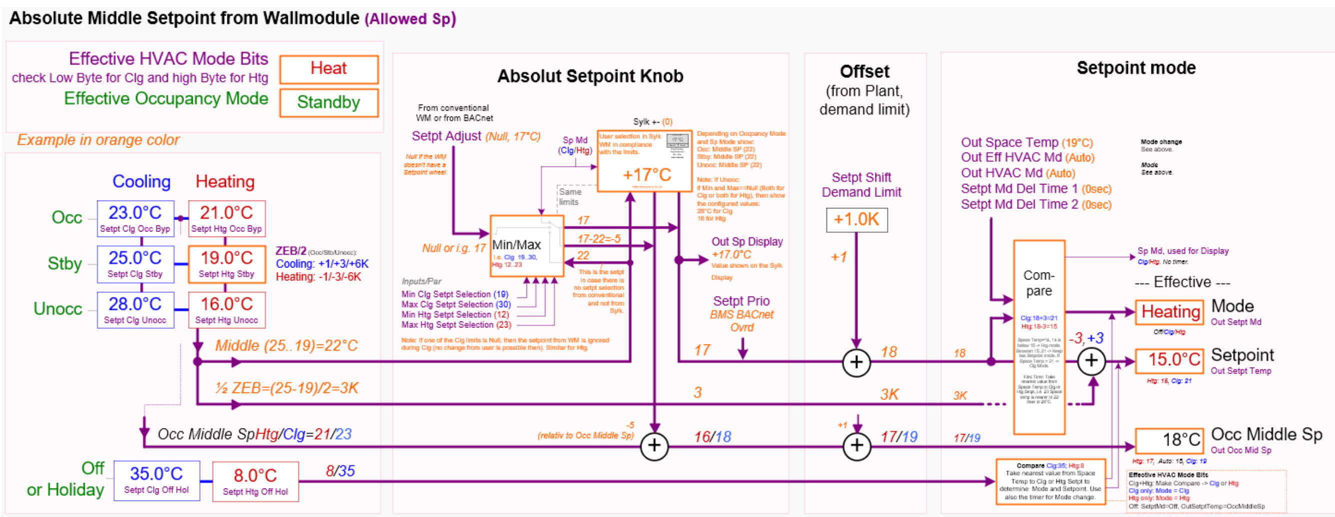


Fig. 473 WmConfigHvacA Function – Absolute Setpoint

Example: Wired Wallmodule on VAV IP controller

1. Onboard IO Wiresheet

Below you will find the print screens of the terminal function blocks. For the terminals Fan speed + Bypass and Wall module LED, no connections to the WmConigHvacA function block are necessary.

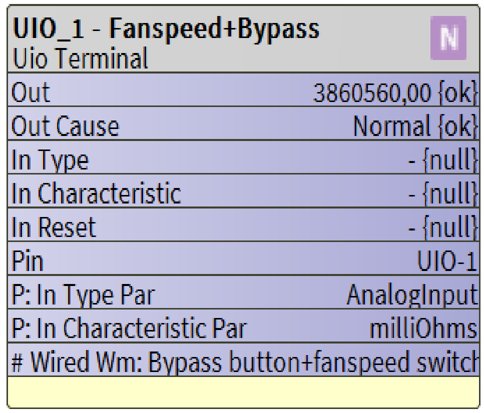


Fig. 474 Uio Terminal (Fanspeed+Bypass)

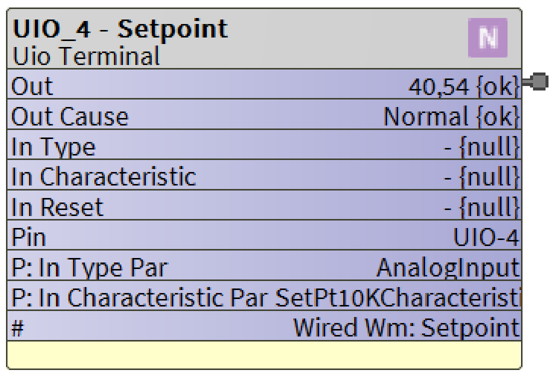


Fig. 475 Uio Terminal (Setpoint)

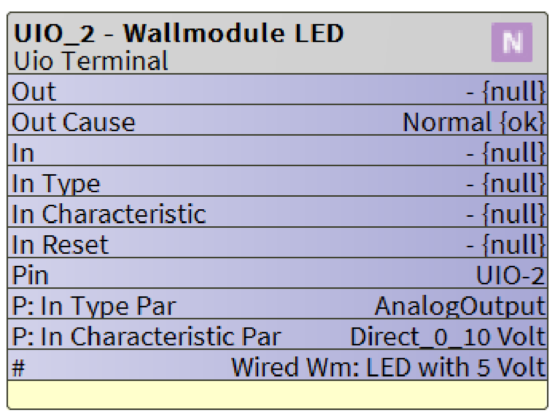


Fig. 476 Uio Terminal (Wall Module LED)

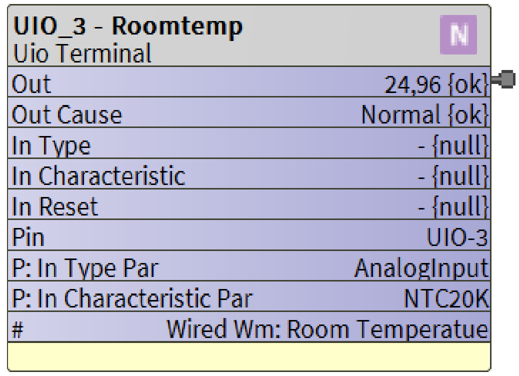


Fig. 477 Uio Terminal (Roomtemp)

2. Logic on periodic wiresheet to support the wired wall module.

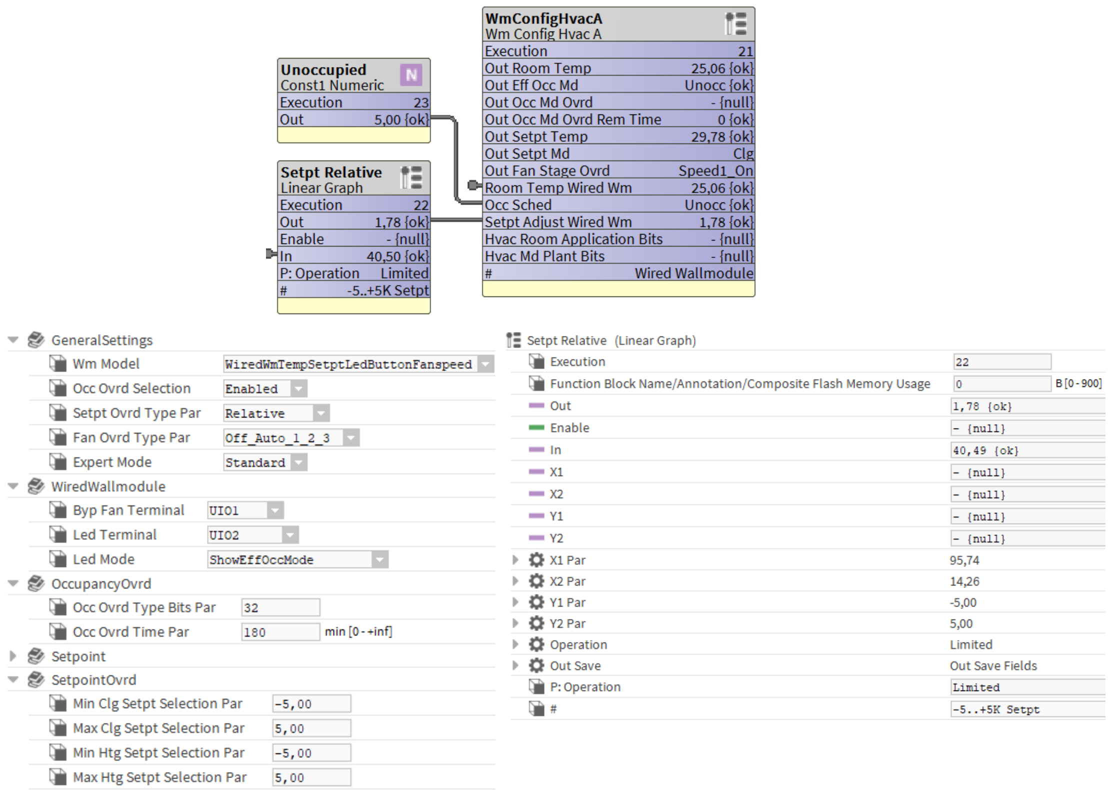


Fig. 478 Logic of wired wall module function block and property sheet

CHAPTER

19 MODBUS FUNCTION BLOCK

A Modbus device is defined via the Modbus device function block. Using the Modbus device configuration view, user can configure various types of read and write registers (for example, Coil, Discrete Input, Holding Register, Input Register, and so on). Refer to the IRM Engineering user guides for more information on Modbus device config view and device copying, duplicating, or templating.

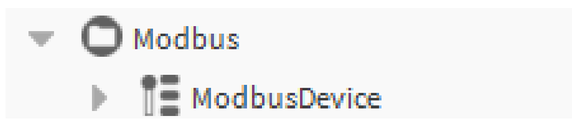


Fig. 479 Modbus Function Block

The Modbus device is shown on the left side in the figure below. The dotted line indicates which Modbus devices have which inputs and outputs.

In the wiresheet view, drag and drop the Modbus Device function block and double click on the Modbus Device to open the configuration view. Add the read and write data points. On the wiresheet, a function block Modbus Read Point or Modbus Write Point is automatically created for each data point. The names of the function blocks correspond to the name of the data points.

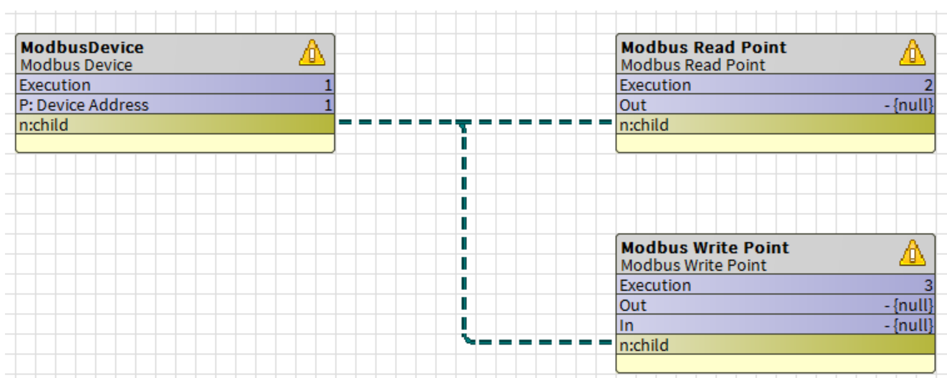


Fig. 480 Modbus Device

When a Modbus device fails to respond to a register poll command, the register is added to the list of "Failed Registers." Following that, the controller only polls this single register every 10 seconds. If the register was successfully read, it is read again in the defined interval.

Modbus Device Settings

In Control Manager, make the following changes.

Table 306 Modbus Device Settings

Device Setting	Description
Modbus Baudrate	**Supported Baud Rate:** 1200, 2400, 4800, 9600, 14400, 19200 (default), 38400, and 57600. **Default Baud Rate:** 19.2 Kbps If the speed is not important, select a low speed (for example, 9.6 kB). This increases the communication reliability.
Modbus Parity	Odd / Even / None **Default:** None
Modbus Stop Bits	1 or 2

Fig. 481   Modbus Device Setting in Control Manager

Modbus Device

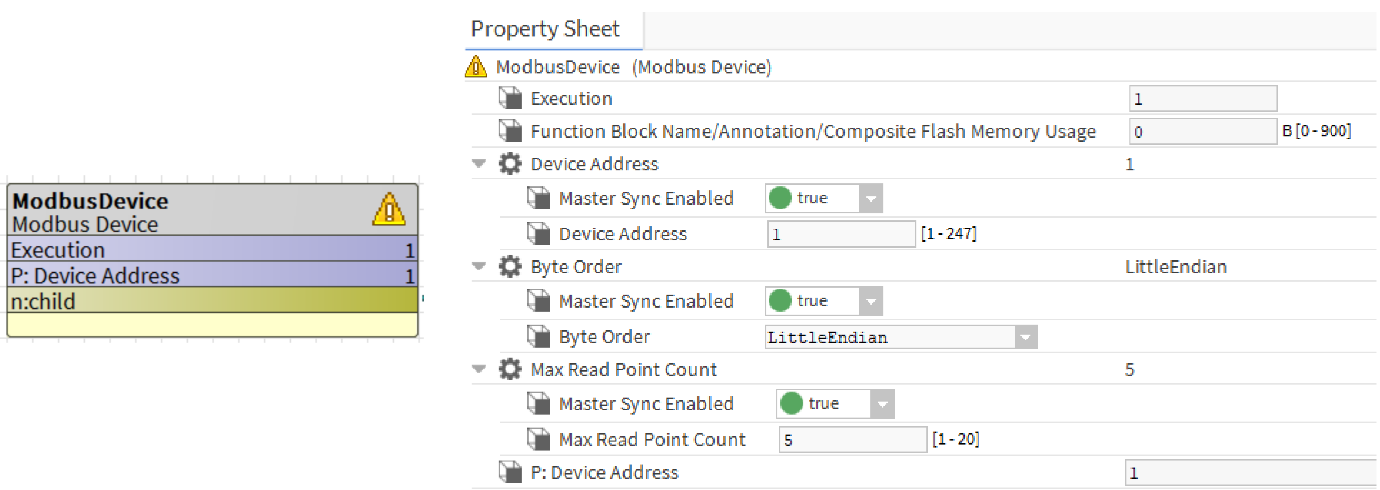


Fig. 482   Modbus Device Function Block and Property Sheet

Output

Table 307 Outputs of Modbus Device

Output Name	Description
n:child	Shows the relation to the corresponding Modbus Read and Write points.

Parameter

Table 308 Parameters of Modbus Device

Parameter Name	Description
Device Address	This is the Modbus device address connected to the Modbus network. Device Address is unique across Modbus network. **Numeric:** 32-Bit (integer); **Range:** 1- 247; **Default:** 0
Byte order	It specifies the byte order to be used when communicating with Modbus registers. Enumerated; **Range:** 0-3; **Default:** 1 **0: Big Endian (for example, Byte1, Byte 2, Byte 3, Byte 4 for a 32-bit integer value)** - For Register and for Word (2 register read), you get MSB first from the Modbus device. **1: Little Endian (for example, Byte 2, Byte 1 or Byte 4, Byte 3, Byte 2, Byte 1)** - For Register and for Word, you get LSB (2 register read) first from the Modbus device. **2: Little Endian with word swap (for example, Byte 2, Byte 1, Byte 4, Byte 3)** - For Register, you get LSB first, and for Word (2 register read), you get MSB first from the Modbus device. **3: Big Endian with word swap (for example, Byte 3, Byte 4, Byte 1, Byte 2)** - For Register, you get MSB first, and for Word (2 register read), you get LSB first from the slave device.
Max Read Point Count	It specifies the maximum number of registers that can be read from the Modbus device in a single polling cycle. This is the value with which the Modbus device will communicate. **Numeric:** 32-Bit (integer); **Valid Address Range:** 1-20; **Default:** 5

Modbus Configuration Screen

The Modbus device is configured by double-clicking on the Modbus Device function block. A special configuration screen is displayed, where Modbus points can be programmed. An example of how to create Modbus points is shown below.

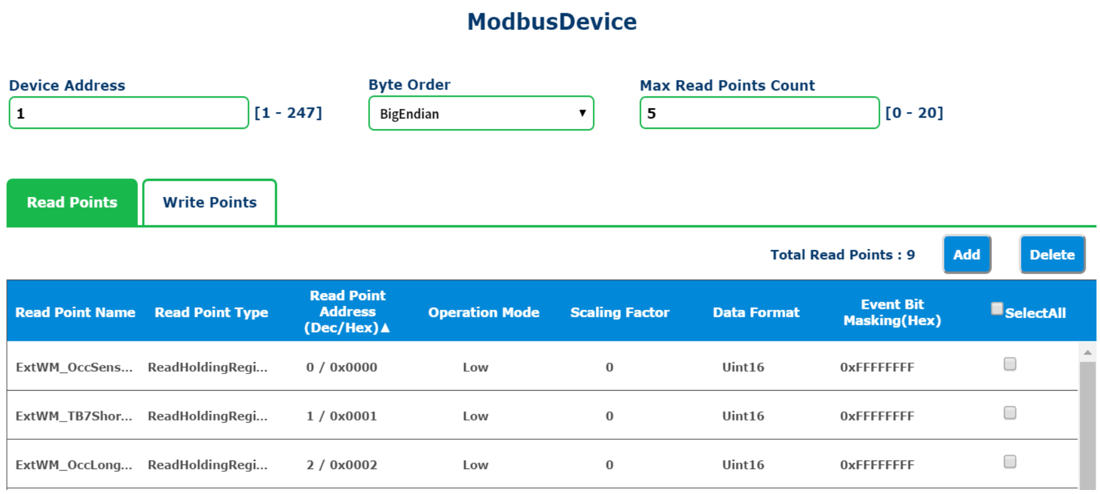


Fig. 483 Modbus Device Configuration Screen

Modbus Read Point

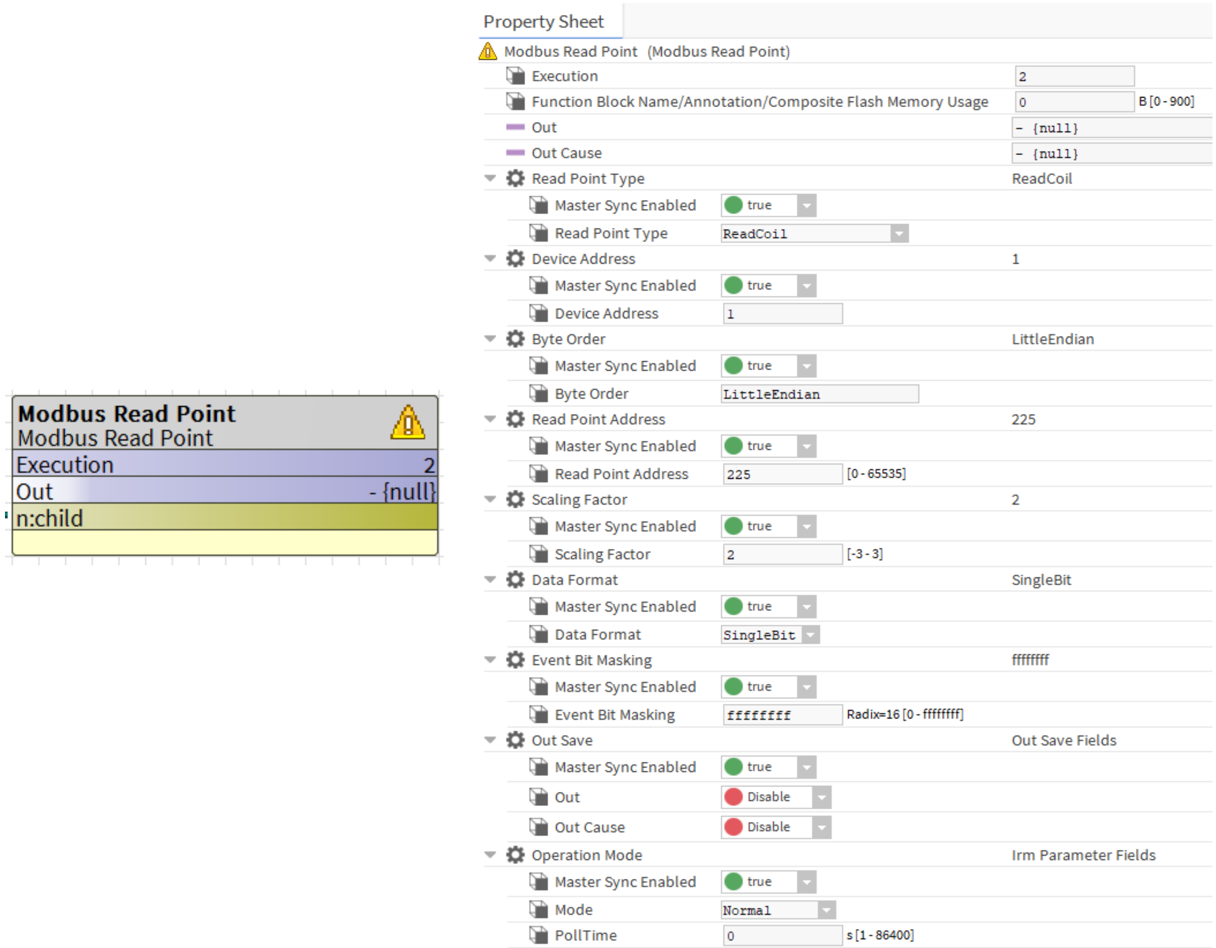


Fig. 484 Modbus Read Point Function Block and Property Sheet

Input

Table 309 Input of Modbus Read Point

Input Name	Description
n:child	Shows the relation to the corresponding Modbus device in the wiresheet.

Output

Table 310 Output of Modbus Read Point

Output Name	Description
Out	Modbus Output: 32-Bit value, Null (This value type can be Boolean, Integer, Unsigned Integer or float) • **Boolean:** This value type represents data in two states: true or false, on or off. This information is stored in a 32-bit (4-byte) register with a value ranging from 0 to 1. • **Signed Integer or Integer:** A signed 32-bit integer, is a variable type taking up 4 bytes in memory, with the ability to hold a value ranging from -2147483648 to 2147483647. The signed part of the integer refers to its ability to represent both positive and negative values. • **Unsigned Integer:** A 32-bit unsigned integer can hold a value from 0 to 4294967295. • **Float:** A float is a 32-bit single-precision value, sometimes called real. With a floating type value, very small and large numbers are possible. When the system cannot read from the register, it will display output as null.
Out Cause	**Exception Code or Cause, Numeric:** 32-bit value **OutCause Enums:** **0:** No error **1:** Illegal register address **2:** Illegal argument **3:** Porting layer error **4:** Insufficient resources **5:** I/O Error **6:** Protocol stack In Illegal state **7:** Retry I/O operation **8:** Timeout error occurred **10:** Illegal function exception **11:** Illegal data address **12:** Illegal data value **13:** Slave device failure **14:** Slave acknowledge **15:** Slave device busy **16:** Memory parity error **17:** Gateway path unavailable **18:** Gateway target device failed to respond **100:** Others **255:** Data Not Ready (When the register is in initial configuration).

Note: *Only Out and OutCause are exposed to the on wiresheet view. All other parameters are hidden.*

Parameter

Table 311 Parameters of Modbus Read Point

Parameter Name	Description
Read Point Type	Modbus Slave Register Type **Enumerated:** 1- 4, **Default:** 1 **1:** Read Coil (FC - 1) **2:** Read Discrete Inputs (FC - 2) **3:** Read Holding Registers (FC - 3) **4:** Read Input Registers (FC - 4)
Device Address	It is the slave device address (Modbus Slave devices connected to the serial network). The Device Address is unique across the Modbus network. **Numeric:** 32-Bit (Integer); **Range:** 1-247; Read-Only (This is not editable by the user; it is provided by the Modbus Device to which this register is configured.)
Byte Order	It specifies the byte order to be used when communicating with Modbus registers. • **0: Big Endian (for example, Byte1, Byte 2, Byte 3, Byte 4 for a 32-bit integer value)** - For Register and for Word (2 register read), you get MSB first from Modbus device. • **1: Little Endian (for example, Byte2, Byte 1 or Byte 4, Byte 3, Byte 2, Byte 1)** - For Register and for Word, you get LSB (2 register read) first from the Modbus device. • **2: Little Endian with word swap (for example, Byte 2, Byte 1, Byte 4, Byte 3)** - For Register, you get LSB first, and for Word (2 register read), you get MSB first from Modbus device. • **3: Big Endian with word swap (for example, Byte 3, Byte 4, Byte 1, Byte 2)** - For Register, you get MSB first, and for Word (2 register read), you get LSB first from Modbus device.
Read Point Address	**Numeric:** 32-Bit (integer); **Default:** 0 A 32-bit integer is a variable type taking up 4 bytes in memory, with the ability to hold a value ranging from - 0 to 65535.
Scaling Factor	To convert the register value to the required units, apply a scaling factor of 10 to it. A scale factor of +2 would multiply the value by 100, while a scale factor of -2 would multiply the value by 0.01. ***Note:*** *The parameter is not valid for Register Type 1 (Read Coil FC-1) and 2 (Read Discrete Inputs FC-2)*

Table 311 Parameters of Modbus Read Point (Continued)

Data Format	Determines the number of consecutive registers written and how the value is converted to register values. Enumerated; **Default:** 0 **0:** Single Bit **1:** Int16 **2:** uint16 **3:** int32 **4:** uint32 **5:** float (4-byte, single precision)
Event Bit Masking	Event Bit Masking determines which bits in the Modbus device data must be retained for processing the Read Data Register. The other bits are ignored. When a successful change of value in the masked bit area occurs, the Event program folder is invoked. **Numeric:** 32Bit (integer) **Range:** 0x00000000 to 0xFFFFFFFF (HEX); **Default Value:** 0xFFFFFFFF (HEX) • **Periodic Folder:** If the function block is in the periodic program wiresheet, there will be a maximum delay of 500 ms before the value is used for logic. • **Event Folder:** If we include the function block in the event program wiresheet, the logic will be executed immediately whenever the value changes to > 0.
Out Save	**Out Save Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. **Out:** To enable or disable the **Out** feature. **Out Cause:** To enable or disable the **Out Cause** feature.
Operation Mode	Enumerated; **Default:** 2: LOW **1:** Do not read (Output is NULL) **2:** Low Polling rate (Register is read after multiple of 5seconds) one register at a time. **3:** Medium Polling rate (Register is read after all high priority registers are read) one register at a time. **4:** High Polling rate (Register is read every 50miliseconds). ***Note:*** *If no high priority polling rate is assigned to any register, medium priority register will be read every 50 milliseconds one register at a time if present.*

Note: *The Modbus Read function block's parameters Device Address, Byte Order, and Max Read Registers Count are copied from the Modbus Device Function Block and are not editable. This can be done in the Modbus Device function block's parent.*

Add Modbus Read Point

Steps to add Modbus Read Point:

Step 1. In the Nav tree, expand the **IRM Program** folder of the controller and double-click on the **Periodic program**.

Step 2. Drag **ModbusDevice** from palette and drop it to periodic program wiresheet.

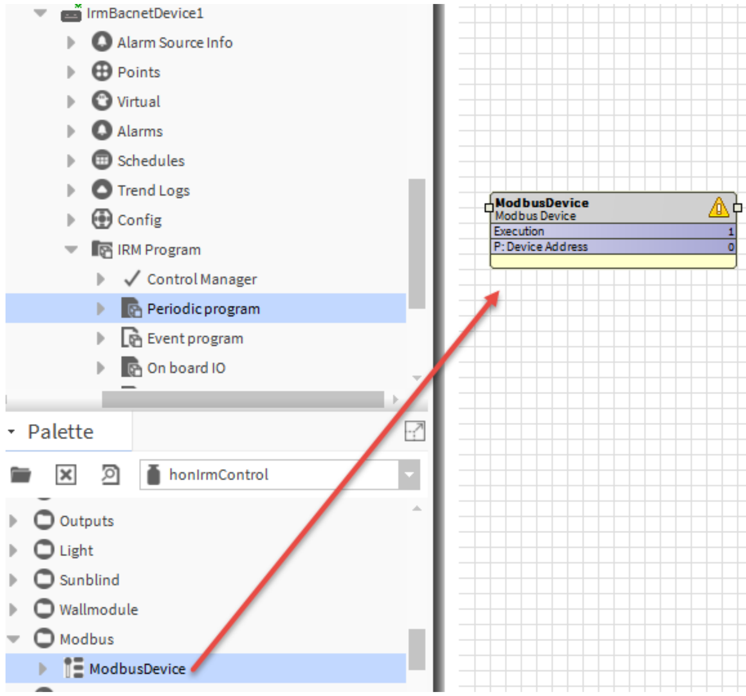


Fig. 485 Adding Modbus Device

If desired, you can change the Modbus device name.

Step 3. Double-click on the Modbus device to view below Modbus Device Configuration View screen.

Step 4. Select **Read Points** tab and click **Add**.

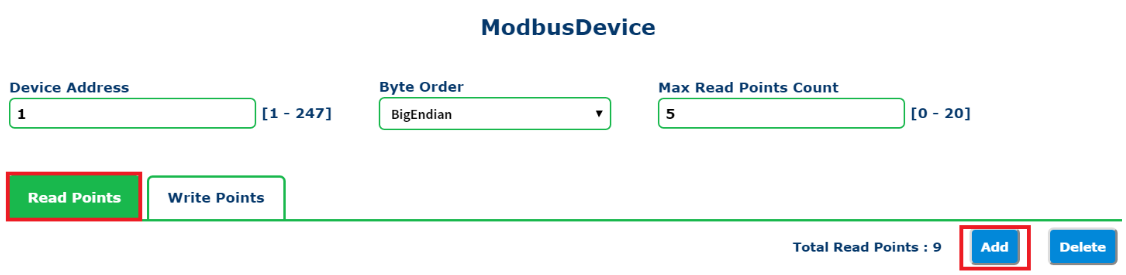


Fig. 486 Modbus Device Configuration Screen

Step 5. Enter the following detail like the Point name, Read Point address, Read Point Type, and Data Format in the **Modbus Read Point** window. In the below example, for Loytec Thermostat and its Modbus details.

Step 6. Click **OK.**

The L-STAT operates as a Modbus slave in Modbus RTU mode. The default Baudrate is set to 57600, the default parity is set to 'none' and the default address is set to 1. The communication with a Modbus master device will work with Modbus function code 0x03 (Read Holding Registers) Modbus function code 0x06 (Preset Single Register).

Register Name	Register Address	Bit Position															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
sensor_value_0	48 0x0030	internal temperature															

Fig. 487 Modbus Registers

Sensor value 0, sensor value 1 and sensor value 3 are 16 Bit signed values. All other sensor values are defined as 16 Bit unsigned since there are no negative values to expect.

Example 1: Modbus Read Device

The point type to read is "Read Holding Registers," and the register is "48." The operation mode must be set to Low (as the temperature is an analog input). Set the operation mode to Medium or High depending on the impact on the control), the scaling factor based on the output value and data format as unsigned Int16, and the Event Bit Masking to 0xFFFFFFFF because there is no need to block the bits.

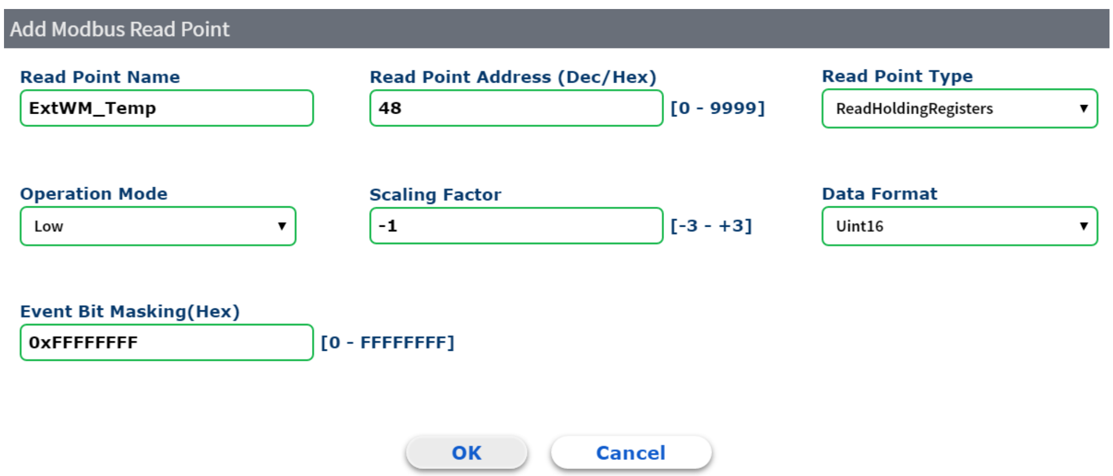


Fig. 488 Modbus Read Device Point configuration

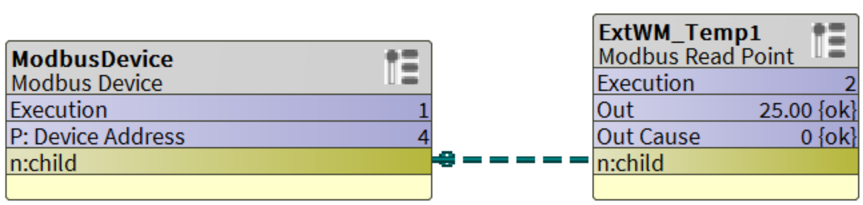


Fig. 489 Modbus Read Device Function Block

Modbus Write Point

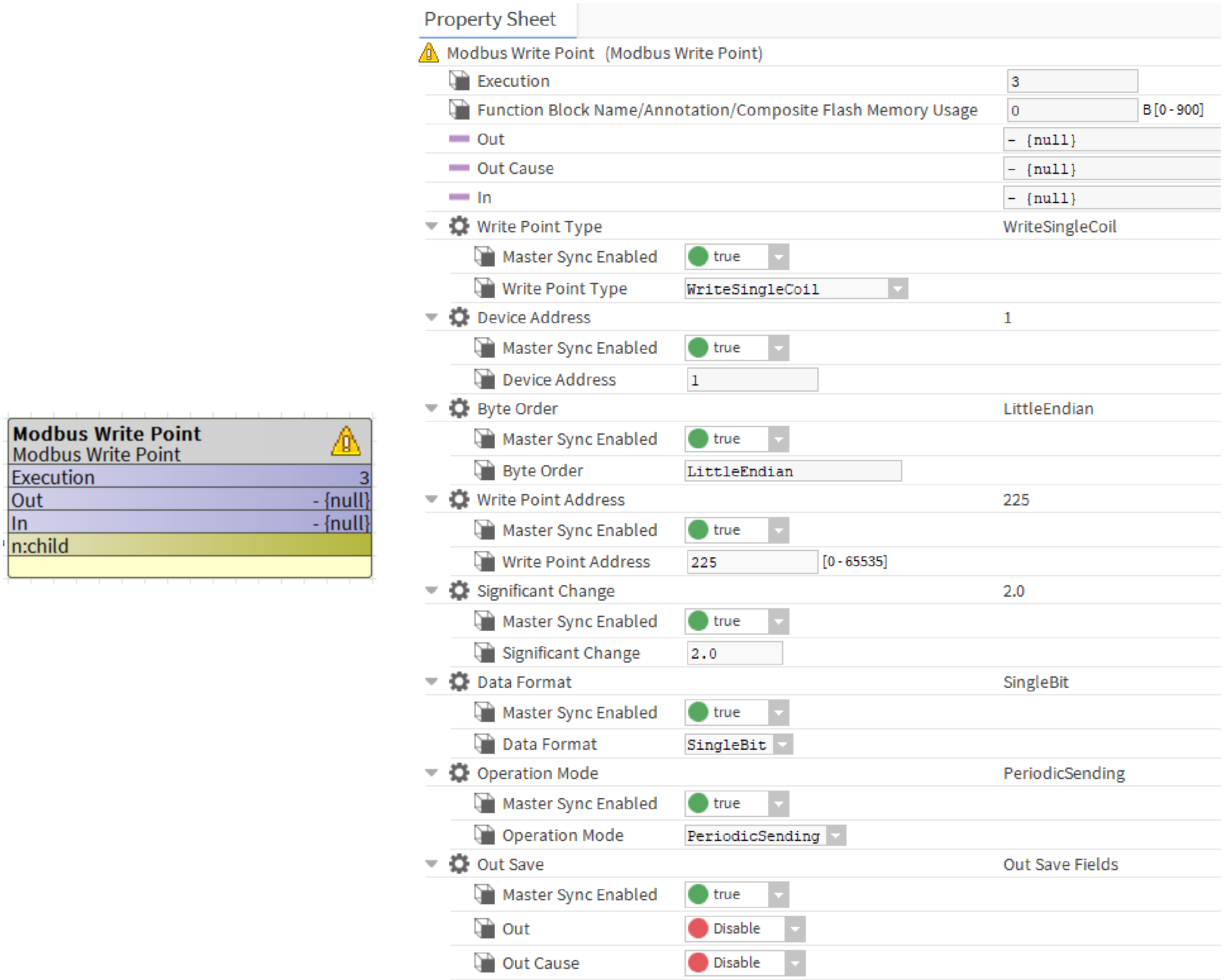


Fig. 490 Modbus Write Point Function Block and Property Sheet

Input

Table 312 Input of Modbus Write Point

Input Name	Description
In	Writes to the Modbus holding register **Numeric:** 32-Bit floating point value (float), “null” = Ignore
n:child	Shows the relation to the corresponding Modbus device in the wiresheet.

Output

Table 313 Output of Modbus Write Point

Output Name	Description
Out	Modbus Output: 32-Bit value, Null (This value type can be Boolean, Integer, Unsigned Integer or float) • **Boolean:** This value type represents the data in two states viz. true or false, on or off, etc. This data is stored in a 32-Bit (4 byte) register ranging from 0-1. • **Signed Integer or Integer:** A signed 32-bit integer is a variable type taking up 4 bytes in memory, with the ability to hold a value ranging from -2147483648 to 2147483647. The signed part of the integer refers to its ability to represent both positive and negative values. • **Unsigned Integer:** A 32-bit unsigned integer can hold a value from 0 to 4294967295. • **Float:** A float is a 32-bit single-precision value, sometimes called real. With a floating type value, very small and large numbers are possible. When the system is not able to read from the register, it will display output as null.
Out cause	**Exception Code or Cause, Numeric:** 32-bit value **Outcause Enums:** **0:** No error **1:** Illegal register address **2:** Illegal argument **3:** Porting layer error **4:** Insufficient resources **5:** I/O Error **6:** Protocol stack In Illegal state **7:** Retry I/O operation **8:** Timeout error occurred **10:** Illegal function exception **11:** Illegal data address **12:** Illegal data value **13:** Slave device failure **14:** Slave acknowledge **15:** Slave device busy **16:** Memory parity error **17:** Gateway path unavailable **18:** Gateway target device failed to respond **100:** Others **255:** Data Not Ready (When the register is in initial configuration).

Parameter

Table 314 Parameter of Modbus Write Point

Parameter Name	Description
Write Point Type	Modbus Slave Register Type **Enumerated:** 1-4, **Default:** 1 **5:** Write Single Coil (FC-5) **6:** Write Single Register (FC-6) **15:** Write Coils (FC-15) **16:** Write Multiple Registers (FC-16)
Device Address	Modbus device address (to which this coil belongs) Device Address is unique across the Modbus network.
Byte Order	It specifies the byte order to be used when communicating with the Modbus registers. • **0: Big Endian (for example, Byte1, Byte 2, Byte 3, Byte 4 for a 32-bit integer value)** - For Register and for Word (2 register read), you get MSB first from the Modbus device. • **1: Little Endian (for example,Byte2, Byte 1 or Byte 4, Byte 3, Byte 2, Byte 1)** - For Register and for Word, you get LSB (2 register read) first from the Modbus device. • **2: Little Endian with word swap (for example, Byte 2, Byte 1, Byte 4, Byte 3)** - For Register, you get LSB first, and for Word (2 register read), you get MSB first from the Modbus device. • **3: Big Endian with word swap (for example, Byte 3, Byte 4, Byte 1, Byte 2)** - For Register, you get MSB first, and for Word (2 register read), you get LSB first from the Modbus device.
Write Point Address	**Numeric:** 32-Bit (integer); **Default:** 0 A 32-bit integer is a variable type taking up 4 bytes in memory, with the ability to hold a value ranging from - 0 to 65535.
Significant Change	The difference between the current value of In and the last transmitted value that will cause the new value to be sent to the Modbus device. **Numeric:** 32-Bit floating point value (float) **Default**: 0.
Data Format	Enumerated; **Range:** 0 – 7; **Default:** 0 **0:** Single Bit **1:** Int16 **2:** uint16 **3:** int32 **4:** uint32 **5:** float (4-byte, single precision)

Table 314 Parameter of Modbus Write Point (Continued)

Parameter Name	Description
Operation Mode	Enumerated; **Range:** 1-2; **Default:** 2-Periodic Sending **1:** Do Not Send **2:** Periodic Sending
Out Save	• **Out Save Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Out:** To enable or disable the **Out** feature. • **Out Cause:** To enable or disable the **Out Cause** feature.

Add Modbus Write Point

Steps to add Modbus Write Point:

Step 1. Double-click on the Modbus device to view below Modbus Device Configuration View screen.

Step 2. Select **Write Points** tab and click **Add**.

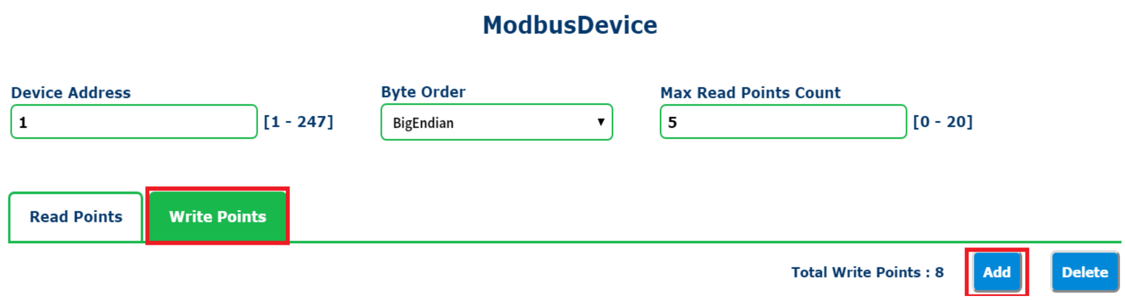


Fig. 491 Modbus Write Device Configuration Screen

Step 3. Enter the following detail like the Point name, Write Point address, Write Point Type, and Data Format in the **Modbus Write Point** window. In the below example, for Loytec Thermostat and its Modbus details.

Step 4. Click **OK.**

Register Name	Register Address	Bit Position															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
set_point_max_0	416 0x01A0	[0x012C]															

Fig. 492   Modbus Registers

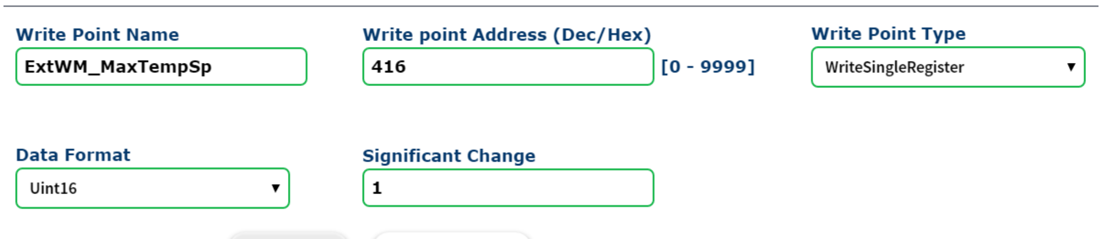


Fig. 493   Modbus Write Device Configuration

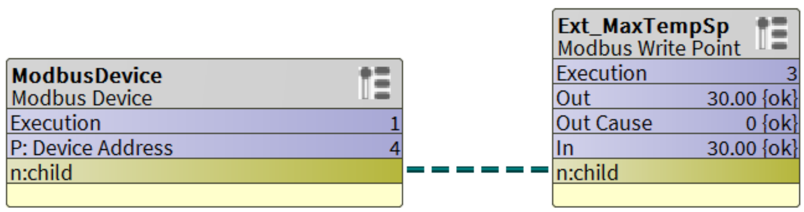


Fig. 494   Modbus Write Device Function Block

CHAPTER

20 DATA FUNCTION BLOCK

The following Date function blocks are available in the honIrmControl Palette that can be configured and used to create the required application logic:

- Alarm
- Counter

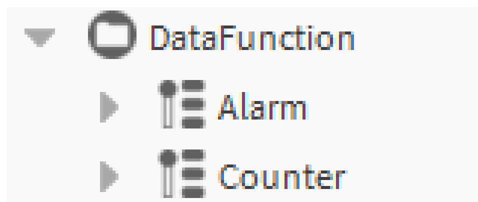


Fig. 495 Data Function Block

Alarm

The Alarm function generates an alarm based on the input's value in relation to the high and low limits. It is possible to create up to 32 alarm function blocks that map to nvoError.

The function block keeps track of the alarm status and delay timer from iteration to iteration. These are cleared upon power-up or reset. It is not required to connect the output of this function block to the input of another function block for this function block to function. (This is because a function block's output has no value if it is not connected.)

The Alarm Function Block is unique in that it also sets or resets a nvoError bit. When the Alarm Status value is set to "true," the configured bit in nvoError is set. When the Alarm Status is set to "false," the configured bit in nvoError is cleared.

nvoError: A multi-byte network variable provided by the engineering tool indicates errors. The nvoError map can be accessed via the controller's property sheet view. The nvoError map is made up of ten one-byte fields because each byte is 8 bits long. A maximum of 80 bits can be used to indicate errors. Each bit corresponds to an alarm.

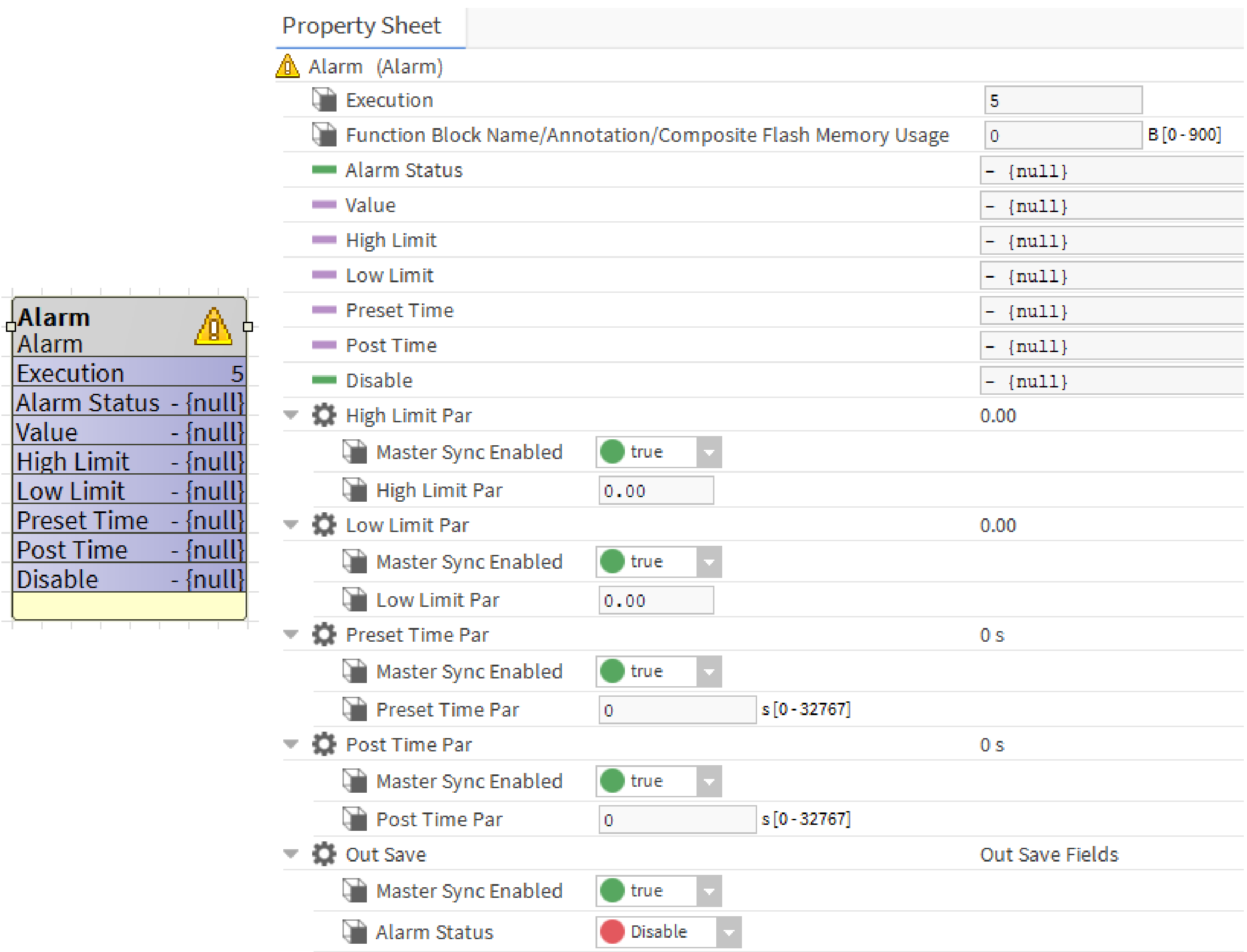


Fig. 496 Alarm Function Block and Property

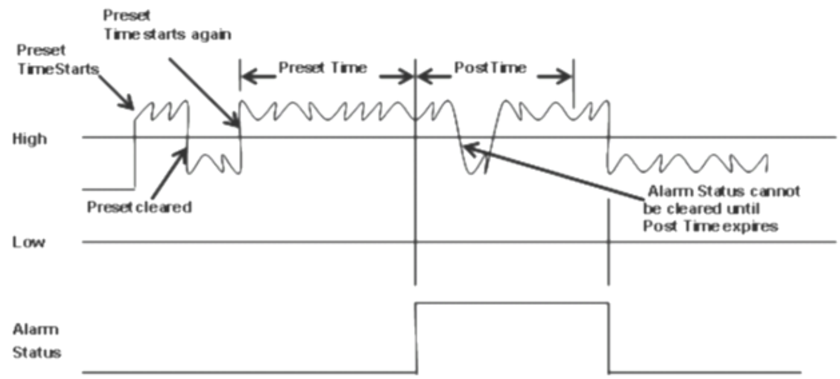


Fig. 497 Alarm Logic Diagram

Inputs

Table 315 Input of Alarm

Input Name	Description
Value	Input Value. **Numeric:** 32-Bit floating Point value >= -inf-<inf, Default: Null
High Limit	Input High Limit. **Numeric:** 32-Bit floating value >= -inf-<inf, Null
Low Limit	Input Low Limit. **Numeric:** 32-Bit floating value >= -inf-<inf, Null
Preset Time	Normal to Alarm delay (Sec). **Numeric:** 32-Bit numeric value (0-32767), Null
Post Time	Alarm lock on delay (Sec). **Numeric:** 32-Bit numeric value (0-32767), Null
Disable	Alarm lock on delay (Sec). **Numeric:** Binary value: 0-1, Null

Output

Table 316 Output of Alarm

Output Name	Description
Alarm Status	Alarm Status False (0), True (1)

Parameter

Table 317 Parameter of Alarm

Parameter Name	Description
High Limit Par	Input High Limit. **Numeric:** 32-Bit floating value: >=-inf-<inf, Default 0
Low Limit Par	Input Low Limit. **Numeric:** 32-Bit floating value: >=-inf-<inf, Default 0
Preset Time Par	Normal to Alarm delay (Sec) **Numeric:** 32-Bit numeric value: 0-32767, Default 0
Post Time Par	Alarm lock on delay (Sec). **Numeric:** 32-Bit numeric value: 0-32767, Default 0
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Alarm Status:** To enable or disable the **Alarm Status** feature.

Examples

The alarm status is "true" if the value is constantly greater than the High Limit or less than the Low Limit for the Preset Time. When you set the alarm to "true," it stays that way for at least the Post Time.If the value is still outside of the limits at the end of the Post Time, the alarm will continue to sound. The alarm status is set to "false" if the value is within limits and the Post Time has expired.

Example 1: Alarm (High Limit Alarm)

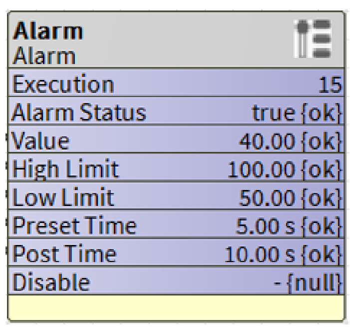


Fig. 498   Alarm - High Limit Alarm

Example 2: Alarm (Low Limit Alarm)

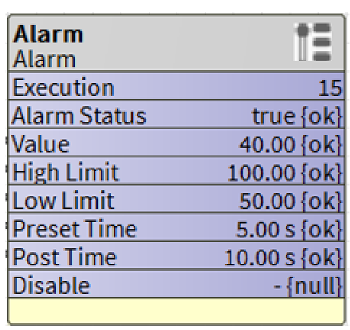


Fig. 499   Alarm - Low Limit Alarm

Counter

The Counter function block counts the input's leading-edge transitions.

- If the enable is "true" and the input changes from "false" to "true," the count value is incremented or decremented. Negative values reduce the number of items in the count.
- If preset is set to "true," count equals preset value. The function block keeps track of the previous state of the input from iteration to iteration so that it can detect a transition. This is cleared upon power-up or reset.

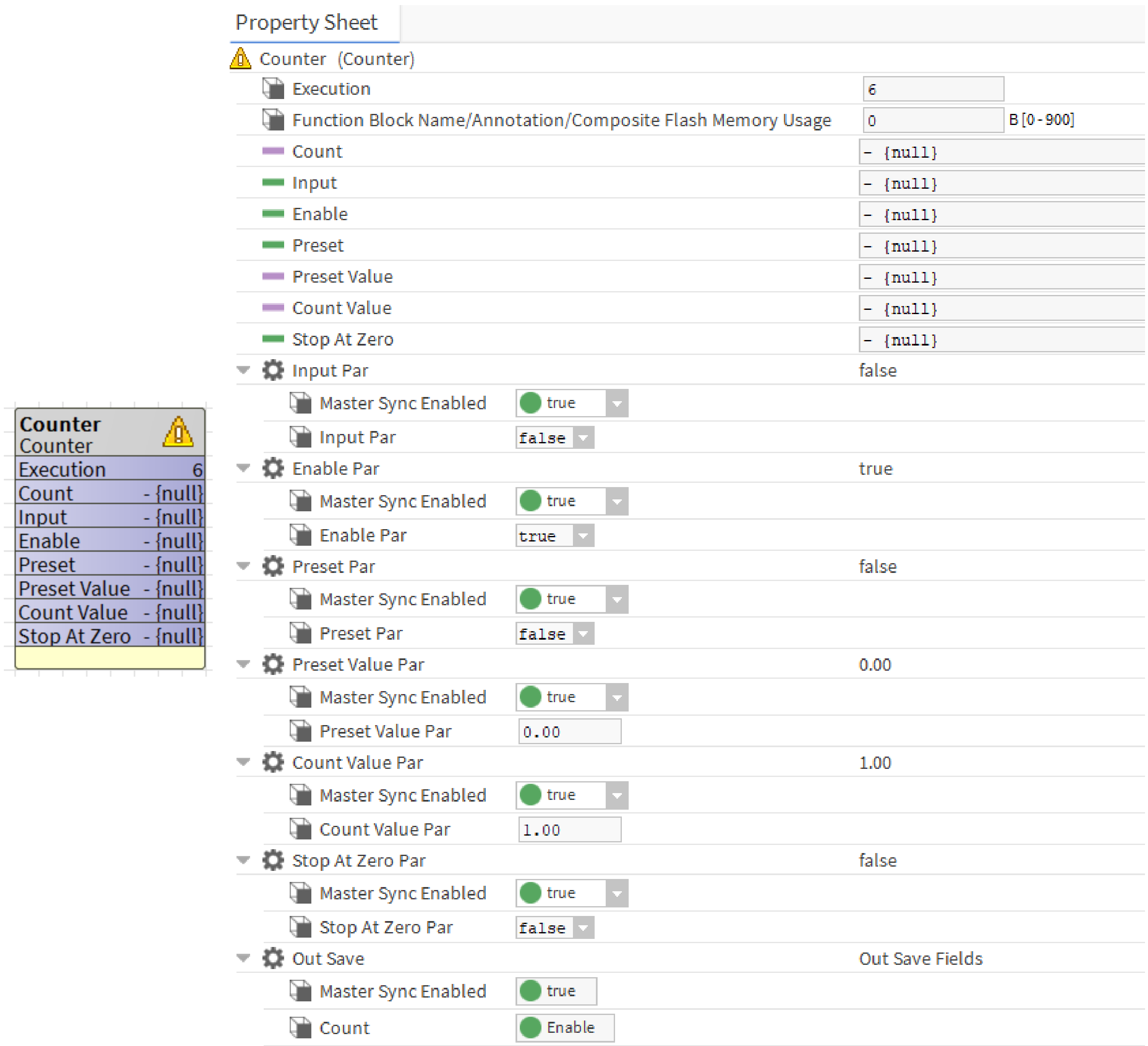


Fig. 500 Counter Function Block and Property Sheet

Inputs

Table 318 Inputs of Counter Function

Input Name	Description
Input	**Numeric:** Binary point value 0 - 1, **Default:** Null
Enable	**Numeric:** Binary value 0 - 1, **Default:** Null
Preset	Preset input will set the counter to preset value. **Numeric:** Binary value 0 - 1, **Default:** Null
Preset value	**Numeric:** 32 Bit Floating value >=-inf-<inf, **Default:** Null
Count value	This value will be used to increment or decrement the counter. Positive values are incremented, while negative values are decremented. **Numeric:** 32-Bit floating value >=-inf-<inf, **Default:** Null
Stop at Zero	When this input is enabled, the counter will stop counting. **Numeric:** Binary value 0 - 1, **Default:** Null

Output

Table 319 Output of Counter Function

Output Name	Description
Count	Counter value

Parameters

Table 320 Parameters of Counter Function

Parameter Name	Description
Input Par	**Numeric:** Binary point value 0 - 1, **Default:** Null.
Enable Par	**Numeric:** Binary value 0 - 1, **Default:** Null.
Preset Par	Preset input will set the counter to preset value. **Numeric:** Binary value 0 - 1, **Default:** Null.
Preset Value Par	**Numeric:** 32 Bit Floating value >=-inf-<inf, **Default:** Null.
Count Value Par	This value will be used to increment or decrement the counter. Positive values are incremented, while negative values are decremented. **Numeric:** 32-Bit floating value >=-inf-<inf, **Default:** Null.

Table 320 Parameters of Counter Function (Continued)

Parameter Name	Description
Stop At Zero Par	When you enable this input, the counter will stop counting. **Numeric:** Binary value: 0-1, **Default:** Null.
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Count:** To enable or disable the **Count** feature.

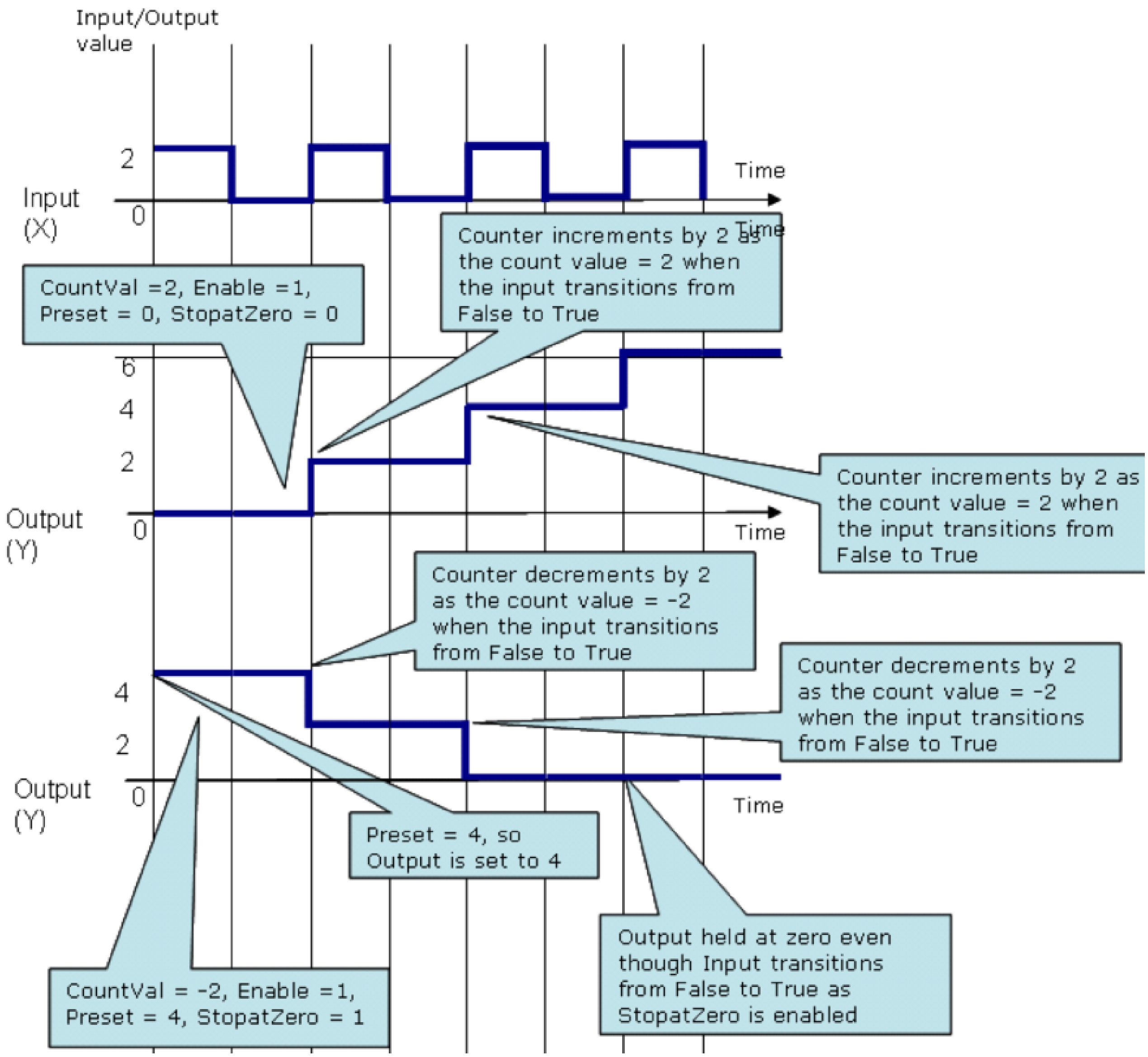


Fig. 501 Transition versus Time with Positive and Negative Count Values

Examples

Example 1: Counter (Incremental Counter)

Incremental counter with a pre-set value of 5 and a counter value of 1. The count value increased from 5 to 6.

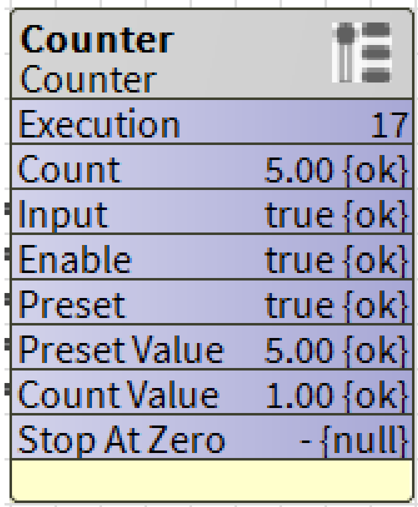


Fig. 502 Incremental Counter

Example 2: Counter (Decremental Counter)

Decremental counter with a predefined value of 5 and a counter value of -1. The count value dropped from 5 to 4. The counter value will be reset to zero.

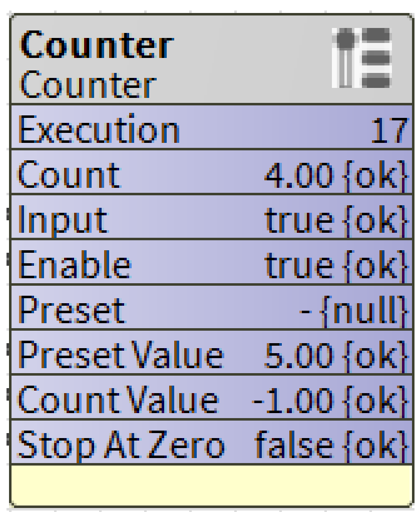


Fig. 503 Decremental Counter

CHAPTER

21 ZONE ARBITRATION FUNCTION BLOCKS

The honIrmControl Palette provides the following Zone Arbitration function blocks that can be configured and used to build the required application logic:

- General SetPoint Calculator
- Occupancy Arbitrator
- Set Temperature Mode
- Temperature Setpoint Calculator

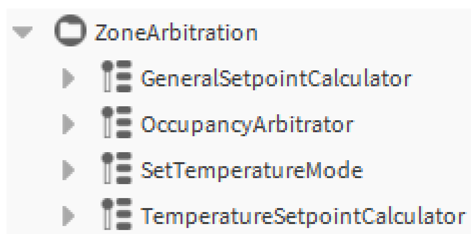


Fig. 504 Zone Arbitration Function Blocks

General SetPoint Calculator

The General Setpoint Calculator function calculates generic setpoints, including reset. It makes use of three configuration parameters: effective occupancy current, effective occupancy current, and effective occupancy current.

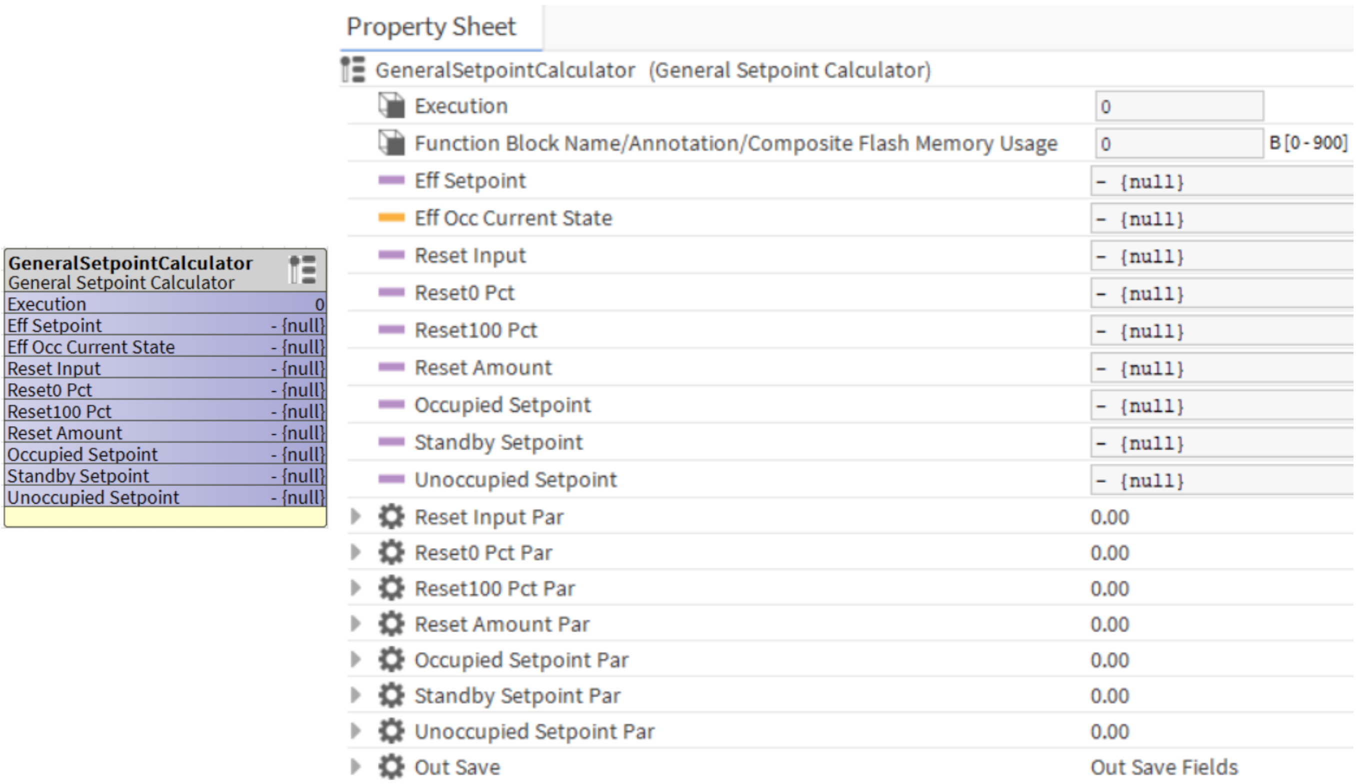
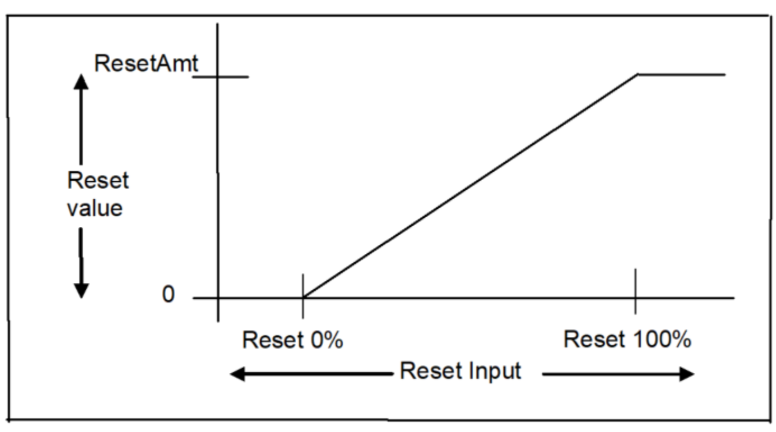


Fig. 505 General Setpoint Calculator Function Block and Property Sheet



Reset Calculation, positive amount, 0% < 100%

Fig. 506 Reset Calculation

Table 321   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

Input

Table 322 Inputs of General Setpoint Calculator

Input Name	Description
Eff Occ Current State	Effective occupied current state. **0:** Occupied **1:** Unoccupied **2:** Bypass **3:** Standby **255:** Null **Numeric:** 32 Bit Integer value 0-3, **Default:** Null.
Reset Input	Reset input to add offset on top of the occupied set point. The offset will be added based on the **reset amount**, **reset input**, reset 0 % and reset 100 %. **Numeric:** 32 Bit Floating value: 0 - 100, **Default:** Null.
Reset0 Pct	Reset begin value. **Numeric:** 32 Bit Floating value: >=-inf-<inf, **Default:** Null.
Reset100 Pct	Reset end value. **Numeric:** 32 Bit Floating value: >=-inf-<inf, **Default:** Null.
Reset Amount	**Numeric:** 32 Bit Floating value: >=-inf-<inf, **Default:** Null.
Occupied Setpoint	**Numeric:** 32 Bit Floating value: >=-inf-<inf, **Default:** Null.
Standby Setpoint	**Numeric:** 32 Bit Floating value: >=-inf-<inf, **Default:** Null.
Unoccupied Setpoint	**Numeric:** 32 Bit Floating value: >=-inf-<inf, **Default:** Null.

Output

Table 323 Output of General Setpoint Calculator

Output Name	Description
Eff Setpoint	Effective Setpoint. Any floating-point number.

Parameter

Table 324 Parameters of General Setpoint Calculator

Parameter Name	Description
Reset Input	Reset Input to add offset on top of the occupied set point. The offset will be added based on the **reset amount**, **reset input**, reset 0 % and reset 100 %. **Numeric:** 32 Bit Floating value: 0 - 100, **Default:** Null.
Reset0 Pct	Reset begin value. **Numeric:** 32 Bit Floating value: >=-inf-<inf, **Default:** Null.
Reset100 Pct	Reset end value. **Numeric:** 32 Bit Floating value: >=-inf-<inf, **Default:** Null.
Reset Amount	**Numeric:** 32 Bit Floating value: >=-inf-<inf, **Default:** Null.
Occupied Setpoint	**Numeric:** 32 Bit Floating value: >=-inf-<inf, **Default:** Null.
Standby Setpoint	**Numeric:** 32 Bit Floating value: >=-inf-<inf, **Default:** Null.
Unoccupied Setpoint	**Numeric:** 32 Bit Floating value: >=-inf-<inf, **Default:** Null.
Out save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Eff Setpoint:** To enable or disable the Effective Setpoint feature.

Examples

Example 1: General Setpoint Calculator

If **Effective Occupancy Current State** = 0 (such as set to an Occupied mode based on the scheduler), the **OccupiedSetPoint** is displayed under **EffectiveSetPoint**.
OccupiedSetPoint = 73.4 °F (23 °C), **StandBySetPoint** = 77 °F (25 °C),
UnOccupiedSetPoint = 80.6 °F (27 °C)

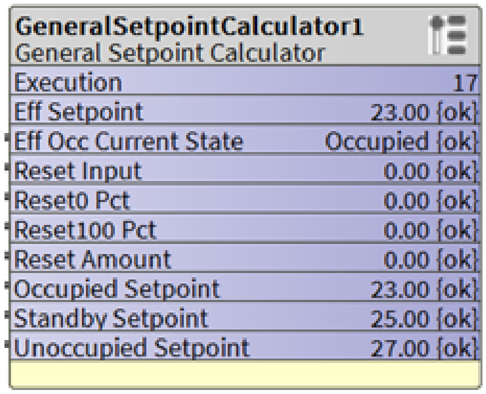


Fig. 507 General Setpoint Calculator

Example 2: General Setpoint Calculator

If **Effective Occupancy Current State** = 0 (such as set to an Occupied mode based on the scheduler), the **OccupiedSetPoint** is displayed under **EffectiveSetPoint**. See the impact of **ResetInput** /**Reset0Pct**, **Reset100Pct**, and **ResetAmount** during an Occupied Mode.

OccupiedSetPoint = 73.4 °F (23 °C), **StandBySetPoint** = 77 °F (25 °C),
UnOccupiedSetPoint = 80.6 °F (27 °C).

ResetInput = 41 °F (5 °C), **Reset** 0 % = 0, **Reset** 100 % = 100, **Reset Amount** = 212 °F (100 °C), Mode will be Occupied.

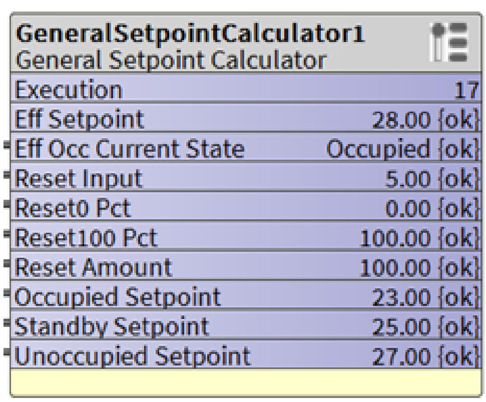


Fig. 508 General Setpoint Calculator

Example 3: General Setpoint Calculator

If **OccupiedSetPoint** = 73.4 °F (23 °C), **StandBySetPoint** = 77 °F (25 °C), **UnOccupiedSetPoint** = 80.6 °F (27 °C), **ResetInput** = 41 °F (5 °C), **Reset** 0 % = 0, Reset 100 % = 100, **Reset Amount** = 50 %, Eff Occ Mode will be Occupied.

In this case 50 % of **ResetInput** (i.e 41 °F (5 °C) * 50/100) = 36.5 °F (2.5 °C) is added to **OccupiedSetPoint** of 73.4 °F (23 °C).

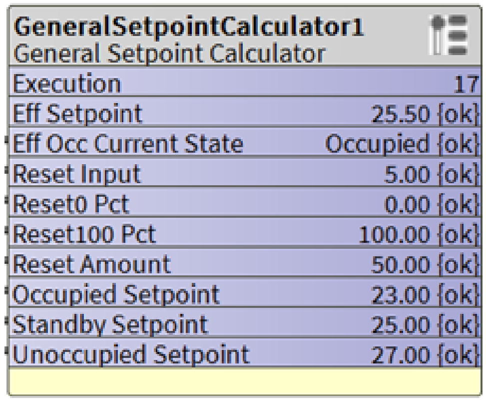


Fig. 509 General Setpoint Calculator

Example 4: General Setpoint Calculator

If **OccupiedSetPoint** = 73.4 °F (23 °C), **StandBySetPoint** = 77 °F (25 °C), **UnOccupiedSetPoint** = 80.6 °F (27 °C), **ResetInput** = 41 °F (5 °C), **Reset** 0 % = 0, **Reset** 100 % = 100, **Reset Amount** = -25 %, Mode will be Occupied.

In this case -25 % of **ResetInput** is deducted (i.e 41 °F (5 °C) * 25/100) = 29.7 °F (-1.25 °C) is subtracted from an **OccupiedSetPoint** of 73 °F (23 °C).

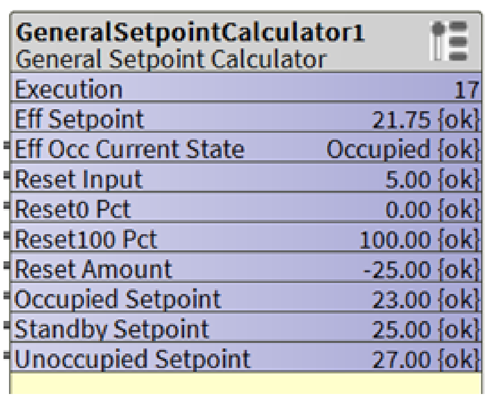


Fig. 510 General Setpoint Calculator Example 4

Example 5: General Setpoint Calculator

If **OccupiedSetPoint** = 73.4 °F (23 °C), **StandBySetPoint** = 77 °F (25 °C), **UnOccupiedSetPoint** = 80.6 °F (27 °C), **ResetInput** = 41 °F (5 °C), **Reset** 0 % = 0, **Reset** 100 % = 100, **Reset Amount** = -25 %, Mode will be Bypass.

In this case -25 % of **ResetInput** is deducted (i.e 41 °F (5 °C) * 25/100) = 29.7 °F (-1.25 °C) is subtracted from an **OccupiedSetPoint** of 73.4 °F (23 °C).

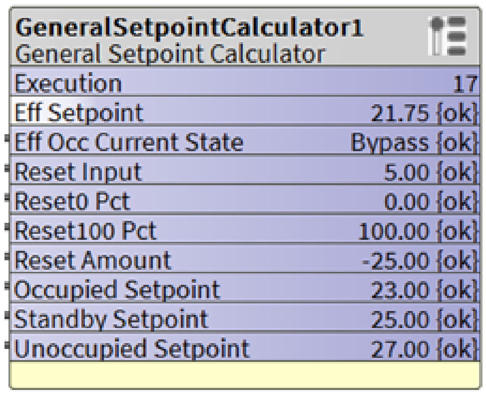


Fig. 511 General Setpoint Calculator Example 5

Example 6: General Setpoint Calculator

If **Effective Occupancy Current State** = 1 (such as, set to an UnOccupied mode based on the scheduler), the **UnOccupiedSetPoint** is displayed under **EffectiveSetPoint.** The **ResetInput** /**Reset0 Pct**, **Reset100Pct** doesn’t impact during an UnOccupied Mode.

OccupiedSetPoint= 73.4 °F (23 °C), **StandBySetPoint** = 77 °F (25 °C) and **UnOccupiedSetPoint** = 80.6 °F (27 °C).

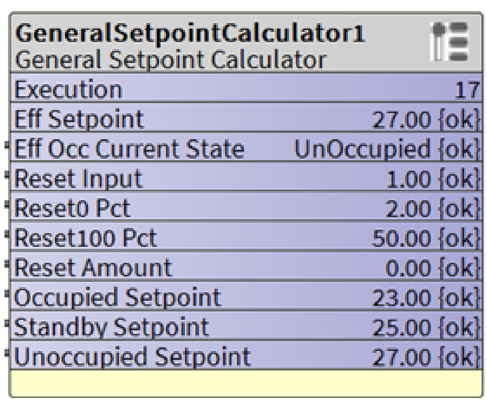


Fig. 512 General Setpoint Calculator Example 6

Occupancy Arbitrator

The Occupancy Arbitrator function computes the current Effective Occupancy State and the Manual Override State.

The value of Manual Override State is determined by the Manual Override Arbitration mechanism. This value is fed into the Occupancy Arbitrator. To evaluate the inputs, the Manual Override Arbitrator utilizes either a NetWins or a LastinWins scheme. NetWins indicates that the network command takes precedence over the wall module command at all times. The last override source is used to determine the final state with LastinWins.

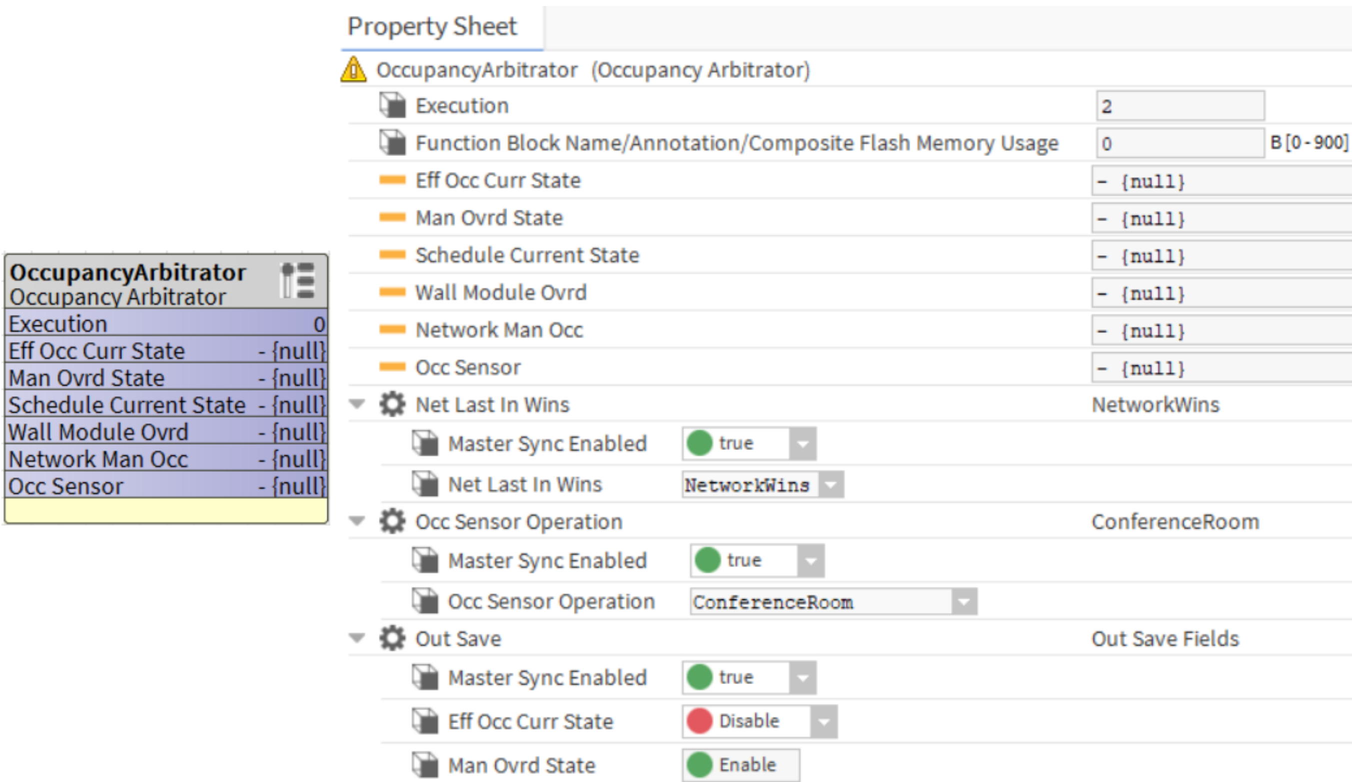


Fig. 513 Occupancy Arbitrator Function Block and Property Sheet

Input

Table 325 Input of Occupancy Arbitrator

Input Name	Description
Schedule Current State	Schedule current state. **0:** Occupied **1:** Unoccupied **3:** Standby **255:** Null **Numeric:** 32-Bit Integer value 0, 1, 3, 255. **Default:** Null

Table 325 Input of Occupancy Arbitrator (Continued)

Input Name	Description
Wall Module Ovrd	Wall module override. **0:** Occupied **1:** Unoccupied **2:** Bypass **3:** Standby **255:** Null **Numeric:** 32-Bit Integer value 0, 1, 2, 3, 255. **Default:** Null
Network Man Occ	Network manual occupancy. **0:** Occupied **1:** Unoccupied **2:** Bypass **3:** Standby **255:** Null **Numeric:** 32-Bit Integer value 0, 1, 2, 3, 255. **Default:** Null.
Occ Sensor	Occupancy sensor **0:** Occupied **1:** Unoccupied **255:** Null **Numeric:** 32-Bit Integer value 0, 1, 255. **Default:** Null.

Output

Table 326 Output of Occupancy Arbitrator

Output Name	Description
Eff Occ Current State	Effective Occupancy Current state. **0:** Occupied **1:** Unoccupied **2:** Bypass **3:** Standby
Manual Override State	Manual Override State. **0:** Occupied **1:** Unoccupied **2:** Bypass **3:** Standby **255:** Null

Parameter

Table 327 Parameters of Occupancy Arbitrator

Parameter Name	Description
Net Last In Wins	Network wins or Last In wins. **0:** Network Wins **1:** Last In Wins **Property Value:** Network Wins, Last Input Wins, **Default:** Network Wins.
Occ Sensor Operation	Occupancy sensor operation. **0:** Conference Room **1:** UnoccupiedCleaningCrew **2:** UnoccupiedTenant **Property Value:** Conference Room, UnoccupiedCleaningCrew, UnoccupiedTenant, **Default:** Conference Room.
Out Save	**Master Sync Enabled:** If you set it to “TRUE,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. **Eff Occ Curr State:** To enable or disable the Effective Occupancy Current State feature. **Man Ovrd State:** To enable or disable the Manual Override State feature.

Examples

Example 1: OccupancyArbitrator (Schedule Current State set to Occupied)

- Input Settings

NetWins/LastInWins Option = NetWins, NetworkManOcc = “null”

WMOverride = “null”, ScheduleCurrentState = Occupied. In this Scenario, the ScheduleCurrentState takes high priority to override the below output states.

- Output Settings

EffOccCurrState = OCCUPIED

ManOvrdState = “null”

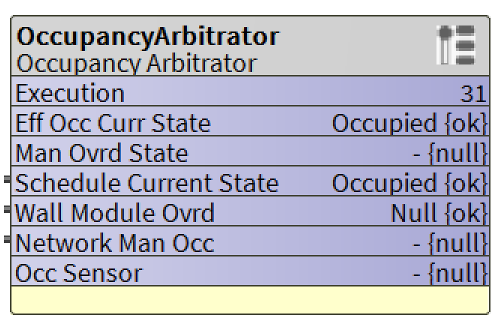


Fig. 514 Occupancy Arbitrator (EffOccCurrState = OCCUPIED)

Example 2: OccupancyArbitrator (Schedule Current State set to Occupied)

- Input Settings

NetWins/LastInWins Option = NetWins, NetworkManOcc = “null”

WMOverride = STANDBY ScheduleCurrentState = Occupied. In this Scenario, the WMOverride takes high priority to override the below output states.

- Output Settings

EffOccCurrState = STANDBY

ManOvrdState = STANDBY

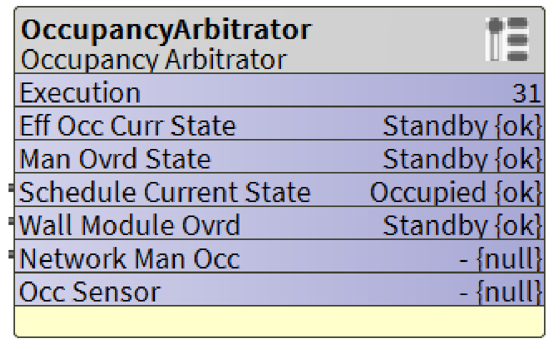


Fig. 515 Occupancy Arbitrator (EffOccCurrState = STANDBY)

Example 3: OccupancyArbitrator (Schedule Current State set to Unoccupied)

- Input Settings

NetWins/LastInWins Option = NetWins, NetworkManOcc = STANDBY

WMOverride = OCCUPIED ScheduleCurrentState = Unoccupied. In this Scenario, the NetworkManOcc takes high priority to override the below output states.

- Output Settings

EffOccCurrState = STANDBY

ManOvrdState = STANDBY

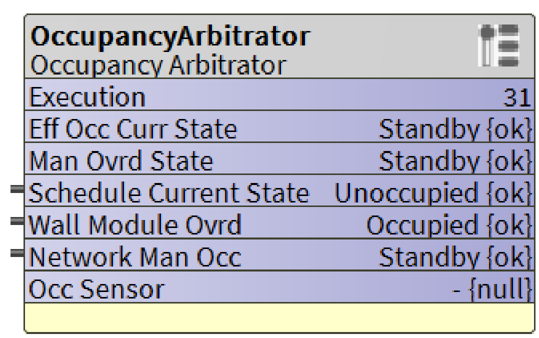


Fig. 516 Occupancy Arbitrator (EffOccCurrState = STANDBY, ManOvrdState = STANDBY)

Example 4: OccupancyArbitrator (Schedule Current State set to Unoccupied)

- Input Settings

NetWins/LastInWins Option = NetWins, NetworkManOcc = OCCNUL

WMOverride = OCCNUL, ManOvrdState = OCCNUL, ScheduleCurrentState = Unoccupied. In this Scenario, the NetworkManOcc takes high priority to override the below output states.

- Output Settings

EffOccCurrState = UnOccupied

ManOvrdState = “null”

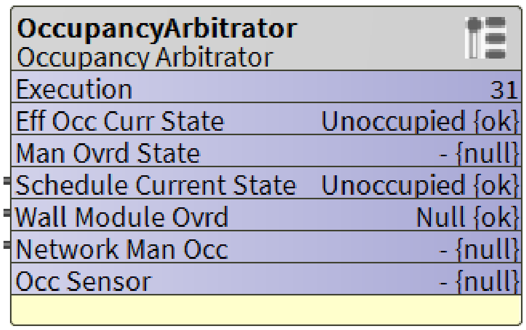


Fig. 517 Occupancy Arbitrator (EffOccCurrState = UnOccupied, ManOvrdState = “null”)

Table 328 Manual Override Arbitration

Last in Wins/ Net Wins	Network Man Occ (note2)	WM Override (note 2)	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.

Table 328 Manual Override Arbitration (Continued)

Last in Wins/ Net Wins	Network Man Occ (note2)	WM Override (note 2)	Manual Override State	Comment
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.
Net Wins	OCC	Don't Care	OCC	Result set to Network Man Occ.
Net Wins	UNOCC	Don't Care	UNOCC	Result set to Network Man Occ.
Net Wins	BYPASS	Don't Care	BYPASS	Result set to Network Man Occ.
Net Wins	STANDB Y	Don't Care	STANDBY	Result set to Network Man Occ.
Net Wins	OCCNUL	OCC	OCC	Result set to the wall module override.
Net Wins	OCCNUL	STANDB Y	STANDBY	Result set to the wall module override.
Net Wins	OCCNUL	BYPASS	BYPASS	Result set to the wall module override.
Net Wins	OCCNUL	UNOCC	UNOCC	Result set to the wall module override.
Net Wins	OCCNUL	OCCNUL	OCCNUL	Override canceled.

Note 1: *Any other input value not listed, is not a valid state. If received, it is treated as OCCNUL.*

Note 2: *For Last in Wins, the value in the above table was just changed from another state and this is the current state.*

Set Temperature Mode

The Set Temperature Mode function block automatically calculates the effective temperature control mode based on the control type, system switch setting, network mode command, temperature setpoints, supply temperature, and space temperature.

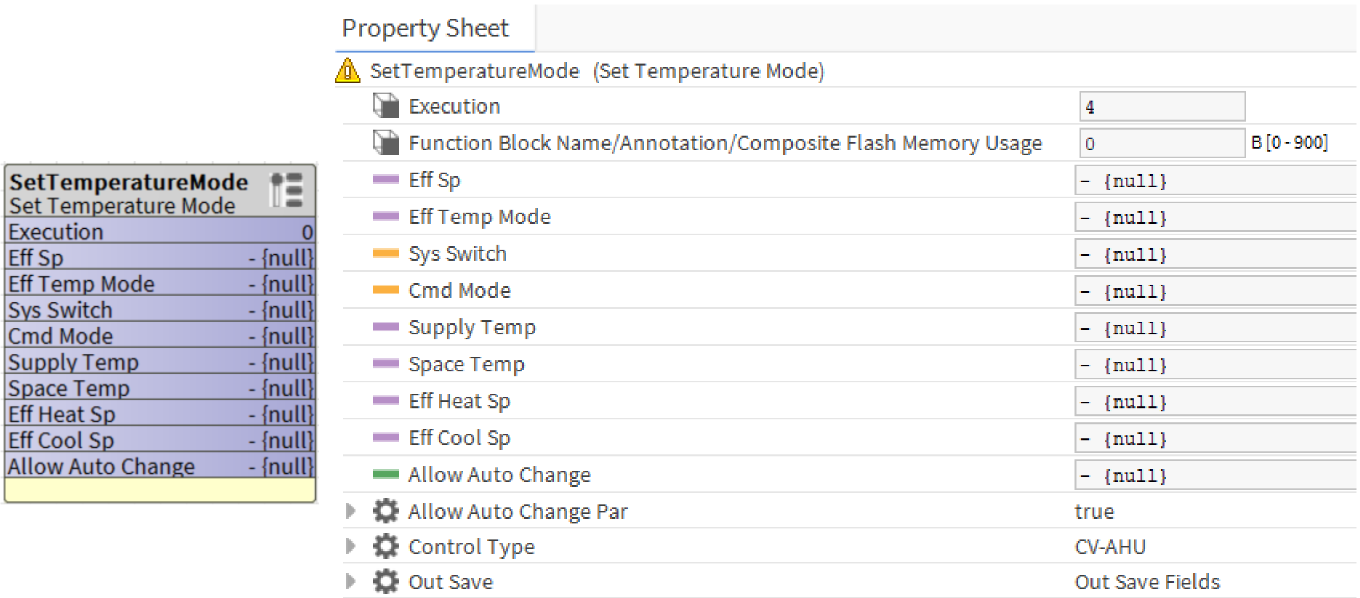


Fig. 518 SetTemperatureMode Function Block and Property Sheet

EffTempMode indicates the current Mode determined by input states and arbitrated by the control logic. The valid enumerated values have the following meanings:

Table 329 EffTempMode Description

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	A controller is commanded off.

Input

Table 330 Input of Set Temperature Mode

Input Name	Description
Sys Switch	Sys Switch input comes from the wall module. Users will be able to switch to Auto, Cool, Heat, Emergency Heat, or off modes. **0:** Auto **1:** Cool Mode **2:** Heat Mode **3:** Emergency Heat Mode. Emergency Heat, also known as “auxiliary heat,” is the second stage of heat that your thermostat runs on when the temperature is too cold for your heat pump to extract heat from the outside. Emergency Heat is triggered when it is 35°F and below outside. Emergency heat should be switched on by the users in emergencies when something happens to the first stage heat source, “heat pump.” **255:** Off Mode. **Numeric:** 32-Bit Integer value 0, 1, 3, 255. **Default:** Null.
Cmd Mode	Cmd Mode input comes from an operator. From the supervisor, an operator can switch to Auto, Heat, Cool, Emergency Heat, or off modes. **0:** Auto **1:** Heat Mode **2:** Cool Mode **3:** Off **4:** Emergency Heat Mode **255:** Off Mode. **Numeric:** 32-Bit Integer value 0, 1, 3, 4, 255. **Default:** Null.
Supply Temp	Supply Temp input is used in VAV applications to switch the temperature mode based on **supply temperature**. If the **supply temperature** is less than 70 deg F / 21 deg C, then the VAV box will switch to cool mode. Else if the **supply temperature** is greater than 75 deg F / 24 deg C, then the VAV box will switch to heat mode. Else VAV box will switch to cool, heat, reheat based on room temperature, effective heat, and cool setpoint. **Numeric:** 32-Bit, Floating value: -58 to 302 °F or -50 to 150 °C **Default:** Null.
Space Temp	Space temperature. **Numeric:** 32-Bit, Floating value: -58 to 302 °F or -50 to 150 °C **Default:** Null.
Eff Heat Sp	Effective heat set point. **Numeric:** 32-Bit, Floating value: -58 to 302 °F or -50 to 150 °C **Default:** Null

Table 330 Input of Set Temperature Mode (Continued)

Input Name	Description
Eff Cool Sp	Effective cool set point. **Numeric:** 32-Bit, Floating value: -58 to 302 °F or -50 to 150 °C **Default:** Null.
Allow Auto Change	Allow Auto Change is the binary input used to allow temperature mode change based on additional validations. The application will be written to make sure the cooling coil is completely closed before switching to heat mode or vice versa using this input. **0:** Don't Allow Auto Change **1:** Allow Auto Change. • If Allow Auto Change = 1, then switching between HEAT_MODE and COOL_MODE is possible. Must have a valid effHeatSP and effCoolSP. • If Allow Auto Change = 1 and effHeatSp > effCoolSp, then effHeatSp will be internally set to effCoolSP. **Numeric:** Binary value 0, 1, **Default:** Null.

Output

Table 331 Output of Set Temperature Mode

Output Name	Description
Eff Sp	Effective setpoint. It is a calculated setpoint derived by its temperature control modes. • If effTempMode = COOL_MODE then val = effCoolSetPt, else val=effHeatSetPt **Numeric:** 32-Bit floating point value, 0-255.
Eff Temp Mode	Effective temperature mode. **0:** Cool Mode Cool air is supplied from a network workstation or a node via the central air supply, and cooling energy is supplied to the controlled space. • **1:** Reheat Mode, cool air is supplied from a network workstation or a node via the central air supply. The air is reheated by a local Heat source. • **2:** Heat Mode, heated air is supplied network workstation or a node via the central air supply, and heated air is supplied to the controlled space. • **3:** Emergency Heat Mode, emergency heat is supplied network workstation or a node via the central air supply. **Numeric:** 32-Bit integer point value, 0 - 3.

Parameter

Table 332 Parameters of Set Temperature Mode

Parameter Name	Description
Allow Auto Change Par	Occupancy sensor **0:** Don't Allow Auto Change **1:** Allow Auto Change. • If Allow Auto Change = 1, then switching between HEAT_MODE and COOL_MODE is possible. Must have a valid effHeatSP and effCoolSP. • If Allow Auto Change = 1 and effHeatSp > effCoolSp, then effHeatSp will be internally set to effCoolSP. **Numeric:** Binary value 0, 1, **Default:** Allow Auto Change
Control Type	ControlType. • **0:** CVAHU (Constant Volume Air Handling Unit), Constant air volume (CVAHUV) is a type of heating, ventilating, and air-conditioning (HVAC) system. In a simple CVAHU system, the supply airflow rate is constant, but the supply air temperature is varied to meet the thermal loads of space. • **1:** VAV (VAV stands for variable air volume, it is a device used to control the air flow in the space. It shuts air supply to minimum flow once the temperature in a room is reached.) The basic difference between the CVAHU and VAV Boxes is that the CVAHU boxes are mainly used in small buildings as they are less expensive, whereas the VAV box provides variable air inside a building. The air temperature is constant and similar to what the control unit delivers. **Property Value:** CV-AHU, VAV, **Default:** CV-AHU
Out save	• **Master Sync Enabled:** If you set it to "TRUE," the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Eff Sp:** To enable or disable the **Effective Setpoint** feature. • **Eff Temp Mode:** To enable or disable the **Effective Temperature Mode** feature.

The CVAHU Arbitration logic for Control Type = 0 (CVAHU) is summarized in the table below.

Table 333 CVAHU Arbitration logic for Control Type = 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 ENUMERATION (254)	HEAT_MODE(2)
X	SS_COOL (1)	CMD_AUTO_MODE(0), CMD_NUL_MODE(255)	COOL_MODE (0)
X	SS_HEAT (2) or ENUMERATION(4) through ENUMERATION (254)	CMD_AUTO_MODE(0), CMD_NUL_MODE(255)	HEAT_MODE(2)
X	SS_EMERGENCY_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)
INVALID	SS_AUTO (0), invalid, unconnected, or a non-listed enumeration.	CMD_AUTO_MODE(0), CMD_NUL_MODE(255)	HEAT_MODE(2)
VALID	SS_AUTO (0), invalid, unconnected, or a non-listed enumeration.	CMD_AUTO_MODE(0), CMD_NUL_MODE(255),	COOL_MODE(0) or HEAT_MODE(2) (refer to below note)

X means Don't Care.

Note: *If Allow Auto Change = 1, then switching between HEAT_MODE and COOL_MODE is possible. Must have a valid effHeatSP and effCoolSP.*
If Allow Auto Change = 1 and effHeatSp > effCoolSp, then effHeatSp will be internally set to effCoolSP.

The VAV Mode arbitration logic for controlType = 1 (VAV) is summarized in the table below:

Table 334 VAV Mode arbitration logic for controlType = 1 (VAV)

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	<21 DEG C <70.0 DEG F	CMD_AUTO_MODE (0), CMD_HEAT_MODE (1), CMD_NUL_MODE (255)	COOL_MODE (0) or REHEAT_MODE (1) (Refer note 1)
Valid	X	<21 DEG C < 70 DEG F	CMD_AUTO_MODE (0), CMD_HEAT_MODE (1), CMD_NUL_MODE (255)	COOL_MODE(0)
Valid	X	21 TO 24 DEG C 70.0 To 75.0 DEG F	CMD_AUTO_MODE (0), CMD_HEAT_MODE (1), CMD_COOL_MODE (2), CMD_NUL_MODE (255)	COOL_MODE (0), REHEAT_MODE (1), HEAT_MODE (2) (Refer note 1 for transition between cool mode and reheat mode)
Valid	X	> 24 DEG C > 75 DEG F	CMD_AUTO_MODE (0), CMD_HEAT_MODE (1), 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), CMD_NUL_MODE (255)	COOL_MODE(0)
Valid	SS_HEAT (2)	Invalid or unconnected	CMD_AUTO_MODE (0), CMD_NUL_MODE (255)	HEAT_MODE(2)
Valid SS_	EMERGENCY_HEAT (3)	Invalid or unconnected	CMD_AUTO_MODE (0), CMD_NUL_MODE (255)	HEAT_MODE(2)
Valid	SS_OFF (255)	Invalid or unconnected	CMD_AUTO_MODE (0), CMD_NUL_MODE (255)	OFF_MODE(255)

Table 334 VAV Mode arbitration logic for controlType = 1 (VAV) (Continued)

Space Temp	sysSwitch	Supply Temp	cmdMode	effTempMode
Valid	SS_AUTO (0), invalid, unconnected, or a non-listed enumeration.	Invalid or unconnected	CMD_AUTO_MODE (0), CMD_NUL_MODE (255),	COOL_MODE(0) or REHEAT_MODE (1) (refer note 1)
Invalid	SS_AUTO (0), invalid, unconnected, or a non-listed enumeration.	Invalid or unconnected	CMD_AUTO_MODE (0), CMD_NUL_MODE (255),	COOL_MODE(0)

X means Don’t Care

Note: *If Allow Auto Change = 1, then switching between REHEAT_MODE and COOL_MODE is possible. Must have a valid effHeatSP and effCoolSP. If in cool mode and* spacetemp < effheat *setpt and space temp < effcoolsetpt – 1.0 then go to reheat mode.*

Note: *If in reheat mode and spacetemp > effCoolSetpt and spacetemp > effHeatsetpt + 1.0 then go to cool mode.*

Examples

Example 1: Set Temperature Mode

If **SysSwitch** = Auto, **ControlType** = 1 (VAV), **CmdMode** = Auto, **SupplyTemp** = 68 °F (20 °C), **SpaceTemp** = 69.8 °F (21 °C) , **EffHeatSp** = 68 °F (20 °C), **EffCoolSp** = 75.2 °F (24 °C) , **EffectiveSp** will be **EffCoolSp** and **effTempMode** is COOL such as If **SupplyTemp** <69.8 °F (21 °C), **effTempMode** will be On Cool Mode.

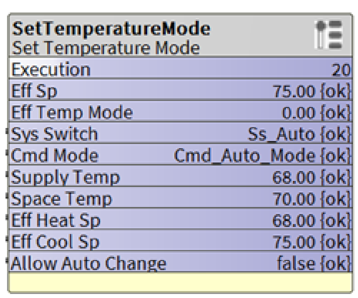


Fig. 519 Set Temperature Mode

Example 2: Set Temperature Mode

If **SysSwitch** = Auto, **ControlType** = 1(VAV), **CmdMode** = Auto, **SupplyTemp** = 68 °F (20 °C), **SpaceTemp** = 69.8 °F (21 °C), **EffHeatSp** = 68 °F (20 °C), **EffCoolSp** = 75.2 °F (24 °C), **EffectiveSp** will be **EffCoolSp** and **effTempMode** is HEAT such as If **SupplyTemp** >75.2 °F (24 °C), **effTempMode** will be On Heat Mode.

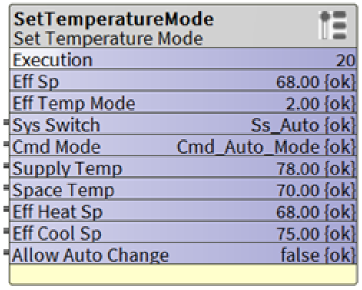


Fig. 520 Set Temperature Mode

Example 3: Set Temperature Mode

If **SysSwitch** =Sscool, **ControlType**=1 (VAV), **CmdMode**=Auto, **SupplyTemp**=59.9 °F (15.5 °C), **SpaceTemp** = 55.6 °F (13 °C), **EffHeatSp** = 68 °F (20 °C), **EffCoolSp** =75.2 °F (24 °C) **EffectiveSp** will be **EffCoolSp** and **effTempMode** is COOL such as If SupplyTemp >75 °F (24 °C), effTempMode will be On Heat Mode.

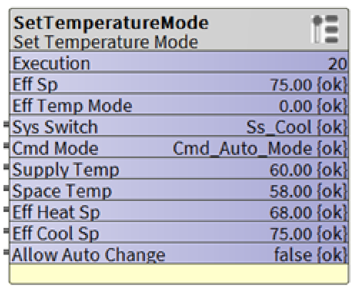


Fig. 521 Set Temperature Mode

Example 4: Set Temperature Mode

If **SysSwitch**=Sscool, **ControlType**=1 (VAV), **CmdMode**=Auto, **SupplyTemp** = 80.6 °F (27°C), **SpaceTemp**=59.9 °F (15.5 °C), **EffHeatSp**=68 °F (20 °C), **EffCoolSp**=75.2 °F (24 °C), **EffectiveSp** will be **EffHeatSp** and **effTempMode** is HEAT such as If SupplyTemp >75, then effTempMode will be On Heat Mode.

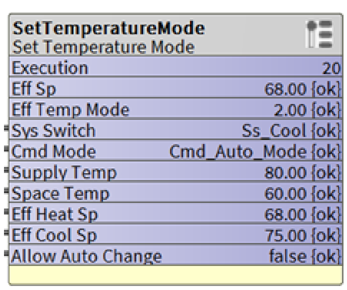


Fig. 522 Set Temperature Mode

Note: *If the system is in COOL Mode and SpaceTemp < effheatSp & SpaceTemp < effcoolsp – 1, system goes to reheat mode. If the system is in Reheat Mode and SpaceTemp > effcoolsp & SpaceTemp > effheatSp + - 1 system goes to cool mode*

Temperature Setpoint Calculator

The Temperature Setpoint Calculator function block calculates the current Effective Heat and Effective Cool setpoints based on the current schedule, occupancy override, and intelligent recovery information.

The user specifies the six setpoints. The six temperature setpoints are either defaulted by the Management Tool to be the first 6 Public variables under Control non-volatile and are not shown as inputs to the block, or the Tool will provide an extension block and show the six setpoints as inputs, depending on the setting of the UserAssignedSPs config.

The orders are Occupied Cool, Standby Cool, Unoccupied Cool, Occupied Heat, Standby Heat, and Unoccupied Heat. When the default setting is used, the user may have more than one TempSPCalc Function Block with this configuration; however, all blocks with this configuration will use the same TempSetpoints mapped to the same 6 Public variables. If the block is configured with the six setpoints in an extension block, each block can have its own setpoints.

Schedule Next state and TUNCOS

The adaptive recovery algorithm makes use of these parameters to recover the heating and cooling setpoints from their unoccupied values. If the user enters a rate of 1°/Hr and there is more than a 192° difference between the OCC and UNOCC setpoints, the algorithm will recover immediately when switching to UNOCC. The maximum TUNCOS is 11520 mins times 1 ° /Hr = 192 ° maximum delta.

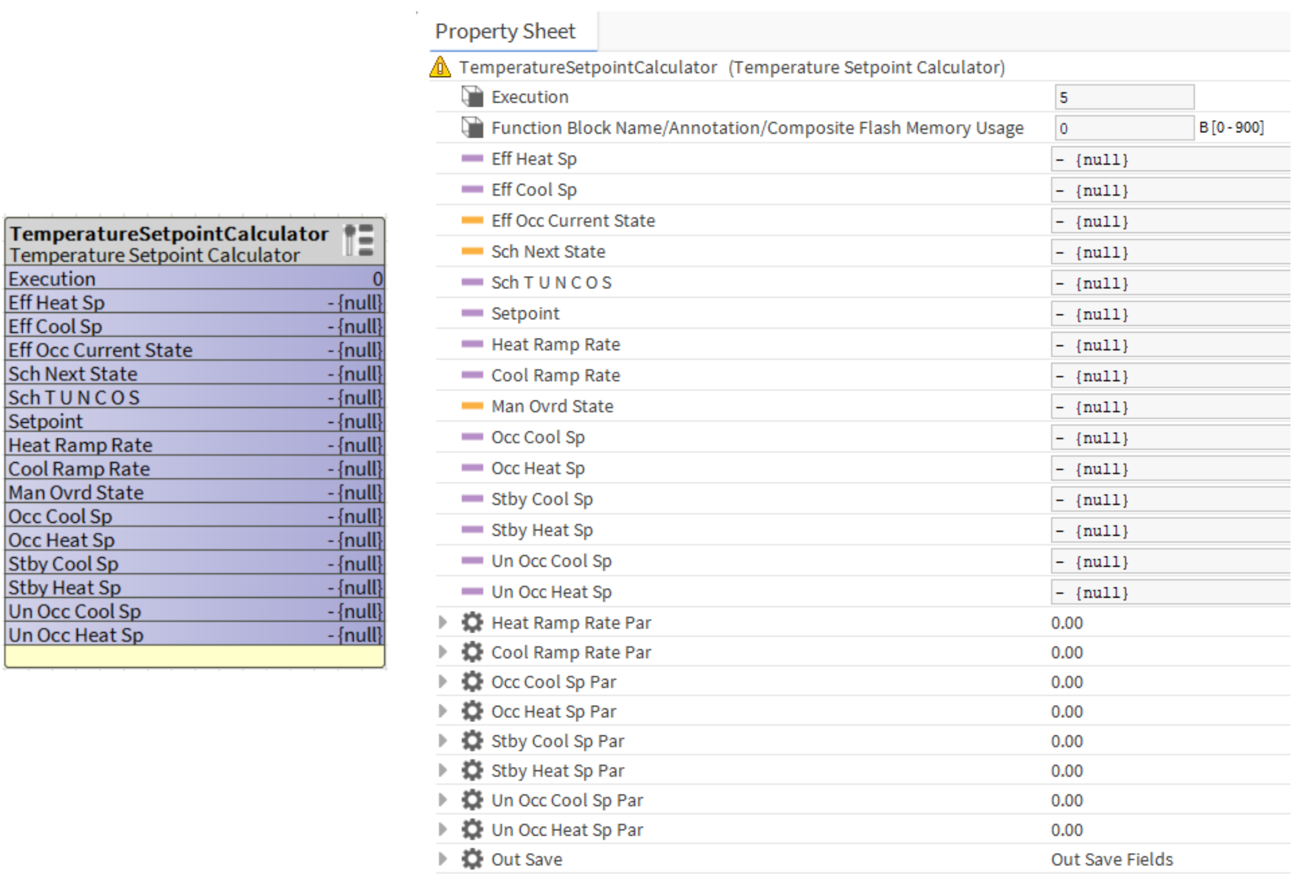


Fig. 523 TemperatureSetPointCalculator Function Block and Property Sheet

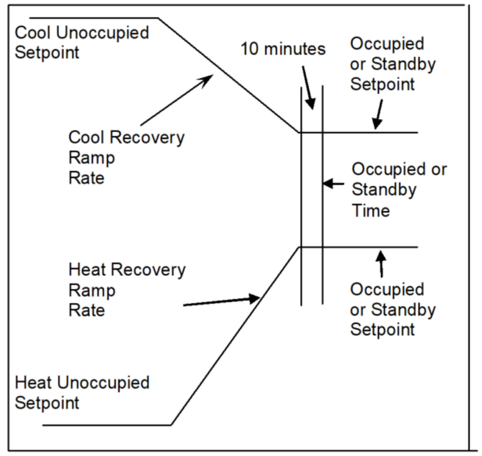


Fig. 524 Heat Cool Recovery Ramps

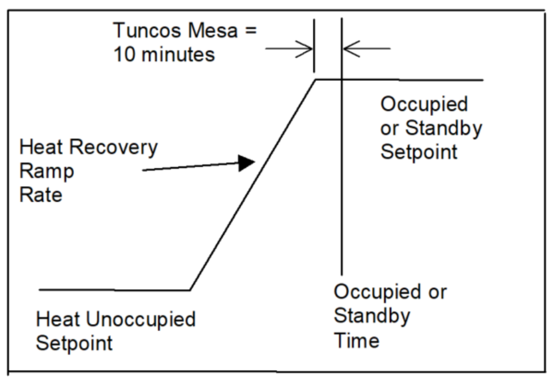


Fig. 525 TUNCOS Mesa Heat Recovery Ramp

Input

Table 335 Inputs of Temperature Setpoint Calculator

Input Name	Description
Eff Heat Sp	Effective heat set point. **Numeric:** 32-Bit Floating value>=-inf-<inf.
Eff Cool Sp	Effective cool set point. **Numeric:** 32-Bit Floating value >=-inf-<inf.
Eff Occ Current State	Effective occupancy current state. **0:** Occupied **1:** Unoccupied **2:** Bypass **3:** Standby. **Numeric:** 32-Bit Integer value 0, 1, 2, 3, 255, **Default:** Null.

Table 335 Inputs of Temperature Setpoint Calculator (Continued)

Input Name	Description
Sch Next State	Schedule next state. **0:** Occupied **1:** Unoccupied **3:** Standby **255:** Null **Numeric:** 32-Bit Integer value 0, 1, 3, 255, **Default:** Null.
Sch T UN C O S	Schedule time until next change of state (min). **Numeric:** 32-Bit Floating value 1-11520. **Default:** Null.
Setpoint	**Numeric:** 32-Bit Floating value >=-inf-<inf. **Default:** Null.
Heat Ramp Rate	**Numeric:** 32-Bit Floating value 0-<inf. **Default:** Null.
Cool Ramp Rate	Cool ramp rate. **Numeric:** 32-Bit Floating value 0-<inf. **Default:** Null.
Man Ovrd State	Manual override state. **0:** Occupied **1:** Unoccupied **2:** Bypass **3:** Standby **255:** Null **Numeric**: Binary value 0-3, 255 **Default:** Null.
Occ Cool Sp	Occupied cool setpoint **Numeric:** 32-Bit Floating value >=-inf-<inf. **Default:** Null.
Occ Heat Sp	Occupied heat setpoint **Numeric:** 32-Bit Floating value >=-inf-<inf. **Default:** Null.
Stby Cool Sp	Standby cool setpoint **Numeric:** 32-Bit Floating value >=-inf-<inf. **Default:** Null.
Stby Heat Sp	Standby heat setpoint **Numeric:** 32-Bit Floating value >=-inf-<inf. **Default:** Null.
Un Occ Cool Sp	UnOccupied heat setpoint **Numeric:** 32-Bit Floating value >=-inf-<inf. **Default:** Null.
Un Occ Heat Sp	UnOccupied heat setpoint **Numeric:** 32-Bit Floating value >=-inf-<inf. **Default:** Null.

Output

Table 336 Output of Temperature Setpoint Calculator

Output Name	Description
Eff Heat Sp	Effective heat set point. **Numeric:** 32-Bit Floating value>=-inf-<inf.
Eff Cool Sp	Effective cool set point. **Numeric:** 32-Bit Floating value >=-inf-<inf.
Eff Occ Current State	Effective occupancy current state. **0:** Occupied **1:** Unoccupied **2:** Bypass **3:** Standby **Numeric:** 32-Bit Integer value 0, 1, 2, 3, 255, **Default:** Null.

Parameter

Table 337 Parameter of Temperature Setpoint Calculator

Parameter Name	Description
Heat Ramp Rate (In & Par)	**Numeric:** 32-Bit Floating value 0-<inf. **Default:** Null
Cool Ramp Rate (In & Par)	**Numeric:** 32-Bit Floating value 0-<inf. **Default:** Null
Occ Cool Sp (In & Par)	Occupied cool setpoint **Numeric:** 32-Bit Floating value >=-inf-<inf. **Default:** Null.
Occ Heat Sp (In & Par)	Occupied heat setpoint **Numeric:** 32-Bit Floating value >=-inf-<inf. **Default:** Null.
Stby Cool Sp (In & Par)	Standby cool setpoint **Numeric:** 32-Bit Floating value >=-inf-<inf. **Default:** Null.
Stby Heat Sp (In & Par)	Standby heat setpoint **Numeric:** 32-Bit Floating value >=-inf-<inf. **Default:** Null.
Un Occ Cool Sp (In & Par)	UnOccupied heat setpoint **Numeric:** 32-Bit Floating value >=-inf-<inf. **Default:** Null.
Un Occ Heat Sp (In & Par)	UnOccupied heat setpoint **Numeric:** 32-Bit Floating value >=-inf-<inf. **Default:** Null.

Table 337 Parameter of Temperature Setpoint Calculator (Continued)

Parameter Name	Description
Out Save	• **Master Sync Enabled:** If you set it to “TRUE,” the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **Eff Heat Sp:** To enable or disable the Effective Heat Setpoint feature. • **Eff Cool Sp:** To enable or disable the Effective Cool Setpoint feature.

Examples

Example 1: Temperature Setpoint Calculator

If **OccCoolSp** = 23, **StbyCoolSp** = 25, **UnOccCoolSp** = 27, **OccHeatSp** = 21, **StbyHeatSp** = 19, **UnOccHeatSp** = 17, Man Override State is **Occupied** = 0, **Sch NextState** = Occupied.

Setpoint is 0, Thus the same **OccCoolSp** is transferred to an **EffectiveCoolSp** & **OccHEATSp** is transferred to an **EffectiveHeatSp.**

When **Setpoint** < 10, the **Setpoint** acts in “**offset**” mode (such as “relative” Setpoint). The Setpoint adjusts the programmed occupied, **StbyHeatSp** and **StbyCoolSp** up and down by the amount of the **Setpoint**. The user must ensure the input range is less than +10 for the offset setpoint to be used. The **Setpoint** doesn’t affect the **unoccupied setpoint**s. During bypass, the **occupied setpoint**s are adjusted. The offset is zero if the **Setpoint** is not connected or the sensor has failed. The user must ensure consistent units. If the **Setpoint** is in degrees F, the programmed setpoints should also be in degrees F.

The effective space temperature setpoints will be calculated based on the following formula:

- Base Setpoint = setpoint from wall module.
- Effective OccCoolSp = Base Setpoint + (**OccCoolSp** - **OccHeatSp**)/2
- Effective OccHeatSp = Base Setpoint - (**OccCoolSp** - **OccHeatSp**)/2

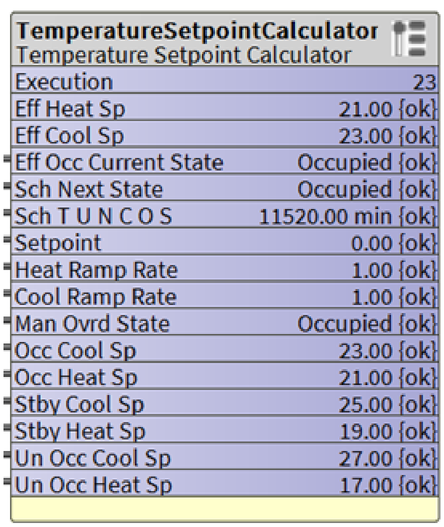


Fig. 526 Temperature Setpoint Calculator

Example 2: Temperature Setpoint Calculator

If **Setpoint** = 5 (such as RelativeSp), thus the same **OccCoolSp** + 5 is transferred to an **EffectiveCoolSp** & **OccHEATSp** + 5 is transferred to an **EffectiveHeatSp**.

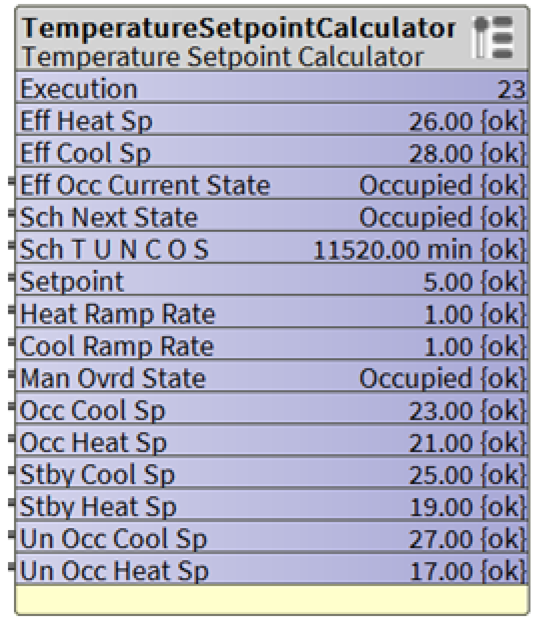


Fig. 527 Temperature Setpoint Calculator

Example 3: Temperature Setpoint Calculator

If **OccCoolSp** = 23, **StbyCoolSp**=25, **UnOccCoolSp** = 27, **OccHeatSp** = 21, **StbyHeatSp**=19, **UnOccHeatSp** =17, **ManOvrdState** is Occupied = 0.

SchNextState = Occupied

Setpoint = 12 (when it is more than 12, it acts as an absolute Sp).

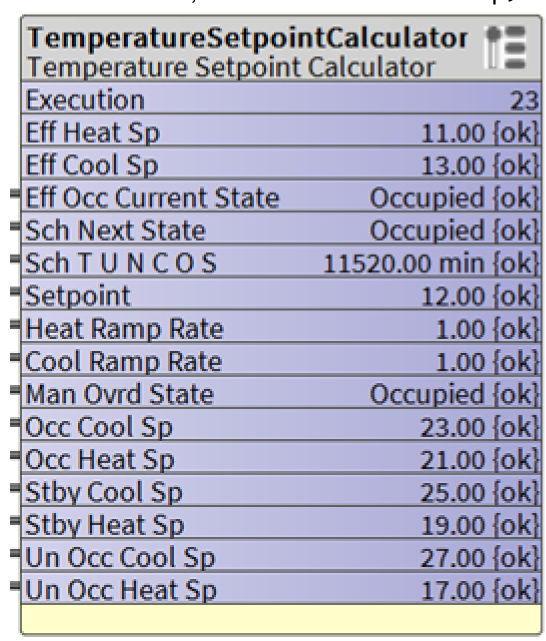


Fig. 528 Temperature Setpoint Calculator

Example 4: Temperature Setpoint Calculator

If **OccCoolSp** = 23, **StbyCoolSp** = 25, **UnOccCoolSp** = 27, **OccHeatSp** = 21, **StbyHeatSp** = 19, **UnOccHeatSp** = 17, **ManOvrdState** is UnOccupied = 1,

SchNextState = Occupied, **ManOvrdState** doesn’t affect the effective occupancy state.

Setpoint = 5 (such as Realative Sp), thus the same **OccCoolSp** + 5 is transferred to an **EffectiveCoolSp** & **OccHEATSp** + 5 is transferred to an **EffectiveHeatSp**.

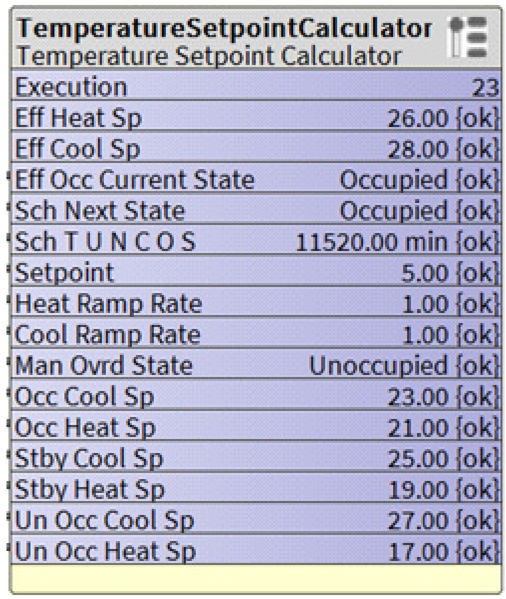


Fig. 529 Temperature Setpoint Calculator

CHAPTER

22 SYLK DEVICE AND PARAMETERS

This chapter describes the Sylk devices and parameters available in the honIrmControl Palette that can be configured and used to build an application. Also, provide information on the status of Sylk component behaviors.

- Sylk Parameters Description
- Sylk Component Status Behaviors

The following is a list of Sylk parameters that Sylk devices support. You can configure the Sylk device parameters based on your requirement.

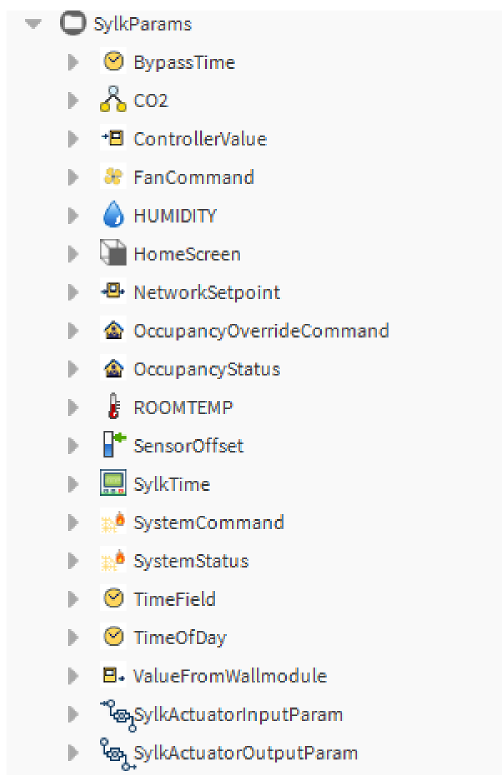


Fig. 530 Sylk Parameters

Sylk Parameters Description

Sylk Parameters are divided into three categories:

- Sylk In Parameters
- Sylk Out Parameters
- Sylk In Out Parameters

Sylk In Parameters

These parameters act as input to the Sylk devices.

Table 338 Sylk In Parameters

Properties	Description
ControllerValue	Value from the controller to the wall module.
OccupancyStatus	Occupancy status in the wall module.
SystemStatus	System status in the wall module.
TimeOfDay	Time of the day to show on Sylk device.
Sylk Actuator Input	Provides input value to the Sylk actuator .

Sylk Out Parameters

These parameters capture input from various sensors and act as output for the Sylk device.

Table 339 Sylk Out Parameters

Properties	Description
CO2	CO2 concentration in the space.
HUMIDITY	Percent humidity of the space.
OccupancyOverrideCommand	Override the unoccupancy mode to occupied mode.
ROOMTEMP	Temperature of the space.
SensorOffset	Provide the sensor offset to the wall module.
ValueFromWallModule	Provide the value from the wall module to the controller.
Sylk Actuator Output	Provides output from the Sylk actuator to the controller.

Sylk In Out Parameters

These parameters are used as input as well as output for the Sylk device.

Table 340 Sylk In Out Parameters

Properties	Description
BypassTime	Provide the bypass time to a wall module.
FanCommand	Provide commands to the fan from the wall module.

Table 340 Sylk In Out Parameters (Continued)

Properties	Description
NetworkSetpoint	Provide the setpoints to the wall module.
SystemCommand	Override the system command from the wall module.
TimeField	Configure the time format of the wall module.

Sylk Device Supported Sylk Parameters

This section provides details of Sylk parameters supported by the Sylk device.

Table 341 TR120x

Parameter	TR120x SBus Wall Modules	
	TR120	TR120H
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
EnumSchedule	Y	Y

Table 342 TR7x

Parameter	TR7x SBus Wall Modules			
	TR75H	TR75	TR71H	TR71
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
EnumSchedule	Y	Y	N	N

Table 343 TR42x

Parameter	TR42x SBus Wall Modules			
	TR42	TR42H	TR42CO2	TR42HCO2
ROOMTEMP	Y	Y	Y	Y
HUMIDITY	N	Y	N	Y
CO2	N	N	Y	Y
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 344 TR40x

Parameter	TR40x SBus Wall Modules			
	TR40	TR40H	TR40CO2	TR40HCO2
ROOMTEMP	Y	Y	Y	Y
HUMIDITY	N	Y	N	Y
CO2	N	N	Y	Y

Table 345 C7400S (Zeleny)

Parameter	(Zeleny)
ROOMTEMP	Y
HUMIDITY	Y

Sylk Component Status Behaviors

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

Table 346 Sylk Status Type and Description

Type	Description
alarm	Point currently has a value in an alarm range, as defined by property in its alarm extension.
fault	Originates from a proxy point only. Typically indicates a configuration or licensing error. If it occurs after a normal operation, it may indicate a “native fault” in the device, or the point’s parent device has a fault status.
overridden	Current point control is from an action, meaning a user-invoked command at either priority level 8 (override) or priority 1 (emergency).
disabled	Originates from a proxy point only. Point (or its parent device or network) has been manually disabled (property enabled = false).
down	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	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	Current point control has entered a null state vs. a specific value and priority level. Typical to fallback operation for a writable point. Note: If linking a null status Out to a simple data slot, the point’s null value is processed.
unackedAlarm	Last point alarm event has not yet received user acknowledgment. Point’s alarm extension uses alarm class requiring acknowledgment

CHAPTER 23 SYLK IN PARAMETERS

The Sylk device receives input from the following parameters:

- Controller Value
- Occupancy Status
- System Status
- Time of Day
- Sylk Actuator Input

Controller Value

This Controller Value parameter displays the value from the controller to the wall module.

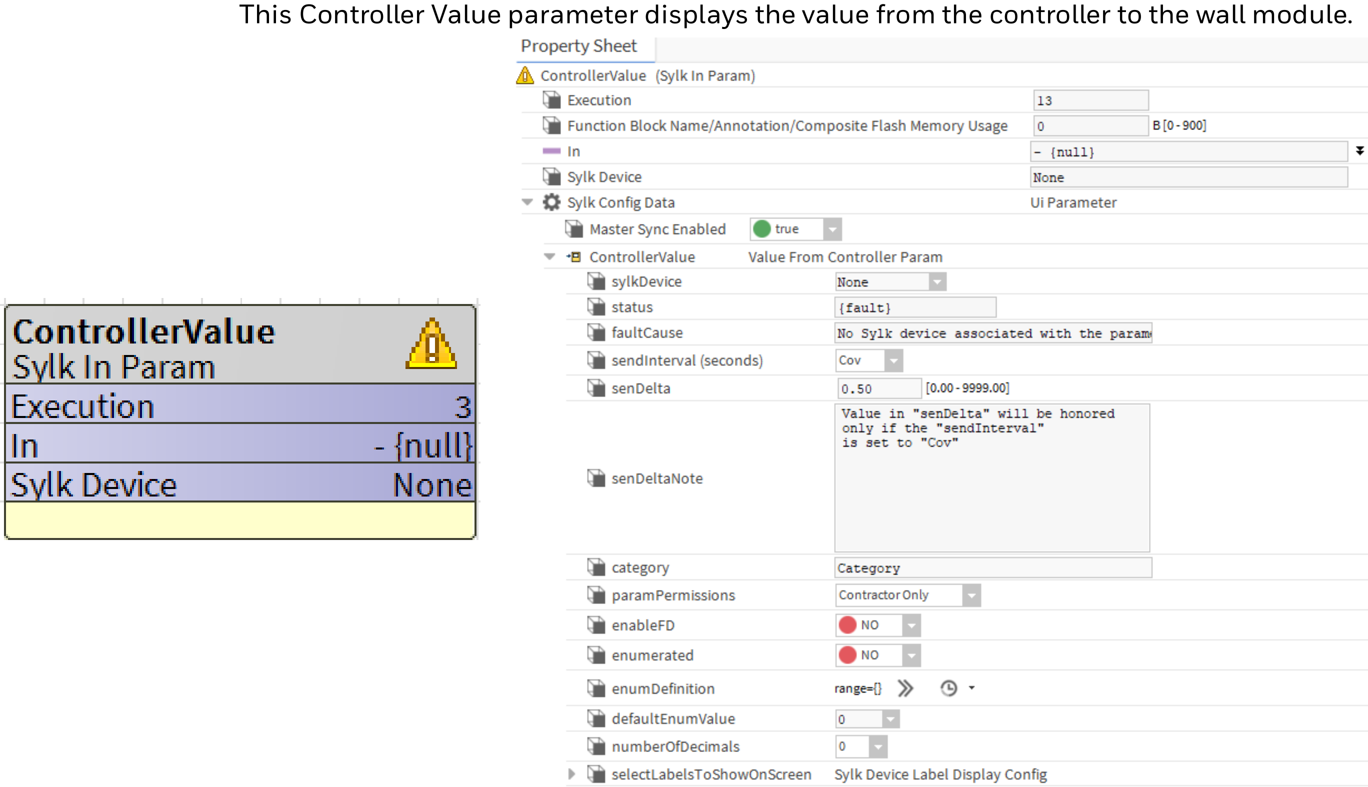


Fig. 531 ControllerValue Function Block and Property Sheet

Inputs

Table 347 Inputs of Controller Value

Input Name	Description
In	To provide the setpoints from the controller to the Wall module.

Output

Table 348 Output of Controller Value

Output Name	Description
NA	NA

Parameters

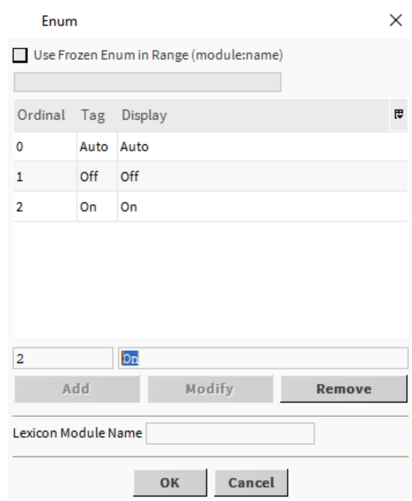
Table 349 Parameter of Controller Value

Parameter Name	Description
sylkdevice	Select the required TR7x device from the drop-down menu.
status	This field is read-only. Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendInterval (seconds)	Send 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. Default is Cov seconds.

Table 349 Parameter of Controller Value (Continued)

Parameter Name	Description
senDelta	SenDelta is another name for the delta value for Significant Event Notification. It allows the controller to send the input parameter value to the Sylk device only when the difference between the last sent value and the current value is greater than or equal to senDelta. Otherwise, the value change will not be propagated to the Sylk device. Example: Assume senDelta is set to 2, the last sent input parameter value is 2, and the current value is 5. The difference in parameter values is 3 (5-2=3), which is greater than the value of senDelta (such as 2). This enables the controller to send the current value of the input parameter (such as 5) to the Sylk device. It is introduced to control the increase in network traffic caused by frequent changes in the values of input parameters. This can be used to tune the controller so that minor value changes do not propagate to the network. The senDelta value is set to 0.5 by default. ***Note:*** *The senDelta feature is only available if the sendInterval is set to "Cov." A senDelta of zero indicates that any change in value is significant and is propagated to the network.*
senDeltaNote	The senDelta recognizes the value only when the sendInterval is set to "Cov."
category	The category is defined by the user while configuring Sylk parameters. The category is displayed as the default 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, a tenant can view the parameters but cannot make any changes in the parameters.
enableFD	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.
enumerated	Using this option allows enabling multiple states of the enum setpoint.

Table 349 Parameter of Controller Value (Continued)

Parameter Name	Description
enumDefinition	If the enumerated is set to “Yes,” then the user can set the multiple states of the Enum setpoint. **To set the Enum states of the Setpoint:** a. Click on the >> button.  **Fig. 532 enumDefinition** b. Set the states **Ordinal, Tag, and Display** as per requirement. For example.  **Fig. 533 Enum** c. Click **OK** to save. ***Note:*** *The enumerated option applicable to Sylk module type is TR 71 and TR 75. Select the parameter type as Controller Value, Value from Wall Module, or Network Setpoint to enable the Enumerated*
defaultEnumValue	This option displays the list of default Enum values from the defined enum range.
numberOfDecimals	Apply to all parameter types except Time. This value affects Increment or Decrement options.
selectLabelsToShowOnScreen	Select the required option to show the parameter.

Occupancy Status

The occupancy status in the wall module.

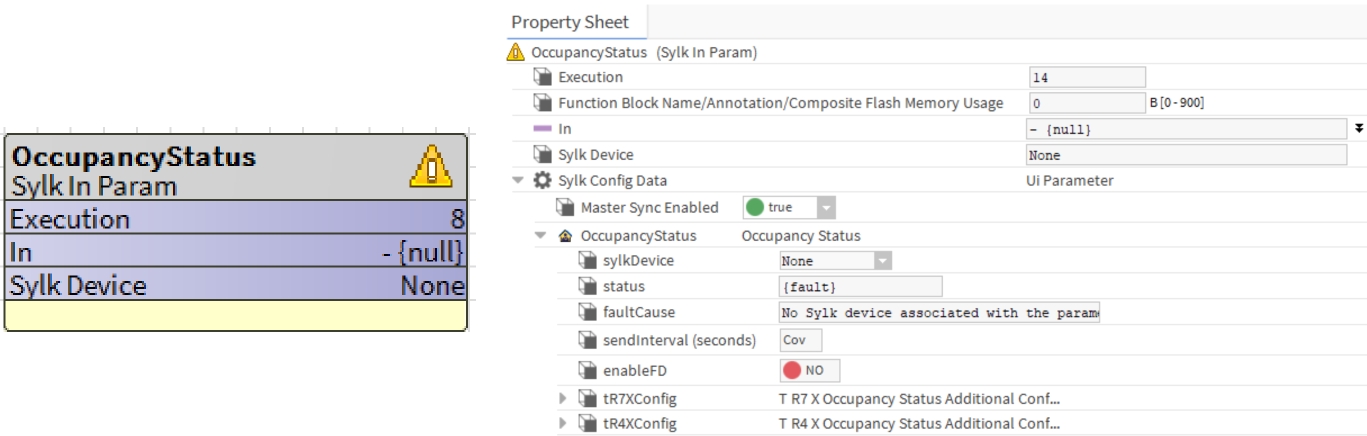


Fig. 534 OccupancyStatus Function Block and Property Sheet

Inputs

Table 350 Input of Occupancy Status

Input Name	Description
In	To provide the System Occupancy Status to the Wall module.

Output

Table 351 Output of Occupancy Status

Output Name	Description
NA	NA

Parameters

Table 352 Parameters of Occupancy Status

Parameter Name	Description
sylkDevice	Select the required TR7x or TR4x device from the drop-down menu.

Table 352 Parameters of Occupancy Status (Continued)

Parameter Name	Description
Status	This field is read-only. Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendInterval (seconds)	Send 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. Default is Cov seconds.
enableFD	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.
tR7XConfig	This option provides options to select how to override status is required to be displayed in the LCD. Following available options: • **Show effective occupancy status:** LCD shows the actual occupancy status considering the IRM application. • **Show the occupancy override status:** LCD shows the occupancy override status initiated from the LCD, independent of the IRM application. • **Do not show occupancy or override status:** LCD does not show occupancy or override, regardless of what the user initiates and the IRM application
tR4XConfig	Configure the configure occupancy override additional settings for TR42.

System Status

This System Status parameter displays the system status in the wall module.

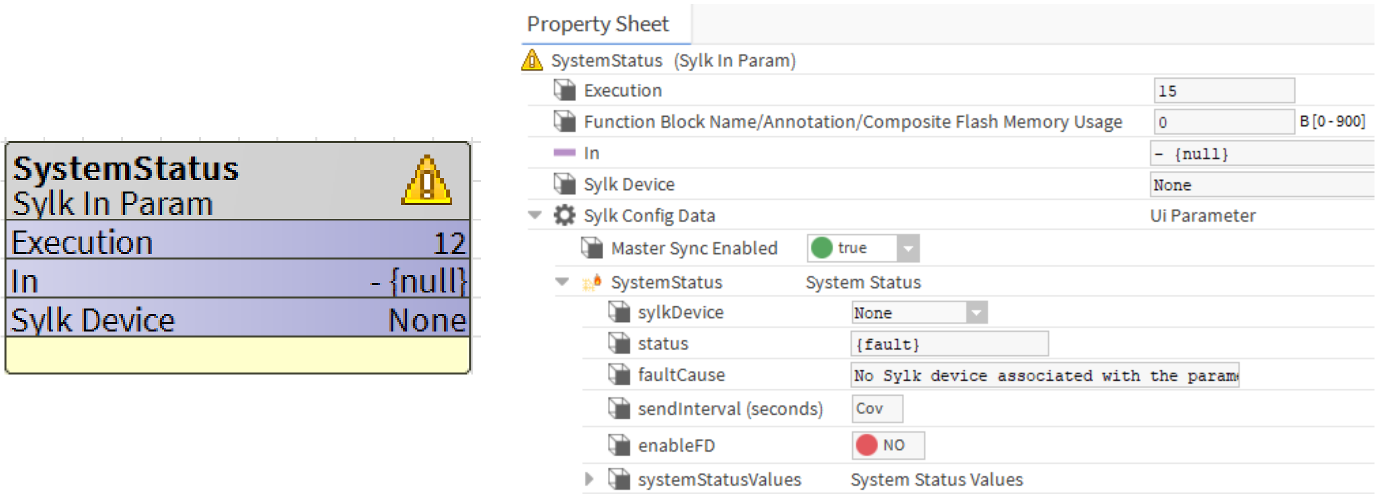


Fig. 535 SystemStatus Function Block and Property Sheet

Inputs

Table 353 Inputs of System Status

Input Name	Description
In	To provide the system status from the controller to the Wall module.

Output

Table 354 Output of System Status

Output Name	Description
NA	NA

Parameters

Table 355 Parameters of System Status

Parameter Name	Description
sylkDevice	Select the required TR7x device from the drop-down menu.

Table 355 Parameters of System Status (Continued)

Parameter Name	Description
status	This field is read-only. Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.
faultCause	Thi field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendInterval (seconds)	Send 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. Default is 60 seconds.
enableFD	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.
SystemStatusValue	To configure the system values within the defined range based on the requirements.

Time of Day

The Time of Day parameter sets the time format for the wall module.

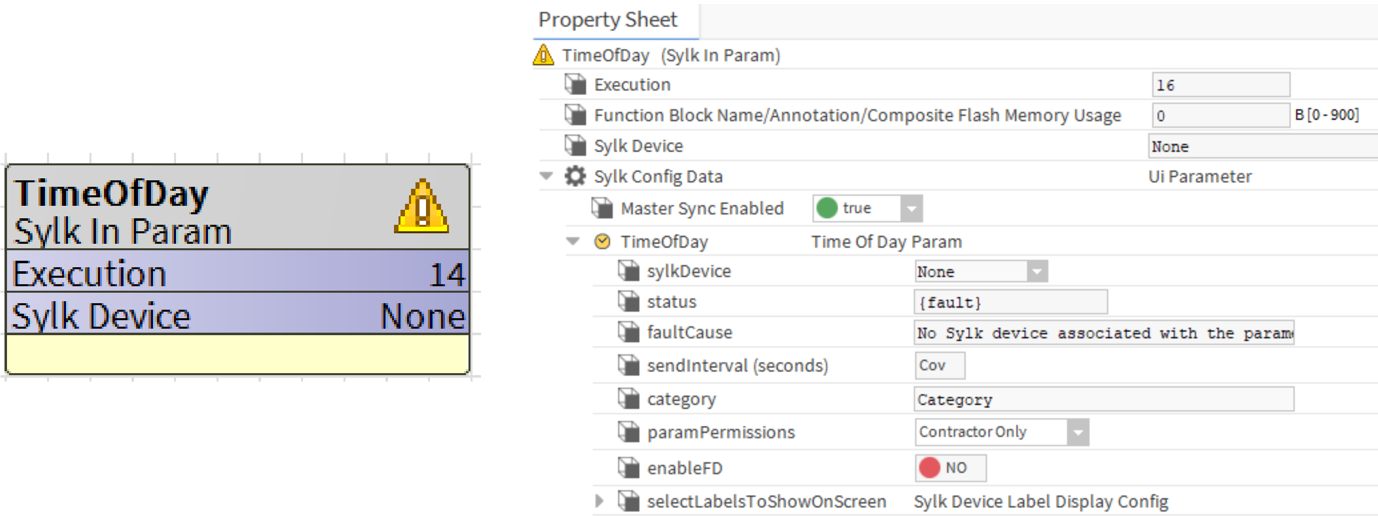


Fig. 536 TimeofDay Function Block and Property Sheet

Inputs

Table 356 Inputs of Time of Day

Input Name	Description
In	To provide the system time to the wall module.

Output

Table 357 Output of Time of Day

Output Name	Description
NA	NA

Parameters

Table 358 Parameters of Time of Day

Parameter Name	Description
sylkDevice	Select the required TR7x device from the drop-down menu.

Table 358 Parameters of Time of Day (Continued)

Parameter Name	Description
status	This field is read-only. Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendInterval (seconds)	Send 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. Default is 60 seconds.
Category	The category is defined by the user while configuring Sylk parameters. The category is displayed as the default 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, a tenant can view the parameters but cannot make any changes in the parameters.
enableFD	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.
selectLabelsToShowOnScreen	This option allows to config the Sylk device label display for the Sylk parameter.

Sylk Actuator Input

The Sylk actuator input controls the controller output or actuator travel time. The actuator travel time is the time required by the actuator to travel from 0 % to 100 % open or 100 % to 0 % open. This time interval varies depending on the type of actuator and ranges from 0 to 500.

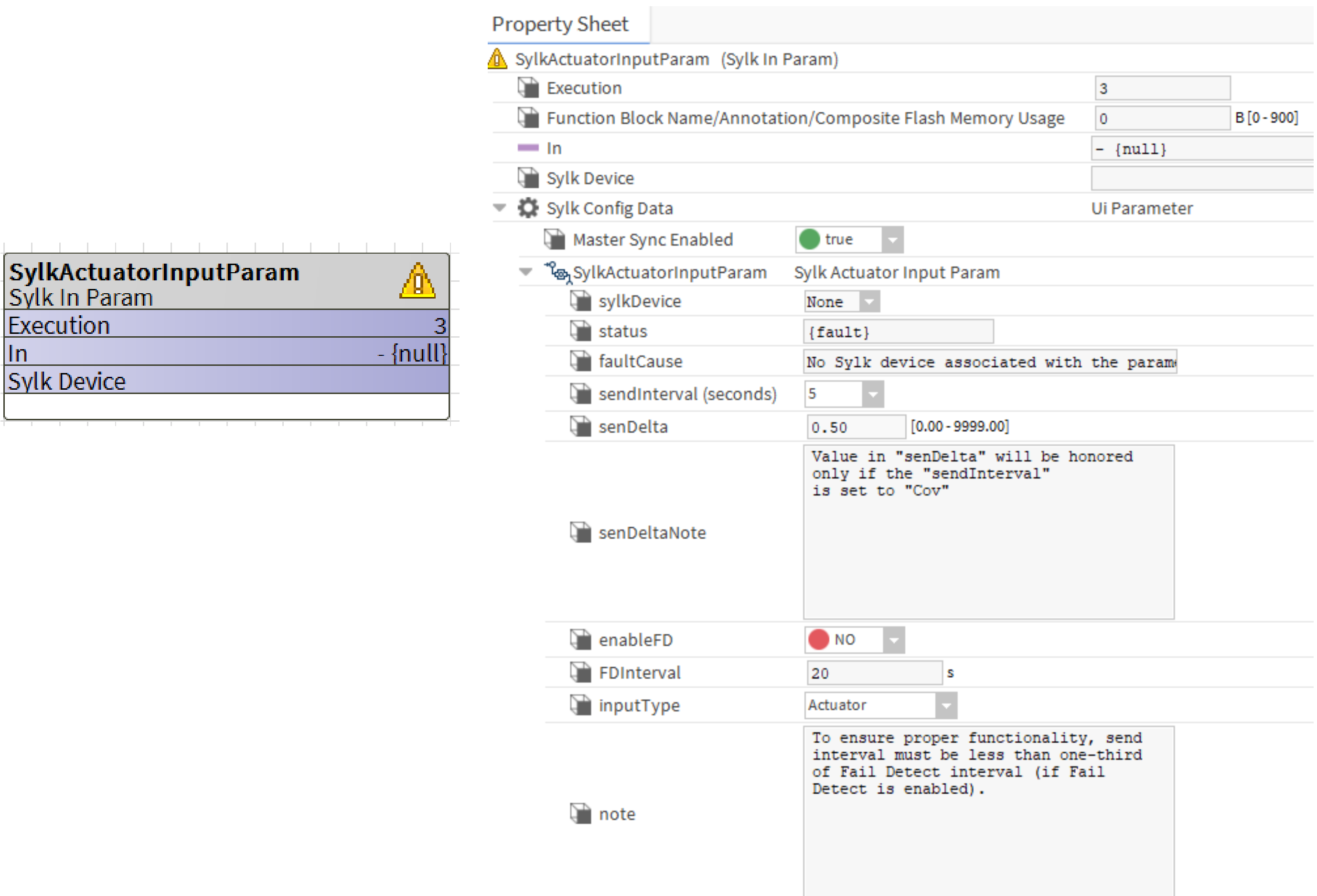


Fig. 537 SylkActuatorInput Function Block and Property Sheet

Inputs

Table 359 Inputs of Sylk Actuator Input

Input Name	Description
In	To provide the value from the controller to the Sylk actuator.

Output

Table 360 Output of Sylk Actuator Input

Output Name	Description
NA	NA

Parameters

Table 361 Parameters of Sylk Actuator Input

Parameter Name	Description
sylkDevice	Select the required SylkActuator device from the drop-down menu.
status	This field is read-only. Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendInterval (seconds)	Send 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. Default is 5 seconds.

Table 361 Parameters of Sylk Actuator Input (Continued)

Parameter Name	Description
senDelta	SenDelta is another name for the delta value for Significant Event Notification. It allows the controller to send the input parameter value to the Sylk device only when the difference between the last sent value and the current value is greater than or equal to senDelta. Otherwise, the value change will not be propagated to the Sylk device. Example: Assume senDelta is set to 2, the last sent input parameter value is 2, and the current value is 5. The difference in parameter values is 3 (5-2=3), which is greater than the value of senDelta (such as 2). This enables the controller to send the current value of the input parameter (such as 5) to the Sylk device. It is introduced to control the increase in network traffic caused by frequent changes in the values of input parameters. This can be used to tune the controller so that minor value changes do not propagate to the network. The senDelta value is set to 0.5 by default. ***Note:*** *The senDelta feature is only available if the sendInterval is set to "Cov." A senDelta of zero indicates that any change in value is significant and is propagated to the network.*
senDeltaNote	The senDelta recognizes the value only when the sendInterval is set to "Cov."
enableFD	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.
FDInterval	FDInterval for Sylk actuator input parameter is 20 secs.
inputType	There are two types of Sylk actuator inputs actuator command and actuator travel time: • **Actuator:** Actuator command comes from the controller output. An actuator moves as per the command. • **Actuator Travel Time:** The Actuator travel time is the time required by the actuator to travel from 0 % to 100 % open or 100 % to 0 % open. This time interval depends on the actuator type and varies from 0 to 500 seconds.
note	Send interval must be less than one-third of Fail Detect interval (if Fail Detect is enabled) to ensure proper functionality.

CHAPTER

24 SYLK OUT PARAMETERS

Following are the parameters that capture input from various sensors and act as output for the Sylk device:

- CO2
- Humidity
- Occupancy Override Command
- ROOMTEMP
- Sensor Offset
- Value From Wall Module
- Sylk Actuator Output

CO2

This parameter provides CO2 concentration from the room.

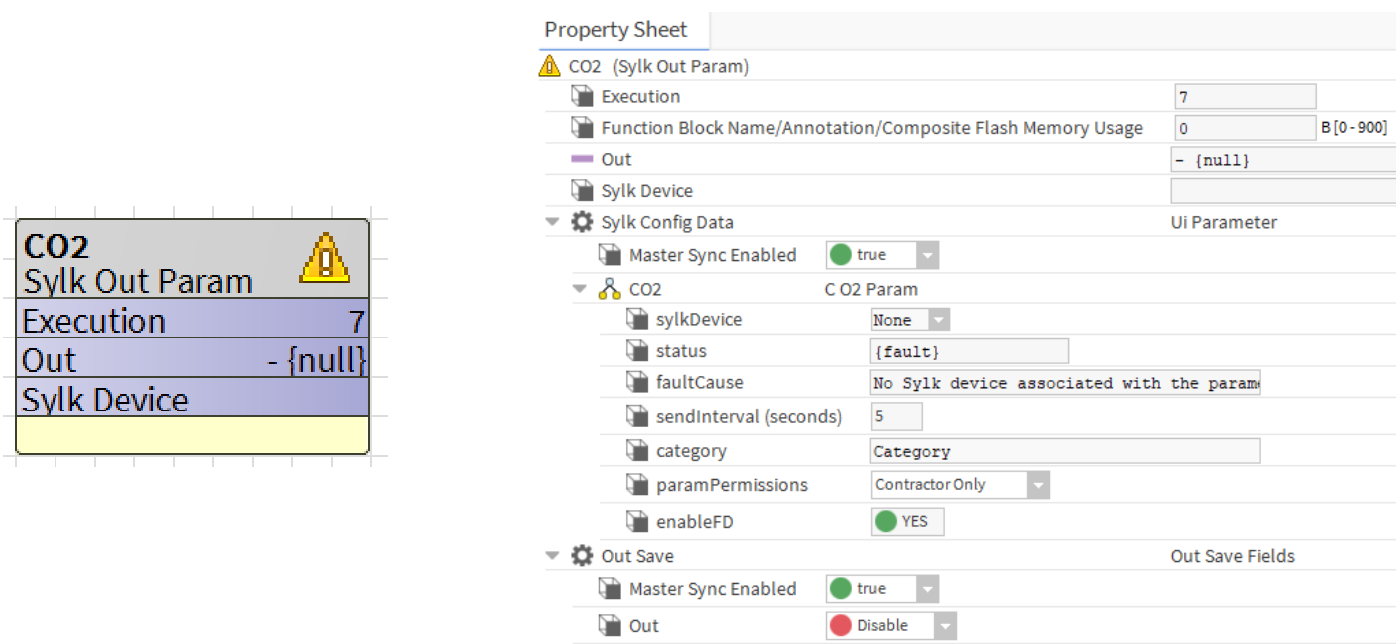


Fig. 538 CO2 Function Block and Property Sheet

Inputs

Table 362 Input of CO2

Input Name	Description
NA	NA

Output

Table 363 Output of CO2

Output Name	Description
Out	Shows the current value of the parameter.

Parameters

Table 364 Parameters of CO2

Parameter Name	Description
sylkDevice	Select the required Tr4x device from the drop-down menu.
Status	This field is read-only. Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendInterval (seconds)	Send 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. Default is 5 seconds.
category	The category is defined by the user while configuring Sylk parameters. The category is displayed as the default category.

Table 364 Parameters of CO2 (Continued)

Parameter Name	Description
paramPermissins	Users 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, a tenant can view the parameters but cannot make any changes in the parameters.
enableFD	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **OUT:** To enable or disable the Out Save feature.

Humidity

The Humidity parameter shows the percentage of humidity in the room.

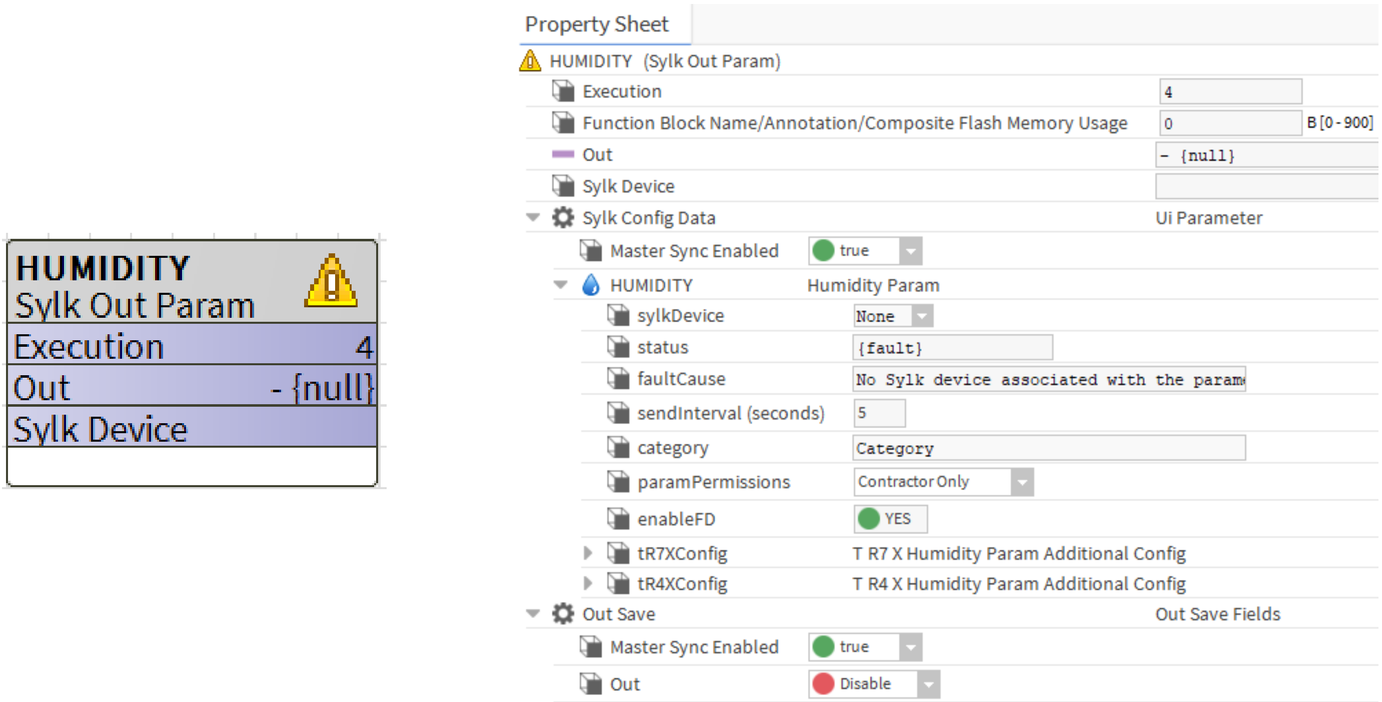


Fig. 539 Humidity Function Block and Property Sheet

Parameters

Table 365 Parameters of Humidity

Parameter Name	Description
sylkDevice	Select the required TR7x or Tr4x device from the drop-down menu.
Status	This field is read-only. Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.

Table 365 Parameters of Humidity (Continued)

Parameter Name	Description
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendInterval (seconds)	Send 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. Default is 5 seconds.
category	The category is defined by the user while configuring Sylk parameters. The category is displayed as the default category.
paramPermissions	Users 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, a tenant can view the parameters but cannot make any changes in the parameters.
enableFD	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.
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.
tR4XConfig	• **Number of Decimals:** Enter the decimal accuracy. • **Default Sensor Offset Value:** Enter the default value for sensor offset.
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **OUT:** To enable or disable the Out Save feature.

Occupancy Override Command

The Occupancy Override Command parameter allows you to switch from Unoccupancy Mode to Occupied Mode.

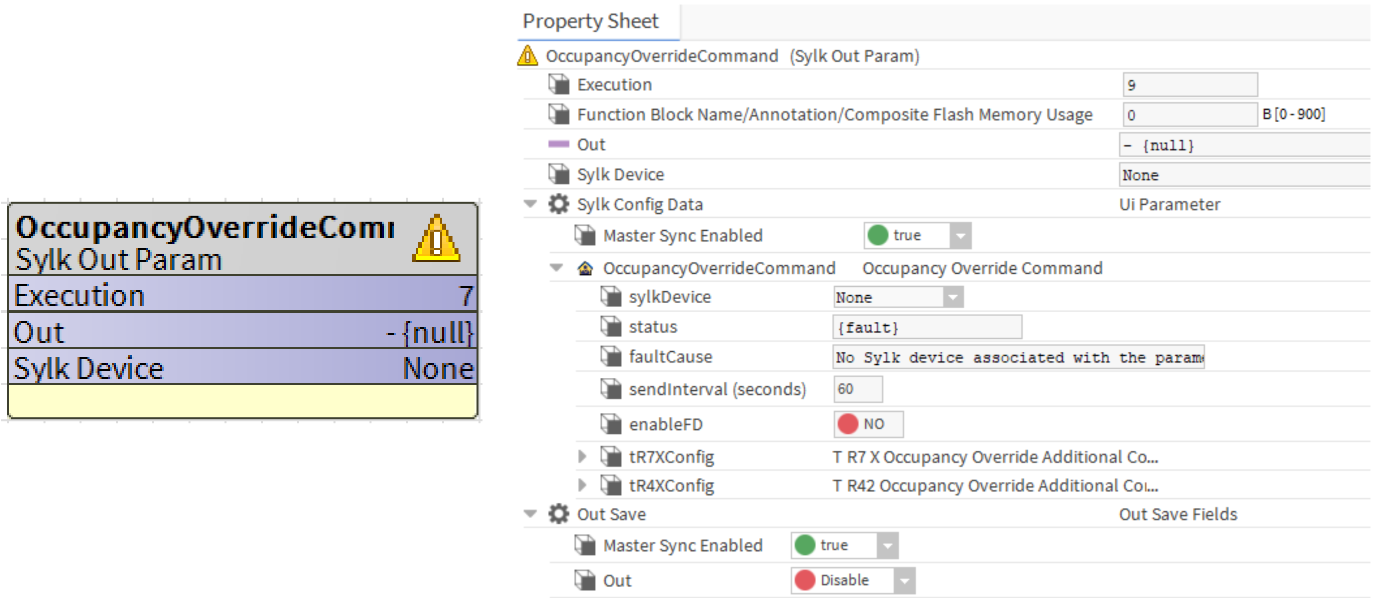


Fig. 540 Occupancy Override Command Function Block and Parameters

Inputs

Table 366 Inputs of Occupancy Override Command

Input Name	Description
NA	NA

Output

Table 367 Output of Occupancy Override Command

Output Name	Description
Out	Provide the override command to the wall module from the wall module.

Parameters

Table 368 Parameter of Occupancy Override Command

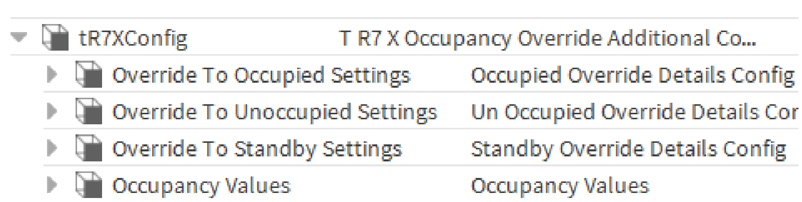
Parameter Name	Description
sylkDevice	Select the required TR7x or TR4x device from the drop-down menu.
Status	This field is read-only Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendinterval (seconds)	The time between the end of a timeout period or the completion of a network request and the next data request on the network is referred to as the send interval. The default value is 60 seconds.
enableFD	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.
tR7XConfig	Configure the configure occupancy override additional settings for TR7x.  **Fig. 541 tR7XConfig**

Table 368 Parameter of Occupancy Override Command (Continued)

<table>
<tr><th>Parameter Name</th><th>Description</th></tr>
<tr><td></td><td>

- **Override To Occupied Settings:** Configure occupied override settings.

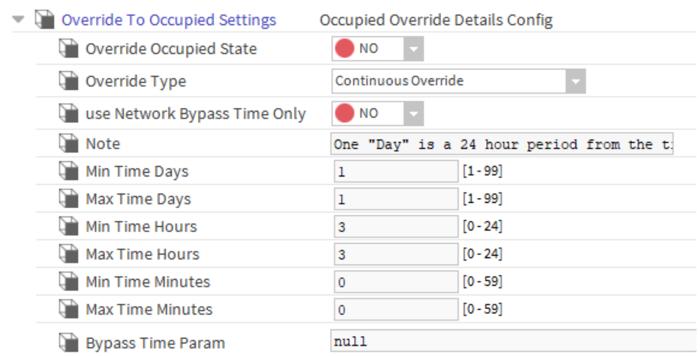


Fig. 542 Override To Occupied Settings

  - **Override Type**

    Continuous Override: This is the default setting. It setting disables the time override type options.

    Time Override in Hours (Bypass): This option enables the time setting details, wherein the minimum time and maximum time can be set in hours and minutes. The default setting is min= max= 3 hours. Make sure that the maximum time is greater than the minimum time. If either case is not true, an error message appears informing that the min value is greater than the max value. The range is from 0 to 24 hrs / 0 to 59 min.

    Time Override in Days (Bypass): This option enables the time setting details, wherein the minimum and maximum time can be set in days. In this case, the hours and minutes options are disabled. The default setting is min = max = 1 day. The range is from 1 to 99 days.
If the range is exceeded, an error message appears to inform about the same.

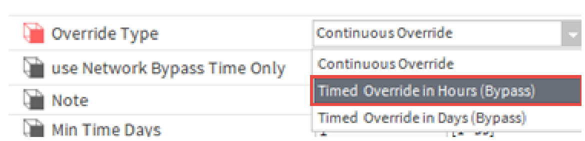


Fig. 543 Override Type

  - **Use Network Bypass Time Only:** If this option is selected, then all other override details are disabled. The timed override will be determined by the application.

Note: *The Use Network Bypass Time only option applies to Occupied override type settings. This option is not available for Unoccupied and Standby override type settings.*

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

Table 368 Parameter of Occupancy Override Command (Continued)

<table>
<tr><th>Parameter Name</th><th>Description</th></tr>
<tr><td></td><td>

- **Time Settings:** Configure occupied override time setting.

Fig. 544 Override Settings

- **By pass time parma:** To provide the bypass time to a wall module

• .**Override To Unoccupied Settings**

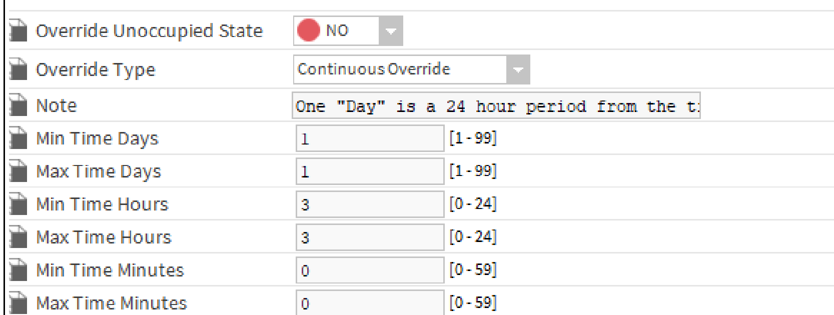


Fig. 545 Override to Unoccupied

- **Override Type**

Continuous Override: The default setting, this setting disables the time override type options.

Time Override in Hours (Bypass): This option enables the time setting details. The minimum time and maximum time can be set in hours and minutes. The default setting is min = max = 3 hours.Ensure that the maximum time is greater than the minimum time or that the minimum time is lesser than the maximum time. If either case is not true, an error message appears informing that the min value is greater than the max value. The range is from 0 to 24 hrs / 0 to 59 min.

Time Override in Days (Bypass): This option enables the time setting details, wherein the minimum and maximum time can be set in days. In this case, the hours and minutes options are disabled. The default setting is min = max = 1 day. The permissible range is 1 to 99 days.

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

Table 368 Parameter of Occupancy Override Command (Continued)

Parameter Name	Description
	(continued below)

Fig. 546 Override Type

- **Time Settings:** Configure unoccupied override time setting.

Fig. 547 Override Settings

- **Override To Standby Settings**

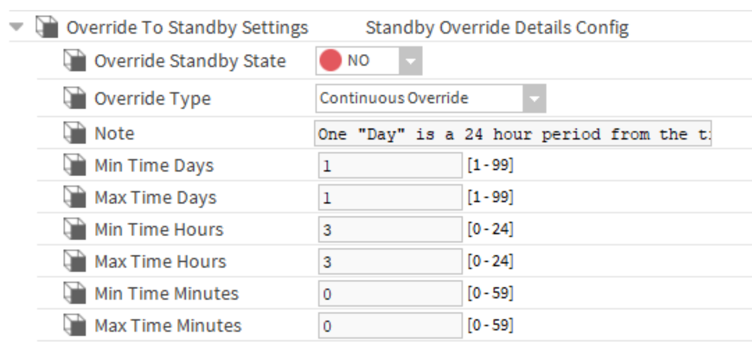


Fig. 548 Override To Standby

- **Override Type**

  Continuous Override: The default setting, this setting disables the time override type options.

  Time Override in Hours (Bypass): This option enables the time setting details. The minimum and maximum time can be set in hours and minutes. The default setting is min= max= 3 hours. Ensure that the maximum time is greater than the minimum time or that the minimum time is lesser than the maximum time. If either case is not true, an error message appears informing that the min value is greater than the max value. The range is from 0 to 24 hrs / 0 to 59 min.

Table 368 Parameter of Occupancy Override Command (Continued)

<table>
<tr><th>Parameter Name</th><th>Description</th></tr>
<tr><td></td><td>

Time Override in Days (Bypass): This option enables the time setting details, wherein the minimum and maximum time can be set in days. In this case, the hours and minutes options are disabled. The default setting is min = max = 1 day. The permissible range is 1 to 99 days..

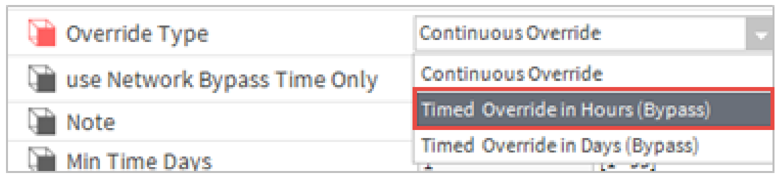


Fig. 549 Override Type

- **Time Settings**: Configure standby override time setting.

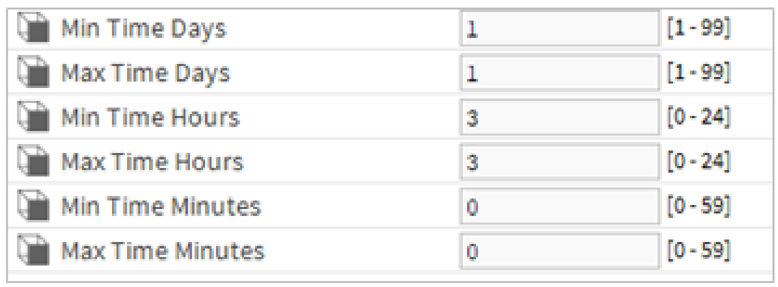


Fig. 550   Override Settings

- **Occupancy Values**

Provides options to define the different values within the mentioned ranges.

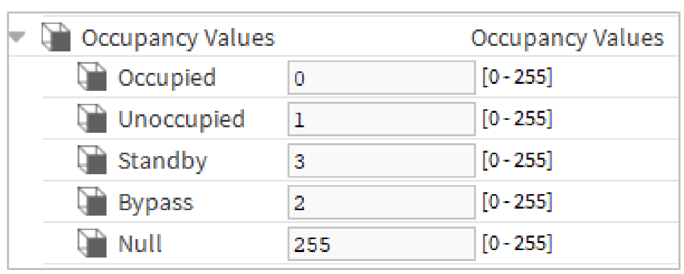


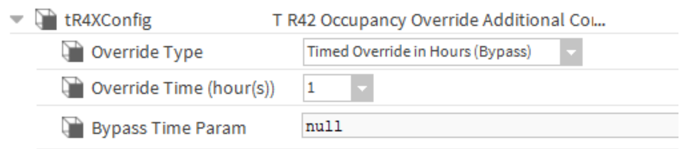
Fig. 551 Occupancy Values

Show effective occupancy status: LCD displays the actual occupancy status considering the IRM application.

Show occupancy override status: LCD displays the occupancy override status initiated from the LCD, independent of the IRM application.

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

Table 368 Parameter of Occupancy Override Command (Continued)

Parameter Name	Description
	Do not show occupancy or override status: LCD does not display occupancy or override, regardless of what the user initiates and the IRM application.
tR4XConfig	This option allows you to configure additional param options.  **Fig. 552 tR4XConfig Additional Parameter Options**
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **OUT:** To enable or disable the Out Save feature.

ROOMTEMP

The ROOMTEMP parameter displays the room temperature.

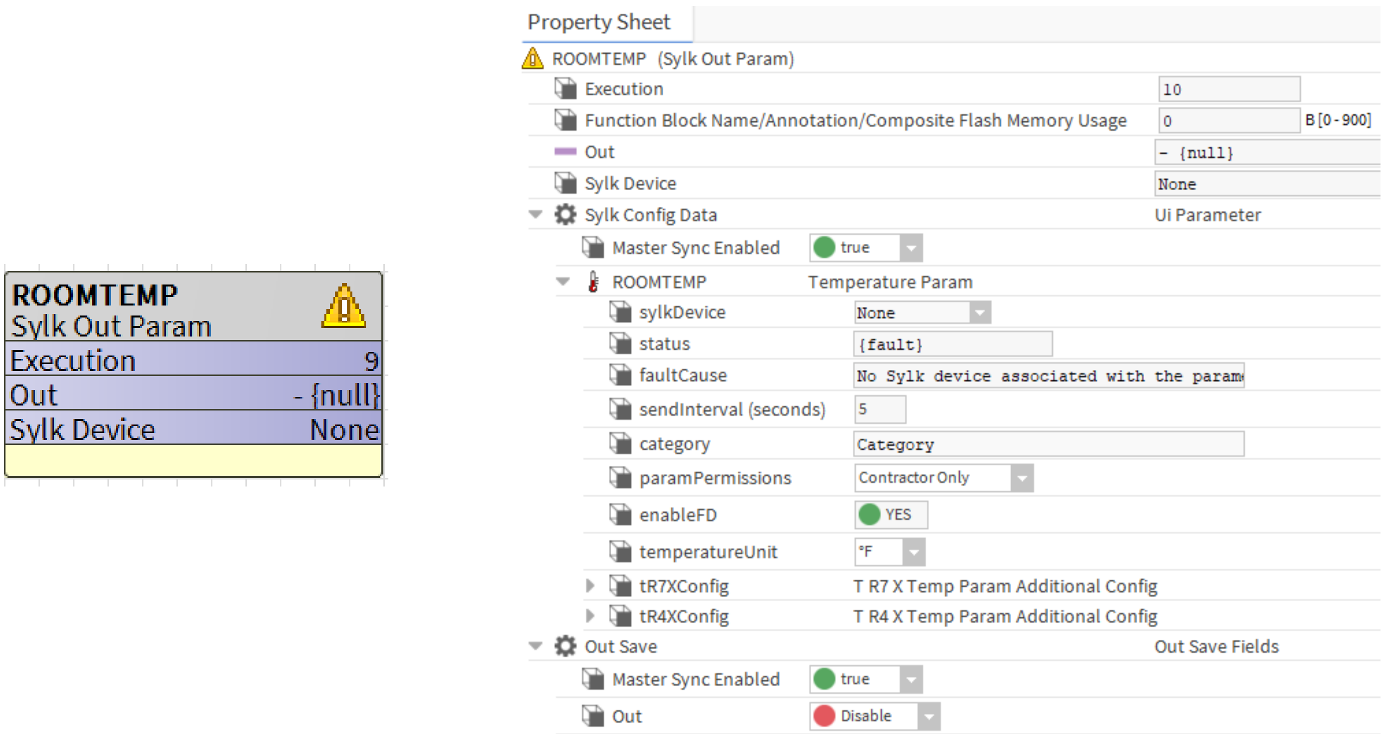


Fig. 553 ROOMTEMP Function Block and Property Sheet

Inputs

Table 369 Inputs of ROOMTEMP

Input Name	Description
NA	NA

Output

Table 370 Output of ROOMTEMP

Output Name	Description
Out	Shows the current value of the parameter.

Parameters

Table 371 Parameters of ROOMTEMP

Parameter Name	Description
sylkDevice	Select the required TR7x or TR4x device from the drop-down menu.
Status	This field is read-only. Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendInterval (seconds)	The time between the end of a timeout period or the completion of a network request and the next data request on the network is referred to as the send interval. The default value is 5 seconds.
Category	The category is defined by the user while configuring Sylk parameters. The category is displayed as the default 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, a tenant can view the parameters but cannot make any changes in the parameters.

Table 371 Parameters of ROOMTEMP (Continued)

Parameter Name	Description
enableFD	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.
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 Humidity option to show the parameter on the home screen.
tR4XConfig	• **Number of Decimals:** Enter the decimal accuracy. • **Default Sensor Offset Value:** Enter the default value for sensor offset.
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **OUT:** To enable or disable the Out Save feature.

Sensor Offset

The Sensor offset parameter provides the sensor offset to the wall module.

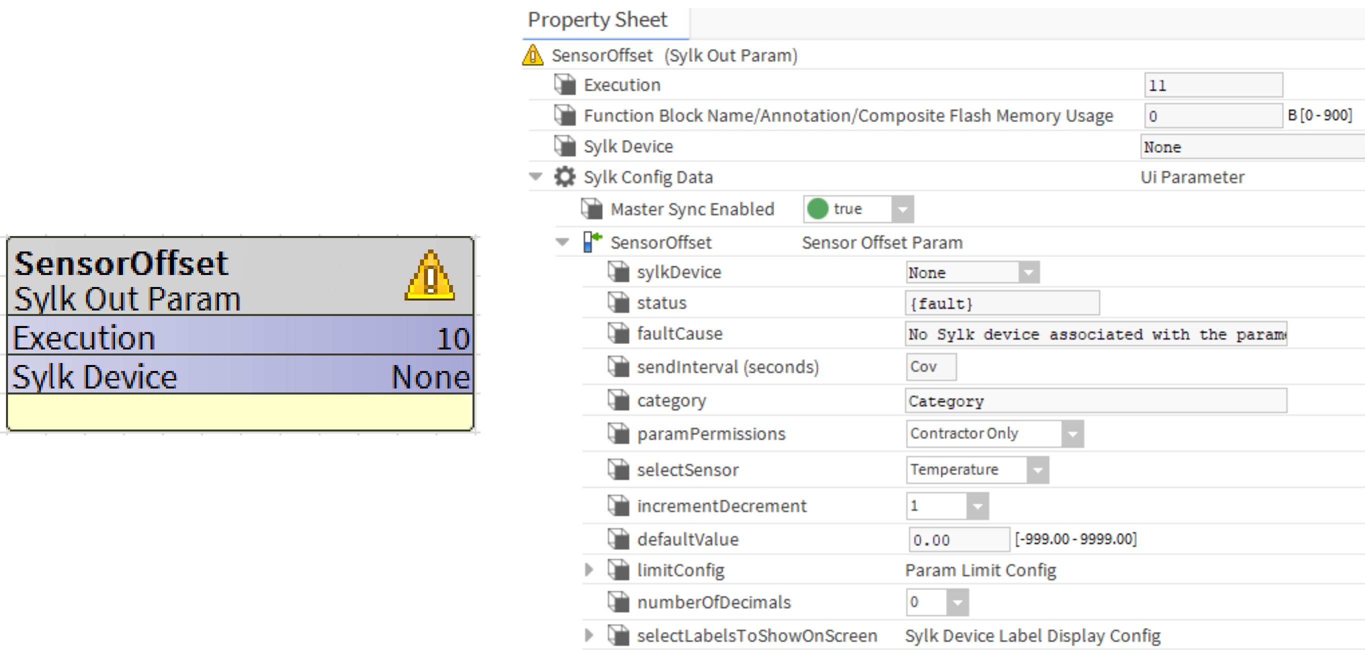


Fig. 554 Sensor Offset Function Block and Property Sheet

Inputs

Table 372 Inputs of Sensor Offset

Input Name	Description
NA	NA

Output

Table 373 Output of Sensor Offset

Output Name	Description
NA	NA

Parameters

Table 374 Parameters of Sensor Offset

Parameter Name	Description
sylkDevice	Select the required TR7x device from the drop-down menu.

Table 374 Parameters of Sensor Offset (Continued)

Parameter Name	Description
Status	This field is read-only. Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendInterval (seconds)	Send 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. Default is Cov.
Category	The category is defined by the user while configuring Sylk parameters. The Category is displayed as the default 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, a tenant can view the parameters but cannot make any changes in the parameters. • If Tenant Read Write is selected, a tenant can view as well as modify the parameters.
sensorSelect	Choose the appropriate sensor (Temperature, Humidity, or CO2).
incrementDecrement	Depends on the number of decimal places selected. • If the number of decimal places is 0, then select 1, 5, 10, 100. • If the number of decimal places is 1, then select 0.1, 0.5, 1, 5, 10, 100. • If number of decimal places is 2, then select 0.01, 0.1, 0.5, 1, 5, 10, 100.

Table 374 Parameters of Sensor Offset (Continued)

Parameter Name	Description
defaultValue	Indicates the value that will be initially downloaded to the wall module. This value must be between the low and high limits. This field is displayed for the following parameter types: • Value from wall module. • Sensor offset value. It is set to 0 by default.
limitConfig	To set the high & low limits of the Offset. • **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:** To set the low limit. • **High Limit:** To set the high limit.
numberOfDecimals	Apply to all parameter types except Time. This value affects IncrementDecrement options.
selectLabelsToShowOnScreen	This option allows to config the Sylk device label display for the Sylk parameter.

Value From Wall Module

The Value From Wall Module parameter provides the value from the wall module to the controller.

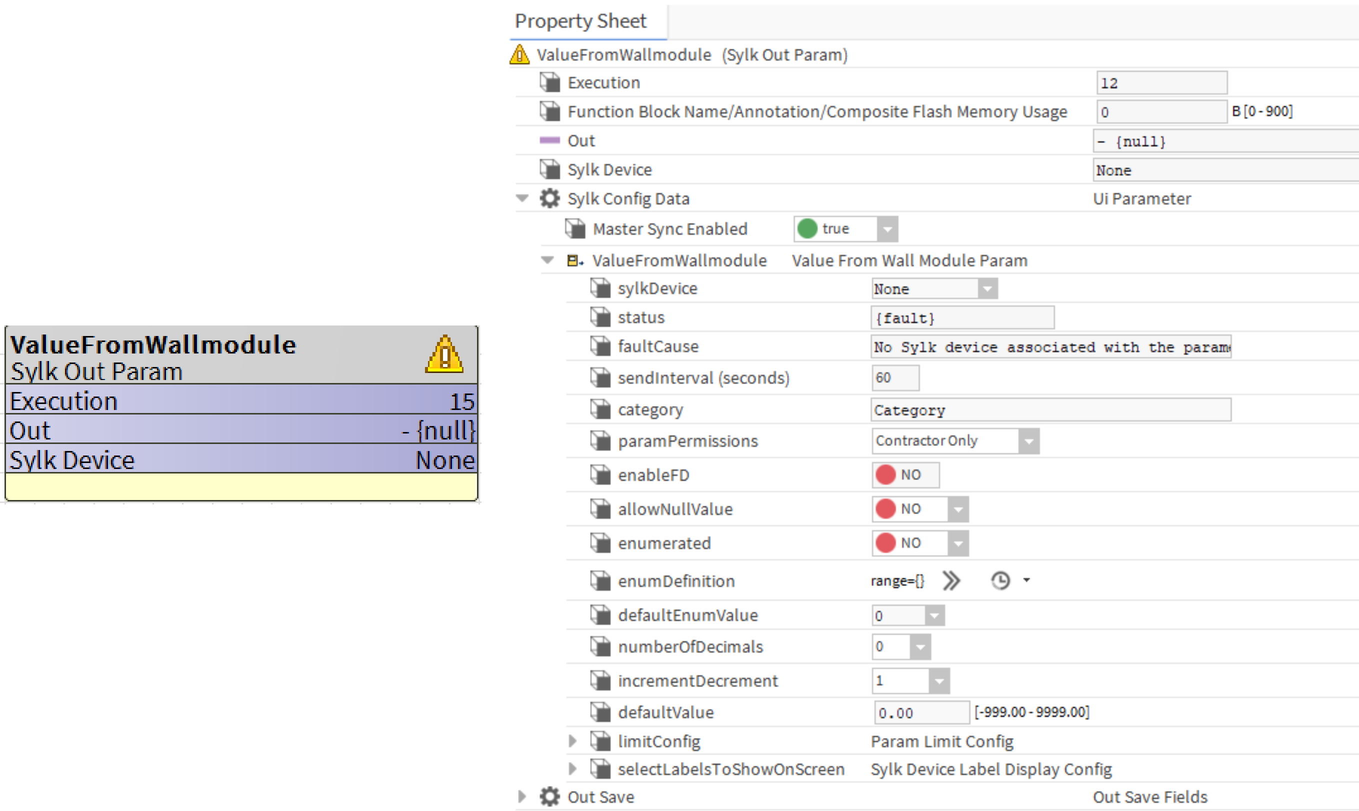


Fig. 555 Value From Wall Module Function Block and Property Sheet

Inputs

Table 375 Inputs of Value From Wall Module

Input Name	Description
NA	NA

Output

Table 376 Output of Value From Wall Module

Output Name	Description
Out	Provides the respective value to the controller from the wall module.

Parameters

Table 377 Parameters of Value From Wall Module

Parameter Name	Description
sylkDevice	Select the required TR7x device from the drop-down menu.
Status	This field is read-only. Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendInterval (seconds)	Send 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. Default is 60 seconds.
Category	The category is defined by the user while configuring Sylk parameters. The Category is displayed as the default 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, a tenant can view the parameters but cannot make any changes in the parameters. • If Tenant Read Write is selected, a tenant can view as well as modify the parameters.
enableFD	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.

Table 377 **Parameters of Value From Wall Module (Continued)**

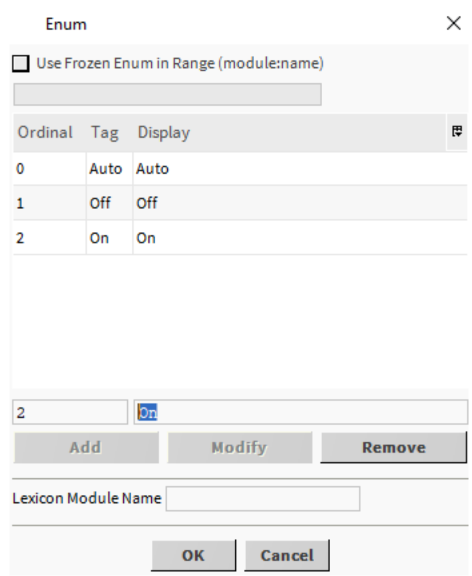
Parameter Name	Description
allowNullValue	Indicate whether null values for the adjustable value are permitted. An additional button press will send the null value when the LCD reaches a low or high limit when this option is enabled.
enumerated	Using this option allows enabling multiple states of the enum setpoint.
enumDefinition	If the enumerated is set to “Yes,” then the user can set the multiple states of the Enum setpoint. To set the Enum states of the Setpoint: a. Click on the >> button.  **Fig. 556 enumDefinition** b. Set the states **Ordinal, Tag, and Display** as per requirement. For example.  **Fig. 557 Enum** c. Click **OK** to save. ***Note:*** *The enumerated option applicable to Sylk module type is TR 71 and TR 75. Select the parameter type as Controller Value, Value from Wall Module, or Network Setpoint to enable the Enumerated.*
defaultEnumValue	This option displays the list of default Enum values from the defined enum range.
numberOfDecimals	Applies to all the parameter types except Time. This value affects Increment or Decrement options.

Table 377 Parameters of Value From Wall Module (Continued)

Parameter Name	Description
incrementDecrement	Depend on the Number of decimal places selected. • If the number of decimal places is 0, then select 1, 5, 10, 100. • If the number of decimal places is 1, then select 0.1, 0.5, 1,5, 10, 100. • If number of decimal places is 2, then select 0.01, 0.1, 0.5, 1, 5, 10, 100.
defaultValue	Indicate the value that will be initially downloaded to the wall module. This value must be between the Low and High limits. This field is displayed for the following parameter types: • Value from wall module. • Sensor offset value. It is set to 0 by default.
limitConfig	This option allows to set the high & low limits of the parameter. • 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: To set low limit. • High Limit: To set the high limit.
selectLabelsToShowOnScreen	This option allows to config the Sylk device label display for the Sylk parameter.
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **OUT:** To enable or disable the Out Save feature.

Sylk Actuator Output

The Sylk actuator output displays feedback from the Sylk actuator input in the form of actuator position, actuator cycle count, actuator status, actuator overridden, or power report.

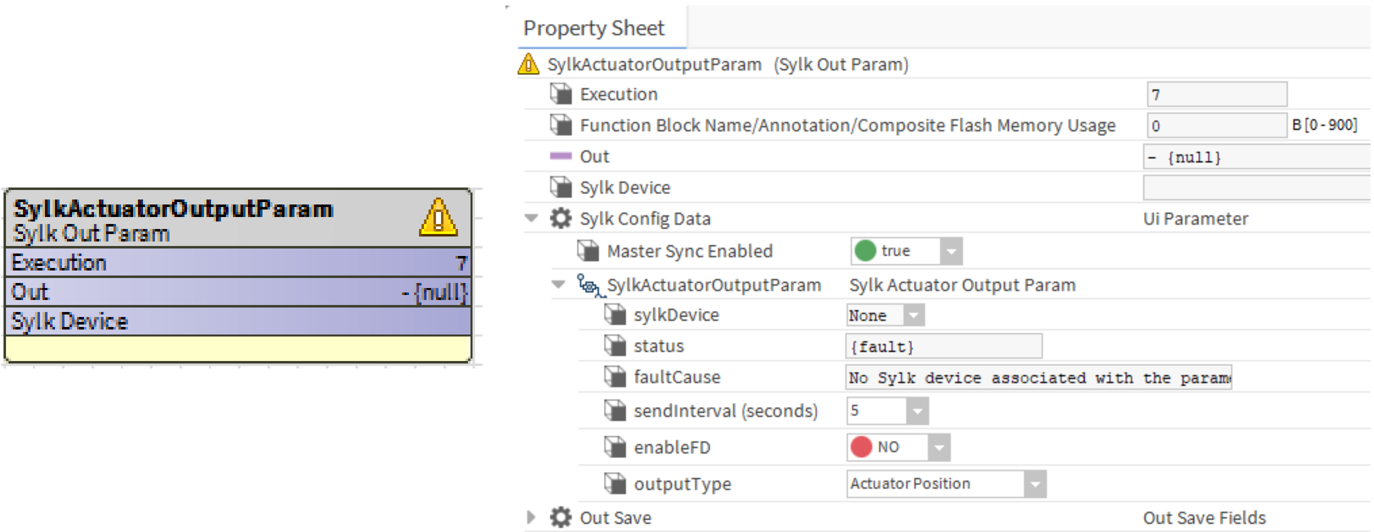


Fig. 558 SylkActuatorOutput Function Block and Property Sheet

Inputs

Table 378 Inputs of Sylk Actuator Output

Input Name	Description
NA	NA

Output

Table 379 Output of Sylk Actuator Output

Output Name	Description
Out	Provides the respective value to the controller from the Sylk actuator.

Parameters

Table 380 Parameters of Sylk Actuator Output

Parameter Name	Description
sylkDevice	Select the required SylkActuator device from the drop-down menu.

Table 380 Parameters of Sylk Actuator Output (Continued)

Parameter Name	Description
status	This field is read-only. Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendInterval (seconds)	Send 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. Default is 5 seconds.
enableFD	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.

Table 380 Parameters of Sylk Actuator Output (Continued)

Parameter Name	Description
outputType	There are five types of Sylk actuator outputs: • **Actuator Position:** This is feedback from the actuator current position ranging between 0 to 100 and can be mapped into the controller for monitoring. • **Actuator Cycle Count:** It shows how many time actuator 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 an actuator is overridden externally. This will return “True” if the unit is in Test Mode. This is 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, a value ranging between 0 to 100.
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **OUT:** To enable or disable the Out Save feature.

CHAPTER

25 SYLK IN OUT PARAMETERS

The Sylk device uses the following parameters as input as well as output:

- Bypass Time
- Fan Command
- Network Setpoint
- System Command
- Time Field
- Sylk Time
- Home Screen

Bypass Time

The Bypass Time parameter specifies the bypass time to be used by the wall module.

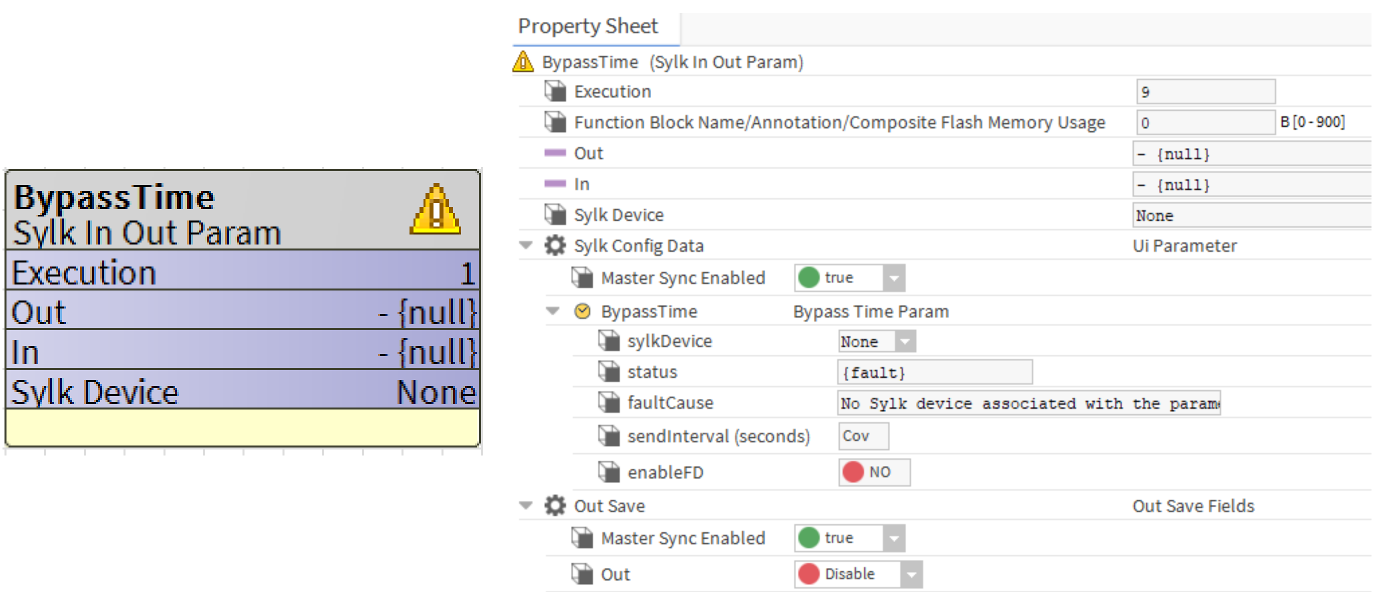


Fig. 559 Bypass Time Function Block and Property Sheet

Inputs

Table 381 Inputs of Bypass Time

Input Name	Description
In	To provide the controller bypass time value to the wall module.

Output

Table 382 Output of Bypass Time

Output Name	Description
Out	To provide the wall module bypass time value to the controller.

Parameters

Table 383 Parameters of Bypass Time

Parameter Name	Description
sylkDevice	Select the required TR7x or Tr4x device from the drop-down menu.
status	This field is read-only. Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendinterval (seconds)	The time between the end of a timeout period or the completion of a network request and the next data request on the network is referred to as the send interval. Cov is the default value.

Table 383 Parameters of Bypass Time (Continued)

Parameter Name	Description
enableFD	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **OUT:** To enable or disable the Out Save feature.

Fan Command

The Fan Command parameter provides commands to the fan from the wall module.

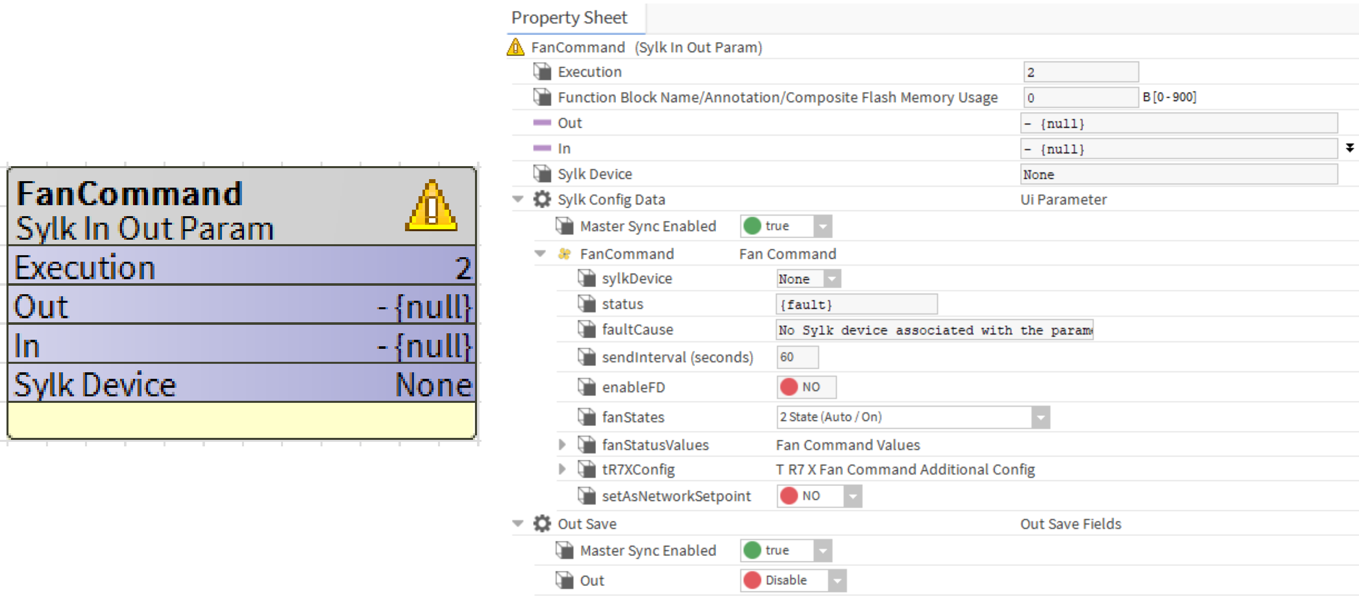


Fig. 560 FanCommand Function Block and Property Sheet

Inputs

Table 384 Inputs of Fan Command

Input Name	Description
In	To provide the Fan Enable Command from the controller to the Wall module.

Output

Table 385 Output of Fan Command

Output Name	Description
Out	To provide the Final Fan Enable Command from the wall module to the controller.

Parameters

Table 386 Parameters of Fan Command

Parameter Name	Description
sylkDevice	Select the required TR7x or Tr4x device from the drop-down menu.
status	This field is read-only. Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
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	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.
fanstates	Select the States of the fan command from the wall module as per requirement. You have the following fan state options: • **2 State (Auto/On):** The default option is On. • **3 State (Auto/On/Off):** The default option is On. • **5 State (Auto/Off/Low/Medium/High):** The default option is Auto.

Table 386 Parameters of Fan Command (Continued)

Parameter Name	Description
fanstateValues	Set the fan state values as per requirement.
tR7XConfig	Set the default state of the fan command.
SetAsNetworksetpo int	This option allows you to specify whether the parameter should be configured as a network set point or as an output only parameter. • If you select Yes, you have access to write and read values. • If you select No, you can only read values from the Sylk device.
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **OUT:** To enable or disable the Out Save feature.

Network Setpoint

The Network Setpoint parameter provides the setpoints to the wall module.

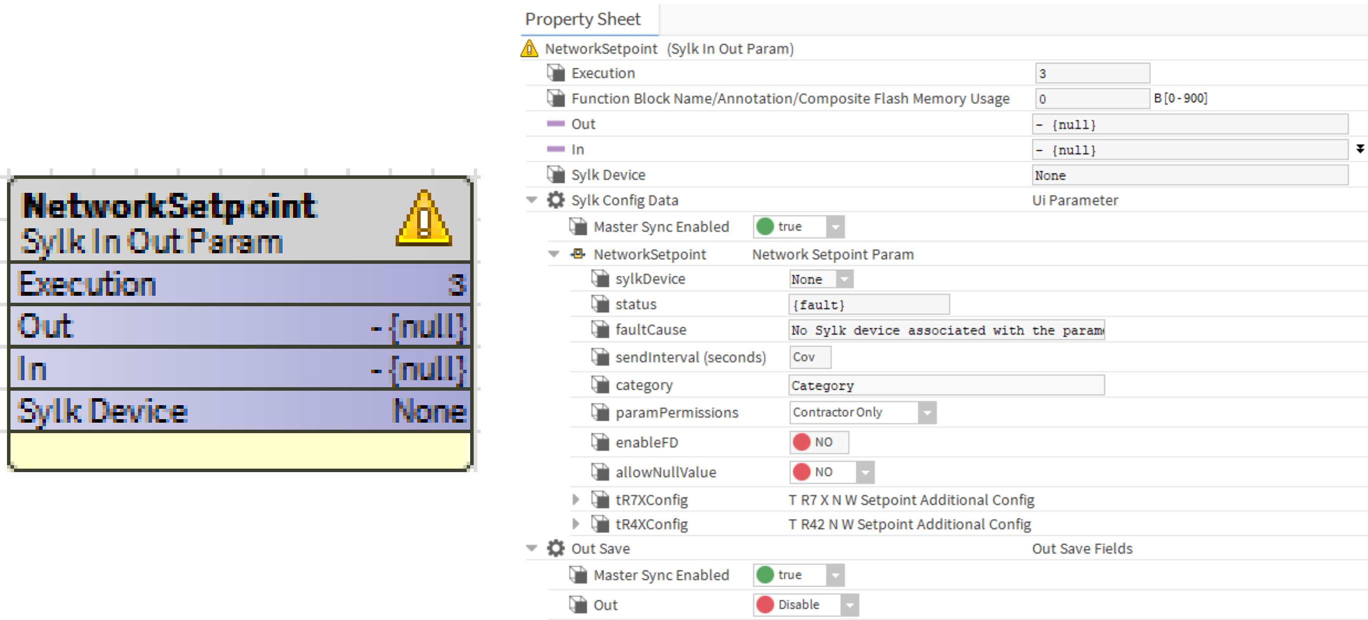


Fig. 561 Network Setpoint Function Block and Property Sheet

Inputs

Table 387 Inputs of Network Setpoint

Input Name	Description
In	To provide the Controller Setpoint Value to the Wall module.

Output

Table 388 Output of Network Setpoint

Output Name	Description
Out	To provide the wall module Setpoint Value to the controller.

Parameters

Table 389 Parameters of Network Setpoint

Parameter Name	Description
sylkDevice	Select the required TR7x or Tr4x device from the drop-down menu.

Table 389 Parameters of Network Setpoint (Continued)

Parameter Name	Description
Status	This field is read-only. Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendinterval (seconds)	The time between the end of a timeout period or the completion of a network request and the next data request on the network is referred to as the send interval. Cov is the default value.
category	The category is defined by the user while configuring Sylk parameters. The Category is displayed as the default 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, a tenant can view the parameters but cannot make any changes in the parameters. • If Tenant Read Write is selected, a tenant can view as well as modify the parameters.
enableFD	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.
allownullValue	Indicate whether null values for the adjustable value are permitted. An additional button press will send the null value when the LCD reaches a low or high limit when this option is enabled.

Table 389 Parameters of Network Setpoint (Continued)

<table>
<tr><th>Parameter Name</th><th>Description</th></tr>
<tr><td>tR7XConfig</td><td>

This option allows you to config the additional param options.

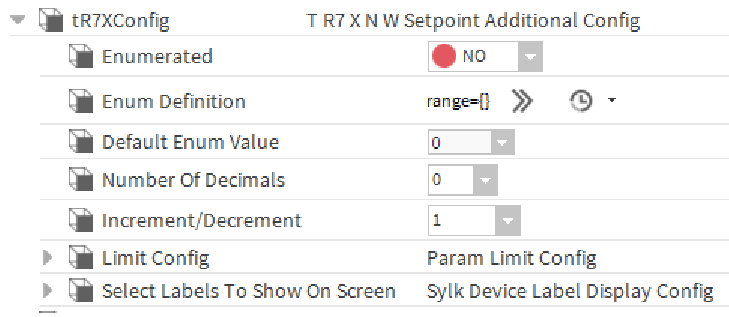


Fig. 562 Config additional parameters

- **enumerated:** Using this option allows enabling multiple states of the enum setpoint.
- **enumDefinition:** If the enumerated is set to “Yes,” then the user can set the multiple states of the Enum setpoint.

To set the Enum states of the Setpoint:

a. Click on the >> button.

Fig. 563 enumDefinition

b. Set the states **Ordinal, Tag, and Display** as per requirement. For example.

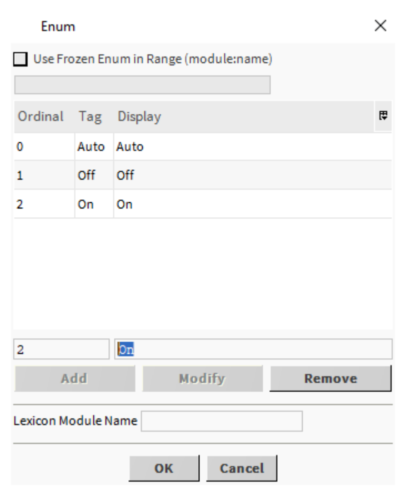


Fig. 564 Enum

c. Click **OK** to save.

Note: *The enumerated option applicable to Sylk module type is TR 71 and TR 75. Select the parameter type as Controller Value, Value from Wall Module, or Network Setpoint to enable the Enumerated.*

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

Table 389 Parameters of Network Setpoint (Continued)

<table>
<tr><th>Parameter Name</th><th>Description</th></tr>
<tr><td></td><td>

- **defaultEnumValue:** This option displays the list of default Enum values from the defined enum range.
- **Number of Decimals:** Apply to all parameter types except Time. This value affects Increment/Decrement options.
- **Increment/Decrement:** Depend on the Number of decimal places selected.

    If the number of decimal places is 0, then select 1, 5, 10, 100.

    If the number of decimal places is 1, then select 0.1, 0.5, 1,5, 10, 100.

    If number of decimal places is 2, then select 0.01, 0.1, 0.5, 1, 5, 10, 100.

Note: *The increment of 5 and 0.5 is only available for the Value from Wall Module, Network Setpoint, and Sensor Offset Value (Internal) parameters in TR71/TR75 model wall modules.*

- **Default value:** Indicate the value that will be initially downloaded to the wall module. This value must be between the Low and High limits. This field is displayed for the following parameter types:

    Value from wall module.

    Sensor offset value. By default, it is set to 0.
- **limitConfig:** This option allows to set the high & low limits of the parameter.

    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: To set low limit.

    High Limit: To set the high limit.

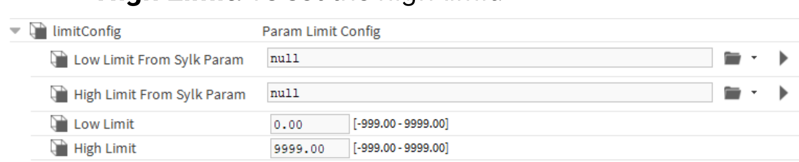
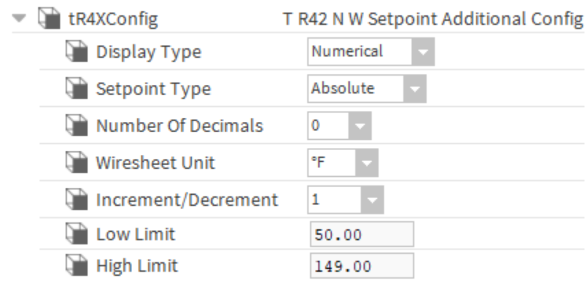


Fig. 565   limitConfig

- **selectLabelsToShowOnScreen :** This option allows to config the Sylk device label display for the Sylk parameter.

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

Table 389 Parameters of Network Setpoint (Continued)

Parameter Name	Description
tR4XConfig	This option allows you to config the additional param options.  **Fig. 566 tR4XConfig**
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **OUT:** To enable or disable the Out Save feature.

Note: *If you are using two network setpoints and if you connect the output of the first block to the input of the 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 the destination component to the source component so that the values are updated in the source component also.*

System Command

The System Command function block is used to override the following wall module to system commands. The wall module sends a command to the AHU or VAV system.

The following five-system command options are available on the wall module:

- Off / Heat (Heat only)
- Off / Cool (Cool only)
- Off / Heat / Cool (No Auto changeover)
- Off / Auto / Heat / Cool (Auto changeover)
- Off / Auto / Heat / Cool / Emergency Heat (Heat Pump)

It is also possible to override the system command output by making a change in the “In” field.

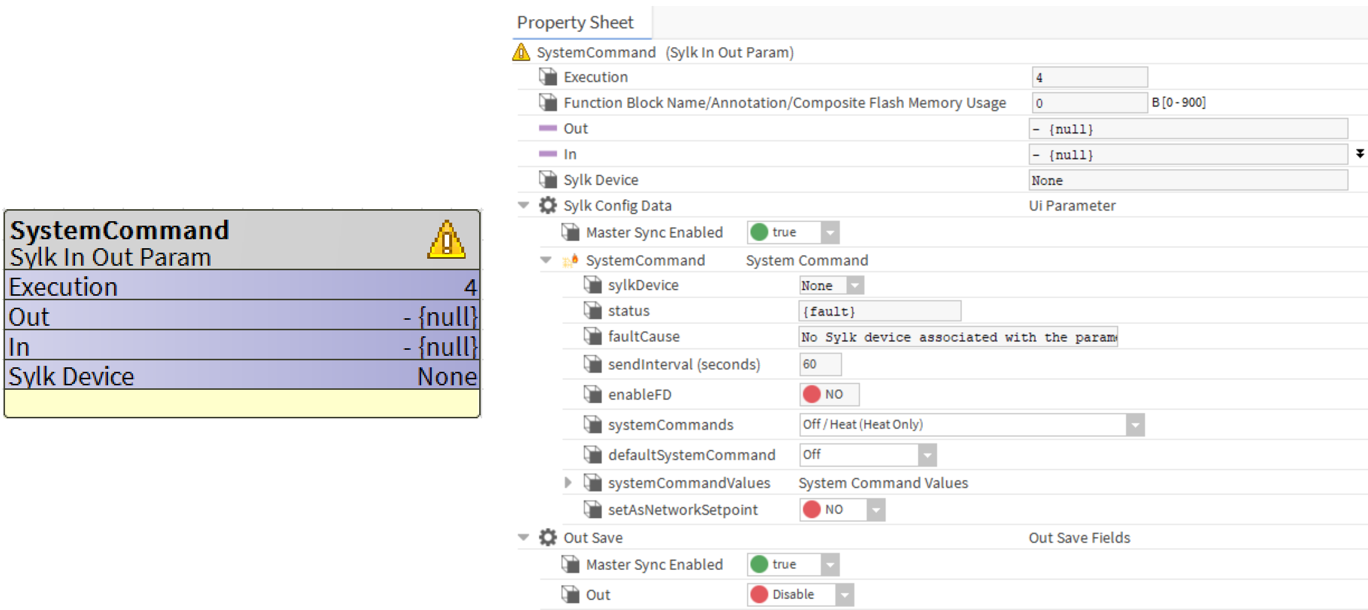


Fig. 567 SystemCommand Function Block and Property Sheet

Inputs

Table 390 Inputs of System Command

Input Name	Description
In	To provide input to the system command from the controller to the Wall module. The following five system command options are available on the wall module: • Off / Heat (Heat only) • Off / Cool (Cool only) • Off / Heat / Cool (No Auto changeover) • Off / Auto / Heat / Cool (Auto changeover) • Off / Auto / Heat / Cool / Emergency Heat (Heat Pump)

Output

Table 391 Output of System Command

Output Name	Description
Out	To provide the final system command from the wall module to the controller. The following five system command options are available on the wall module: • Off / Heat (Heat only) • Off / Cool (Cool only) • Off / Heat / Cool (No Auto changeover) • Off / Auto / Heat / Cool (Auto changeover) • Off / Auto / Heat / Cool / Emergency Heat (Heat Pump)

Parameters

Table 392 Parameters of System Command

Parameter Name	Description
sylkDevice	Select the required TR7x device from the drop-down menu.

Table 392 **Parameters of System Command (Continued)**

Parameter Name	Description
Status	This field is read-only. Shows the following status of the parameter: • alarm • fault • overridden • disabled • down • stale • null • unackedAlarm For further details, see Sylk Component Status Behaviors on page 532.
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendInterval (seconds)	Send 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. The default value is 60 seconds.
enableFD	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.

Table 392 Parameters of System Command (Continued)

<table>
<tr><th>Parameter Name</th><th>Description</th></tr>
<tr><td>systemCommands</td><td>The following system commands options are available on the wall module:

Off / Heat (Heat only)
Off / Cool (Cool only)
Off / Heat / Cool (No Auto changeover)
Off / Auto / Heat / Cool (Auto changeover)
Off / Auto / Heat / Cool / Emergency Heat (Heat Pump)

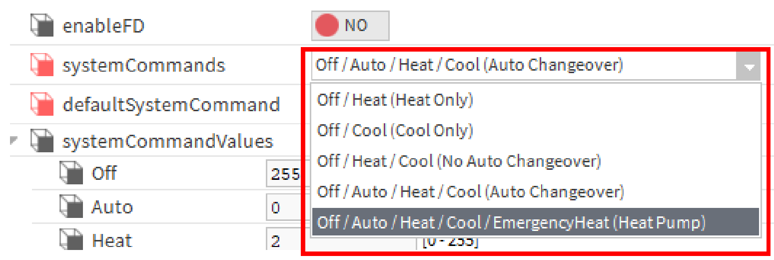


Fig. 568   Default Value is Off / Heat (Heat only).</td></tr>
<tr><td>defaultSystemCommands</td><td>Allows to set the default value for the system command options. Following five default system command options are available:

Off: Controller is off.
Auto: If the system command is set to Auto, the controller will automatically switch between "Cool" and "Heat" modes based on the value of the Room Temperature Sensor or Supply Air Temperature Sensor.
Heat: If the system command is set to Heat, the central air supply supplies heated air to network workstations or nodes, and heated air is supplied to the controlled space.
Cool: If the system command is set to Cool, the cool air is supplied to the controlled space via the central air supply from a network workstation or a node, and cooling energy is supplied to the controlled space.
Emergency Heat: If the system command is set to Emergency Heat, The central air supply provides emergency heat to network workstations or nodes.

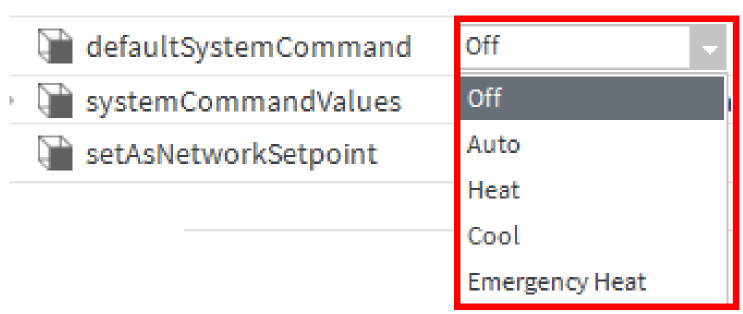
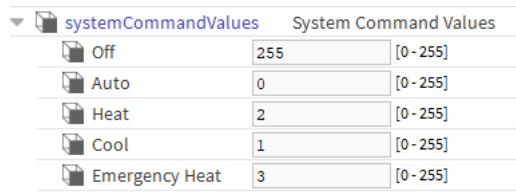


Fig. 569   defaultSystemCommands</td></tr>
</table>

Table 392 Parameters of System Command (Continued)

Parameter Name	Description
systemCommandValues	System command values display the various command values that are enabled based on the valid system commands that are selected. Incorrect values are disabled (depending on the system configuration options selected). You can switch between Off, Auto, Heat, Cool, and Emergency Heat modes. **Off:** 255 **Auto:** 0 **Heat:** 2 **Cool:** 1 **Emergency Heat:** 3 When the temperature is too cold for your heat pump to extract heat from the outside, the thermostat switches to emergency heat, also known as "auxiliary heat." When the temperature outside is 35 °F or lower, Emergency Heat is activated. When something happens to the first stage heat source, "heat pump," users should turn on emergency heat. **Numeric:** 32-bit integer (0, 1, 3, 255). Null is the default.  **Fig. 570 systemCommandValues Property Sheet**
SetAsNetworkSetpoint	Connect the System command block to the Network input or setpoint and the Network output point using this option. If this option is not selected, the System command block can only refer to the Network output point and not the Network input or set point. ***Note:*** *Set as Network Setpoint is disabled by default. The default value is No. By clicking YES or NO, you can enable or disable this option.* ***Note:*** *Set as Network Setpoint option is available only for the TR71, TR75, TR120 model.*
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **OUT:** To enable or disable the Out Save feature.

Time Field

The parameters display allows you to customize the time format of the wall module.

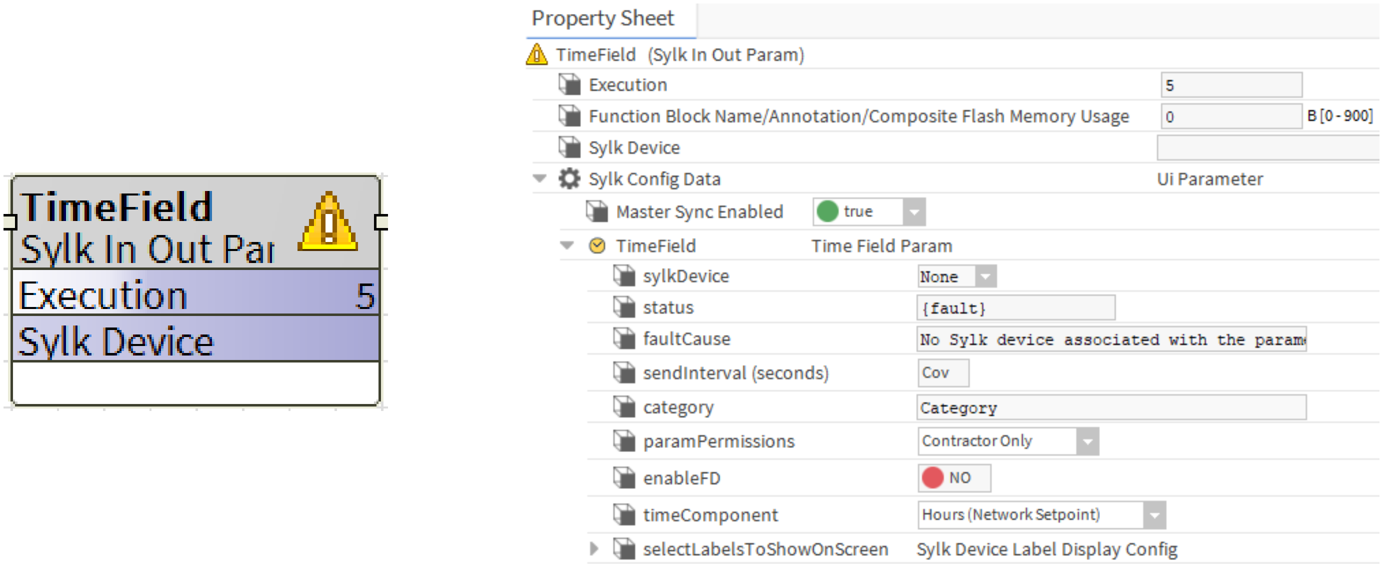


Fig. 571 TimeField Function Block and Property Sheet

Inputs

Table 393 Inputs of Time Field

Input Name	Description
In	To provide the controller time to the wall module.

Output

Table 394 Output of Time Field

Output Name	Description
Out	To provide the wall module time to the controller.

Parameters

Table 395 Parameters of Time Field

Parameter Name	Description
sylkDevice	Select the required TR7x device from the drop-down menu.

Table 395 Parameters of Time Field (Continued)

Parameter Name	Description
Status	This field is read-only. Shows the following status of the parameter: • alarm, fault, overridden, disabled, down, stale, null, and unackedAlarm. For further details, see Sylk Component Status Behaviors on page 532.
faultCause	This field is read-only. It indicates the cause of the parameter's fault state. This property is empty unless a fault occurs. When multiple errors occur, the fault cause displays only the top errors on the list. If the error is resolved, the next error message will be displayed.
sendInterval (seconds)	Send 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. Default is 60 seconds.
category	The category is defined by the user while configuring Sylk parameters. The Category is displayed as the default 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, a tenant can view the parameters but cannot make any changes in the parameters. • If Tenant Read Write is selected, a tenant can view as well as modify the parameters.
enableFD	The time it takes for a failure to be reported to the Sylk device. • **Yes:** If the parameter has not received an update from the BACnet network source within the fail detect time, an alarm is generated, and the current value is set to invalid. ***Note:*** *The time it takes to detect a failure is determined by the update rate that has been set.* • **No:** No means that the object will retain the last value written to it until the BACnet network source changes.
timeComponent	Set the time parameter as per requirement.
selectLabelsToShowOnScreen	To display the parameter on the home screen.
Out Save	• **Master Sync Enabled:** If you set it to TRUE, the last output will be set as output for one cycle after the controller restarts. The application can use it to return to the same state before the controller restart. • **OUT:** To enable or disable the Out Save feature.

Sylk Time

Allows to configure Sylk time.

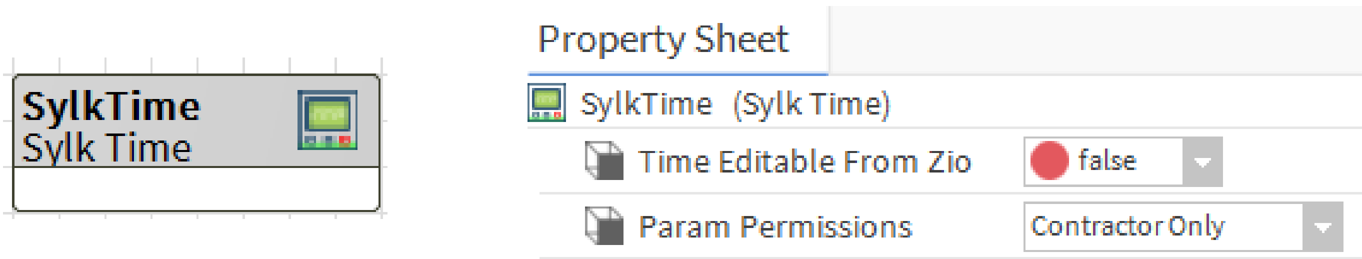


Fig. 572 SylkTime Function Block and Property Sheet

Input

Table 396 Inputs of Sylk Time

Input Name	Description
NA	NA

Output

Table 397 Output of Sylk Time

Output Name	Description
Out	To use the time of the wall module.

Parameter

Table 398 Parameters of Sylk Time

Parameter Name	Description
Status	Read-only point. It shows the status of the parameter. (alarm, fault, overridden, disabled, down, stale, null, unackedAlarm). For further details, see Sylk Component Status Behaviors.
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, a tenant can view the parameters but cannot make any changes in the parameters. • If Tenant Read Write is selected, a tenant can view as well as modify the parameters.

Home Screen

The home screen allows to configured the available options in the wall module.

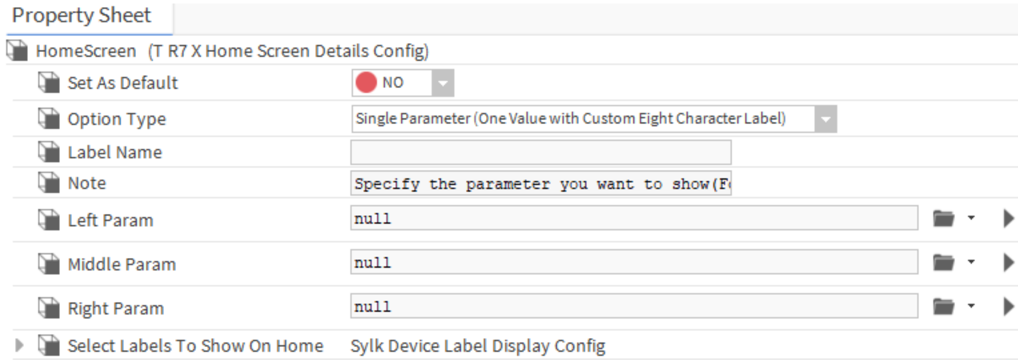


Fig. 573 HomeScreen Property Sheet

Steps to configure Home Screen parameter:

Step 1. Double-click homeScreenOptions present under the respective T R7 X module to configure the properties.

- **Set As Default:** Select this option if the current Home screen needs to be set as default.
- **Option Type:** Select the type of screen:
- Single parameter (one value with custom eight character label): This option displays one of the labels on the home screen.
- Multiple Parameter (Up To Three Values with Fixed Label): This option displays all three labels on the home screen.
- **Label Name:** Add the label name as per requirement.
- **Note:** Specify the parameter you want to show.
- **Left Parameter:** Select the parameter that needs to be shown on the left side of the home screen.
- **Middle Parameter:** Select the parameter that needs to be shown in the middle of the home screen.
- **Right Parameter:** Select the parameter that needs to be shown on the 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.

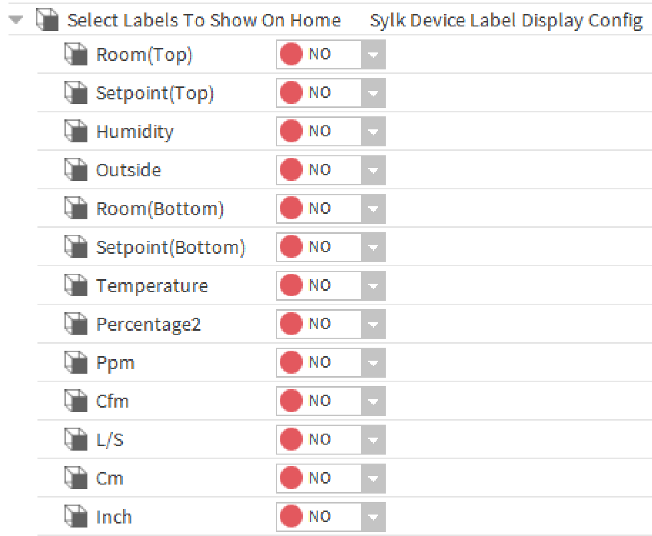


Fig. 574 Sylk Device Label Display Config

Step 2. Click **Save** to save the changes
or click **Refresh** if you do not want to save the changes.

Honeywell Building Technologies

715 Peachtree St NE Atlanta, Georgia
30308, USA
buildings.honeywell.com

Honeywell Products and Solutions

SARL, Z.A. La Pièce, 16, 1180 Rolle
Switzerland
buildings.honeywell.com

EN2B-0415GE51-R0423

Honeywell