Modbus custom register block RutOS

[Main Page](#) > [General Information](#) > [Configuration Examples](#) > [Modbus](#) > **Modbus custom register block RutOS**
❌
☐

Contents

Introduction

The goal of this manual is to help configuring **Teltonika Networking Devices** to get additional custom Modbus registers for device monitoring via Modbus if default registers are not enough. Default Modbus registers can be found on our Wiki page: **Monitoring via Modbus**

Configuring Teltonika Networking Device

For this example we will be using script to store data into regfile from which then you can read data via Modbus. To achieve follow this steps: ❌

1. **Enable** router to be as **Modbus TCP slave** (**Services -> Modbus -> Modbus TCP Slave**);
2. **Enable Custom Register Block** in the webUI (**Services -> Modbus -> Modbus TCP Slave**);
3. Register file path: **tmp/regfile** (this is where new register will be stored);
4. First register number: **1025** (by default it is 1025 and recommend to keep it, if necessary it could be changed accordantly);
5. Register count: **32** (how much "space" there is given for additional Modbus registers, the more register you want to bigger count should be used).
6. Save & Apply

Creating Custom Modbus Registers

In order to create custom Modbus register which would allow you to read certain routers information via Modbus, you will need to create a script or any other method for putting certain information into **regfile**.

Script which would log certain data into Modbus register. To add new **script**, connect to the device via **SSH** and use:

```
$ vi /bin/extramodbus (this script is for this manual purpose, yours might be different)
```

Insert following **script** (this script is for this manual purpose, yours might be different) example:

```
#!/bin/ash
while true
do
date > /tmp/regfile  #where data will be stored
df | awk 'NR==3 {print $3}' >> /tmp/regfile
df | awk 'NR==4 {print $3}' >> /tmp/regfile
sleep 5
done
```

After saving change execution rights with this command:

```
$ chmod +x /bin/<scriptname> #example: chmod +x /bin/extramodbus
```

Run the script to start logging data to regfile, to do that simply type in your terminal: <script name> &. For example:

```
$ extramodbus &
```

And script will run in the background.

With this script we will be able to get routers local time and couple application values each 5 seconds. Data stored in **regfile (cat /tmp/regfile)** looks like this:

```
Wed Jun 17 15:11:29 EEST 2020
69
27
```

To view collected data in HEX execute this command via SSH:

```
$hexdump -v /tmp/regfile

0000000 5765 6420 4a75 6e20 3137 2031 353a 3132
0000010 3a32 3020 4545 5354 2032 3032 300a 2d36
0000020 380a 3137 0a00
0000025
```

Reading Created Custom Modbus Registers with Modbus TCP Master

To read data via Modbus you can use:

- ComTestPro program;
- Modbus TCP Master;
- QModMaster;

Modbus TCP Master configuration

- Add a new slave device

- Enable
- Type **name**
- Write **Slave ID** (1)
- And your **Slave's** (Router's) **IP**
- Write **request name** and add it

- Select **Data type** (ASCII)
- Select **function** (Read holding registers)
- Type **first register number** (1025)
- Type **register count** (15 in this scenario)
- Enable this request
- With Modbus TCP master you can read data just by using ASCII data type and interactive test button.

To determine how many registers to specify in the **Register count/Values** field you should divide character count by two. For example, if you have 28 characters you should input 14 to **Register count/Values** field.

Reading Created Custom Modbus Registers with QModMaster

For this example we will be using **QModMaster** program, which allows us to get Modbus data from device in **Hex**. So in order to get **date** from the router by reading **custom Modbus registers** we need to use following configuration:

- Protocol: Modbus TCP;
- IP Address: 192.168.1.1 (or 192.168.2.1 for TRB1 series) (router's LAN or WAN IP);
- Port: 502 (port number on which Modbus TCP Slave is listening);
- Device: 1
- Register: 1025;
- Registers: 15 (register's length).

With this configuration you should get routers **date** in **hex**. Below is the example of configuration and what reply we get from the router:

And as you can see we received a reply in hex:

4672 6920 4F63 7420 3134 2031 313A 4737 3A31 3820 4545 5354 2032 3032 320A

By using certain websites, like this: **hex converter** we can analyze the data by entering the received hex reply. After conversion we can see the routers date which in **ASCII** is:

Fri Oct 14 11:47:18 EEST 2022