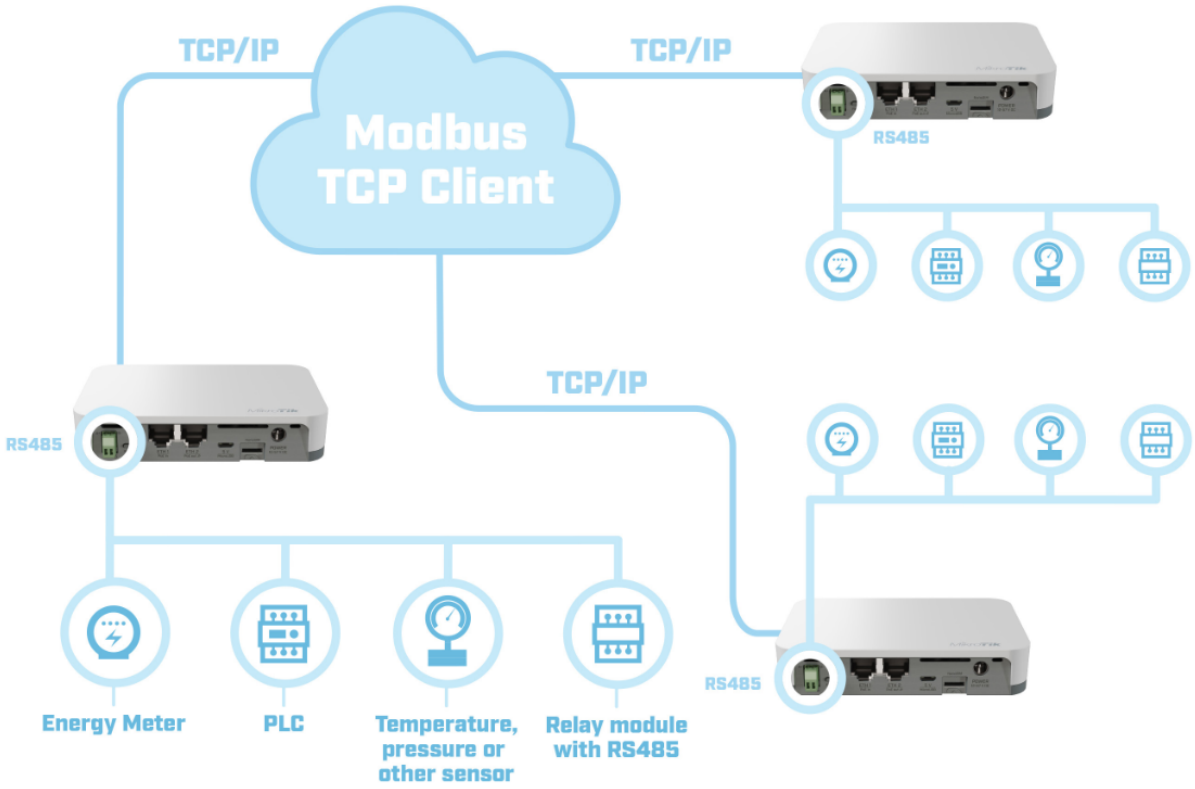
RB924i-2nD-BT5&BG77 Modbus configuration

- Introduction
  - Modbus TCP packet structure
- Configuration
- Sending function code commands via Modbus RTU
- Sending function code "3" command via Modbus TCP
- Modbus security

Introduction

One of the protocols that is widely used in IoT architectures is called Modbus.



You can find more information about this protocol by following this link.

The Modbus-supported device should be connected to the RS485 port. A two-wire connection (A+/B-) is supported by the device (a 2-pin terminal block).



It is important to note that both pins should not be mixed up. Inverting the "A" and "B" connections (incorrect polarisation) will result in communication failure.

Another thing to keep in mind is the Modbus cable distance. If the cable length is less than 50 meters - there should be no issues. If 50+ meters cable is used, 120 Ohm termination resistance should be installed at the end of the cable.

The KNOT supports:

- **Modbus TCP server scenario**, which allows you to use 3d-party Modbus clients to send function code commands to the KNOT's IP address using Modbus TCP/IP connection (TCP port 502). You can use any Modbus Client software (there are plenty on the internet) or it can be easily written /configured in Python. In this case, the KNOT "translates/bridges" Modbus TCP/IP requests to Modbus RTU.
- **Modbus RTU controller scenario** (available only starting with v**7.10**), which allows you to send function code commands directly from the RouterOS command line via Modbus/RTU.

Modbus TCP packet structure

You can find more information on Modbus packet structure using this link.

The query packet consists of Transaction ID, Protocol, Length, Unit ID, Function Code and Function Parameters fields and the maximum size is 260 bytes.

- Transaction ID - integer from 0 to 65535 used to identify the answer to the specific query;
- Protocol - 0x0000 stands for Modbus;
- Length - shows how many bytes will follow the Length field;
- Unit ID - usually called "Device address", 0x00-0xff. Each end-device in the specific system should have a unique ID;
- Function Code - describes which function will be called. Each device has multiple registers and these functions are used to interact with the registers. Codes are defined by the standards but the manufacturers can also implement their own functions with specific codes;
- Function Parameters/data - usually specifies register address, data length and data to be written in case the "write" function is used.

Default Unit ID, Function Codes and Parameters can be found in the datasheet for the specific product. Answer messages are structured similarly and should be described in the product-datasheet as well.

Configuration

Sub-menu: `/iot modbus`

Once the board is connected to the Modbus device, check "modbus" port configuration (baud-rate can be changed if the connected Modbus device expects another value):

```
[admin@MikroTik] > /port print
Columns: DEVICE, NAME, CHANNELS, USED-BY, BAUD-RATE
# DEVICE  NAME    CHANNELS  USED-BY  BAUD-RATE
0         modbus         1  modbus        9600
1 1-1     modem          4                9600
```

By default, a port with the name "*modbus"* is assigned to the Modbus service, but the service itself is disabled. In order to activate the "Modbus" service, you need to issue the command, as shown below:

```
[admin@MikroTik] > /iot modbus print
                disabled: yes
           hardware-port: modbus
                tcp-port: 502
                 timeout: 1000ms
  disable-security-rules: yes
[admin@MikroTik] > /iot modbus set disabled=no
```

Additionally, when you want to allow Modbus clients to communicate with the Modbus server using TCP(502)/IP protocol, you need to make sure that the IP address is configured accordingly and that this IP+502 TCP port is accessible from the outside (in case "slave" devices are connecting from the WAN side). You can find information on how to configure the firewall using the firewall manual.

Property	Description
disabled (yes | no; Default: yes)	To enables or disable Modbus functionality.
hardware-port (*list of string*; Default: modbus)	Assign a port to the service. Use the default "modbus" port.
tcp-port (*integer:0..4294967295*; Default: 502)	Specify the TCP port that the device will use for Modbus TCP communication.
timeout (*integer:0..1000*; Default: 1000)	Specify the timeout value, in milliseconds. Maximum allowed time during which a Modbus request is sent, and the reply is to be received. If the time between the request and the reply is longer than the configured value, the "failure" is returned.
disable-security-rules (yes | no; Default: yes)	To enable or disable security-rule feature.

Sending function code commands via Modbus RTU

> Available only starting with v**7.10.** This is meant for the "**Modbus RTU**" scenario. This feature allows sending function code commands directly from RouterOS. For the "Modbus TCP" scenario, you have to use 3d-party software to send function code commands.

Sub-menu: `/iot modbus transceive`

Property	Description
address (*integer:0..255*; Default:)	Specify device address or unit ID.
function (*integer:0..255*; Default:)	Specify the function code. For example: • function=3 → Read Holding Register function code; • function=6 → Write Single Holding Register function code.

data (*string, max length 504*; Default:)	Input data string that usually specifies register address and data to be sent. For example: • data=20000001 The data string is 4 bytes long. "0x2000" is the command to the register (2 bytes long) and "0x0001" (2 bytes long) is the device's address. See the Modbus device's specifications for the exact data string that needs to be sent.
values (*integer:0..4294967295*; Default:)	An alternative way to send the data (see **data** parameter above). In this case, each value specified represents 1 byte of the data payload/string. For example: • values=32,0,0,1 Equals to the output of data=20000001. 32(dec)=0x20(hex), 0(dec)=0x00(hex), 0(dec)=0x00(hex), 1(dec)=0x01(hex) → 0x20000001.

An example of sending function code 3 command using the "transceive" feature:

```
/iot modbus transceive address=1 function=3 data=20000001
   address: 1
  function: 3
      data: 01030164
    values: 100
      time: apr/27/2023 14:05:15
    status: ok
```

From the output above, we can tell that Modbus connected device replied with a value "100" or data "01030164" (01→ address; 03→ function code; 01→ number of bytes; 64 (hex to dec=100)→ the reply).

Sending function code "3" command via Modbus TCP

Specifically for function code 3 (this will not work for any other function codes), you can use an additional option, like shown below:

```
[admin@device] > {:local output [/iot modbus read-holding-registers slave-id=0x03 num-regs=0x1 reg-addr=0x0 as-
value once];:put [($output->"values")]}
2349
[admin@device] > {:local output [/iot modbus read-holding-registers slave-id=0x03 num-regs=0x5 reg-addr=0x0 as-
value once];:put [($output->"values")]}
2353;3;500;75;38
```

Parameters that can vary are "reg-addr" (address in the data string), "num-regs" (data string command to the register/number of registers), and "slave-id" (device address or unit ID). All of them depend on which values are supported by the MODBUS-connected device.

Modbus security

Available in the **7.12 beta7 Testing** firmware.

Sub-menu: `/iot modbus security-rules`

Modbus security feature allows you to set up specific rules, that can restrict or allow access to the Modbus-connected device. **This works only for Modbus TCP communication** (the rules will not work for Modbus RTU communication). It makes it possible to allow specific IP ranges/networks to have access to specific function code commands.

To enable the feature, input the command:

```
/iot modbus set disable-security-rules=no
```

After you enable the feature, add the rules themselves. For example:

```
/iot modbus security-rules add ip-range=0.0.0.0/0 allowed-function-codes=3,6
```

The rule above will allow any possible IP address to send function code 3 and 6 commands to the Modbus-connected device. Everything else is restricted.

Property	Description
ip-range (*Address/Netmask*; Default:)	Specify the IP address ranges (or networks) that will be allowed to send function code commands.
allowed-function-codes (*integer:0..4294967295*; Default:)	Specify the function codes that will be accessible only from the configured ip-range.