iSMACONTROLLI

iSMA-B-AAC20

User Manual

Modbus

Table of Contents

1 Introduction

This manual contains information about Modbus protocol in the iSMA-B-AAC20 controller. The iSMA-B-AAC20 controller supports the following types of Modbus protocol:

- Modbus RTU/ASCII;
- Modbus TCP;
- Modbus TCP slave.
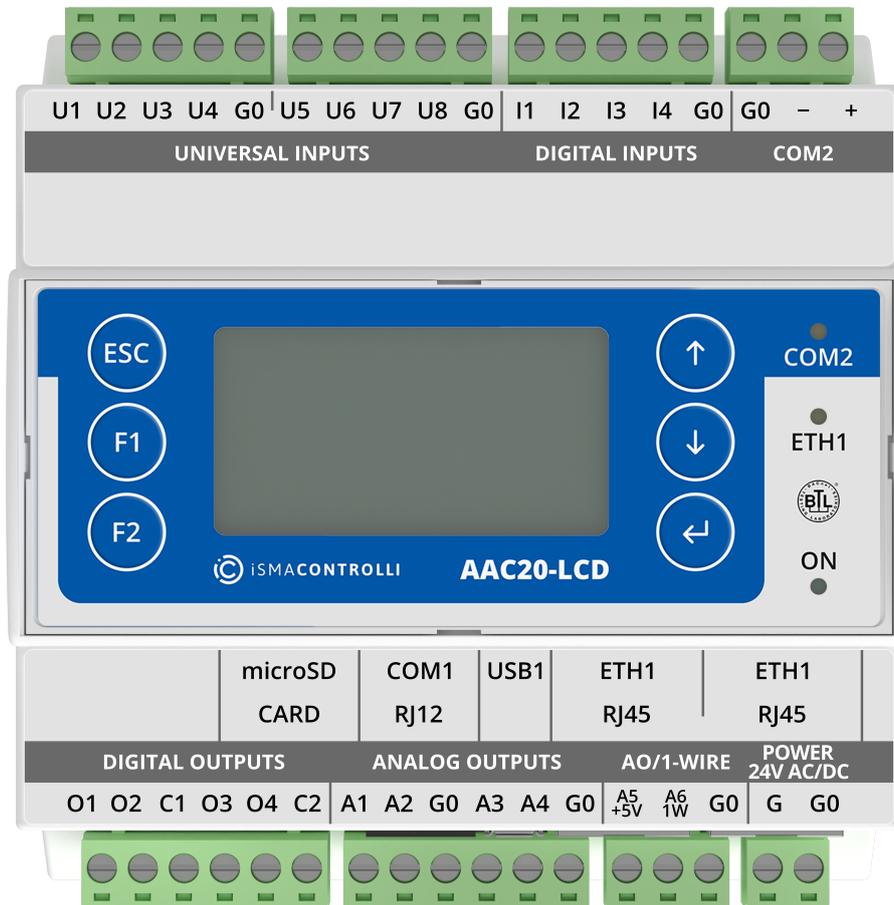
Figure 1. AAC20-LCD controller

1.1 Revision History

Rev.	Date	Description
1.7	17 Apr 2023	Added description of the iSMA Room Devices Modbus kit
1.6	28 Feb 2022	• Rebranded • Corrected license points information
1.5	27 Jan 2020	• Added RJ12 network description • Added Safety Rules, Technical Specification and Dimensions • Updated Modbus register table • Replaced environment of programming from Workplace to iSMA Tool • BTL compliance

Rev.	Date	Description
1.3	20 Apr 2017	• Added Modbus type of registers addressing in Modbus TCP Slave • Added information about accuracy increasing and Resistance Register multiply in PT1000 or NI1000 input working type
1.0	28 Aug 2015	First edition

Table 1. Revision history

2 Sedona Modbus

The Modbus protocol defines a message structure and format used in communication transactions. The Modbus devices communicate using a master-slave method, in which only the master device can initiate a communications transaction. There can be only one master device on a Modbus network. All other devices must be Modbus slaves.

> **WARNING!**
>
> Before programming the Modbus kits, please check if the latest kit version is used. The latest kits are available in the AAC20 Software Bundle on iSMA CONTROLLI support website: ismacontrolli.com

2.1 Modbus Registers

A Modbus device holds transient (real-time) data and persistent (configuration) data in the addressable registers. Here, the term "registers" implies all addressable data, but this is a loose interpretation. Using a Modbus nomenclature, all accessible data in a Modbus slave is contained in the following four available groups of data flags and registers (including the Modbus master access that is possible):

- **coil status:** (or simply "coils"): single-bit flags that represent the status of digital (Boolean) outputs of the slave, that is, On/Off output status; the Modbus master can both read from and write to coils;
- **input status:** (or simply "inputs"): single-bit flags that represent the status of digital (Boolean) inputs of the slave, that is, On/Off output status; the Modbus master can read (only) inputs;
- **input register:** 16-bit registers that store data collected from the field by the Modbus slave; the Modbus master can read (only) input registers;
- **holding register:** 16-bit registers that store general-purpose data in the Modbus slave the Modbus master can both read from and write to input registers.

2.2 Modbus Data Addresses

The Modbus device is not required to contain all four groups of data. For example, a metering device may contain only holding registers. However, for each data group implemented, a specific addressing scheme is used. The requests for data (made to a device) must specify a data address (and range) of interest.

Modbus data in a device is addressed as follows:

- coils: addressed at 00000, 0nnnn decimal, or "0x" addresses;
- inputs: addressed at 10000, 1nnnn decimal, or "1x" addresses;
- input register: addressed at 30000, 3nnnn decimal, or "3x" addresses;
- holding registers: addressed at 40000, 4nnnn decimal, or "4x" addresses.

Note: The data addressing (at least in decimal and hex formats) is zero-based, where the first instance of a data item, for example, coil 1, is addressed as item number 0. As another example, holding register 108 is addressed as 107 decimal or 006B hex.

3 Installation of iSMA Modbus Kits

There are 5 Modbus kits in the iSMA-B-AAC20 controller:

- **iSMA Modbus Async Network:** to serve the Modbus RS485 master port of the iSMA-B-AAC20 controller;
- **iSMA Modules:** extension of the Modbus Async Network to serve the iSMA MIX, Wireless, and MINI devices using Modbus Async protocol;
- **iSMA Modbus TCP Network:** to serve the Modbus TCP master of the iSMA-B-AAC20 device using IP connection;
- **iSMA Modbus TCP Slave Network:** to serve the Modbus TCP slave of the iSMA-B-AAC20 controller using IP connection;
- **iSMA Modbus RJ12 Network**: to serve the Modbus Async using RJ12 connection.

To install the Modbus kits, import the kits to the iSMA Tool (possibly as part of the package of various kits in a zip file). To do this, choose on the top bar menu Sedona -> Import Sedona Files.

After a successful import of the files, upload the files to your device using the Kit Manager Application.

> WARNING! Before programming the Modbus network, please check if the latest kit version is used. The latest kits are available on the iSMA CONTROLLI support website: ismacontrolli.com
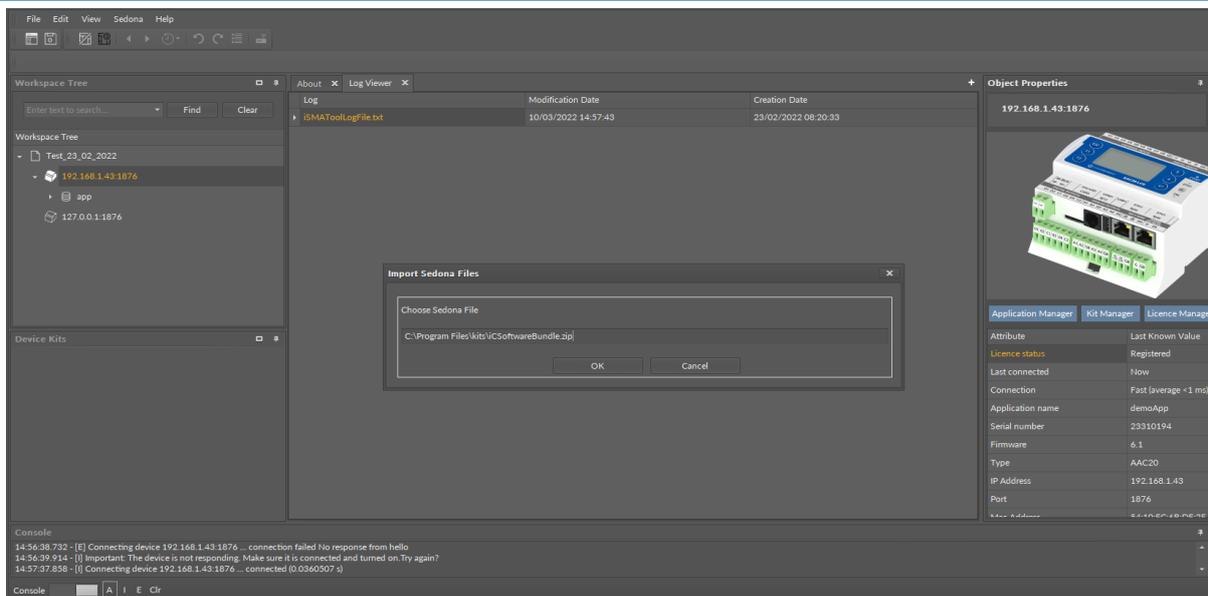
Figure 2. Importing newest Sedona kits to the iSMA Tool

3.1 Available Sockets

The iSMA-B-AAC20 controller has 16 sockets for Modbus network. 3 out of 16 sockets are permanently occupied for:

- Modbus Server;
- SOX;
- web server.

Consequently, there are 13 sockets left to use in the device, for example, the Modbus TCP network can communicate with 13 devices with different IP addresses and connect them to application (adding more devices automatically forces them into the fault status). Also, adding any of the iSMA Weather or iSMA MailService kits occupies 1 socket per each kit (which becomes apparent after adding the kit and its components, saving the application, and rebooting the controller). The iSMA MailService kit can occupy more sockets if the mail service is configured for one account on one host–each next host occupies next sockets.

4 ModbusAsyncNetwork

This section provides a collection of procedures to use the iSMA-B-AAC20 Modbus drivers to build networks of devices with the Modbus points. The iSMA-B-AAC20 controller has one RS485 port, which can be used as a Modbus RTU / ASCII master.

The Modbus Async Network kit consists of 4 types of components:

- Modbus Network;
- Modbus Device;
- Modbus Data Points;
- Modbus Points folder.

4.1 Modbus Async License and Limitation

In the standard license there are available 500 data points, and this number cannot be expanded. The number of available points is shown in the ModbusAsyncNetwork component in the Free Points slot.

> WARNING! Each device and data point is counted as one point. For example, to read 7 data points from 15 devices: Points number = 15 *(1 + 7) = 105.

4.2 Modbus Async Network Component

The ModbusAsyncNetwork is the main component, which is responsible for servicing the RS485 physical port. The component must be placed under the Drivers folder. The Modbus Network sets parameters such as communication baud rate and data format, testing, etc., and keeps statistics.
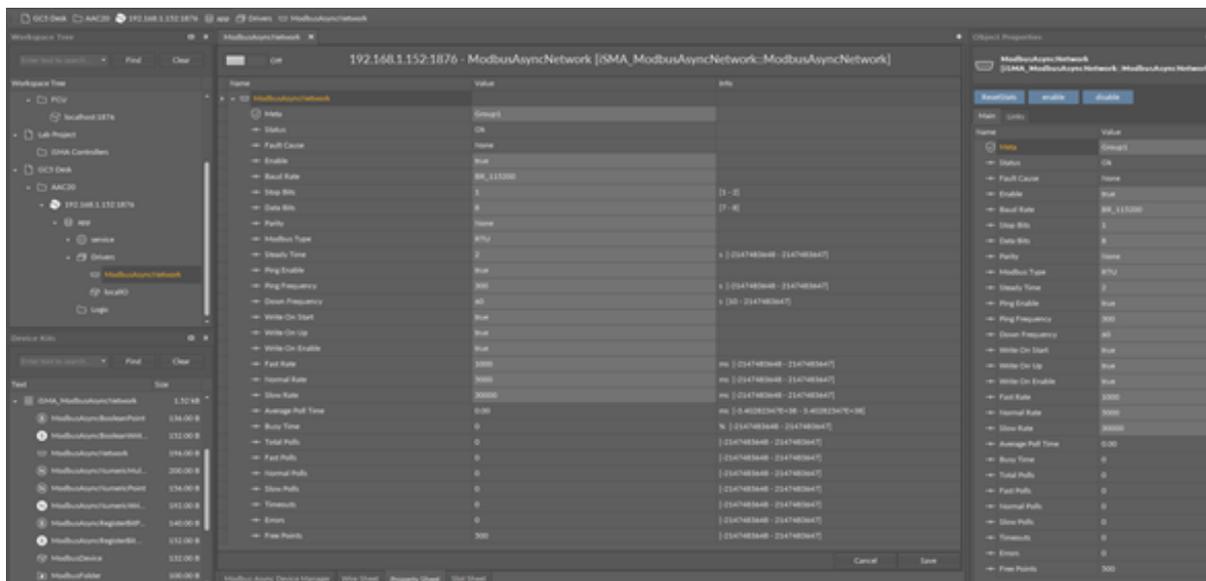
Figure 3. ModbusAsyncNetwork component

The ModbusAsyncNetwork component has the following slots:

- **Status**: Network's status;
    - Available states: OK (network is working properly), Disabled (network is disabled, the Enable slot is in false), OK some device/point down (error in the device or points);
- **Fault Cause**: fault cause description;

- **Enable**: this option switches on or switches off the Modbus Network;
    - true (network enabled), false (network disabled);
- **Steady Time**: network start-up delay time after a power-up or reset;
- **Baud Rate**: the Modbus RS485 port baud rate;
    - Available options: 2400, 4800, 9600, 19200, 38400, 57600, 115200 bps;
- **Stop Bits**: stop bit definition;
    - Available options: 1-bit, 2-bits;
- **Data Bits**: data bits definition;
    - Available options: 7-bits or 8-bits;
- **Parity**: parity bit definition;
    - Available options: None, Odd, Even, Always1, Always0;
- **Modbus Type**: Modbus type definition;
    - Available options: RTU or ASCII;
- **Ping Enable**: enables testing of the device's connection;
- **Ping Frequency**: time between testing messages to check the device's connection;
- **Down Frequency**: time between testing messages for devices or points, which have got the down status;
- **Write On Start**: executes a write action in device writable components in the Modbus network after a reset or power-up;
- **Write On Up**: executes a write action in device writable components in the Modbus network after restoring of the connection with Modbus device;
- **Write On Enable**: executes a write action in device „Writable" components in the Modbus network after enabling the device;
- **Fast Rate**: time between messages in the fast mode poll frequency;
- **Normal Rate**: time between messages in the normal mode poll frequency;
- **Slow Rate**: time between messages in the slow mode poll frequency;
- **Average Poll Time**: average time for sending/receiving of one message;
- **Busy Time**: percentage of Modbus network usage;
- **Total Polls**: total number of messages;
- **Fast Polls**: number of messages sent in the fast mode;
- **Normal Polls**: number of messages sent in the normal mode;
- **Slow Polls**: number of messages sent in the slow mode;
- **Timeouts**: number of lost messages, the difference between sent and received messages;
- **Errors**: number of error messages (for example, with the wrong CRC);
- **Free points**: number of available physical points in Modbus network.

The ModbusAsyncNetwork component has the following actions available under the right-click or in the Object Properties window:

- **Reset Stats**: resets network's statistics and starts counting from the beginning;
- **Enable/Disable**: switching the Modbus network on/off.

4.3 ModbusAsyncDevice

The ModbusAsyncDevice is a component, which is responsible for servicing a physical device connected to the Modbus network. The AAC20 device acts as a Modbus master to all other Modbus devices on the attached RS485 port. Each device is represented by a Modbus device and has a unique Modbus address (1 to 247) as well as other Modbus config data and starting addresses for Modbus data items (coils, inputs, input registers, holding registers). The component has a Ping action available under the right-click, which

sends a test message to the device to check the device status. Each ModbusAsyncDevice has a 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 use as the device status test (meaning Monitor ping requests). Ping requests are generated at the network-level by the configurable network monitor (ModbusNetwork -> Ping Enabled). If enabled, a network's monitor periodically pings (queries) this address. If any response from the device is received, including an exception response, this is considered a proof of communication, and the Modbus client device is no longer considered down if it was previously marked so.
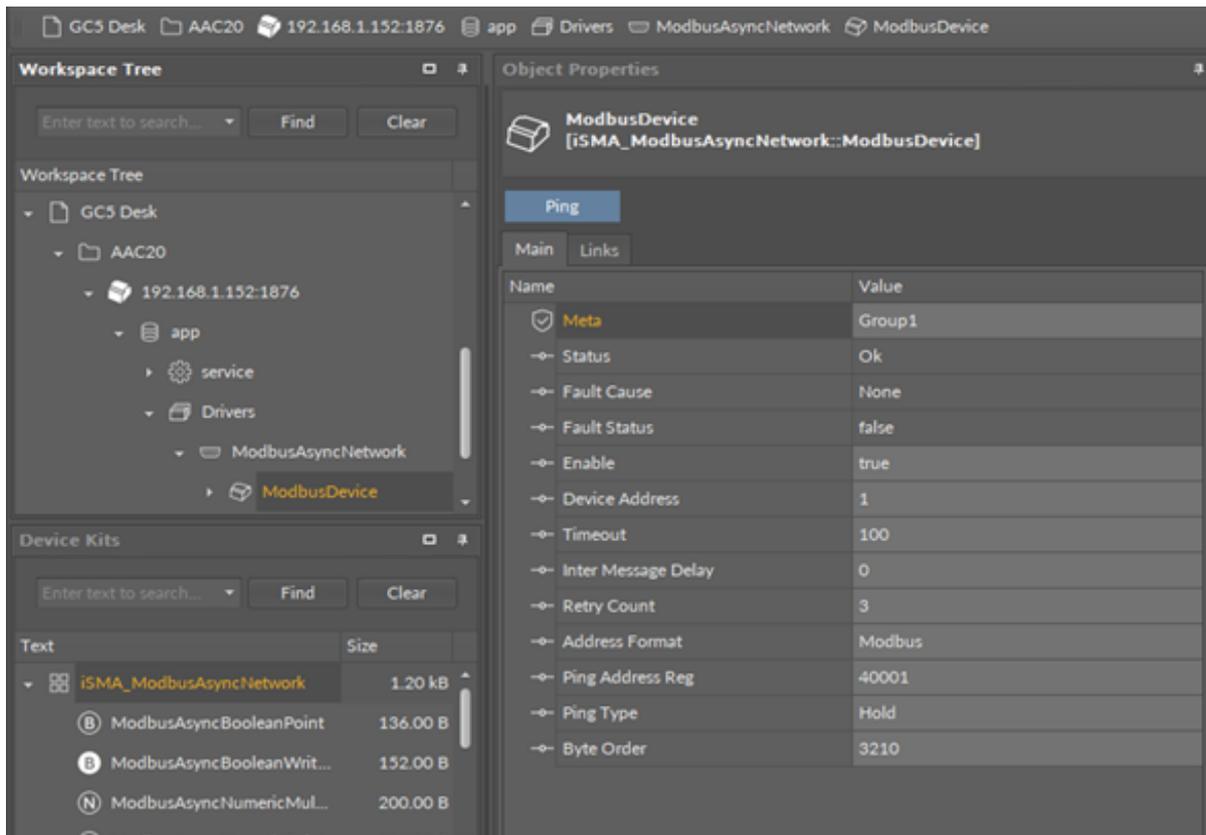

Figure 4. ModbusAsyncDevice component

The ModbusAsyncDevice component has the following slots:

- **Status**: Device's actual status (read-only);
    - Available options: OK (device is working properly), Disable (device is disabled, the Enable slot is in false), Down (device is not available), Ok, some points down/error (error in points reading), Network disabled (Modbus network is disabled);
- **Fault Cause**: fault cause description;
- **Fault Status**: device error status (true: device communication error);
- **Enable**: enables/disables the device;
- **Device Address**: Modbus device physical address (0: network broadcast address, 1-248 addressing range);
- **Timeout**: max. device response time from the device request;
- **Inter Message Delay**: time between messages sent to the device;
- **Retry Count**: max. number of error messages (CRC error, lost messages);
- **Address Format**: Modbus address format (Modbus, decimal);
- **Ping Address Reg**: input or Holding type register's number, which will be read for device connection test;
- **Ping Type**: tested register type: Input/Holding;

• **Byte Order:** byte reading order, for 32-bit: 3210 (Big endian), 1032 (Little endian).

4.4 Modbus Data Points

In the Modbus protocol each device has an implemented Modbus table. Sedona has 7 components to read/write data from this table:

• Boolean Point: reads Boolean values (Modbus command 0x02);
• Boolean Writable: reads/writes Boolean values (read: Modbus command 0x02, write: Modbus command 0x05);
• Numeric Point: reads numeric values (Modbus commands: 0x03 for reading holding registers, 0x04 for reading input registers);
• Numeric Writable: reads/writes numeric values (Modbus commands: 0x03 and 0x04 for reading, 0x06 for writing 16-bits Int, SInt values, 0x10 for writing 32-bits Long, SLong, Float values);
• Numeric Multi Point: reads up to eight 16-bits registers (Modbus commands 0x03 and 0x04);
• RegisterBitPoint: reads Boolean values from a specified register in the device (Modbus command 0x02);
• RegisterBitWritable: reads/writes Boolean values from/to a specified register (read: Modbus command 0x02, write: Modbus command 0x05).

4.4.1 ModbusAsyncBooleanPoint

The ModbusAsyncBooleanPoint is a component, which is responsible for reading Boolean values from the device. The component has a Read action available under the right-click, which forces the reading of the point.
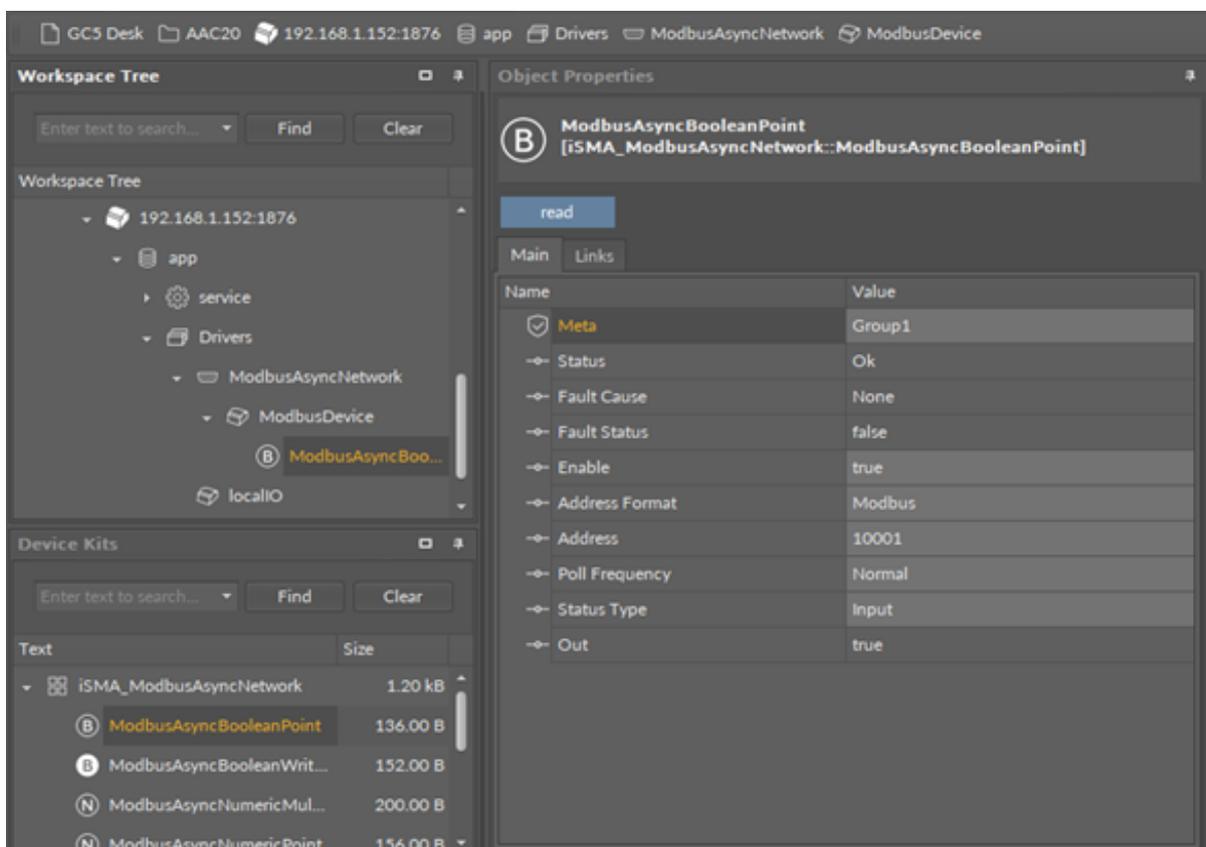

Figure 5. ModbusAsyncBoolean

Slots

The ModbusAsyncBooleanPoint component has the following slots:

- **Status**: point's status;
    - Available states: OK (point is working properly), Disabled (point is disabled, the Enable slot is in false), Down/Timeout (point is not available), Device Down (device is not available), Wrong address format (incorrect address format according to the address format setting slot), Device disabled (device is disabled), Network disabled (Modbus network is disabled);
- **Fault Cause**: fault cause description;
- **Fault Status**: point error status;
    - Available options: true (point read error), false;
- **Enable**: enables/disables the point;
    - Available options: true (point enabled), false (point disabled);
- **Address Format:** register address format;
    - Available options: Modbus, decimal;
- **Address**: register address;
- **Poll Frequency**: reading poll frequency;
    - Available options: fast, normal, slow;
- **Status Type**: type of reading register;
    - Available options: input: 0x02, coil: 0x01;
- **Out**: current value of the read register.

4.4.2 ModbusAsyncBooleanWritable

The ModbusAsyncBooleanWritable is a component which is responsible for sending and reading Boolean values from the device.
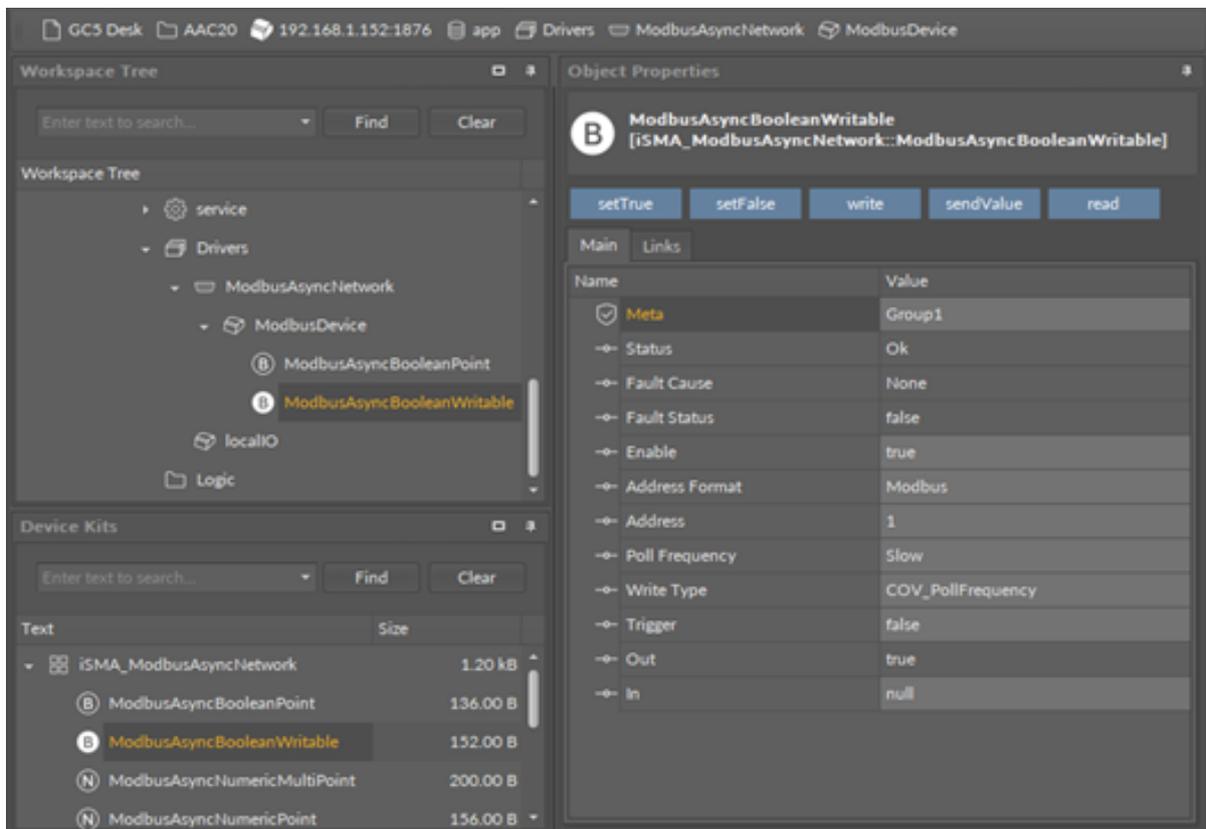

Figure 6. ModbusAsyncBooleanWritable component

Slots

The ModbusAsyncBooleanWritable component has the following slots:

- **Status**: point's status;
    - Available states: OK (point is working properly), Disabled (point is disabled, the Enable slot is in false), Down/Timeout (point is not available), Device Down (device is not available), Wrong address format (incorrect address format according to the address format setting slot), Device disabled (device is disabled), Network disabled (Modbus network is disabled).
- **Fault Cause**: fault cause description;
- **Fault Status**: point error status;
    - Available options: true (point read/write error), false;
- **Enable**: enables/disables the point
    - Available options: true (point enabled), false (point disabled),
- **Address Format:** register address format
    - Available options: Modbus, decimal;
- **Address**: register address;
- **Poll Frequency**: reading poll frequency;
    - Available options: fast, normal, slow;
- **Write Type**: writing mode;
    - Available options: COV (only on input change), COV_PollFrequrency (on input change and periodically), PollFrequrency (only periodically), COV_LinkSet (link-back forward triggered by COV);
- **Trigger**: forcefully send the value (on rising edge), regardless of the current poll mode;
- **Out**: output slot, the current value of read/write register;
- **In:** input slot.

Actions

The ModbusAsyncBooleanWritable component has the following actions available under the right-click:

- **Set True/Set False:** writes a value to the In slot and sends it to the device (not active when slot In has a connected link);
- **Write**: sends a value from the In slot to the device;
- **Read**: reads a value from the device and sends to the Out slot.

4.4.3 ModbusAsyncNumericPoint

The ModbusAsyncNumericPoint is a component, which is responsible for reading numeric values from the device. The component has a Read action available under the right-click, which forces the reading of the point.
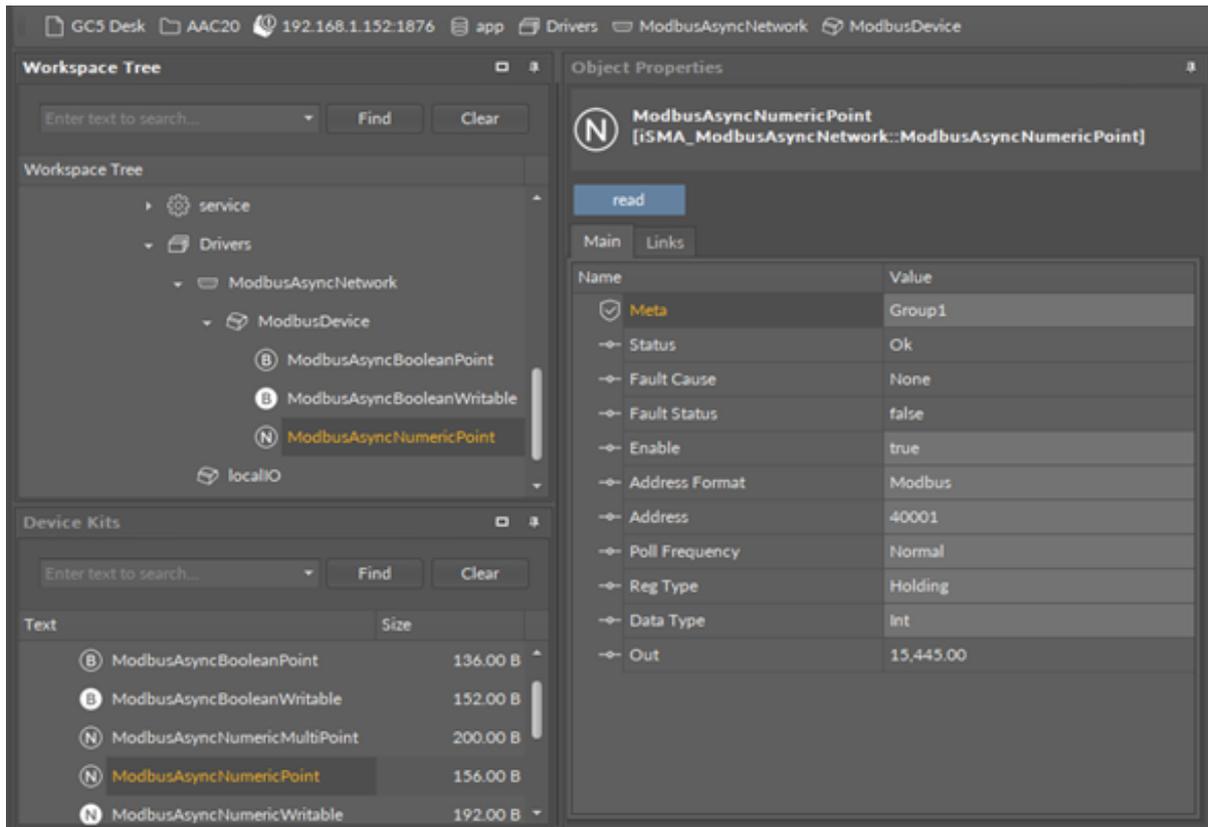
Figure 7. ModbusAsyncNumericPoint

Slots

The ModbusAsyncNumericPoint component has the following slots:

- **Status**: point's status;
    - Available states: OK (point is working properly), Disabled (point is disabled, the Enable slot is false), Down/Timeout (point is not available), Device Down (device is not available), Wrong address format (incorrect address format according to address format setting slot), Device disabled (device is disabled), Network disabled (Modbus network is disabled);
- **Fault Cause**: fault cause description;
- **Fault Status**: point error status;
    - Available options: true (point read error), false;
- **Enable**: enables/disables the point;
    - Available options: true (point enabled), false (point disabled);
- **Address Format:** register address format;
    - Available options: Modbus, decimal;
- **Address**: register address;
- **Poll Frequency**: reading poll frequency;
    - Available options: fast, normal, slow;
- **Reg Type**: type of reading register;
    - Available options: input: 0x04, holding: 0x03;
- **Data Type**: reading register data type;
    - Available options: Int: 16-bits, Long: 32-bits, Float: 32-bits float-point, SInt: 16-bits with sign, Slong: 32-bits with sign;
- **Out**: current value of the read register.

4.4.4 ModbusAsyncNumericWritable

The ModbusAsyncNumericWritable is a component, which is responsible for sending and reading numeric values from the device.
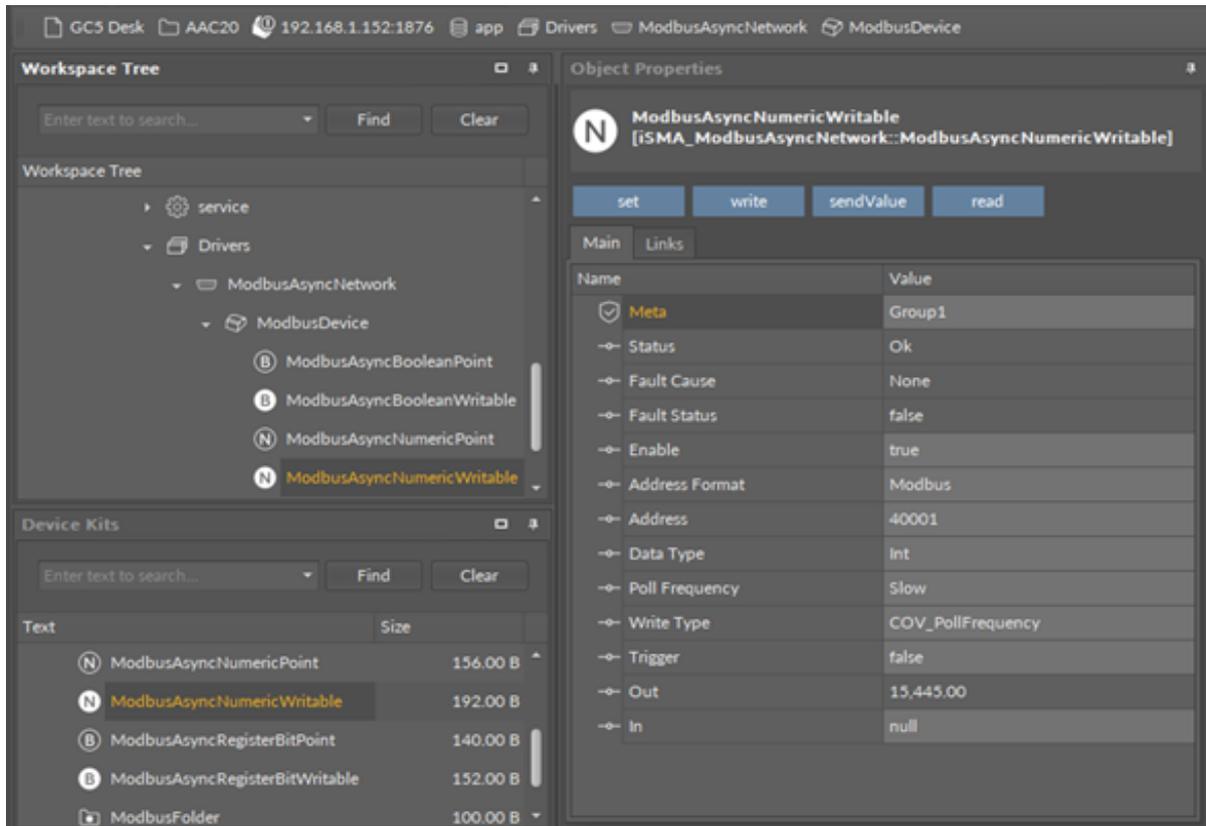
Figure 8. ModbusAsyncNumericWritable component

Slots

The ModbusAsyncNumericWritable component has the following slots:

- **Status**: point's status, available states:
  - Available states: OK (point is working properly), Disabled (point is disabled, the Enable slot is in false), Down/Timeout (point is not available), Device Down (device is not available), Wrong address format (incorrect address format according to the address format setting slot), Device disabled (device is disabled), Network disabled (Modbus network is disabled).
- **Fault Cause**: fault cause description;
- **Fault Status**: point error status;
  - Available options: true (point read/write error), false;
- **Address Format:** register address format;
  - Available options: Modbus, decimal;
- **Address**: register address;
- **Data Type**: read/write register data type;
  - Available options: Int: 16-bits, Long: 32-bits, Float: 32-bits float-point, SInt: 16-bits with sign, Slong: 32-bits with sign, IntF16- use Function 16, SIntF16: use Function 16 (Function 16: Modbus function for sending one register);
- **Poll Frequency**: reading poll frequency;
  - Available options: fast, normal, slow;
- **Write Type**: writing mode;

  - ◦ Available options: COV - only on input change, COV_PollFrequency: on input change and periodically, PollFrequency - only periodically, COV_LinkSet (Link-back forward triggered by COV);
- **Trigger**: forcefully send the value (on rising edge), regardless of the current poll mode,
- **Out**: output slot, the current value of the device register,
- **In:** input slot.

Actions

The ModbusAsyncNumericWritable component has the following actions available under the right mouse button:

- **Set**: writes a value to the In slot and sends it to the device;
- **Write**: sends a value from the In slot to the device;
- **Read**: reads a value from the device and sends it to the Out slot.

4.4.5 ModbusAsyncNumericMultiPoint

The ModbusAsyncNumericMultiPoint is a component, which is responsible for reading up to eight 16-bits registers from the device in one message. The component uses 0x03 or 0x04 Modbus commands. The component has a Read action available under the right-click, which forces the reading of the point.
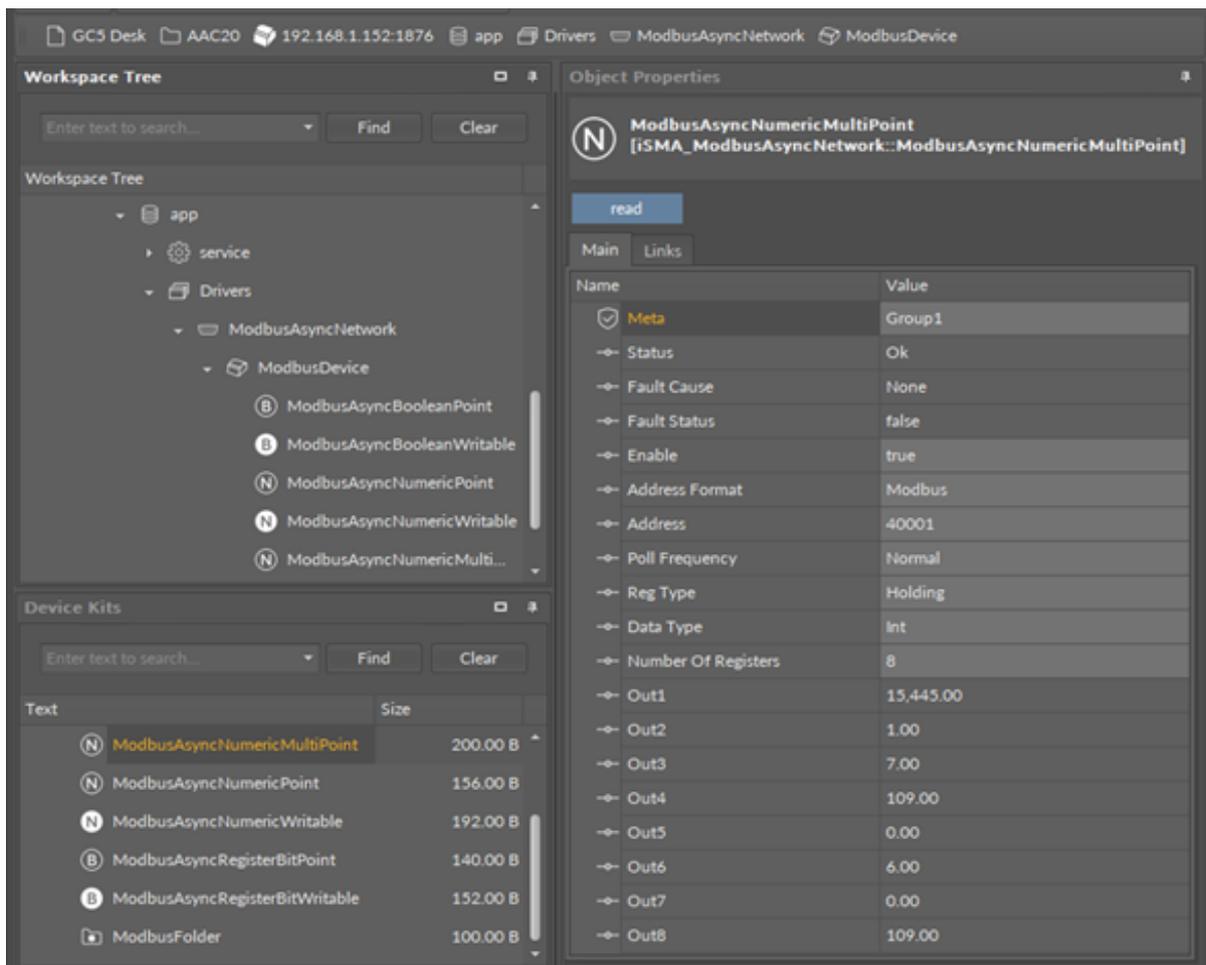
Figure 9. ModbusAsyncNumericMultiPoint component

Slots

The ModbusAsyncNumericMultipoint component has the following slots:

- **Status**: point's status;
    - Available states: OK (point is working properly), Disabled (point is disabled, the Enable slot is false), Down/Timeout (point is not available), Device Down (device is not available), Wrong address format (incorrect address format according to address format setting slot), Device disabled (device is disabled), Network disabled (Modbus network is disabled);
- **Fault Cause**: fault cause description;
- **Fault Status**: point error status;
    - Available options: true (point read error), false;
- **Enable**: enables/disables the point;
    - Available options: true (point enabled,) false (point disabled);
- **Address Format:** Register address format;
    - Available options: Modbus, decimal;
- **Address**: register address;
- **Poll Frequency**: reading poll frequency;
    - Available options: fast, normal, slow;
- **Reg Type**: type of reading register;
    - Available options: input - 0x04, holding - 0x03;
- **Data Type**: read data type: Int (unsigned values), Sint (signed values);
- **Number Of Registers**: number of registers read in one message;
- **Out**: current value of the read register.

4.4.6 ModbusAsyncRegisterBitPoint

The ModbusAsyncRegisterBitPoint component is responsible for reading Boolean values from a bit in a specified register in the device. The component has to be placed under the ModbusAsyncDevice component.
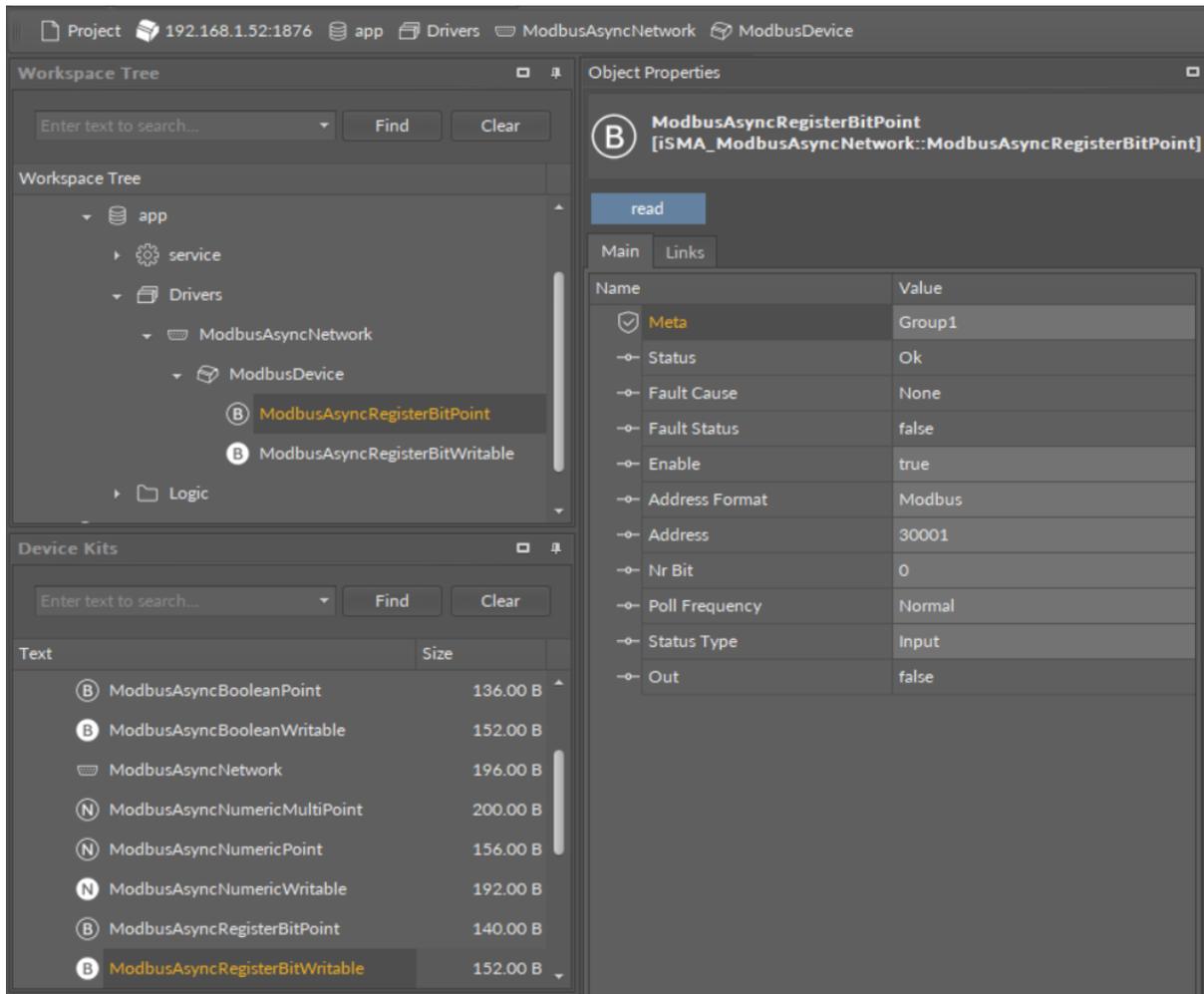
Figure 10. ModbusAsyncRegisterBitPoint component

Slots

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.

Action

The ModbusAsyncRegisterBitPoint component offers the following action:

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

4.4.7 ModbusAsyncRegisterBitWritable

The ModbusAsyncRegisterBitWritable component is responsible for sending to and reading Boolean values from a bit in a specified register in the device. The component has to be placed under the ModbusAsyncDevice component.
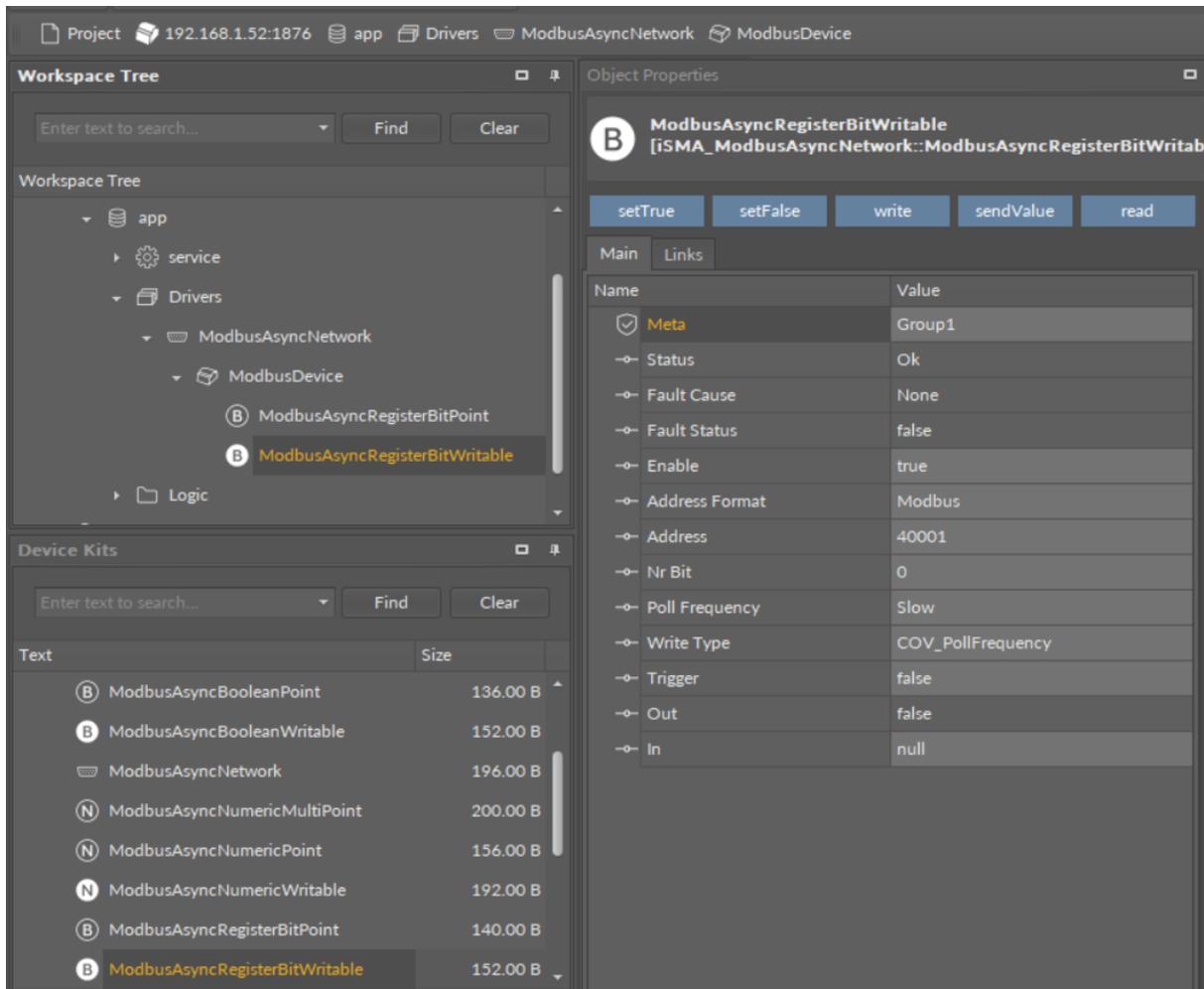
Figure 11. ModbusAsyncRegisterBitWritable component

Slots

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_PollFequrency: on the In slot change and periodically, PollFequrency: only periodically, COV_LinkSet : only on the In slot change using the "reverse following the link" function);
- **Trigger:** allows to trigger the remote enforcement of sending (on rising edge);

- **Out:** the current value of reading bit;
- **In:** the input slot.

Action

The ModbusAsyncRegisterBitWritable component offers the following actions:

- **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.

4.5 ModbusFolder

The ModbusFolder is a component which groups and organizes the Modbus points components. The ModbusFolder has the Description slot, where up to 32 characters may be inserted.
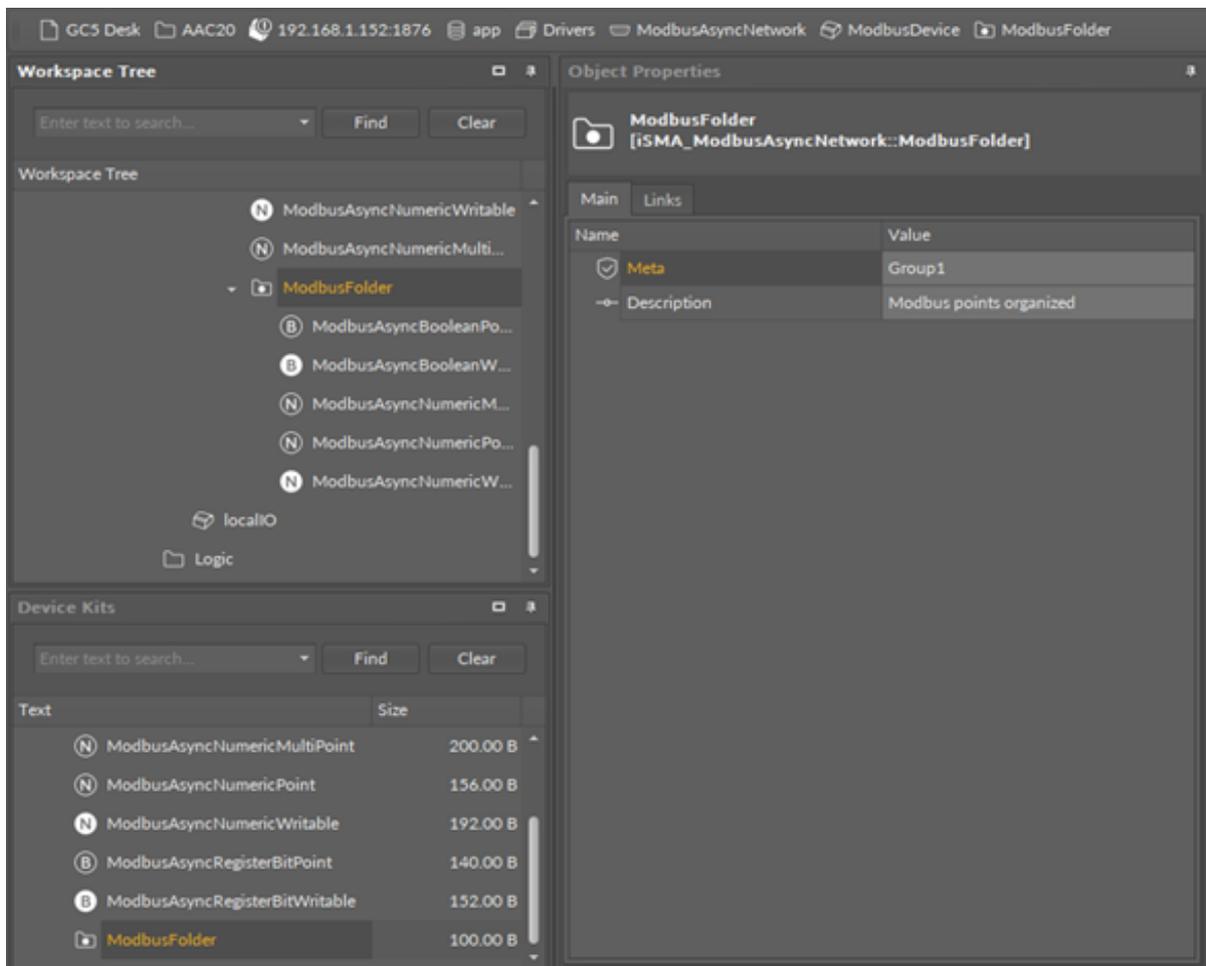
Figure 12. ModbusFolder component

5 ModbusTCPNetwork

The iSMA-B-AAC20 controller has an implemented Modbus TCP protocol. It means that, configured as a Master device, it can read/write data to Slave devices using the IP connection.

5.1 Modbus TCP License and Limitation

In the standard license there are available 500 data points, and this number cannot be expanded. The number of available points is shown in the Modbus TcpNetwork component in the Free Points slot.

> WARNING! Each device and data point is counted as one point. For example, to read 4 data points from 2 devices: Points number = 2 *(1 + 4) = 10.

5.2 ModbusTCPNetwork Component

The ModbusTCPNetwork is the main component, which is responsible for servicing an IP communication to slave devices. The component must be placed under the Drivers folder. The ModbusTCPNetwork sets parameters such as communication baud rate and data format, testing, etc., and keeps statistics.
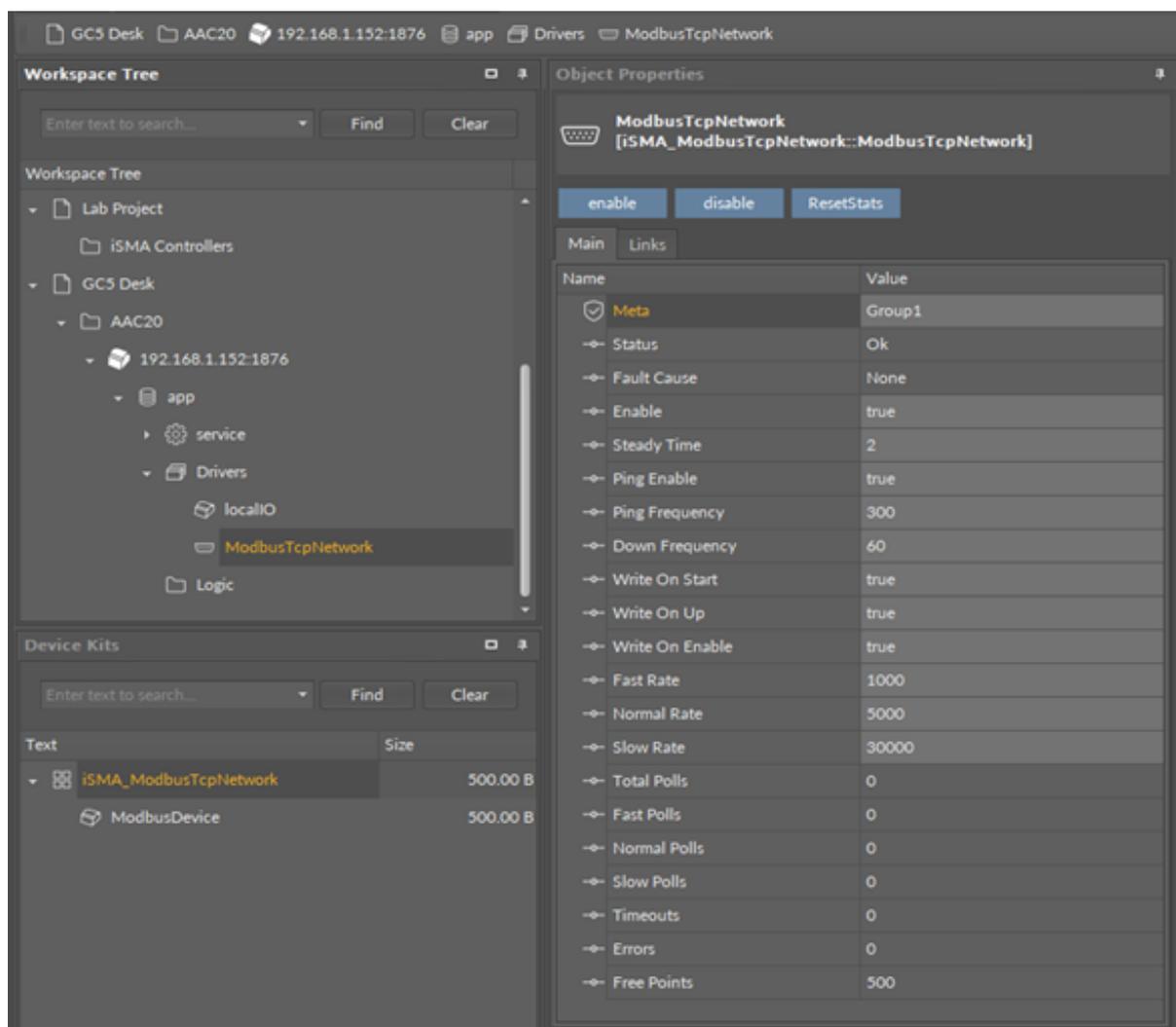

Figure 13. ModbusTCPNetwork component

The ModbusTCPNetwork component has the following slots:

- **Status**: Network's status;
  - ◦ Available states: OK (network is working properly), Disabled (network is disabled, the Enable slot is in false), OK some device/point down (error in the device or points);
- **Fault Cause**: fault cause description;
- **Enable**: option to switch on or switch off Modbus network;
  - ◦ Available options: true (network enabled), false (network disabled);
- **Steady Time**: network's delay time to start-up after a power-up or reset;
- **Ping Enable**: enables the device's connection testing function;
- **Ping Frequency**: time between testing messages to check device connection;
- **Down Frequency**: time between testing messages for devices or points which have got status down;
- **Write On Start**: executes a write action in device writable components in the Modbus network after a reset or power-up;
- **Write On Up**: executes a write action in device writable components in the Modbus network after restoring the connection with the Modbus device;
- **Write On Enable**: executes a write action in device writable components in the Modbus network after enabling the device;
- **Fast Rate**: time between messages in the fast mode poll frequency;
- **Normal Rate**: time between messages in the normal mode poll frequency;
- **Slow Rate**: time between messages in the slow mode poll frequency;
- **Average Poll Time**: average time for sending/receiving of one message
- **Busy Time**: percentage of Modbus network usage;
- **Total Polls**: total number of messages;
- **Fast Polls**: number of messages sent in the fast mode;
- **Normal Polls**: number of messages sent in the normal mode;
- **Slow Polls**: number of messages sent in the slow mode;
- **Timeouts**: number of lost messages, the difference between sent and received messages;
- **Errors**: number of error messages (for example, with the wrong CRC);
- **Free points**: number of available physical points in the Modbus network.

The ModbusTCPNetwork component has the following actions available under the right-click or in the Object Properties window:

- **Reset Stats**: resets network's statistics and starts counting from the beginning;
- **Enable/Disable**: switching the Modbus network on/off.

5.3 ModbusTCPDevice

The ModbusDevice is a component which is responsible for servicing physical Modbus TCP slave devices. Each Modbus device is represented by an IP address, port number (default for Modbus 502), and Device address (1 to 247).

The component has a Ping action available under the right-click, which sends a test message to the device to check the device status. Each ModbusDevice has a "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 use as the device status test (meaning "Monitor" ping requests). Ping requests are generated at the network-level by the configurable network monitor (ModbusNetwork

-> Ping Enabled). When enabled, a network's monitor periodically pings (queries) this address. If any response id received from the device, including an exception response, this is considered a proof of communication, and the Modbus client device is no longer considered "down" if it had been previously marked "down".
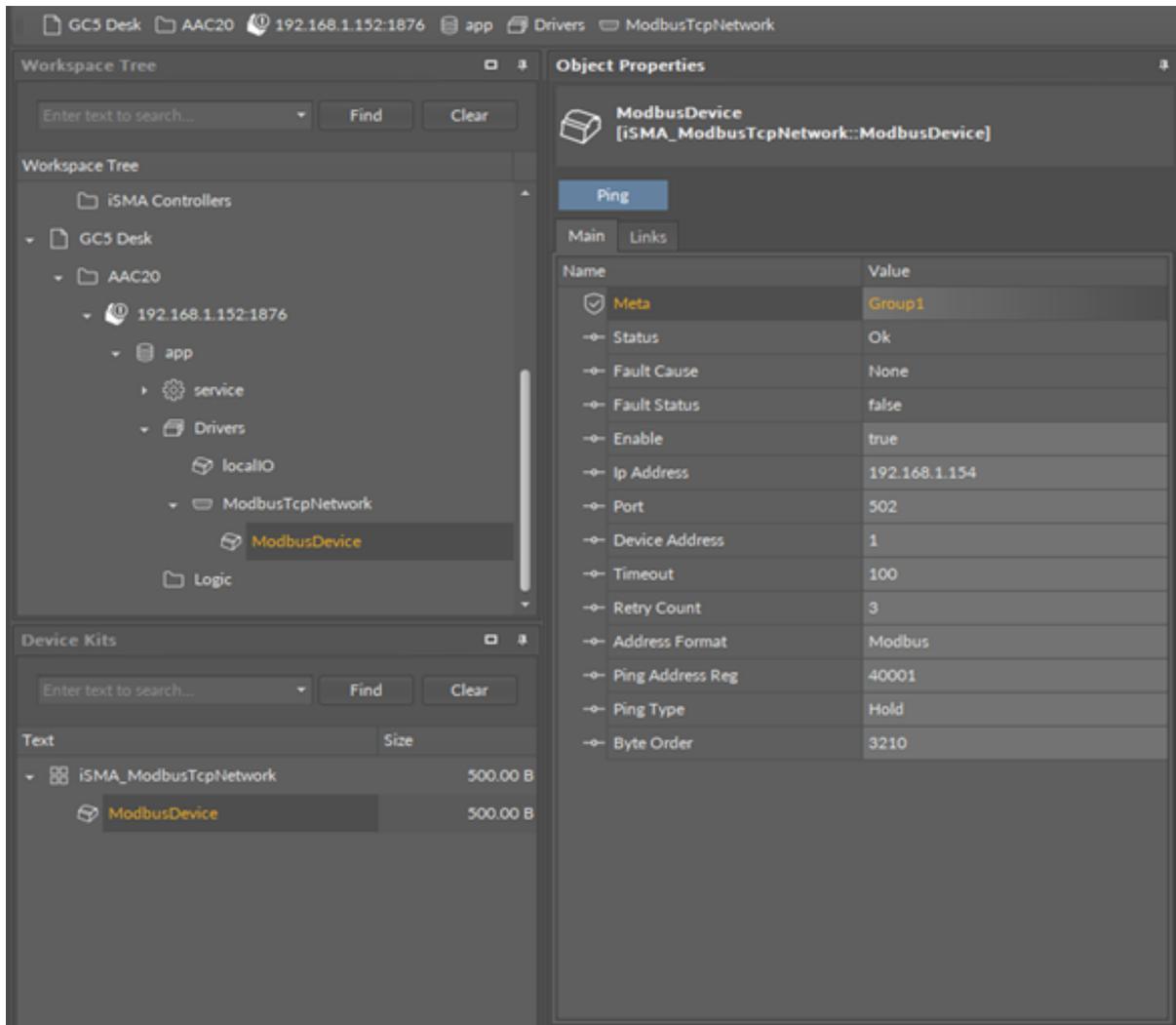
Figure 14. ModbusTCPDevice component

The ModbusTCPDevice component has the following slots:

- **Status**: Device's actual status (read-only);
  - Available states: OK (device is working properly), Disable (device is disabled, the Enable slot is in false), Down (device is not available), Ok, some points down/error (error in points reading), Network disabled (Modbus network is disabled);
- **Fault Cause**: fault cause description;
- **Fault Status**: device error status;
  - Available states: true (device communication error), false;
- **Enable**: enables/disables the device;
- **IP Address**: slave device (gateway) IP address;
- **Port**: slave device (gateway) Modbus TCP port number (default 502);
- **Device Address**: Modbus device address (0 - broadcast, 1-248 addressing range);
- **Timeout**: max. device response time from the device request;
- **Inter Message Delay**: time between messages sent to the device;
- **Retry Count**: max. number of error messages (CRC error, lost messages);
- **Address Format**: Modbus address format (Modbus, decimal);

- **Ping Address Reg**: any register (Input/Holding) number for device connection test;
- **Ping Type**: tested register type: Input/Holding;
- **Byte Order:** byte reading order , for32-bit: 3210 (Big endian), 1032 (Little endian).

5.4 Modbus TCP Data Points

In the Modbus protocol each device has an implemented Modbus table. Sedona has 5 components to read/write data from this table:

- Boolean Point: reads Boolean values (Modbus command 0x02);
- Boolean Writable: reads/writes Boolean values (read: Modbus command 0x02, write: Modbus command 0x05);
- Numeric Point: reads numeric values (Modbus commands: 0x03 for reading holding registers, 0x04 for reading input registers);
- Numeric Writable: reads/writes numeric values (Modbus commands: 0x03 and 0x04 for reading, 0x06 for writing 16-bits Int, SInt values, 0x10 for writing 32-bits Long, SLong, Float values);
- Numeric Multi Point: reads up to eight 16-bits registers (Modbus commands 0x03 and 0x04);
- (V1.7) RegisterBitPoint: reads Boolean values from a specified register in the device (Modbus command 0x02);
- (V1.7) RegisterBitWritable: reads/writes Boolean values from/to a specified register (read: Modbus command 0x02, write: Modbus command 0x05).

5.4.1 ModbusBooleanPoint

The ModbusBooleanPoint is a component, which is responsible for reading Boolean values from the device. The component has a Read action available under the right-click, which forces the reading of the point.
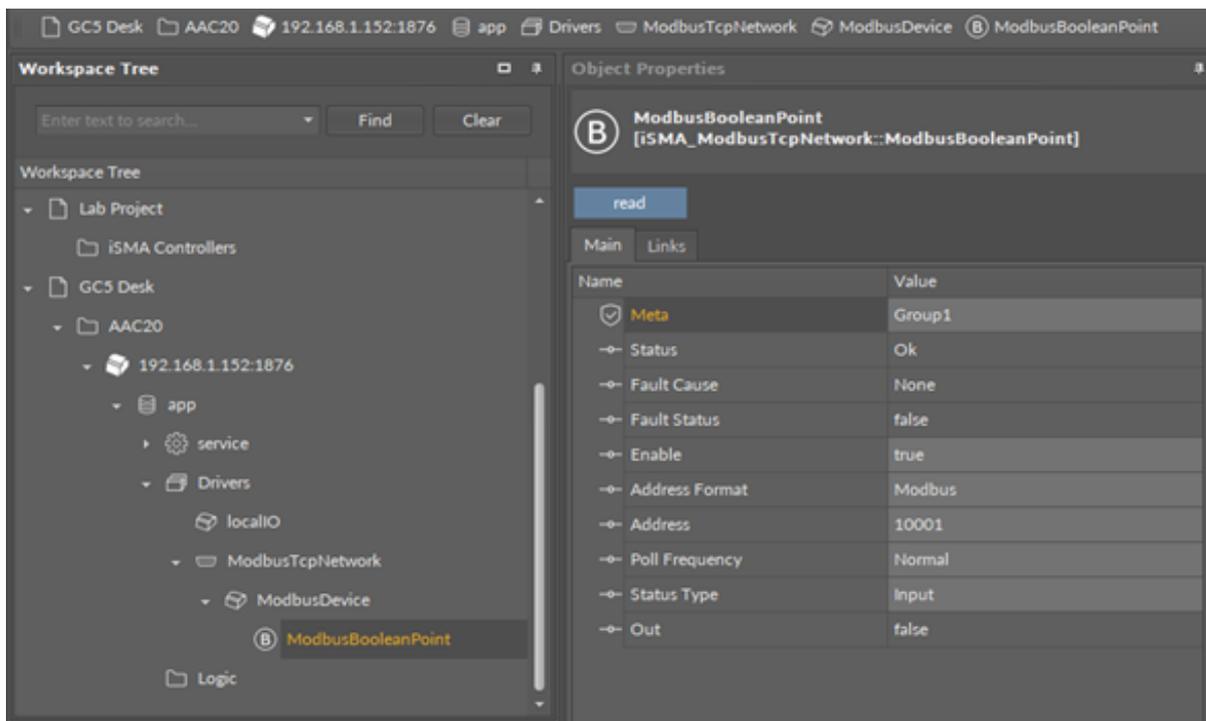
Figure 15. ModbusBooleanPoint component

Slots

The ModbusBooleanPoint component has the following slots:

- **Status**: point's status;
    - Available states: OK (point is working properly), Disabled (point is disabled, the Enable slot is in false), Down/Timeout (point is not available), Device Down (device is not available), Wrong address format (incorrect address format according to the address format setting slot), Device disabled (device is disabled), Network disabled (Modbus network is disabled);
- **Fault Cause**: fault cause description;
- **Fault Status**: point error status;
    - Available options: true (point read error), false;
- **Enable**: enables/disables the point;
    - Available options: true (point enabled), false (point disabled);
- **Address Format:** register address format;
    - Available options: Modbus, decimal;
- **Address**: register address;
- **Poll Frequency**: reading poll frequency;
    - Available options: fast, normal, slow;
- **Status Type**: type of reading register;
    - Available options: input: 0x02, coil: 0x01;
- **Out**: current value of the read register.

Action

- **Read:** forces reading of a point.

5.4.2 ModbusBooleanWritable

The ModbusBooleanWritable is a component, which is responsible for sending and reading Boolean values from the device.
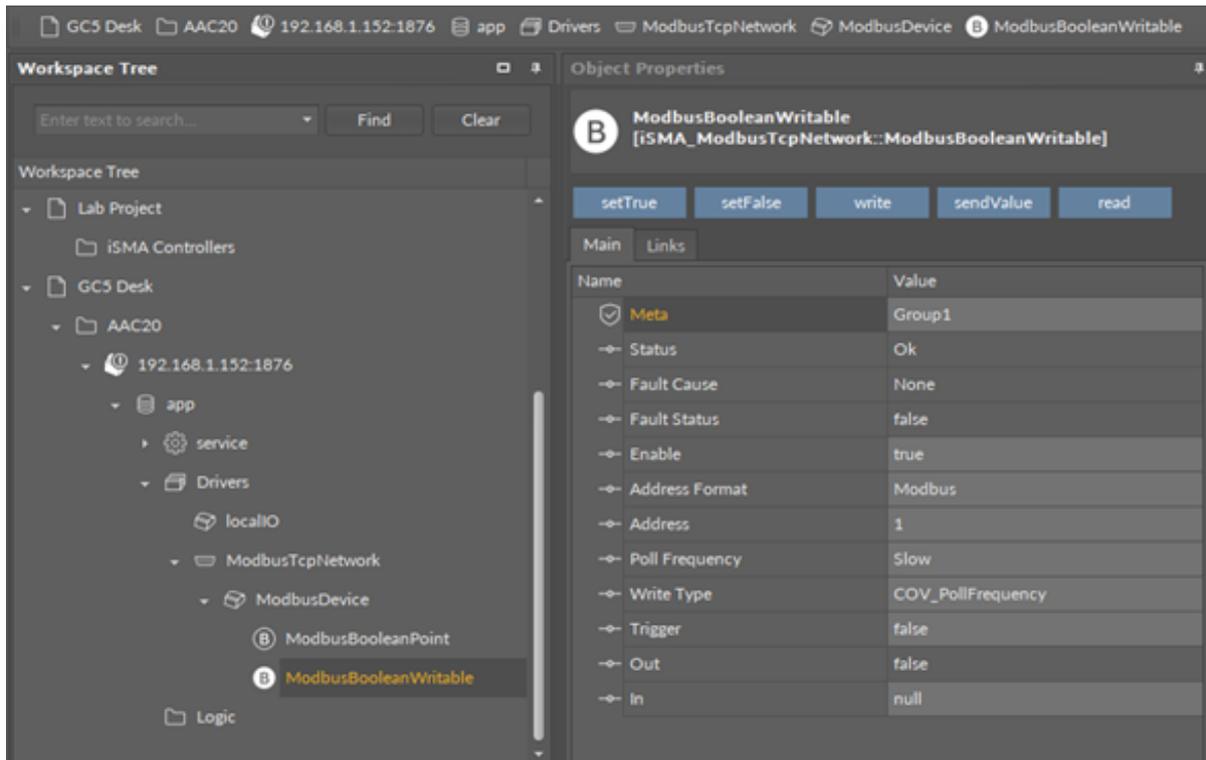
Figure 16. ModbusBooleanWritable component

Slots

The ModbusBooleanWritable component has the following slots:

- **Status**: point's status;
    - Available states: OK (point is working properly), Disabled (point is disabled, the Enable slot is in false), Down/Timeout (point is not available), Device Down (device is not available), Wrong address format (incorrect address format according to the address format setting slot), Device disabled (device is disabled), Network disabled (Modbus network is disabled).
- **Fault Cause**: fault cause description;
- **Fault Status**: point error status;
    - Available options: true (point read/write error), false;
- **Enable**: enables/disables the point
    - Available options: true (point enabled), false (point disabled),
- **Address Format:** register address format
    - Available options: Modbus, decimal;
- **Address**: register address;
- **Poll Frequency**: reading poll frequency;
    - Available options: fast, normal, slow;
- **Write Type**: writing mode;
    - Available options: COV (only on input change), COV_PollFrequrency (on input change and periodically), PollFrequrency (only periodically), COV_LinkSet (link-back forward triggered by COV);
- **Trigger**: forcefully send the value (on rising edge), regardless of the current poll mode;
- **Out**: output slot, the current value of read/write register;
- **In:** input slot.

Actions

The ModbusTCPBooleanWritable component has the following actions available under the right-click:

- **Set True/Set False:** writes a value to the In slot and sends it to the device (not active when slot In has a connected link);
- **Write**: sends a value from the In slot to the device;
- **sendValue:**
- **Read**: reads a value from the device and sends to the Out slot.

5.4.3 ModbusNumericPoint

The ModbusNumericPoint is a component, which is responsible for reading numeric values from the device. The component has a Read action available under the right-click, which forces the reading of the point.
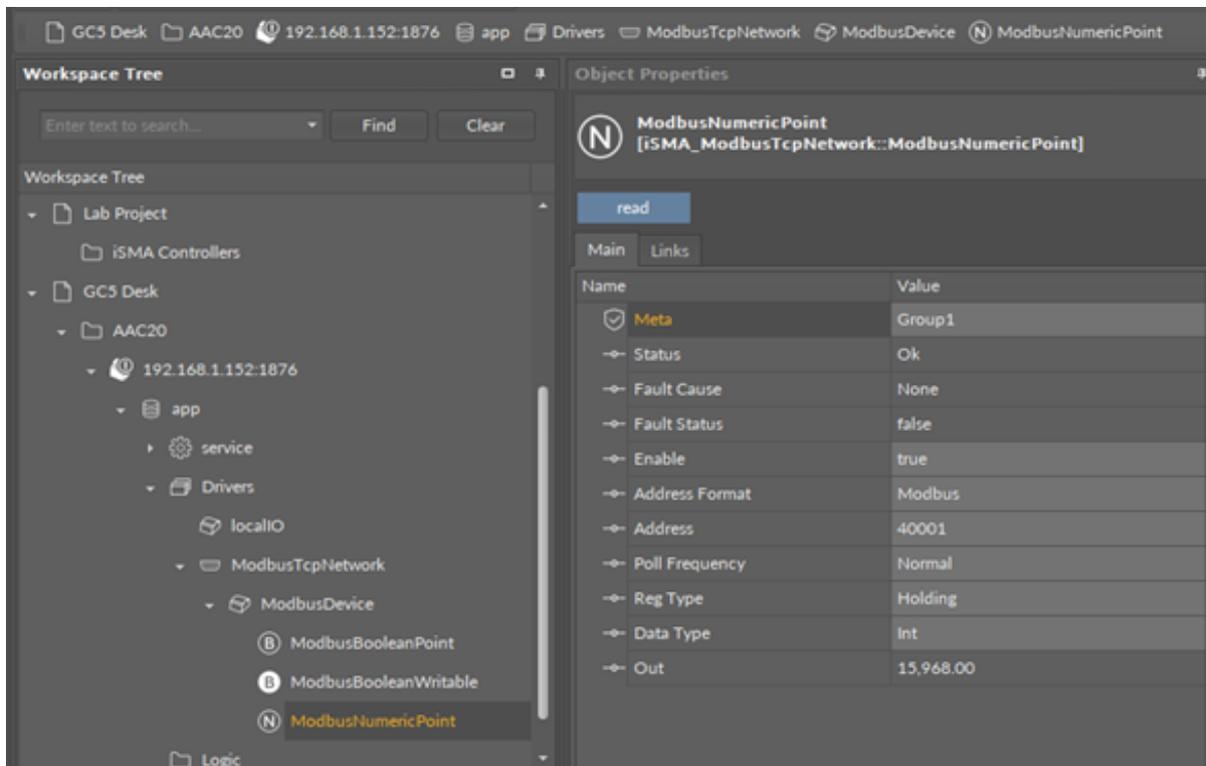
Figure 17. ModbusNumericPoint component

Slots

The ModbusNumericPoint component has the following slots:

- **Status**: point's status;
  - ○ Available states: OK (point is working properly), Disabled (point is disabled, the Enable slot is false), Down/Timeout (point is not available), Device Down (device is not available), Wrong address format (incorrect address format according to address format setting slot), Device disabled (device is disabled), Network disabled (Modbus network is disabled);
- **Fault Cause**: fault cause description;
- **Fault Status**: point error status;
  - ○ Available options: true (point read error), false;
- **Enable**: enables/disables the point;

- ◦ Available options: true (point enabled), false (point disabled);
- **Address Format**: register address format;
  - ◦ Available options: Modbus, decimal;
- **Address**: register address;
- **Poll Frequency**: reading poll frequency;
  - ◦ Available options: sast, normal, slow;
- **Reg Type**: type of reading register;
  - ◦ Available options: input: 0x04, holding: 0x03;
- **Data Type**: reading register data type;
  - ◦ Available options: Int: 16-bits, Long: 32-bits, Float: 32-bits float-point, SInt: 16-bits with sign, Slong: 32-bits with sign;
- **Out**: current value of the read register.

Action

- **Read:** forces reading of a point.

5.4.4 ModbusNumericWritable

The ModbusNumericWritable is a component, which is responsible for sending and reading numeric values from the device.
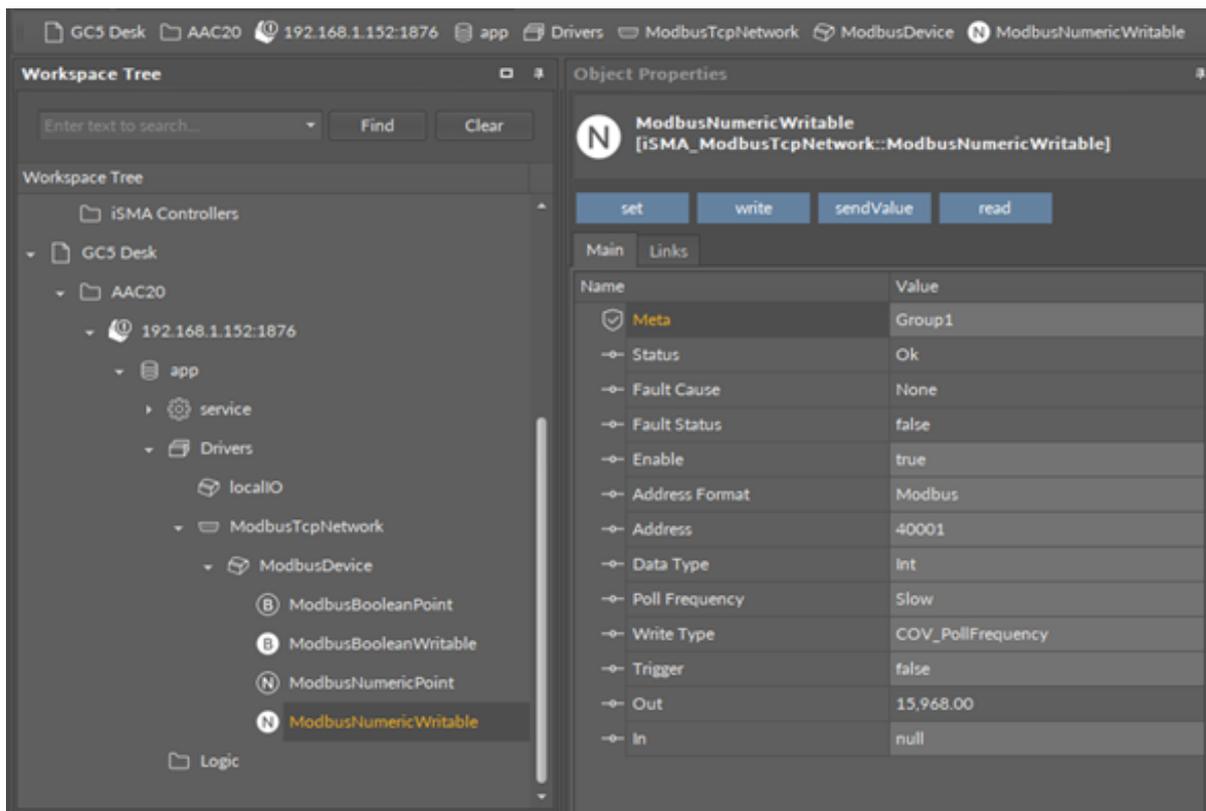

Figure 18. ModbusNumericWritable

Slots

The ModbusNumericWritable component has the following slots:

- **Status**: point's status, available states:
- ◦ Available states: OK (point is working properly), Disabled (point is disabled, the Enable slot is in false), Down/Timeout (point is not available), Device Down (device

is not available), Wrong address format (incorrect address format according to the address format setting slot), Device disabled (device is disabled), Network disabled (Modbus network is disabled).

- **Fault Cause**: fault cause description;
- **Fault Status**: point error status;
- Available options: true (point read/write error), false;
- **Address Format**: register address format;
  - Available options: Modbus, decimal;
- **Address**: register address;
- **Data Type**: read/write register data type;
  - Available options: Int: 16-bits, Long: 32-bits, Float: 32-bits float-point, SInt: 16-bits with sign, Slong: 32-bits with sign, IntF16- use Function 16, SIntF16: use Function 16 (Function 16: Modbus function for sending one register);
- **Poll Frequency**: reading poll frequency;
  - Available options: fast, normal, slow;
- **Write Type**: writing mode;
  - Available options: COV - only on input change, COV_PollFrequency: on input change and periodically, PollFrequency - only periodically, COV_LinkSet (Link-back forward triggered by COV);
- **Trigger**: forcefully send the value (on rising edge), regardless of the current poll mode,
- **Out**: output slot, the current value of the device register,
- **In:** input slot.

Actions

The ModbusNumericWritable component has the following actions available under the right mouse button:

- **Set**: writes a value to the In slot and sends it to the device;
- **Write**: sends a value from the In slot to the device;
- **sendValue:**
- **Read**: reads a value from the device and sends it to the Out slot.

5.4.5 ModbusNumericMultiPoint

The ModbusNumericMultiPoint is a component, which is responsible for reading up to 8 registers from the device in one message. The component uses 0x03 and 0x04 Modbus commands. The component has a Read action available under the right-click, which forces the reading of the point.
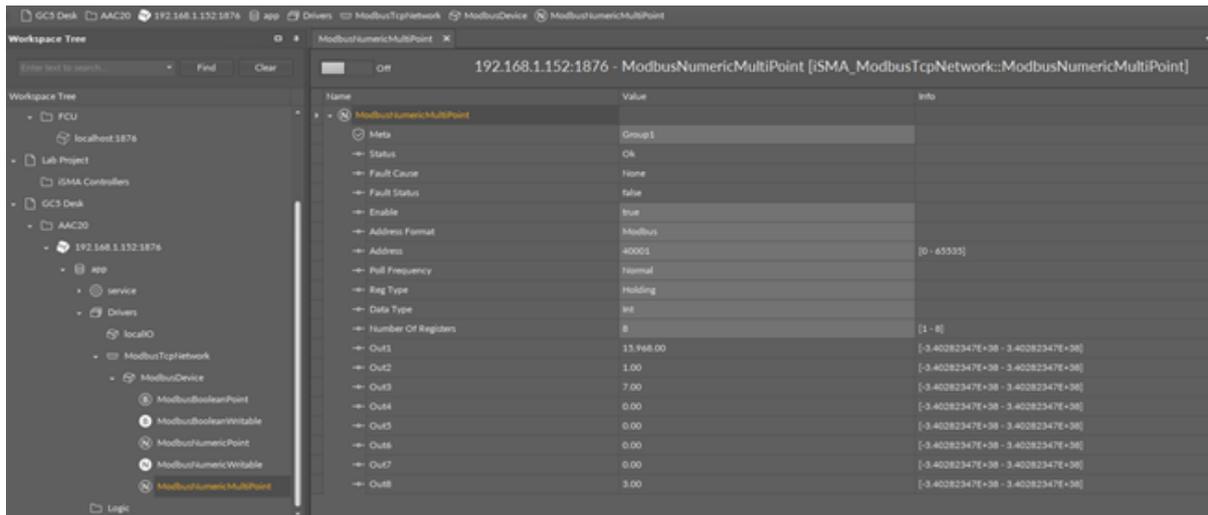
Figure 19. ModbusNumericMultiPoint

Slots

The ModbusNumericMultipoint component has the following slots:

- **Status**: point's status;
-
  - Available states: OK (point is working properly), Disabled (point is disabled, the Enable slot is false), Down/Timeout (point is not available), Device Down (device is not available), Wrong address format (incorrect address format according to address format setting slot), Device disabled (device is disabled), Network disabled (Modbus network is disabled);
- **Fault Cause**: fault cause description;
- **Fault Status**: point error status;
  - Available options: true (point read error), false;
- **Enable**: enables/disables the point;
  - Available options: true (point enabled,) false (point disabled);
- **Address Format:** Register address format;
  - Available options: Modbus, decimal;
- **Address**: register address;
- **Poll Frequency**: reading poll frequency;
  - Available options: fast, normal, slow;
- **Reg Type**: type of reading register;
  - Available options: input - 0x04, holding - 0x03;
- **Data Type**: read data type: Int (unsigned values), Sint (signed values);
- **Number Of Registers**: number of registers read in one message;
- **Out**: current value of the read register.

5.4.6 RegisterBitPoint

The RegisterBitPoint component is responsible for reading Boolean values from a bit in a specified register in the device. The component has to be placed under the ModbusDevice component in the ModbusTCPNetwork.
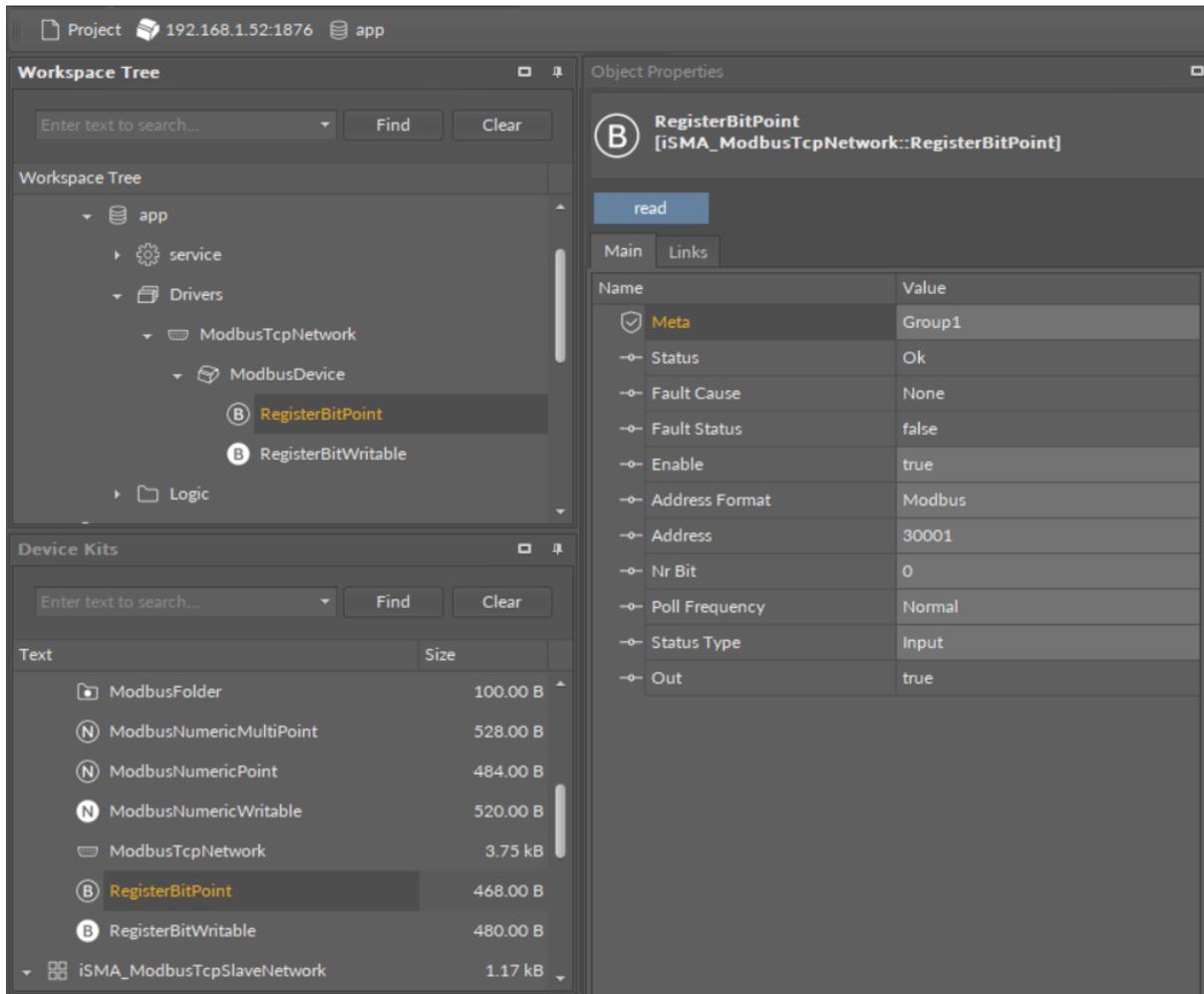
Figure 20. RegisterBitPoint

Slots

The RegisterBitPoint 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.

Action

The RegisterBitPoint component offers the following action:

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

5.4.7 RegisterBitWritable

The RegisterBitWritable component is responsible for sending to and reading Boolean values from a bit in a specified register in the device. The component has to be placed under the ModbusDevice component.
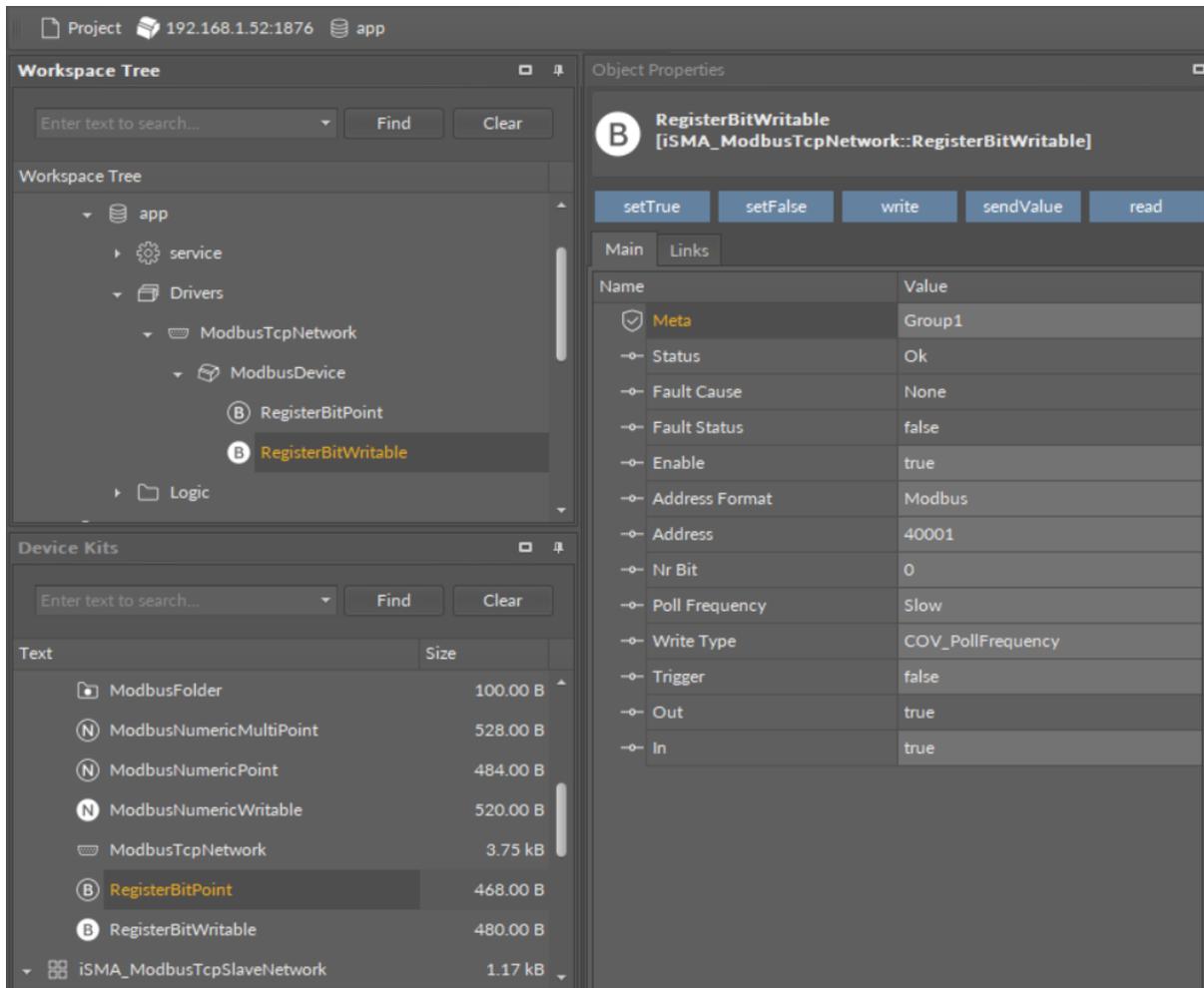
Figure 21. RegisterBitWritable

Slots

The RegisterBitWritable 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_PollFequrency: on the In slot change and periodically, PollFequrency: only periodically, COV_LinkSet : only on the In slot change using the "reverse following the link" function);
· **Trigger:** allows to trigger the remote enforcement of sending (on rising edge);
· **Out:** the current value of reading bit;

- **In:** the input slot.

Action

The RegisterBitWritable component offers the following actions:

- **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.

5.5 ModbusFolder

The ModbusFolder is a component which groups and organizes the Modbus points components. The ModbusFolder has a Description slot where up to 32 characters may be inserted.
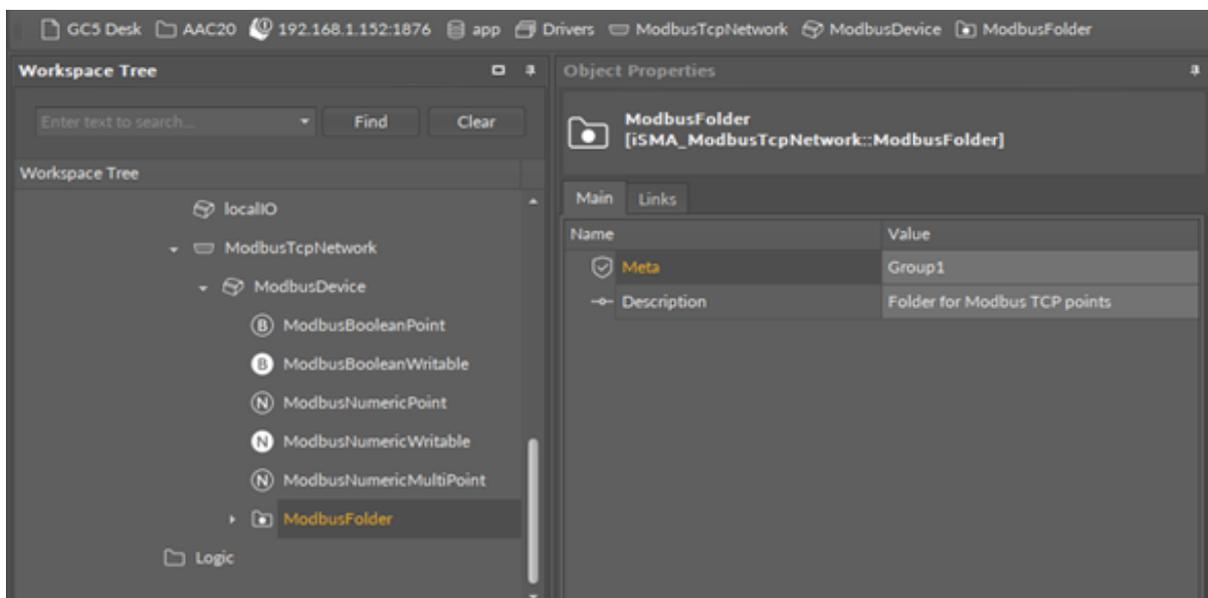
Figure 22. ModbusFolder component

6 ModbusTCPSlaveNetwork

The controller has a built-in Modbus TCP/IP server (Modbus TCP slave device) application, which is always active. The controller has a built-in register table to read/write data. Addresses from 0 to 999 (decimal numeration) are reserved for controller registers. To see list of registers and registers parameters go to the List of Modbus Registers chapter. Addresses from 1000 to 2999 are free to use and can be adopted in the user application.
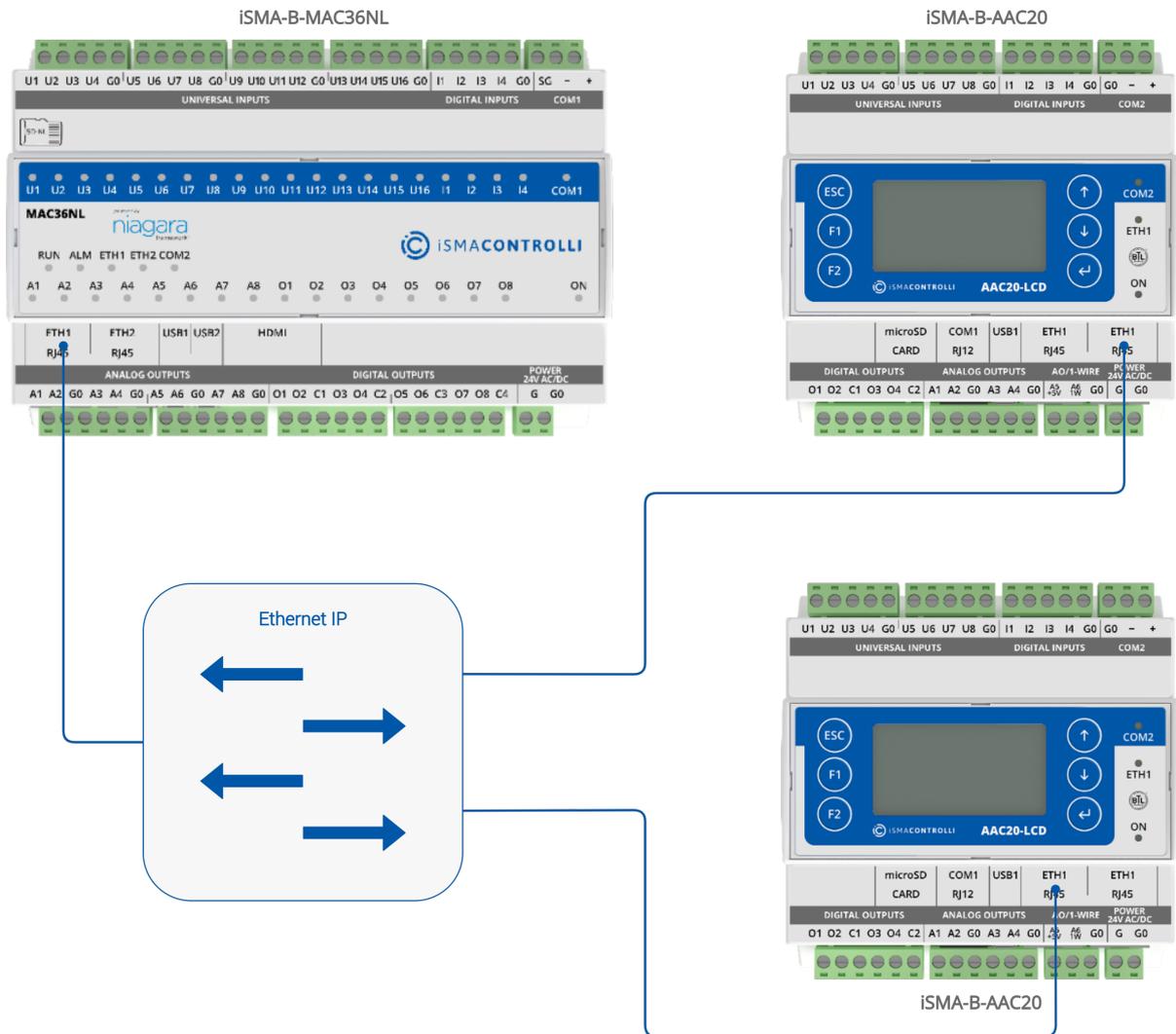
Figure 23. Modbus slave network

6.1 ModbusTCPSlaveNetwork Component

The ModbusTCPSlaveNetwork is always enabled, and it does not have to be configured in order to read the controller registers. The ModbusTCPSlaveNetwork is used only for changing parameters (parameters can be changed also from the controller's configuration web page) and to set up user registers.
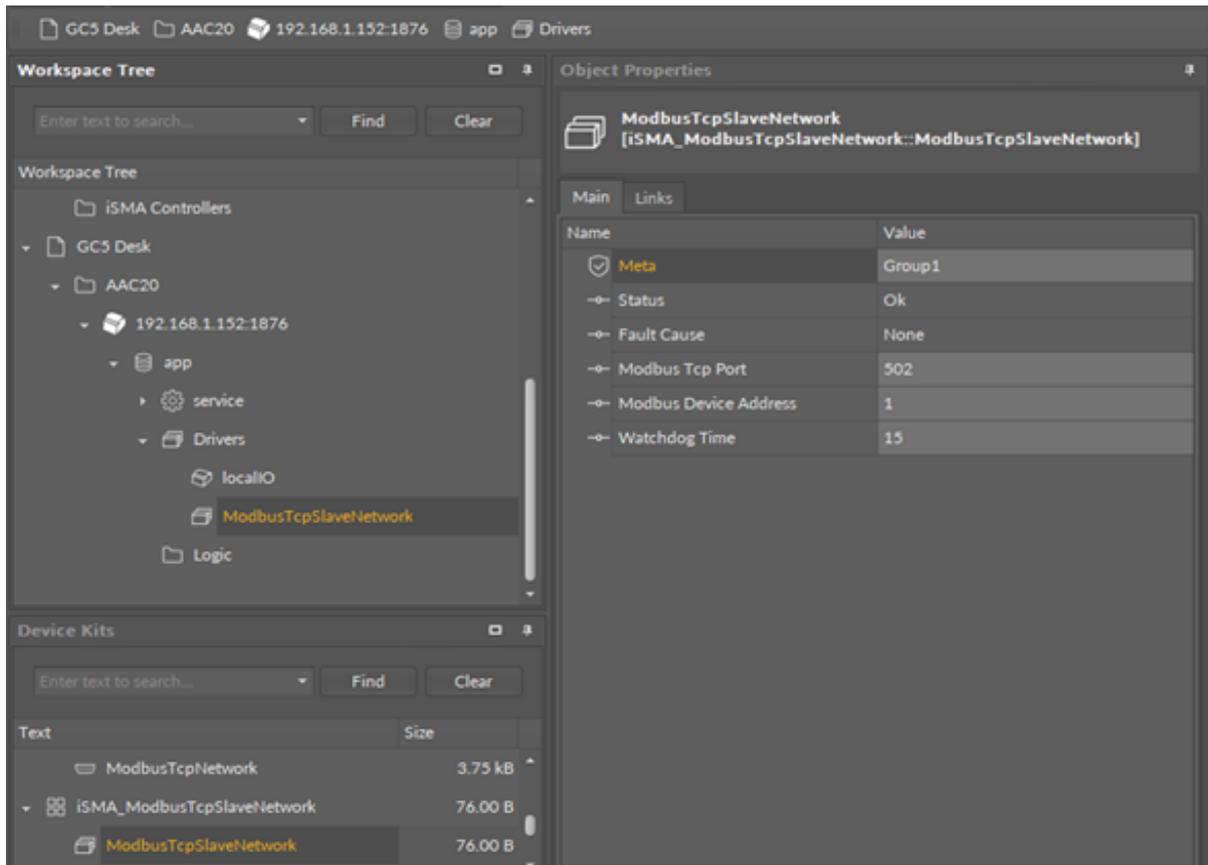
Figure 24. ModbusTCPSlaveNetwork component

The ModbusTCPSlaveNetwork component has the following slots:

- **Status**: network's status;
- **Fault Cause**: fault cause description;
- **Modbus TCP Port**: Modbus TCP port number (default 502);
- **Modbus Device Address**: controller Modbus address;
- **Watchdog Time**: time between received valid messages, after which the controller will reset default values on outputs; value 0 disables this function.

6.2 Modbus TCP Slave Data Points

The Modbus TCP Slave Data Points are served by two components placed under the ModbusTCPSlaveNetwork component:

- **BooleanValue**: reads/writes Boolean values;
- **NumericValue**: reads/writes numeric values.

WARNING! There is only one table for both values. Data points addresses are assigned manually, please take care not to override one register from many components.

WARNING! The BooleanValue and NumericValue both have the read and write function. To read-only use the Out slot only. Leave the In slot not connected with null (for Boolean) or nan (for numeric) value.

> WARNING! Using controller outputs in the Sedona application will disable writing function to the controller output registers. In this case, Sedona application has a higher priority.

6.2.1 BooleanValue

The BooleanValue component is responsible for reading and writing Boolean values in the controller's Modbus table. Values can be read only for Modbus master (bit type: discrete input) or read and write for Modbus master (bit type: coil).

Only one Modbus table is used for writing the Boolean and numeric values. Before addressing the component make sure that the register is not in use by any another component.
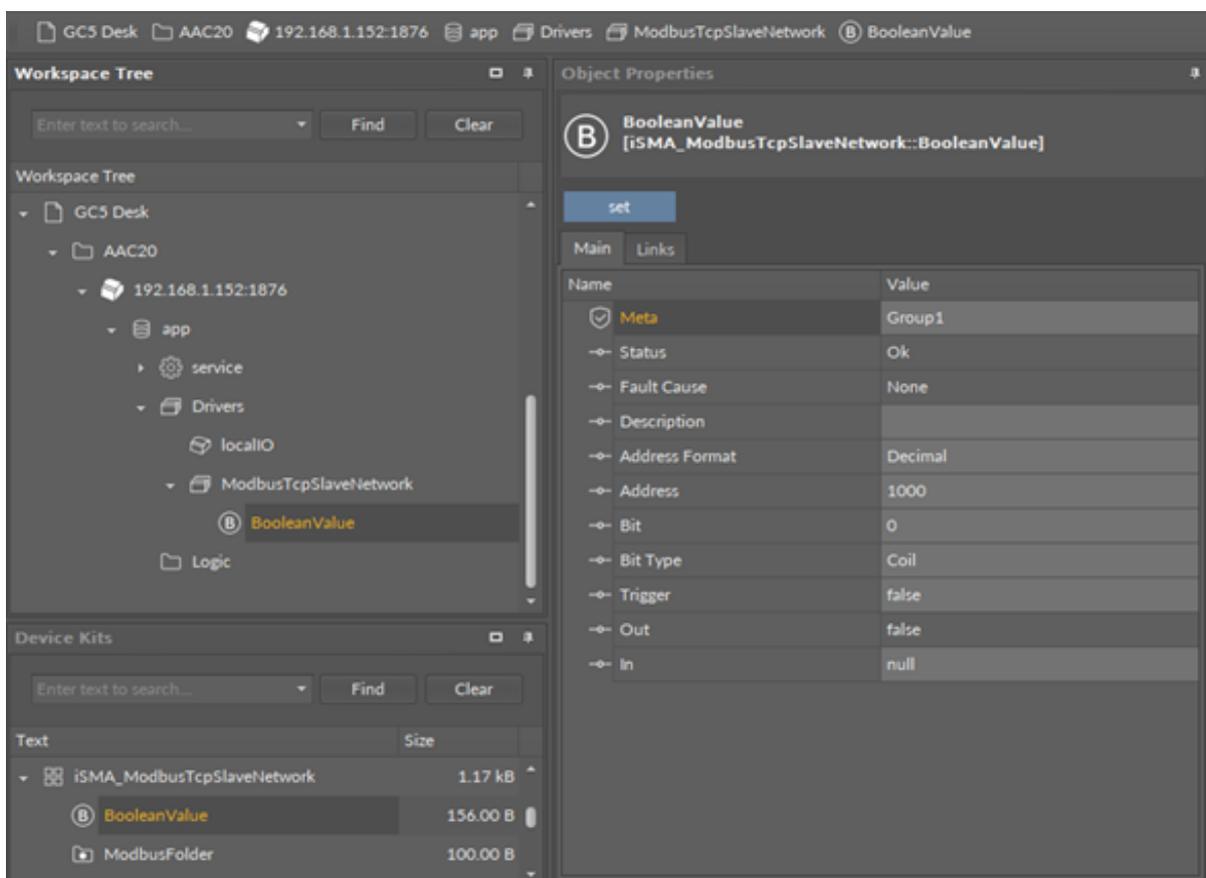
Figure 25. BooleanValue component

Slots

The BooleanValue component has the following slots:

- **Status**: point's current status;
- **Fault Cause**: fault cause description;
- **Description**: point description label up to 32 characters;
- **Address Format:** Modbus addressing format: Modbus/decimal;
- **Address**: register address (from 0 to 65535);
- **Bit**: bit number in 16-bits register (from 0 to 15);
- **Bit Type**: bit type for Master Device: Coil (read/write), Discrete Input (read-only);
- **Trigger**: forcefully send the Input value to controller Modbus table (on rising edge);

- **Out**: output slot, the current value of the device register;
- **In:** input slot.

6.2.2 NumericValue

The NumericValue component is responsible for reading and writing numeric values to the controller's Modbus table. Values can be read only for the Modbus master (register type: Input Register) or read and write for the Modbus Master (register type: Holding Register).

Only one Modbus table is used for writing the Boolean and Numeric values. Before addressing the component make sure that the register is not in use by any another component.

> WARNING! Data Types: Long, SLong, and Float use 32-bits format and use two registers. Next free register in the table is the Register address + 2. For example: Float value register address is 1010, the next register must be addressed 1012.
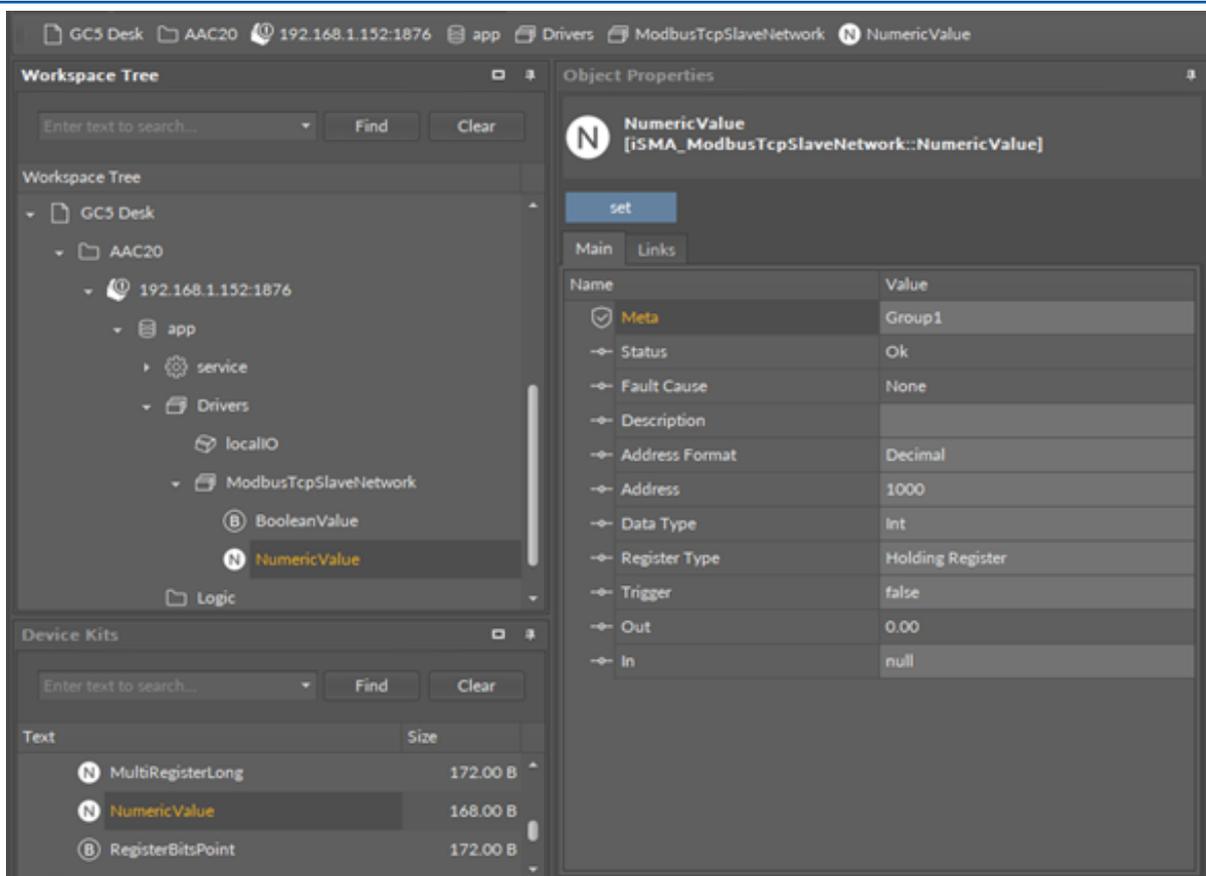
Figure 26. NumericValue component

Slots

The NumericValue component has the following slots:

- **Status**: point's status;
- **Fault Cause**: fault cause description;
- **Description**: point description label up to 32 characters;
- **Address Format:** Modbus addressing format: Modbus/decimal;
- **Address**: register address (from 0 to 65535);

- **Data Type:** variable data type: Int, Sint, Long, Slong, Float;

> WARNING! Long, Slong and Float are 32bit and they use 2 registers.

- **Register Type**: register type for master device: Holding Register (read/write), Input Register (read-only);
- **Trigger**: forcefully send the Input value to controller Modbus table (on rising edge);
- **Out**: output slot, the current value of the device register;
- **In:** input slot.

6.2.3 MultiRegisterFloat

The MultiRegisterFloat component is responsible for reading float-type multi-register values. The component can read data from up to four registers.
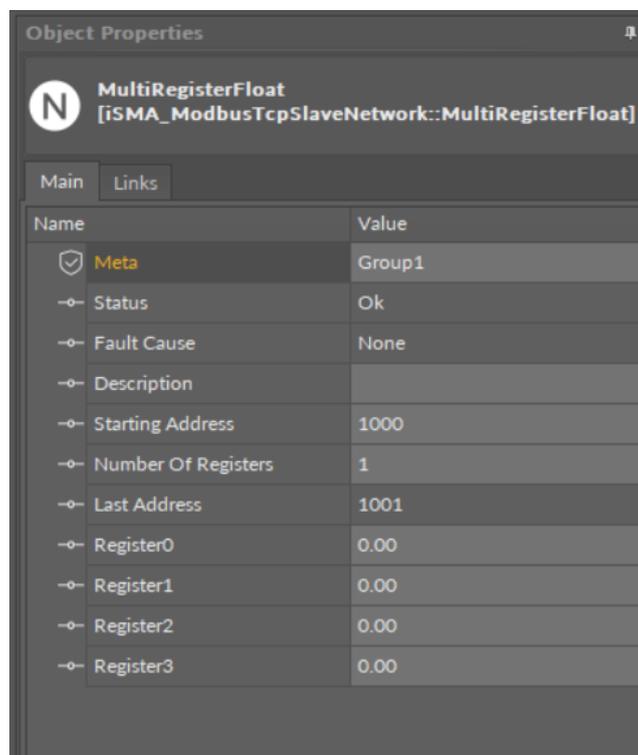
Figure 27. MultiRegisterFloat component

Slots

The MultiRegisterFloat component has the following slots:

- **Status**: point's status;
- **Fault Cause**: fault cause description;
- **Description**: point description label up to 32 characters;
- **Starting Address:** Modbus address of a first register of the multi-register value (from 0 to 65535);
- **Number of Registers**: number of registers of the multi-register value;
- **Last Address:** Modbus address of a last register of the multi-register value (from 0 to 65535);
- **Register0-4**: values of each register defined in a scope of Starting to Last Address.

6.2.4 MultiRegisterInt

The MultiRegisterInt component is responsible for reading integer-type multi-register values. The component can read data from up to eight registers.
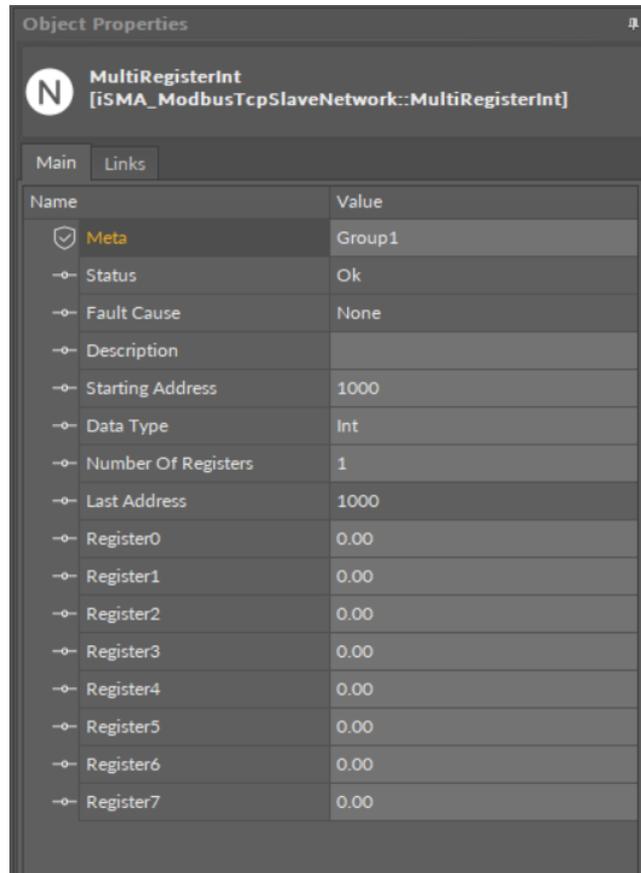
Figure 28. MultiRegisterInt component

Slots

The MultiRegisterInt component has the following slots:

- **Status**: point's status;
- **Fault Cause**: fault cause description;
- **Description**: point description label up to 32 characters;
- **Starting Address:** Modbus address of a first register of the multi-register value (from 0 to 65535);
- **Data Type:** defines a data type (Int, SInt);
- **Number of Registers**: number of registers of the multi-register value;
- **Last Address:** Modbus address of a last register of the multi-register value (from 0 to 65535);
- **Register0-7**: values of each register defined in a scope of Starting to Last Address.

6.2.5 MultiRegisterLong

The MultiRegisterLong component is responsible for reading long-type multi-register values. The component can read data from up to four registers.
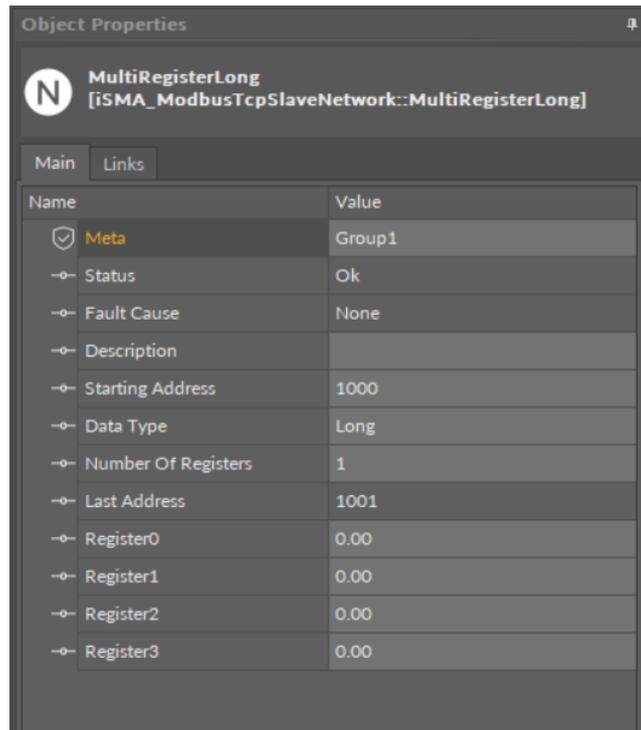
Figure 29. MultiRegisterLong component

Slots

The MultiRegisterLong component has the following slots:

- **Status**: point's status;
- **Fault Cause**: fault cause description;
- **Description**: point description label up to 32 characters;
- **Starting Address:** Modbus address of a first register of the multi-register value (from 0 to 65535);
- **Data Type:** defines a data type (Long, SLong);
- **Number of Registers**: number of registers of the multi-register value;
- **Last Address:** Modbus address of a last register of the multi-register value (from 0 to 65535);
- **Register0-4**: values of each register defined in a scope of Starting to Last Address.

6.2.6 RegisterBitsPoint

The RegisterBitsPoint component is responsible for reading values from a bit in a specified register in the device. The component has to be placed in the ModbusTCPSlaveNetwork component.
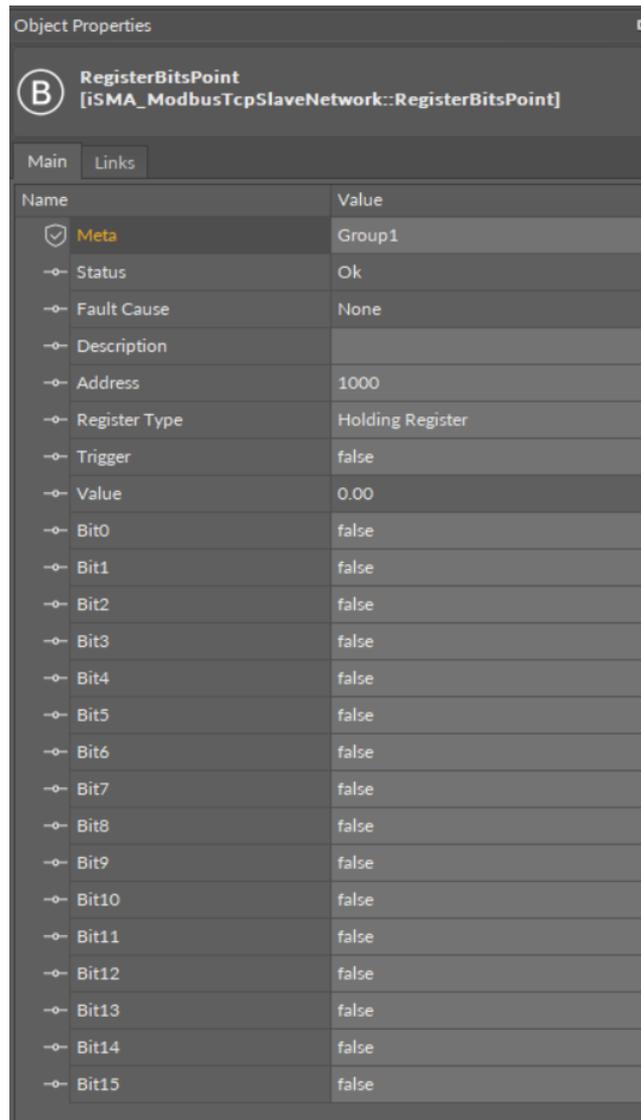
Figure 30. RegisterBitsPoint component

Slots

The RegisterBitsPoint component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **Description:** point's description label;
- **Address:** allows to set the register address, which the values will be read from;
- **Register Type:** defines a type of the addressed register;
- **Trigger:** invokes a reading action;
- **Value:** shows a decimal representation of the Boolean bits values;
- **Bit0-Bit15:** Boolean value of each of the register's bits.

6.2.7 ModbusAsyncSlaveExtension

The ModbusAsyncSlaveExtension component activates the Modbus Async slave network for the device. The component has to be placed in the ModbusTCPSlaveNetwork component.
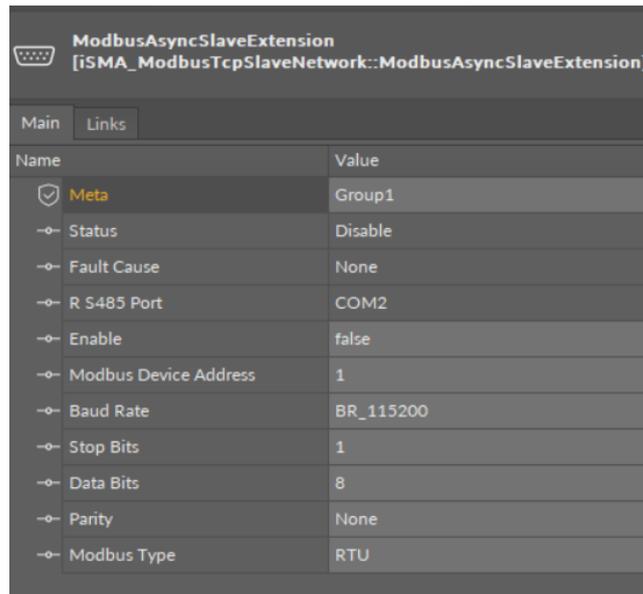
Figure 31. ModbusAsyncSlaveExtension component

Slots

The ModbusAsyncSlaveExtension component has the following slots:

- **Status:** shows the point's status;
- **Fault Cause:** shows the fault cause description;
- **RS485 Port:** shows a number of port used for the Modbus Async slave network communication;
- **Enable:** enables or disables functioning of the Modbus Async slave network;
- **Modbus Device Address:** allows to set the device's Modbus address;
- **Baud Rate**: allows to set the Modbus RS485 port baud rate;
  - Available options: 2400, 4800, 9600, 19200, 38400, 57600, 115200 bps;
- **Stop Bits**: allows to set the stop bit definition;
  - Available options: 1-bit, 2-bits;
- **Data Bits**: allows to set the data bits definition;
  - Available options: 7-bits or 8-bits;
- **Parity**: allows to set the parity bit definition;
  - Available options: None, Odd, Even, Always1, Always0;
- **Modbus Type**: allows to set the Modbus type definition;
  - Available options: RTU or ASCII.

6.2.8 ModbusFolder (TCPSlave)

The ModbusFolder is a component which groups and organizes the Modbus points components. The ModbusFolder has a Description Slot where up to 32 characters may be inserted.

7 ModbusRJ12Network

In a standard license there are available 500 data points, and this number cannot be expanded. The number of available points is shown in the ModbusRJ12Network component in the Free Points slot.

The iSMA-B-AAC20 controller has one RJ12 port, which can be used as a Modbus RTU/ASCII master.

7.1 Modbus RJ12 License and Limitation

In the standard license there are available 500 data points, and this number cannot be expanded. The number of available points is shown in the ModbusRJ12Network component in the Free Points slot.

> WARNING! Each device and data point is counted as one point. For example, to read 7 data points from 15 devices: Points number = 15 *(1 + 7) = 105.

7.2 ModbusRJ12Network Component

The ModbusRJ12Network is the main component, which is responsible for servicing an RJ12 physical port. The component must be placed under the Drivers folder. The ModbusRJ12Network sets parameters such as communication baud rate and data format, testing, etc., and keeps statistics.
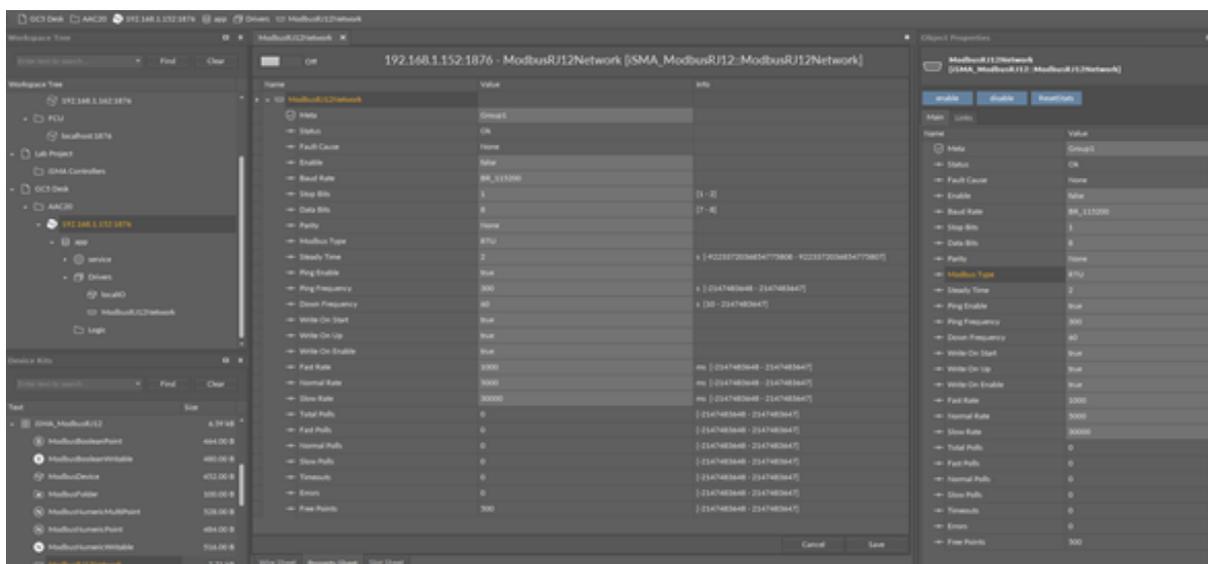
Figure 32. ModbusRJ12Network

The ModbusRJ12Network component has the following slots:

- **Status**: Network's status;
  - ○ Available states: OK (network is working properly), Disabled (network is disabled, the Enable slot is in false), OK some device/point down (error in the device or points);
- **Fault Cause**: fault cause description;
- **Enable**: option to switch on or switch off Modbus network;
  - ○ Available options: true (network enabled), false (network disabled);
- **Steady Time**: network's delay time to start-up after a power-up or reset;
- **Baud Rate**: for Modbus RS485 port baud rate;

- ◦ Available options: 2400, 4800, 9600, 19200, 38400, 57600, 115200 bps;
- **Stop Bits**: stop bit definition;
  - ◦ Available options: 1-bit, 2-bits;
- **Data Bits**: data bits definition;
  - ◦ Available options: 7-bits or 8-bits;
- **Parity**: parity bit definition;
  - ◦ Available options: None, Odd, Even, Always1, Always0;
- **Modbus Type**: Modbus type definition;
  - ◦ Available options: RTU or ASCII;
- **Ping Enable**: enables the device's connection testing function;
- **Ping Frequency**: time between testing messages to check device connection;
- **Down Frequency**: time between testing messages for devices or points, which have got status down;
- **Write On Start**: executes a write action in device writable components in the Modbus network after a reset or power-up;
- **Write On Up**: executes a write action in device writable components in the Modbus network after restoring the connection with the Modbus device;
- **Write On Enable**: executes a write action in device writable components in the Modbus network after enabling the device;
- **Fast Rate**: time between messages in the fast mode poll frequency;
- **Normal Rate**: time between messages in the normal mode poll frequency;
- **Slow Rate**: time between messages in the slow mode poll frequency;
- **Total Polls**: total number of messages;
- **Fast Polls**: number of messages sent in the fast mode;
- **Normal Polls**: number of messages sent in the normal mode;
- **Slow Polls**: number of messages sent in the slow mode;
- **Timeouts**: number of lost messages, the difference between sent and received messages;
- **Errors**: number of error messages (for example, with the wrong CRC);
- **Free points**: number of available physical points in the Modbus network.

The ModbusTCPNetwork component has the following actions available under the right-click or in the Object Properties window:

- **Reset Stats**: resets network's statistics and starts counting from the beginning;
- **Enable/Disable**: switching the Modbus network on/off.

7.3 Modbus RJ12 Wiring

The RJ12 port connector is located between the SD card and USB slots. The connector provides Modbus bus wires, ground potential G0, and power supply directly connected to G terminal from the power supply connector (the external devices can be powered through the RJ12 connector). A wiring diagram is shown in the figure below.
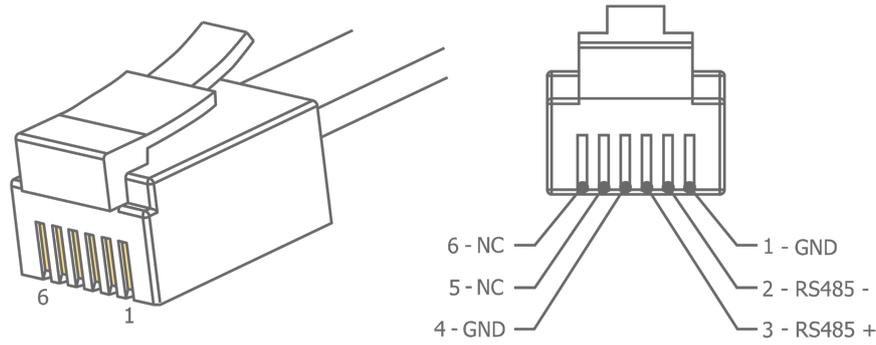
Figure 33. Modbus RJ12 wiring

The RJ12 pins description:

- **Pin1:** G0 potential, (SD card side);
- **Pin2:** RS485 - (B);
- **Pin3:** RS485 + (A);
- **Pin4:** G0 potential;
- **Pin5:** G potential, directly connected to G terminal in power supply;
- **Pin6**: G potential, directly connected to G terminal in power supply (USB side).

Connection of the devices is shown in the figure below.
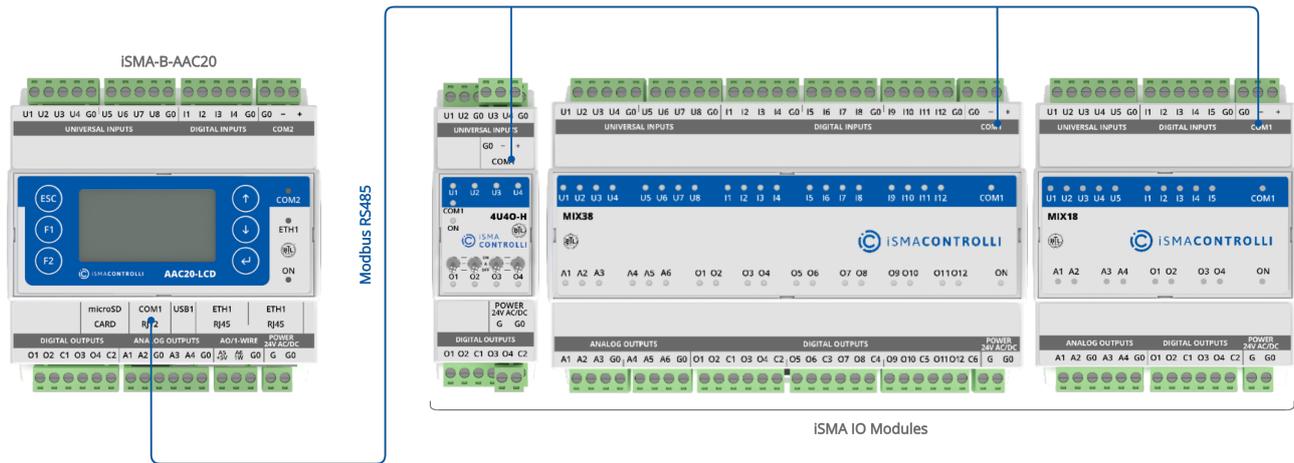
Figure 34. RJ12 to RS485 Modbus connection

7.4 ModbusRJ12Device

The ModbusRJ12Device is a component which is responsible for servicing a physical device connected to the ModbusRJ12Network. The device is a Modbus master to all other Modbus devices on the attached RJ12 port. Each device is represented by the ModbusDevice, and has a unique Modbus address (from 1 to 247) as well as the other Modbus config data and starting addresses for Modbus data items (coils, inputs, input registers, holding registers). The component has a Ping action available under the right-click, which sends a test message to the device to check the device status. Each ModbusDevice has a "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 use as the device status test (meaning "Monitor" ping requests). Ping requests are generated at the network level by the configurable network monitor (ModbusNetwork -> Ping Enable). When enabled the network monitor periodically pings (queries) this address. If any response is received from

the device, including an exception response, this is considered a proof of communication, and the Modbus client device is no longer considered "down" if it had been previously marked "down".
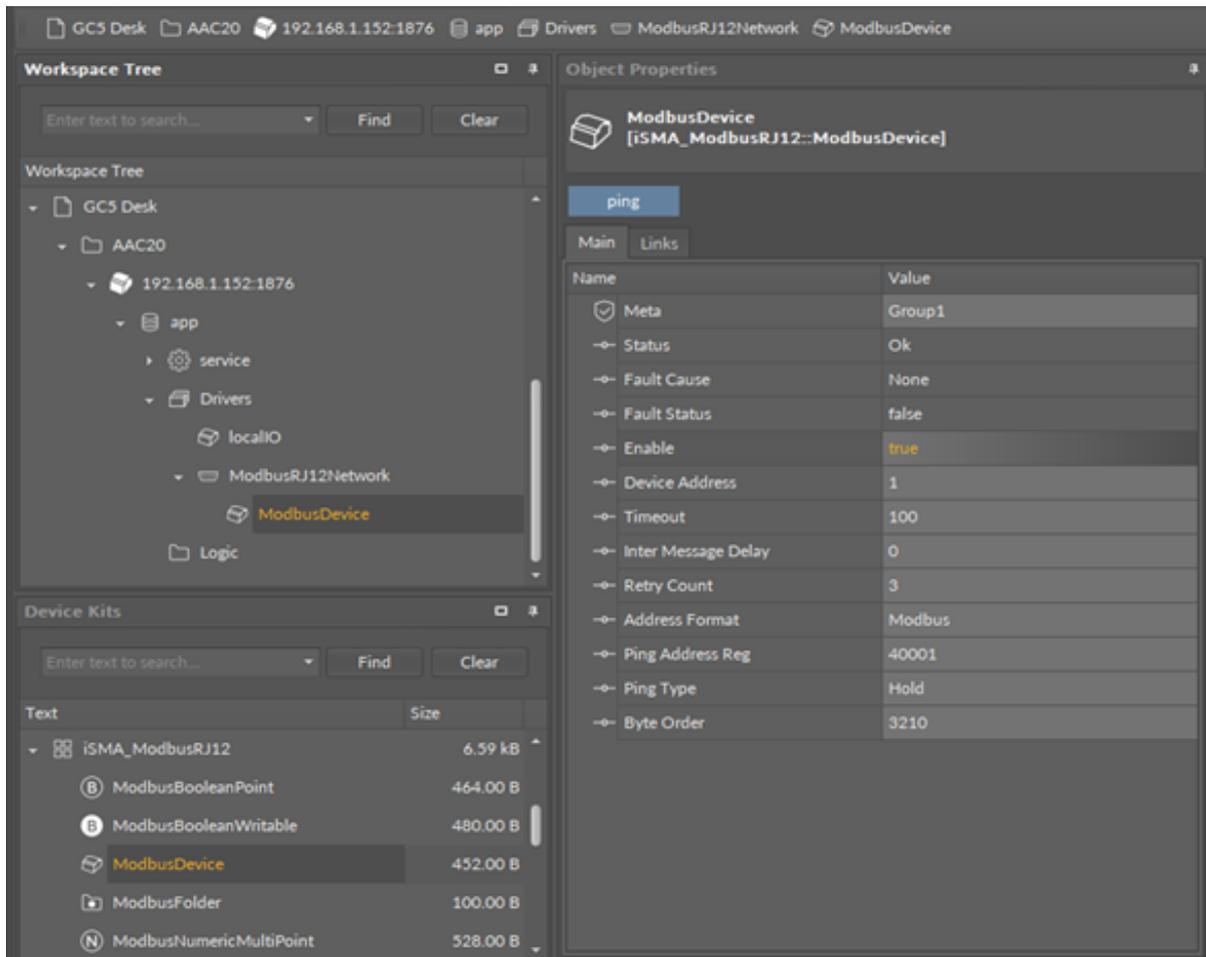
Figure 35. ModbusRJ12Device component

The ModbusRJ12Device component has the following slots:

- **Status**: Device's actual status (read-only);
    - ○ Available states: OK (device is working properly), Disable (device is disabled, the Enable slot is in false), Down (device is not available), Ok, some points down/error (error in points reading), Network disabled (Modbus network is disabled);
- **Fault Cause**: fault cause description;
- **Fault Status**: device error status;
    - ○ Available states: true (device communication error), false;
- **Enable**: enables/disables the device;
- **IP Address**: slave device (gateway) IP address;
- **Device Address**: Modbus device address (0 - broadcast, 1-248 addressing range);
- **Timeout**: max. device response time from the device request;
- **Inter Message Delay**: time between messages sent to the device;
- **Retry Count**: max. number of error messages (CRC error, lost messages);
- **Address Format**: Modbus address format (Modbus, decimal);
- **Ping Address Reg**: any register (Input/Holding) number for device connection test;
- **Ping Type**: tested register type: Input/Holding;
- **Byte Order:** byte reading order , for 32-bit: 3210 (Big endian), 1032 (Little endian).

7.5 Modbus RJ12 Data Points

The ModbusRJ12Network uses the same data points as the ModbusAsyncNetwork.

In the Modbus protocol each device has an implemented Modbus table. Sedona has 7 components to read/write data from this table:

- Boolean Point: reads Boolean values (Modbus command 0x02);
- Boolean Writable: reads/writes Boolean values (read: Modbus command 0x02, write: Modbus command 0x05);
- Numeric Point: reads numeric values (Modbus commands: 0x03 for reading holding registers, 0x04 for reading input registers);
- Numeric Writable: reads/writes numeric values (Modbus commands: 0x03 and 0x04 for reading, 0x06 for writing 16-bits Int, SInt values, 0x10 for writing 32-bits Long, SLong, Float values);
- Numeric Multi Point: reads up to eight 16-bits registers (Modbus commands 0x03 and 0x04);
- RegisterBitPoint: reads Boolean values from a specified register in the device (Modbus command 0x02);
- RegisterBitWritable: reads/writes Boolean values from/to a specified register (read: Modbus command 0x02, write: Modbus command 0x05).

7.5.1 BooleanPoint

The ModbusRJ12Network uses the same data points as the ModbusAsyncNetwork.

The ModbusAsyncBooleanPoint is a component, which is responsible for reading Boolean values from the device. The component has a Read action available under the right-click, which forces the reading of the point.

Slots

The ModbusAsyncBooleanPoint component has the following slots:

- **Status**: point's status;
    - Available states: OK (point is working properly), Disabled (point is disabled, the Enable slot is in false), Down/Timeout (point is not available), Device Down (device is not available), Wrong address format (incorrect address format according to the address format setting slot), Device disabled (device is disabled), Network disabled (Modbus network is disabled);
- **Fault Cause**: fault cause description;
- **Fault Status**: point error status;
    - Available options: true (point read error), false;
- **Enable**: enables/disables the point;
    - Available options: true (point enabled), false (point disabled);
- **Address Format:** register address format;
    - Available options: Modbus, decimal;
- **Address**: register address;
- **Poll Frequency**: reading poll frequency;
    - Available options: fast, normal, slow;
- **Status Type**: type of reading register;
    - Available options: input: 0x02, coil: 0x01;
- **Out**: current value of the read register.

7.5.2 BooleanWritable

> The ModbusRJ12Network uses the same data points as the ModbusAsyncNetwork.

The ModbusAsyncBooleanWritable is a component which is responsible for sending and reading Boolean values from the device.

Slots

The ModbusAsyncBooleanWritable component has the following slots:

- **Status**: point's status;
    - Available states: OK (point is working properly), Disabled (point is disabled, the Enable slot is in false), Down/Timeout (point is not available), Device Down (device is not available), Wrong address format (incorrect address format according to the address format setting slot), Device disabled (device is disabled), Network disabled (Modbus network is disabled).
- **Fault Cause**: fault cause description;
- **Fault Status**: point error status;
    - Available options: true (point read/write error), false;
- **Enable**: enables/disables the point
    - Available options: true (point enabled), false (point disabled),
- **Address Format:** register address format
    - Available options: Modbus, decimal;
- **Address**: register address;
- **Poll Frequency**: reading poll frequency;
    - Available options: fast, normal, slow;
- **Write Type**: writing mode;

- Available options: COV (only on input change), COV_PollFrequency (on input change and periodically), PollFrequency (only periodically), COV_LinkSet (link-back forward triggered by COV);
- **Trigger**: forcefully send the value (on rising edge), regardless of the current poll mode;
- **Out**: output slot, the current value of read/write register;
- **In:** input slot.

Actions

The ModbusAsyncBooleanWritable component has the following actions available under the right-click:

- **Set True/Set False:** writes a value to the In slot and sends it to the device (not active when slot In has a connected link);
- **Write**: sends a value from the In slot to the device;
- **Read**: reads a value from the device and sends to the Out slot.

7.5.3 NumericPoint

> The ModbusRJ12Network uses the same data points as the ModbusAsyncNetwork.

The ModbusAsyncNumericPoint is a component, which is responsible for reading numeric values from the device. The component has a Read action available under the right-click, which forces the reading of the point.

Slots

The ModbusAsyncNumericPoint component has the following slots:

- **Status**: point's status;
  - Available states: OK (point is working properly), Disabled (point is disabled, the Enable slot is false), Down/Timeout (point is not available), Device Down (device is not available), Wrong address format (incorrect address format according to address format setting slot), Device disabled (device is disabled), Network disabled (Modbus network is disabled);
- **Fault Cause**: fault cause description;
- **Fault Status**: point error status;
  - Available options: true (point read error), false;
- **Enable**: enables/disables the point;
  - Available options: true (point enabled), false (point disabled);
- **Address Format:** register address format;
  - Available options: Modbus, decimal;
- **Address**: register address;
- **Poll Frequency**: reading poll frequency;
  - Available options: fast, normal, slow;
- **Reg Type**: type of reading register;
  - Available options: input: 0x04, holding: 0x03;
- **Data Type**: reading register data type;
  - Available options: Int: 16-bits, Long: 32-bits, Float: 32-bits float-point, SInt: 16-bits with sign, Slong: 32-bits with sign;
- **Out**: current value of the read register.

7.5.4 NumericWritable

> The ModbusRJ12Network uses the same data points as the ModbusAsyncNetwork.

The ModbusAsyncNumericWritable is a component, which is responsible for sending and reading numeric values from the device.

Slots

The ModbusAsyncNumericWritable component has the following slots:

- **Status**: point's status, available states:
    - Available states: OK (point is working properly), Disabled (point is disabled, the Enable slot is in false), Down/Timeout (point is not available), Device Down (device is not available), Wrong address format (incorrect address format according to the address format setting slot), Device disabled (device is disabled), Network disabled (Modbus network is disabled).
- **Fault Cause**: fault cause description;
- **Fault Status**: point error status;
    - Available options: true (point read/write error), false;
- **Address Format:** register address format;
    - Available options: Modbus, decimal;
- **Address**: register address;
- **Data Type**: read/write register data type;
    - Available options: Int: 16-bits, Long: 32-bits, Float: 32-bits float-point, SInt: 16-bits with sign, Slong: 32-bits with sign, IntF16- use Function 16, SIntF16: use Function 16 (Function 16: Modbus function for sending one register);
- **Poll Frequency**: reading poll frequency;
    - Available options: fast, normal, slow;
- **Write Type**: writing mode;
    - Available options: COV - only on input change, COV_PollFrequency: on input change and periodically, PollFrequency - only periodically, COV_LinkSet (Link-back forward triggered by COV);
- **Trigger**: forcefully send the value (on rising edge), regardless of the current poll mode,
- **Out**: output slot, the current value of the device register,

- **In:** input slot.

Actions

The ModbusAsyncNumericWritable component has the following actions available under the right mouse button:

- **Set**: writes a value to the In slot and sends it to the device;
- **Write**: sends a value from the In slot to the device;
- **Read**: reads a value from the device and sends it to the Out slot.

7.5.5 NumericMultiPoint

> The ModbusRJ12Network uses the same data points as the ModbusAsyncNetwork.

The ModbusAsyncNumericMultiPoint is a component, which is responsible for reading up to eight 16-bits registers from the device in one message. The component uses 0x03 or 0x04 Modbus commands. The component has a Read action available under the right-click, which forces the reading of the point.

Slots

The ModbusAsyncNumericMultipoint component has the following slots:

- **Status**: point's status;

- Available states: OK (point is working properly), Disabled (point is disabled, the Enable slot is false), Down/Timeout (point is not available), Device Down (device is not available), Wrong address format (incorrect address format according to address format setting slot), Device disabled (device is disabled), Network disabled (Modbus network is disabled);
- **Fault Cause**: fault cause description;
- **Fault Status**: point error status;
    - Available options: true (point read error), false;
- **Enable**: enables/disables the point;
    - Available options: true (point enabled,) false (point disabled);
- **Address Format:** Register address format;
    - Available options: Modbus, decimal;
- **Address**: register address;
- **Poll Frequency**: reading poll frequency;
    - Available options: fast, normal, slow;
- **Reg Type**: type of reading register;
    - Available options: input - 0x04, holding - 0x03;
- **Data Type**: read data type: Int (unsigned values), Sint (signed values);
- **Number Of Registers**: number of registers read in one message;
- **Out**: current value of the read register.

7.5.6 RegisterBitPoint (RJ12)

The ModbusRJ12Network uses the same data points as the ModbusAsyncNetwork.

The ModbusAsyncRegisterBitPoint component is responsible for reading Boolean values from a bit in a specified register in the device. The component has to be placed under the ModbusAsyncDevice component.

Slots

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.

Action

The ModbusAsyncRegisterBitPoint component offers the following action:

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

7.5.7 RegisterBitWritable (RJ12)

> The ModbusRJ12Network uses the same data points as the ModbusAsyncNetwork.

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

Slots

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_PollFequrency: on the In slot change and periodically, PollFequrency: only periodically, COV_LinkSet : only on the In slot change using the "reverse following the link" function);

- **Trigger:** allows to trigger the remote enforcement of sending (on rising edge);
- **Out:** the current value of reading bit;
- **In:** the input slot.

Action

The ModbusAsyncRegisterBitWritable component offers the following actions:

- **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.

7.6 ModbusFolder

The ModbusFolder is a component which groups and organizes the Modbus points components. The ModbusFolder has the Description slot, where up to 32 characters may be inserted.
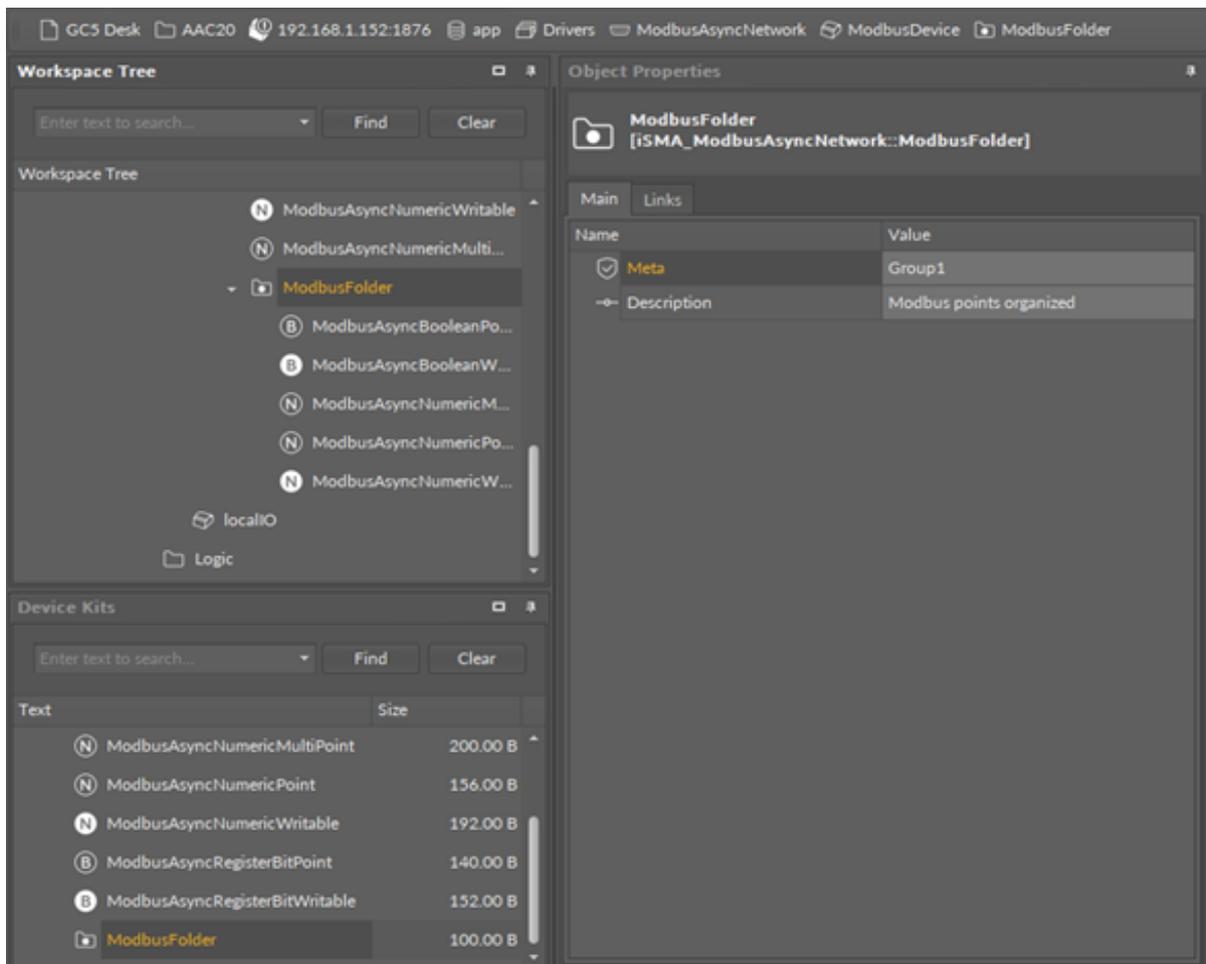
Figure 36. ModbusFolder component

8 Gateway Mode

The iSMA-B-AAC20 controller can work as a Modbus TCP/RS485 gateway. By default, this option is enabled until there is no ModbusAsyncNetwork component in the Sedona application or the component is disabled (ModbusAsyncNetwork -> Enable slot in the false state).

The RS485 communication parameters can be set up by:

- Controller configuration web page (RS485 Configuration tab);
- ModbusAsyncNetwork component (remember to set false in the Enable slot);
- Controller Modbus register table (see the iSMA-B-AAC20 Modbus table chapter).
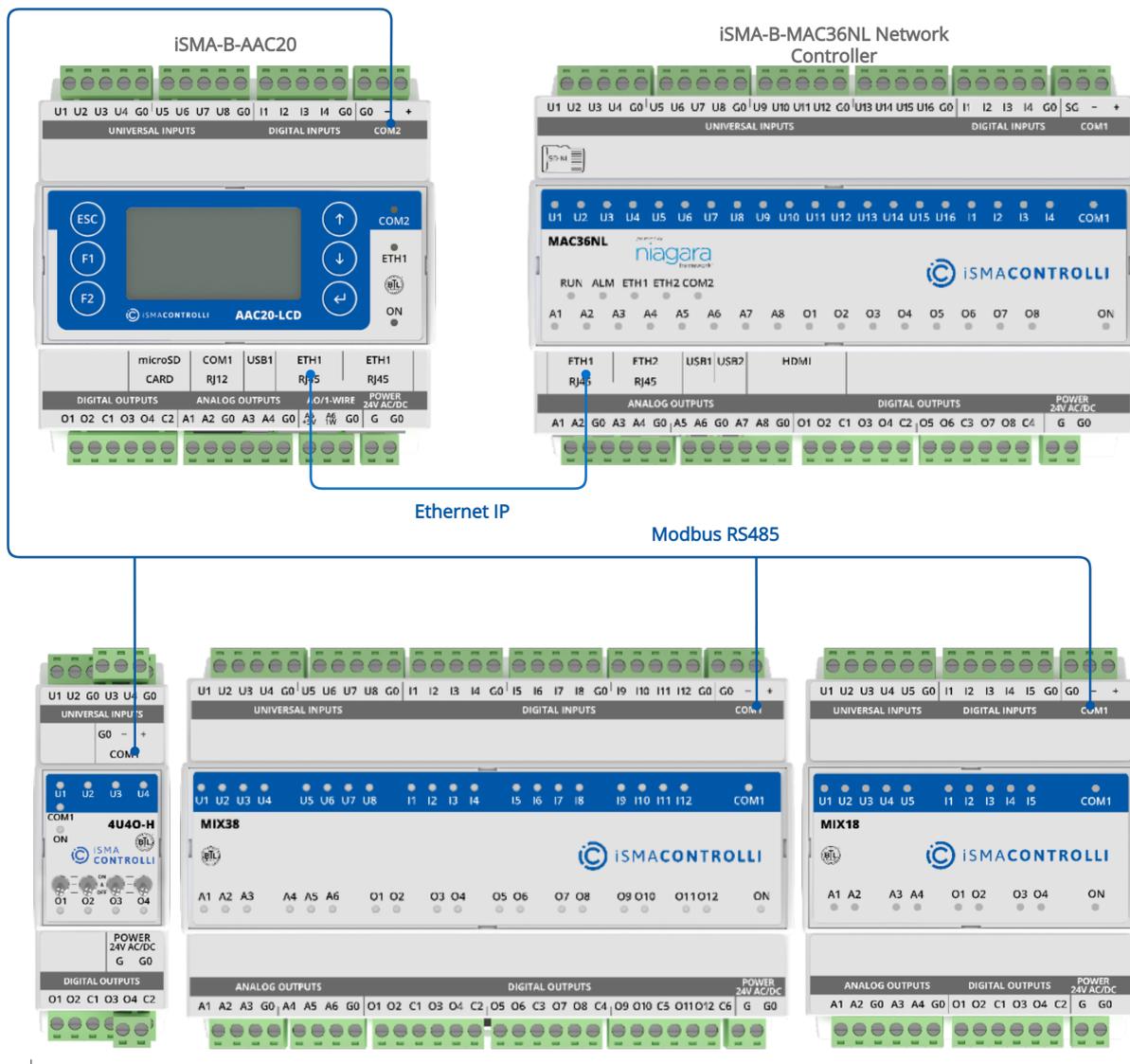
iSMA IO Modules
Figure 37. Modbus TCP gateway topology

9 iSMA Room Devices Modbus

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.

9.1 FanSpeed

This page is a source page for (V1.7) FanSpeed (AAC).

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

The FanSpeed component is responsible for configuring fan settings in the panel.
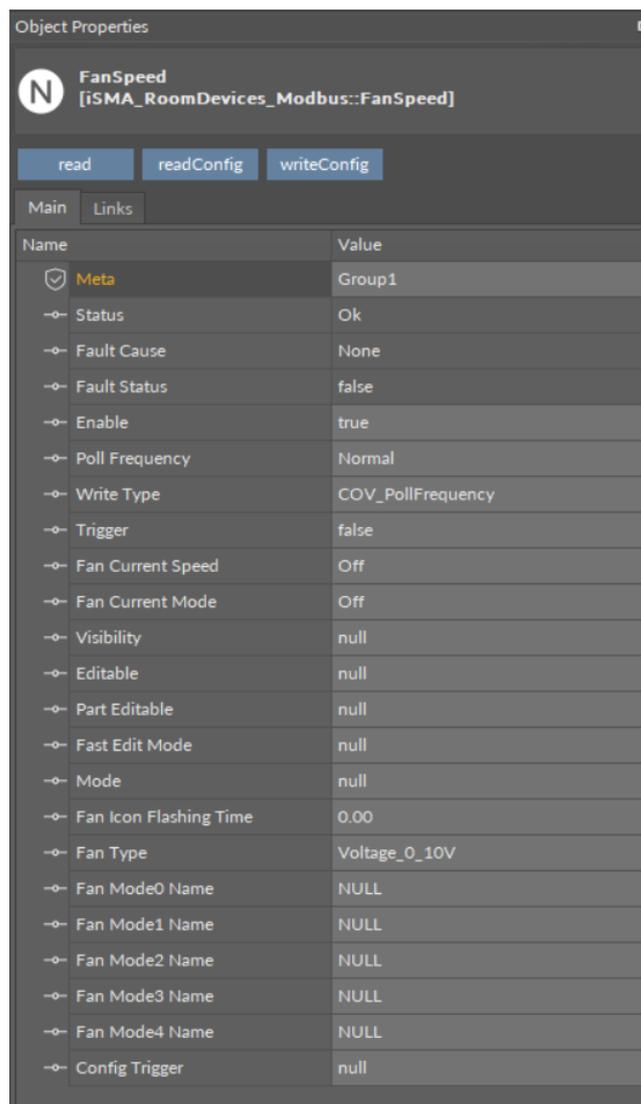
Figure 38. The FanSpeed component

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).

9.2 LpCO2Alarm

This page is a source for (V1.7) LpCO2Alarm (AAC).

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

The LpCO2Alarm component is dedicated to the configuration of the high limit alarm function in the panel.
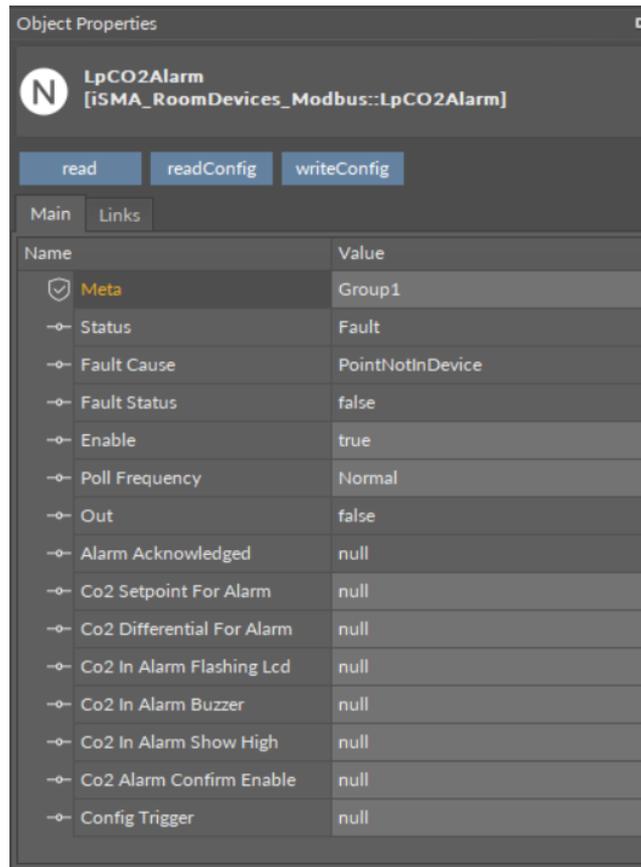
Figure 39. The LpCO2Alarm component

The LpCO2Alarm 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 CO2 alarm status;
- **Alarm Acknowledged:** informs if the alarm has been confirmed by a system operator;
- **Co2 Setpoint For Alarm:** sets a CO2 alarm setpoint value in ppm;
- **Co2 Differential For Alarm:** sets a CO2 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 LpCO2Alarm component has the following right-click menu actions:

- **Read:** reads the panel's CO2 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).

9.3 LpCO2Sensor

This page is a source for (V1.7) LpCO2Sensor (AAC).

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

The LpCO2Sensor component is responsible for reading values and configuration of the CO2 sensor.
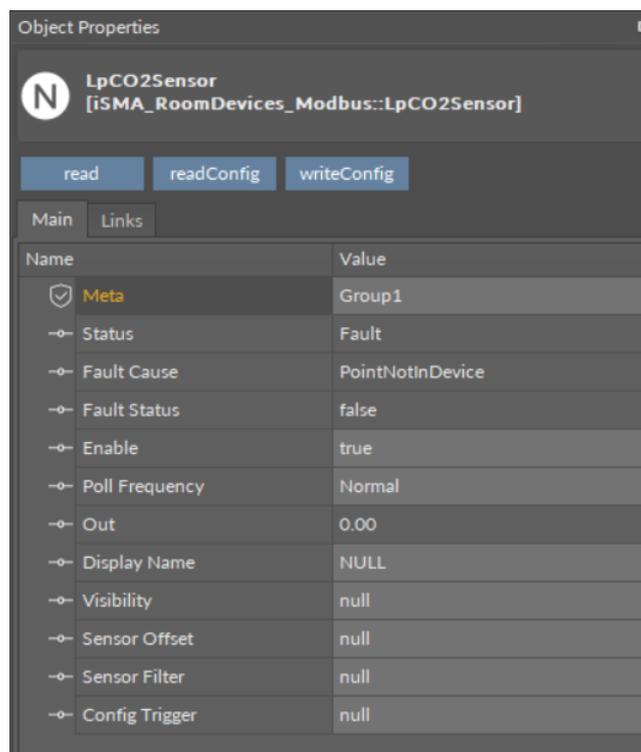
Figure 40. The LpCO2Sensor component

The LpCO2Sensor 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 CO2 sensor value;
- **Display Name:** allows to set the CO2 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 CO2 sensor offset value;
- **Sensor Filter:** sets the CO2 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 CO2 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).

9.4 LpHumiditySensor

This page is a source for (V1.7) LpHumiditySensor (AAC).

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

The LpHumiditySensor component is responsible for configuration of the humidity sensor and reading its value.
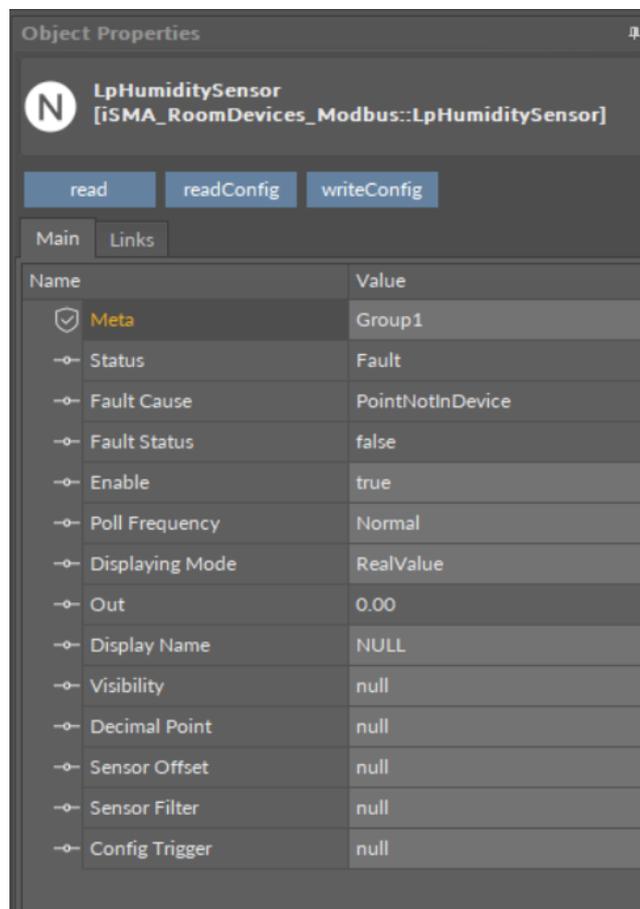
Figure 41. The LpHumiditySensor component

The LpHumiditySensor 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).

9.5 LpMainMenuBoolean

This page is a source for (V1.7) LpMainMenuBoolean (AAC).

> **Warning!**
>
> Component applicable only for the LP panel.

The LpMainMenuBoolean component is responsible for reading/writing and configuration of a single Boolean parameter, which is placed in the LP panel main menu.
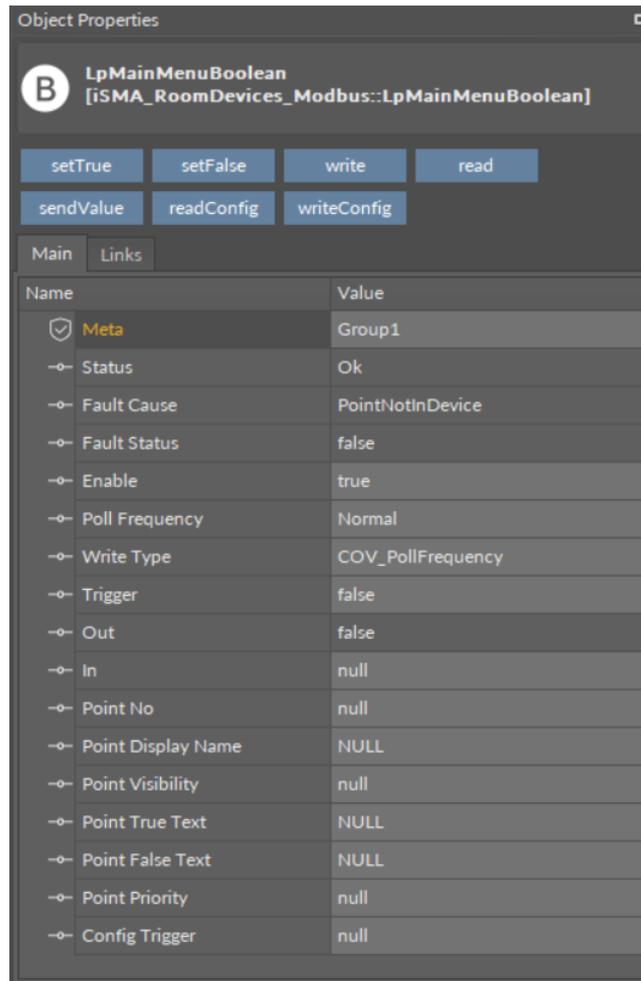
Figure 42. The LpMainMenuBoolean component

The LpMainMenuBoolean 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 LpMainMenuBoolean 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).

9.6 LpMainMenuNumeric

This page is a source for (V1.7) LpMainMenuNumeric (AAC).

> **Warning!**
>
> Component applicable only for the LP panel.

The LpMainMenuNumeric component is responsible for reading/writing and configuration of a single numeric parameter, which is placed in the LP panel main menu.
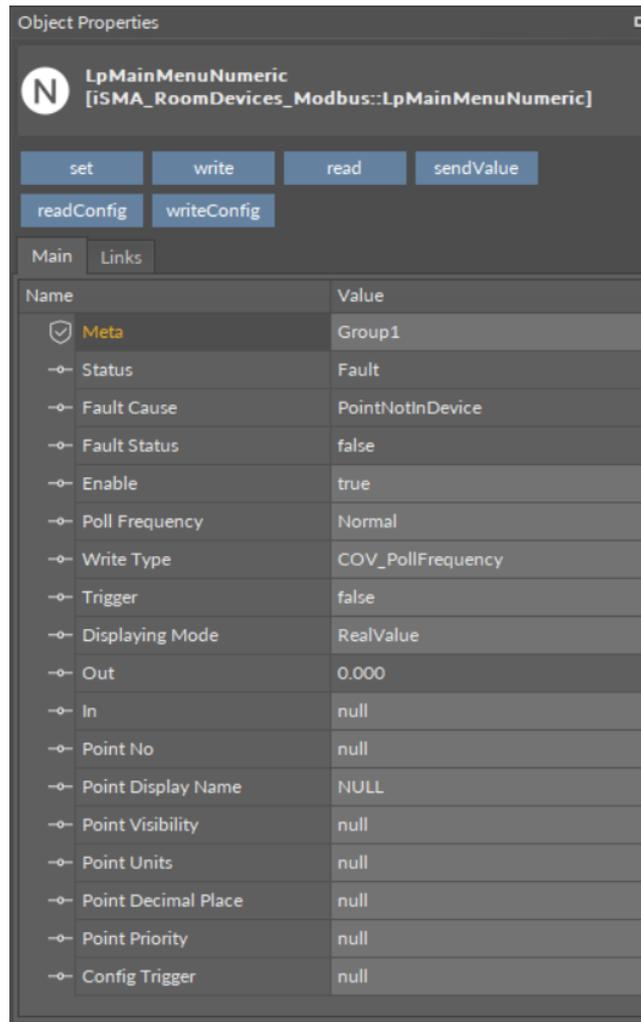
Figure 43. The LpMainMenuNumeric component

The LpMainMenuNumeric 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 LpMainMenuNumeric 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).

9.7 LpSubmenuBoolean

This page is a source for (V1.7) LpSubmenuBoolean (AAC).

> **Warning!**
>
> Component applicable only for the LP panel.

The LpSubmenuBoolean 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.
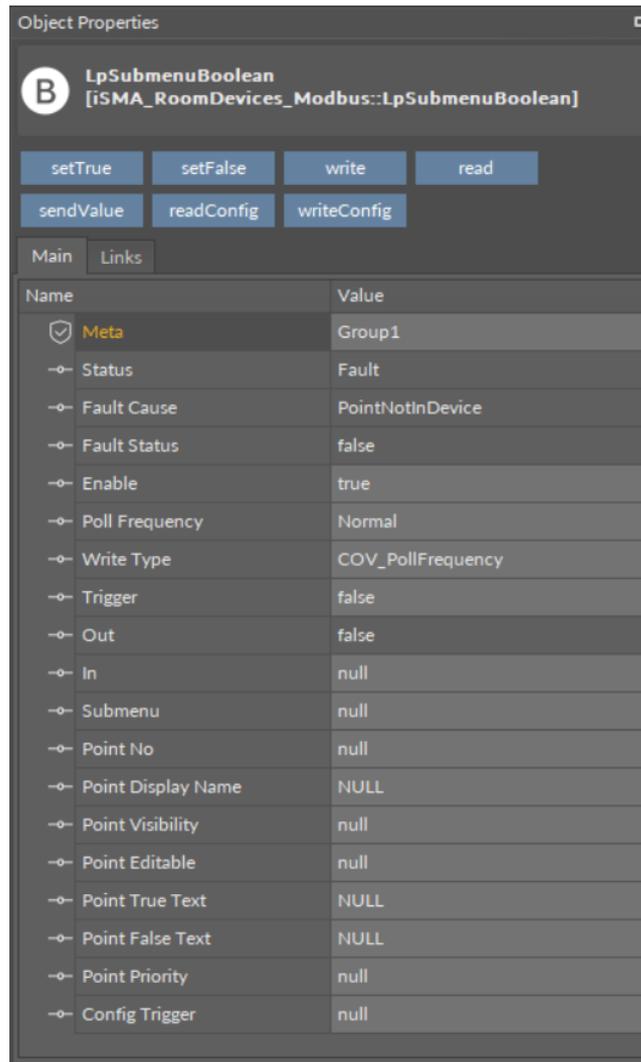
Figure 44. The LpSubmenuBoolean component

The LpSubmenuBoolean 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 LpSubmenuBoolean 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).

9.8 LpSubmenuNumeric

This page is a source for (V1.7) LpSubmenuNumeric (AAC).

> **Warning!**
>
> Component applicable only for the LP panel.

The LpSubmenuNumeric 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.
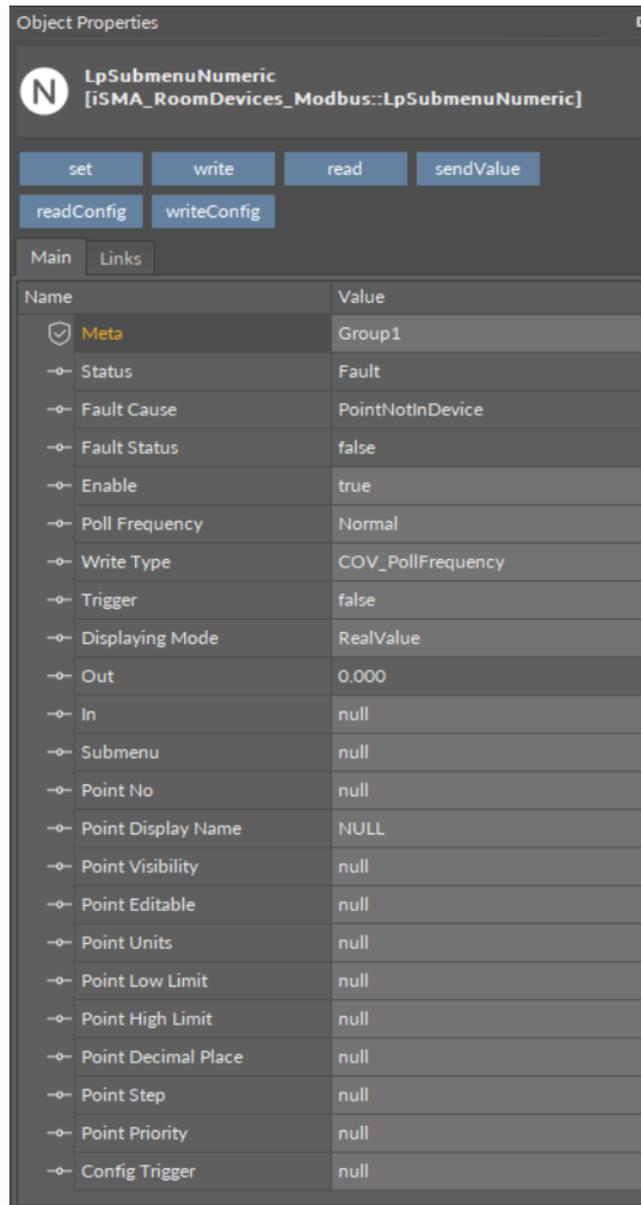
Figure 45. The LpSubmenuNumeric component

The LpSubmenuNumeric 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 LpSubmenuNumeric 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).

9.9 LpTemperatureSensor

This page is a source for (V1.7) LpTemperatureSensor (AAC).

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

The LpTemperatureSensor component is responsible for reading values and configuration of the temperature sensor in the panel.
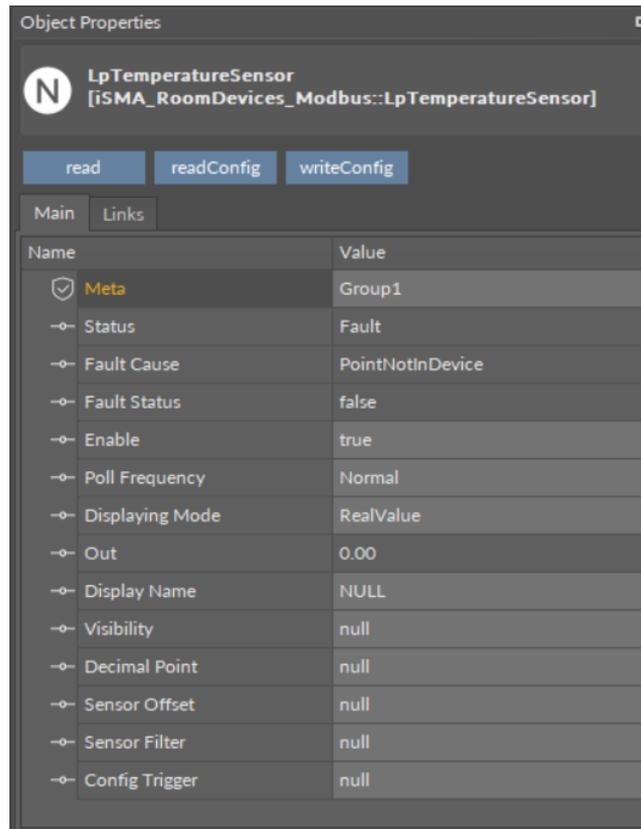
Figure 46. The LpTemperatureSensor component

The LpTemperatureSensor 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 LpTemperatureSensor 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).

9.10 Occupancy

This page is a source for (V1.7) Occupancy (AAC).

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

The Occupancy component is responsible for configuring occupancy settings in the panel.
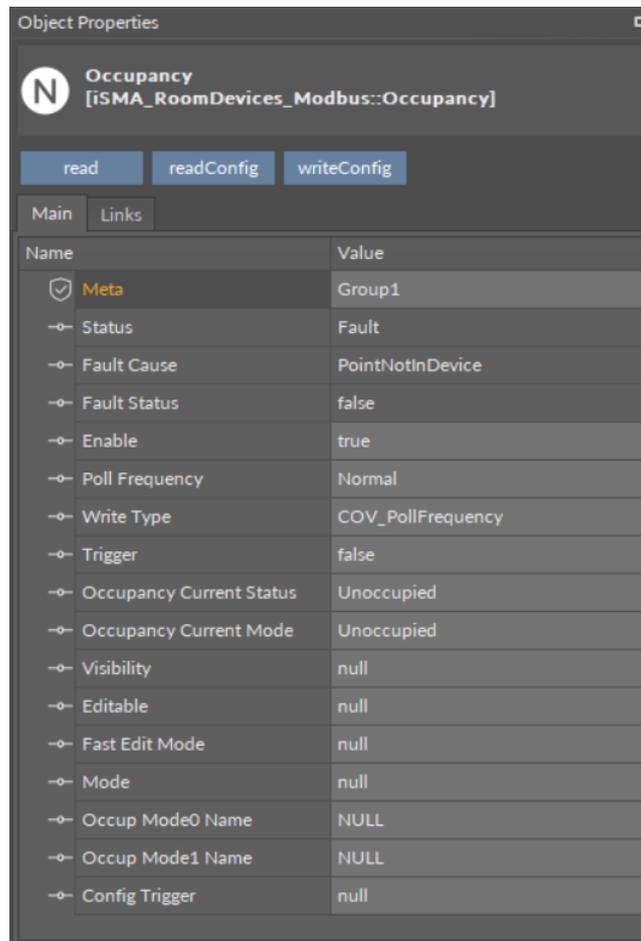
Figure 47. The Occupancy component

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).

9.11 TemperatureSetpoint

This page is a source for (V1.7) TemperatureSetpoint (AAC).

> **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. The Occupancy 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).

10 iSMA Module

The iSMA Modules is an extension of the ModbusAsyncNetwork designed to easily serve iSMA devices series like MIX modules, MINI modules, and Wireless Module using Modbus RTU/ASCII protocol. The iSMA Modules kit contains prepared components for serving physical inputs, outputs, and configuration parameters. The iSMA Modules kit consists of 4 types of components:

- ModbusNetwork;
- iSMADevice;
- iSMAIOPoints;
- iSMADeviceConfig.

10.1 iSMADevice

> WARNING! The iSMA Device must be placed under the ModbusNetwork component from the ModbusAsyncNetwork kit.

The iSMADevice is a component designed to cooperate with iSMA devices hardware. This component has built-in parameters to work with all iSMA devices in the ModbusAsyncNetwork.
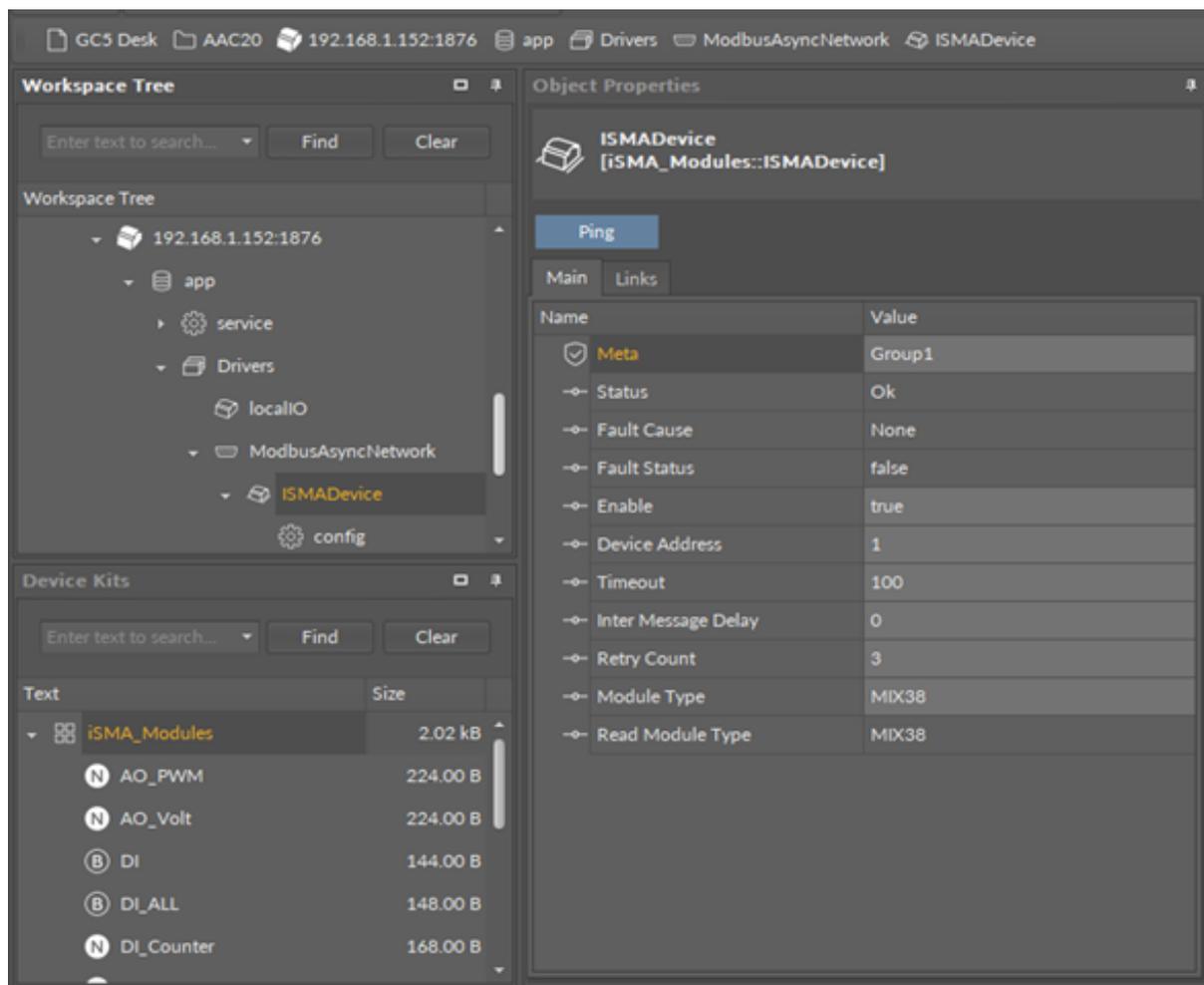
Figure 49. iSMADevice component

10.2 iSMADeviceConfig

The iSMADeviceConfig is a special component dedicated to set up the iSMA series device parameters. Adding and removing of the iSMADeviceConfig component is done by the Module Type slot in the iSMADevice component. To add a configuration component a proper module type is selected from a drop-down list. If the connection is established, the module type is displayed in the Read Module Type slot.

> WARNING! This component has no auto-refresh option. To read or to write the device configuration the component action must be used. It is recommended to read module configuration before changing parameters.
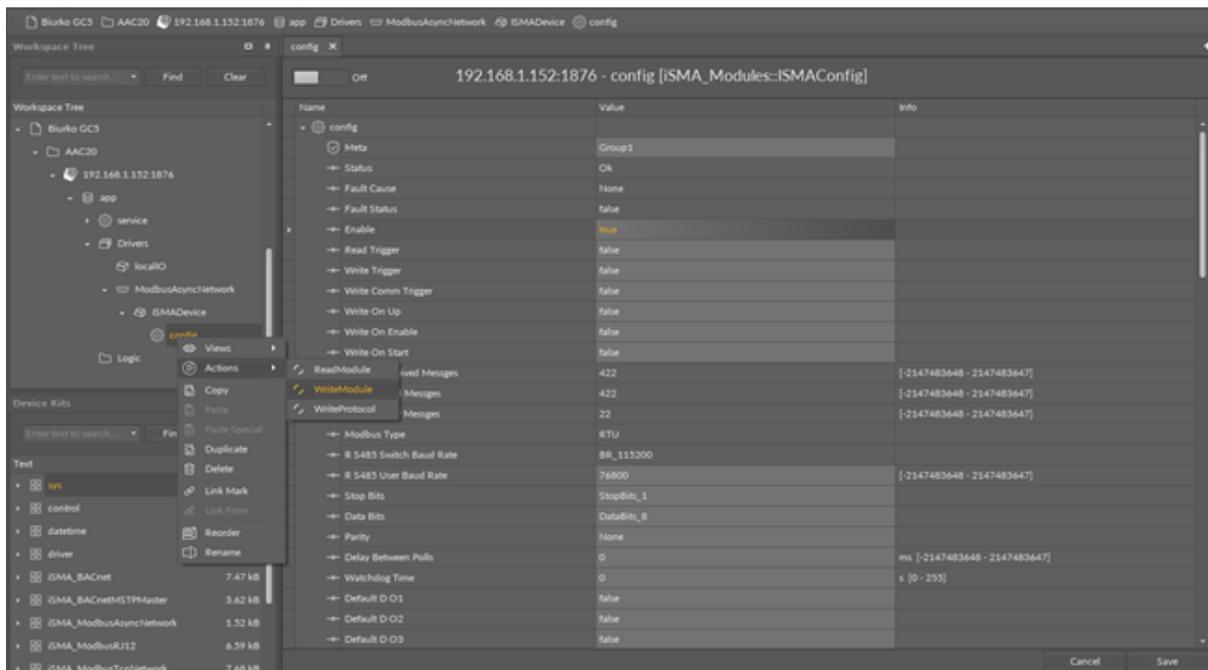
Figure 50. iSMAConfig component

10.3 Digital Inputs Components

In the iSMA Module kit there are available two types of components to read digital inputs:

- DI: reads individual digital inputs (input number is selected in the component property sheet);
- DI_ALL: reads all digital inputs using one register.

Note: DI_ALL component has 12 input slots (DI component - up to 12 inputs), which correspond to the largest module iSMA-B-MIX38. Using a module with a smaller number of inputs makes the surplus inputs inactive and always in a false state.
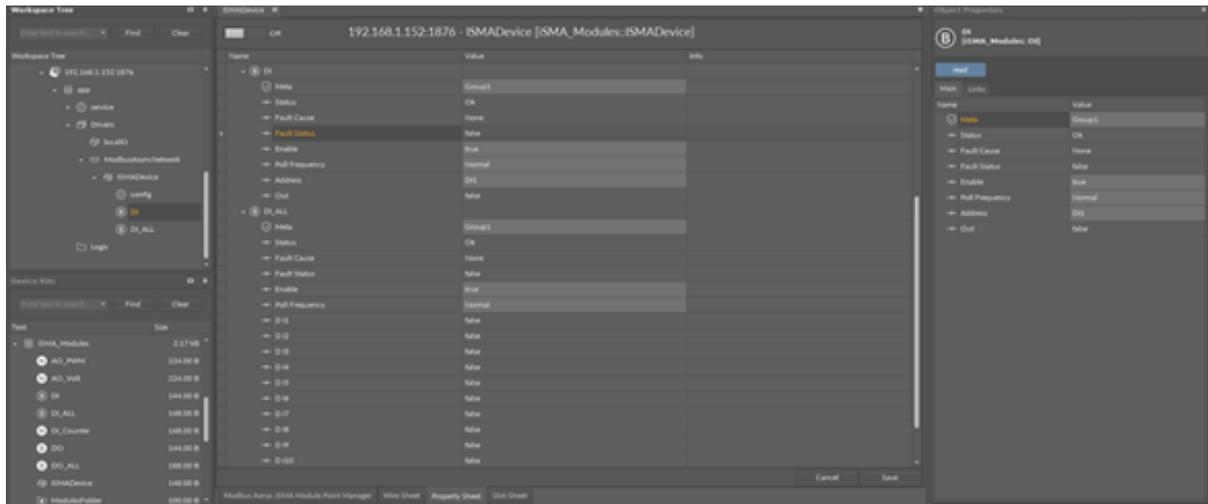
Figure 51. iSMA Module DI components

10.4 Digital Outputs Components

In the iSMA Module kit there are available two types of components to read/write digital outputs:

· **DO:** reads/writes individual digital outputs (output number is selected in component property sheet);
· **DO_ALL:** reads all digital outputs using one register.

Note: DO_ALL component has 12 input slots (DO component - up to 12 outputs), which correspond to the largest module iSMA-B-MIX38. Using module with a smaller output number makes the surplus outputs inactive.
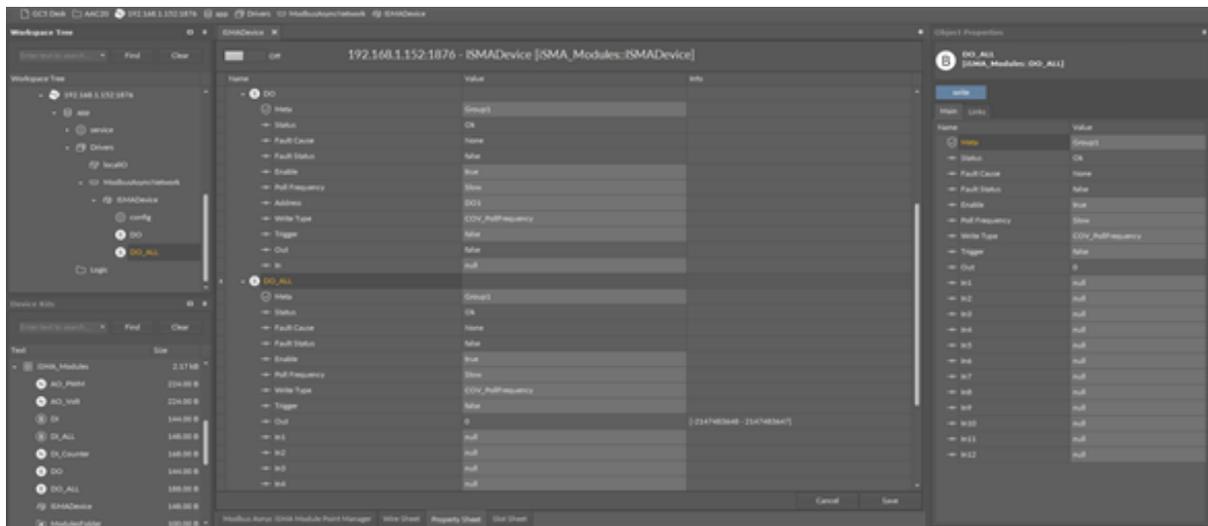
Figure 52. iSMA Module DO components

10.5 Universal Inputs Components

In the iSMA Module kit there are available five types of components to read universal inputs:

· **UI_Temp:** reads a temperature value from the NTC sensor connected to the input;
· **UI_Res:** reads a resistance value between the universal input and G0;
· **UI_Volt:** reads a voltage value between the universal input and G0;
· **UI_DI:** reads a Boolean value (dry contact) from a single universal input;

- **UI_DI_ALL:** reads a Boolean value (dry contact) from all universal inputs in one register.
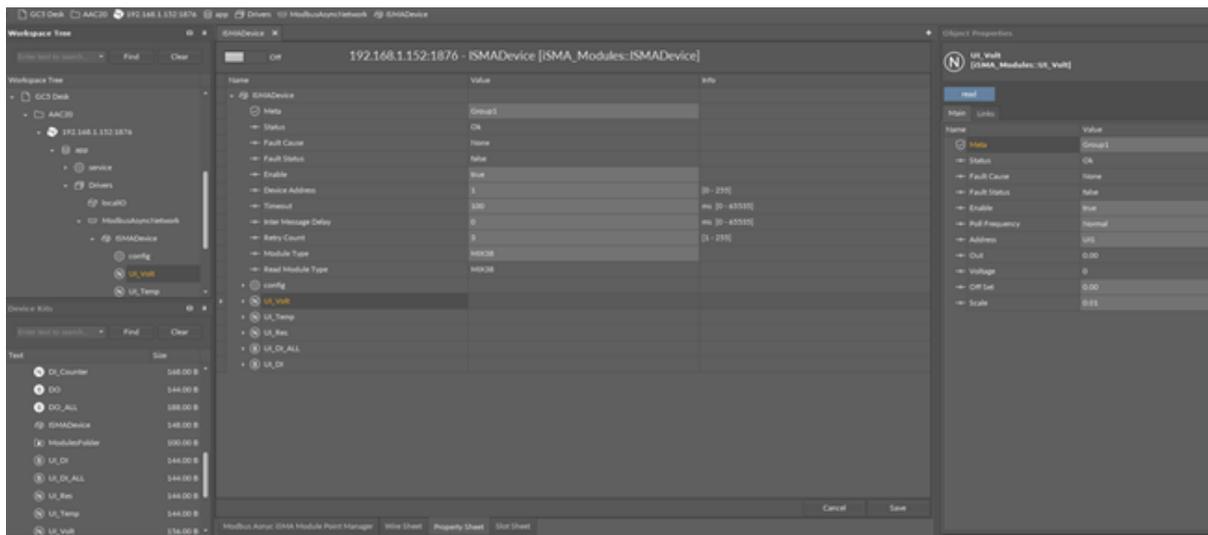
Figure 53. iSMA Module UI components

The universal inputs configuration (sensor type, voltage measurement, filter time, and resolution) is done in the iSMADeviceConfig component.

The input number is selected in the component property sheet, sensor type in module config component.

Note: UI components have 8 inputs, which correspond to the largest module iSMA-B-MIX38 and Mini iSMA-B-8U. Using module with smaller input number causes that inputs above module inputs number are inactive and always have 0 value.

10.6 Analog Outputs Components

In the iSMA Module kit there are available two types of component to read/write device analog outputs:

- **AO_Volt:** to set up a voltage signal (0-10000mV) on the analog output;
- **AO_PWM:** to set up a PWM signal (0-100%).

Selection, if output works in voltage or PWM mode, is made in the module config component.
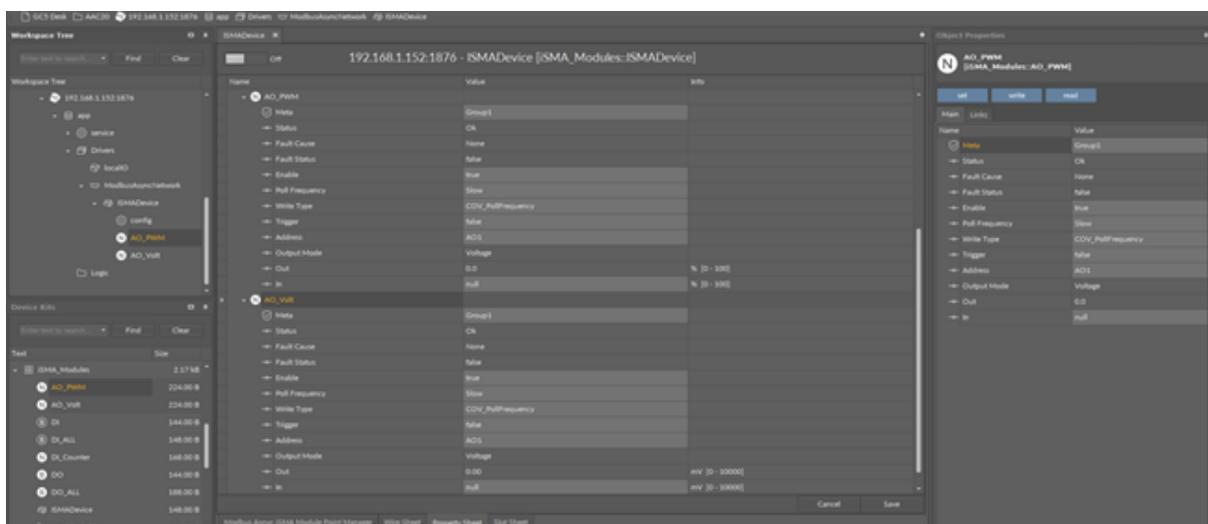
Figure 54. iSMA Module AO components

The input number is selected in the component property sheet, sensor type–in the module config component. Notice that AO components have 6 inputs, which correspond to the largest module iSMA-B-MIX38. Using a module with a smaller output number makes the surplus outputs inactive.

10.7 ModbusFolder (iSMAModule)

The ModbusFolder is a component which groups and organizes the Modbus points components. The ModbusFolder has the Description Slot, where up to 32 characters may be inserted.

10.8 iSMAFolder

The iSMAFolder is a component which groups and organizes the iSMA Module I/O point components. Because of Sedona components, names are limited to 7 characters, ModbusFolder has a Description Slot which can use up to 32 characters.

11 List of Modbus Registers

Modbus Address	Decimal Address	Hex Address	Register Name	Access	Description
40001	0	0x00	VERSION AND TYPE	Read-only	Controller firmware version and type
40002	1	0x01	HARDWARE VERSION	Read-only	Controller Hardware version
40003	2	0x02	MODBUS ADDRESS	Read&write Memory	Controller Modbus TCP/IP slave address
40004	3	0x03	UPTIME_LO	Read-only	
40005	4	0x04	UPTIME_HI	Read-only	
40008	7	0x07	COM2_STEADY_TIME	Read & Write Memory	
40009	8	0x08	RS485 BAUD RATE	Read&write Memory	Transmission baud rate is defined by the user calculated using the formula: The default value is 11520 (115200 bps)
40010	9	0x09	RS485 STOP BITS	Read&write Memory	Supported values are 1 and 2 The default value is 1
40011	10	0x0A	RS485 DATA BITS	Read&write Memory	Supported values are 7 and 8 The default value is 7
40012	11	0x0B	RS485 PARITY BIT	Read&write Memory	The default value is 0 (no parity) Allowed values:

Allowed values (for RS485 PARITY BIT):

Value	Description
0 (default)	none
1	Odd
2	Even
3	Always 1
4	Always 0

Modbus Address	Decimal Address	Hex Address	Register Name	Access	Description
40014	13	0x0D	RS485 MODBUS PROTOCOL TYPE	Read&write Memory	Protocol Type (40014) 0 – RTU, 1 – ASCII The default value is 0 - RTU.
40016	15	0x0F	STATE OF DIGITAL INPUTS	Read-only	State of digital inputs
40017	16	0x10	STATE OF UNIVERSAL INPUTS WORKING AS DIGITAL INPUTS	Read-only	Status of universal inputs working as digital inputs
40018	17	0x11	STATE OF DIGITAL OUTPUTS	Read&write Memory	State of digital outputs
40019	18	0x12	STATE OF ANALOG OUTPUTS WORKING AS DIGITAL OUTPUTS	Read&write Memory	State of analog outputs working as digital outputs
40022	21	0x15	COUNTER RESET	Read&write Memory	Set bit in register to reset corresponding counter.
40023	22	0x16	COUNTER 1 LSB	Read&write Memory	32-bit counters for each digital input counting pulses.
40024	23	0x17	COUNTER 1 MSB		
40025	24	0x18	COUNTER 2 LSB	Read&write Memory	
40026	25	0x19	COUNTER 2 MSB		
40027	26	0x1A	COUNTER 3 LSB	Read&write Memory	
40028	27	0x1B	COUNTER 3 MSB		
40029	28	0x1C	COUNTER 4 LSB	Read&write Memory	
40030	29	0x1D	COUNTER 4 MSB		
40046	45	0x2D	COUNTER 12 MSB		

Modbus Address	Decimal Address	Hex Address	Register Name	Access	Description
40071	70	0x46	UNIVERSAL INPUT VOLTAGE 1	Read-only	Voltage measurement value is expressed in mV. Formula for the current measurements: I=U/200 where: U – register value, 200 – value of attached resistor Temperature is expressed in Celsius degrees * 10 For a result, divide the registry value by 10. Selection of the type sensor is done using UNIVERSAL INPUT CONFIGURATION register from 40151 to 40158 for each input separately
40072	71	0x47	UNIVERSAL INPUT TEMPERATURE 1	Read-only	
40073	72	0x48	UNIVERSAL INPUT VOLTAGE 2	Read-only	
40074	73	0x49	UNIVERSAL INPUT TEMPERATURE 2	Read-only	
40075	74	0x4A	UNIVERSAL INPUT VOLTAGE 3	Read-only	
40076	75	0x4B	UNIVERSAL INPUT TEMPERATURE 3	Read-only	
40077	76	0x4C	UNIVERSAL INPUT VOLTAGE 4	Read-only	
40078	77	0x4D	UNIVERSAL INPUT TEMPERATURE 4	Read-only	
40079	78	0x4E	UNIVERSAL INPUT VOLTAGE 5	Read-only	
40080	79	0x4F	UNIVERSAL INPUT TEMPERATURE 5	Read-only	
40081	80	0x50	UNIVERSAL INPUT VOLTAGE 6	Read-only	
40082	81	0x51	UNIVERSAL INPUT TEMPERATURE 6	Read-only	

Modbus Address	Decimal Address	Hex Address	Register Name	Access	Description
40083	82	0x52	UNIVERSAL INPUT VOLTAGE 7	Read-only	
40084	83	0x53	UNIVERSAL INPUT TEMPERATURE 7	Read-only	
40085	84	0x54	UNIVERSAL INPUT VOLTAGE 8	Read-only	
40086	85	0x55	UNIVERSAL INPUT TEMPERATURE 8	Read-only	
40087	86	0x56	UNIVERSAL INPUT VOLTAGE 1	Read-only	
40088	87	0x57	UNIVERSAL INPUT VOLTAGE 2	Read-only	
40089	88	0x58	UNIVERSAL INPUT VOLTAGE 3	Read-only	
40090	89	0x59	UNIVERSAL INPUT VOLTAGE 4	Read-only	
40091	90	0x5A	UNIVERSAL INPUT VOLTAGE 5	Read-only	
40092	91	0x5B	UNIVERSAL INPUT VOLTAGE 6	Read-only	
40093	92	0x5C	UNIVERSAL INPUT VOLTAGE 7	Read-only	
40094	93	0x5D	UNIVERSAL INPUT VOLTAGE 8	Read-only	
40095	94	0x5E	UNIVERSAL INPUT TEMPERATURE 1	Read-only	
40096	95	0x5F	UNIVERSAL INPUT TEMPERATURE 2	Read-only	
40097	96	0x60	UNIVERSAL INPUT TEMPERATURE 3	Read-only	
40098	97	0x61	UNIVERSAL INPUT TEMPERATURE 4	Read-only	
40099	98	0x62	UNIVERSAL INPUT TEMPERATURE 5	Read-only	
40100	99	0x63	UNIVERSAL INPUT TEMPERATURE 6	Read-only	

Modbus Address	Decimal Address	Hex Address	Register Name	Access	Description
40101	100	0x64	UNIVERSAL INPUT TEMPERATURE 7	Read-only	
40102	101	0x65	UNIVERSAL INPUT TEMPERATURE 8	Read-only	
40103	102	0x66	RESISTIVE INPUT 1 LSB	Read-only	Resistance measurement result expressed in Ω. Value range from 0 Ω to 1 000 000 Ω. Note: In PT1000 or NI1000 input working type the reading accuracy increase and the register value is multiply by 10
40104	103	0x67	RESISTIVE INPUT 1 MSB	Read-only	
40105	104	0x68	RESISTIVE INPUT 2 LSB	Read-only	
40106	105	0x69	RESISTIVE INPUT 2 MSB	Read-only	
40107	106	0x6A	RESISTIVE INPUT 3 LSB	Read-only	
40108	107	0x6B	RESISTIVE INPUT 3 MSB	Read-only	
40109	108	0x6C	RESISTIVE INPUT 4 LSB	Read-only	
40110	109	0x6D	RESISTIVE INPUT 4 MSB	Read-only	
40111	110	0x6E	RESISTIVE INPUT 5 LSB	Read-only	
40112	111	0x6F	RESISTIVE INPUT 5 MSB	Read-only	
40113	112	0x70	RESISTIVE INPUT 6 LSB	Read-only	
40114	113	0x71	RESISTIVE INPUT 6 MSB	Read-only	
40115	114	0x72	RESISTIVE INPUT 7 LSB	Read-only	

Modbus Address	Decimal Address	Hex Address	Register Name	Access	Description
40116	115	0x73	RESISTIVE INPUT 7 MSB	Read-only	
40117	116	0x74	RESISTIVE INPUT 8 LSB	Read-only	
40118	117	0x75	RESISTIVE INPUT 8 MSB	Read-only	
40121	120	0x78	VALUE OF ANALOG OUTPUT 1	Read&write	The voltage at the analog outputs are given in the mV range from 0 to 10000 mV
40122	121	0x79	VALUE OF ANALOG OUTPUT 2	Read&write	
40123	122	0x7A	VALUE OF ANALOG OUTPUT 3	Read&write	
40124	123	0x7B	VALUE OF ANALOG OUTPUT 4	Read&write	
40125	124	0x7C	VALUE OF ANALOG OUTPUT 5	Read&write	
40126	125	0x7D	VALUE OF ANALOG OUTPUT 6	Read&write	
40134	133	0x85	BACNET_DEVICE_ID_LO	Read & Write Memory	
40135	134	0x86	BACNET_DEVICE_ID_HI	Read & Write Memory	
40136	135	0x87	COM1_BAUD_RATE	Read & Write Memory	
40137	136	0x88	COM1_STOP_BITS	Read & Write Memory	
40138	137	0x89	COM1_DATA_BITS	Read & Write Memory	
40139	138	0x8A	COM1_PARITY_BITS	Read & Write Memory	
40140	139	0x8B	COM1_RESPONSE_DELAY	Read & Write Memory	
40141	140	0x8C	WATCHDOG TIME	Read&write Memory	Time in second before watchdog reset in case no transmission. A 0 value

Modbus Address	Decimal Address	Hex Address	Register Name	Access	Description
					disables watchdog. The default value is 0 s.
40143	142	0x8E	DEFAULT STATE OF DIGITAL OUTPUTS	Read&write Memory	State of digital outputs assigned at the start of the module and watchdog reset. The default value is 0.
40144	143	0x8F	DEFAULT STATE OF ANALOG OUTPUTS (DIGITAL)	Read&write Memory	State of analog outputs assigned at the start of the module and watchdog reset. The default value is 0.
40145	144	0x90	DEFAULT STATE OF ANALOG OUTPUT 1	Read&write Memory	In the registers is stored value in mV of voltage that appears at the analog output after power on or watchdog reset. The default value is 0.
40146	145	0x91	DEFAULT STATE OF ANALOG OUTPUT 2	Read&write Memory	
40147	146	0x92	DEFAULT STATE OF ANALOG OUTPUT 3	Read&write Memory	
40148	147	0x93	DEFAULT STATE OF ANALOG OUTPUT 4	Read&write Memory	
40149	148	0x94	DEFAULT STATE OF ANALOG OUTPUT 5	Read&write Memory	
40150	149	0x95	DEFAULT STATE OF ANALOG OUTPUT 6	Read&write Memory	
40151	150	0x96	UNIVERSAL INPUT 1 CONFIGURATION	Read&write Memory	Configuration of universal input and type of temperature sensor. The default value is 1.

Value	Description / Sensor
0	Resistance measurement off
1	10K3A1 NTC
2	10K4A1 NTC

Modbus Address	Decimal Address	Hex Address	Register Name	Access	Description	
					3	10K NTC Carel
					4	20K6A1 NTC
					5	2,2K3A1 NTC B=3975K
					6	3K3A1 NTC
					7	30K6A1 NTC
					8	SIE1
					9	TAC1
					10	SAT1
					16	Pt1000
					17	Ni1000
					+128	Voltage measurement off
40152	151	0x97	UNIVERSAL INPUT 2 CONFIGURATION	Read&write Memory		
40153	152	0x98	UNIVERSAL INPUT 3 CONFIGURATION	Read&write Memory		
40154	153	0x99	UNIVERSAL INPUT 4 CONFIGURATION	Read&write Memory		
40155	154	0x9A	UNIVERSAL INPUT 5 CONFIGURATION	Read&write Memory		
40156	155	0x9B	UNIVERSAL INPUT 6 CONFIGURATION	Read&write Memory		
40157	156	0x9C	UNIVERSAL INPUT 7 CONFIGURATION	Read&write Memory		
40158	157	0x9D	UNIVERSAL INPUT 8 CONFIGURATION	Read&write Memory		
40159	158	0x9E	FILTER TIME CONSTANT OF THE UNIVERSAL INPUT 1	Read&write Memory	Filter time constant, expressed in seconds in the range from 0 to 60 seconds. A value of 0 disables the	

Modbus Address	Decimal Address	Hex Address	Register Name	Access	Description
					filter. The default value is 2 s. Filter time constant, expressed in seconds in the range from 0 to 60 seconds. A value of 0 disables the filter. The default value is 2 s.
40160	159	0x9F	FILTER TIME CONSTANT OF THE UNIVERSAL INPUT 2	Read&write Memory	
40161	160	0xA0	FILTER TIME CONSTANT OF THE UNIVERSAL INPUT 3	Read&write Memory	
40162	161	0xA1	FILTER TIME CONSTANT OF THE UNIVERSAL INPUT 4	Read&write Memory	
40163	162	0xA2	FILTER TIME CONSTANT OF THE UNIVERSAL INPUT 5	Read&write Memory	
40164	163	0xA3	FILTER TIME CONSTANT OF THE UNIVERSAL INPUT 6	Read&write Memory	
40165	164	0xA4	FILTER TIME CONSTANT OF THE UNIVERSAL INPUT 7	Read&write Memory	
40166	165	0xA5	FILTER TIME CONSTANT OF THE UNIVERSAL INPUT 8	Read&write Memory	
40167	166	0xA6		Read&write Memory	Resolution of universal inputs. When bit is set measurement at

Modbus Address	Decimal Address	Hex Address	Register Name	Access	Description
			RESOLUTION OF THE UNIVERSAL INPUTS		corresponding input is done with 16-bit resolution. By default, all measurements are done with 12-bit resolution.
40168	167	0xA7	ANALOG OUTPUT 1 CONFIGURATION	Read&write Memory	By default, all measurements are done with 12-bit resolution. Configuring the mode of analog output according to the following table: Value / Description 0 (default) / Voltage output 0-10V 1 / PWM 1Hz 2 / PWM 10Hz 3 / PWM 100Hz 4 / PWM 0.1Hz 5 / PWM 0.01Hz
40169	168	0xA8	ANALOG OUTPUT 2 CONFIGURATION	Read&write Memory	
40170	169	0xA9	ANALOG OUTPUT 3 CONFIGURATION	Read&write Memory	
40171	170	0xAA	ANALOG OUTPUT 4 CONFIGURATION	Read&write Memory	
40172	171	0xAB	ANALOG OUTPUT 5 CONFIGURATION	Read&write Memory	
40173	172	0xAC	ANALOG OUTPUT 6 CONFIGURATION	Read&write Memory	

Table 2. List of Modbus registers