

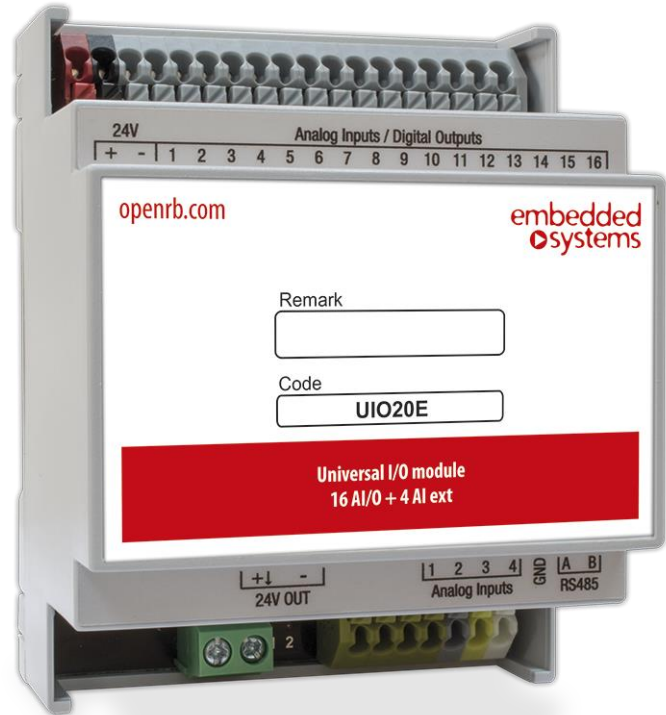
## Universal 20 channel I/O device

### ENG - Data sheet

Issue date 26.11.2021

### Application

Universal 20 channel IO device is designed to be used in meter reading applications or other building automation applications as an extension module to LogicMachine series devices. The configuration of the device is done through separate Modbus tab in LogicMachine. 4 channels can be used as analog 0-30V inputs, binary inputs. 16 channels can be used as analog 0-30V inputs, binary inputs, digital outputs. The device is designed for DIN-rail mounting and requires 4 DIN-units.







## Caution

### Security advice

The installation and assembly of electrical equipment may only be performed by skilled electrician. The devices must not be used in any relation with equipment that supports, directly or indirectly, human health or life or with application that can result danger of people, animals or real value

### Mounting advice

The devices are supplied in operational status. The cables connections included can be clamped to the housing if required.

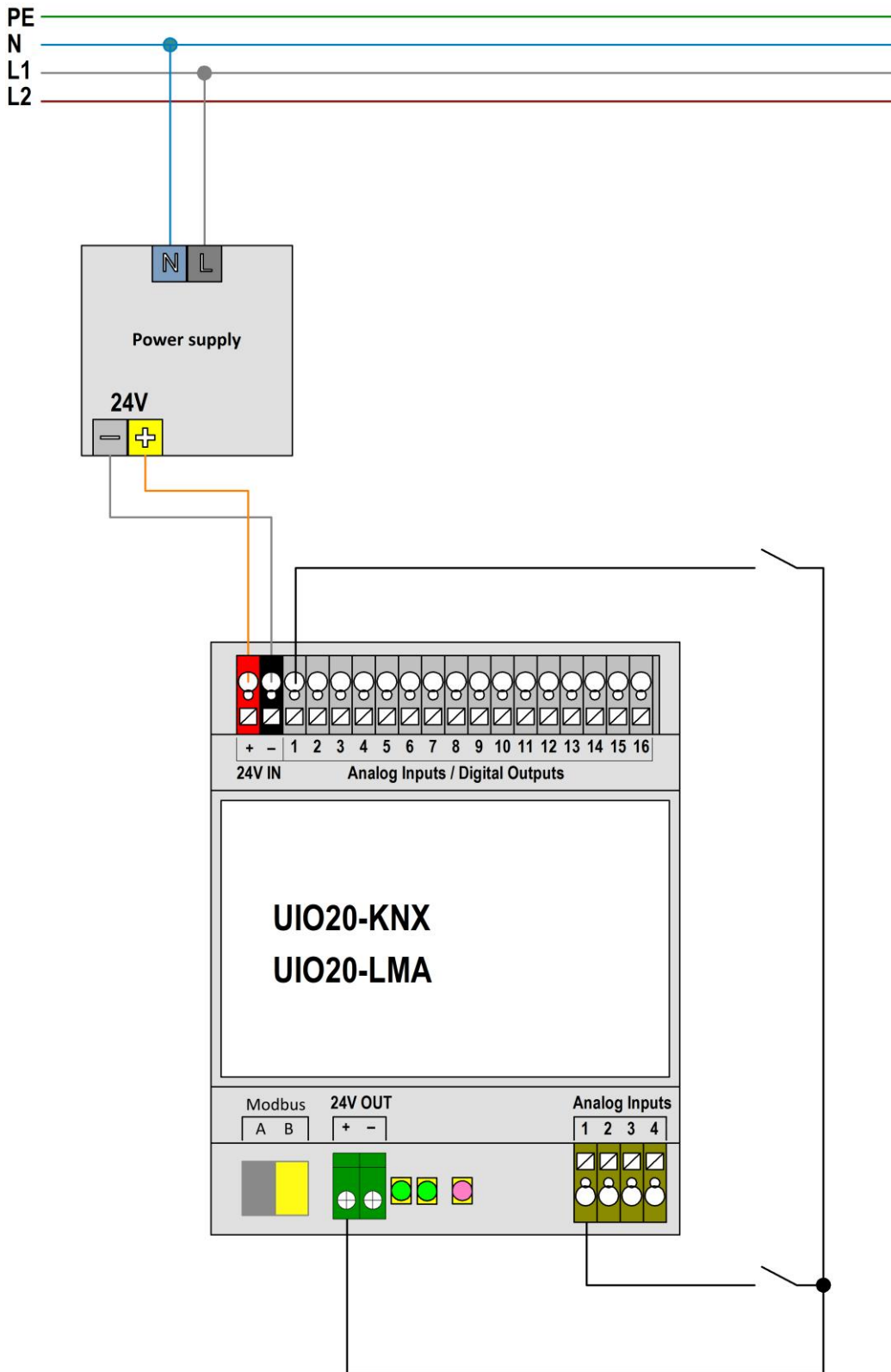
### Electrical connection

The devices are constructed for the operation of protective low voltage (SELV). Grounding of device not needed. When switching the power supply on or off, power surges must be avoided.

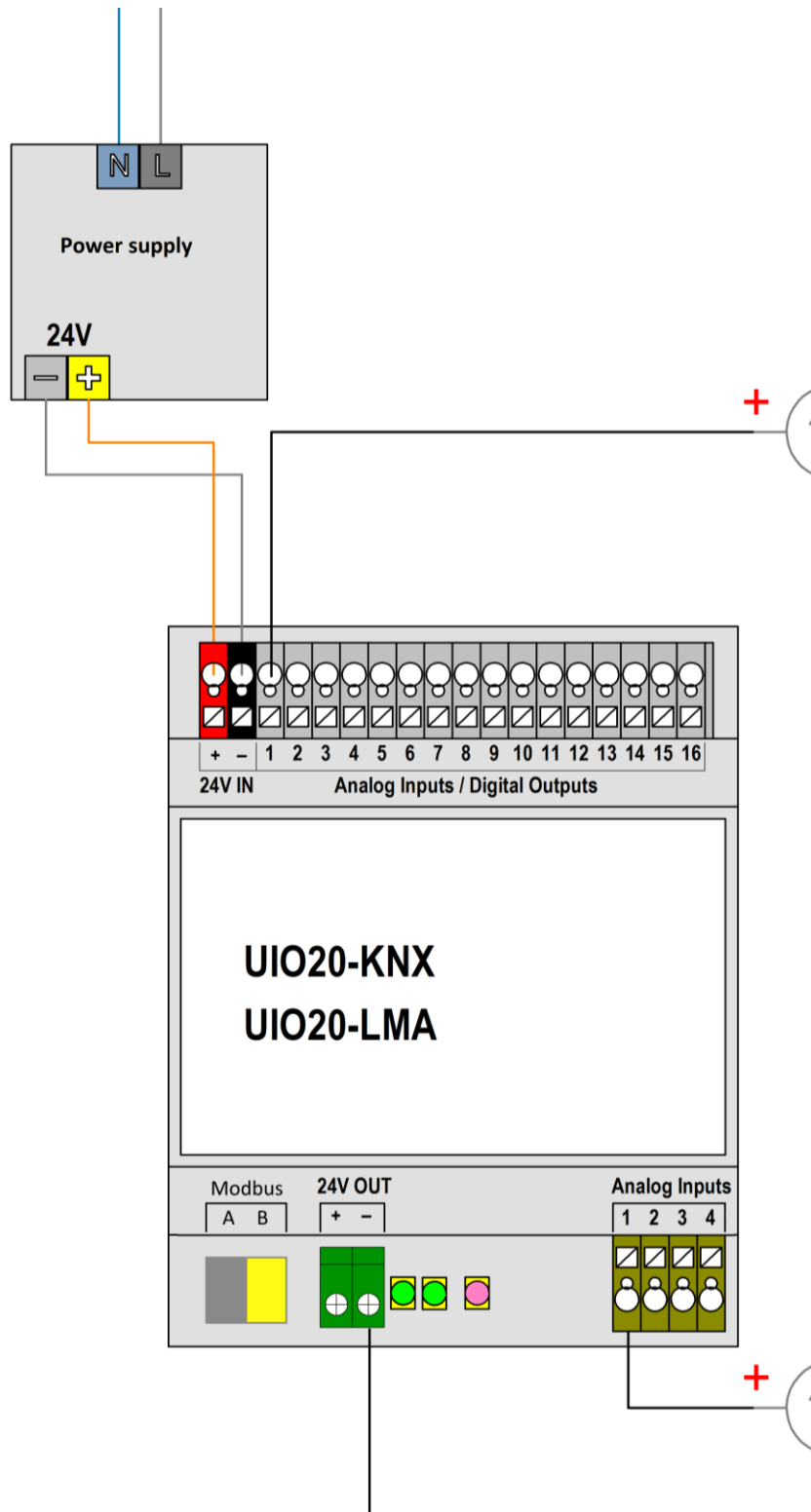


## Push-button connection

We do not recommend to interconnect pushbuttons to Modbus device because of the polling – the commands will not be executed in real-time.



## Voltage sensor connection



## Default address

247

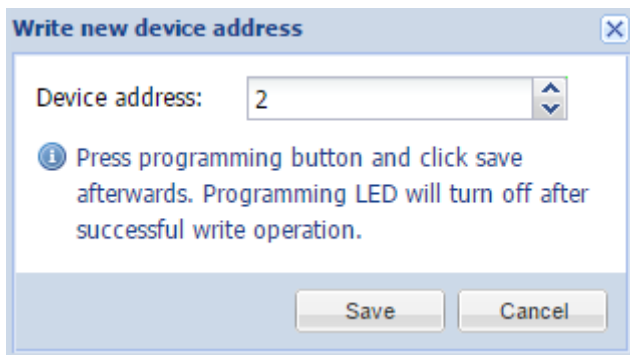
## Default RTU settings

*Baudrate:* 115200

*Parity:* Even

## Configuring UIO20 on LogicMachine

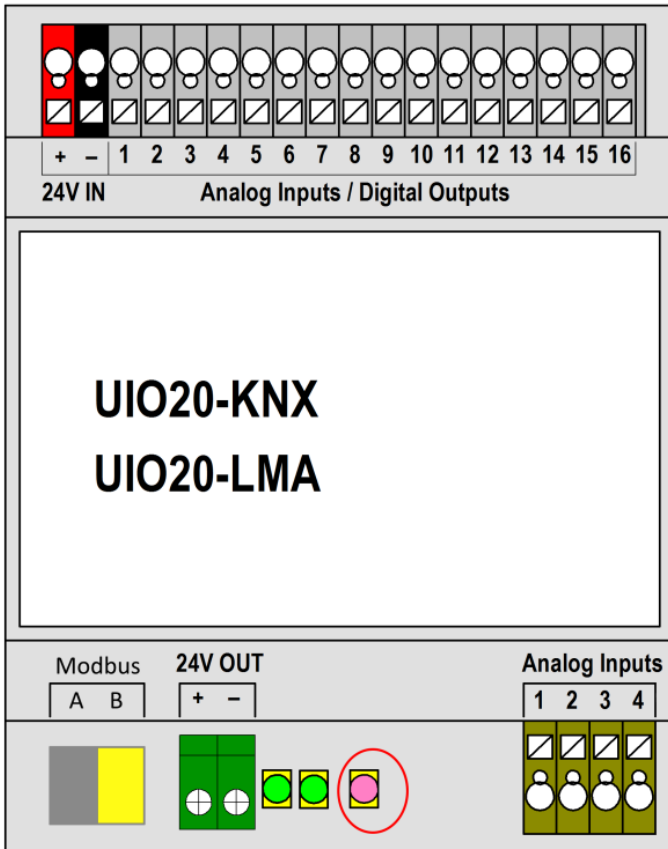
### *Program address for UIO20 Modbus device*



There is a separate Write address button to program address for UIO20 device. Press programming button and click save afterwards. Programming LED will turn off after successful write operation. When doing Write address, the device is programmed with respective RTU settings from the Modbus mapper.

Note that Baud-rate and Parity settings are written together with Device address. So in case you need to set the device to work on for example 9600 baud-rate, set it first in Modbus RTU Settings window and then execute write device address.





### UIO20 Modbus device profile

In LogicMachine Modbus devices are added as \*.json profile files. You can download UIO20.json here:

<http://openrb.com/wp-content/uploads/2015/06/UIO20.json>

A fragment from UIO20 profile file:

```
{
  "manufacturer": "Embedded Systems",
  "description": "Universal 16+4 I/O module",
  "mapping": [
    { "name": "Output 1", "bus_datatype": "bool", "type": "coil", "address": 0, "writable": 1 },
    { "name": "Input 1", "bus_datatype": "float16", "type": "inputregister", "address": 0,
      "value_multiplier": 0.001, "units": "V" }
  ]
}
```

**Name** – Object name, e.g. Output 2 (String, Required)

**Bus\_datatype** - KNX object data type, key from **dt** table, e.g. float32 (String/Number, Required)

**Type** – Modbus register type, possible values: **coil** **discreteinput** **register** **inputregister** (String, Required)

**Address** – Register address (0-based) (Number, Required)

**Writable** - Set to **true** to enable writing to register if **type** is either **coil** or **register** (Boolean)

**Datatype** – Modbus value data type. If set, conversion will be done automatically. Possible values: **uint16** **int16** **float16** **uint32** **int32** **float32** **uint64** **int64** **quad10k** **s10k** (String)

**Value\_delta** – New value is sent when the difference between previously sent value and current value is larger than delta. Defaults to 0 (send after each read) (Number)

**Value\_multiplier** – Multiply resulting value by the specified number, **value = value\_base + value \* value\_multiplier** (Number)

**Value\_bitmask** – Bit mask to apply, shifting is done automatically based on least significant 1 found in the mask (Number)

**Value\_nan** – Array of 16-bit integers. If specified and read operation returns the same array no further processing of value is done (Array)

**Value\_conv** – Apply one of built-in conversion functions (String, Internal)

**Value\_custom** – Name of a built-in enumeration or a list of **key -> value** mapping, resulting value will be 0 if key is not found (String/Object)

**Internal** – Not visible to user when set to **true**, should be used for **scale** registers (Boolean)

**Units** – KNX object units/suffix (String)

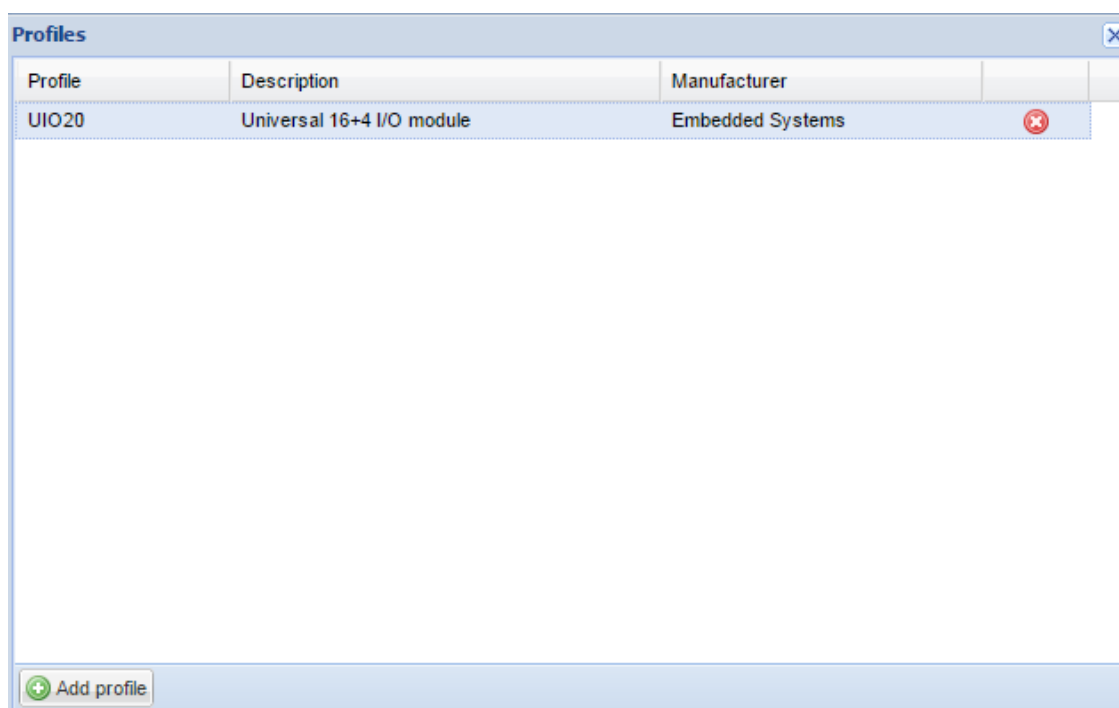
**Address\_scale** – Address of register containing value scale, **value = value \* 10 ^ scale** (Number)

**Read\_count** – Number of register to read at once (for devices that only support reading of a specific block of registers) (Number)

**Read\_swap** – Swap register order during conversion (endianness) (Boolean)

**Read\_offset** – Position of first register of data from the block of registers (0-based) (Number)

When the Modbus device profile file is created, upload it by clicking on *Profiles* button.

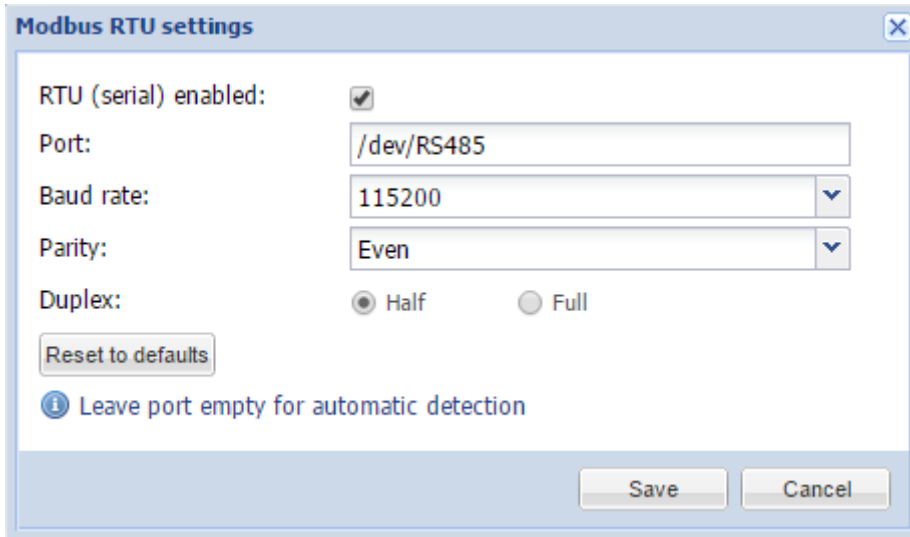


UIO20E



## Modbus RTU settings

If the communication is over Modbus RTU protocol (over RS-485 serial port), you should do base serial port settings by clicking on *RTU settings* button.



The screenshot shows a dialog box titled "Modbus RTU settings" with a close button in the top right corner. The settings are as follows:

- RTU (serial) enabled:
- Port:
- Baud rate:  (dropdown arrow)
- Parity:  (dropdown arrow)
- Duplex:  Half  Full
- Reset to defaults:
- Information icon:
- Buttons:

**RTU (serial) enabled** – define either RTU is enabled or not

**Port** – port name. In case of several RS-485 ports on the device, the name of the port is incremented by one, e.g. RS485-1, RS485-2, RS485-3 etc.

**Baud rate** – baud rate for the connection

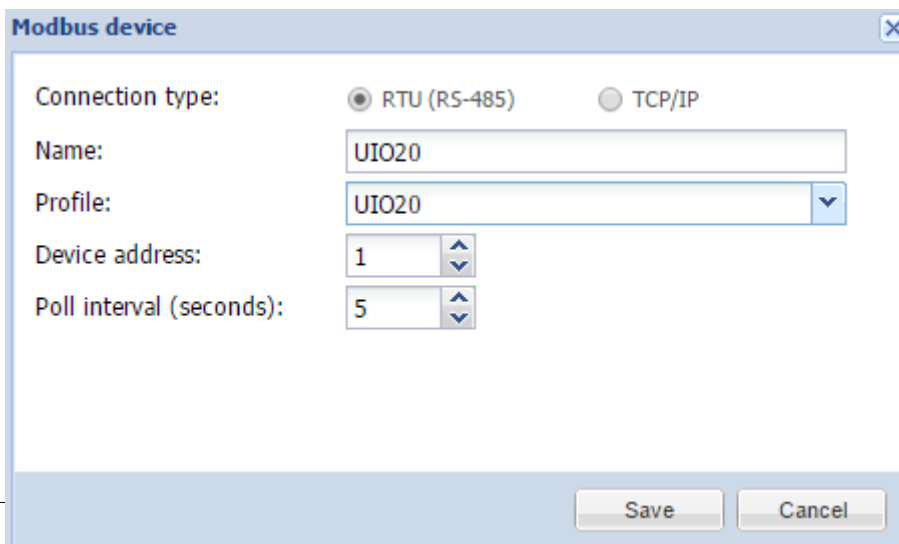
**Parity** – parity for the connection

**Duplex** – specify either it is *half* or *full* duplex

**Reset to defaults** – reset RTU settings to defaults

## Adding Modbus device

Once profiles are defined and RTU settings set, add Modbus device by clicking *Add device* button.



The screenshot shows a dialog box titled "Modbus device" with a close button in the top right corner. The settings are as follows:

- Connection type:  RTU (RS-485)  TCP/IP
- Name:
- Profile:  (dropdown arrow)
- Device address:  (up/down arrows)
- Poll interval (seconds):  (up/down arrows)
- Buttons:

**Connection type** – define either it is Modbus RTU or Modbus TCP connection

**Name** – name of the device


**Profile** – profile of the device

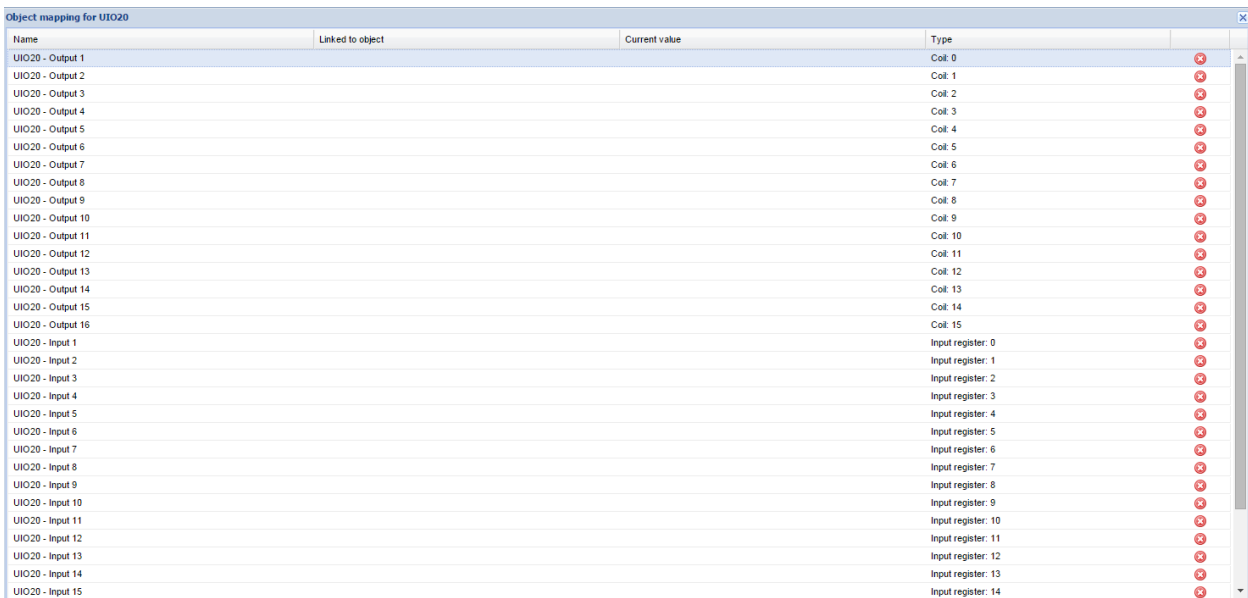
**Device address** – device address

**Poll interval (seconds)** – interval to poll the device

**IP** – IP address of the device in case Modbus TCP is used

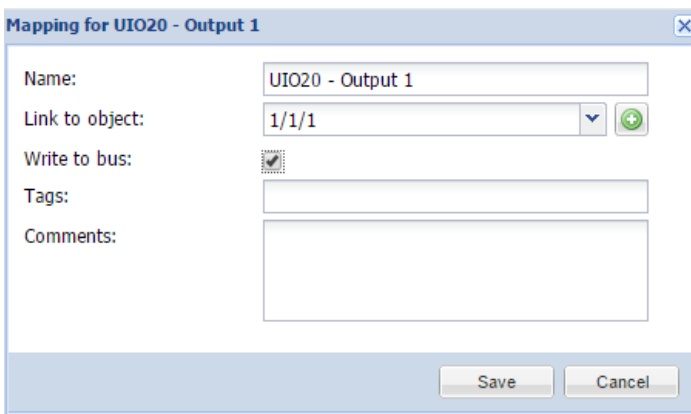
**Port** – Communication port of the device in case Modbus TCP is used

Once the device is added, you can do mapping to KNX addresses by clicking on  icon. First, you see a list of all objects on the Modbus device.



Name	Linked to object	Current value	Type
UIO20 - Output 1			Coil 0
UIO20 - Output 2			Coil 1
UIO20 - Output 3			Coil 2
UIO20 - Output 4			Coil 3
UIO20 - Output 5			Coil 4
UIO20 - Output 6			Coil 5
UIO20 - Output 7			Coil 6
UIO20 - Output 8			Coil 7
UIO20 - Output 9			Coil 8
UIO20 - Output 10			Coil 9
UIO20 - Output 11			Coil 10
UIO20 - Output 12			Coil 11
UIO20 - Output 13			Coil 12
UIO20 - Output 14			Coil 13
UIO20 - Output 15			Coil 14
UIO20 - Output 16			Coil 15
UIO20 - Input 1			Input register: 0
UIO20 - Input 2			Input register: 1
UIO20 - Input 3			Input register: 2
UIO20 - Input 4			Input register: 3
UIO20 - Input 5			Input register: 4
UIO20 - Input 6			Input register: 5
UIO20 - Input 7			Input register: 6
UIO20 - Input 8			Input register: 7
UIO20 - Input 9			Input register: 8
UIO20 - Input 10			Input register: 9
UIO20 - Input 11			Input register: 10
UIO20 - Input 12			Input register: 11
UIO20 - Input 13			Input register: 12
UIO20 - Input 14			Input register: 13
UIO20 - Input 15			Input register: 14

Click on specific object to do mapping.



**Mapping for UIO20 - Output 1**

Name: UIO20 - Output 1

Link to object: 1/1/1

Write to bus:

Tags:

Comments:

Save Cancel