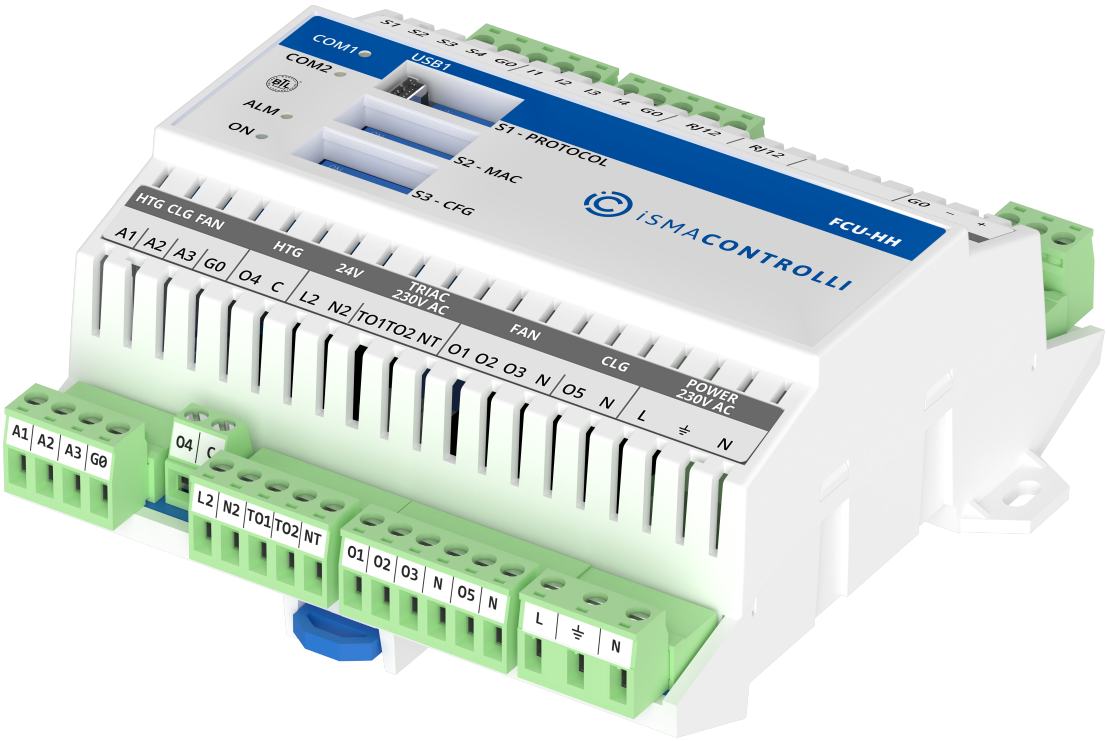
iSMACONTROLLI

iSMA-B-FCU

User Manual

Programming



iSMA CONTROLLI S.p.A. - Via Carlo Levi 52, 16010 Sant'Olcese (GE) - Italy | support@ismacontrolli.com

Table of Contents

1 Introduction

This user manual describes programming of the iSMA-B-FCU controller.

1.1 Revision History

Rev.	Date	Description
1.5	19 Jul 2024	• Updated iSMA Room Devices Modbus kit components
1.4	21 Feb 2023	• Corrections in the iSMA Room Devices Modbus kit • FP panel references • Editorial corrections
1.3	21 Apr 2022	• Rebranded • Updated iSMA Room Devices Modbus kit • Touch Point panel references
1.2	27 Jan 2020	• The replaced environment of programming from the Workplace to iC Tool • Added description of the iSMA Building kit
1.1	20 Aug 2017	The reason for the creation of the new version of the document: • Rebuilt main algorithm of FCU default application (several components of iSMA_FCU kit rebuilt). • New functionality added (new components to iSMA_FCU kit). Changes in Document: • The new way of switching Heating and Cooling Modes by FCU_HeatingCoolingSwitch component. • Added: description of FCU_PI component • Added: description of FCU_PID component • Added: description of FCU_TemperatureAnalog component • Added: description of FCU_TemperatureBinary component • Added: description of FCU_ValueHolder component • Added: description of FCU_WindowStatusSwitch component • Added: description of PWM component • Added: description of Room Device kit • Added: description of Advance Control kit
1.0	20 Dec 2016	First edition

Table 1. Revision history

2 Overview

Each new iSMA-B-FCU device is equipped with a default application, firmware, and kits. The default application can be modified for individual purposes, used with no changes, or the user can create a new, custom application.

Modifying the default application, or creating a new one, can be done only online (in real time), using the SOX protocol and iC Tool. The size of application cannot exceed 64 kB. Available memory can be checked in the Mem Available slot (under the plat component).

Using the FCU Updater, the modified or created application can be downloaded from one iSMA-B-FCU device and uploaded to other iSMA-B-FCU device(s). The iSMA-B-FCU device has two built-in RS485 ports:

- **COM1:** port with the screw connector; the port can be used for communication using the Modbus RTU/ASCII or BACnet protocol (including BACnet client-server communication);
- **COM2:** port with two RJ12 connectors; the port can be used for communication using the Modbus Async protocol.

Each iSMA-B-FCU device has a set of kits, which are installed with the firmware. These kits are required for proper operation of default application, and can be also used to develop a custom application. The kits cannot be changed or delayed. The iSMA-B-FCU device is equipped with the following kits:

- **sys:** the Sedona core system module;
- **control:** basic function blocks library;
- **inet:** IP and UDP/TCP socket APIs;
- **sox:** the Sox service for remote management;
- **iSMA_BACnetMasterSlave:** the kit for master-slave communication.

The iSMA-B-FCU device can work in defined groups, where one device is master and the remaining devices (slaves) follow the master parameters. This way, it is possible to share up to 100 points. This function is available only in the BACnet MS/TP protocol, using the RS485 port (COM1). Each master device can have up to 5 slave devices.

- **iSMA_FCU:** the kit includes components used to develop the FCU application; it consists of components controlling temperature outputs, fan, etc. ;
- **iSMA_ModbusAsyncNetwork:** the kit includes components for the Modbus Async communication;

The Modbus Async can be used to communicate with other devices connected to the built-in RS485 port (COM2). It is possible to read/write up to 200 points this way. There is no restriction about the number of connected devices.

- **iSMA_platFCU:** the kit includes components for all types of inputs and outputs servicing, components for communication with the higher-level system (using Slave Network component), and NV components.

The SlaveNetwork component is used to manage the BACnet MS/TP or Modbus RTU/ASCII protocol, using the RS485 port (COM1). The SlaveNetwork component allows for sharing of up to 200 Analog Value and up to 128 Binary Value objects. A total number of memory cells for NV numeric (and integer) components cannot exceed 200. A total number of memory cells for NV Boolean components cannot exceed 128.

Note: Method of calculating memory cells for NV components is described in the Plat service chapter.

The iSMA-B-FCU device has 18 built-in physical inputs and outputs:

- 4 special inputs;
- 4 digital inputs;
- 3 analog outputs;
- 5 digital outputs;
- 2 triac outputs.

The device is equipped with a S3 - CFG DIP switch, which allows to manage the eight binary signals. Using the DIP switch is recommended to manage the configuration of the device.

The device is also equipped with the alarm LED (ALM), which allows for signalizing states of the iSMA-B-FCU device predefined in the application. For example, it can be used for signalizing alarms.

The components for servicing of all inputs, outputs shown below are placed under the iSMA_platFCU palette.

3 Programming in iC Tool

As a significant part of the end-to-end iSMA CONTROLLI solution, the iC Tool gives the customer a convenient way to create and manage custom applications for the Sedona-based controller.

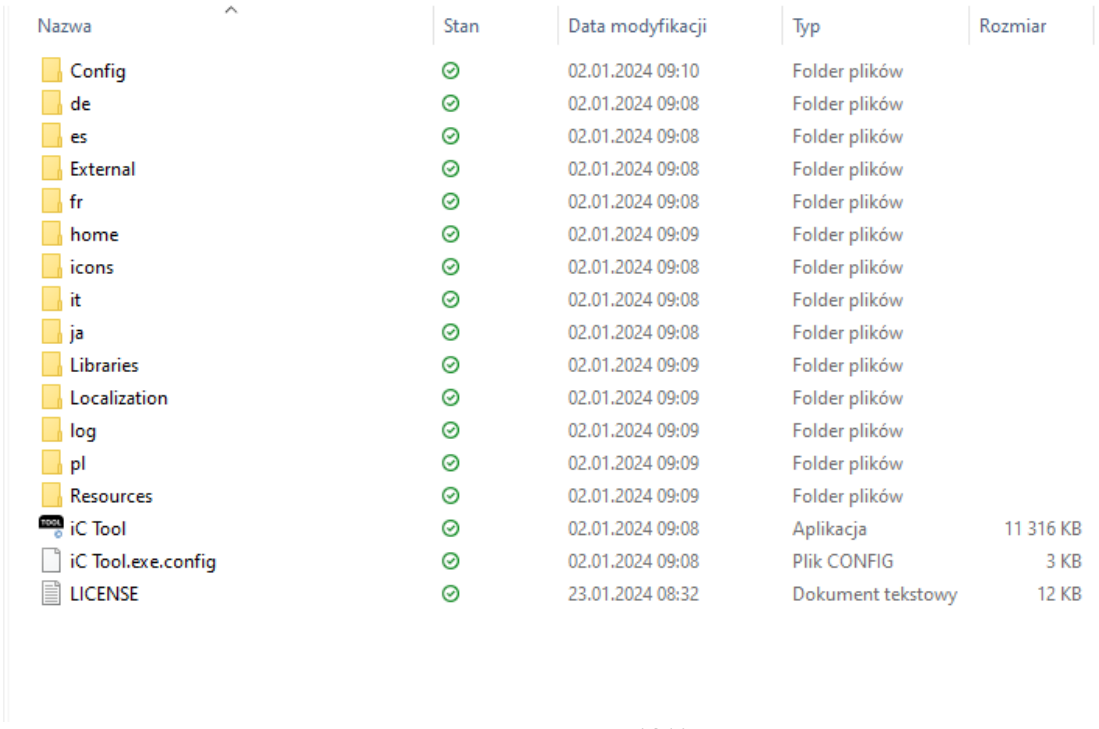
The iC Tool covers all requirements to create and manage applications: it has a wire sheet for convenient visual programming, property sheets for details; it offers kits management, real-time monitoring of system states and slots values, logs and historical data, deployment and backup.

3.1 iC Tool Installation

The iC Tool is a software created for Microsoft Windows system. The oldest supported version of the operating system is Windows 7. The iC Tool is delivered as a compressed folder, which needs to be extracted in a chosen location on a hard drive, unless the access to the extracted folder is restricted by the system (e.g., Program Files is not a recommended location).

In order to download the iC Tool software bundle, which includes all files necessary to run the program efficiently (a zipped file iCTool_Vx.x.x.zip), go to the iSMA CONTROLLI web page ismacontrolli.com and to the iC Tool/Software Bundle folder.

Extracting the zipped package reveals the folders and additional files described below. In order to run the iC Tool, open the iCTool.exe file.



Nazwa	Stan	Data modyfikacji	Typ	Rozmiar
Config	✓	02.01.2024 09:10	Folder plików	
de	✓	02.01.2024 09:08	Folder plików	
es	✓	02.01.2024 09:08	Folder plików	
External	✓	02.01.2024 09:08	Folder plików	
fr	✓	02.01.2024 09:08	Folder plików	
home	✓	02.01.2024 09:09	Folder plików	
icons	✓	02.01.2024 09:08	Folder plików	
it	✓	02.01.2024 09:08	Folder plików	
ja	✓	02.01.2024 09:08	Folder plików	
Libraries	✓	02.01.2024 09:09	Folder plików	
Localization	✓	02.01.2024 09:09	Folder plików	
log	✓	02.01.2024 09:09	Folder plików	
pl	✓	02.01.2024 09:09	Folder plików	
Resources	✓	02.01.2024 09:09	Folder plików	
iC Tool	✓	02.01.2024 09:08	Aplikacja	11 316 KB
iC Tool.exe.config	✓	02.01.2024 09:08	Plik CONFIG	3 KB
LICENSE	✓	23.01.2024 08:32	Dokument tekstowy	12 KB

Figure 1. iC Tool folders

The extracted folders have the following functions:

- **Config:** a folder containing a record of user's individual settings regarding windows location and other iC Tool work settings, such as a language chosen for the iC Tool interface.
- **External:** a folder containing an API .dll file.

- **home:** a folder where all the data created by user are saved, i.e., device backups, applications, etc. It is also a folder where the kits library, available the in iC Tool, is located.
- **icons:** a folder with graphical files such as the iC Tool interface icons.
- **Localization:** a folder with the text files providing the iC Tool language sources.
- **log:** a folder, where the logs of the iC Tool, which also appeared in Console window, are saved. When contacting iSMA CONTROLLI technical support, it is advised to copy the last file with logs form that folder
- **de, es, fr, it, ja, pl, Resources, ru:** folders with system libraries.

To properly install and work with the iC Tool the computer must meet the following minimal requirements:

- processor (CPU): Intel Core i3-3xxx or equivalent;
- memory: 4 GB RAM;
- storage: 50 GB internal hard driver;
- Ethernet 100 Mbit or 1 Gbit NIC;
- MS Windows 7 (recommended MS Windows 10);
- .NET Framework 4.6.2 or higher.

WARNING! If the iC Tool is being run for the first time, it asks to accept the EULA license. The license must be accepted to run the program. Failure to do so closes the iC Tool.

Note: For the iC Tool to work properly it needs to be run periodically at least once a month, on a computer connected to the Internet for about an hour, depending on the data transfer rate. It enables the iC Tool to automatically download the latest data, such as kits and updates.

The iC Tool is a portable software. It is transferable and it can be installed on a portable data storage device, such as a USB memory stick. It allows the iC Tool to be run directly from a portable data storage device on any PC, including offline ones.

3.2 Connecting to Device

WARNING! Before starting to program the iSMA-B-FCU device using the SOX protocol, it is recommended to connect the device to 230 V AC power supply.

- Before connecting the iSMA-B-FCU device to the iC Tool, run the FCU Updater.

Connect the iSMA-B-FCU device to the USB port in your computer–SOX and Console buttons should be active, which means that the FCU Updater is communicated with the iSMA-B-FCU device.

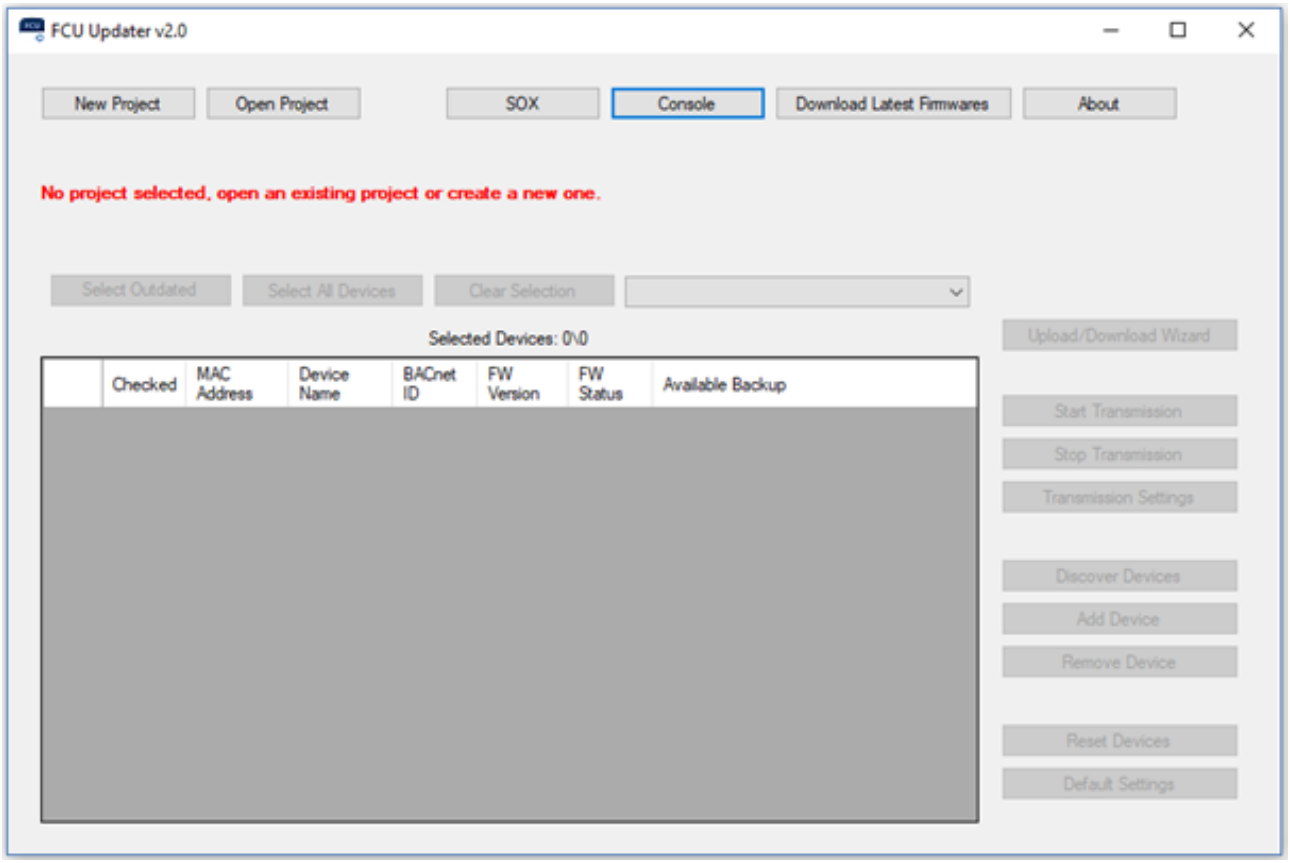


Figure 2. FCU Updater

- Click the SOX button to open a pop-up window, and then click the Begin Communication button.

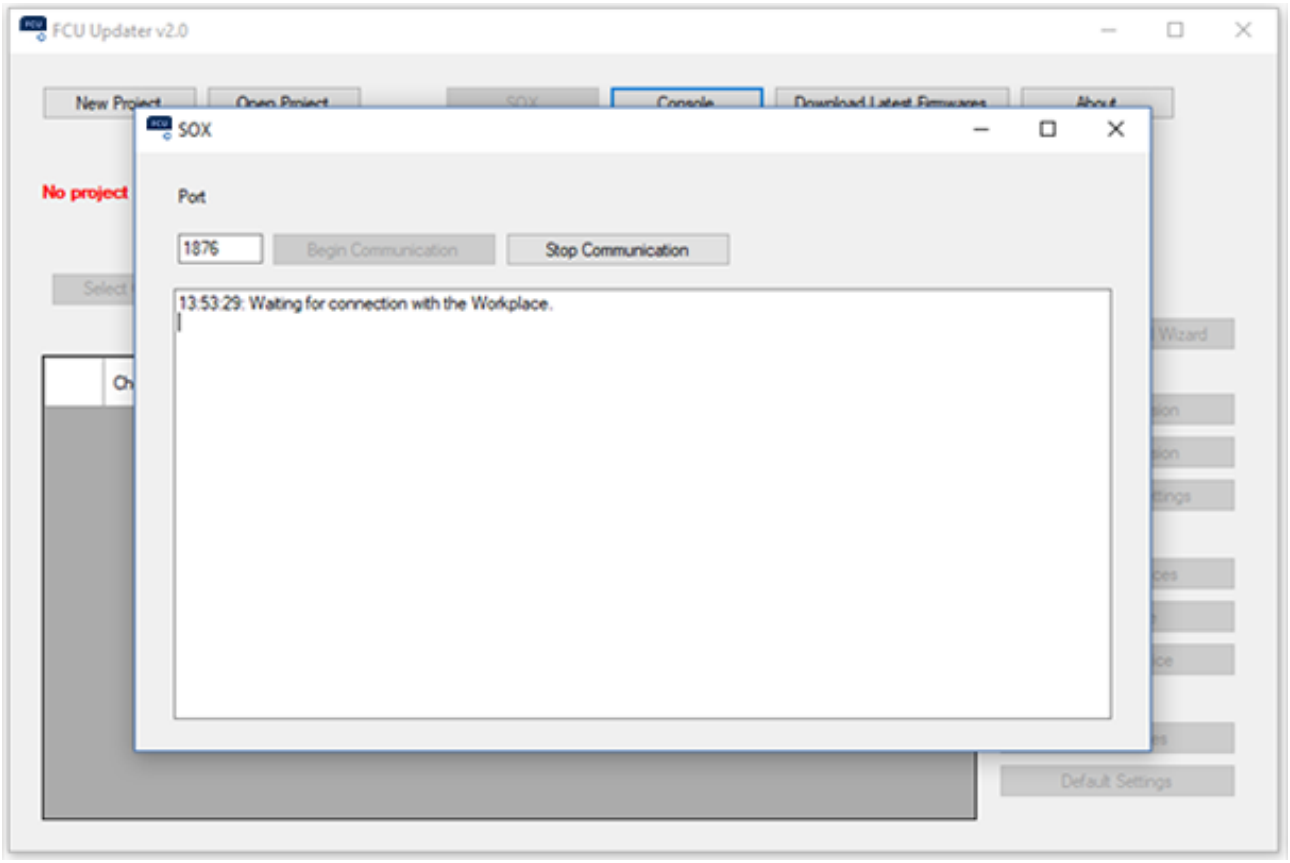


Figure 3. Beginning communication in the FCU Updater

- Open the iC Tool.
- In order to connect to the device, select the following options:

Right-click on the project folder and select the Add Device option.

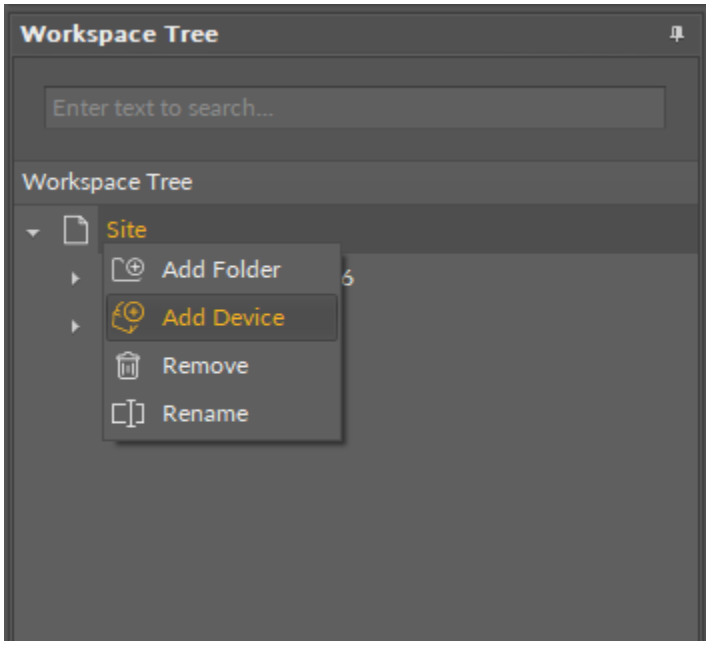


Figure 4. Adding device

Note: The newest version of the iC Tool should be installed on the PC for connection with the newest version of the iSMA-B-FCU.

- Upon selecting the above option, an authorization window pops up.

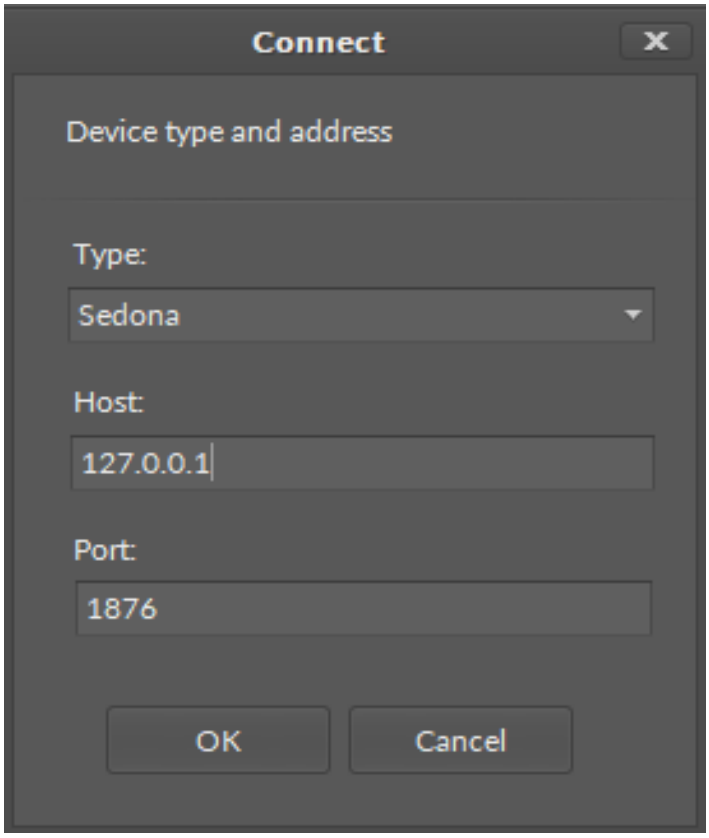


Figure 5. Connect window

Please enter the following values:

- **type:** Sedona;
- **IP address:** 127.0.0.1;
- **port:** 1876.

Log in using the following username and password:

- **username:** admin
- **password:** empty box (no characters).

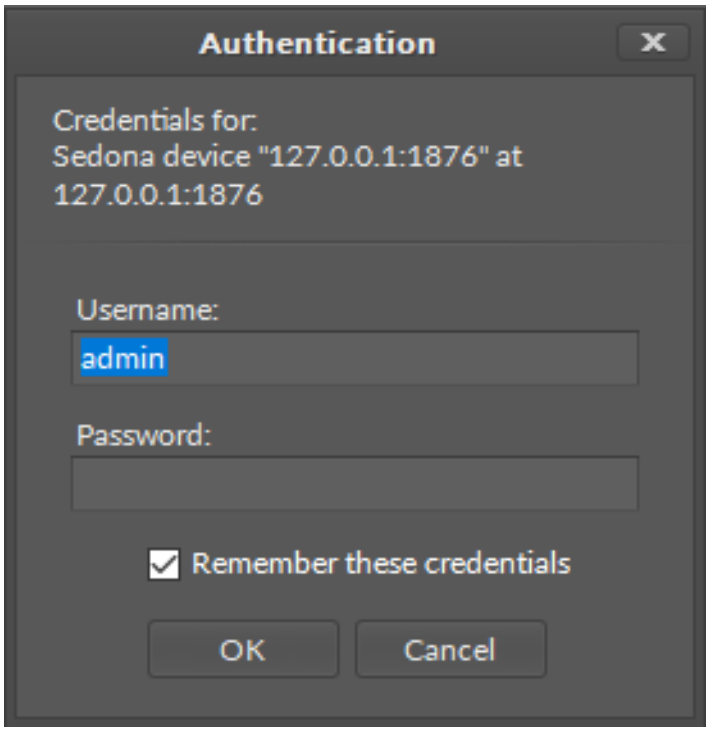


Figure 6. Authentication window

4 App

The application consists of services and components available from the palettes. Components are processed in every working cycle of the device. Services cover certain components enabling system functions, such as user management. All items should be placed under the App main component. If the application has been modified, the App icon is displayed with a reminder that the application should be saved. There is a possibility to turn on the autosave of the application.

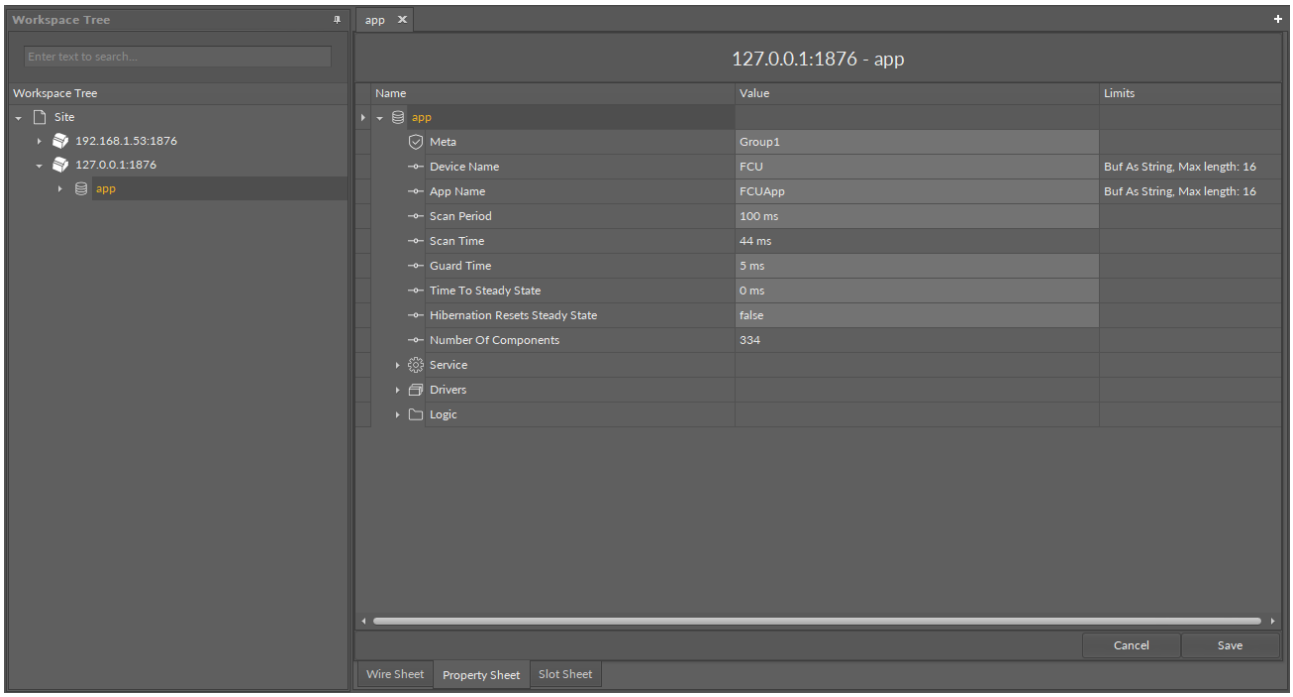


Figure 7. The app component

The app component has the following slots:

- **Device Name:** allows to set the device name:
- **App Name:** allows to set the application name;
- **Scan Period:** allows to set one cycle execution time;
- **Scan Time:** shows the real-time of one cycle execution;
- **Guard Time:** allows to set the reserve time to finish system tasks;
- **Time To Steady State:** allows to set the time from app start to steady-state;
- **Hibernation Resets Steady State:** not active in the iSMA-B-FCU;
- **Number Of Components:** shows the number of components used in the application.

The app component offers the following actions, available in the context menu:

- **Save:** saves the Sedona application in the device's flash memory;
- **Restart:** restarts the application (Sedona Virtual Machine);
- **Reboot:** reboots the device;
- **Quit:** closes SOX connection;
- **Hibernate:** not active.

The app component has to be the parent component for the following components:

- Service;
- Drivers.

These components are described in the following sections.

4.1 Logic

The Logic Folder is a folder for grouping components used in developing the logic of the application, such as function blocks, NV components, etc. These components can be placed anywhere under the app component, but it is recommended to place them in the Logic folder–it allows for easier interpretation of all applications.

The Logic folder is a component available from the sys kit palette; the Folder component is a component with no properties, and it is used exclusively to arrange applications visually.

Grouping of the Logic components can be also done in the Rate Folder component, also available from the sys kit palette. The Rate Folder component is a folder used to reduce the rate of application performance. While developing major applications, it allows reducing demand for processing power, consuming it for major or quick functions of the device algorithm. The folder's parameter "App Cycles To Skip" defines how frequently, in a number of scanning cycles, components under the folder are processed.

4.2 Drivers

The Drivers component has been created to manage networks used by the application.

All components responsible for networks and components associated with them, used by the iSMA-B-FCU device, have to be placed under the Drivers component. It could be Local IO Network (which manages physical inputs and outputs built-in the iSMA-B-FCU device), or networks responsible for protocols, which allow communicating with other devices (in case of the iSMA-B-FCU device: Modbus Async, Slave Network, and BACnet Master-Slave).

The Drivers component has to be placed under the app component.

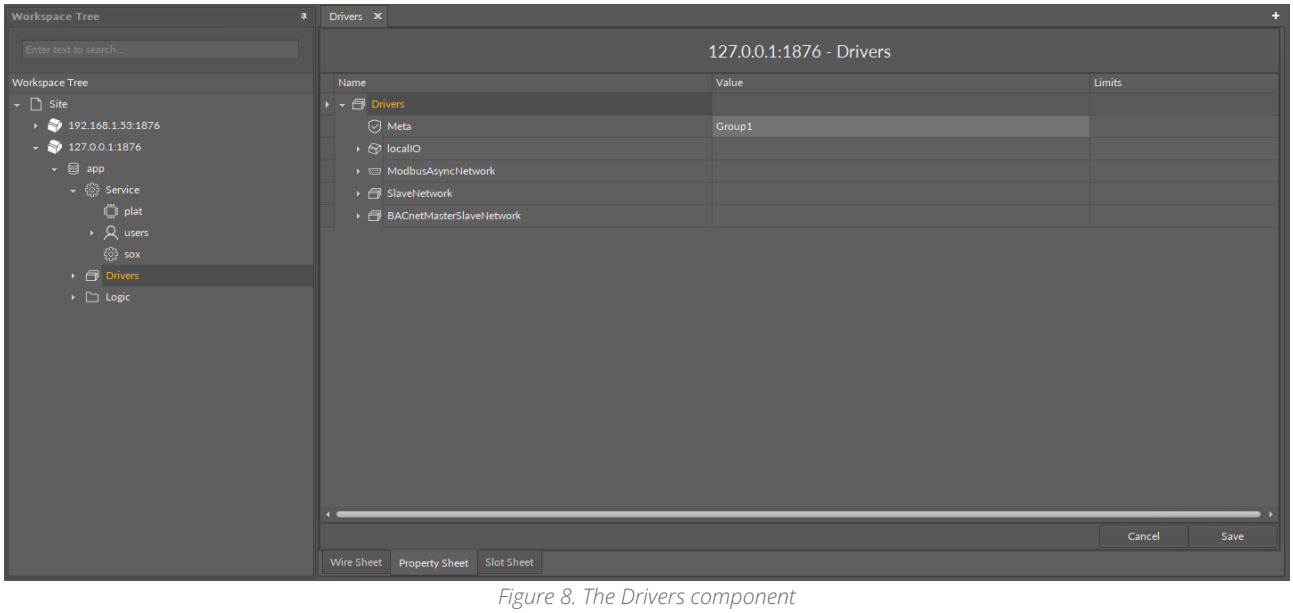


Figure 8. The Drivers component

4.3 Service

The Service component has been created to manage services. This component is a parent component for components, which do not take a direct part in the operation of the application (plat, users). The Service component has to be placed under the app component.

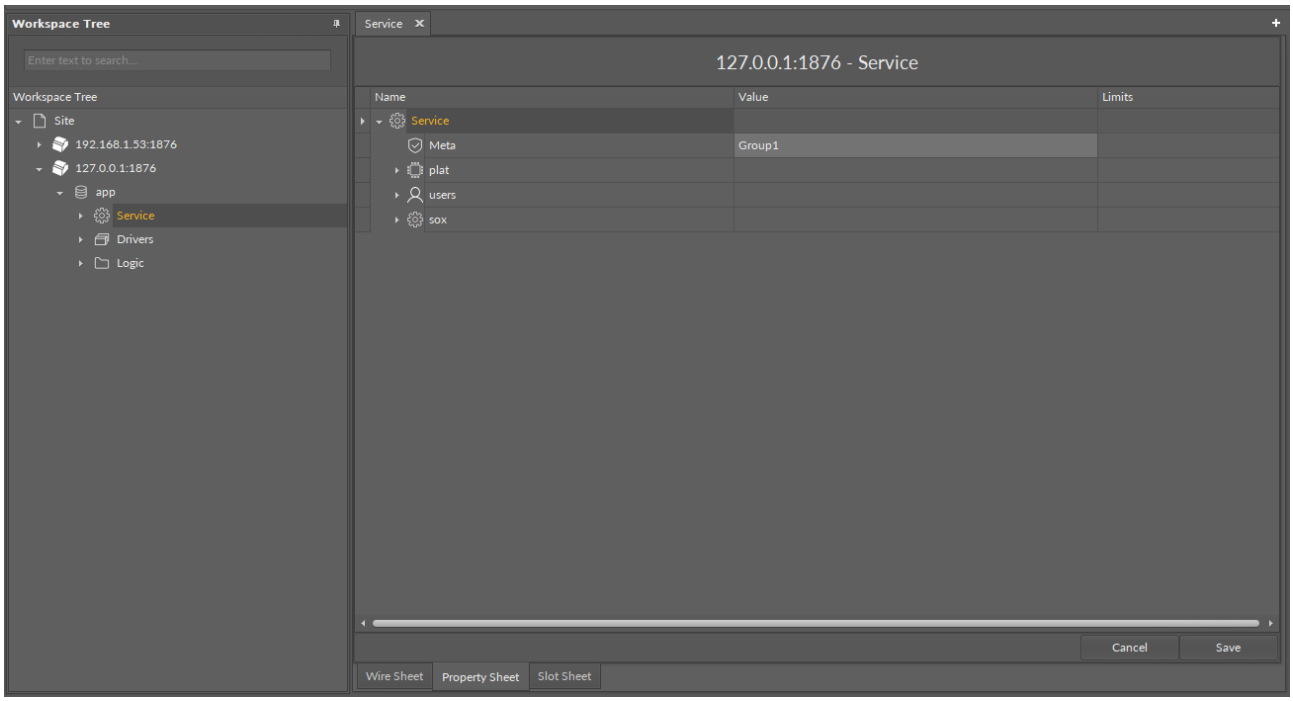


Figure 9. The Service component

4.3.1 users

The users component is responsible for the user support. It allows to add and remove users as well as define their access rights to individual components. Each Sedona application component has a Meta slot, used to assign it to one or more groups. Sedona has 4 predefined groups. The component has to be placed under the Service component.

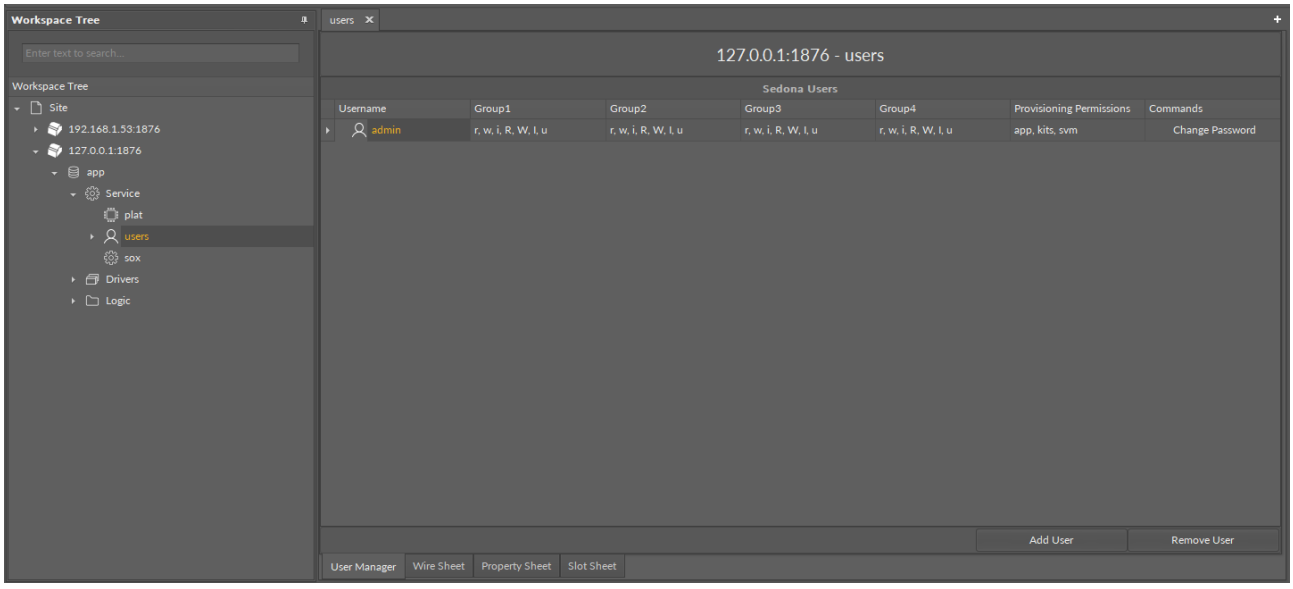


Figure 10. The users component

Users can have the following types of rights:

- **Operator Read:** allows to read components and read values of operator properties;
- **Operator Write:** allows to change values of operator properties;
- **Operator Invoke:** allows to invoke operator actions;
- **Admin Read:** allows to read values of properties, read links, and create components links;

- **Admin Write:** allows to change values' properties, add components, sort dub components, rename components, create links to components, delete links to components;
- **Admin Invoke:** allows to invoke admin actions of components;
- **Admin User:** allows to manage users (read, write, edit, delete).

Provisioning Permissions:

- **Can provision app:** can read/write app.sab file,
- **Can provision kits:** can read/write kits.scode file,
- **Can provision svm:** can read/write SVM files.

4.3.2 sox

The SOX protocol is a standard protocol used to communicate with Sedona devices. In this case, SOX is used to communicate the iSMA-B-FCU device with the iC Tool, using a USB connection.

Note: SOX is a service type protocol, executed after application components. If there is little time difference between the Scan Period and Scan Time values (see the App component), the services do not have sufficient time to be executed, and the programming interface can slow down. SOX is designed to be run over the UDP/IP protocol (default port 1876), but in case of the iSMA-B-FCU device, the SOX protocol runs over a USB interface.

The sox component has to be placed under the Service component.

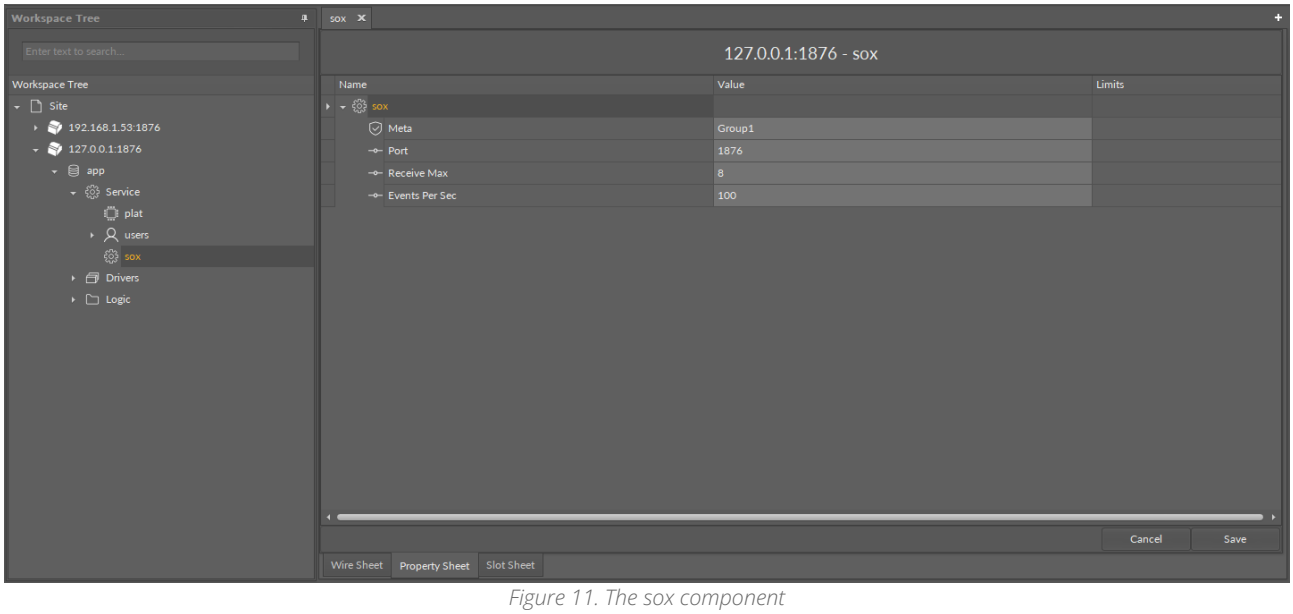


Figure 11. The sox component

The sox component has the following slots:

- **Port:** Sox UDP port (default value 1876);
- **Receive Max:** the maximum number of messages in the receiving window;
- **Events Per Sec:** the maximum number of Async events (telegrams) sent per second.

4.3.3 plat

The plat component shows the device's main parameters. This component is placed under the Service component, and is associated with the device hardware.

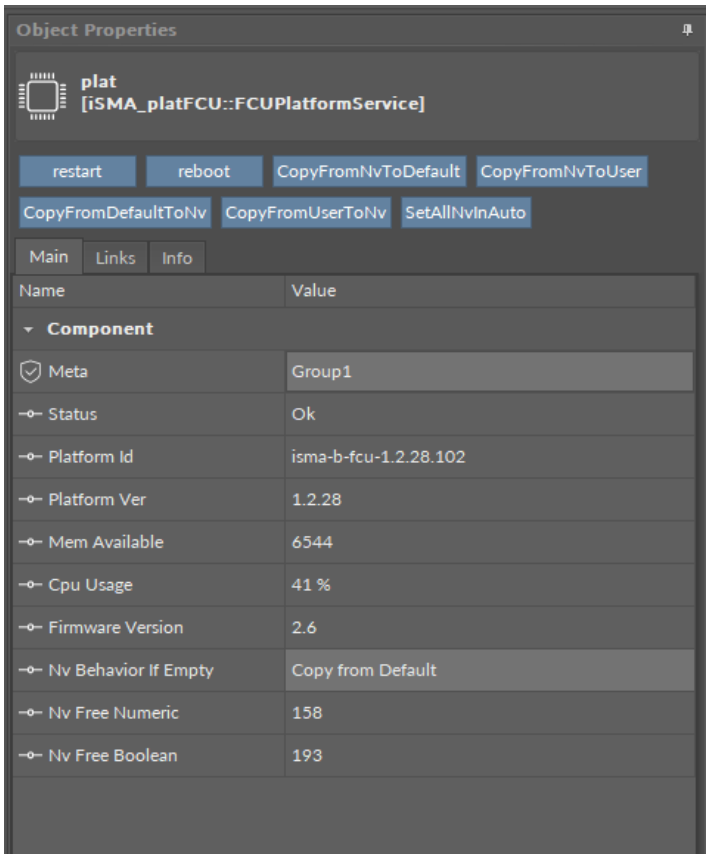


Figure 12. The plat component

The plat component has the following slots:

- **Status:** shows the platform status;
- **Platform ID:** shows the platform ID;
- **Platform Ver:** shows the platform version;
- **Mem Available:** shows the RAM memory available in the controller;

Note: The whole application for the iSMA-B-FCU device cannot exceed 64 kB.

- **Cpu Usage:** shows the CPU usage from the last 5 seconds;
- **Firmware Version:** shows the controller's firmware version;
- **Nv Behavior If Empty:** if the non-volatile memory is empty after copying the NV component, the output value will be 0 (Leave 0 option) or it will be copied from the Default slot (Copy From Default option); the default option is to copy the value from the Default slot;
- **Nv Free Numeric:** shows the number of available numeric non-volatile components;
- **NV Free Boolean:** shows the number of available Boolean non-volatile components.

Note: Number of components physically added to the application does not have to be equal to memory cells. It can be calculated in the way shown below:

- Each added NV Numeric (or Integer) component (not NV Net component) uses two memory cells–one numeric cell for value and one Boolean cell for Hand/Auto mode;
- Each added NV Boolean component (not NV Net component) uses two Boolean memory cells–one for value and one cell for Hand/Auto mode;
- Each NV Net Numeric component uses one numeric cell–these components work only in the Auto mode, so the memory cell is required only for the value;
- Each NV Net Boolean component uses one Boolean cell–these components work only in the Auto mode, so the memory cell is required only for the value;

- For Example, if 2 NV Numeric components, 3 NV Boolean components, 4 NV Net Numeric components, and 5 NV Net components are added to the application, the usage of memory cells can be calculated as follows:

Type of Component	Number of Components	Used Boolean Memory Cells	Used Numeric Memory Cells
NV Numeric	2	2 (for Auto/Hand modes)	2 (for values)
NV Boolean	3	6 (3 for Auto/Hand modes and 3 for values)	0
NV Net Numeric	4	0	4 (for values)
NV Net Boolean	5	5 (for values)	0
Total		13	6

Table 2. Memory cells calculation

As presented in the table, this application uses 13 Boolean memory cells and 6 numeric memory cells. In this case, the Free NV Boolean and Free NV Numeric slots (under the plat component) will display:

- Free NV Boolean: 227

227 = 240 (available Boolean memory cells) – 13 (Boolean memory cell used in application)

- Free NV Numeric: 224

224 = 230 (available Numeric memory cells) – 6 (Numeric memory cell used in application)

The plat component offers the following actions, available in the context menu:

- **Restart:** restarts the application (Sedona Virtual Machine);
- **Reboot:** reboots the device;
- **Copy From Nv To Default:** copies values from the Out slot to the Default slot in all NV components (see the NV component chapter);
- **Copy From Nv To User:** copies values from the Out slot to the User slot in all NV components (see the NV component chapter);
- **Copy From Default To NV:** copies values from the Default slot to the Out slot in all NV components (see the NV component chapter);
- **Copy From User To NV:** copies values from the User slot to the Out slot in all NV components (see the NV component chapter);
- **Set All Nv In Auto:** sets all NV components in the auto mode.

5 iSMA platFCU Kit

The iSMA platFCU kit contains components for servicing physical inputs and outputs of the iSMA-B-FCU devices. It also contains non-volatile components and non-volatile net components for servicing the slave network. All these components are described in the sections below.

5.1 LocalIO

The iSMA-B-FCU device is equipped with 18 physical inputs and outputs. To learn more about the device's inputs and outputs please see: Inputs and Outputs.

5.1.1 LocalIO Component

The LocalIO is the main component servicing the physical inputs and outputs. Under this component, all components used to reading or setting inputs or outputs have to be placed. The LocalIO component has to be placed under the Drivers component.

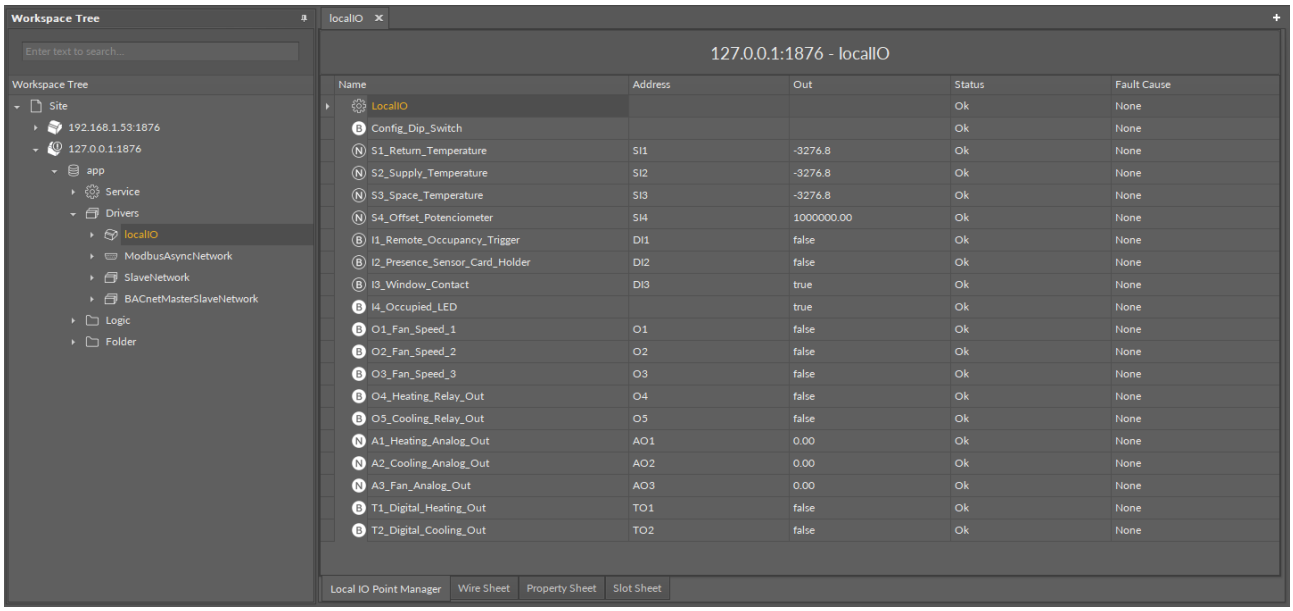


Figure 13. The LocalIO component

5.1.2 Watchdog

The Watchdog is a component designed for controlling communication by the RS485 port (COM1) and the USB (only for communication with the iSMA Tool using the SOX protocol or, in case of read/write Modbus registers, using the USB connection). The Watchdog component has to be placed under the LocalIO component.

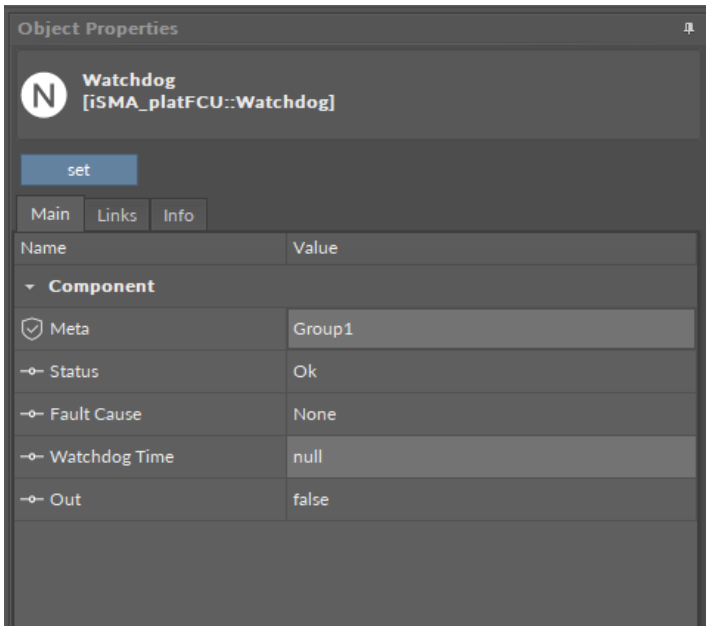


Figure 14. The Watchdog component

The Watchdog component has the following slots:

- **Status:** indicates the status of the Watchdog component;
- **Fault Cause:** shows the fault cause description;
- **Watchdog Time:** sets the time between the reception of valid messages, the 0 value disables this function;
- **Out:** the output of the Watchdog. If the Watchdog is disabled (the Watchdog Time slot is set to 0), or the time between the reception of valid messages does not exceed the Watchdog Time, the Out slot is set to false. If the time between the reception of valid messages does exceed the Watchdog Time, the Out slot is set to true.

The Watchdog component offers the following action, available in the context menu:

- **Set:** writes a value to the Watchdog Time slot.

5.1.3 LocalIOFolder

The LocalIOFolder is a folder dedicated to grouping I/O components.

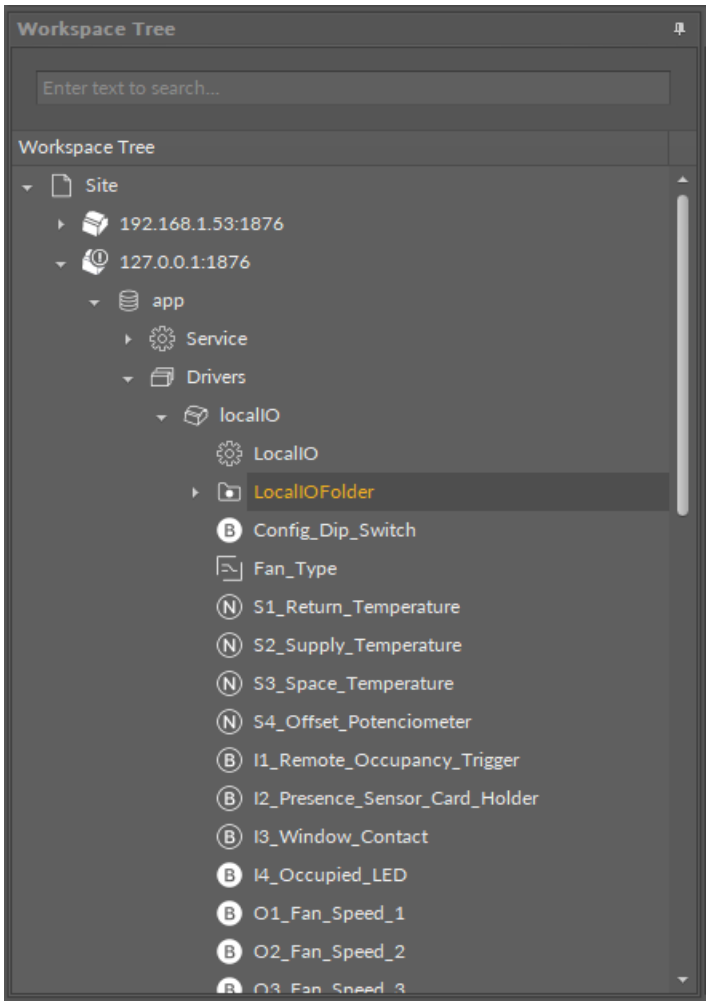


Figure 15. The LocalIOFolder component

5.1.4 DO

The DO is a component designed for servicing digital output. The DO component has to be placed under the LocalIO component.

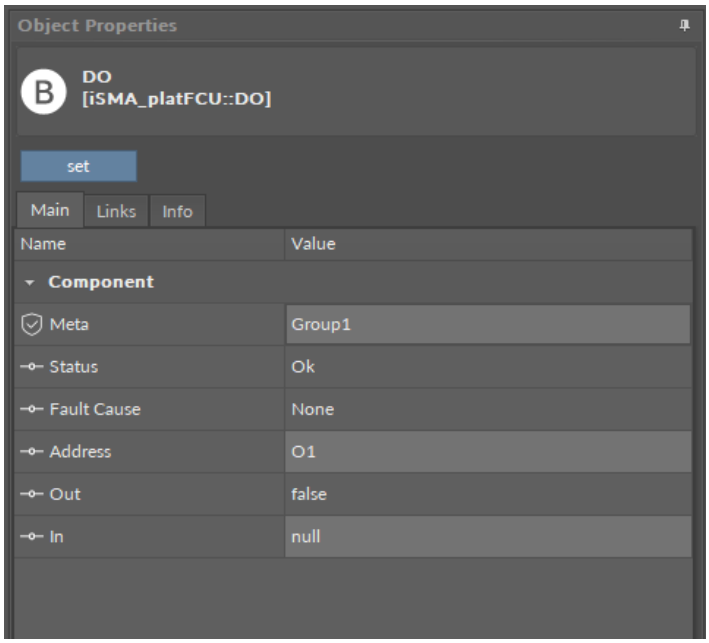


Figure 16. The DO component

The DO component has the following slots:

- **Status:** indicates the status of the DO component;
- **Fault Cause:** shows the fault cause description;
- **Address:** sets the number of the physical output of the iSMA-B-FCU device (O1, O2, O3, O4, O5);
- **Out:** displays the actual state of the output (true/false);
- **In:** the input slot.

The DO component offers the following action, available in the context menu:

- **Set:** writes a value to the Out slot, and sends it to the device.

5.1.5 SIDigital

The SIDigital is a component designed for servicing special inputs in the dry contact read mode. The SIDigital component has to be placed under the LocalIO component.

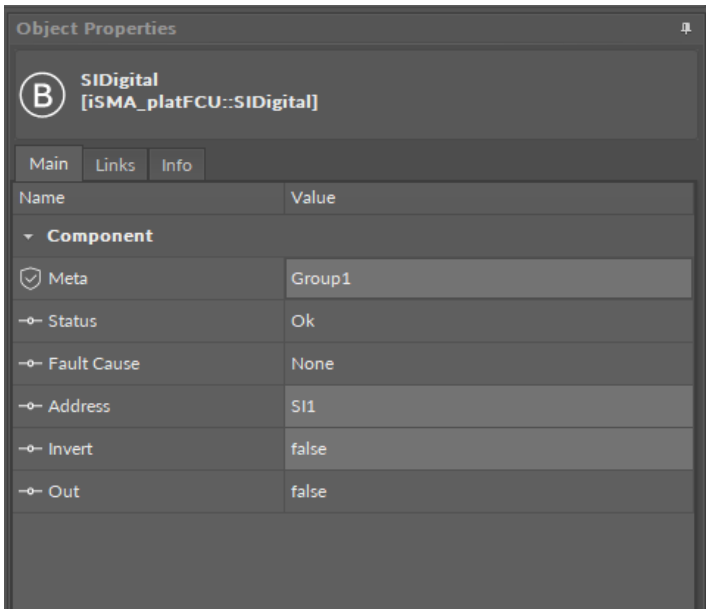


Figure 17. The SIDigital component

The SIDigital component has the following slots:

- **Status:** indicates the status of the SIDigital component;
- **Fault Cause:** shows the fault cause description;
- **Address:** sets the number of the physical input of the iSMA-B-FCU device (SI1, SI2, SI3, SI4);
- **Invert:** shows the value negative to the value read from the special input (in dry contact mode) of the iSMA-B-FCU device,
- **Out:** displays the actual value of the input (true or false).

5.1.6 SIResistance

The SIResistance is a component designed for servicing special input in the resistance read mode. The SIResistance component has to be placed under the LocalIO component.

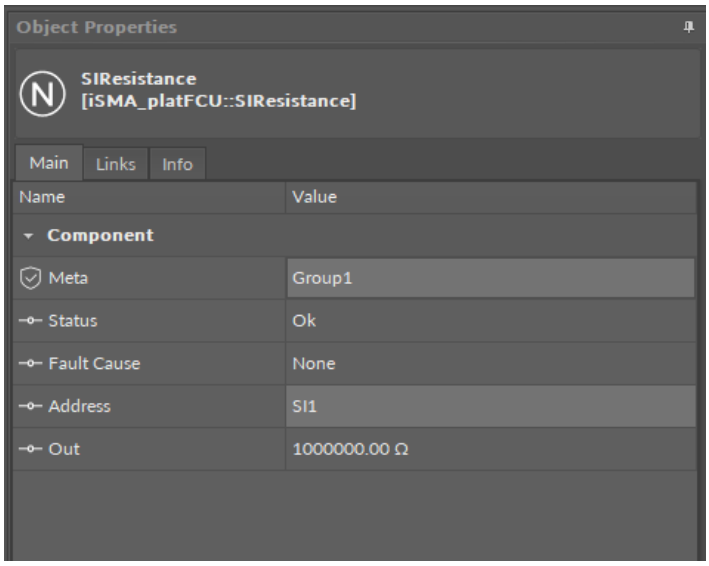


Figure 18. The SIResistance component

The SIResistance component has the following slots:

- **Status:** indicates the status of the SIResistance component;
- **Fault Cause:** shows the fault cause description;
- **Address:** sets the number of the physical input of the iSMA-B-FCU device (SI1, SI2, SI3, SI4);
- **Out:** displays the actual value of the input; if the voltage measurement is enabled, measuring of resistance is disabled, and the Out slot displays the last value of resistance.

5.1.7 TOutPWM

The TOutPWM is a component designed for servicing the triac output working in the PWM mode. The TOutPWM component has to be placed under the LocalIO component.

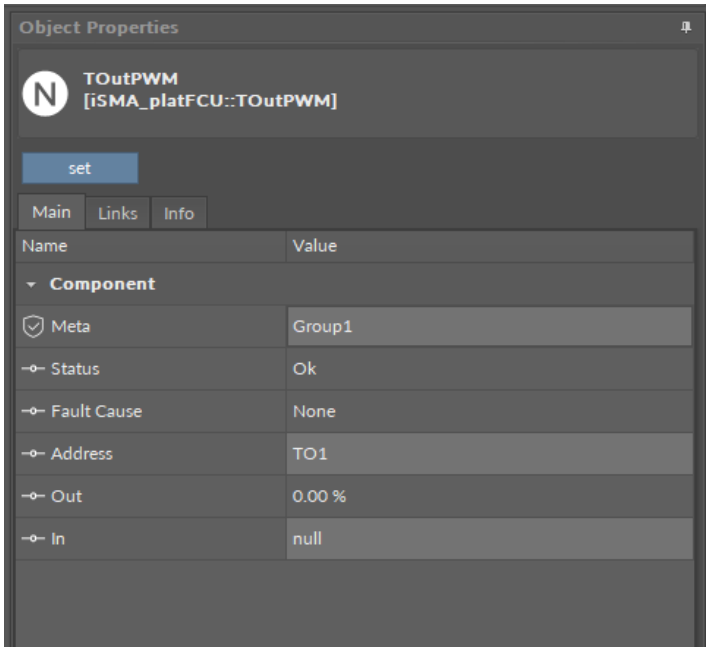


Figure 19. The TOutPWM component

The TOutPWM component has the following slots:

- **Status:** indicates the status of the TOutPWM component;

- **Fault Cause:** shows the fault cause description;
- **Address:** sets the number of the physical triac output of the iSMA-B-FCU device (TO1, TO2);
- **Out:** displays the actual value of the output;
- **In:** the input slot.

The TOutPWM component offers the following action, available in the context menu:

- **Set:** writes a value to the Out slot, and sends it to the device.

5.1.8 DI

The DI is a component designed for reading the digital input (true or false). The DI has to be placed under the LocalIO component.

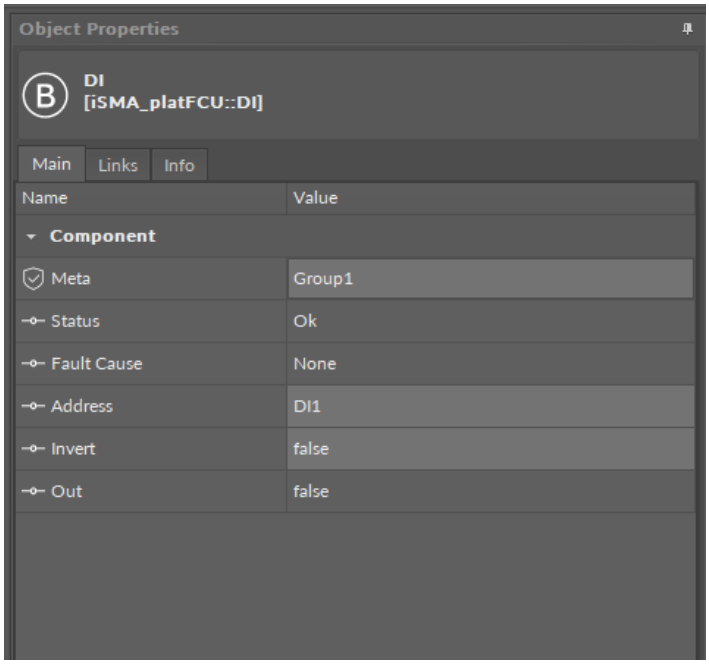


Figure 20. The DI component

The DI component has the following slots:

- **Status:** indicates the status of the DI component;
- **Fault Cause:** shows the fault cause description;
- **Address:** sets the number of the physical input of the iSMA-B-FCU device (DI1, DI2, DI3, DI4);
- **Invert:** shows the value negative to the value read from the digital output of the iSMA-B-FCU device;
- **Out:** displays the actual value of the input (true or false).

5.1.9 SITemperature

The SITemperature is a component designed for servicing special input in the temperature read mode (in order to get a reliable reading, the user must select the appropriate sensor type in the component LocalIOConfig). The SITemperature component has to be placed under the LocalIO component.

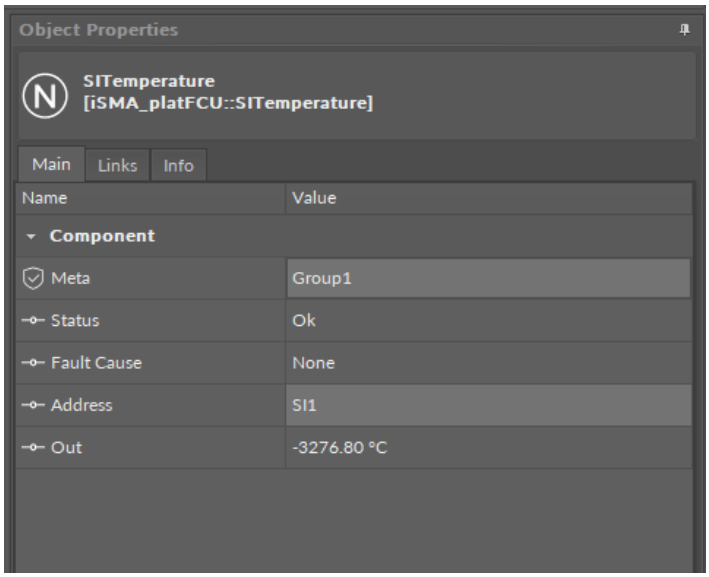


Figure 21. The SITemperature component

The SITemperature component has the following slots:

- **Status:** indicates the status of the SITemperature component;
- **Fault Cause:** shows the fault cause description;
- **Address:** sets the number of the physical input of the iSMA-B-FCU device (SI1, SI2, SI3, SI4);
- **Out:** displays the actual value of the input. If the voltage measurement is enabled, measuring of temperature is disabled, and the Out slot displays the last value of the temperature, measured before switching to the voltage measurement.

5.1.10 TODigital

The TODigital is a component designed for servicing the triac output operating working in digital mode. The TODigital component has to be placed under the LocalIO component.

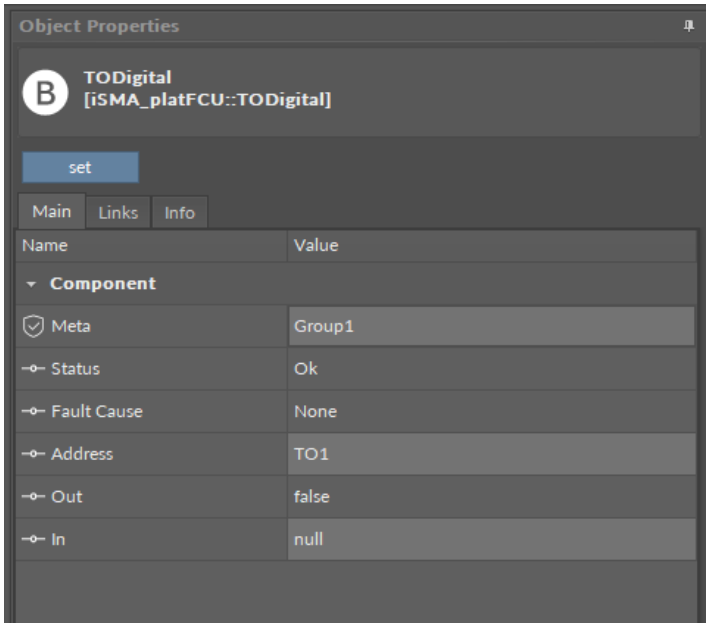


Figure 22. The TODigital component

The TODigital component has the following slots:

- **Status:** indicates the status of the TODigital component;

- **Fault Cause:** shows the fault cause description;
- **Address:** sets the number of the physical triac output of the iSMA-B-FCU device (TO1, TO2);
- **Out:** displays the actual value of the output;
- **In:** the input slot.

The TODigital component offers the following action, available in the context menu:

- **Set:** writes a value to the Out slot, and sends it to the device.

5.1.11 SIVoltage

The SIVoltage is a component designed for servicing the special input in the voltage read mode. For proper operation of the component, the type of servicing input has to be set to the Voltage_Measurement. The SIVoltage component has to be placed under the LocalIO component.

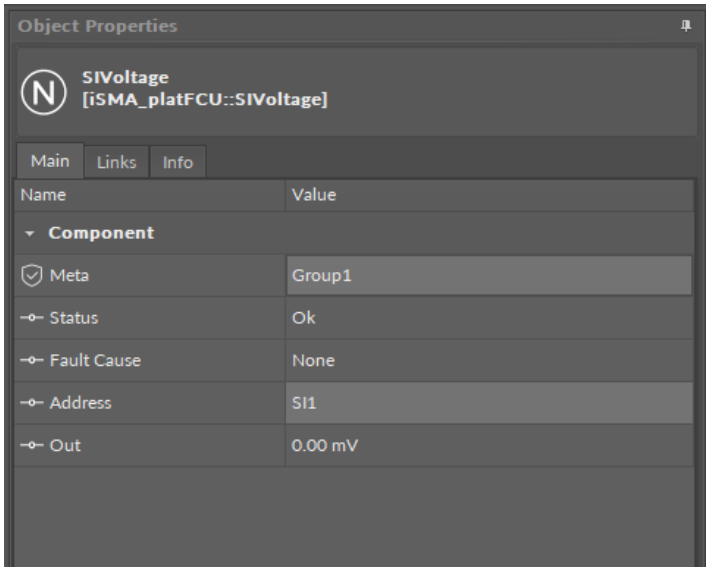


Figure 23. The SIVoltage component

The SIVoltage component has the following slots:

- **Status:** indicates the status of the SIVoltage component;
- **Fault Cause:** shows the fault cause description;
- **Address:** sets the number of the physical input of the iSMA-B-FCU device (SI1, SI2, SI3, SI4);
- **Out:** displays the actual value of the input. If the temperature measurement is enabled, measuring of voltage is disabled, and the Out slot displays the last value of voltage, measured before switching to the temperature measurement.

5.1.12 DIPSwitch

The DIPSwitch is a component designed to read states of eight binary signals set by the CFG DIP switch, mounted in the iSMA-B-FCU device. Using the CFG DIP switch is recommended to manage the configuration of the application. The DIPSwitch component has to be placed under the LocalIO component.

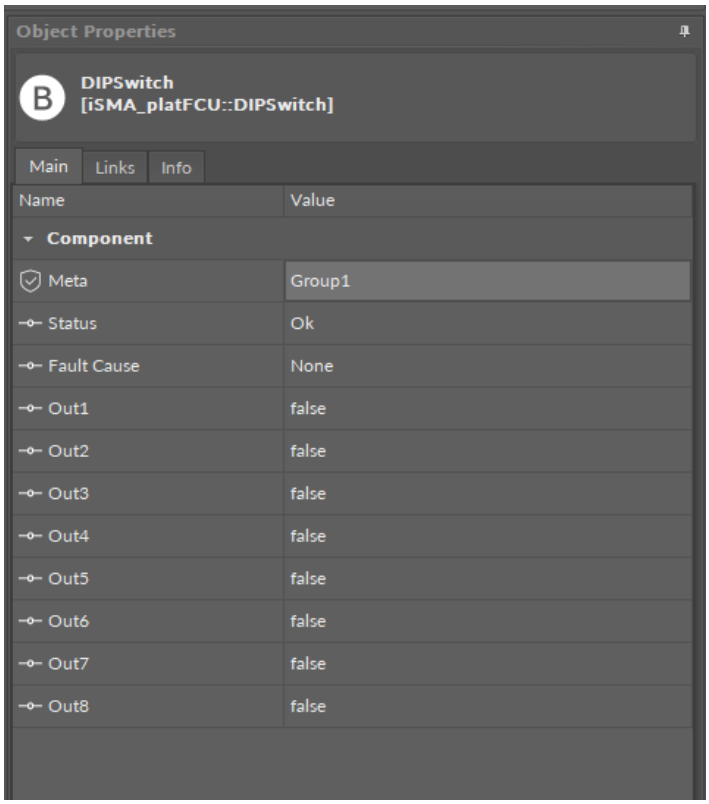


Figure 24. The DIPSwitch component

The DIPSwitch component has the following slots:

- **Status:** indicates the status of the DIPSwitch component;
- **Fault Cause:** shows the fault cause description;
- **Out1-Out8:** displays the actual states of each binary signal from the GFG DIP switch, according to the figure below:

Figure 25. The DIP switch configuration

5.1.13 In4Out

The In4Out is a component servicing the digital input 4 (I4) operating as the digital output. The In4Out component has to be placed under the LocalIO component.

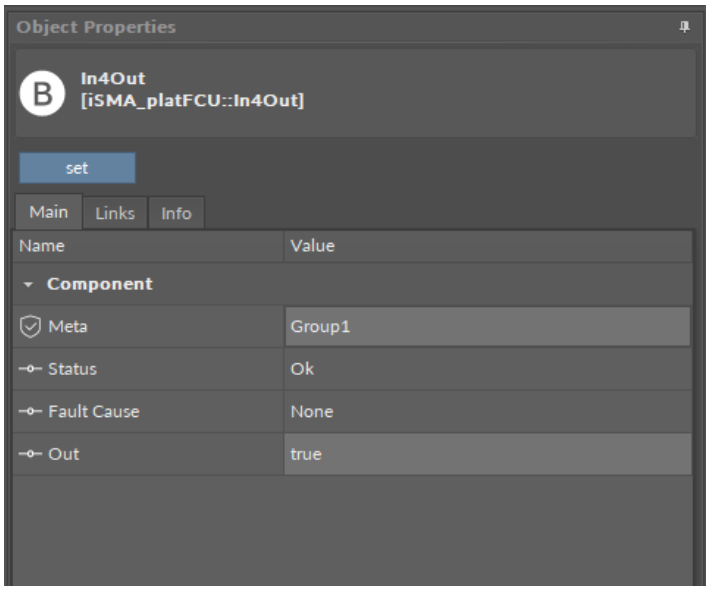


Figure 26. The In4Out component

The In4Out component has the following slots:

- **Status:** indicates the status of the In4Out component;
- **Fault Cause:** shows the fault cause description;
- **Out:** displays the actual state of the output, which is sent to the iSMA-B-FCU device. The true state overrides the low state (I4 terminal is connected to G0 ground–voltage between them is equal to 0 V DC). The false state does not affect the operation of the component–voltage between I4 and G0 equals 5 V DC.

The In4Out component offers the following action, available in the context menu:

- **Set:** writes a value to the Out slot, and sends it to the device.

5.1.14 LedAlarm

The LedAlarm is a component for servicing the alarm LED, mounted in the iSMA-B-FCU device. It allows to signalize operating states of the iSMA-B-FCU devices defined in the application. For example, it can be used for signalizing alarms.

The LedAlarm component has to be placed under the LocalIO component.

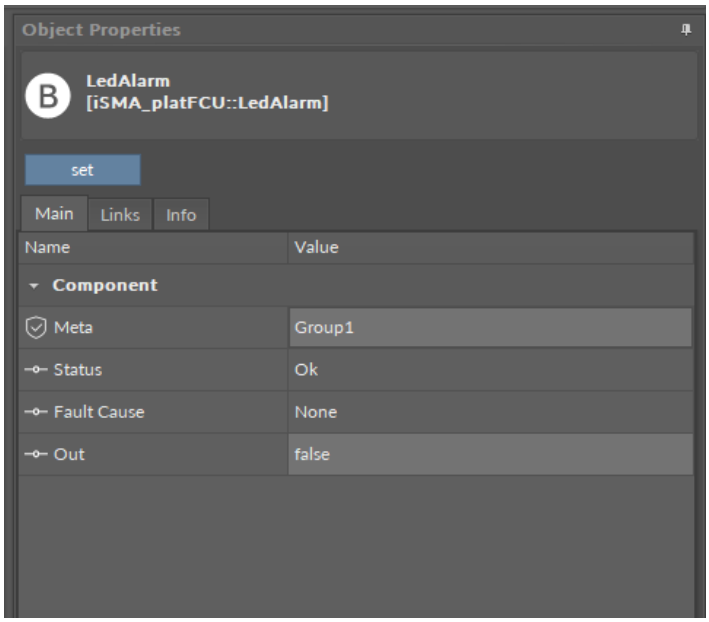


Figure 27. The LedAlarm component

The LedAlarm component has the following slots:

- **Status:** indicates the status of the LedAlarm component;
- **Fault Cause:** shows the fault cause description;
- **Out:** displays the actual state of the alarm LED, which is sent to the iSMA-B-FCU device. The true state–the LED lights up; the false state–the LED is off.

The LedAlarm component offers the following action, available in the context menu:

- **Set:** writes a value to the Out slot, and sends it to the device.

5.1.15 AOVoltage

The AOVoltage is a component designed for servicing the analog output working as a voltage output (range 0 mV-10 000 mV). The AOVoltage has to be placed under the LocalIO component.

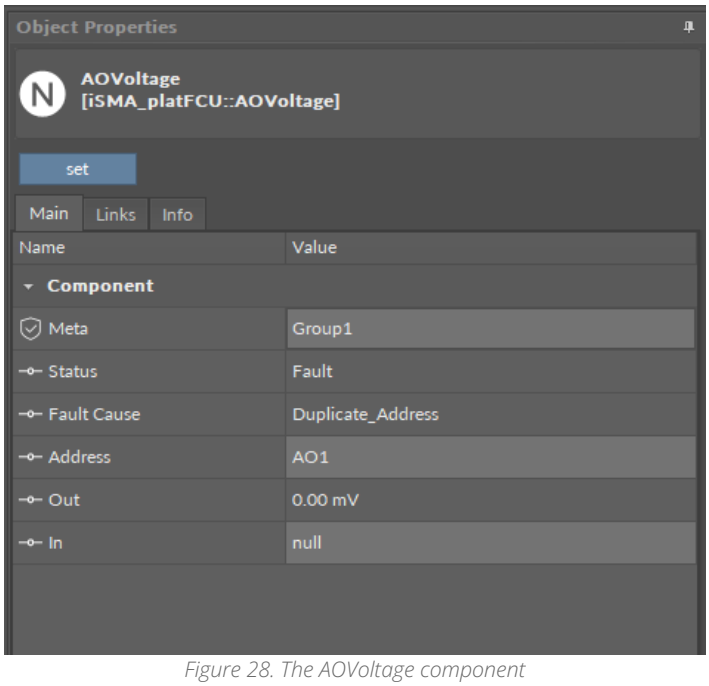


Figure 28. The AOVoltage component

The AOVoltage component has the following slots:

- **Status:** indicates the status of the AOVoltage component;
- **Fault Cause:** shows the fault cause description;
- **Address:** sets the number of the physical output of the iSMA-B-FCU device (AO1, AO2, AO3);
- **Out:** displays the actual value of the output (0 mV-10 000 mV);
- **In:** the input slot.

The AOVoltage component offers the following action, available under the right mouse button:

- **Set:** writes a value to the Out slot, and sends it to the device.

5.1.16 AODigital

The AODigital is a component designed for servicing analog output in the digital mode (false–0 V, true–10 V). The AODigital has to be placed under the LocalIO component.

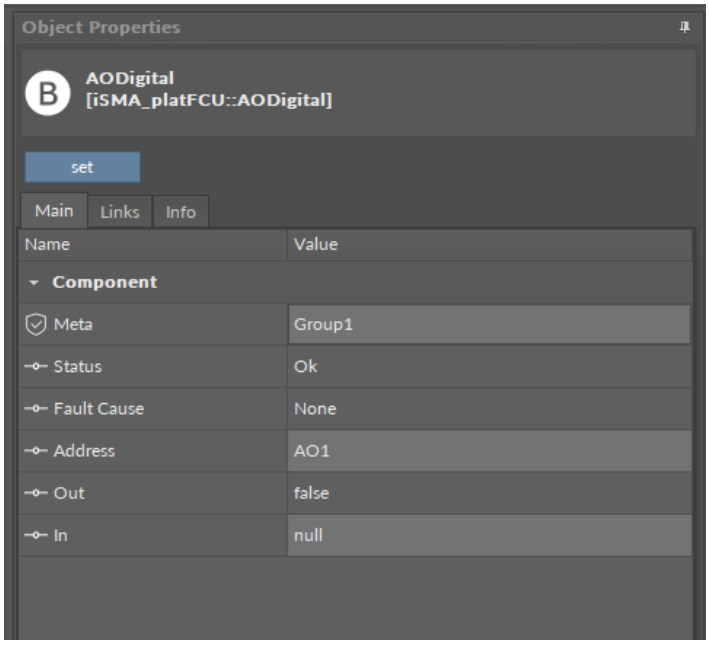


Figure 29. The AODigital component

The AODigital component has the following slots:

- **Status:** indicates the status of the AODigital component;
- **Fault Cause:** shows the fault cause description;
- **Address:** sets the number of the physical output of iSMA-B-FCU device (AO1, AO2, AO3);
- **Out:** displays the actual state of output (true/false);
- **In:** the input slot.

The AODigital component offers the following action, available in the context menu:

- **Set:** writes a value to the Out slot, and sends it to the device.

5.1.17 DICounter

The DICounter is a component dedicated for reading the high-speed counter of digital inputs; its values are saved to the EEPROM memory.

The DICounter has to be placed under the LocalIO component.

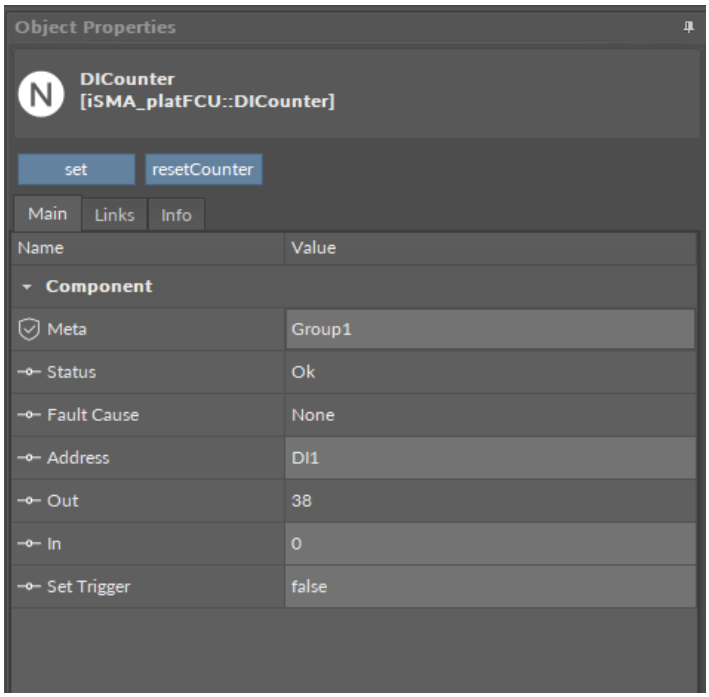


Figure 30. The DICounter component

The DICounter component has the following slots:

- **Status:** indicates the status of the DICounter component;
- **Fault Cause:** shows the fault cause description;
- **Address:** sets the number of the physical Input of the iSMA-B-FCU device (DI1, DI2, DI3, DI4);
- **Out:** displays the actual value of the counter servicing the selected input;
- **In:** the value set to Out slot, if the Set Trigger slot has changed from false to true;
- **Set Trigger:** if the state of the slot has changed from false to true, the value of the In slot is set to the Out slot.

The DICounter component offers the following actions, available in the context menu:

- **Set:** allows to set the value of the counter; the action overrides the Out and In slots;
- **Reset Counter:** allows to set the value to 0; the action overrides the Out and In slots.

5.1.18 LocalIOConfig

The LocalOConfig is a component designed for configuration of the physical inputs/outputs of the device. This component has to be placed under the LocalIO component.

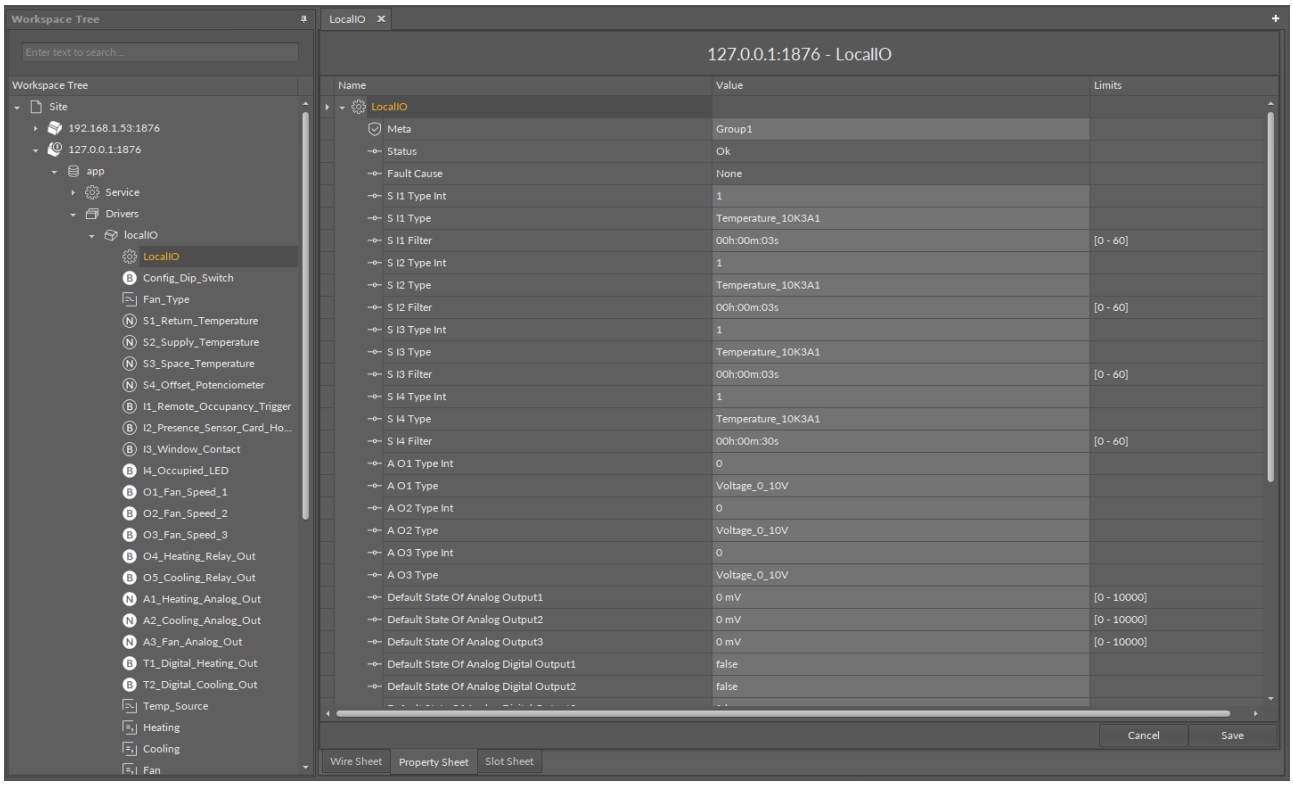


Figure 31. The LocalIOConfig component

The LocalOConfig component has the following slots:

- **SIn Type Int:** (where n is a number of the special input) sets the integer value corresponding to the type of temperature sensor;
- **SIn Type:** sets the type of temperature sensor connected to the special input; table stored in the device allows to convert the value of sensor resistance into temperature;
  - Available options: Voltage_Measurement, Temperature_10K3A1, Temperature_10K4A1, Temperature_10K, Temperature_20K6A1, Temperature_2_2K3A1, Temperature_3K3A1, Temperature_30K6A1, Temperature_SEI1, Temperature_TAC1, Temperature_SAT1.
- **SIn Filter:** sets the time constant of the low pass filter (to eliminate signal noise);
- **AOx Type Int:** sets the integer value corresponding to the analog output mode;
- **AOx Type:** sets the analog output mode: voltage 0-10 V or PWM;
- **Default State Of Analog Output x:** sets the analog output default value after the reboot of the controller (expressed in mV);
- **Default Digital State Of Analog Output x:** sets the default value of the analog output, operating in digital mode, after the reboot of the device;
- **TOn Type Int:** (where n is a number of the triac output) sets the integer value corresponding to the TOn Type.
- **TOn Type:** sets the triac output mode: digital or PWM;
- **Default State Of Triac Output x:** sets the default value of the triac output, operating in PWM mode, after the reboot of the device (expressed in %);
- **Default State Of Triac Digital Output x:** sets the default value of the triac output, operating in digital mode, after the reboot of the device;
- **Mode:** sets the mode the device works in, normal or fan–if set to the fan mode, the digital outputs O1, O2, and O3 cannot be set to high state at the same time; if set to the normal mode, this protection is disabled;
- **Default State Of Digital Output x:** the digital output default after the reboot of the device;

Any settings of the LocalIOConfig component are stored in the component, and can be transferred to other devices (quick set-up of multiple devices).

5.2 NV Components

The NV components (non-volatile) are components, which value can be recorded in the device's EEPROM non-volatile memory. Whenever the device is restarted or the power is down, the values of NV components remain saved.

The device has three types of NV components, which support different types of variables:

- Boolean variables: the NVBooleanWritable component;
- Integer variables: the NVIntegerWritable component;
- Numeric (float) variables: the NVNumericWritable component.

The NV components can operate in the Auto mode (the "In" slot values are transferred to the "Out" slot), or in the Hand mode (the "Out" value is entered manually by the user and cannot be changed by the application).

Since the values of the components are not stored in the Sedona application, but in the non-volatile memory of the device, if the application is copied between two devices, the output values are not saved, and it will derive the values stored in the local EEPROM memory. To copy the NV components to another device along with their values (e.g., setpoints), use global actions of the plat component:

Step 1: Use the global action Copy From NV The Default / Copy From NV To User.

Step 2: Save the application, and copy it to another device.

Step 3: Use the global action on the target device Copy From Default To NV / Copy From User To NV.

5.2.1 NVNumericWritable

The NVNumericWritable is a component that stores the output value in the non-volatile memory of the device. After the power failure or rebooting the device, the component value is restored from this particular memory. The space meter of the occupied EEPROM memory is located in the plat component.

Note: The way of calculating memory cells for NV components is described in the Plat service section.

The NVNumericWritable component is also used to integrate numeric (float) variables from various sources. It is done using "reverse following the link" function. The Out slot is connected to the In slots of various protocols, for example, BACnet or Modbus variable. Upon changing a value in one of the components, the device will perform the set action on the NV component to synchronize the values in all the connected components.

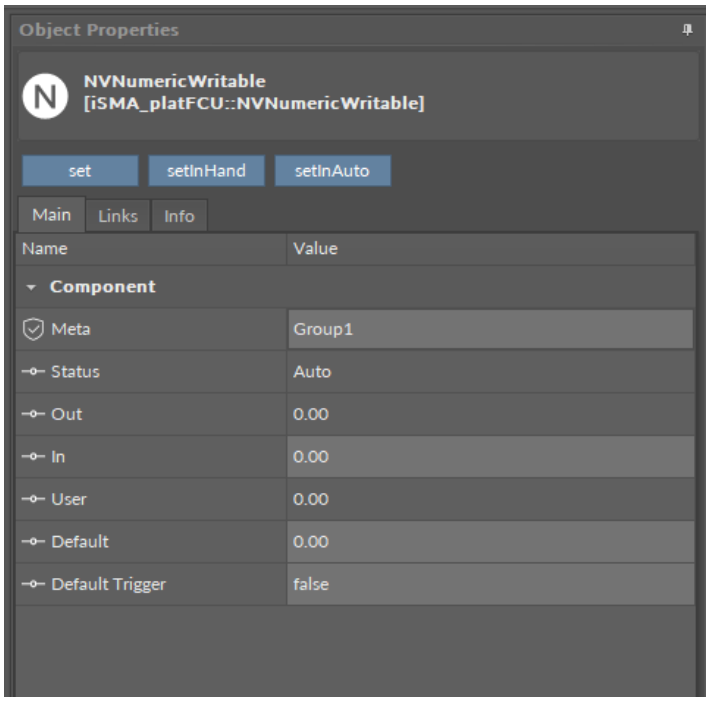


Figure 32. The NVNumericWritable component

The NVNumericWritable component has the following slots:

- **Status:** the point's actual status (Auto/Hand);
- **Out:** the output slot;
- **In:** the input slot;
- **User:** the user value slot (setting by Set action);
- **Default:** the default value slot (set by the global command from plat action);
- **Default Trigger:** copying trigger from Default to Out.

The NVNumericWritable component offers the following actions, available under the right mouse button:

- **Set:** sets the User slot and In the slot if there is no link to the In slot;
- **Set In Hand:** sets the value to the Out slot, and blocks changing it from any other slots;
- **Set In Auto:** switches off the Hand mode, and sets the Out slot according to the In slot's value.

5.2.2 NVIntegerWritable

The NVIntegerWritable is a component that stores the output value in the non-volatile memory of the device. After the power failure or rebooting the device, the component value is restored from this particular memory. The space meter of the occupied EEPROM memory is located in the plat component.

Note: The way of calculating memory cells for NV components is described in the Plat service section.

The NVIntegerWritable component is also used to integrate integer variables from various sources. It is done using the "reverse following the link" function. The Out slot is connected to the In slots of various protocols, for example, BACnet or Modbus variable. Upon changing a value in one of the components, the device will perform the set action on the NV component to synchronize the values in all the connected components.

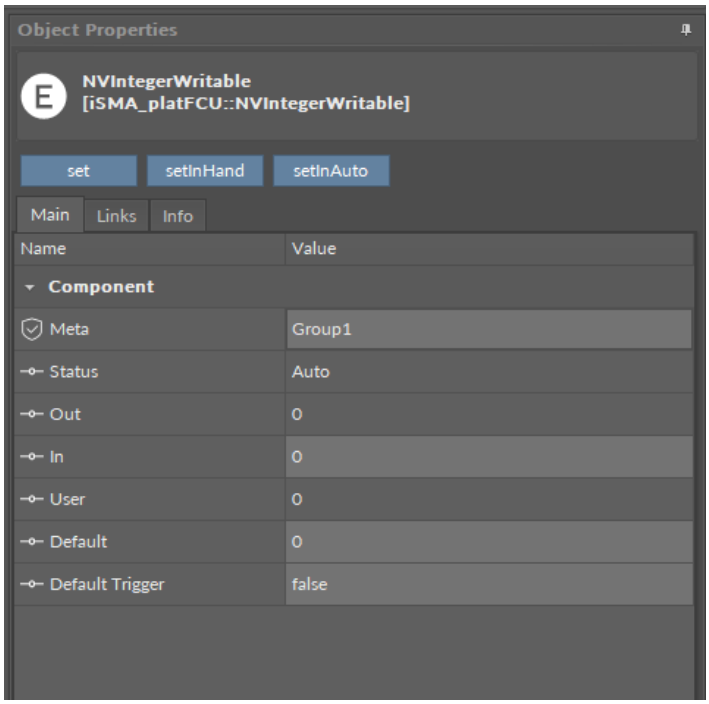


Figure 33. The NVIntegerWritable component

The NVIntegerWritable component has the following slots:

- **Status:** the point's actual status (Auto/Hand);
- **Out:** the output slot;
- **In:** the input slot;
- **User:** the user value slot (setting by Set action);
- **Default:** the default value slot (set by the global command from plat action);
- **Default Trigger:** copying trigger from Default to Out.

The NVIntegerWritable component offers the following actions, available under the right mouse button:

- **Set:** sets the User slot and In the slot if there is no link to the In slot;
- **Set In Hand:** sets the value to the Out slot, and blocks changing it from any other slots;
- **Set In Auto:** switches off the Hand mode, and sets the Out slot according to the In slot's value.

5.2.3 NVBooleanWritable

The NVBooleanWritable is a component that stores the output value in the non-volatile memory of the device. After the power failure or rebooting the device, the component's value is restored from this particular memory. The space meter of the occupied EEPROM memory is located in the plat component.

Note: The way of calculating memory cells for NV components is described in the Plat service section.

The NVBooleanWritable component is also used to integrate Boolean variables from various sources. It is done using the "reverse following the link" function. The Out slot is connected to the In slots of various protocols, for example, BACnet or Modbus variable. Upon changing a value in one of the components, the device will perform the set action on the NV component to synchronize the values in all the connected components.

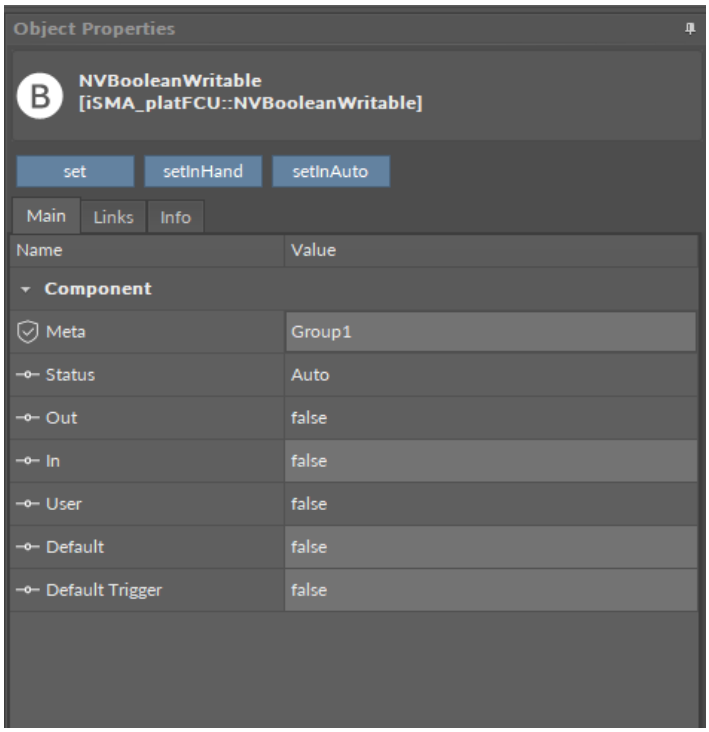


Figure 34. The NVBooleanWritable component

The NVBooleanWritable component has the following slots:

- **Status:** the point's actual status (Auto/Hand);
- **Out:** the output slot;
- **In:** the input slot;
- **User:** the user value slot (setting by Set action);
- **Default:** the default value slot (set by the global command from plat action);
- **Default Trigger:** copying trigger from Default to Out.

The NVBooleanWritable component offers the following actions, available in the context menu:

- **Set:** sets the User slot and In the slot if there is no link to the In slot;
- **Set In Hand:** sets the value to the Out slot, and blocks changing it from any other slots;
- **Set In Auto:** switches off the Hand mode, and sets the Out slot according to the In slot's value.

5.3 SlaveNetwork

The slave network is used for communication of the iSMA-B-FCU device with the upper-level systems (for example, BMS). Communication can be realized by the Modbus RTU/ASCII or the BACnet MS/TP protocol (depends on the PROTOCOL DIP switch configuration).

The SlaveNetwork component is used to manage the BACnet MS/TP or Modbus RTU/ASCII protocols, using the RS485 port. The SlaveNetwork component has to be placed under the Drivers component.

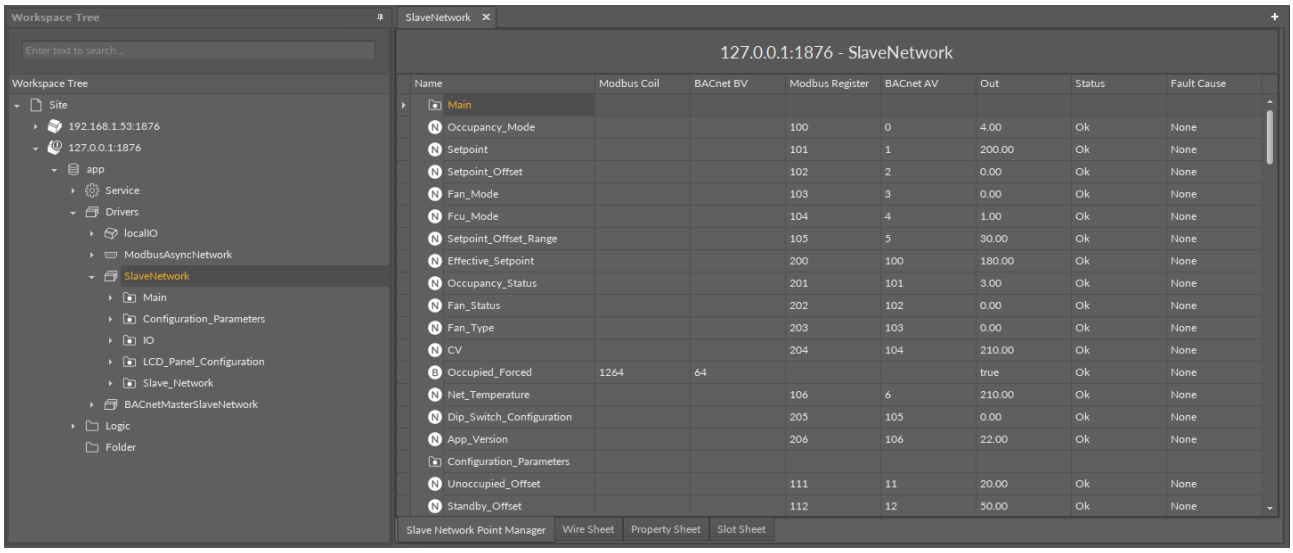


Figure 35. The SlaveNetwork component

The SlaveNetwork component has the following slots:

- **Status:** indicates the network status;
- **Fault Cause:** shows the fault cause description;
- **Baudrate:** indicates the baud rate of communication (set by the PROTOCOL DIP Switch);
- **Data Bits:** displays the number of data bits; the default value is 8;
- **Parity Bits:** configures the parity bits: Parity_Disabled, Odd, Even;
- **Stop Bits:** displays the number of stop bits; the default value is 1;
- **Protocol:** displays the protocol used for communication: Modbus or BACnet (set by the PROTOCOL DIP switch);
- **Slave Address:** displays the actual address of the device (set by the MAC DIP switch);
- **Device Name:** sets the BACnet device name;
- **BACnet Id:** displys the actual BACnet Id; the default value equals to 826000 + Slave Address. The BACnet Id can be set using the Set BACnet ID action.

The SlaveNetwork component has the following action, available in the context menu:

- **Set BACnet ID:** sets the BACnet ID of the device.

5.3.1 NVNetBoolean

The NVNetBoolean is a component that stores the output value in the non-volatile EEPROM memory of the device. After rebooting the device or power failure, the component's value is restored from this particular memory. The NVNetBoolean component occupies a single memory cell of the Boolean type.

The component has to be placed under the SlaveNetwork component.

Note: The method of calculating memory cells for NV components is described in the Plat service section.

The NVNetBoolean component is also used to integrate Boolean variables from various sources. It is done using the "reverse following the link" function. The Out slot is connected to the In slots of various protocols, for example, BACnet or Modbus variables. After changing a value in one of the components, the device performs the Set action on

the NVNet component to synchronize the values in all connected components. This option is enabled only if the Link Set slot is set to true.

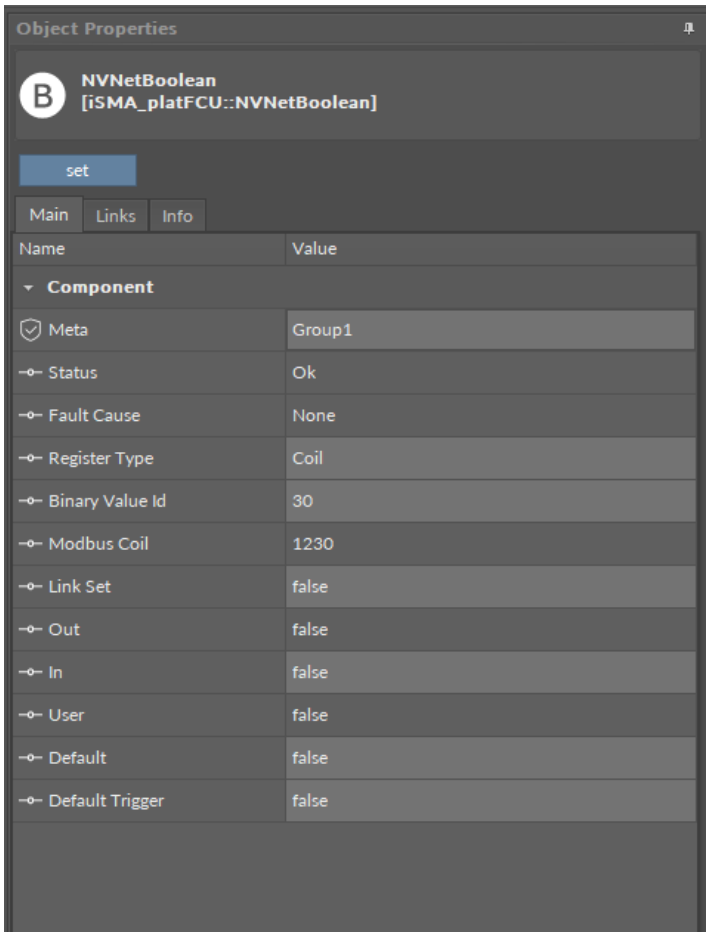


Figure 36. The NVNetBoolean component

The NVNetBoolean component has the following slots:

- **Status:** the actual status of the component;
- **Fault Cause:** shows the fault case description;
- **Register Type:** allows to set the register type for the master device (Coil: read/write, Discrete Input: read-only);
- **Binary Value Id:** allows to set the Id of the BACnet object;
- **Modbus Coil:** displays the number of the modbus coil;
  - Modbus Coil = 1200 + Binary Value Id,
- **Link Set:** if the slot is set to true, the "reverse following the link" function is activated, and it invokes the Set action in the component linked to the In slot;
- **Out:** the output slot;
- **In:** the input slot;
- **User:** the user value slot (set by the Set action);
- **Default:** the default value slot (set by the global command from the plat action);
- **Default Trigger:** copies values from the Default slot to the Out slot on the rising edge.

The NVNetBoolean component offers the following action, available in the context menu:

- **Set:** this action sets the User slot and the In slot, if there is no link on the In slot.

5.3.2 NVNetNumeric

The NVNetNumeric is a component that stores the output value in the non-volatile EEPROM memory of the device. After rebooting the device or power failure, the component value is restored from this particular memory. The NVNetNumeric component occupies a single memory cell of the numeric type.

The NVNetNumeric component has to be placed under the SlaveNetwork component.

Note: The method of calculating memory cells for NV components is described in the Plat service section.

The NVNetNumeric component is used also to integrate numeric (float) variables from various sources. It is done using the "reverse following the link" function. The Out slot is connected to the In slots of various protocols, for example, BACnet or Modbus variable. After changing a value in one of the components, the device performs the Set action on the NVNet component to synchronize the values in all the connected components. This option is enabled only if the Link Set slot is set to true.

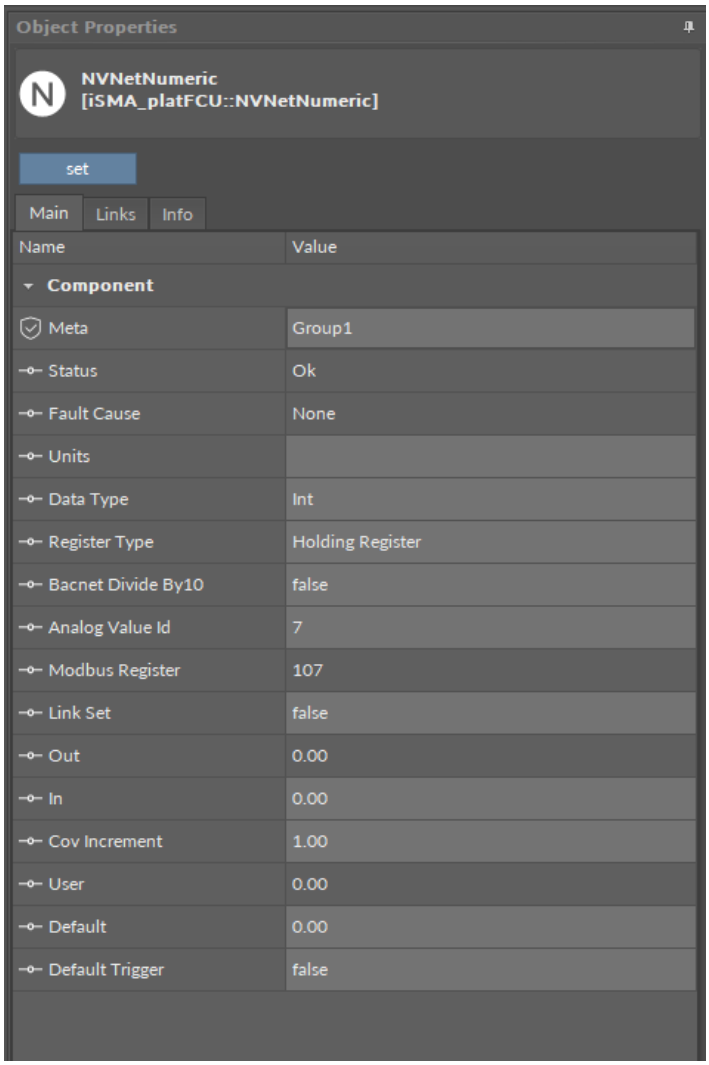


Figure 37. The NVNetNumeric component

The NVNetNumeric component has the following slots:

- Status: the actual status of the component;
- Fault Cause: shows the fault case description;
- Units: allows to set the BACnet unit of the point;

- **Data Type:** allows to set the type of data (Int, Long, Float, SInt, SLong);

WARNING!

Data types: Long, Slong, and Float, consume 2 registers, therefore each next network variable address must leave one address gap.

- **Register Type:** allows to set the register type for the master device (Holding Register: read/write, Input Register: read-only);
- **BACnet Divide By10:** if the slot is set to true, and the device communicates with the BACnet protocol, the value of the read/write point is equal to the value received from the Out slot divided by 10;
- **Analog Value Id:** allows to set the Id of the BACnet object;
- **Modbus Register:** displays the number of the Modbus register;

Note: Modbus Register = 100 + Analog Value Id.

- **Link Set:** if the slot is set to true, the "reverse following the link" function is activated, and it invokess the Set action in the component linked to the In slot;
- **Out:** the output slot;
- **In:** the input slot;
- **Cov Increment:** allows to set the minimum change of value;
- **User:** the user value slot (set by the Set action);
- **Default:** the default value slot (set by the global command from the plat action);
- **Default Trigger:** copies values from the Default slot to the Out slot on the rising edge.

The NVNetNumeric component offers the following action, available in the context menu:

- **Set:** the action sets the User slot and the In slot (if there is no link to the In slot).

5.3.3 NVNet

The NVNet components (non-volatile net) are the type of components, the value of which can be recorded in the device's EEPROM non-volatile memory. Whenever a device is restarted or the power is down, the values of NVNet components remain saved. These components can also be used to sending values using the BACnet MS/TP or Modbus RTU protocols (depending on the PROTOCOL DIP switch configuration). The device has two types of NV components, broken down by the type of variables they support.

The components include:

- Boolean variables: the NVNetBoolean component;
- Numeric (float) variables: the NVNetNumeric component.

All NVNet components have to be placed under the SlaveNetwork component.

Note: The iSMA-B-FCU device supports up to 200 NVNetNumeric or up to 200 NVNetBoolean components. The number of free NV memory cells can be checked under platform properties.

Note: The method of calculating memory cells for NV components is described in the plat component section. Since the values of the components are not stored in the Sedona application but in the non-volatile memory of the device, in the situation when an application is copied between two devices, output values are not saved and will assume

the values stored in the local EEPROM memory. To copy NV Net components to another device along with their values (e.g., setpoint), use global actions of the plat component:

Step 1: Use the global action Copy From NV The Default / Copy From NV To User.

Step 2: Save the application, and copy it to another device.

Step 3: Use the global action on the target device Copy From Default To NV / Copy From User To NV.

6 iSMA Room Devices Modbus Kit

The iSMA Room Devices Modbus kit is an extension of the Modbus Async Network kit, which allows to easily manage the iSMA-B-LP, Touch Point, and FP devices. With the kit's components, the user can build an application that easily communicates and configures the LP/Touch Point/FP panels.

Note: Components in the kit that contain the Lp- prefix in their names work both for the LP, Touch Point, and FP panels, **except for** components for menus: LpMainMenuBoolean, LpMainMenuNumeric, LpSubmenuBoolean, LpSubmenuNumeric. Also, components for CO2 and humidity sensors work only with the LP and Touch Point panels.

6.1 FanSpeed

Note: Component applicable for the LP, Touch Point, and FP panels.

The FanSpeed component is responsible for configuring fan settings in the panel.

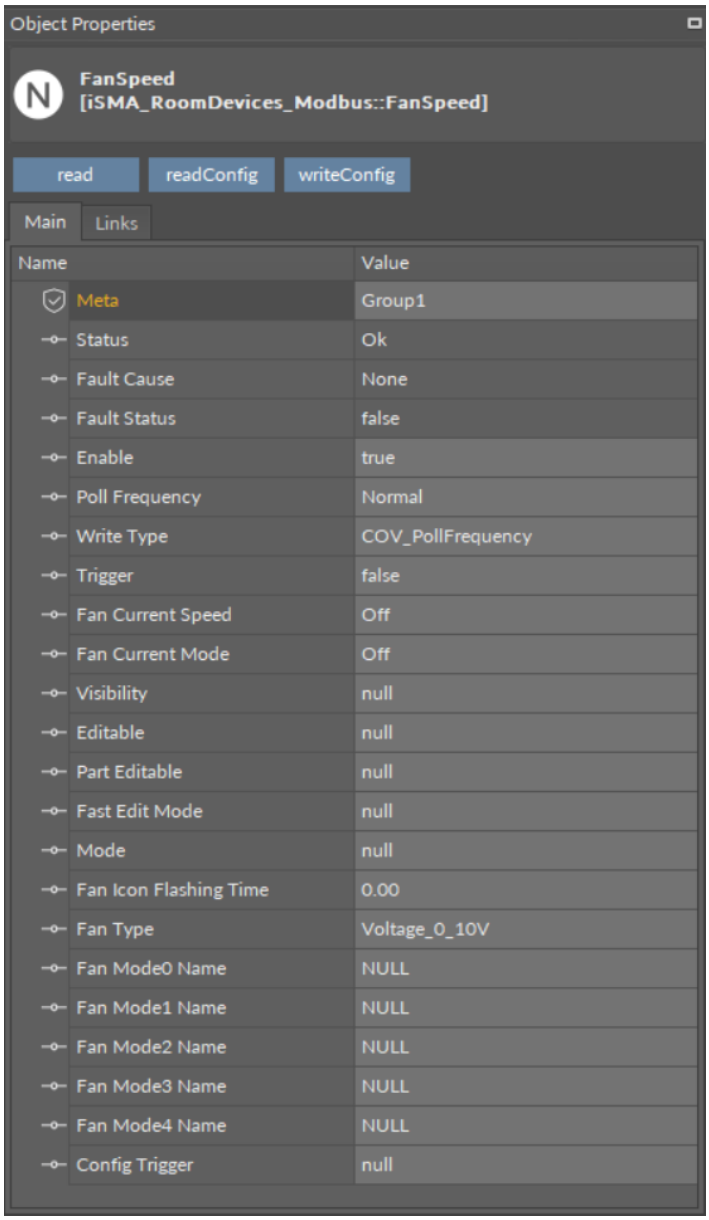


Figure 38. FanSpeed component

The FanSpeed component has the following slots:

- **Status:** shows the component's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** informs about the point error status (true: point read/write error);
- **Enable:** enables or disables the component (true: enabled, false: disabled);
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Write Type:** defines a method of writing values (COV, COV_PollFrequency, PollFrequency, COV_LinkSet);
- **Trigger:** allows to force sending values on rising edge;
- **Fan Current Speed:** sets the current speed of fan (Off, ManualSpeed1, ManualSpeed2, ManualSpeed3, AutoSpeed1, AutoSpeed2, AutoSpeed3)
- **Fan Current Mode:** sets the current mode of fan (Off, ManualSpeed1, ManualSpeed2, ManualSpeed3, Auto).
- **Visibility:** allows to activate or deactivate the point on the display (only for the LP panel);
- **Editable:** enables or disables editing of a fan speed in the panel;

- **Part Editable:** allows to set an editing mode in the panel (FullyEditable, AutoOffMode);
- **Fast Edit Mode:** enables a fast edit mode in the panel (only for the LP panel);
- **Mode:** identifies a way of controlling the panel (LocalMode, BmsMode);
- **Fan Icon Flashing Time:** allows to set a flashing time of icons on the display (only for the LP panel);
- **Fan Type:** sets a type of fan (0-10 V, 1-speed, 2-speed, 3-speed);
- **Fan Mode0 Name-Fan Mode4 Name:** allows to set different fan mode names (up to 4, only ASCII characters; only for the LP panel);
- **Config Trigger:** on rising edge sends configuration parameters to the device components (Editable, Part Editable, Mode, Fan Type).

The FanSpeed component has the following actions:

- **Read:** reads the panel's fan values and updates the Fan Current Speed and Fan Current Mode slots;
- **Read Config:** reads configuration parameters from the panel (Editable, Part Editable, Mode, Fan Type);
- **Write Config:** writes configuration parameters to the panel (Editable, Part Editable, Mode, Fan Type).

6.2 CO2Alarm

Note: Component applicable for the LP and Touch Point panels.

The CO2Alarm component is dedicated to the configuration of the high limit alarm function in the panel.

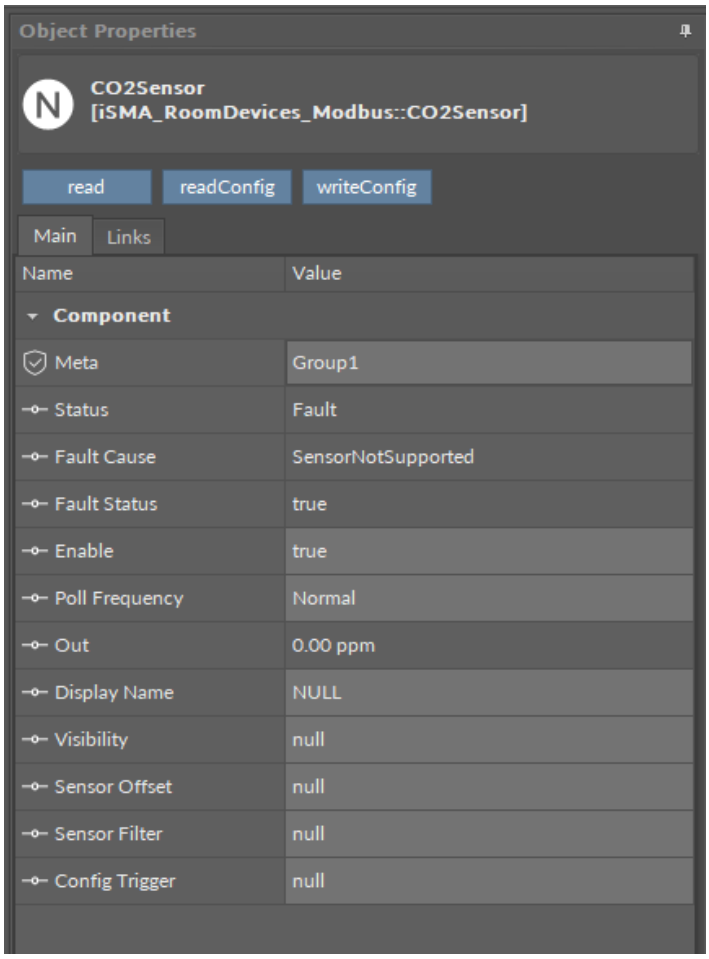


Figure 39. CO2Alarm component

The CO2Alarm component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** informs about the point error status (true: point read/write error);
- **Enable:** enables or disables the point (true: enabled, false: disabled);
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Out:** the CO_2 alarm status;
- **Alarm Acknowledged:** informs if the alarm has been confirmed by a system operator;
- **Co2 Setpoint For Alarm:** sets a CO_2 alarm setpoint value in ppm;
- **Co2 Differential For Alarm:** sets a CO_2 alarm setpoint differential value in ppm;
- **Co2 Alarm Flashing Lcd:** sets the active or inactive a background illumination flashing;
- **Co2 Alarm Buzzer:** activates or inactivates a sound alarm;
- **Co2 Alarm In Alarm Show High:** activates or inactivates the "High" label display (only for the LP panel);
- **Co2 Alarm Confirm Enable:** activates or inactivates an alarm acknowledgement by any button;
- **Config Trigger:** sends configuration parameters to the panel on rising edge.

The CO2Alarm component has the following right-click menu actions:

- **Read:** reads the panel's CO_2 alarm status and updates the Out slot;
- **Read Config:** reads configuration parameters from the panel, (Co2 Setpoint For Alarm, Co2 Differential For Alarm, Co2 Alarm Flashing Lcd, Co2 Alarm Buzzer, Co2 Alarm In Alarm Show High, Co2 Alarm Confirm Enable);

- **Write Config:** writes configuration parameters to the panel (Co2 Setpoint For Alarm, Co2 Differential For Alarm, Co2 Alarm Flashing Lcd, Co2 Alarm Buzzer, Co2 Alarm In Alarm Show High, Co2 Alarm Confirm Enable).

6.3 CO2Sensor

Note: Component applicable for the LP and Touch Point panels.

The CO2Sensor component is responsible for reading values and configuration of the CO_2 sensor.

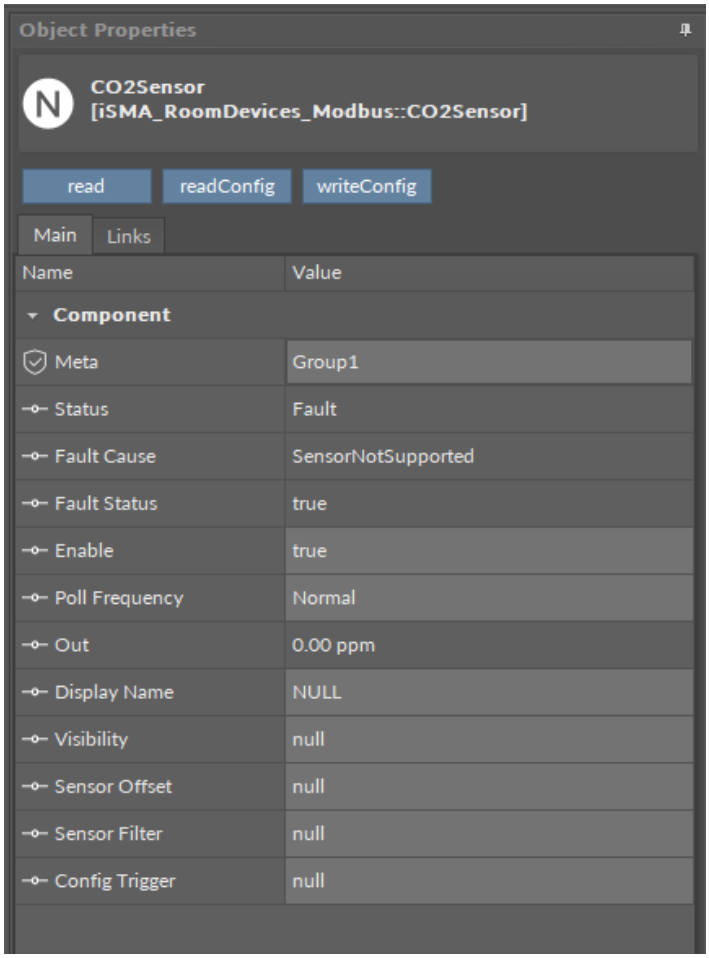


Figure 40. CO2Sensor component

The CO2Sensor component has the following slots:

- **Status:** shows the component's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** informs about the point error status (true: point read/write error);
- **Enable:** enables or disables the component (true: enabled, false: disabled);
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Out:** the CO_2 sensor value;
- **Display Name:** allows to set the CO_2 sensor name on the display (up to 4 characters, only ASCII characters, only for the LP panel);
- **Visibility:** activates or inactivates the sensor value on the display;
- **Sensor Offset:** sets the CO_2 sensor offset value;
- **Sensor Filter:** sets the CO_2 sensor reading filter time in seconds;

- **Config Trigger:** sends configuration parameters to the device components on rising edge (Display Name, Visibility, Sensor Offset, Sensor Filter).

The LpCO2Sensor component has the following right-click menu actions:

- **Read:** reads the remote device CO_2 sensor value and updates the Out slot;
- **Read Config:** reads configuration parameters from the panel (Display Name, Visibility, Sensor Offset, Sensor Filter);
- **Write Config:** writes configuration parameters to the panel (Display Name, Visibility, Sensor Offset, Sensor Filter).

6.4 HumiditySensor

Note: Component applicable for the LP and Touch Point panels.

The HumiditySensor component is responsible for configuration of the humidity sensor and reading its value.

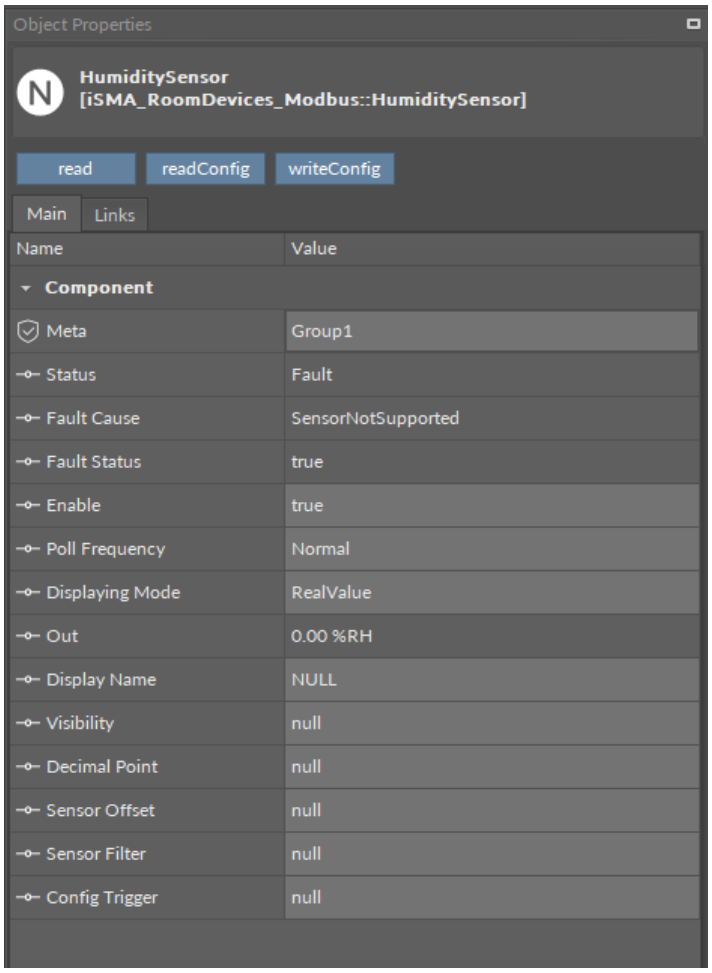


Figure 41. HumiditySensor component

The HumiditySensor component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** shows the point error status (true: point read/write error);
- **Enable:** enables or disables the point (true: enabled, false: disabled);
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);

- **Displaying Mode:** allows to set the display mode (RealValue: the Out value is divided by 10, RegisterValue: the value is taken directly from the register);
- **Out:** the humidity sensor value;
- **Display Name:** allows to set the humidity sensor display name on the display (up to 4 characters, only ASCII characters, only for the LP panel);
- **Visibility:** activates or inactivates the humidity sensor value on the display;
- **Decimal Point:** allows to activate or deactivate the decimal place on the display (inactive or displays decimal point on first, second or third position);
- **Sensor Offset:** sets the humidity sensor offset value;
- **Sensor Filter:** sets the humidity sensor reading filter time in seconds;
- **Config Trigger:** sends configuration parameters to the device components on rising edge.

The LpHumiditySensor component has the following right-click menu actions:

- **Read:** reads the remote device humidity sensor value and updates the Out slot;
- **Read Config:** reads configuration parameters from the panel (Display Name, Visibility, Decimal Point, Sensor Offset, Sensor Filter);
- **Write Config:** writes configuration parameters to the panel (Display Name, Visibility, Decimal Point, Sensor Offset, Sensor Filter).

6.5 MainMenuBoolean

Warning!

Component applicable only for the LP panel.

The MainMenuBoolean component is responsible for reading/writing and configuration of a single Boolean parameter, which is placed in the LP panel main menu.

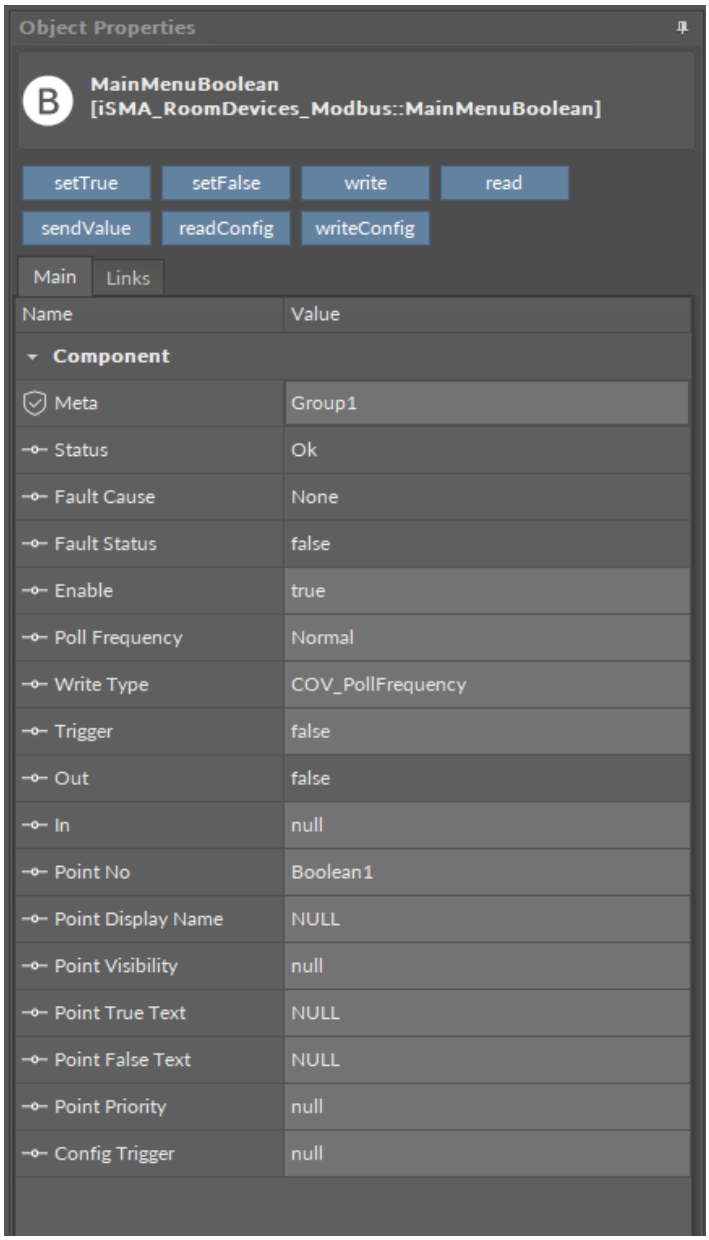


Figure 42. MainMenuBoolean component

The MainMenuBoolean component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** shows the point error status (true: point read/write error);
- **Enable:** enables or disables the point (true: enabled, false: disabled);
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Write Type:** allows to set the writing mode (COV: only on the In slot change, COV_PollFrequency: on the In slot change and periodically, PollFrequency: only periodically, COV_LinkSet: only on the In slot change, using the "reverse following the link" function);
- **Trigger:** allows to force sending values on rising edge;
- **Out:** the main menu point output slot, the current value;
- **In:** the main menu point input slot;
- **Point No:** the panel's main menu point number;
- **Point Display Name:** allows to set the point display name on the LCD screen (up to 4, only ASCII characters);

- **Point Visibility:** allows to activate or deactivate the point on the display;
- **Point True Text:** allows to set the 4 characters LCD display text in the true state (only ASCII characters);
- **Point False Text:** allows to set the 4 characters LCD display text in the false state (only ASCII characters);
- **Point Priority:** allows to set the displaying priority on the LCD screen (starting from the lowest value);
- **Config Trigger:** sends configuration parameters to the device components on rising edge.

The MainMenuBoolean component has the following right-click menu actions:

- **Set True:** sets the true state in the In slot and sends it to the main menu point;
- **Set False:** sets the false state in the In slot and sends it to the main menu point;
- **Write:** sends the In slot state to the main menu point;
- **Read:** reads the panel main menu point value and sets the Out slot;
- **Send Value:** sends the point user value to the main menu without changing the In slot, from the pop-up window;
- **Read Config:** reads the main menu point configuration parameters from the panel (Point Display Name, Point Visibility, Point True Text, Point False Text, Point Priority);
- **Write Config:** writes the main menu point configuration parameters to the panel (Point Display Name, Point Visibility, Point True Text, Point False Text, Point Priority).

6.6 MainMenuNumeric

Warning!

Component applicable only for the LP panel.

The MainMenuNumeric component is responsible for reading/writing and configuration of a single numeric parameter, which is placed in the LP panel main menu.

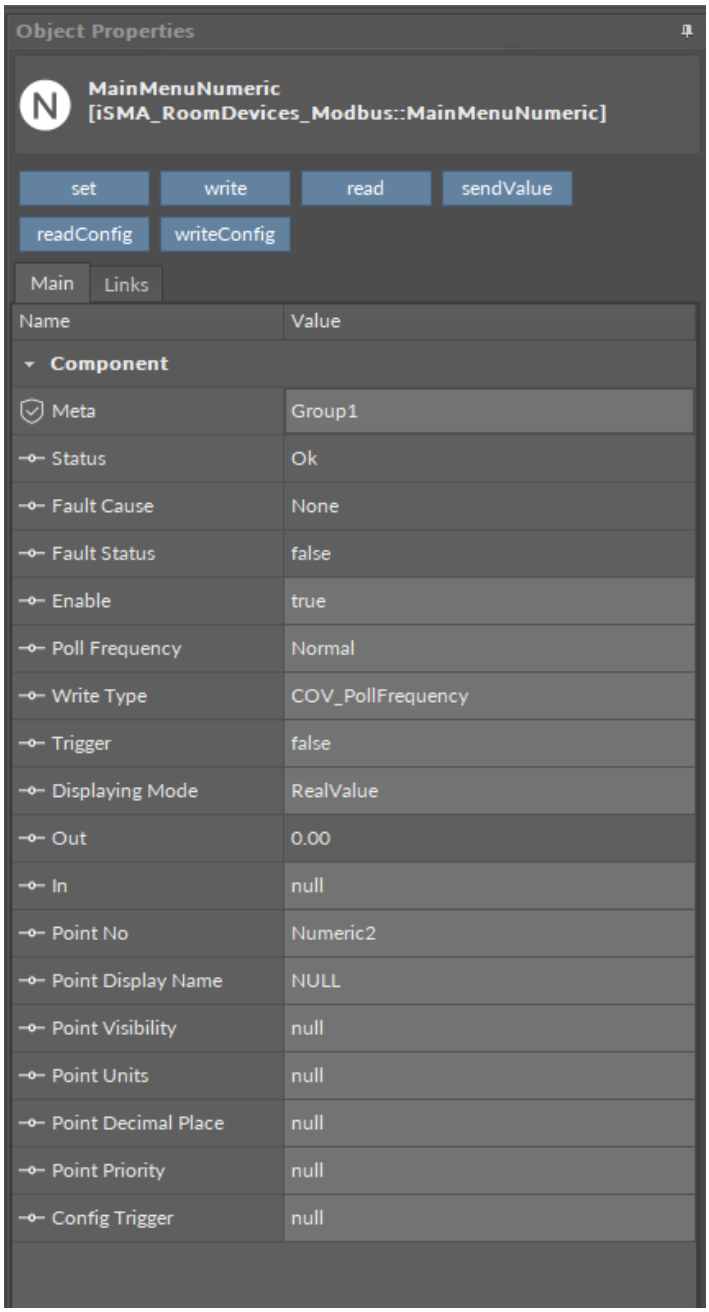


Figure 43. MainMenuNumeric component

The MainMenuNumeric component has the following slots:

- **Status:** shows the point status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** informs about the point error status (true: point read/write error);
- **Enable:** enables or disables the point (true: enabled, false: disabled);
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Write Type:** allows to set the writing mode (COV: only on input change, COV_PollFrequency: on the In slot change and periodically, PollFrequency: only periodically, COV_LinkSet: on the In slot change, using the "reverse following the link" function);
- **Trigger:** allows to force sending values on rising edge;
- **Displaying Mode:** allows to set the displaying mode (RealValue: the Out value is divided by 10; RegisterValue: the value is taken directly from the register);
- **Out:** the point's output slot, the current value;

- **In:** the point's input slot;
- **Point No:** the panel main menu number;
- **Point Display Name:** allows to set the point's display name on the LCD screen (up to 4, only ASCII characters);
- **Point Visibility:** allows to activate or deactivate the point on the display;
- **Point Units:** allows to set the point's units on the display (inactive or displays value unit: °C, °F, Pa, Lx, ppm, m3/h, %RH, L/s, %, h);
- **Point Decimal Place:** allows to set the decimal place on the display (inactive or displays decimal point on first, second, or third position);
- **Point Priority:** allows to set the displaying priority on the LCD screen (starting from the lowest value);
- **Config Trigger:** sends configuration parameters to the device components on rising edge.

The MainMenuNumeric component has the following right-click menu actions:

- **Set:** sets the In slot and sends it to the main menu point;
- **Write:** sends the In slot and sends it to the main menu point;
- **Read:** reads the panel main menu point value and sets the Out slot;
- **Send Value:** sends the user value to main menu point without changing the input slot, from the pop-up window;
- **Read Config:** reads the main menu point configuration parameters from the panel (Point Display Name, Point Visibility, Point Decimal Place, Point Priority);
- **Write Config:** writes the main menu point configuration parameters to the panel (Point Display Name, Point Visibility, Point Decimal Place, Point Priority).

6.7 SubmenuBoolean

Warning!

Component applicable only for the LP panel.

The SubmenuBoolean component is responsible for reading/writing and configuration of a single (one of 8 points) Boolean parameter, which is placed in the LP panel submenus. There are 6 submenus in LP panel: Temperature, Fan, Light, Blind, Alarm, and Occupancy, and each submenu can have up to 8 Boolean points.

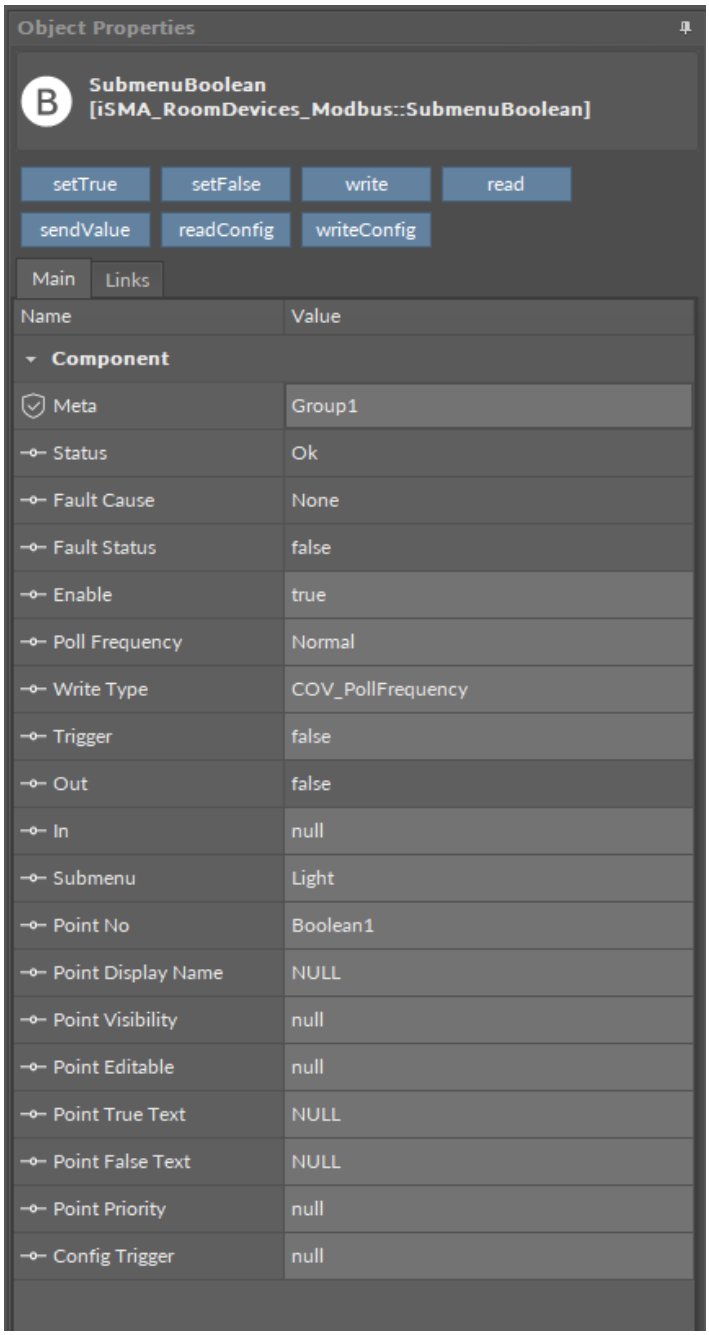


Figure 44. SubmenuBoolean component

The SubmenuBoolean component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** shows the point error status true: point read/write error);
- **Enable:** enables or disables the point (true: enabled, false: disabled);
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Write Type:** allows to set the writing mode (COV: only on input change, COV_PollFrequency: on the In slot change and periodically, PollFrequency: only periodically, COV_LinkSet: on the In slot change, using the "reverse following the link" function);
- **Trigger:** allows to force sending values on rising edge;
- **Out:** the output slot, the current value of read/write register;
- **In:** the register input slot;

- **Submenu:** the panel submenu number;
- **Point No:** the submenu point number;
- **Point Display Name:** allows to set the 4 characters submenu point LCD display name (only ASCII characters);
- **Point Visibility:** allows to activate or deactivate the point on the display;
- **Point Editable:** enables or disables the editing of the point;
- **Point True Text:** allows to set the 4 characters LCD display text in the true state (only ASCII characters);
- **Point False Text:** allows to set the 4 characters LCD display text in the false state (only ASCII characters);
- **Point Priority:** allows to set the displaying priority on the LCD screen (starting from the lowest value);
- **Config Trigger:** sends configuration parameters to the device components on rising edge.

The SubmenuBoolean component has the following right-click menu actions:

- **Set True:** sets the In slot to true and sends it to the submenu point;
- **Set False:** sets the In slot to false and sends it to the submenu point;
- **Write:** sends the In state to the submenu point;
- **Read:** reads the LP panel submenu point value and sets the Out slot;
- **Send Value:** sends the user value to the submenu point without changing the input slot, from the pop-up window;
- **Read Config:** reads the submenu point configuration parameters from the LP panel (Point Display Name, Point Visibility, Point True Text, Point False Text, Point Priority);
- **Write Config:** writes the submenu point configuration parameters to the LP panel (Point Display Name, Point Visibility, Point True Text, Point False Text, Point Priority).

6.8 SubmenuNumeric

Warning!

Component applicable only for the LP panel.

The SubmenuNumeric component is responsible for reading/writing and configuration of a single (one of 8 points) numeric user parameter, which is placed in one of the LP panel submenus. There are 6 submenus in the LP panel: Temperature, Fan, Light, Blind, Alarm, and Occupancy, and each submenu can have up to 8 numeric points.

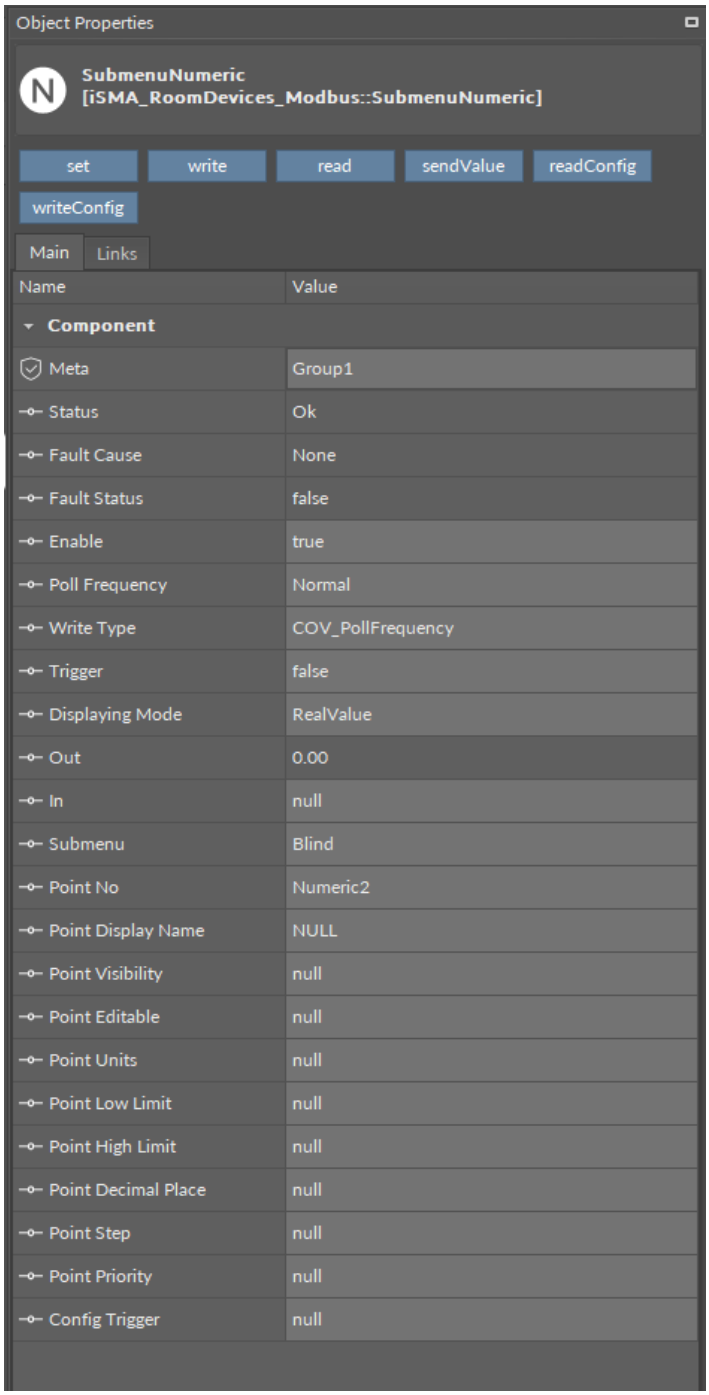


Figure 45. SubmenuNumeric component

The SubmenuNumeric component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** shows the point error status true: point read/write error);
- **Enable:** enables or disables the point (true: enabled, false: disabled);
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Write Type:** allows to set the writing mode (COV: only on input change, COV_PollFrequency: on the In slot change and periodically, PollFrequency: only periodically, COV_LinkSet: on the In slot change, using the "reverse following the link" function);
- **Trigger:** allows to force sending values on rising edge;

- **Displaying Mode:** allows to set the displaying mode (RealValue: the Out value is divided by 10; RegisterValue: the value is taken directly from the register);
- **Out:** the output slot, the current value of the read/write register;
- **In:** the register input slot;
- **Submenu:** the panel submenu number;
- **Point No:** the submenu point number;
- **Point Display Name:** allows to set the 4 characters submenu point LCD display name (only ASCII characters);
- **Point Visibility:** allows to activate or deactivate point on display;
- **Point Editable:** enables or disables the editing of the point;
- **Point Units:** allows to set the point's units on the display (inactive or displays value unit: °C, °F, Pa, Lx, ppm, m3/h, %RH, L/s, %, h);
- **Point Low Limit:** allows to set the submenu parameter low limit value;
- **Point High Limit:** allows to set the submenu parameter high limit value;
- **Point Decimal Place:** allows to set the point's decimal place on the display (inactive or displays decimal point on first, second, or third position);
- **Point Step:** allows to set the Out value changing step;
- **Point Priority:** allows to set the displaying priority on the LCD screen (starting from the lowest value);
- **Config Trigger:** sends configuration parameters to the device components on rising edge.

The SubmenuNumeric component has the following right-click menu actions:

- **Set:** sets the In slot and sends it to the submenu point;
- **Write:** sends the In slot and sends it to the submenu point;
- **Read:** reads the LP panel submenu point value and sets the Out slot;
- **Send Value:** sends the user value to the submenu point without changing the input slot, from the pop-up window;
- **Read Config:** reads the submenu point configuration parameters from the LP panel (Point Display Name, Point Visibility, Point Editable, Point Units, Point Low Limit, Point High Limit, Point Decimal Place, Point Step, Point Priority);
- **Write Config:** writes the submenu point configuration parameters to the LP panel (Point Display Name, Point Visibility, Point Editable, Point Units, Point Low Limit, Point High Limit, Point Decimal Place, Point Step, Point Priority).

6.9 TemperatureSensor

Note: Component applicable for the LP, Touch Point, and FP panels.

The TemperatureSensor component is responsible for reading values and configuration of the temperature sensor in the panel.

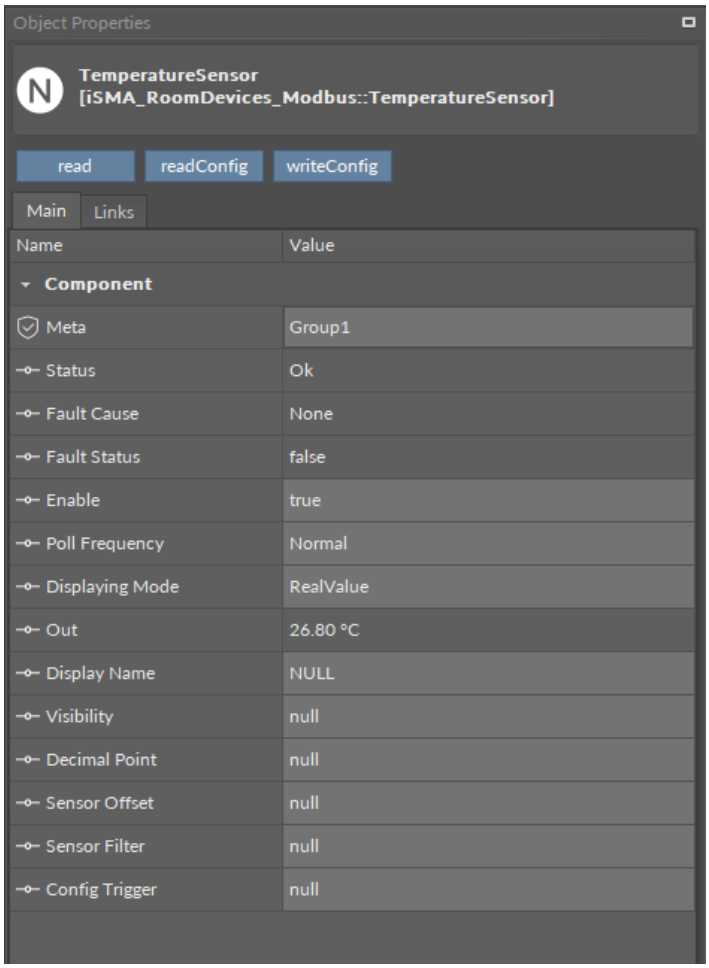


Figure 46. TemperatureSensor component

The TemperatureSensor component has the following slots:

- **Status:** shows the component's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** informs about the point error status (true: point read/write error);
- **Enable:** enables or disables the component (true: enabled, false: disabled);
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Displaying Mode:** allows to set the displaying mode (RealValue: the output value is divided by 10; RegisterValue: the value is taken directly from the register);
- **Out:** the temperature sensor value output slot;
- **Display Name:** allows to set the temperature sensor display name on the LCD display (up to 4 characters, only ASCII characters, only for the LP panel);
- **Visibility:** activates or inactivates the sensor value on the display;
- **Decimal Point:** allows to set the display of the decimal point (inactive or displays decimal point on first, second, or third position; only for the LP panel);
- **Sensor Offset:** sets the temperature sensor offset value;
- **Sensor Filter:** sets the temperature sensor reading filter time in seconds;
- **Config Trigger:** on rising edge sends configuration parameters to the device components (Display Name, Visibility, Decimal Point, Sensor Offset, Sensor Filter).

The TemperatureSensor component has the following actions:

- **Read:** reads the panel's temperature sensor value and updates the Out slot;
- **Read Config:** reads configuration parameters from the panel (Display Name, Visibility, Decimal Point, Sensor Offset, Sensor Filter);

- **Write Config:** writes configuration parameters to the panel (Display Name, Visibility, Decimal Point, Sensor Offset, Sensor Filter).

6.10 Occupancy

Note: Component applicable for the LP, Touch Point, and FP panels.

The Occupancy component is responsible for configuring occupancy settings in the panel.

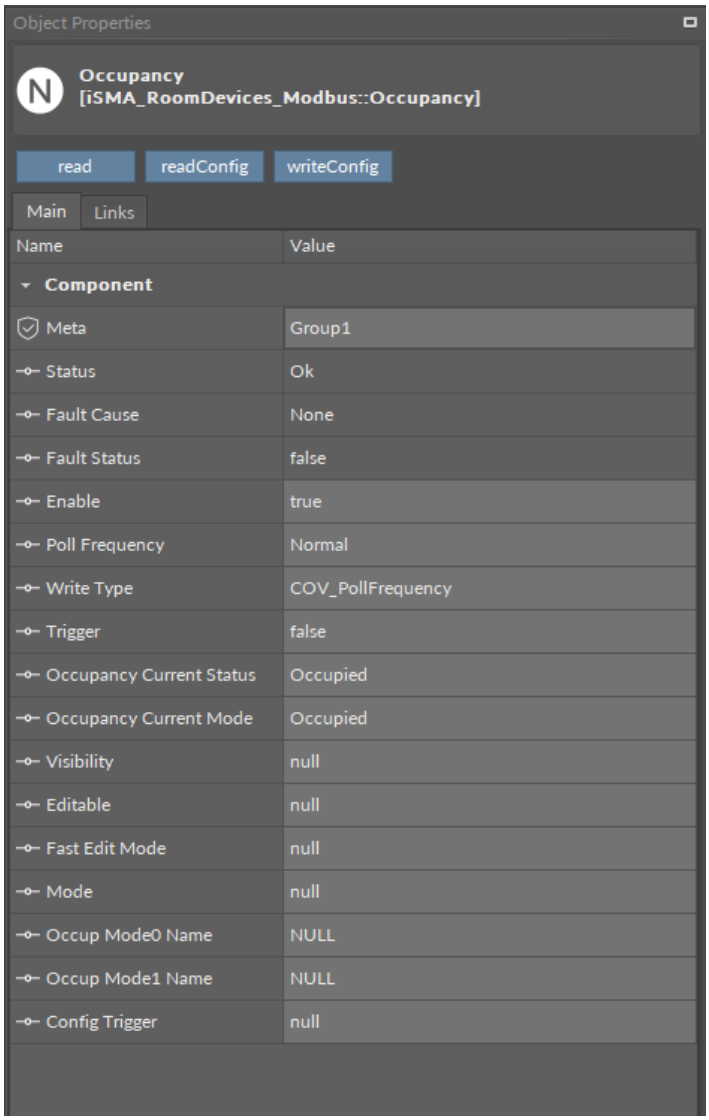


Figure 47. Occupancy component

The Occupancy component has the following slots:

- **Status:** shows the component's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** informs about the point error status (true: point read/write error);
- **Enable:** enables or disables the component (true: enabled, false: disabled);
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Write Type:** defines a method of writing values (COV, COV_PollFrequency, PollFrequency, COV_LinkSet);
- **Trigger:** allows to force sending values on rising edge;

- **Occupancy Current Status:** sets the current status of occupancy (Unoccupied, Occupied, Standby, ForcedOccupied);
- **Occupancy Current Mode:** sets the current mode of occupancy (Unoccupied, Occupied);
- **Visibility:** allows to activate or deactivate the point on the display (only for the LP panel);
- **Editable:** enables or disables editing of an occupancy in the panel;
- **Fast Edit Mode:** enables a fast edit mode in the panel (only for the LP panel);
- **Mode:** identifies a way of controlling the panel (LocalMode, BmsMode);
- **Occup Mode0 Name-Occup Mode1 Name:** allows to set different occupancy mode names (up to 4, only ASCII characters; only for the LP panel);
- **Config Trigger:** on rising edge sends configuration parameters to the device components (Editable, Mode).

The Occupancy component has the following actions:

- **Read:** reads the panel's occupancy values and updates the Occupancy Current Status and Occupancy Current Mode slots;
- **Read Config:** reads configuration parameters from the panel (Editable, Mode);
- **Write Config:** writes configuration parameters to the panel (Editable, Mode).

6.11 TemperatureSetpoint

Note: Component applicable for the LP, Touch Point, and FP panels.

The TemperatureSetpoint component is responsible for configuring a temperature setpoint in the panel.

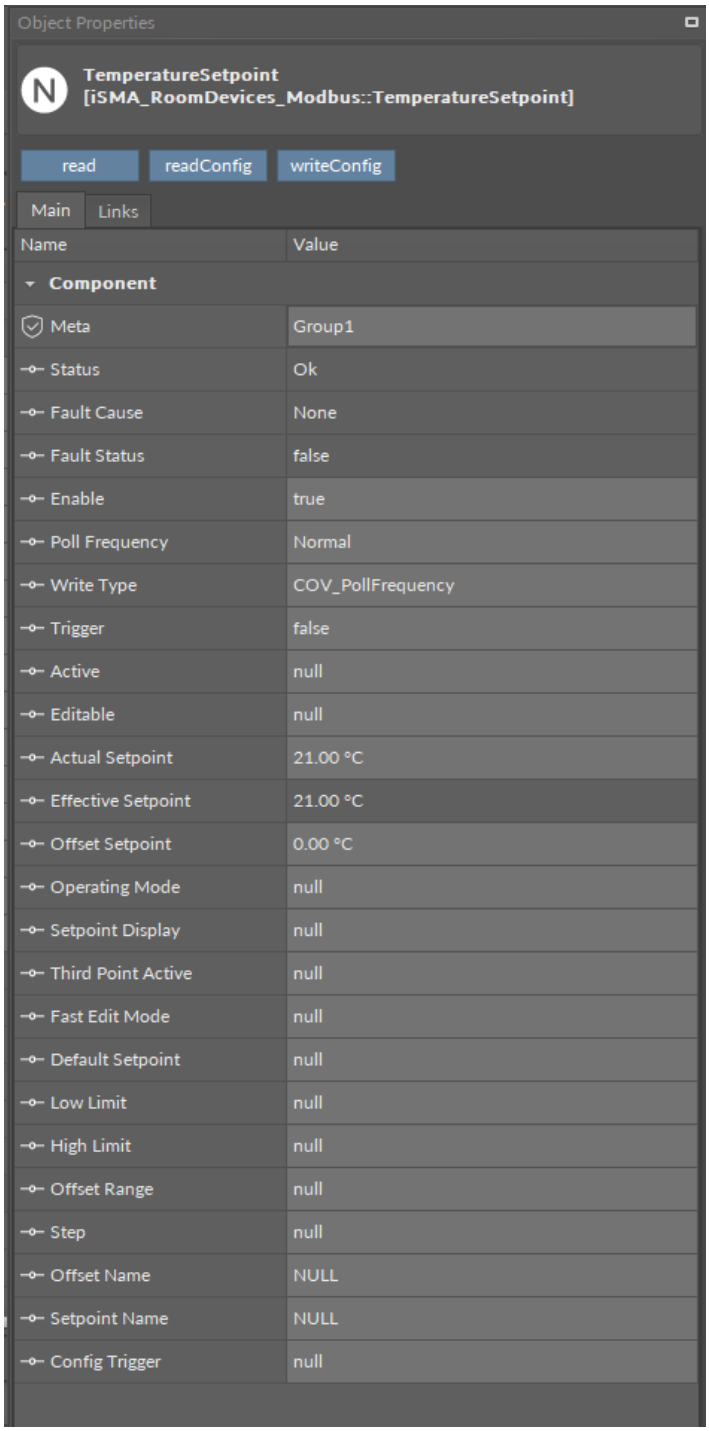


Figure 48. TemperatureSetpoint component

The TemperatureSetpoint component has the following slots:

- **Status:** shows the component's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** informs about the point error status (true: point read/write error);
- **Enable:** enables or disables the component (true: enabled, false: disabled);
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Write Type:** defines a method of writing values (COV, COV_PollFrequency, PollFrequency, COV_LinkSet);
- **Trigger:** allows to force sending values on rising edge;
- **Active:** allows to activate or deactivate the point on the display (only for the LP panel);
- **Editable:** enables or disables editing of a temperature setpoint in the panel;

- **Actual Setpoint:** reads a current temperature setpoint value set in the panel;

Note: If the Write Type slot is set to COV_LinkSet and the Actual Setpoint slot is linked to the Out slot in the NVNumericWritable component, after reading the value from the panel, it is sent to the NVNumericWritable component.

- **Effective Setpoint:** displays an effective setpoint value (actual setpoint and offset);
- **Offset Setpoint:** allows to set an offset for a temperature setpoint value;
- **Operating Mode:** defines a mode of calculating a setpoint (OffsetValue, SetpointValue);
- **Setpoint Display:** defines a way of displaying a setpoint (OffsetValue, SetpointValue);
- **Third Point Active:** allows to activate or deactivate a decimal point on the display;
- **Fast Edit Mode:** enables a fast edit mode in the panel (only for the LP panel);
- **Default Setpoint:** defines a default setpoint for the panel;
- **Low Limit:** sets the lowest limit for a setpoint value;
- **High Limit:** sets the highest limit for a setpoint value;
- **Offset Range:** sets a range of the offset value for a setpoint (only for the LP panel);
- **Step:** identifies a minimum difference between next setpoint values (step value);
- **Offset Name:** allows to set a name for the temperature setpoint offset (up to 4, only ASCII characters; only for the LP panel)
- **Setpoint Name:** allows to set a name for the temperature setpoint (up to 4, only ASCII characters; only for the LP panel);
- **Config Trigger:** on rising edge sends configuration parameters to the device components (Editable, Mode).

The TemperatureSetpoint component has the following actions:

- **Read:** reads the panel's temperature setpoint value and updates the Out slot;
- **Read Config:** reads configuration parameters from the panel (Active, Editable, Operating Mode, Setpoint Display, Third Point Active, Default Setpoint, Low Limit, High Limit, Offset Range, Step);
- **Write Config:** writes configuration parameters to the panel (Active, Editable, Operating Mode, Setpoint Display, Third Point Active, Default Setpoint, Low Limit, High Limit, Offset Range, Step).

7 Advance Control Kit

In order to facilitate the building of user applications, the Advance Control kit has been developed. Using components of this kit, the user can build an advanced application in a simple way.

7.1 DimmerSwitch

The DimmerSwitch component controls a light dimmer with a use of a single-button (one digital input) or two buttons (two digital inputs).

In the SingleSwitch mode, switch 1 has a defined function for short and long press. A short press is when the button is pressed for less than the time defined in the Short In slot. A long press is when the button is pressed for longer than the time defined in the Short In slot. The short press is dedicated to on/off switching, the long press is dedicated to changing the dimming value. Each short press toggles between on and off state. During the long-press, the component increases or decreases the dimming value.

In the DoubleSwitch mode, each button has two functions defined: switch 1 is for switching on (short press) and increasing the dimmer value (long press), switch 2 is for switching off (short press) and decreasing the dimmer value (long press). The short press is when the high state time is less than the time defined in the Short In slot. The long press is when the short time elapses, and the button is still in a high state.

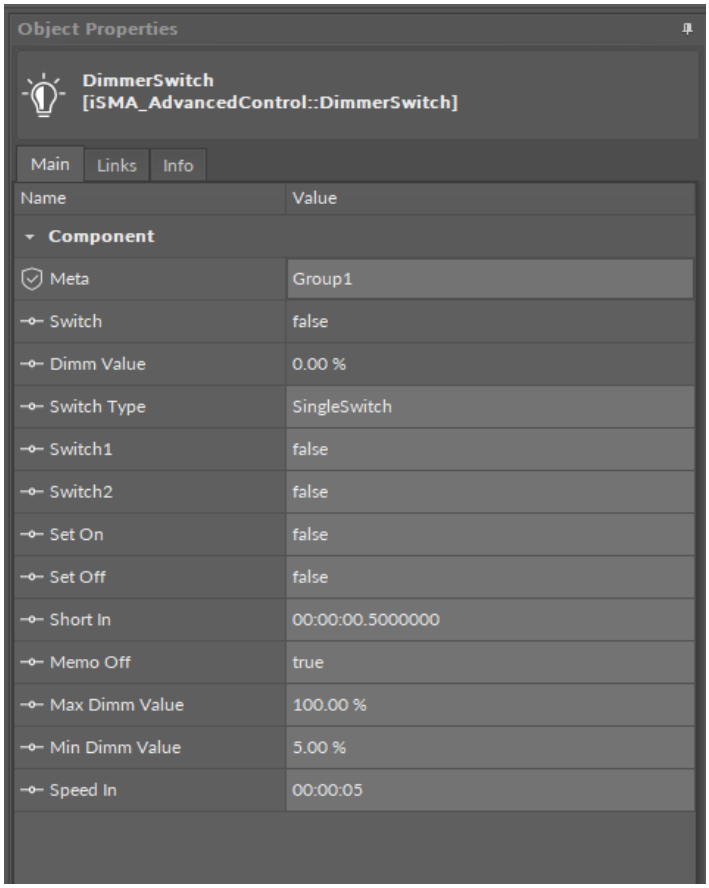


Figure 49. DimmerSwitch component

The DimmerSwitch component has the following slots:

- **Switch:** the Out slot for the dimmer, shows the digital value for on and off states;

- **Dimm Value:** the Out slot for the dimmer, shows the analog value in the range 0-100%;
- **Switch Type:** the button input;
- **Switch1:** the button input, main for the SingleSwitch mode, on or increasing output in the DoubleSwitch mode;
- **Switch2:** the button input, not active for the SingleSwitch mode, off or decreasing output in the DoubleSwitch mode;
- **Set On:** triggers the dimmer switch to on state (to max. level);
- **Set Off:** triggers the dimmer switch to off state;
- **Short In:** sets the time for the short button press;
- **Memo Off:** enables or disables the memory function of the Dimm Value during switch off;
- **Max Dimm Value:** sets the max. dimmer analog value;
- **Min Dimm Value:** sets the min. dimmer analog value;
- **Speed In:** sets the dimming speed time.

7.2 ActionTrigger

The ActionTrigger component remotely invokes action from the Sedona component. Sedona does not allow for making links to component's actions, so actions can be invoked manually from a programming software (for example, the iSMA Tool), or by a dedicated component. The ActionTrigger component has 3 input slots, each dedicated to a different Sedona variable type (typically, only one of them is used, the one corresponding to the component type). The program creates a link from the component, which the action will be invoked from, to the ActionTrigger component input slot. The Action Name slot defines, which action is to be invoked.

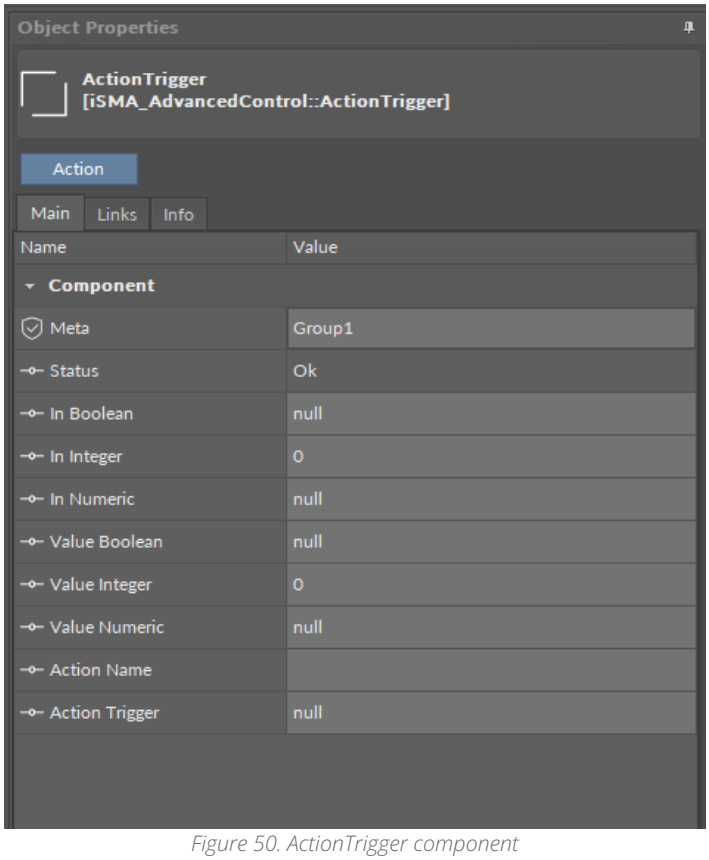


Figure 50. ActionTrigger component

The ActionTrigger component has the following slots:

- **Status:** shows the component's status;
- **In Boolean:** the input slot to create a link connection between components of the Boolean type;
- **In Integer:** the input slot to create a link connection between components of the integer type;
- **In Numeric:** the input slot to create a link connection between components of the numeric/float type;
- **Action Name:** allows to define the action name from the linked component, which will be invoked;
- **Action Trigger:** allows to invoke the action from the component linked to one of the input slots, defined in the Action Name slot.

The ActionTrigger component has the following action:

- **Action:** manually invokes the action from the linked component.

7.3 RaiseLower

The RaiseLower component simplifies a control of 3-point valve actuators. This component has the following functions:

- analog input, works with PID regulators;
- 2 digital outputs for 3-point direct control valve actuators;
- analog output for 3-point control valve actuators by voltage level (additional device required);
- midnight reset function to automatically adjust physical and virtual valve position.

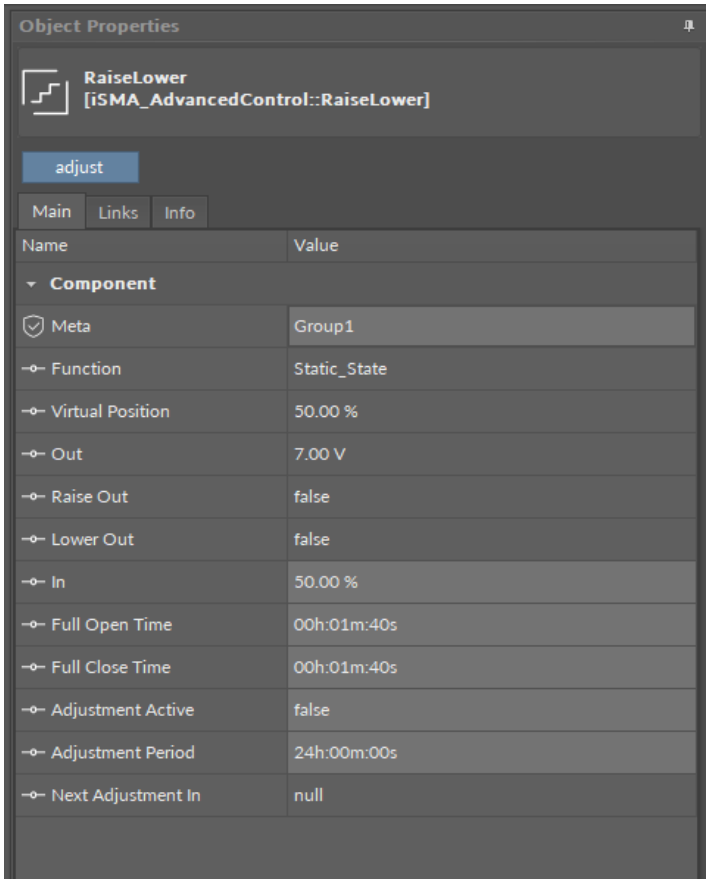


Figure 51. RaiseLower component

The RaiseLower component has the following slots:

- **Function:** the current function description (Lower_State, Static_State, Raise_State, Adjustment_Opening, Adjustment_Closing);
- **Virtual Position:** shows the valve virtual position in %;
- **Out:** the analog output slot;
- **Rise Out:** the digital output for the rising function;
- **Rise Low:** the digital output for the lowing function;
- **In:** allows to set the valve position demand;
- **Full Open Time:** allows to set the time for opening valve position from 0% to 100%;
- **Full Close Time:** allows to set the time for closing valve position from 100% to 0%;
- **Adjustment Active:** triggers the remote adjustment procedure;
- **Adjustment Period:** allows to set the adjustment procedure recall period in hours;
- **Next Adjustment In:** shows the time (in minutes) to the next adjustment procedure.

The RaiseLower component has the following action:

- **Adjust:** valve adjustment procedure recall (adjusts physical and virtual valve position).

Out Value	Raise	Lower	Description
0	Off	Off	Off
4	Off	On	Lower
7	Off	Off	Static
10	On	Off	Rise

Table 3. The RaiseLower component values

8 BACnet Master-Slave Kit

The iSMA-B-FCU device can work in defined networks, where one device is a master and remaining devices (slaves) follow the master's parameters. The single master device can have up to 5 slave devices, and it is possible to share up to 100 points with them.

Note: This function is available only in the BACnet MS/TP protocol, using the RS485 port (COM1).

8.1 BACnetMasterSlaveFolder

The BACnetMasterSlave folder is a component, which groups and organizes the BACnet Master-Slave components.

The BACnetMasterSlave folder has no slots or actions.

8.2 BACnetMasterSlaveNetwork

The BACnetMasterSlaveNetwork is the main component responsible for the communication between a master device and slave devices by the BACnet MS/TP protocol, using the RS485 port (COM1). The BACnetMasterSlaveNetwork sets parameters such as slave devices Id, communication parameters (such as a poll frequency or maximum write time), and reads the status of slave devices. The component has to be placed under the Drivers folder.

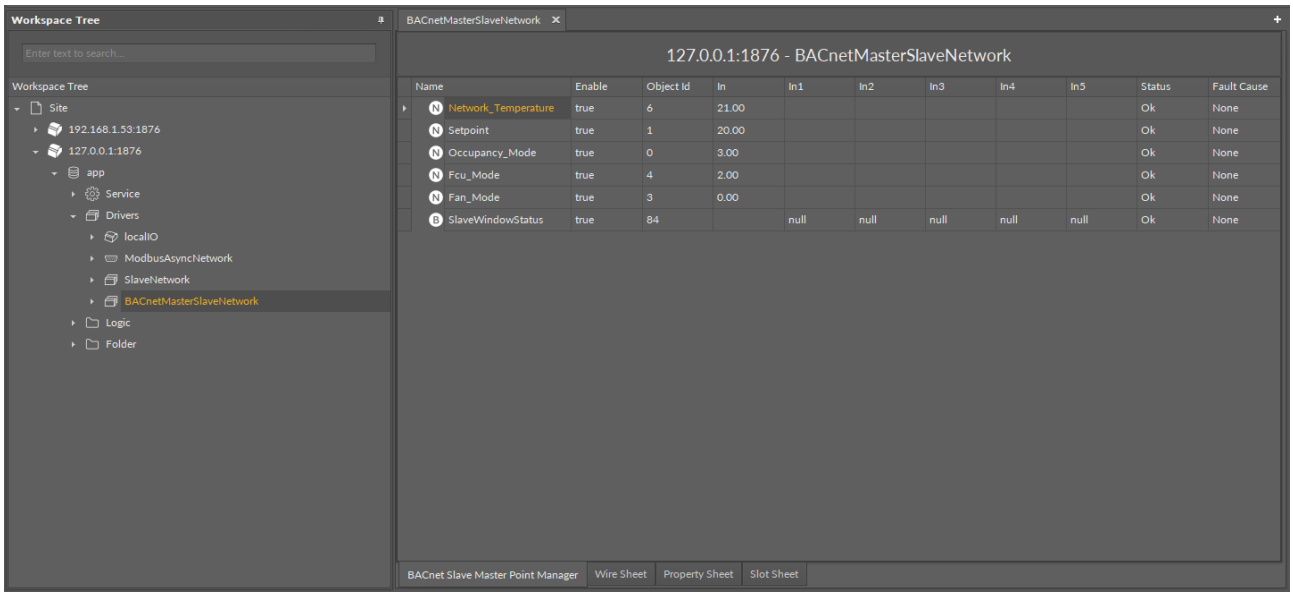


Figure 52. BACnetMasterSlaveNetwork component

The BACnetMasterSlaveNetwork component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Enable:** enables or disables the network (true: enabled, false: disabled);
- **Slave1DeviceId-Slave5DeviceId:** allow to set the BACnet Ids of slave devices;
- **Slave Status1-Slave Status5:** reads statuses of slave devices;
- **Poll Frequency:** allows to set the polling frequency of all read-only points, min. 1;
- **Max Write Time:** allows to set the maximum time between sending values of all writeable points–if the value equals 0, values of writable points are sent only "on value change";

- **Ping Frequency:** allows to set the time between testing messages to check slave devices connection, min. 1;
- **Ping Trigger:** triggers the Ping action on the rising edge of value;
- **Free Points:** shows the number of free BACnet Master-Slave points.

The BACnetMasterSlaveNetwork component offers the following action, available in the context menu:

- **Ping:** sends a test message to slave devices to check their statuses.

8.2.1 BinaryValueWrite

The BinaryValueWrite component is responsible for sending binary values to slave devices. Values are written if the value of In slot has changed, in time periods defined in the Max Write Time slot (only if the value of the Max Write Time slot is higher than 0). Writing can be also invoked by the Set or Write actions. The BinaryValueWrite component has to be placed under the BACnetMasterSlaveNetwork component.

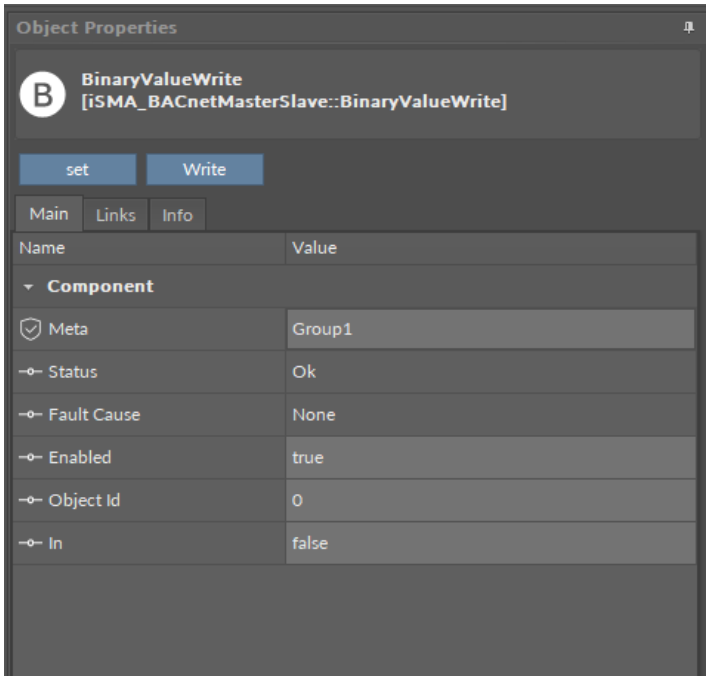


Figure 53. BinaryValueWrite component

The BinaryValueWrite component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Enable:** enables or disables the network (true: enabled, false: disabled);
- **Object Id:** allows to set the BACnet Id of the point;
- **In:** the input slot, which value is sent to slave devices.

The BinaryValueWrite component offers the following actions, available under the right mouse button:

- **Set:** writes the value to the In slot and sends it to slave devices;
- **Write:** sends the value from the In slot to slave devices.

8.2.2 AnalogValueRead

The AnalogValueRead component is responsible for reading analog values from slave devices. Values are read in time periods defined in the Poll Frequency slot. Reading can be also forced using the Read action. The AnalogValueRead component has to be placed under the BACnetMasterSlaveNetwork component.

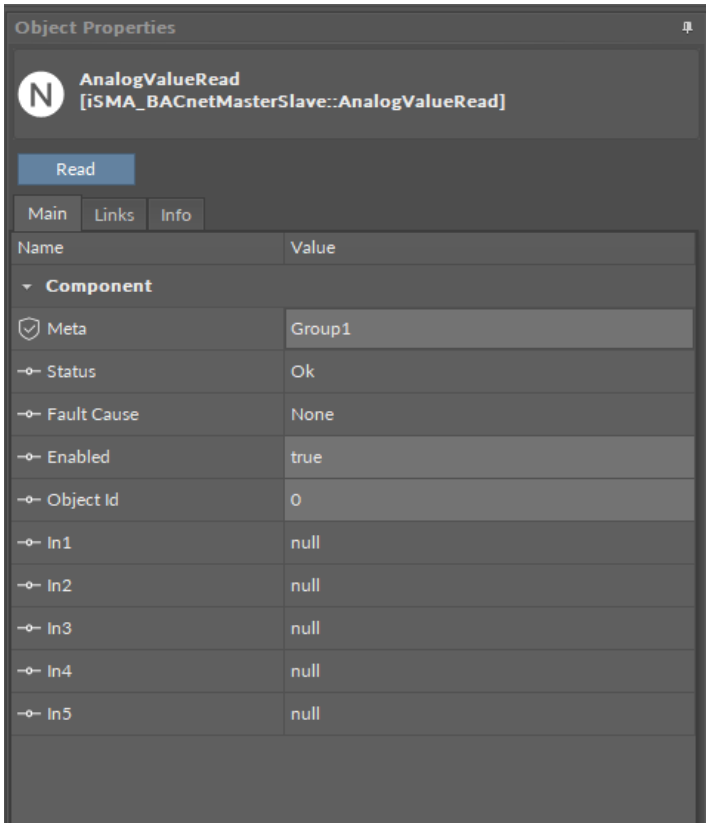


Figure 54. AnalogValueRead component

The AnalogValueRead component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Enable:** enables or disables the network (true: enabled, false: disabled);
- **Object Id:** allows to set the BACnet Id of the point;
- **In1-In5:** slots storage values of points from the corresponding slave devices.

Note: If the communication with some slave devices is be broken, the corresponding slot stores the last value, which has been read, and the Status slot displays "Some Points Down".

The AnalogValueRead component offers the following action, in the context menu:

- **Read:** Action enforces the reading of the point.

8.2.3 AnalogValueWrite

The AnalogValueWrite component is responsible for sending analog values to slave devices. Values are written if the value of In slot has changed, in time periods defined in the Max Write Time slot (only if the value of the Max Write Time slot is higher than 0). Writing can be also invoked by the Set or Write actions. The AnalogValueWrite component has to be placed under the BACnetMasterSlaveNetwork component.

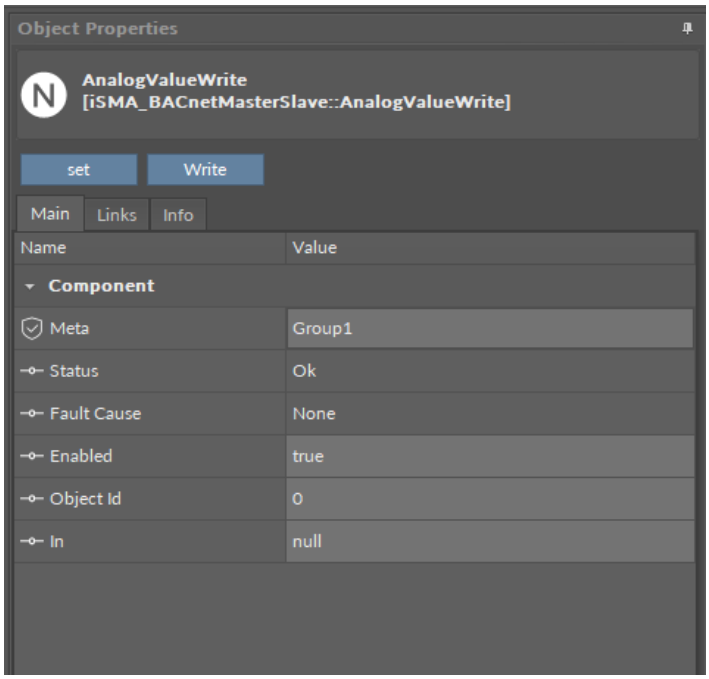


Figure 55. AnalogValueWrite component

The AnalogValueWrite component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Enable:** enables or disables the network (true: enabled, false: disabled);
- **Object Id:** allows to set the BACnet Id of the point;
- **In:** the input slot, which value is sent to slave devices.

The AnalogValueWrite component offers the following actions, available in the context menu:

- **Set:** writes the value to the In slot and sends it to slave devices;
- **Write:** sends the value from the In slot to slave devices.

8.2.4 BinaryValueRead

The BinaryValueRead component is responsible for reading binary values from slave devices. Values are read in time periods defined in the Poll Frequency slot. Reading can be also invoked using the Read action. The BinaryValueRead component has to be placed under the BACnetMasterSlaveNetwork component.

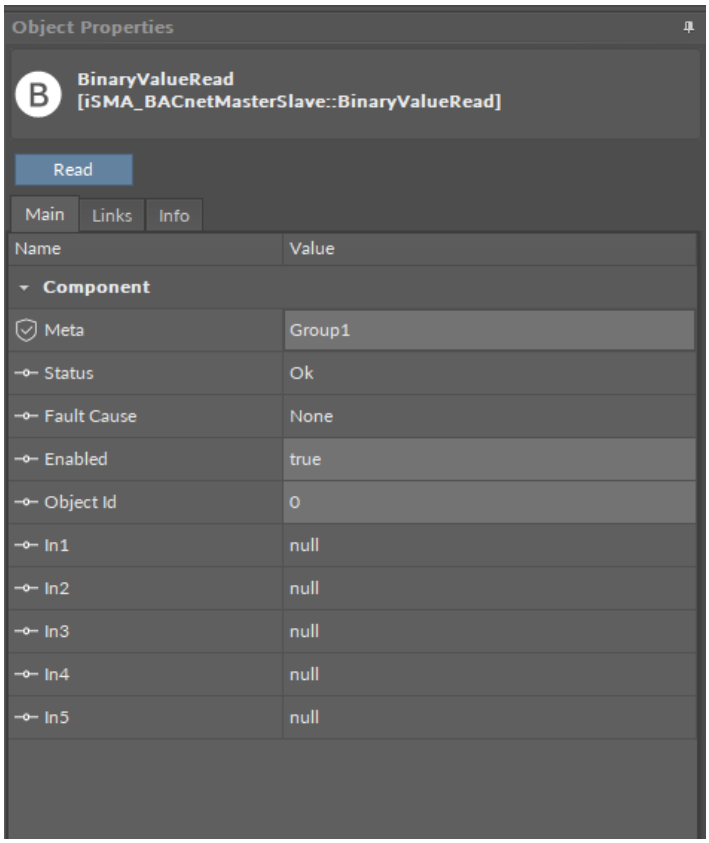


Figure 56. BinaryValueRead component

The BinaryValueRead component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Enable:** enables or disables the network (true: enabled, false: disabled);
- **Object Id:** allows to set the BACnet Id of the point;
- **In1-In2:** shows the statuses of points from slave devices.

The BinaryValueRead component offers the following action, available in the context menu:

- **Read:** enforces reading of the point.

9 Modbus Async Network Kit

This section provides a collection of procedures to be used for the iSMA-B-FCU Modbus drivers in order to build networks of devices with Modbus points. The iSMA-B-FCU device has two RS485 ports. The COM2 port (with RJ12 connector) can be used as a Modbus RTU/ASCII master. There is no software limit of devices connected to the Modbus Async bus, but the total number of all Modbus points cannot exceed 200.

9.1 Modbus Folder

The ModbusFolder is a component grouping and organizing Modbus Async Network kit's components.

The ModbusFolder component has no slots or actions.

9.2 ModbusAsyncNetwork

The Modbus network is the main component responsible for servicing the RS485 port (COM2). The ModbusAsyncNetwork component has to be placed under the Drivers folder. The ModbusAsyncNetwork sets parameters such as the communication baud rate and data format, testing, etc., and maintains statistical data.

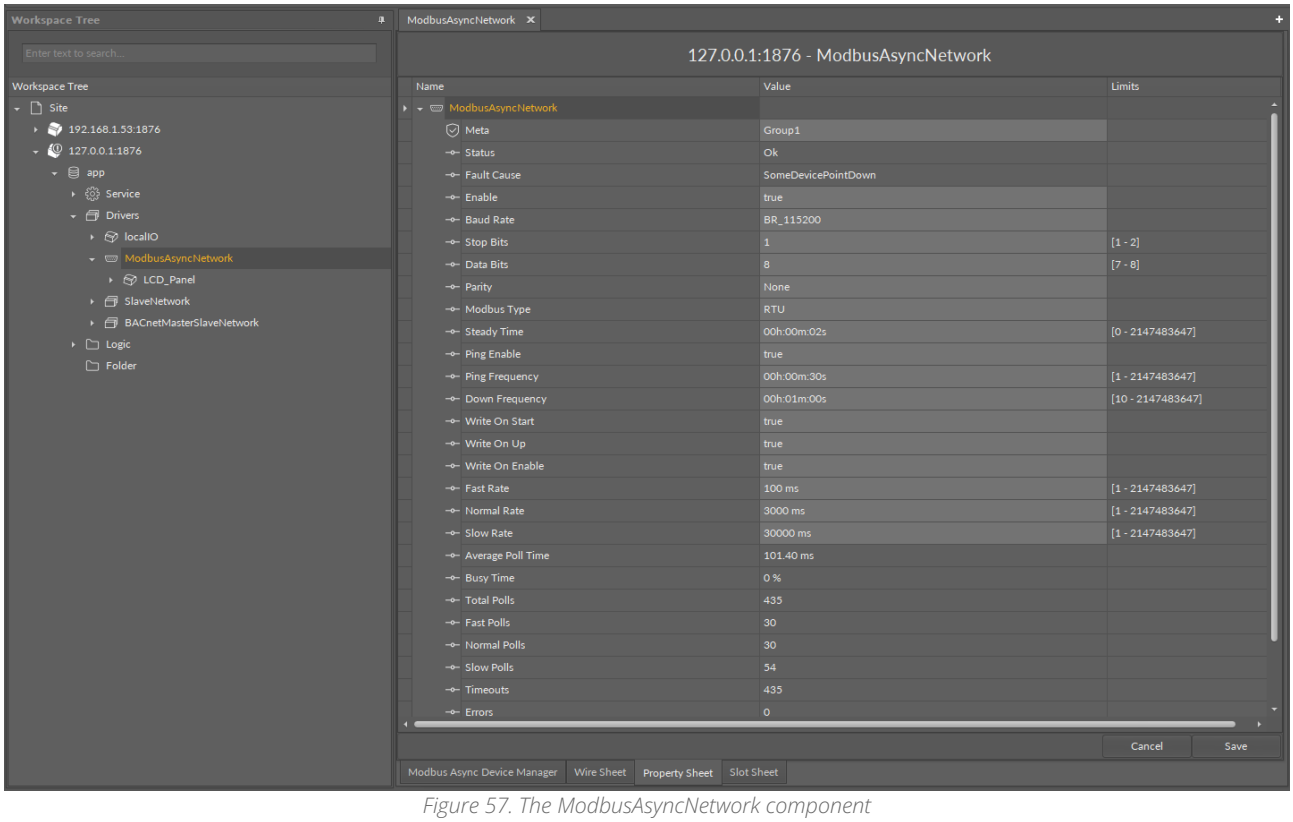


Figure 57. The ModbusAsyncNetwork component

The ModbusAsyncNetwork component has the following slots:

- **Status:** shows the network status;
- **Fault Cause:** shows the fault cause description;
- **Enable:** switches on/off the Modbus network (true: network enabled, false: network disabled);
- **BaudRate:** allows to set the Modbus RS485 port baud rate (2400, 4800, 9600, 19200, 38400, 57600, 115200 bps);

- **Stop Bits:** displays the number of stop bits (1 bit, 2 bits);
- **Data Bits:** displays the number of data bits (7 bits or 8 bits);
- **Parity:** configures the parity bit (None, Odd, Even, Always1, Always0);
- **Modbus Type:** allows to set the Modbus type (RTU or ASCII);
- **Steady Time:** allows to set the network start-up delay time after the power-up or reset;
- **Ping Enable:** enables devices connection testing function;
- **Ping Frequency:** allows to set the time lapse between the testing message and check device connection;
- **Down Frequency:** allows to set the time lapse between testing message for devices or points, which have gone into the down status (min. value 10s);
- **Write On Start:** writes action in the device's writable components in the Modbus network after the reset or power-up;
- **Write On Up:** writes action in the device's writable components in the Modbus network after the connection with Modbus device has been restored;
- **Write On Enable:** writes action in the device's writable components in the Modbus network after enabling the device;
- **Fast Rate:** allows to set the time between messages in the fast mode poll frequency;
- **Normal Rate:** allows to set the time between messages in the normal mode poll frequency;
- **Slow Rate:** allows the time between messages in the slow mode poll frequency;
- **Average Poll Time:** shows the average time of sending/receipt of one message;
- **Busy Time:** shows the percentage of the Modbus network usage;
- **Total Polls:** shows the total number of messages;
- **Fast Polls:** shows the number of messages sent in the fast mode;
- **Normal Polls:** shows the number of messages sent in the normal mode;
- **Slow Polls:** shows the number of messages sent in the slow mode;
- **Timeouts:** shows the number of lost messages (the difference between messages sent and received);
- **Errors:** shows the number of error messages (for example, with the wrong CRC);
- **Free points:** shows the number of available physical points in the Modbus network.

The ModbusAsyncNetwork component offers the following actions, available in the context menu:

- **Reset Stats:** resets statistics of the Modbus Async network;
- **Enable:** enables the Modbus Async network;
- **Disable:** disables the Modbus Async network.

9.2.1 ModbusAsyncBooleanPoint

The ModbusAsyncBooleanPoint is a component responsible for reading Boolean values from the device. The component has to be placed under the ModbusAsyncDevice component.

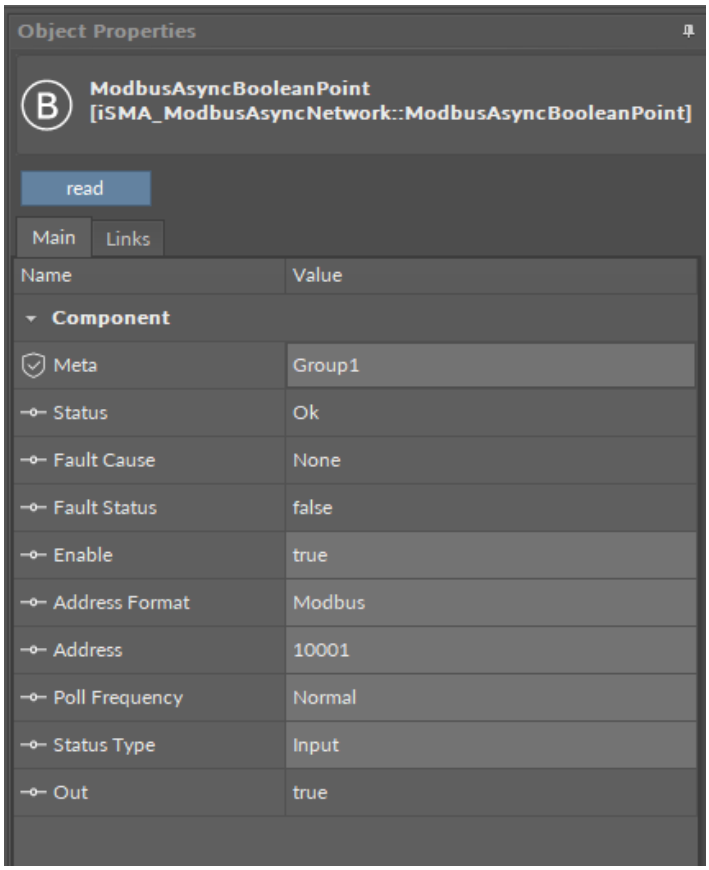


Figure 58. ModbusAsyncBooleanPoint

The ModbusAsyncBooleanPoint component has the following slots:

- **Status:** shows the point's status,
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** shows the point error status (true: point read error);
- **Enable:** enables/disables the point (true–point enabled, false–point disabled);
- **Address Format:** allows to set the register address format (Modbus, decimal);
- **Address:** allows to set the register address;
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Status Type:** allows to set the type of reading the register (input, coil);
- **Out:** the current value of reading register.

The ModbusAsyncBooleanPoint component offers the following action, available in the context menu:

- **Read:** enforces reading of the point.

9.2.2 ModbusAsyncRegisterBitPoint

The ModbusAsyncRegisterBitPoint component is responsible for reading Boolean values from the device. The component has to be placed under the ModbusAsyncDevice component.

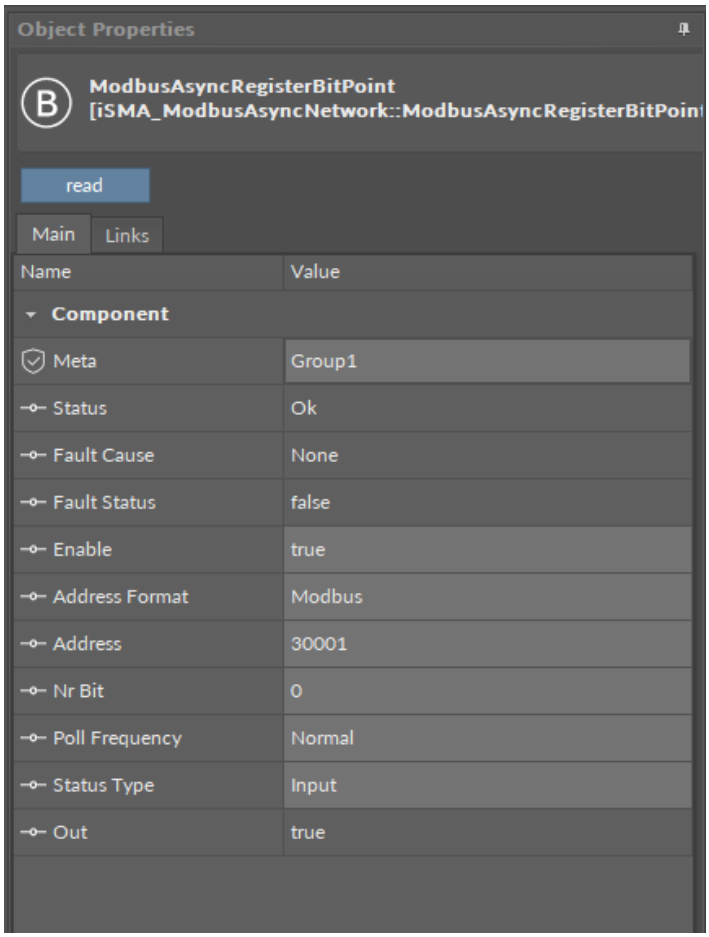


Figure 59. ModbusAsyncRegisterBitPoint

The ModbusAsyncRegisterBitPoint component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** informs about the point error status (true: point read error);
- **Enable:** enables or disables the point (true: point enabled, false: point disabled);
- **Address Format:** allows to set the register address format (Modbus, decimal);
- **Address:** allows to set the register address;
- **Nr Bit:** allows to set the bit number in the register;
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Status Type:** allows to set the type of reading the register (input, coil);
- **Out:** the current value of the read bit.

The ModbusAsyncRegisterBitPoint component offers the following action, available in the context menu:

- **Read:** enforces reading of the point.

9.2.3 ModbusAsyncNumericPoint

The ModbusAsyncNumericPoint component is responsible for reading numeric values from the device. The component has to be placed under the ModbusAsyncDevice component.

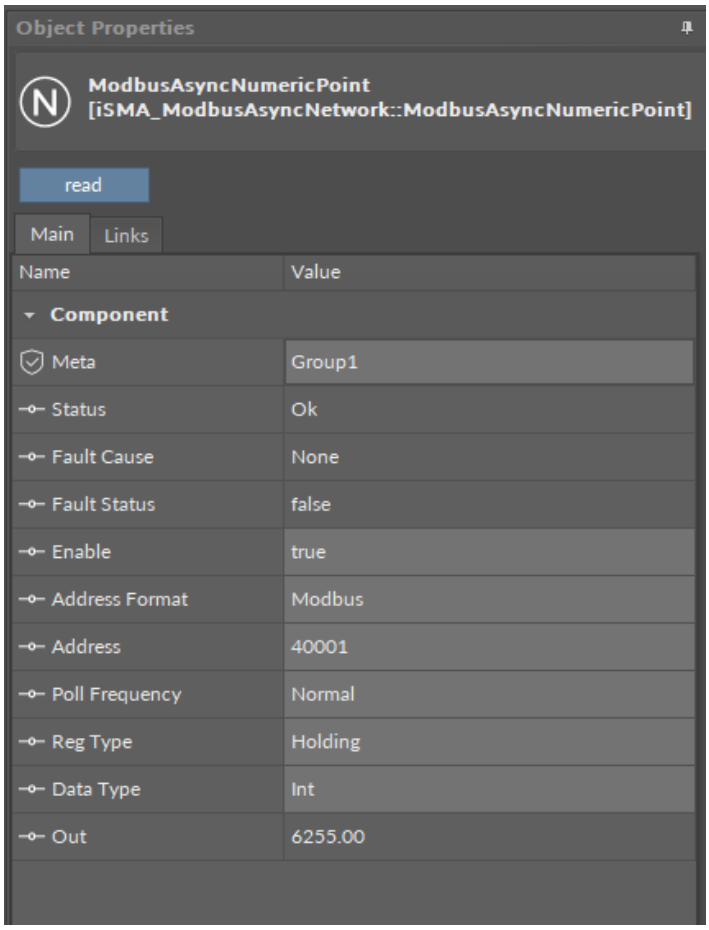


Figure 60. ModbusAsyncNumericPoint

The ModbusAsyncNumericPoint component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** informs abou the point's error status (true: point read error);
- **Enable:** enables or disables the point (true: point enabled, false: point disabled);
- **Address Format:** allows to set the register address format (Modbus, decimal);
- **Address:** allows to set the register address;
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Reg Type:** allows to set the type of reading the register (input, holding);
- **Data Type:** allows to set the read register data type (Int: 16-bits, Long: 32- bits, Float: 32-bits float-point, SInt: 16-bits with the sign, Slong: 32-bits with the sign);
- **Out:** the current value of the read register.

The ModbusAsyncNumericPoint component offers the following action, available in the context menu:

- **Read:** enforces reading of the point.

9.2.4 ModbusAsyncNumericWritable

The ModbusAsyncNumericWritable component is responsible for sending and reading numeric values from the device. The component has to be placed under the ModbusAsyncDevice component.

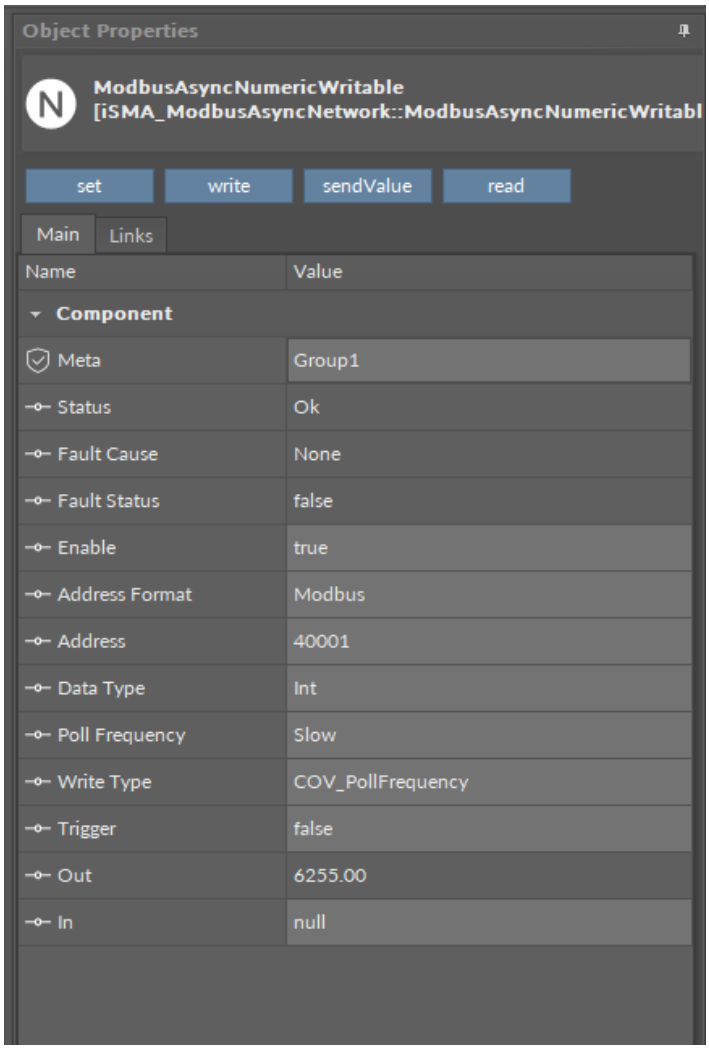


Figure 61. ModbusAsyncNumericWritable

The ModbusAsyncNumericWritable component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** informs about the point error status (true: point read/write error);
- **Enable:** enables or disables the point (true: point enabled, false: point disabled);
- **Address Format:** allows to set the register address format (Modbus, decimal);
- **Address:** allows to set the register address;
- **Data Type:** allows to set the read/write register data type (Int: 16-bits, Long: 32-bits, Float: 32-bits float-point, SInt: 16-bits with the sign, Slong: 32-bits with the sign);
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Write Type:** allows to set the writing mode (COV: only on input change, COV_PollFrequency: on input change and periodically, PollFrequency: only periodically, COV_LinkSet: only on input change using the "reverse following the link" function);
- **Trigger:** allows to force sending values on rising edge;
- **Out:** the output slot, shows the current value of device register;
- **In:** the input slot.

The ModbusAsyncNumericWritable component offers the following actions, available in the context menu:

- **Set:** writes the value to the In slot and sends it to the device;
- **Write:** sends the value from the In slot to the device;

- **Read:** reads the value from the device and sends it to the Out slot;
- **Send Value:** sends the user value from the pop-up window, without changing the In slot.

9.2.5 ModbusAsyncDevice

The ModbusAsyncDevice is a component responsible for servicing a physical device connected to the Modbus network. The component has the Ping action, available in the context menu, which sends a test message to the device to check its status. Each ModbusAsyncDevice has the Ping Address container slot with 3 properties slots (Address Format, Ping Address Reg, Ping Type). These properties specify a particular data address (either input register or holding register) to be used as the device status test (meaning the monitoring ping requests). Ping requests are generated at the network-level by the configurable network monitor (ModbusNetwork -> Ping Enabled). If enabled, the network's monitor periodically pings (queries) this address. Reception of any response from the device, including an exception response, is considered proof of communication, and the Modbus client device is no longer considered down if it had previously gone into the Down status.

The ModbusAsyncDevice component has to be placed under the ModbusAsyncNetwork component.

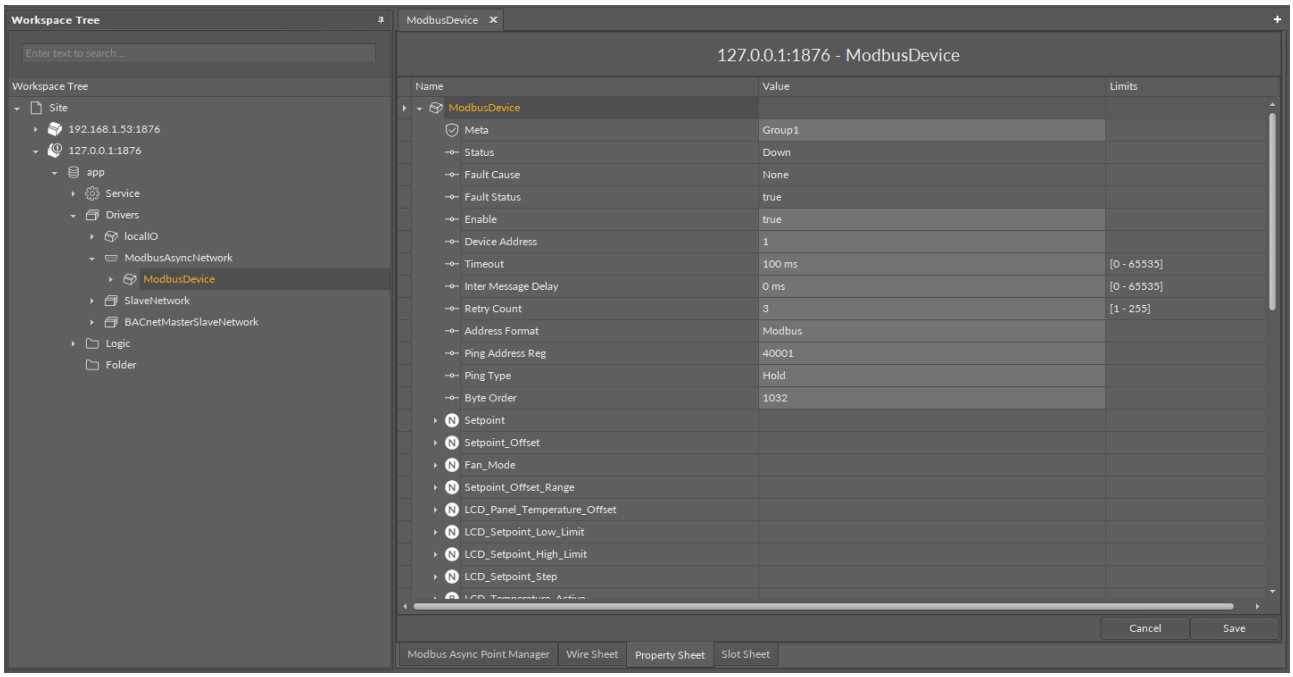


Figure 62. The ModbusAsyncDevice component

The ModbusAsyncDevice component has the following slots:

- **Status:** show the device's actual status (read-only);
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** informs about the device error status (true: device communication error);
- **Enable:** enables/disables the device;
- **Device Address:** the Modbus device physical address (0: network broadcast address, 1-248 addressing range);
- **Timeout:** the maximum device response time calculated from the device request;
- **Inter Message Delay:** allows to set the time between sending messages to the device;

- **Retry Count:** allows to set the maximum number of error messages (CRC error, lost messages);
- **Address Format:** allows to set the Modbus address format (Modbus, decimal);
- **Ping Address Reg:** allows to set any register number (input or holding type), which will be read for the device connection test;
- **Ping Type:** the tested register type (input/holding);
- **Byte Order:** allows to set the byte order reading 32 bit (3210–big endian, 1032–little endian).

The ModbusAsyncDevice component offers the following action, available in the context menu:

- **Ping:** sends a test message to the device to check its status.

9.2.6 ModbusAsyncBooleanWritable

The ModbusAsyncBooleanWritable is a component responsible for sending and reading Boolean values from the device. The component has to be placed under the ModbusAsyncDevice component.

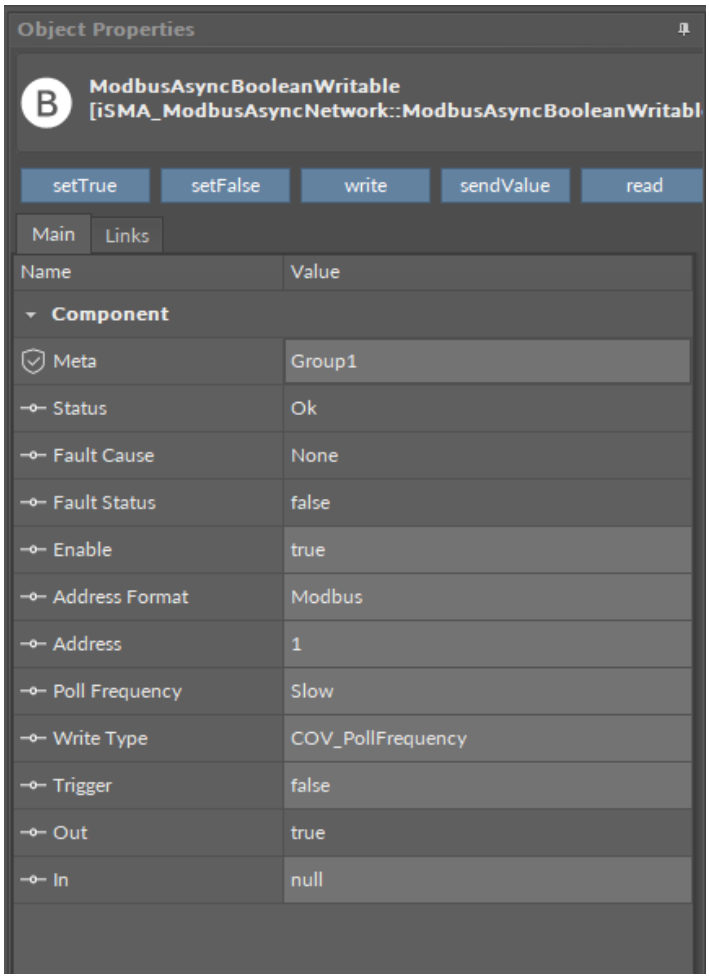


Figure 63. ModbusAsyncBooleanWritable

The ModbusAsyncBooleanWritable component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** informs about the point' error status (true: point read/write error);
- **Enable:** enables or disabled the point (true: point enabled, false: point disabled);

- **Address Format:** allows to set the register address format (Modbus, decimal);
- **Address:** allows to set the register address;
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Write Type:** allows to set the writing mode (COV: only on input change, COV_PollFrequency: on input change and periodically, PollFrequency: only periodically, COV_LinkSet: only on input change using the "reverse following the link" function);
- **Trigger:** allows to force sending values on rising edge;
- **Out:** the output slot, the current value of read/write registry;
- **In:** the input slot.

The ModbusAsyncBooleanWritable component offers the following actions, available in the context menu:

- **Set True/Set False:** writes a value to the In slot and sends it to the device (not active if the In slot has a link connected);
- **Write:** sends the value from In slot to the device;
- **Send Value:** sends the user value from the pop-up window, without changing the In slot;
- **Read:** reads the value from the device and sends it to the Out slot.

9.2.7 ModbusAsyncNumericMultiPoint

The ModbusAsyncNumericMultiPoint component is responsible for reading up to 8 16-bit registers from the device in one message. The component uses the 0x03 Modbus command. The component has to be placed under the ModbusAsyncDevice component.

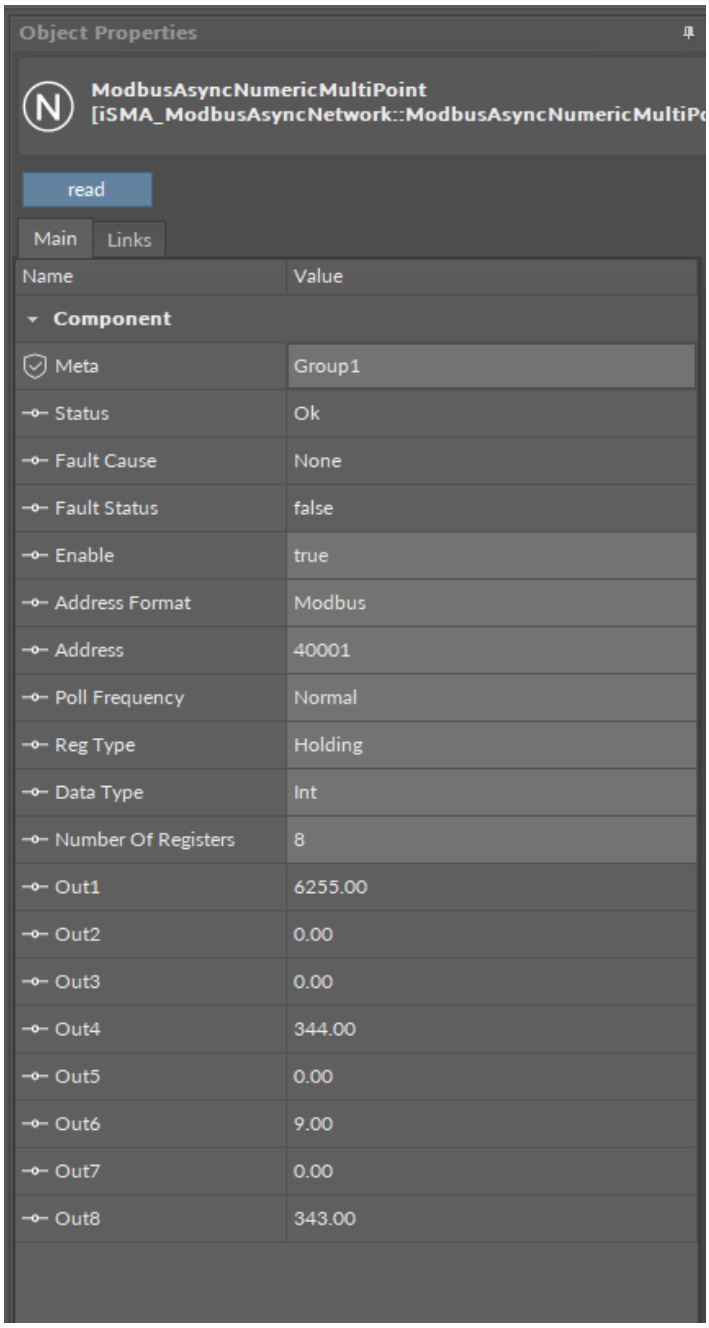


Figure 64. ModbusAsyncNumericMultiPoint

The ModbusAsyncNumericMultiPoint component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** informs about the point's error status (true: point read error);
- **Enable:** enables or disables the point (true: point enabled, false: point disabled);
- **Address Format:** allows to set the register address format (Modbus, decimal);
- **Address:** allows to set the register address;
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Reg Type:** allows to set the type of reading the register (input, holding);
- **Data Type:** allows to set the reading data type: Int (unsigned values), Sint (signed values);
- **Number Of Registers:** allows to set the number of registers read in one message;
- **Out1-Out8:** shows the current values of read registers.

The ModbusAsyncNumericMultiPoint component offers the following action, available in the context menu:

- **Read:** enforces reading of the point.

9.2.8 ModbusAsyncRegisterBitWritable

The ModbusAsyncRegisterBitWritable component is responsible for sending and reading Boolean values from the device. The component has to be placed under the ModbusAsyncDevice component.

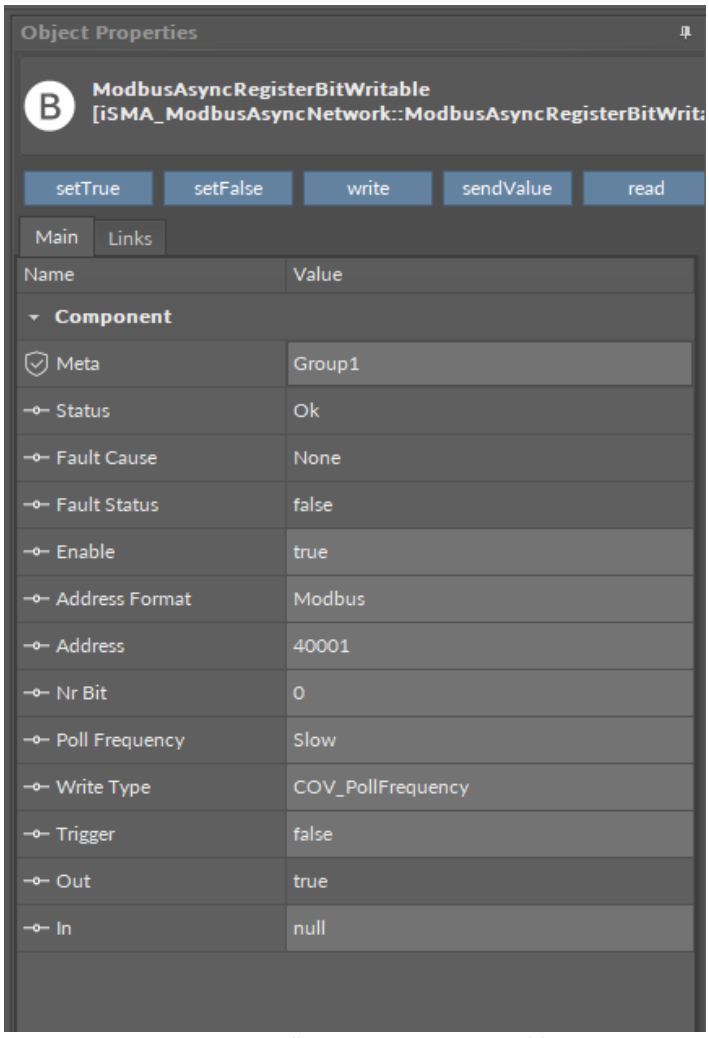


Figure 65. ModbusAsyncRegisterBitWritable

The ModbusAsyncRegisterBitWritable component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Fault Status:** informs about the point error status (true: point read error);
- **Enable:** enables or disables the point (true: point enabled, false: point disabled);
- **Address Format:** allows to set the register address format (Modbus, decimal);
- **Address:** allows to set the register address;
- **Nr Bit:** allows to set the bit number in the register;
- **Poll Frequency:** allows to set the reading poll frequency (fast, normal, slow);
- **Write Type:** allows to set the writing mode (COV: only on the In slot change, COV_PollFrequency: on the In slot change and periodically, PollFrequency: only

periodically, COV_LinkSet : only on the In slot change using the "reverse following the link" function);
- Trigger: allows to force sending values on rising edge;
- Out: the current value of reading bit;
- In: the input slot.

Actions

The ModbusAsyncRegisterBitWritable component offers the following actions, available in the context menu:

- Set True/Set False: writes the value to the In slot and sends it to the device (not active if the In slot has a link connected);
- Write: sends the value from the In slot to the device;
- Read: reads the value from the device and sends it to the Out slot;
- Send Value: sends the value to the device, without changing the value on the In slot.

10 iSMA FCU Kit

The iSMA FCU kit includes dedicated components, which can be used in typical FCU application.

10.1 FCU_PI

The FCU_PI component is a regulator with proportional and integral actions.

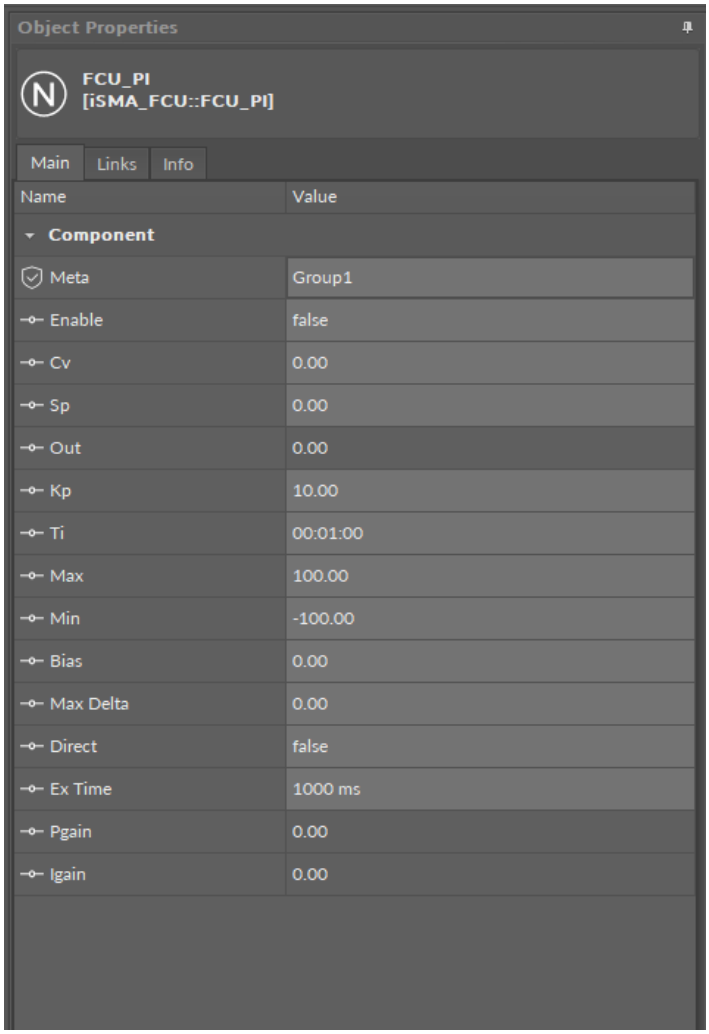


Figure 66. The FCU_PI component

The FCU_PI component has the following slots:

- **Enable:** enables or disables the component–if the component is disabled (the Enable slot is set to false), the Out slot is set to 0;
- **Cv:** the numeric input slot with the controlled value;
- **Sp:** allows to set the setpoint for controlled value;
- **Out:** the output slot of the component;
- **Kp:** allows to set the value of proportional gain constant;
- **Ti:** allows to set the value of the integral time constant (setting the slot to 0, disables this integral action);
- **Max:** allows to set the maximum value of the output of component;
- **Min:** allows to set the minimal value of the output of component;
- **Bias:** allows to set the bias value–this value is added to the Out slot if Ti slot is set to 0;

- **Max Delta:** allows to set the maximum amount the Out slot can change by in the exTime (setting to 0 disables this function);
- **Direct:** allows to set the acting process;
    - Available settings: true (direct-acting process), false (reverse the acting process);
- **Ex Time:** allows to set the period of loop execution;
- **Pgain:** shows the value of the output signal, which was calculated by the proportional action;
- **Igain:** shows the value of the output signal, which was calculated by the integral action.

10.2 FCU_OutputsSwitch

The FCU_OutputsSwitch component allows to manage outputs for the temperature and fan control, according to the FCU configuration (2 or 4 pipe system, analog or binary temperature control, etc.).

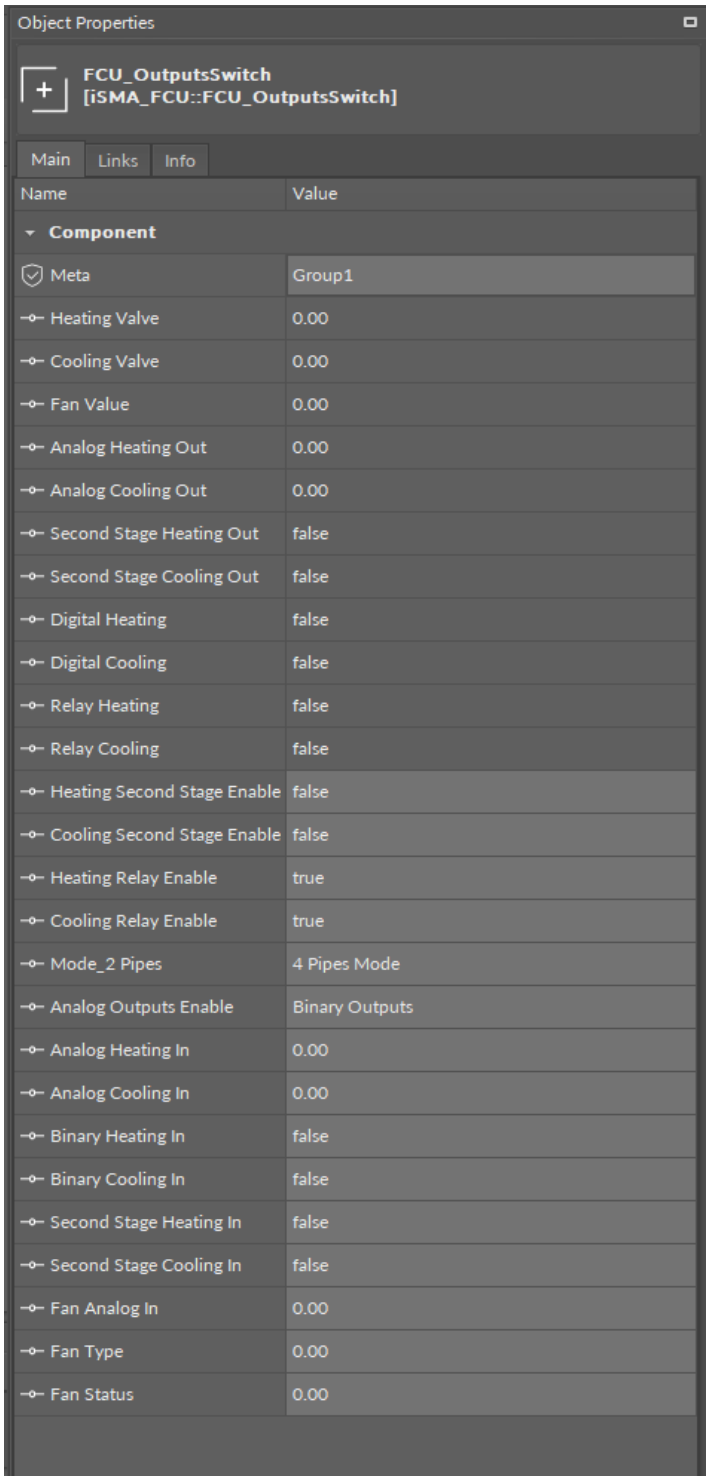


Figure 67. The FCU_OutputsSwitch component

The FCU_OutputsSwitch component has the following slots:

- **Heating Valve:** displays the status of the heating valve; the status is displayed differently depending on the Analog Outputs Enable setting (binary or analog):
  - Available information for the temperature binary output: 0 (closed) or 1 (open);
  - Available information for the temperature analog output: displays the value of the Analog Heating In slot;
- **Cooling Valve:** displays the status of the the cooling valve; the status is displayed differently depending on the Analog Outputs Enable setting (binary or analog):
  - Available information for the temperature binary output: 0 (closed) or 1 (open);

    - Available information for the temperature analog output: displays the value of the Analog Cooling In slot;
- **Fan Value:** displays the status of the fan; the status is displayed differently depending on the Fan Type setting (binary or analog):
    - Available information for the fan binary outputs: 0, 1, 2, or 3;
    - Available information for the fan analog outputs: displays the value of the Fan Analog In slot;
- **Analog Heating Out:** the output slot for the analog heating valve;

> **Analog Heating Out Values**
>
> For the 2-pipe system (the **Mode _2 Pipes** slot is set to true), if the **Analog Outputs Enable** slot is set to true, the **Analog Heating Out** slot displays the value of the **Analog Heating In** slot or **Analog Cooling In** slot, depending on which slot has the value greater than 0.
>
> For the 4-pipe system (the **Mode _2 Pipes** slot is set to false), the **Analog Heating Out** slot can only display the value of the **Analog Heating In** slot.
>
> If the **Analog Outputs Enable** slot is set to false (the component uses only the binary outputs), the **Analog Heating Out** slot is set to 0.

- Analog Cooling Out: the output slot for the analog cooling valve;

> **Analog Cooling Out Values**
>
> For the 4-pipe system (the **Mode _2 Pipes** slot is set to false), the **Analog Cooling Out** slot displays the value of the **Analog Cooling In** slot.
>
> If the **Analog Outputs Enable** slot is set to false (the component uses only the binary outputs), or the **Mode _2 Pipes** slot is set to false (for the 4-pipe system), the **Analog Cooling Out** slot is set to 0.

- **Second Stage Heating Out:** the output slot for the second stage heating–the slot displays the value from the Second Stage Heating In slot;

Note: If the Heating Second Stage Enable slot is set to false, or the Heating Relay Enable slot is set to false, the Second Stage Heating Out slot cannot be set to true.

- **Second Stage Cooling Out:** the output slot for the second stage cooling–the slot displays the value from the Second Stage Cooling In slot;

Note: If the Cooling Second Stage Enable slot is set to false, or the Cooling Relay Enable slot is set to false, the Second Stage Cooling Out slot cannot be set to true.

- **Digital Heating:** the output slot for the digital heating (recommended to service the heating valve switched on/off by triacs);

Digital Heating Values

For the 2-pipe system (the Mode _2 Pipes slot is set to true), if the Analog Outputs Enable slot is set to false, the Digital Heating slot displays the value of the Binary Heating In slot or the Binary Cooling In slot, depending on which slot has the true value.

For the 4-pipe system (the Mode _2 Pipes slot is set to false), the Digital Heating slot can only display the value of Binary Heating In slot.

If the Analog Outputs Enable slot is set to true (the component uses only the analog outputs), the Digital Heating slot is set to false.

- Digital Cooling: the output slot for the digital cooling (recommended to service the cooling valve switched on/off by triacs);

Digital Cooling Values

For the 4-pipe system (the Mode _2 Pipes slot is set to false), the Digital Heating slot displays the value of Binary Cooling In slot.

If the Analog Outputs Enable slot is set to true (the component uses only the analog outputs), or the Mode _2 Pipes slot is set to false (for the 4-pipe system), the Digital Cooling slot is set to false.

- Relay Heating: the output slot for digital heating in the first or second stage (recommended to service the heating valve switched by the relay output or electrical heaters);

Relay Heating Values

If the Heating Relay Enable slot is set to true, and the Heating Second Stage Enable slot is set to false (heating in the first stage only), the value from the Binary Heating In slot is set to the Relay Heating slot.

If the Heating Relay Enable slot is set to true, and the Heating Second Stage Enable slot is set to true (heating in the first and second stage), the value from the Second Stage Heating In slot is set to the Relay Heating slot.

If the Heating Relay Enable slot is set to false (the heating relay is disabled), the Relay Heating slot is set to false.

- Relay Cooling: the output slot for the digital cooling in the first or second stage (recommended to service the cooling valve switched by the relay output or electrical coolers);

Relay Cooling Values

If the Cooling Relay Enable slot is set to true, and the Cooling Second Stage Enable slot is set to false (cooling in the first stage only), the value from the Binary Cooling In slot is set to the Relay Cooling slot.

If the Cooling Relay Enable slot is set to true, and the Cooling Second Stage Enable slot is set to true (cooling in the first and second stage), the value from the Second Stage Cooling In slot is set to the Relay Cooling slot.

If the Cooling Relay Enable slot is set to false (the cooling relay is disabled), the Relay Cooling slot is set to false.

- Heating Second Stage Enable: allows to enable or disable the second stage heating;
  - Available settings: true (enabled,) false (disabled);
- Cooling Second Stage Enable: allows to enable or disable the second stage cooling;
  - Available settings: true (enabled,) false (disabled);
- Heating Relay Enable: allows to enable or disable the relay for heating;
  - Available settings: true (enabled,) false (disabled);
- Cooling Relay Enable: allows to enable or disable the relay for cooling;
  - Available settings: true (enabled,) false (disabled);
- Mode_2 Pipes: allows to switch between the 2-pipe system and 4-pipe system;
  - Available settings: 2-pipe system, 4-pipe system;
- Analog Outputs Enable: allows to switch between the analog or binary control of the temperature outputs;
  - Available settings: Analog Outputs, Binary Outputs;
- Analog Heating In: sets the analog value for heating;
- Analog Cooling In: sets the analog value for cooling;
- Binary Heating In: sets the binary value for the first stage heating;
- Binary Cooling In: sets the binary value for the first stage cooling;
- Second Stage Heating In: sets the binary value for the second stage heating;
- Second Stage Cooling In: sets the binary value for the second stage cooling;
- Fan Analog In: sets the analog value for the fan;
- Fan Type: sets the type of fan;
  - Available settings: 0 (fan with analog output), other values (fan with binary outputs);
- Fan Status: sets the current fan speed;
  - Available settings: 0 (Off), 1 or 4 (Speed 1), 2 or 5 (Speed 2), 3 or 6 (Speed 3).

10.3 FCU_MasterSlave

The FCU_MasterSlave component allows to automatically calculate the BACnet device ID of slave devices in the BACnet master-slave network, depending on the BACnet device ID of master devices. This function is called Auto Binding. The table below shows values of the master BACnet device ID and the corresponding BACnet device IDs of slave devices for the Auto Binding function:

Master Id	Slave 1 ID	Slave 2 ID	Slave 3 ID	Slave 4 ID	Slave 5 ID
826101	826001	826002	826003	826004	826005

Master Id	Slave 1 ID	Slave 2 ID	Slave 3 ID	Slave 4 ID	Slave 5 ID
826102	826006	826007	826008	826009	826010
826103	826011	826012	826013	826014	826015
826104	826016	826017	826018	826019	826020
826105	826021	826022	826023	826024	826025
826106	826026	826027	826028	826029	826030
826107	826031	826032	826033	826034	826035
826108	826036	826037	826038	826039	826040
826109	826041	826042	826043	826044	826045
826110	826046	826047	826048	826049	826050
826111	826051	826052	826053	826054	826055
826112	826056	826057	826058	826059	826060
826113	826061	826062	826063	826064	826065
826114	826066	826067	826068	826069	826070
826115	826071	826072	826073	826074	826075
826116	826076	826077	826078	826079	826080
826117	826081	826082	826083	826084	826085
826118	826086	826087	826088	826089	826090
826119	826091	826092	826093	826094	826095
826120	826096	826097	826098	826099	826100
Other	0	0	0	0	0

Table 4. The MasterSlave Id

The Auto Binding function can be disabled (by setting the Local Remote Auto Binding slot to true). In this case, the IDs of slave devices have to be set by the user (in the Remote Slave 1 Device Id-Remote Slave 5 Device Id slots).

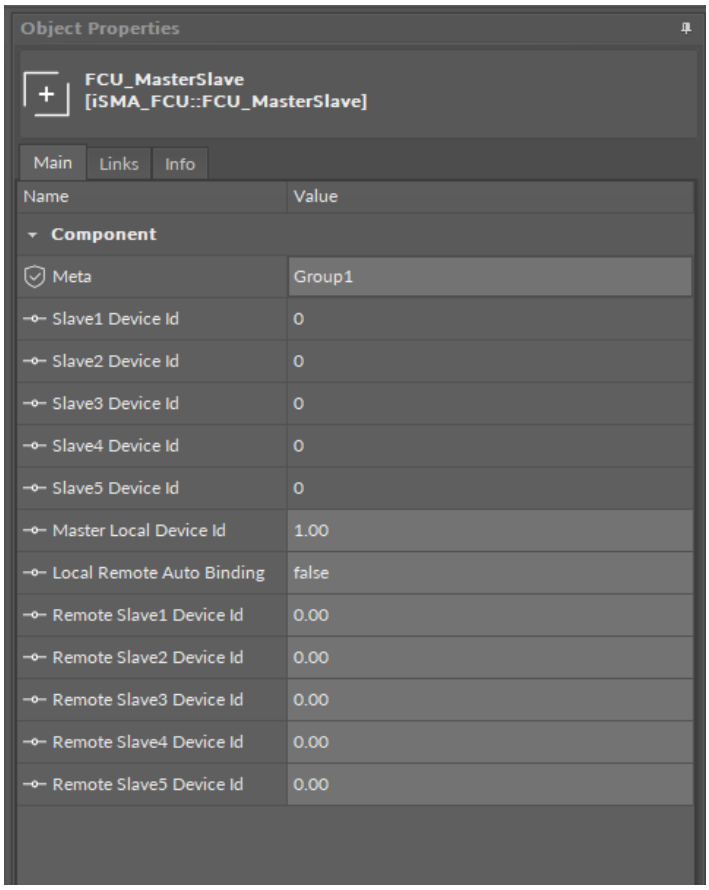


Figure 68. The FCU_MasterSlave component

The FCU_MasterSlave component has the following slots:

- **Slave1 Device Id-Slave5 Device Id:** display the IDs calculated or set for five slave devices;
- **Master Local Device Id:** sets the ID of the master device.

Note: If the component uses the Auto Binding function, the value of the Master Local Device Id slot has to be set to the value ranged from 826101 to 826120. For other values, all output slots (Slave1 Device Id-Slave5 Device Id) will be set to 0.

- **Local Remote Auto Binding:** allows to switch between Auto Binding and Remote Binding functions;
  - Available settings: true (Remote Binding–IDs of each slave device are set to the corresponding values of the Remote Slave1 Device Id-Remote Slave5 Device Id slots), false (Auto Binding–IDs of each slave device are calculated according to the table above);
- **Remote Slave1 Device Id-Remote Slave5 Device Id:** allows to set remote IDs of slave devices. If the Local Remote Auto Binding slot is set to true, these values are set to corresponding outputs (slots Slave1 Device Id-Slave5 Device Id).

10.4 FCU_TemperatureBinary

The FCU_TemperatureBinary component allows calculating binary values for the Heating Binary Output and Cooling Binary Output slots, according to the difference between the Setpoint and Cv slots.

The FCU_TemperatureBinary component can work in two temperature control modes:

- One-stage mode: only the Heating Binary Output and Cooling Binary Output slots are switched on/off,
- Two-stage mode: the Heating Binary Output and Cooling Binary Output slots are used for the first stage, and dedicated binary outputs (the Heating Second Stage Binary Output and Cooling Second Stage Binary Output slots) are used for the second stage.

Note: Operation in the two-stage mode can be selected by setting the Heating Second Stage Enable and/or Cooling Second Stage Enable slots to true. If one of these slots is set to false, the component will operate in the one-stage mode for corresponding temperature mode (Heating or Cooling).

For proper operation, the component has to be enabled (the Temperature Binary Enable slot set to true), and the Fan Active slot has to be set to true. If the second condition is not met, the component is enabled, but main outputs are blocked–values of the Heating Binary Output, Cooling Binary Output, Heating Second Stage Binary Output, and Cooling Second Stage Binary Output slots are set to false.

The conditions for switching on/off heating binary output slots for one-stage mode (the Heating Binary Output and Cooling Binary Output slots) are presented in the figures below:

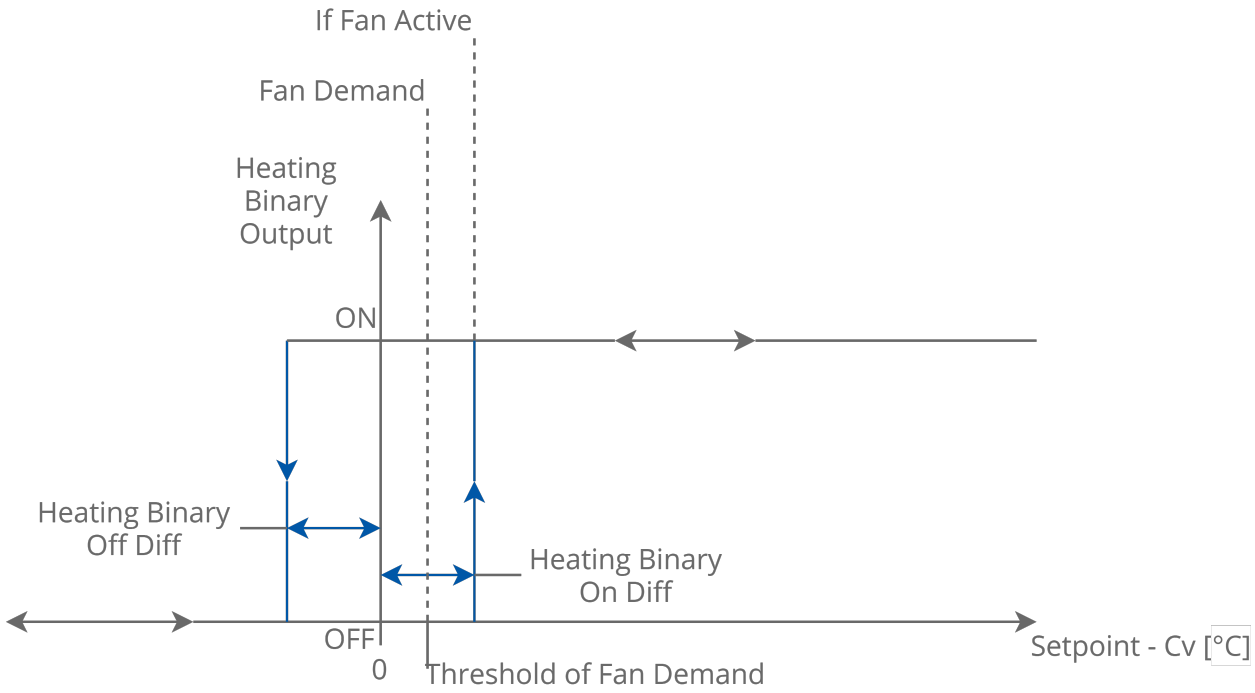


Figure 69. Binary control of temperature for 1st stage only - heating mode

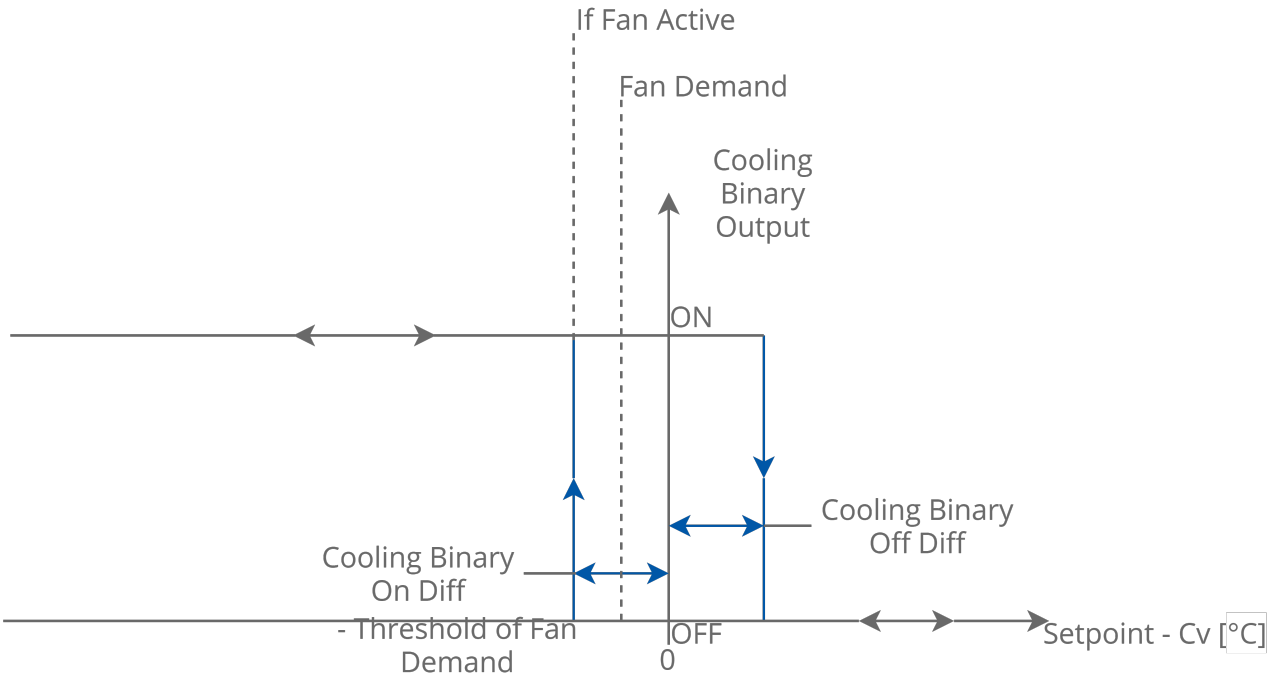


Figure 70. Binary control of temperature for 1st stage only - cooling mode

The values of the Heating Binary Output and Cooling Binary Output slots for the first stage, and the values of the Heating Second Stage Binary Output and Cooling Second Stage Binary Output slots (for the second stage) for the two-stage mode are calculated as shown in the figures below:

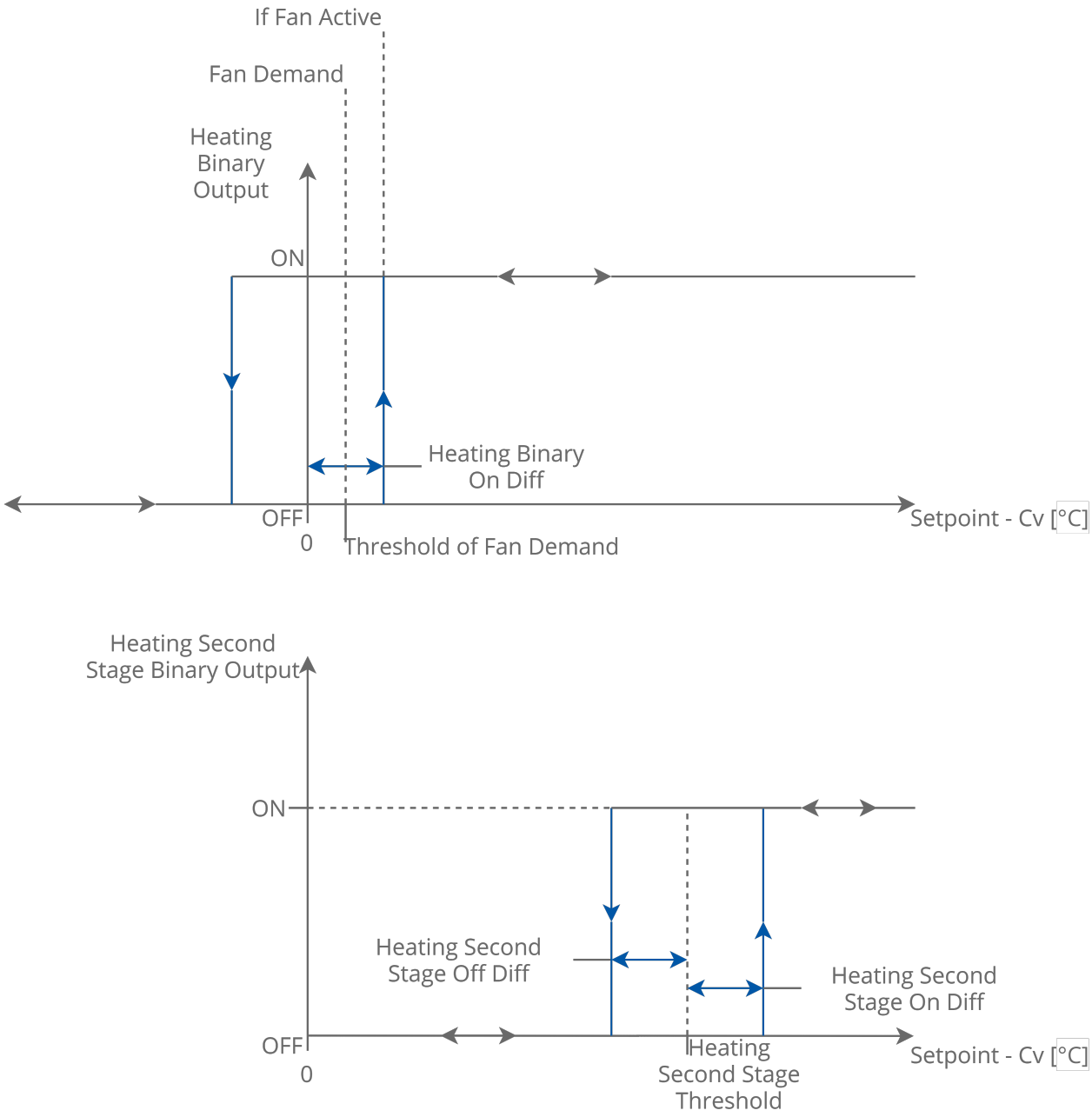


Figure 71. Binary control of temperature for 1st and 2nd stage - heating mode

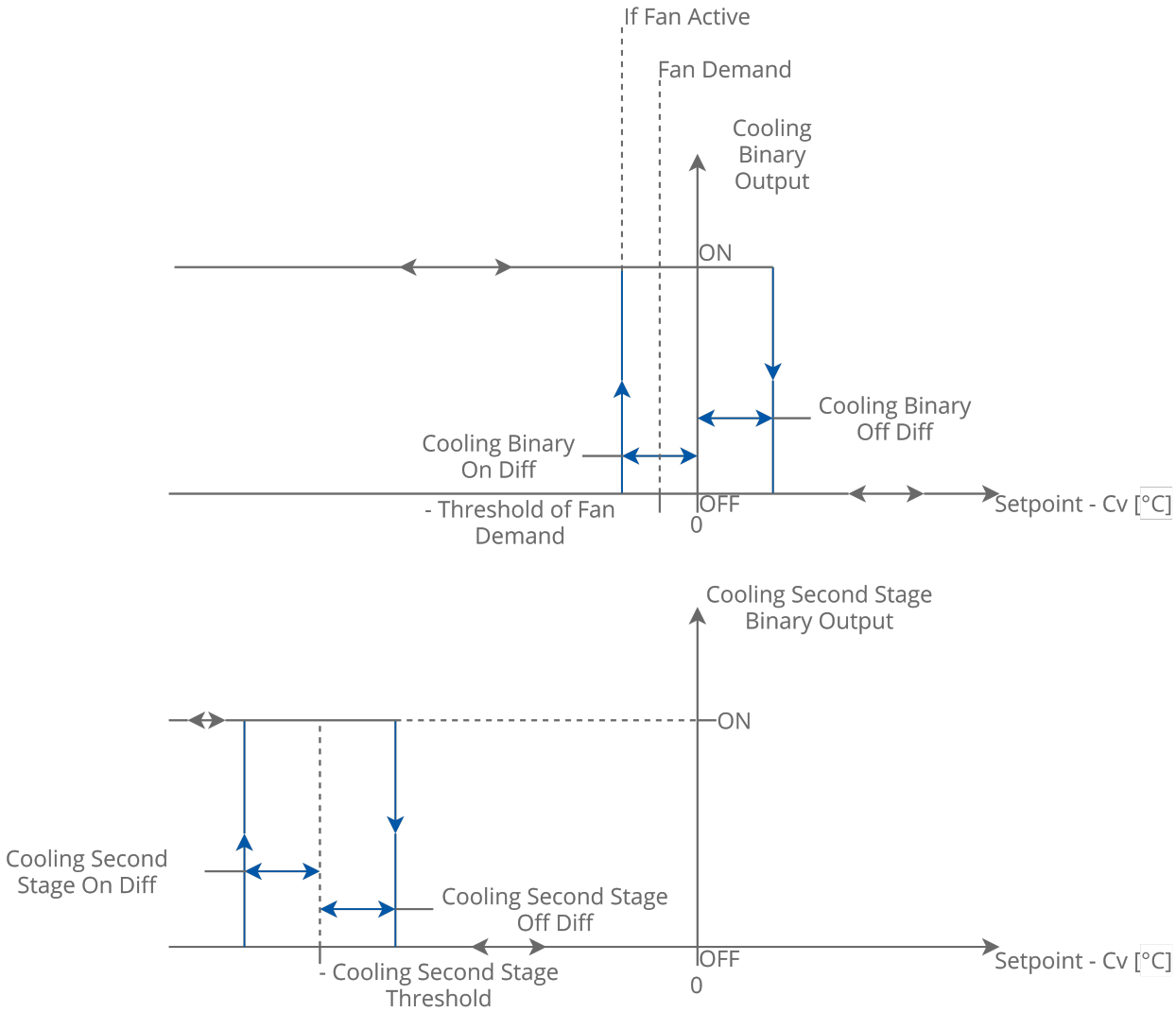


Figure 72. Binary control of temperature for 1st and 2nd stage - cooling mode

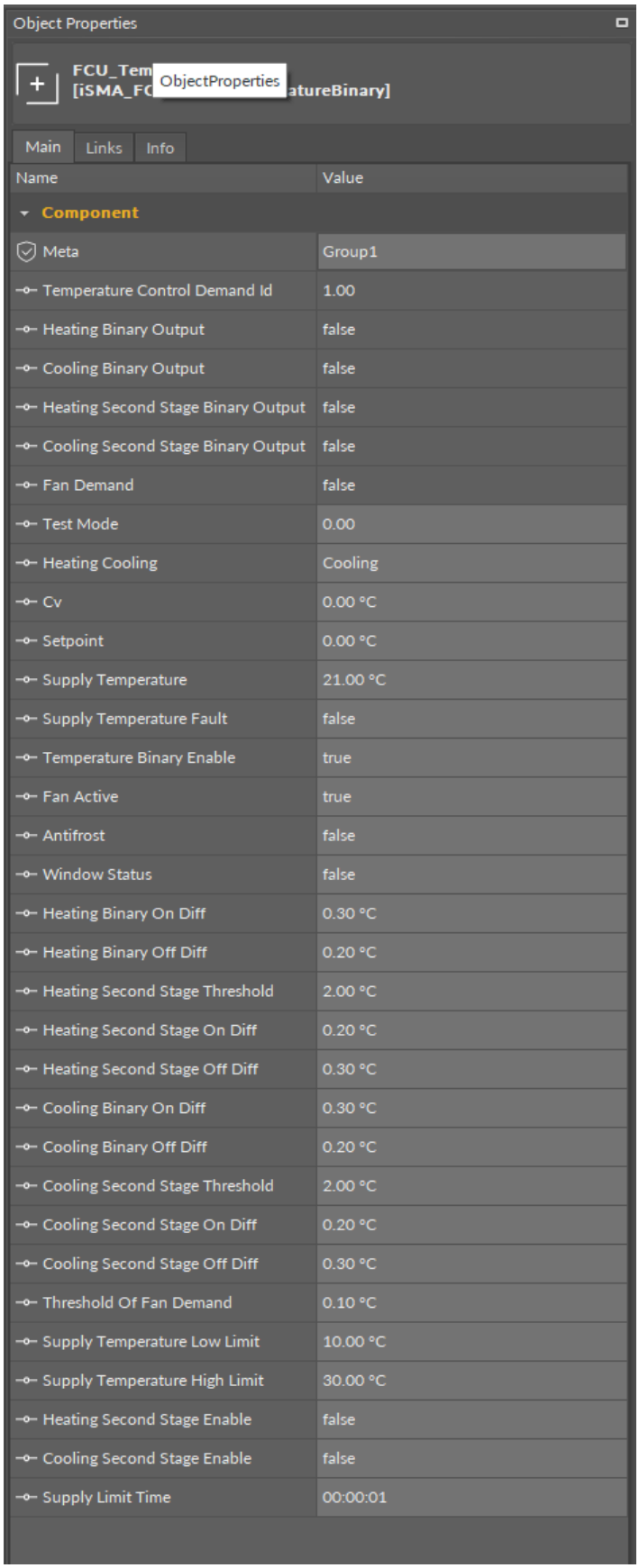


Figure 73. FCU_TemperatureBinary component

The FCU_TemperatureBinary component has the following slots:

- **Temperature Control Demand:** shows the current component temperature demand;

    - Available information: 1 (HeatingDemand), 2 (CoolingDemand);
- **Heating Binary Output:** shows the state of the heating demand;
- **Cooling Binary Output:** shows the state of the cooling demand;
- **Heating Second Stage Binary Output:** shows the state of heating in the second stage;
- **Cooling Second Stage Binary Output:** shows the state of cooling in the second stage;
- **Fan Demand:** shows the fan demand.;
    - Available information: true–if the absolute value from the Control Value slot exceeds the value set in the Fan Demand Threshold slot, the Fan Demand slot is set to true, false–in other cases;
- **Test Mode:** allows to set one of the predefined test modes;
    - Available settings: 0 (the component uses its main algorithm), 1 (full heating), 2 (full cooling);

Full Heating Test Mode

In the Full Heating mode, the **Fan Demand** slot is set to true. The **Heating Binary Output** slot is set to true, and the **Heating Second Stage Binary Output** slot is set to true (only if the **Heating Second Stage Enable** slot is set to true).

Full Cooling Test Mode

In the Full Cooling mode, the **Fan Demand** slot is set to true. The **Cooling Binary Output** slot is set to true, and the **Cooling Second Stage Binary Output** slot is set to true (only if the **Cooling Second Stage Enable** slot is set to true).
Note: Before starting the Full Heating or Full Cooling mode, the value of the **Fan Active** slot has to be set to true.

- **Heating Cooling:** sets the current temperature mode;
    - Available settings: heating, cooling;

Note: If the component operates in one of the above modes, outputs corresponding to the other mode are blocked. For example, if the component works in the Heating mode, the Cooling Binary Output and the Cooling Second Stage Binary Output slot are set to false.

- **Cv:** the current value of controlled temperature;
- **Setpoint:** sets the setpoint for the controlled temperature;
- **Supply Temperature:** allows to read the value of the supply temperature;
- **Supply Temperature Fault:** allows to read the information about the fault of the supply temperature;
- **Temperature Binary Enable:** allows to enable or disable the TemperatureBinary component–if the component is disabled, all outputs are set to false;
    - Available settings: true (enabled), false (disabled);
- **Fan Active:** informs the TemperatureBinary component that the fan is switched on; if fan is switched off, outputs for heating and cooling valves actuators (the Heating Binary Output and Cooling Binary Output slots) are set to false, and the binary output slots for heating and cooling in the second stage (the Heating Second Stage Binary Output and Cooling Second Stage Binary Output slot) are set to false–these slots can be set to true only when the Fan Active slot is set to true;
- **Antifrost:** allows to switch on the Antifrost mode;
    - Available settings: true (enabled), false (disabled);

Note: In the Antifrost mode enabled, the Fan Demand slot is set to true, the Heating Binary Output is set to true, and the Heating Second Stage Binary Output is set to true (only if the Heating Second Stage Enable slot is set to true).

Note: The Antifrost mode has higher priority than the main algorithm, but it can be overridden by the Test mode.

- Window Status: allows to switch on the Window Open mode;
  - Available settings: true (Window Open mode enabled), false (Window Open mode disabled–the component operates in saving energy mode, all outputs are set to false;

Note: The Window Open mode can be overridden only by the Antifrost mode or the Test mode.

- Heating Binary On Diff: sets the temperature (the difference between Setpoint and Cv) above which the Heating Binary Output is switched on;
- Heating Binary Off Diff: sets the temperature (the difference between Setpoint and Cv) above which the Heating Binary Output is switched off;
- Heating Second Stage Threshold: sets the threshold of the temperature (the difference between Setpoint and Cv) above which (with the hysteresis) the heating in the second stage is switched on;
- Heating Second Stage On Diff: sets the differential for hysteresis of switching on the heating in the second stage;
- Heating Second Stage Off Diff: sets the differential for hysteresis of switching off the heating in the second stage;
- Cooling Binary On Diff: sets the temperature (the difference between Setpoint and Cv) above which the Cooling Binary Output is switched on;
- Cooling Binary Off Diff: sets the temperature (the difference between Setpoint and Cv) above which the Cooling Binary Output is switched off;
- Cooling Second Stage Threshold: sets the threshold of the temperature (the difference between Setpoint and Cv) above which (with the hysteresis) the cooling in the second stage is switched on;
- Cooling Second Stage On Diff: sets the differential for hysteresis of switching on the cooling in the second stage;
- Cooling Second Stage Off Diff: sets the differential for hysteresis of switching off the cooling in the second stage;
- Threshold Of Fan Demand: sets the threshold of the temperature (the difference between Setpoint and Cv) above which the Fan Demand slot is set to true;
- Supply Temperature Low Limit: sets the minimum acceptable value of the supply temperature–this value is used in the Supply Air Temperature Limitation function;
- Supply Temperature High Limit: sets the maximum acceptable value of the supply temperature–this value is used in the Supply Air Temperature Limitation function,
- Heating Second Stage Enable: allows to enable or disable the heating in the second stage;
  - Available settings: true (enabled), false (disabled);
- Cooling Second Stage Enable: allows to enable or disable the cooling in the stage;
  - Available settings: true (enabled), false (disabled);
- Supply Limit Time: sets the delay time for the activation of the Supply Air Temperature Limitation function.

Supply Air Temperature Limitation

In order to maintain room conditions comfortable for the user, the supply air can have a temperature limitation. This function is available only if the supply air sensor is connected and works correctly. The supply air temperature can have a high limit defined by the Supply Temperature High Limit slot, and a low limit defined by the Supply Temperature Low Limit slot. The range between the Supply Temperature Low Limit and Supply Temperature High Limit values is called a comfort range.

- Supply Air Temperature limitation in the first stage binary control

In binary control, if the supply air temperature approaches the comfort range by 1°C, the FCU_TemperatureBinary component starts the countdown set in the Supply Limit Time slot delay time . After this time, if the supply air temperature value is still out of the comfort range, the component disables the heating (if the temperature value is above the Supply Temperature High Limit) or cooling (if temperature value is above the Supply Temperature Low Limit). If the supply air temperature value returns to the comfort range, the component resets the delay counter and returns to normal operation.

- Supply Air Temperature limitation in the second stage binary control

If the Supply Air Temperature value is out of the comfort range, the component disables the second stage, and starts counting the delay time set in the Supply Limit Time slot. After this time, if the supply air temperature value is still out of the comfort range, the FCU_TemperatureBinary component disables the heating (if the temperature value is above the Supply Temperature High Limit) or cooling (if the temperature value is above the Supply Temperature Low Limit). If the supply air temperature value returns to the comfort range, the component resets the delay counter, enables the second stage, and returns to normal operation.

10.5 FCU_TemperatureSensorsSwitch

The FCU_Temperature_Sensors_Switch component allows for switching temperature sensors, according to the selected source.

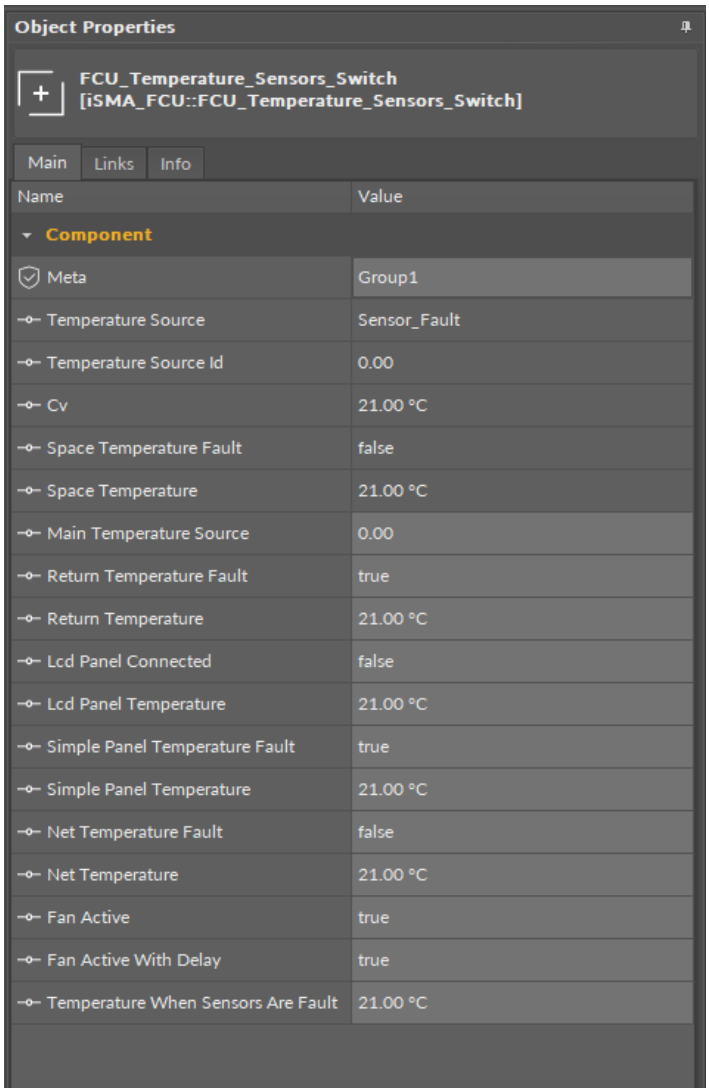


Figure 74. FCU_TemperatureSensorsSwitch component

The FCU_Temperature_Sensors_Switch component has the following slots:

- **Temperature Source:** informs about the source of the temperature set to the Cv slot;
  - Available information: (0) Sensor Fault (fault status of the sensor selected by the Main Temperature Source slot), (1) LCD Panel, (2) Simple Panel, (3) Return temperature, (4) Net temperature;
- **Temperature Source Id:** the numeric output with a value corresponding to the Temperature Source slot;
- **Cv:** the current input value (measured temperature), switched according to the value of the Main Temperature slot;

The Cv slot dependencies:

If the sensor selected in the **Main Temperature Source** slot has not gone into any fault or false state (slots corresponding to such fault–**Return Temperature Fault**, **Lcd Panel Connected**, **Simple Panel Temperature Fault**, **Net Temperature Fault**–are set to false), the temperature from the source selected in the **Main Temperature Source** is set to the **Cv** slot. If the **Main Temperature Source** slot is set to LCD Panel, and the panel is connected (the **Lcd Panel Connected** slot is set to true), the value from the

Lcd Panel Temperature slot is set to the Cv slot. If the LCD panel is not connected (the Lcd Panel Connected slot is set to false), a value from the Temperature When Sensors Are Fault slot is set to Cv slot.
If the sensor selected in the Main Temperature Source slot has gone into any fault or false state , the temperature from the Temperature When Sensors Are Fault slot is set to the Cv slot.
If the value from the Return Temperature slot is set to the Cv slot, it is possible to switch the temperature from Return to Space. The Space Temperature will be downloaded when the fan is off. There could also be a delay for switching to Return Temperature after the fan starts to blow the duct. The Fan Active and Fan Active With Delay slots are used for this purpose. If one or both slots have false states, then the value from the Space Temperature slot is set to the Cv slot (instead of the value from the Return Temperature slot). If both slots have the same true states, then the value from the Return Temperature slot is set to the Cv slot.
Note: For the proper operation of this Return to Space switching function, an external component is required, which delays the value informing about the fan status. The value without delay has to be connected to the Fan Active slot, and the delayed value has to be connected to the Fan Active With Delay slot.
Note: If the Return Temperature Fault slot is set to true (sensor has gone into fault), it transfers the value from the Temperature When Sensors Are Fault slot (instead of the Return Temperature) to the Cv slot. If the Space Temperature Fault slot is set to true (sensor has gone into fault), the function is inactive.

- Space Temperature Fault: indicates the status of the space temperature;
  - Available information: true (fault), false (no fault).
- Space Temperature: shows the output value of the space temperature;

The Space Temperature slot dependencies:

The Lcd Panel Temperature has the highest priority; if the panel is connected (the Lcd Panel Connected slot is set to true), the value from the Lcd Panel Temperature slot is transferred to the Space Temperature slot.
If the LCD panel is disconnected (the Lcd Panel Connected slot is set to false), and there is no fault of the simple panel temperature sensor (the Simple Panel Temperature Fault is set to false), the value from the Simple Panel Temperature slot is transferred to the Space Temperature slot.
If the LCD panel is disconnected, the simple panel sensor is in fault, and there is no fault of the Net Temperature, the value from the Net Temperature slot is transferred to the Space Temperature slot.
In case none of the conditions mentioned above is met, and there is no fault of the temperature sensor which is used to calculate the Cv value, the value from the Cv slot is set to the Space Temperature slot.
Note: In all cases described above, the Space Temperature Fault slot is set to false (there is no fault). In any other case, this slot is set to true.

- Main Temperature Source: allows to select the source of temperature; available values:
  - Available settings: 0 (LCD panel), 1 (Simple Panel), 2 (return temperature), 3 (net temperature).

- **Return Temperature Fault:** allows to read the status of the return temperature sensor;
  - Available settings: true (sensor fault), false (sensor's operation is correct);
- **Return Temperature:** allows to read the temperature from the return temperature sensor;
- **Lcd Panel Connected:** allows to read the status of the LCD panel;
  - Available settings: true (panel connected), false (panel disconnected);
- **Lcd Panel Temperature:** allows to read the temperature from the LCD panel temperature sensor;
- **Simple Panel Temperature Fault:** allows to read the status of the Simple Panel temperature sensor;
  - Available settings: true (sensor fault), false (sensor's operation is correct);
- **Simple Panel Temperature:** allows to read the temperature from the Simple Panel temperature sensor;
- **Net Temperature Fault:** allows to read the status of the net temperature sensor;
  - Available settings: true (sensor fault), false (sensor's operation is correct);
- **Net Temperature:** allows to read the net temperature value;
- **Fan Active:** allows to read the status of the fan;
  - Available settings: true (fan switched on), false (fan switched off);
- **Fan Active With Delay:** allows to read the status of the fan (with external delay);
  - Available settings: true (fan switched on), false (fan switched off);
- **Temperature When Sensors Are Fault:** the temperature, which is to be set to the Cv slot if the sensor selected in the MainTemperature Source is faulty.

10.6 FCU_EffectiveSetpointCalculator

The FCU_EffectiveSetpointCalculator is a component, which calculates the value of the effective temperature setpoint, according to the setpoint value from the upper-level system (for example, BMS), offset of this value, occupancy mode, and the heating or cooling modes.

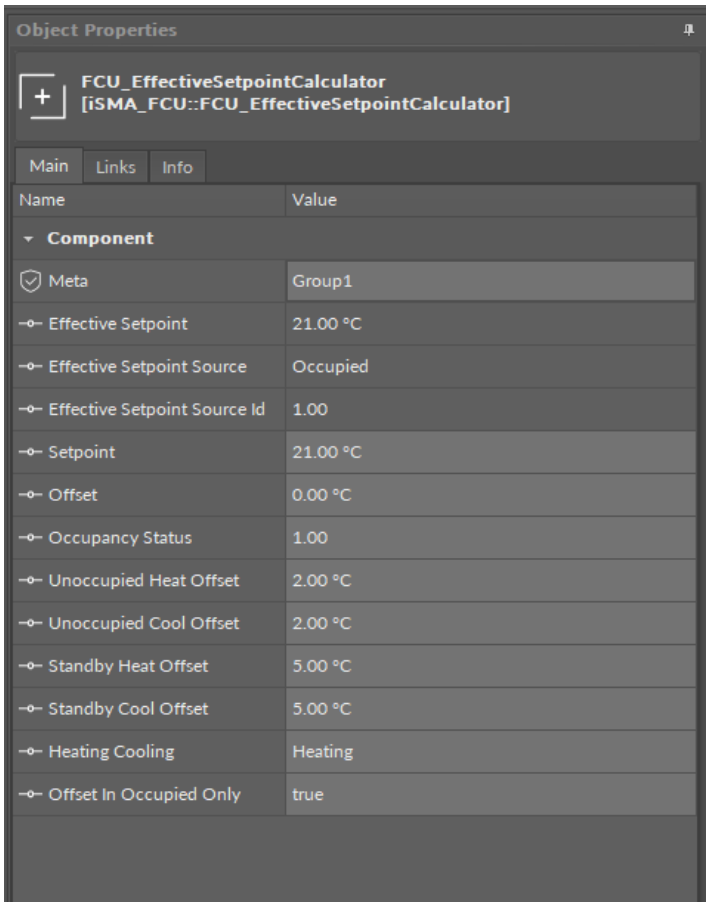


Figure 75. FCU_EffectiveSetpointCalculator component

The FCU_EffectiveSetpointCalculator component has the following slots:

- **Effective Setpoint:** the main output slot of a component, which value is equal to the calculated effective temperature setpoint;
- **Effective Setpoint Source:** displays information about the way the setpoint is actually calculated, depending on the occupancy status and heating or cooling mode setting;
  - Available information: 1 (Occupied), 2 (Unoccupied_Heating), 3 (Unoccupied_Cooling), 4 (Standby_Heating), 5 (Standby_Cooling);
- **Effective Setpoint Source Id:** displays the numeric value, corresponding to the Effective Setpoint Source slot;
- **Setpoint:** the main input of component, which receives the value of the temperature setpoint from the upper-level system;
- **Offset:** the value of the setpoint offset;
- **Occupancy Status:** sets the occupancy status;
  - Available settings: 0 (Unoccupied), 1 (Occupied), 2 (Standby);
- **Unoccupied Heat Offset:** the offset value subtracted from the setpoint in the Unoccupied mode when the algorithm works in the heating mode;
- **Unoccupied Cool Offset:** the offset value added to the setpoint in the Unoccupied mode when the algorithm works in the cooling mode;
- **Standby Heat Offset:** the offset value subtracted from the setpoint in the Standby mode when the algorithm works in the heating mode;
- **Standby Cool Offset:** the offset value added to the setpoint in the Standby mode when the algorithm works in the cooling mode;
- **Heating Cooling:** sets the temperature mode;
  - Available settings: true (heating), false (cooling);

- **Offset In Occupied Only:** determines whether the calculation of the Effective Setpoint value is to be included in the process of calculating the value of the Offset slot if a component is not in the Occupied mode;
    - Available settings: true (means that for the Unoccupied and Standby modes, the value of the Offset slot is not to be included in calculating the Effective Setpoint), false (the value of the Offset slot is to be included in calculating the Effective Setpoint in all occupancy modes).

Methods of calculating the Effective Setpoint value, according to settings of the EffectiveSetpointCalculator component, are presented in the below table:

Occupancy Status	Heating / Cooling	Offset in Occupied Only	Effective Setpoint	Effective Setpoint Source
0 (Unoccupied)	Heating	False	Effective Setpoint = Setpoint + Offset - Unoccupied Heating Offset	2 (Unoccupied Heating)
0 (Unoccupied)	Cooling	False	Effective Setpoint = Setpoint + Offset + Unoccupied Cooling Offset	2 (Unoccupied Cooling)
0 (Unoccupied)	Heating	True	Effective Setpoint = Setpoint - Unoccupied Heating Offset	2 (Unoccupied Heating)
0 (Unoccupied)	Cooling	True	Effective Setpoint = Setpoint + Unoccupied Cooling Offset	2 (Unoccupied Cooling)
1 (Occupied)	-	-	Effective Setpoint = Setpoint + Offset	0 (Occupied)
2 (Standby)	Heating	False	Effective Setpoint = Setpoint + Offset - Standby Heating Offset	2 (Standby Heating)
2 (Standby)	Cooling	False	Effective Setpoint = Setpoint + Offset + Standby Cooling Offset	2 (Standby Cooling)
2 (Standby)	Heating	True	Effective Setpoint = Setpoint - Standby Heating Offset	2 (Standby Heating)
2 (Standby)	Cooling	True	Effective Setpoint = Setpoint + Standby Cooling Offset	2 (Standby Cooling)

Table 5. The effective setpoint calculation

10.7 FCU_ChangeOfStateDelay

The FCU_ChangeOfStateDelay component allows for delaying the binary value for the time defined in the Delay Time slot. The component is an extended version of the On Delay and Off Delay–both functions are used in one component, and there is the

possibility to set the status defined by the user to the Output slot (during and after delay time).

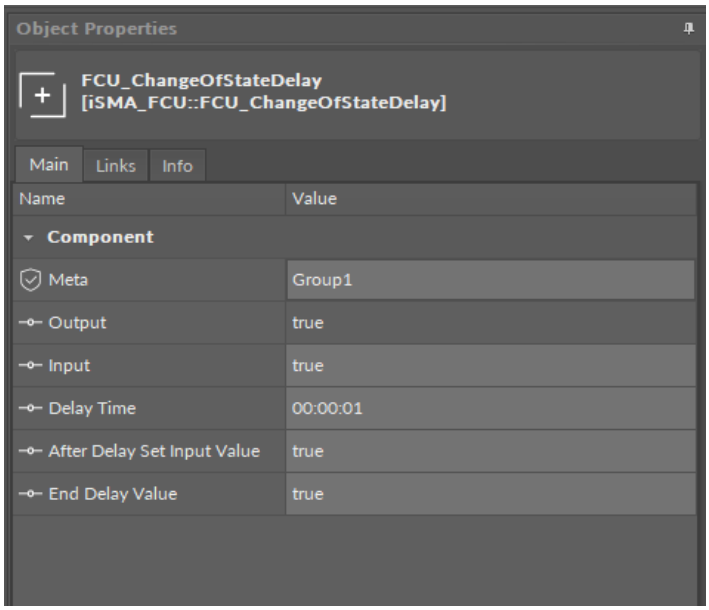


Figure 76. FCU_ChangeOfStateDelay component

The FCU_ChangeOfStateDelay component has the following slots:

- **Output:** the binary output slot;
- **Input:** the binary input slot– the component starts counting delay time on the change of value (disregarding rising or falling edge);
- **Delay Time:** allows to set the delay time in seconds;
- **After Delay Set Input Value:** allows to set the component's mode;
    - Available settings: true, false;

True

If the state of Input slot has changed, and counting of delay time has been initiated, the Output slot changes to the state from the Input slot. If counting is in progress, the previous state is set to the Output slot. For example, if the Input slot is in true state, and it changes to false, it means that the Output slot is still set to true (counting is in progress). If the delay time ends, the Output slot sets to false.

False

If the state of the Input slot has changed, and counting of delay time has been initiated, the Output slot changes to the state from the End Delay Value slot. During the counting process, the Output slot is set in opposition to the state of the End Value Slot.

- **End Delay Value:** allows to set the state, which is set to the Output slot, after the counting process has ended. During the counting process, the Output slot is set in opposition to the state of the End Value Slot. This option is available only if the After Delay Set Input Value slot is false.

10.8 FCU_ValueHolder

The FCU_ValueHolder component allows to hold the previous value of input by a defined time. If the value of the Input slot has changed, the value of the Output slot is set to the

previous value of the Input slot for the time defined in the Holding Time slot. After this time, the Output slot is set to current value from the Input slot.

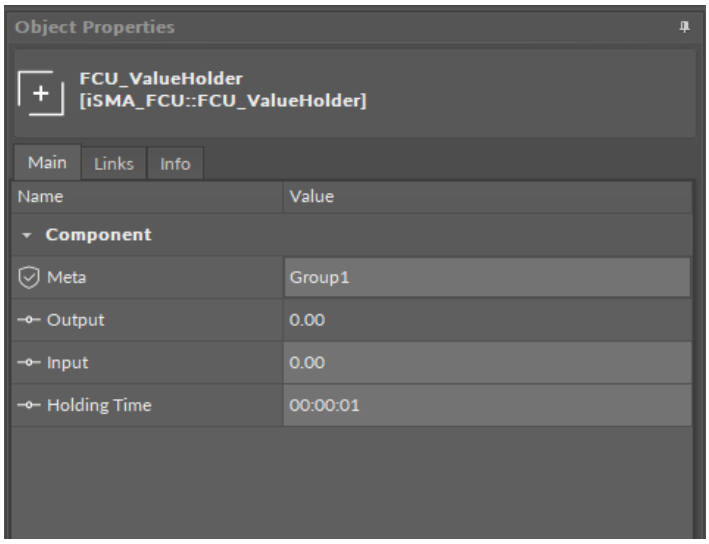


Figure 77. FCU_ValueHolder component

The FCU_ValueHolder component has the following slots:

- **Input:** the input slot;
- **Output:** the output slot;
- **Holding Time:** the time in which the input value is held after the change.

10.9 FCU_OccupancyCalculator

The FCU_OccupancyCalculator component manages the occupancy status, depending on the occupancy data provided from the BMS system or remote devices such as room panels, presence sensors, or cardholders.

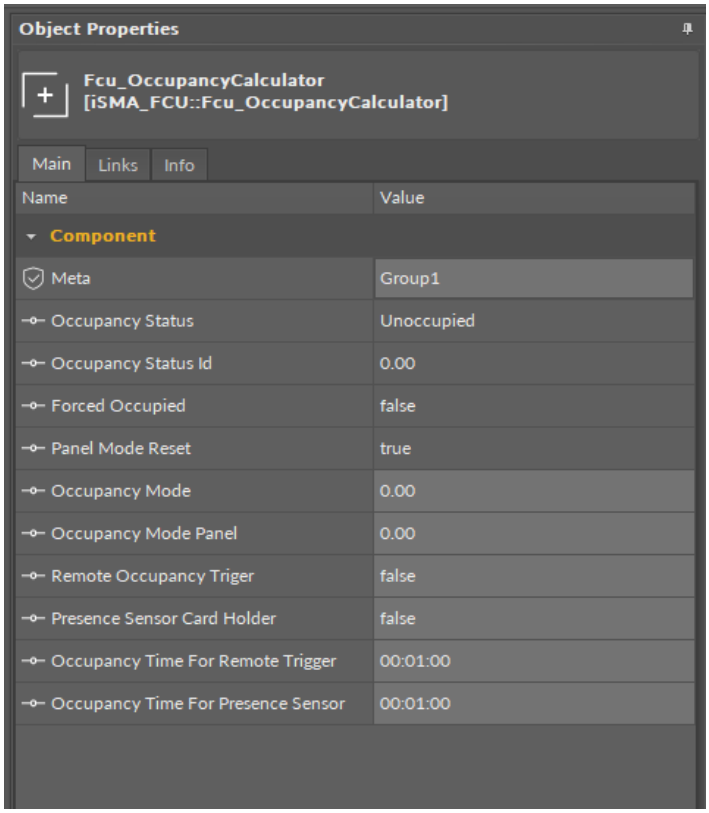


Figure 78. FCU_OccupancyCalculator component

The FCU_OccupancyCalculator component has the following slots:

- **Occupancy Status:** shows the current occupancy status;
  - Available information: (0) Unoccupied, (1) Occupied, (2) Standby;
- **Occupancy Status Id:** the output numeric value corresponding to the Occupancy Status slot;
- **Forced Occupied:** the binary output slot providing the information about the occupancy status source:
  - True: the occupied mode has been forced, which means, that the component works in the occupied mode, but this mode has not been calculated according to the Occupancy Mode slot, only forced by the room panel (the Occupancy Mode Panel slot set to Occupied), presence sensor cardholder (detected rising edge on the Presence Sensor Card Holder slot), or by the remote trigger (detected rising edge on the Remote Occupancy Trigger slot);
  - False: the occupied mode has not been forced;
- **Panel Mode Reset:** the binary output, resetting the value from a component connected to Occupancy Mode Panel slot after switching off the Occupancy Mode, which has been forced by the room panel. By default, the Panel Mode Reset slot is set to false. It is set to true for one application cycle if the time of the Occupancy mode forced by the room panel has passed;
- **Occupancy Mode:** the occupancy mode set from the higher-level system (for example, the panel dedicated to the iSMA-B-FCU device);
  - Available information: (0) Unoccupied, (1) Occupied, (2) Standby;
- **Occupancy Mode Panel:** the occupancy mode set from an external source (for example, the iSMA-B-LP room panel);
  - Available values: (0) Unoccupied, (1) Occupied;

Note: The value of the Occupancy Mode Panel slot allows to force the occupied mode from the room panel. If the Occupancy Mode slot is set to 0 or 2 (unoccupied or standby values), setting the value of the Occupancy Mode Panel slot to Occupied forces the occupied mode for the time defined in the Occupancy Time For Remote Trigger slot. During this time the Forced Occupied slot is set to true. After this time, the value of the Panel Mode Reset slot is set to true for one application cycle, and then component goes back to the previous occupancy mode. The occupied mode forced this way can be also cancelled by setting the Occupancy Mode Panel slot back to 0 (unoccupied).

- **Remote Occupancy Trigger:** the binary input slot, recommended for remote occupancy triggers, allows to force the occupied mode. If the Occupancy Mode slot is set to 0 or 2 (unoccupied or standby), the rising edge detected on this slot will force the occupied mode for the time defined in the Occupancy Time For Remote Trigger slot. The time countdown starts when a falling edge is detected in the Remote Occupancy Trigger slot. During this time, the Forced Occupied slot remains set to true. After this time, the component goes back to the previous occupancy mode. The occupied mode forced this way cannot be cancelled.
- **Presence Sensor Card Holder:** the binary input slot, recommended for presence sensors or cardholders, allows to force the occupied mode. If the Occupancy Mode slot is set to 0 or 2 (unoccupied or standby), the rising edge detected on this slot will force the occupied mode for the time defined in the Occupancy Time For Presence Sensor slot. The time countdown starts when a falling edge is detected in the Presence Sensor Card Holder slot. During this time, the Forced Occupied slot remains set to

true. After this time, the component goes back to the previous occupancy mode. The occupied mode forced this way cannot be cancelled.

- **Occupancy Time For Remote Trigger:** the time for which the occupancy mode, forced by the room panel or remote trigger, is held once the falling edge is detected on the Remote Occupancy Trigger slot;
- **Occupancy Time For Presence Sensor:** the time for which the occupancy mode, forced by the presence sensor or cardholder, is once the falling edge is detected on the Presence Sensor Card Holder slot.

10.10 FCU_HeatingCoolingSwitch

The FCU_HeatingCoolingSwitch component allows for switching between heating and cooling temperature modes, depending on the current temperature, the given setpoint, and the occupancy status.

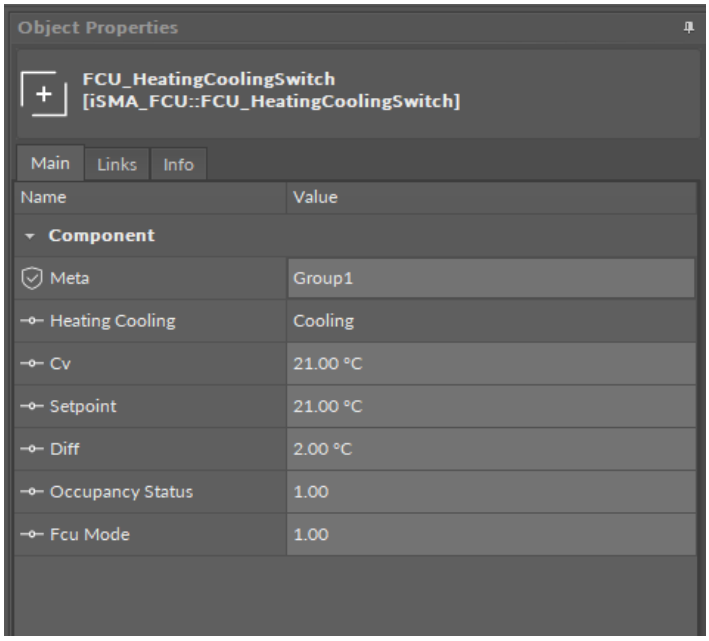


Figure 79. FCU_HeatingCoolingSwitch component

The FCU_HeatingCoolingSwitch component has the following slots:

- **Heating Cooling:** the main output of the component; shows the operating status of the component–whether it works in cooling mode (false) or heating mode (true);
- **Cv:** the current input value (measured temperature);
- **Setpoint:** the setpoint value (temperature setpoint);
- **Diff:** the deadband set for the current input value–if the Setpoint value is, for example, 25, the Diff value set to 2 means that for the Cv slot values from 23 to 27 no action is taken;
- **Occupancy Status:** sets the occupancy status;
  - Available information: (0) unoccupied, (1) occupied, (2) standby;
- **Fcu Mode:** indicating the FCU mode, for example, from the higher-level system.
  - Available values: (0) off, (1) auto, (2) heating only, (3) cooling only), (4) fan only.

Predefined modes for different values of the Fcu Mode slot:

- (0) Off: the Heating Cooling slot is permanently set to true (heating), regardless of the measured temperature and setpoint values.
- (1) Auto: the Heating Cooling slot switches between the heating and cooling modes;
- (2) Heating Only: the Heating Cooling slot is permanently set to true (heating), regardless of the measured temperature and setpoint values.
- (3) Cooling Only: the Heating Cooling slot is permanently set to false (cooling), regardless of the measured temperature and setpoint values.
- (4) Fan Only: the Heating Cooling slot is permanently set to false (cooling), regardless of the measured temperature and setpoint values.

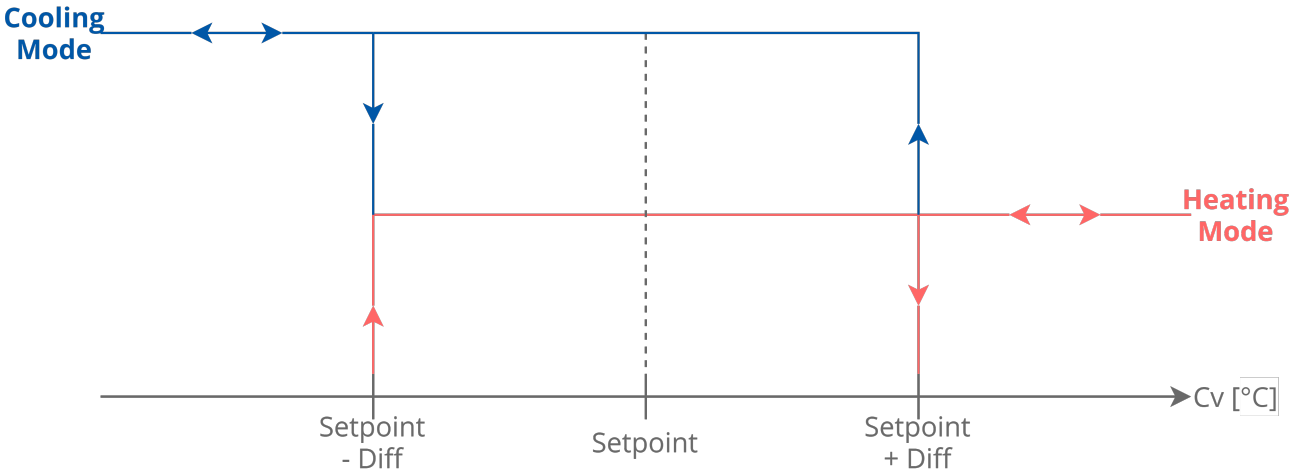


Figure 80. Switching between heating and cooling modes

Note: The heating and cooling modes can only be switched, when the Occupancy Status slot is set to 1 (Occupied mode). If the Occupancy Status slot is set to any other value, the Heating Cooling slot is set to the last mode, which has been calculated in the Occupied mode.

10.11 FCU_WindowStatusSwitch

The FCU_WindowStatusSwitch component calculates the window status (open or closed), according to the information from up to 6 Window Status slots of one master and five slave devices. If the Master Window Status slot is set to true, and the Window Statuses of active slave devices are set to true, the Window Status Out slot is set to true (window closed). In other cases, the Window Status Out slot is set to false (window open).

Note: The slave device is considered inactive if its Slave Status slot is set to false. The window status from the inactive device is not used in the Window Status Out calculation.

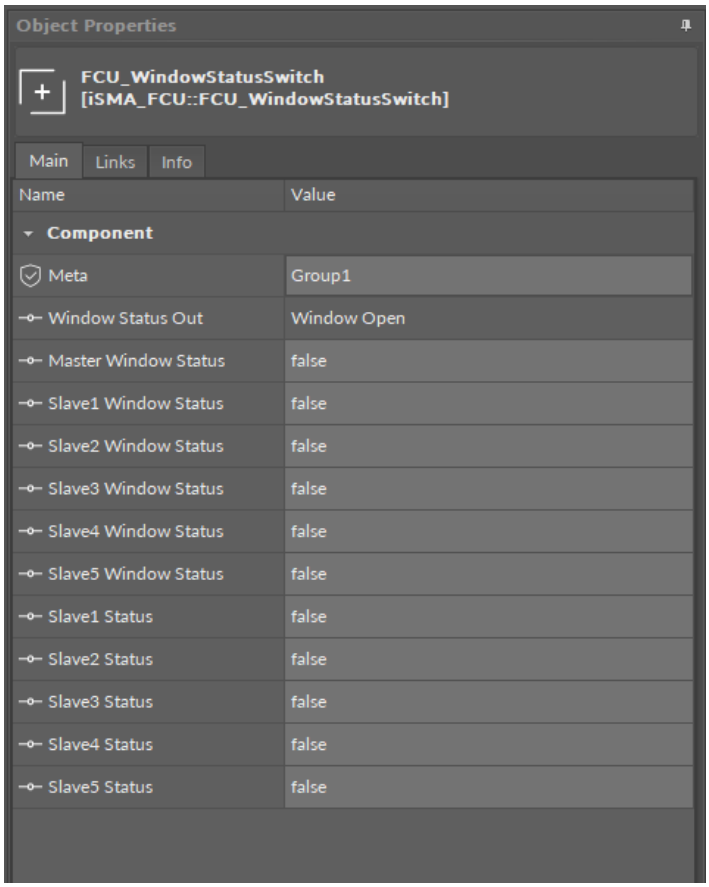


Figure 81. FCU_WindowStatusSwitch component

The FCU_WindowStatusSwitch component has the following slots:

- **Window Status Out:** informs about the current of window status (open or closed);
  - Available information: Window Closed, Window Open;
- **Master Window Status:** allows to read the window status from the master device;
  - Available settings: true (closed), false (open);
- **Slave1 Window Status-Slave5 Window Status:** allow to read the window statuses from slave devices.
  - Available settings: true (closed), false (open);
- **Slave1 Status-Slave5 Status:** allow to read the statuses of slave devices.
  - Available settings: true (slave device active), false (slave device inactive).

10.12 FCU_Multiply

The FCU_Multiply component allows to multiply up to 10 different values by the same value.

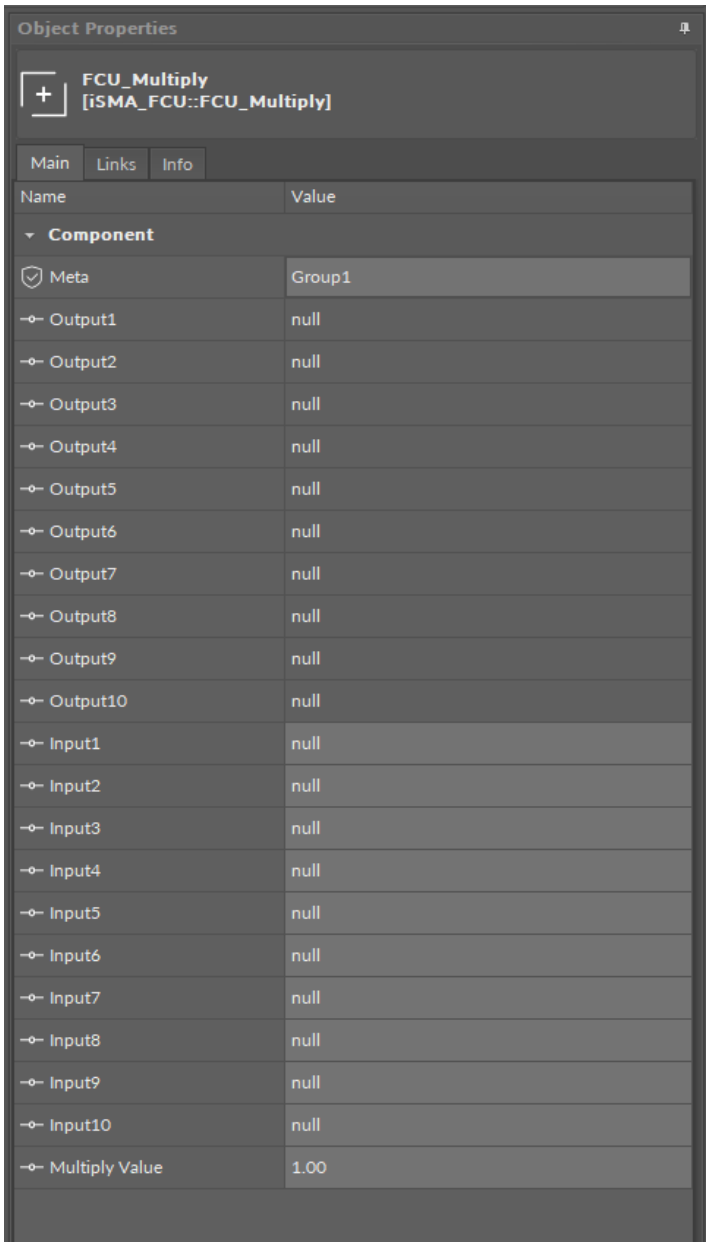


Figure 82. FCU_Multiply component

The FCU_Multiply component has the following slots:

- **Output1-Output10:** the output slots of the component, showing results of multiplication;
- **Input1-Input10:** the input slots of the component, values to be multiplied;
- **Multiply Value:** the value by which input values are multiplied.

10.13 FCU_Antifrost

The FCU_Antifrost component is used to protect against a drop in the space temperature below the set threshold (with the hysteresis).

If the value of the space temperature drops below the value equal to Threshold – Diff slots (6^{o}C for values in the figure below), and there is no sensor fault, the Antifrost slot changes to true. This process is run until the room temperature increases above the value equal to Threshold + Diff slots (8^{o}C for values in the figure below). If the

temperature sensor is faulty (the Sensor Fault slot is true), the Antifrost slot is always set to false–the component is inactive.

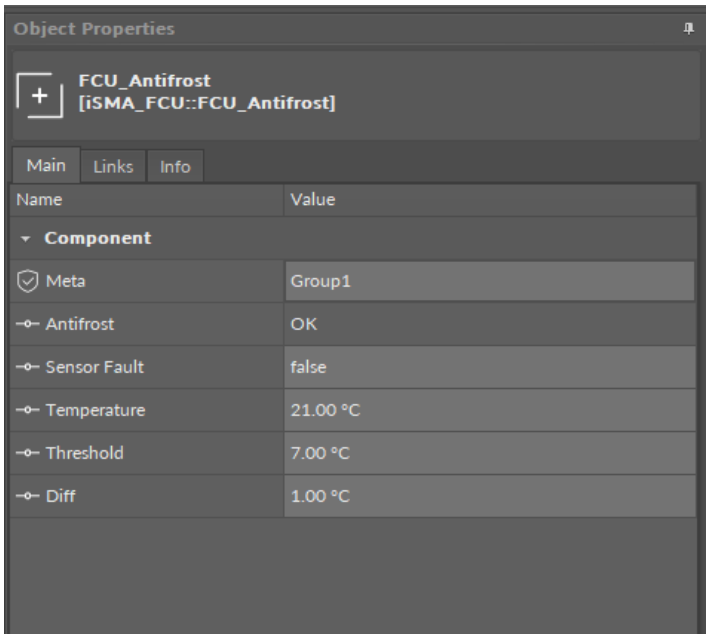


Figure 83. FCU_Antifrost component

The FCU_Antifrost component has the following slots:

- **Antifrost:** the Boolean output slot (true: antifrost, false: OK (no antifrost);
- **Sensor Fault:** the Boolean input slot, which provides the information about the fault of the temperature sensor;

Note: If the Sensor Fault slot is true, the Antifrost component is inactive (because the value read from the temperature sensor is incorrect), and the value of the Antifrost slot is set to false. If the Sensor Fault slot is false, the Antifrost component is active,

- **Temperature:** the value from a temperature sensor for controlling the space;
- **Threshold:** allows to set the threshold temperature value;
- **Diff:** allows to set the differential for hysteresis.

10.14 PWM

The PWM component implements a pulse-width modulation (PWM) mechanism. The PWM is based on two slots:

- **Period slot:** the time period of the cycle, and
- **Duty Cycle slot:** the value of the pulse width, expressed as percentage. The Duty Cycle slot defines the period of time (value from the Period slot), during which the Out slot is set to true. During the remaining period of time, Out slot will be set to "false".

For example, according to the values in the figure below:

- The Out slot will be set to true for 10 seconds: value from the Duty Cycle slot multiplied by the value from the Period slot (10% * 100s);
- After this, the Out slot will be set to false for 90 seconds (90% * 100s);
- The state of the Out slot is changed in the above way periodically (in periods defined in the Period slot).

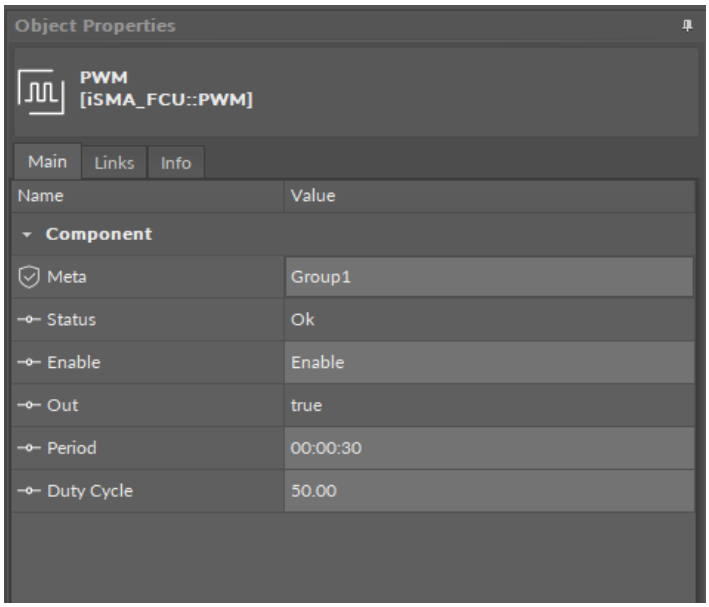


Figure 84. PWM component

The PWM component has the following slots:

- **Status:** displays the status of the component;
- **Enable:** enables/disables the component; true: enabled, false: disabled;
- **Out:** the binary output slot with the current state calculated by the component's algorithm;
- **Period:** the numeric input slot with the period of modulation;
- **Duty Cycle:** the numeric input slot corresponding to the pulse width.

10.15 FCU_PID

The FCU_PID component is a regulator with proportional, integral and derivative actions.

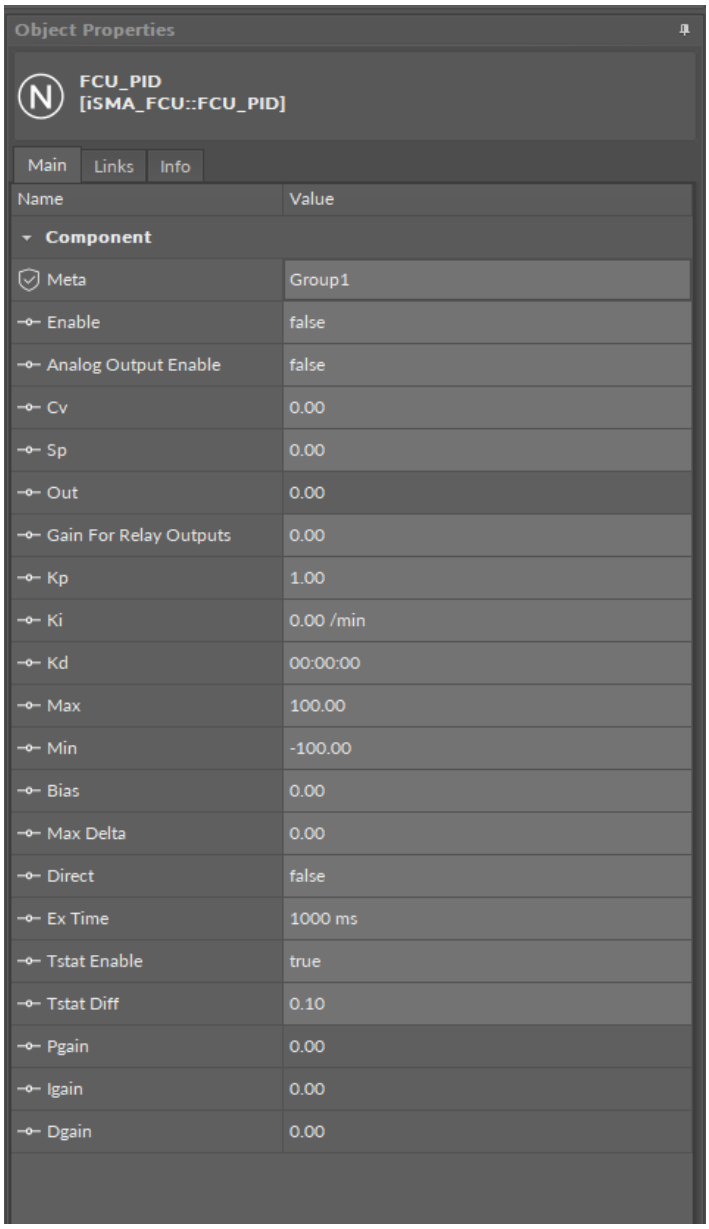


Figure 85. FCU_PID component

The FCU_PID component has the following slots:

- **Enable:** enables or disables the component–if the component is disabled (the Enable slot is set to false), the Out slot is set to 0;
- **Analog Output Enable:** the binary input slot with the information about control type;
    - Available settings: true, false;

True: sets the analog control. The component uses available actions:
- proportional with gain equal to the value of Kp slot;
- integral, according to the constant set in the Ki slot;
- derivative, according to the constant set in the Kd slot.

False: sets the binary control. The component works using only proportional gain. In this type of control, the proportional gain is set in the Gain For Relay Outputs slot.

- **Cv:** the controlled value;
- **Sp:** allows to set the setpoint for the controlled value;
- **Out:** the output slot of the component;

- **Gain For Relay Outputs:** allows to set the gain used to calculate the Out slot if the component controls binary outputs (the Analog Output Enable slot is set to false);
- **Kp:** allows to set the value of the proportional gain constant;
- **Ki:** allows to set the value of the integral constant (setting to 0 disables the integral action);
- **Kd:** allows to set the value of the derivate constant (setting to 0 disables the derivate action);
- **Max:** allows to set the maximum value of the output of component;
- **Min:** allows to set the minimum value of the output of component;
- **Bias:** allows to set the bias value–this value is added to the Out slot if Ti slot is set to 0;
- **Max Delta:** allows to set the maximum amount the Out slot can change by in the exTime (setting to 0 disables this function);
- **Direct:** allows to set the acting process;
    - Available settings: true (direct-acting process), false (reverse acting process);
- **Ex Time:** allows to set the period of loop execution;
- **Tstat Enable:** enables/disables the Tstat function;
- **Tstat Diff:** allows to set the differential value for the Tstat function;

Note: The Tstat function allows to reset the integral action if the absolute value of the difference between the Cv slot and Sp slot is lower than the value from the Tstat Diff slot. In this case, the sum of error (used for calculating the integral gain) is set to 0.

- **Pgain:** allows to set the value of the output signal, which was calculated by the proportional action;
- **Igain:** allows to set the value of the output signal, which was calculated by the integral action;
- **Dgain:** allows to set the value of the output signal, which was calculated by the derivative action.

10.16 FCU_TemperatureAnalog

The FCU_TemperatureAnalog component allows to calculate numeric values for the Heating Analog Output and Cooling Analog Output slots, according to the external analog Control Value from range -100%-100% (for example, from the analog PID regulator). The component is dedicated to control valve actuators with analog inputs (or actuators, which can be controlled by the triac outputs with PWM). Negative values of the Control Value are used to calculate the Cooling Analog Output slot, and positive values are used to calculate the Heating Analog Output slot.

The FCU_TemperatureAnalog component can work in two temperature control modes:

- One-stage mode: only analog outputs (the Heating Analog Output and Cooling Analog Output slots) are calculated according to the Control Value;
- Two-stage mode: analog outputs are used for the first stage, and dedicated binary outputs (the Heating Second Stage Binary Output and Cooling Second Stage Binary Output) are used for the second.

Note: Operation in two stages mode can be selected by setting the Heating Second Stage Enable and/or Cooling Second Stage Enable slots to true. If one of these slots is set to false, the component will operate in one stage mode for corresponding temperature mode (heating or cooling).

For proper operation, the component has to be enabled (the Temperature Analog Enable slot set to true), and the Fan Active slot has to be set to true. If the second condition is not met, the component is enabled, but the main outputs are blocked–values of the Heating Analog Output and Cooling Analog Output slots are set to 0, and the Heating Second Stage Binary Output and Cooling Second Stage Binary Output slots are set to false.

The values of the Heating Analog Output and Cooling Analog Output slots for one-stage mode control are calculated as shown in the figures below:

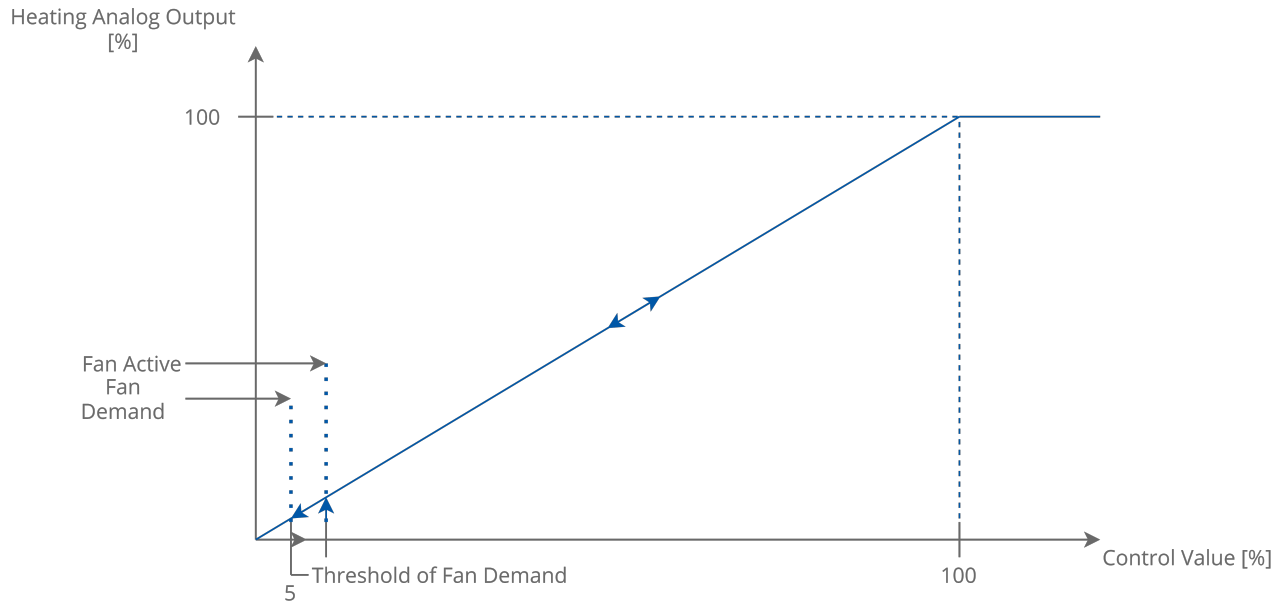


Figure 86. Analog control of temperature for 1st stage only

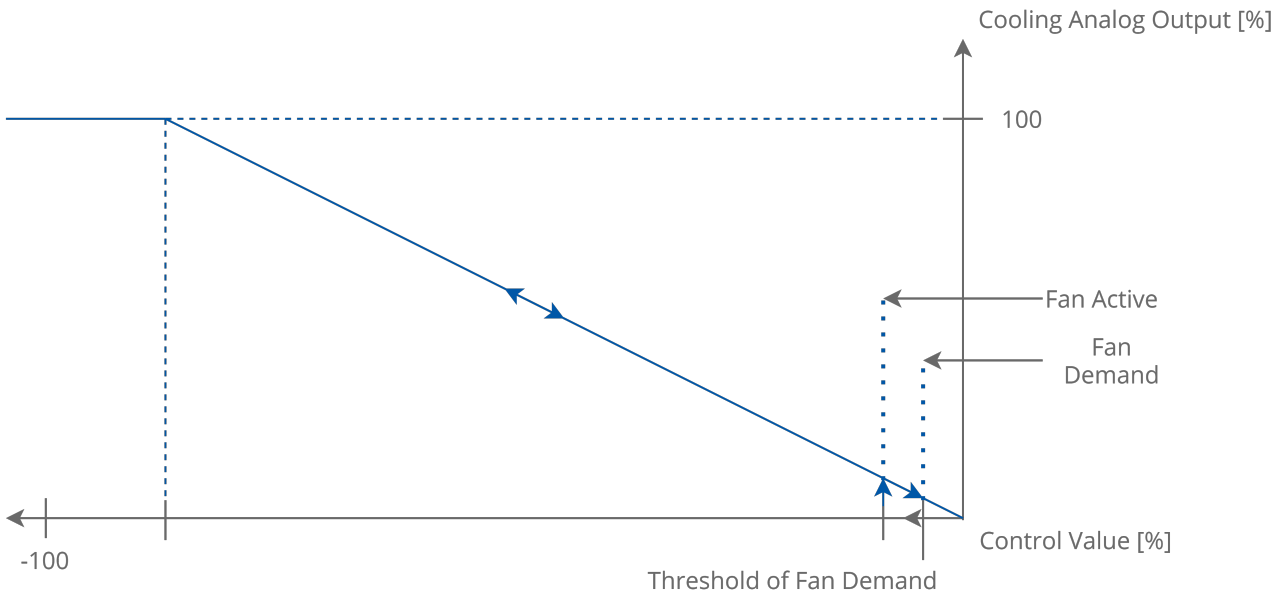


Figure 87. Analog control of temperature for 1st stage only

In this mode, the slots for the second stage are not used (the Heating Second Stage Binary Output and Cooling Second Stage Binary Output slots are still set to false).

The values of the Heating Analog Output and Cooling Analog Output slots for the first stage, and the values of the Heating Second Stage Binary Output and Cooling Second Stage Binary Output slots (for the second stage) for two-stage mode are calculated as shown in the figures below:

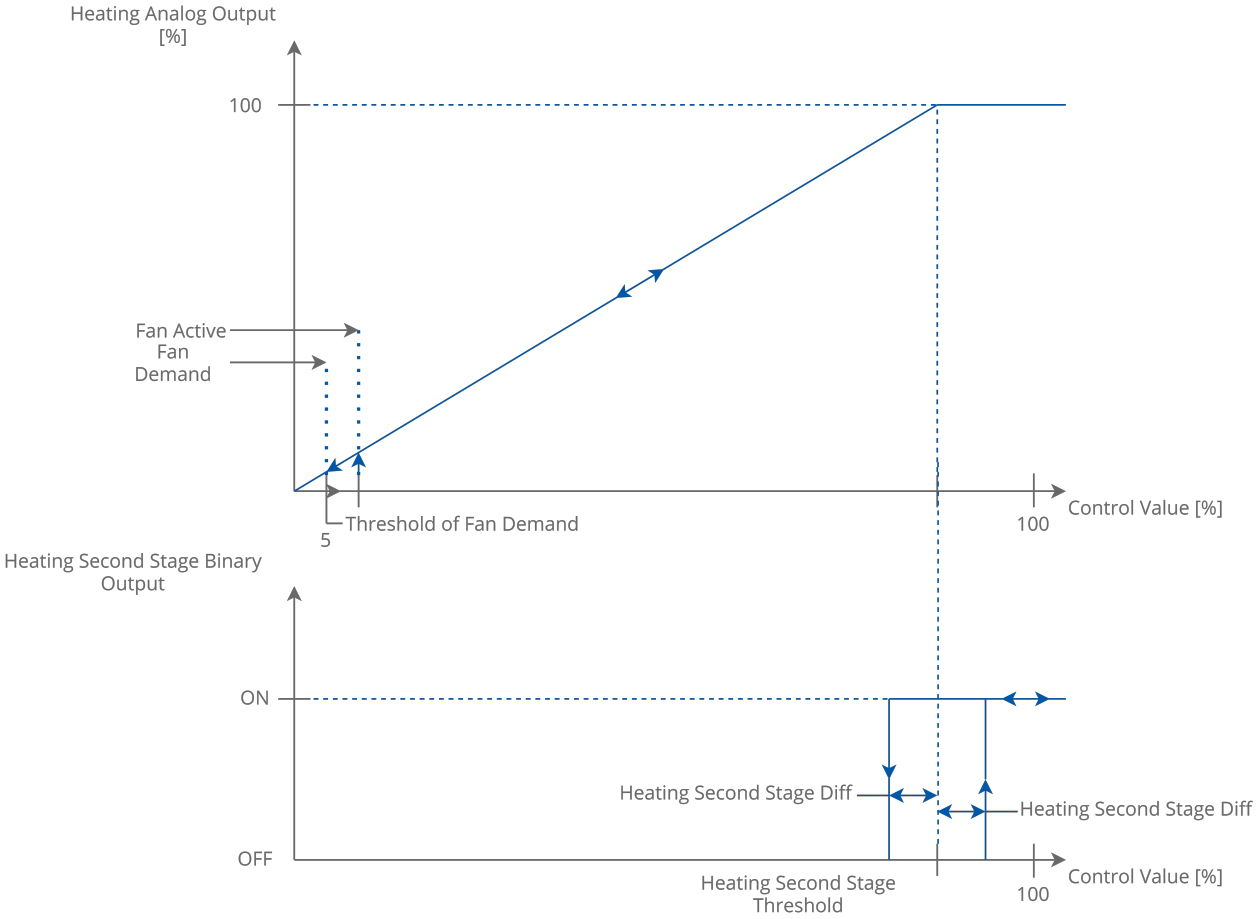


Figure 88. Analog control of temperature for 1st and 2nd stage - heating mode

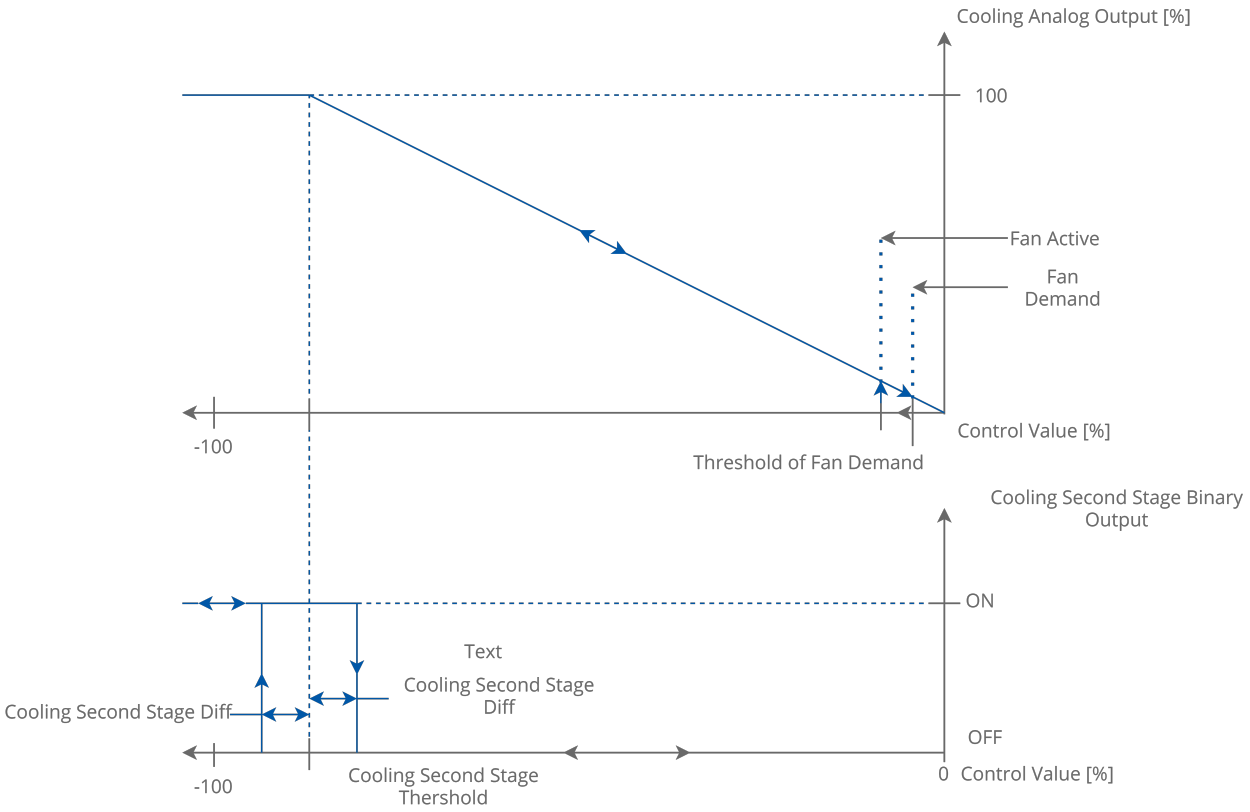


Figure 89. Analog control of temperature for 1st and 2nd stage - cooling mode

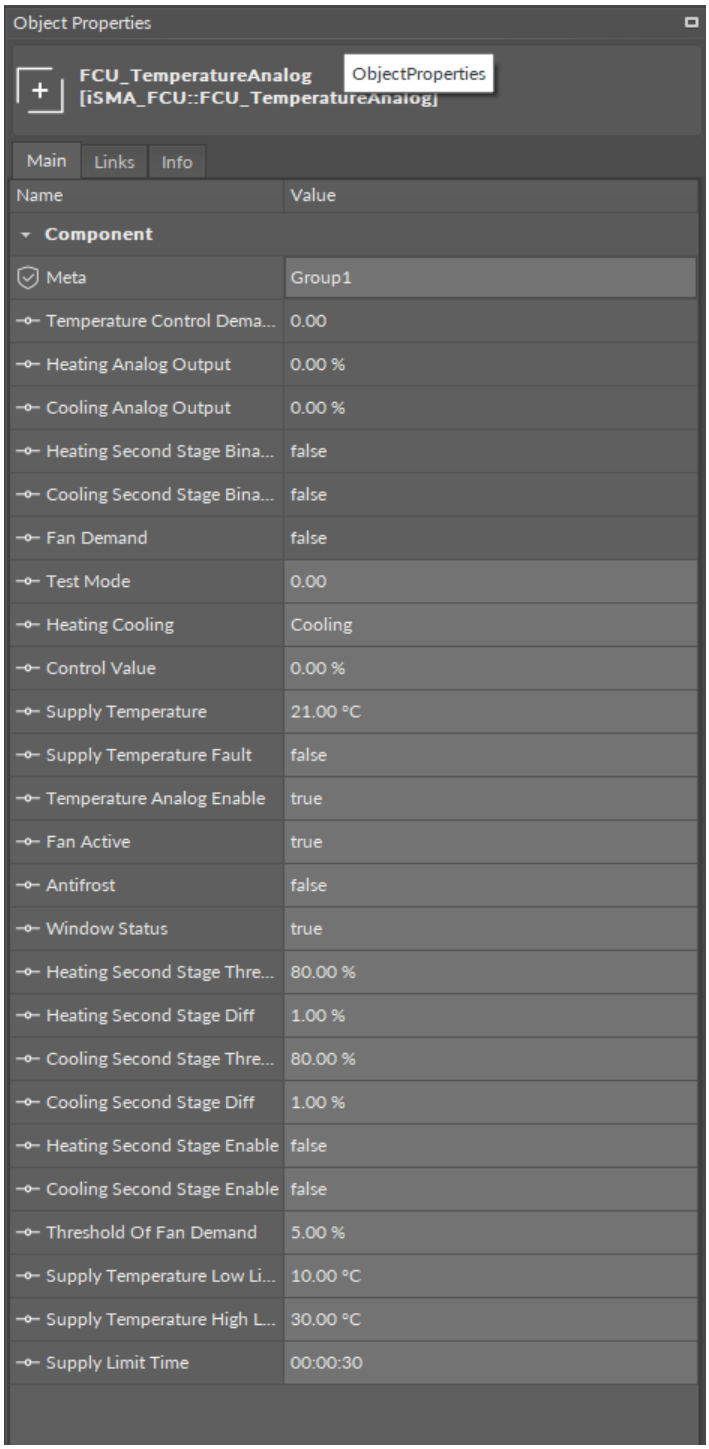


Figure 90. FCU_TemperatureAnalog component

The FCU_TemperatureAnalog component has the following slots;

- **Temperature Control Demand:** shows the current component temperature demand;
  - Available information: (1) HeatingDemand, (2) CoolingDemand;
- **Heating Analog Output:** shows the heating demand level, expressed in percentage;
- **Cooling Analog Output:** shows the cooling demand level, expressed in percentage;
- **Heating Second Stage Binary Output:** shows the state of heating in the second stage;
- **Cooling Second Stage Binary Output:** shows the state of cooling in the second stage;
- **Fan Demand:** shows the fan demand;
  - Available information: true–if the absolute value from the Control Value slot exceeds the value set in the Fan Demand Threshold slot, false–in other cases;

- **Test Mode:** allows to set one of the predefined test modes;
  - Available settings: 0 (the component uses its main algorithm), 1 (full heating), 2 (full cooling);

Full Heating Test Mode

In the Full Heating mode, the **Fan Demand** slot is set to true. The **Heating Analog Output** slot is set to 100%, and the **Heating Second Stage Binary Output** slot is set to true (only if the **Heating Second Stage Enable** slot is set to true).

Full Cooling Test Mode

In the Full Cooling mode, the **Fan Demand** slot is set to true. The **Cooling Analog Output** slot is set to 100%, and the **Cooling Second Stage Binary Output** slot is set to true (only if the **Cooling Second Stage Enable** slot is set to true).

Note: Before starting the Full Heating or Full Cooling mode, the value of the **Fan Active** slot has to be set to true.

- **Heating Cooling:** sets the current temperature mode;
  - Available settings: true (heating), false (cooling);

Note: If the component operates in one of the above modes, outputs corresponding to the other mode are blocked. For example, if the component works in the Heating mode, the Cooling Analog Output is set to 0%, and the Cooling Second Stage Binary Output slot is set to false.

- **Control Value:** receives the value from an external component (for example, from the PID regulator, the Loop component); based on the received value, the values for output slots of the component are calculated; range: -100%-100%. Negative values of the Control Value are used for calculating the open level of the cooling valve and positive values for calculating the open level of the heating valve.
- **Supply Temperature:** allows to read the value of the supply temperature;
- **Supply Temperature Fault:** allows to read the information about the fault of the supply temperature;
- **Temperature Analog Enable:** allows to enable or disable the TemperatureAnalog component–if the component is disabled, analog outputs are set to 0, and binary outputs are set to false;
  - Available settings: true (enabled), false (disabled);
- **Fan Active:** informs the TemperatureAnalog component that the fan is switched on; if the slot is set to false, analog outputs (the Heating Analog Output and Cooling Analog Output slots) are set to 0, and binary outputs (the Heating Second Stage Binary Output and Cooling Second Stage Binary Output slots) are set to false–these slots can be set to other values (calculated by the main algorithm) only when the Fan Active slot is set to true.
- **Antifrost:** allows to switch on the Antifrost mode;
  - Available settings: true (enabled), false (disabled);

Note: In the Antifrost mode enabled, the **Fan Demand** slot is set to true, the **Heating Analog Output** is set to 100%, and the **Heating Second Stage Binary Output** is set to true (only if the **Heating Second Stage Enable** slot is set to true).

Note: The Antifrost mode has higher priority than the main algorithm, but it can be overridden by the Test mode.

- **Window Status:** allows to switch on the Window Open mode;

- Available settings: true (Window Open mode disabled), false (Window Open mode enabled–the component operates in saving energy mode, analog outputs are set to 0, and binary outputs for the second stage are set to false);

Note: The Window Open mode can be overridden only by the Antifrost mode or the Test mode.

- **Heating Second Stage Threshold:** sets the threshold of the Control Value, above which (with the hysteresis) the heating in second stage is switched on;
- **Heating Second Stage Diff:** sets the differential for hysteresis of switching on/off the heating in second stage;
- **Cooling Second Stage Threshold:** sets the threshold of the Control Value, above which (with the hysteresis) the cooling in second stage is switched on;
- **Cooling Second Stage Diff:** sets the differential for hysteresis of switching on/off the cooling in second stage;
- **Heating Second Stage Enable:** allows to enable or disable the heating in second stage;
  - Available settings: true (enabled), false (disabled);
- **Cooling Second Stage Enable:** allows to enable or disable the cooling in second stage;
  - Available settings: true (enabled), false (disabled);
- **Threshold Of Fan Demand:** sets the threshold of the Control Value, above which the Fan Demand slot is set to true;
- **Supply Temperature Low Limit:** sets the minimum acceptable value of the supply temperature–this value is used in the Supply Air Temperature Limitation function;
- **Supply Temperature High Limit:** sets the maximum acceptable value of the supply temperature–this value is used in the Supply Air Temperature Limitation function;
- **Supply Limit Time:** sets the delay time for activation of the Supply Air Temperature Limitation function.

Supply Air Temperature Limitation

In order to maintain room conditions comfortable for the user, the supply air can have a temperature limitation. This function is available only if the supply air sensor is connected and works correctly. The supply air temperature can have a high limit defined by the Supply Temperature High Limit slot, and a low limit defined by the Supply Temperature Low Limit slot. The range between the Supply Temperature Low Limit and Supply Temperature High Limit values is called a comfort range.

- Supply Air Temperature limitation in the first stage analog control

In analog control, if the supply air temperature approaches the comfort range by 1°C, the **FCU_TemperatureAnalog** component starts the countdown set in the **Supply Limit Time** slot delay time . After this time, if the supply air temperature value is still approaching the comfort range limit by 1°C, the component starts a built-in algorithm, which reduces the air temperature (if the temperature value is close to or above the **Supply Temperature High Limit**), or increases the air temperature (if the temperature value is close to or below the **Supply Temperature Low Limit**). If the supply air temperature value returns to the comfort range ±1°C, the component will reset delay counter and return to normal operation.

- Supply Air Temperature limitation in the second stage analog control

In analog control, if the supply air temperature approaches the comfort range by 1°C, the **FCU_TemperatureAnalog** component disables the second stage, and starts counting the delay time set in the **Supply Limit Time** slot. After this time, if the supply

air temperature value is still approaching the comfort range by 1°C, the component starts the built-in algorithm, which reduces the air temperature (if the temperature value is close to or above the **Supply Temperature High Limit**), or increases the air temperature (if the temperature value is close to or below the **Supply Temperature Low Limit**). If the supply air temperature value returns to the comfort range ±1°C, the component resets the delay counter, enables the second stage, and returns to normal operation.

10.17 FCU_SensorFault

The FCU_SensorFault component allows to detect the sensor fault according to the temperature value.

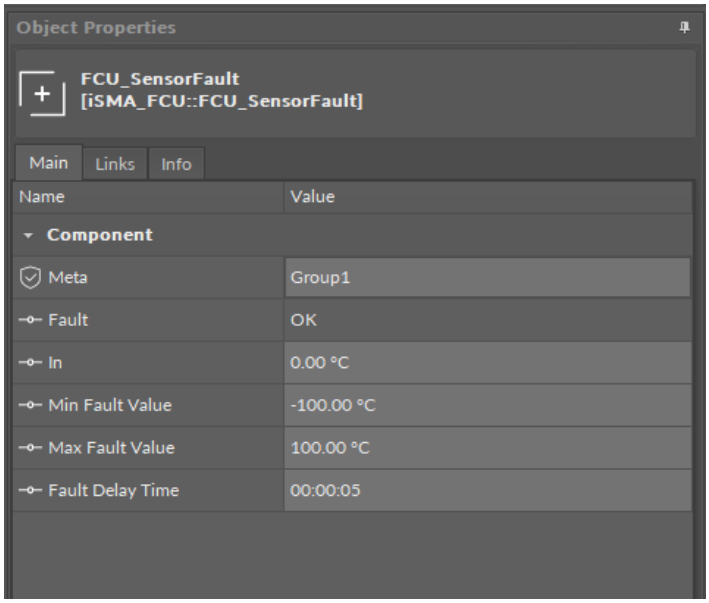


Figure 91. FCU_SensorFault component

The FCU_SensorFault component has the following slots:

- **Fault:** the binary output switch;

If the temperature from the sensor (set in the In slot) is lower than the value of the Min Fault Value, or higher than the value of the Max Fault Value slot, for the time longer than set to the Fault Delay Time slot, the Fault slot is set to true (sensor is fault). In other cases, the Fault slot is set to false (sensor is OK).

- **In:** the input slot for the temperature value from the sensor;
- **Min Fault Value:** allows to set the value of the minimum acceptable temperature;
- **Max Fault Value:** allows to set the value of the maximum acceptable temperature;
- **Fault Delay Time:** allows to set the delay time for detecting the sensor's fault.

10.18 FCU_ModeCalculator

The FCU_ModeCalculator component allows to switch the main modes of application (for example, to override the fan mode according to others values, or to enable/disable temperature control), according to a fan mode, Fcu Mode, and type of temperature control (analog or binary). The component can be used to protect against the incorrect operation of FCU device.

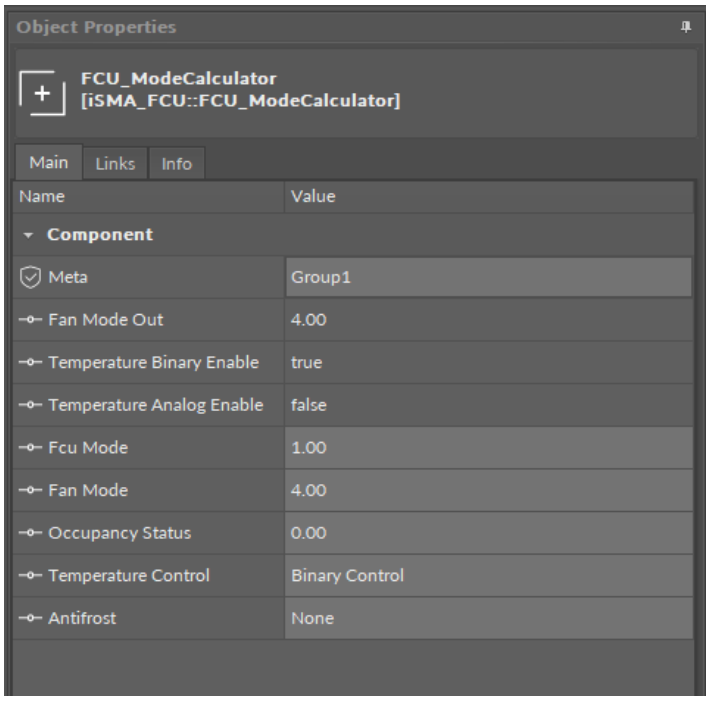


Figure 92. FCU_ModeCalculator component

The FCU_ModeCalculator component has the following slots:

- **Fan Mode Out:** the output of the fan mode value, calculated according to the algorithm of component;

Modes Dependencies

If the value of the **Fcu Mode** slot is equal to 0 (FCU is switched off from the upper-level system), the **Fan Mode Out** slot is also set to 0 (the fan is also switched off).
If the value of the **Fan Mode** slot is lower than 4 (the fan is off, or works in one of the manual modes), and the value of the **Occupancy Status** slot is not equal to 1 (the component works in the unoccupied or standby mode), the **Fan Mode Out** slot is set to 4 (the fan works in the auto mode). In other cases, the value of the **Fan Mode Out** slot is equal to the value of the **Fan Mode** slot.

- **Temperature Binary Enable:** enables or disables the binary temperature mode, according to the algorithm of component;

Temperature Binary Modes Dependencies

If the **Temperature Control** slot is set to false (binary control), and the value of the **Fan Mode Out** slot is higher than 0 (the fan is switched on), the **Temperature Binary Enable** slot is set to true.
If the **Temperature Control** slot is set to false (binary control), and the **Antifrost** slot is set to true (the component works in the antifrost mode), the **Temperature Binary Enable** slot is set to true. In other cases, the **Temperature Binary Enable** slot is set to false.

- **Temperature Analog Enable:** enables or disables the analog temperature mode, according to the algorithm of the component;

Temperature Analog Modes Dependencies

If the **Temperature Control** slot is set to true (analog control), and the value of the **Fan Mode Out** slot is higher than 0 (the fan is switched on), the **Temperature Analog Enable** slot is set to true.
If the **Temperature Control** slot is set to true (analog control), and the **Antifrost** slot is set to true (the component works in the antifrost mode), the **Temperature Analog Enable** slot is set to true. In other cases, the **Temperature Analog Enable** slot is set to false.

- **FCU Mode:** sets the mode of the FCU (receives numeric values);
  - Available settings: (0) Off, (1) Auto, (2) Heating Only, (3) Cooling Only, (4) Fan Only;

Note: If the FCU Mode slot is set to 4 (fan only), and the Antifrost slot is set to false (no antifrost mode), the Temperature Binary Enable and Temperature Analog Enable slots are overridden to false.

- **Fan Mode:** sets the fan mode (receives numeric values);
  - Available settings: (0) Off, (1) Manual Speed 1, (2) Manual Speed 2, (3) Manual Speed 3, (4) Auto;
- **Occupancy Status:** sets the occupancy status (receives numeric values);
  - Available settings: (0) Unoccupied, (1) Occupied, (2) Standby;
- **Temperature Control:** sets the mode of the temperature control;
  - Available settings: true (analog control), false (binary control);
- **Antifrost:** enables or disables the antifrost mode;
  - Available settings: true (Antifrost enabled), false (Antifrost disabled).

10.19 FCU_FanControl

The FCU_FanControl is a component to control the fan. The component has been created for 1-, 2-, or 3-speed fans, and for fans with analog inputs (the type of the fan can be selected by the user). The fan algorithm can be split into two modes:

- **Standard:** the demand for the fan is active, and the fan speed is calculated based on the temperature value. The standard mode conditions:
  - the FanDemand slot is false;
  - the Antifrost slot is false;
  - the HeatingOccupiedActive slot is false;
  - the CoolingOccupiedActive islot s false;
  - the TestMode slot is 0.
- **Non-standard:** the additional parameters override the fan speed. The non-standard mode must comprise at least one of the slots: FanDemand, Antifrost, HeatingOccupiedActive, CoolingOccupiedActive, is true, or the Test Mode is higher than 0.

In the standard mode, the fan is switched on when the internal variable, the Fan Control Value (calculated on the basis of the CV and Setpoint slots), is higher than the Fan Speed 1 Threshold, and switched off when the Fan Control Value is lower than the Fan Off Threshold.

The non-standard operation is defined by the slots states combinations and is described below.

Figure 93. FCU_FanControl component

The FCU_FanControl component has the following slots:

- **Fan Status:** indicates the current status of the fan;
  - Available information: Off (0), Speed 1 manual (1), Speed 2 manual (2), Speed 3 manual (3), Speed 1 auto (4), Speed 2 auto (5), Speed 3 auto (6);
- **Fan Status Id:** shows the numeric value corresponding to the Fan Status slot;
- **Fan Analog Out:** the component's output for the fan with analog input, expressed as percentage; for fans with discrete inputs (the Fan Type slot is set to 1, 2 or 3), the value of the Fan Analog out is equal to 0%;

- **Speed 1, Speed 2, Speed 3:** the component's outputs for fans with binary inputs; for fans with analog inputs (the Fan Type slot is set to 0), states of the Speed 1, Speed 2, and Speed 3 slots are set to false, and cannot be changed by the algorithm of the component;

Note: The FanControl component has a built-in protection against enabling several speeds at the same time, which could cause physical damage to the fan. If the current fan speed has to be changed to another one, all binary outputs responsible for fan speeds are disabled for 1 second, and only then the new speed is enabled.

- **Fan Active:** allows to confirm the operation of the fan;

Note: If the value of the Fan Status ID slot is higher than 0, the Fan Active slot is set to true. In other cases, the state of the Fan Active slot is set to false.

- **Fan Mode:** the main input of the component;
  - Available settings: Off (0), Speed 1 manual (1), Speed 2 manual (2), Speed 3 manual (3), Auto (4);

Fan Modes:

- (0) Off: the fan is disabled.
- (1) Speed1(Manual): the fan works with speed 1, regardless of temperature values. If the Fan Type slot is set to 0 (fan with analog input), the value of the Fan Analog Out is set to the value from the Fan Speed 1 Threshold slot.
- (2) Speed2(Manual): the fan works with speed 2, regardless of temperature values. If the Fan Type slot is set to 0 (fan with analog input), the value of the Fan Analog Out is set to the value from the Fan Speed 2 Threshold slot.
- (3) Speed3(Manual): the fan works with speed 3, regardless of temperature values. If the Fan Type slot is set to 0 (fan with analog input), the value of the Fan Analog Out is set to the value from the Fan Speed 3 Threshold slot.
- (4) Auto: the fan works in the automatic mode, the current speed depends on the current space temperature and the setpoint.

Note: The value of the Fan Mode slot (or the current speed, without changing the Fan Mode slot) can be overridden by the built-in algorithm of the component, disregarding the value that is set to the Fan Mode slot. It can occur in the following cases:

- The component works in the Unoccupied or Standby mode (the value of the Occupancy Status slot is set to 0 or 2)–the Fan Mode slot is overridden to the Auto mode always when the set value is different than 0. The overriding stops if the component works in the Occupancy mode (value of the Occupancy Status slot is set to 1).
- The window is open (the Window Status slot is set to false)–the Fan Mode slot is overridden to 0 (Off mode). The overriding stops when the Window Status slot changes to true.
- The component works in the Antifrost mode (the Antifrost slot is set to true, even the Window Status slot is set to false)–the current speed is overridden by the maximum value available for the type of the fan (depending on the value of the Fan Type slot). The overriding stops when Antifrost slot changes to false.

- The component works in the testing mode (the value of the **Test Mode** slot is not equal to 0)–the current speed is overridden by the maximum value available for the type of the fan (depending on the value of the **Fan Type** slot). The overriding stops when the **Test Mode** slot changes to 0.

- **Fan Type:** sets the type of the controlled fan (receives numeric values);
  - Available settings: (0) analog, (1) fan with 1 binary speed, (2) fan with 2 binary speeds, (3) fan with 3 binary speeds;

Note: The FCU_FanControl component has a built-in protection against enabling speeds higher than these resulting from the value of the Fan Type slot. For example, if the Fan Type slot is set to 1 (fan with 1 binary speed), it is not possible to enable speeds higher than 1. This protection pertains only to fans with binary outputs.

- **Occupancy Status:** sets the occupancy status (receives numeric values).
  - Available settings: (0) unoccupied, (1) occupied, (2) standby;
- **Test Mode:** allows to enable or disable the testing mode. This mode is inactive if the value of the slot equals 0. In other cases, fan works in the testing mode–the current speed will be overridden by the maximum value available for the fan type (depending on the value of the Fan Type slot);
- **Cv:** the measured temperature, which is used for calculating the fan speed in the Auto mode;
- **Setpoint:** sets the temperature setpoint, which is used for calculating the fan speed in the Auto mode;
- **Fan Demand:** allows to force switch the fan on if it is off (the Fan Active slot is set to false);

Fan Demand:

If the fan is off (the **Fan Mode** slot is set to 0), or it works in the Auto mode (the **Fan Mode** slot is set to 4), but the speed calculated by the algorithm equals 0, the fan can be switched on by setting the **Fan Demand** slot to true. In this case, the fan works with speed 1 (for fans with binary inputs), or with the analog value set in the **Fan Speed 1 Threshold** slot (for the fan with analog input). If the speed (or analog output) calculated by the algorithm is higher than speed 1 (or the value from the **Fan Speed 1 Threshold** slot, for analog output), the speed switched on using the **Fan Demand** slot, is overridden by this value.

- **Heating Cooling:** allows to set the temperature mode, which the fan works in;
  - Available settings: true (heating), false (cooling);
- **Antifrost:** allows to switch on the Antifrost mode;
  - Available settings: true (Antifrost mode enabled, the current speed will be overridden by the maximum value available for the fan type, depending on the value of the Fan Type slot), false (Antifrost mode disabled);
- **Window Status:** allows to enable the Window Is Open mode;
  - Available settings: true (Window Is Open mode disabled), false (Window Is Open mode enabled, the current value of the Fan Mode slot will be overridden to Off);

Note: The Window Is Open mode can be overridden only by the Antifrost mode or the Test mode.

- **Fan Off Threshold, Fan Speed 1 Threshold, Fan Speed 2 Threshold, Fan Speed 3 Threshold:** set values of thresholds used for switching fan speeds (in Auto mode, for fans with binary inputs), calculating the value of the Fan Analog Out slot (in manual modes, for the fan with analog output);
- **Fan Scale:** sets the value, which is used for calculating the fan speed in the Auto mode;

Note: Calculating the fan speeds is based on the internal variable named Fan Control Value. The way of calculating this value is presented by the figure below:

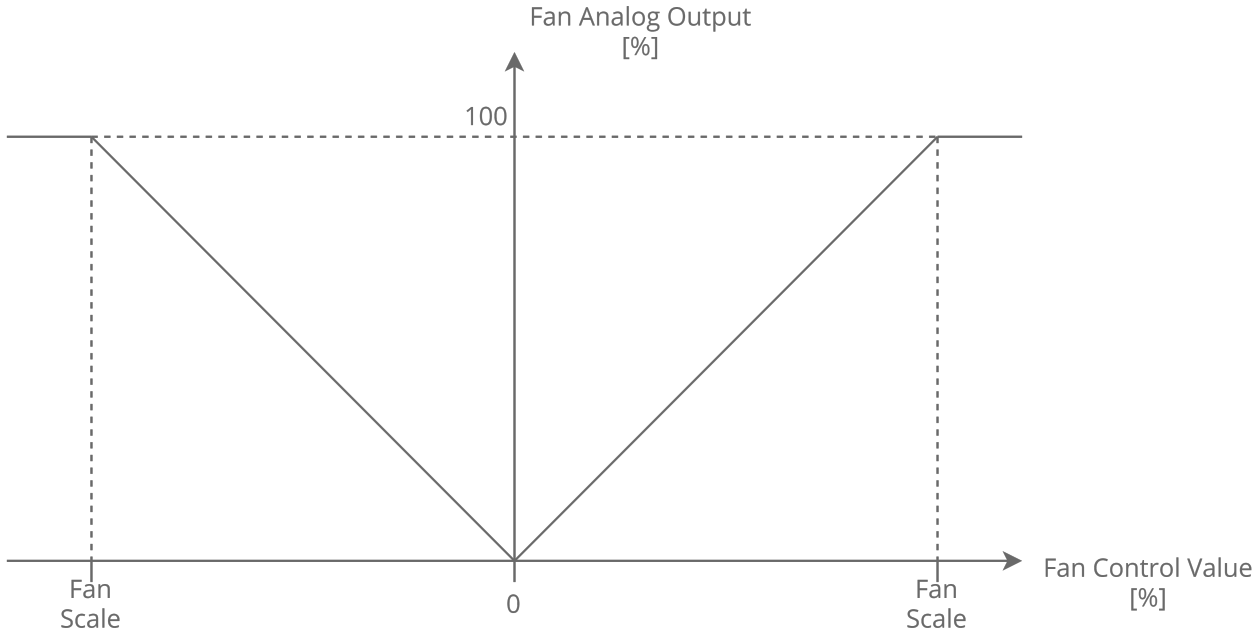


Figure 94. The way of calculating the fan control value

The Fan Control Value is used to calculate the current speed of the fan (for fans with binary inputs), or the value of the Fan Analog Out slot (for fans with analog input). The way of calculating both values is presented by the below figures:

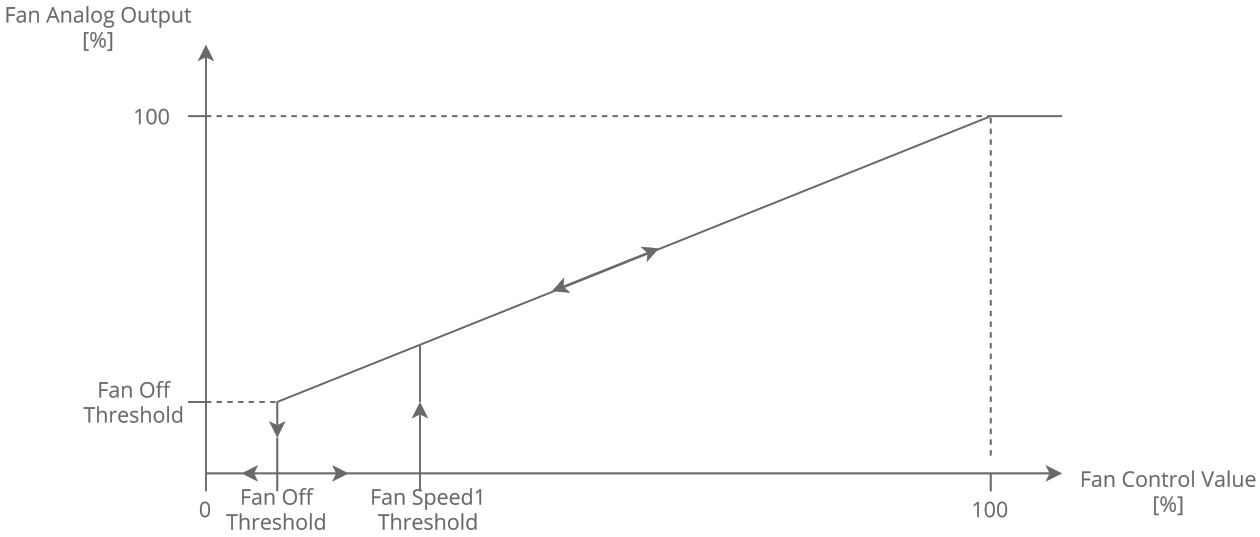


Figure 95. Control of fan with analog inputs in the auto mode

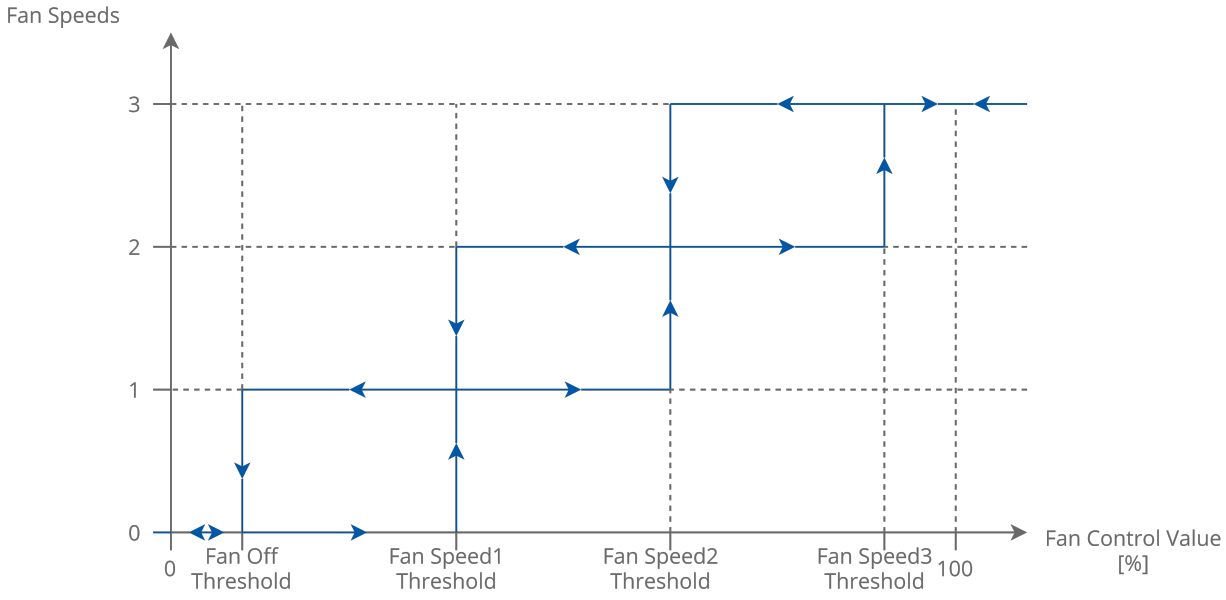


Figure 96. Control of fan with binary inputs in the auto mode

- **Heating Occupied Active:** allows to enable or disable the function enforcing fan operation;
  - Available settings: true (enabled), false (disabled);

Heating Occupied Active Enabled

If the fan works in the Auto mode (the **Fan Mode** slot is set to 4), the space is occupied (the **Occupancy Status** slot is set to 1), and the FCU_FanControl component works in the heating temperature mode (the **Heating Cooling** slot is set to true), the fan will always be switched on, even if the value of the **Setpoint** slot is lower than value of the **Cv** slot, in which case, according to the control algorithm, the fan should be switched off.

- **Cooling Occupied Active:** allows to enable or disable the function enforcing fan operation;
  - Available settings: true (enaled), false (disabled);

Cooling Occupied Active Enabled

If the fan works in the Auto mode (the **Fan Mode** slot is set to 4), the space is occupied (the **Occupancy Status** slot is set to 1), and the FCU_FanControl component works in the cooling temperature mode (the **Heating Cooling** slot is set to false), the fan will always be switched on, even if the value of the **Setpoint** slot is higher than the value of the **Cv** slot, in which case, according to the control algorithm, the fan should be switched off.

Note: The way of calculating the current speed of the fan (for fans with binary inputs) or value of Fan Analog Out slot (for the fan with analog input), when Cooling/Heating Occupied Active function is enabled, is presented in the figures below:

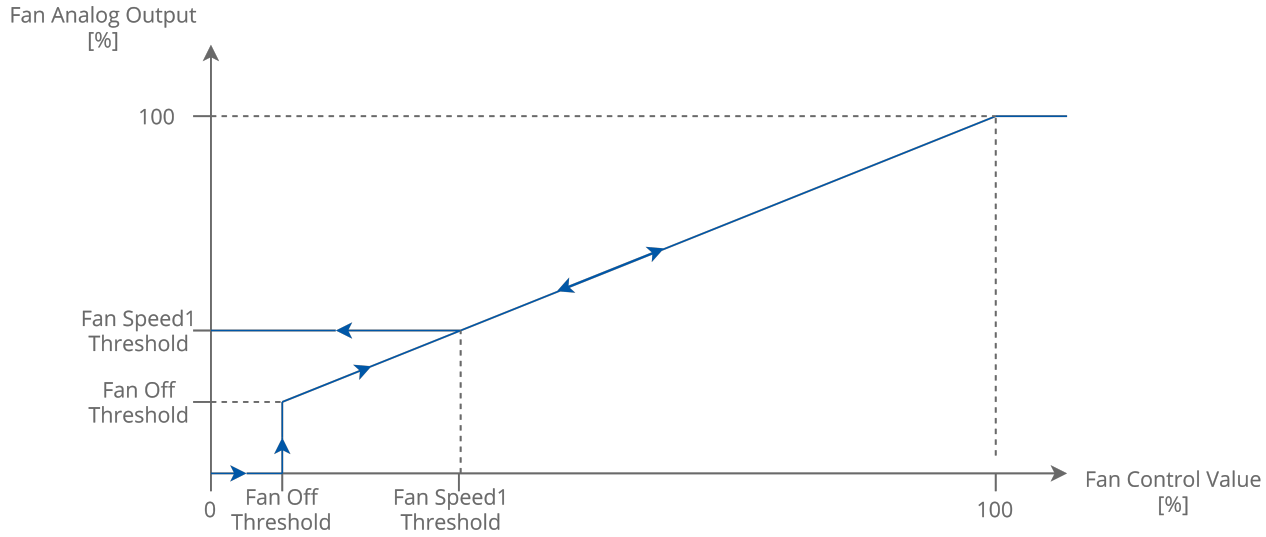


Figure 97. Heating/Cooling occupied active function for a fan with an analog input

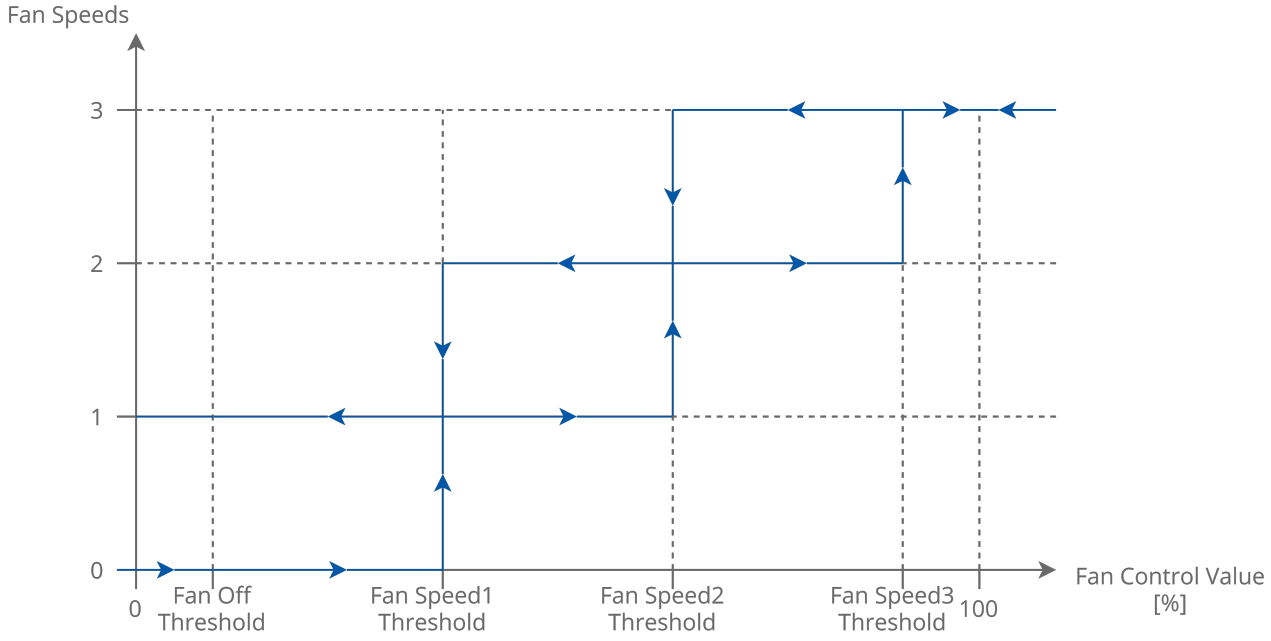


Figure 98. Heating/Cooling occupied active function for fans with binary inputs

- **Fan Delay Off:** sets the value of the delay-off time, expressed in seconds; each time, the value of the Fan Status ID slot is higher than 0 and the fan should be switched off, it remains working for the time equal to the value of this slot; after this time, the fan will switch off. If the slot is set to 0, the function is disabled;
- **Soft Start Time:** sets the time, in which the fan is working in the Soft Start mode, expressed in seconds. If the slot is set to 0, the function is disabled.
- **Soft Start Value:** sets the value for the Soft Start mode, expressed as percentage.

Note: If the Soft Start Value is lower than the Fan Speed 1 Threshold, the value will be taken from the Fan Speed 1 Threshold slot.

Note: The Soft Start function is dedicated to fans with analog input. If the fan start with small control value ramp lasts too long or is impossible, overheating of the driver or motor can occur. In this function, the fan start output value will be increased to the Soft Star Value for the time defined in the Fan Soft Start Time. If the time of the soft start is finished, the Fan Analog Out slot is set to the current value calculated by the algorithm of the component.

11 iSMA Building Kit

The iSMA Building kit has been created to simplify the creation of the applications for the blind/shutter control.

11.1 Sunblind

The Sunblind component allows to control the sunblind with one function block. It best suits simple projects, where only simple up/down manual or BMS control is required. There is a dedicated slot for monitoring the current position of the sunblind.

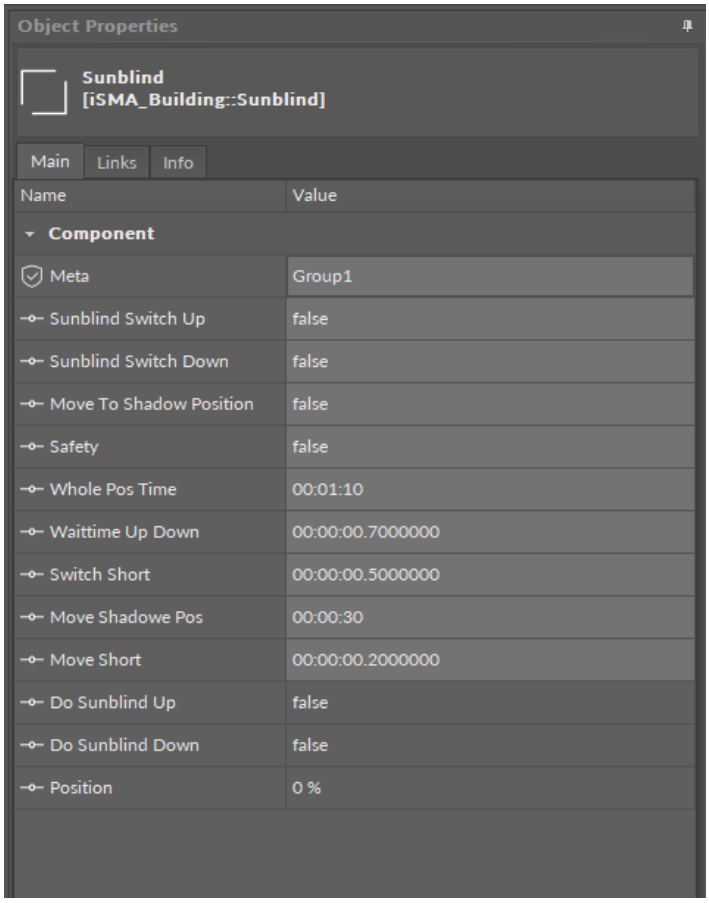


Figure 99. Sunblind component

The Sunblind component has the following configuration slots:

- **Sunblind Switch Up:** the Boolean input dedicated to move the sunblind up;
- **Sunblind Switch Down:** the Boolean input dedicated to move the sunblind down;
- **Move To Shadow Position:** the Boolean input dedicated to move the sunblind to the shadow position (protection against the sunlight);
- **Safety:** the Boolean input dedicated to move the Sunblind to complete up position (protection against the strong wind);
- **Whole Pos Time:** allows to set the time needed for the sunblind motor to completely open/close the sunblind (0-300 sec);
- **Waittime Up Down:** allows to set the delay between changing the direction of the sunblind. (0.6-3 sec);
- **Switch Short:** allows to set the time defining short press of the switch, which results in changing the slats position;
- **Move Shadow Pos:** allows to set the time needed for the sunblind motor to set the sunblind in the shadow position (0-300 sec);

- **Move Short:** allows to set the time defining the short moving of the sunblind in case if the impulse was shorter than the Switch Short value. If the motor is moving, such impulse results in stopping the sunblind motor;
- **Do Sunblind Up:** the Boolean output dedicated to move the sunblind up;
- **Do Sunblind Down:** the Boolean output dedicated to move the sunblind down;
- **Position:** allows to monitor the position of the sunblind (0-100%).

11.2 AdvanceSunblind

The AdvanceSunblind component has been created to extend the possibilities of the Sunblind component. It is suited best for projects, where control of the sunblind with the possibility to change the slats positioning is required. There are dedicated slots, which allow changing the operation mode of the sunblind, depending on the length of the impulse–Short Pulse/Middle /Long Pulse. The component also allows to define special function positions, such as Shadow, Safety, and Cleaning, which can be invoked by slots “Move to …”.

Figure 100. AdvanceSunblind component

The AdvanceSunblind component has the following configuration slots:

- **Status:** shows the component status:
- **Fault Cause:** shows the description of the fault:
  - Available information: None (the component is working properly), TooSmallShortPulse (the ScanPeriod value is twice bigger comparing to the ShortPulse), TooSmallLongPulse (the ScanPeriod value is twice bigger comparing to the LongPulse), ShortPulseIsGreaterThenLong (the ShortPulse value is bigger or equal to the LongPulse);
- **Sunblind Switch Up:** the Boolean input dedicated to move the sunblind up;
- **Sunblind Switch Down:** the Boolean input dedicated to move the sunblind down;
- **Move To Shadow Position:** the Boolean input dedicated to move the sunblind to the Shadow position (protection against the sunlight).

Note: The Move To Shadow Position slot has the lowest priority. If this function is in operation, it is still possible to control the sunblind by the Sunblind Switch Up and Sunblind Switch Down inputs.

- Move To Safety Position: the Boolean input dedicated to move the sunblind to the safety position (protection against the strong wind).

Note: The Move To Safety Position slot has the highest priority. Ifthis function is in operation, it is not possible to control the sunblind by the Sunblind Switch Up and Sunblind Switch Down inputs.

- Move To Cleaning Position: the Boolean input dedicated to move the sunblind to the cleaning position;

Note: If this function is in operation, it is not possible to control the sunblind by the Sunblind Switch Up and Sunblind Switch Down inputs.

- Short Pulse: allows to the time (ms) defining the length of the impulse, which is considered as a short impulse–such impulse causes the change of position of the slats as long as the button is pressed;
- Long Pulse: allows to the time (ms) defining the length of the impulse, which is considered as a long impulse–such impulse causes the change of position of the sunblind for as long as the button is pressed;
- Middle Pulse: allows to set the impulse, which is longer than the Short Pulse and shorter than the Long Pulse–such impulse causes the full opening/closing of the sunblind depending on the pressed button Up or Down;
- Runtime Up: allows to set the time of fullly opening of the sunblind (sec);
- Runtime Down: allows to set the time of fullly closing of the sunblind (sec);
- Up Synch Time: allows to set the additional time for moving up the sunblind, to make sure it is always fully open;
- Slats Opening Time: allows to set the time (ms) to completely change the position of the slats (from open to close, and otherwise);
- Shadow Position: allows to set the position (%) of the sunblind to make a shadow, by default 30%;
- Shadow Slats Position: allows to set the slots position [%] of the sunblind to make a shadow, by default 30%;
- Safety Position: allows to set the safety position of the sunblind (protection against the strong wind), by default 0%;
- Safety Slats Position: allows to set the the safety slats position of the sunblind (protection against the strong wind), by default 0%;
- Cleaning Position: allows to set the cleaning position of the sunblind, by default 100%;
- Cleaning Slats Position: allows to set the slats cleaning position of the sunblind, by default 100%;
- Do Sunblind Up: the Boolean output dedicated to move the sunblind up;
- Do Sunblind Down: the Boolean output dedicated to move the sunblind down;
- Position: allows monitoring the position of the sunblind (0-100 %);
- Slats Position: allows monitoring the position of the slats of the sunblind(0-100 %).

Note: Additionally, for the kit to operate correctly, it is required to set the Scan Period time in the app component 15-20 ms longer than the value in the Scan Time, for the logic to be responsive. According to these parameters, the position of the sunblind is calculated, so it will allow controlling the sunblind precisely.