webdyn

Flexitron group

TITAN

Application Note 27

Reading Modbus Devices and Sending Readings to the Cloud via HTTP

Reading Modbus Devices and Sending Readings to the Cloud via HTTP

1. Scenario Details

TITAN-based devices have all the typical functionalities of 4G/3G/2G routers, as well as a series of added features that make them one of the most feature-packed routers on the market.

One of the added features is the ability to interrogate Modbus RTU and TCP devices autonomously, subsequently sending the data to an HTTP, FTP or MQTT server.

As always, this capability will be illustrated using a simple example.

2. Description of the Example

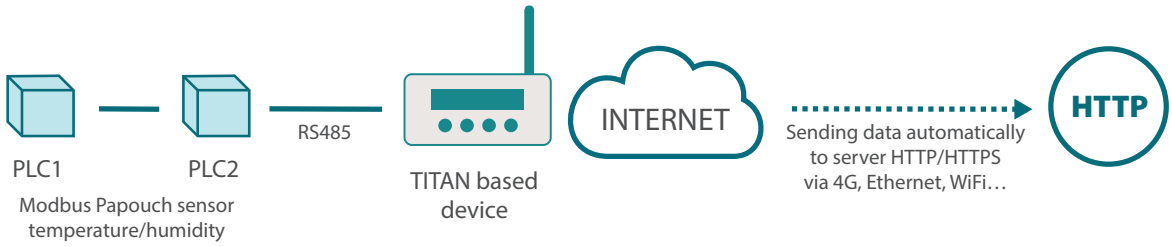
In this example, a TITAN-based device will be configured to collect, store and send the Modbus registers of 2 PLCs via HTTP. The readings will be done every minute.

The following Modbus registers must be read from PLC1:

1;10;11;12;55;56;69;70;72;73;74;75;76;77;78;79;80;100;101;102;103;104;105;106;107;108;109;120;121;122;123;124;130;131;132;133;152;153;154;160;161;162;163;164;165;166;170

The following registers must be read from PLC2:

10;11;12;13;14



This means we need to obtain a map of a number of registers, which are not always consecutive, from PLC1. PLC2 is easier as we only need 5 consecutive registers

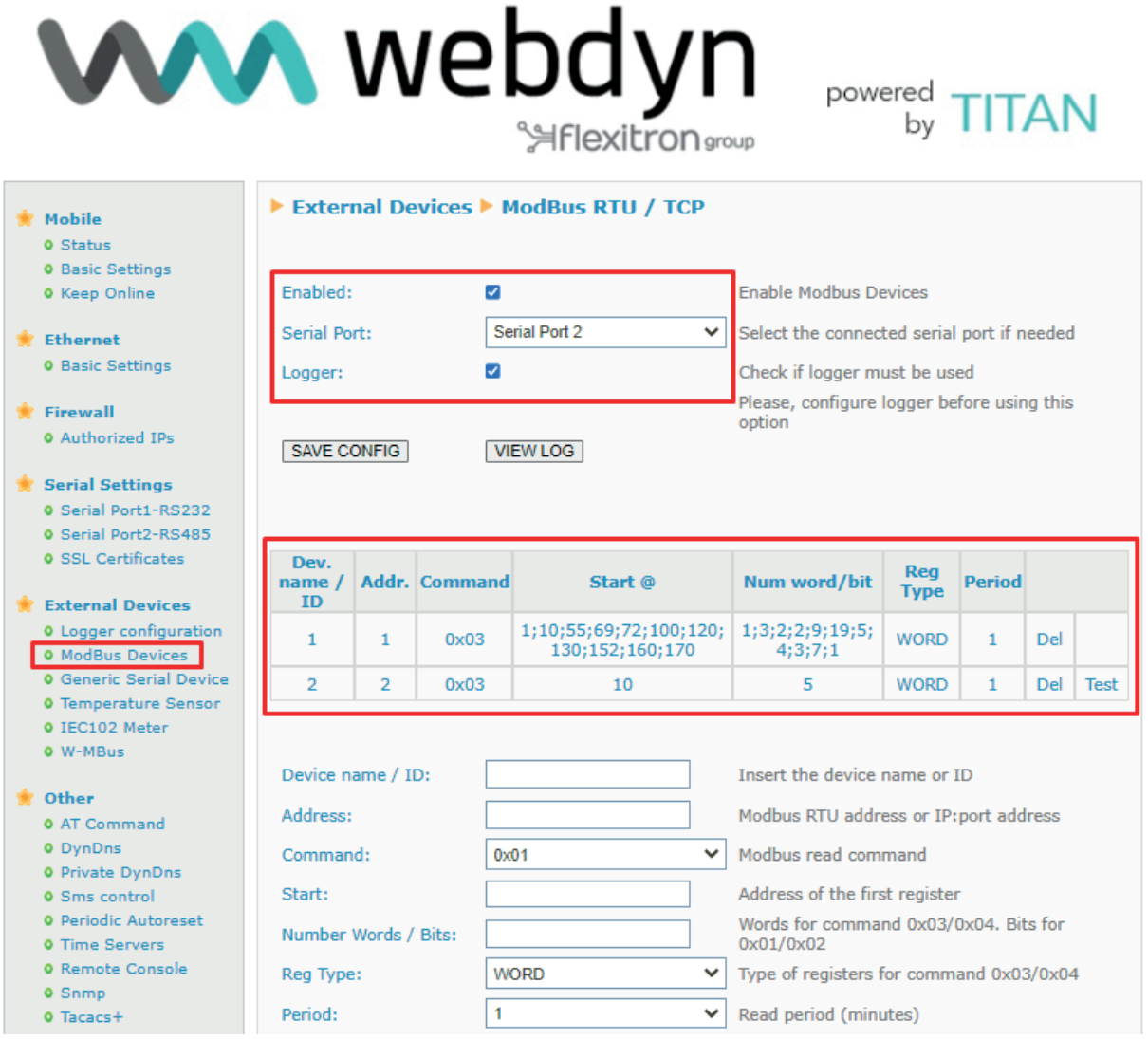
The PLCs are RS485 devices, so we will use Modbus RTU, but the scenario is also perfectly valid for Modbus TCP devices (with Ethernet) or a mix of both (Modbus TCP and Modbus RTU).

3. Configuring the Serial Port of the Titan-Based Device to which the Modbus Devices will be Connected

Let's imagine that the PLCs, which have RS485 ports, have the following serial port configuration: 9600,8,N,1. The first task is to configure the Serial Port2-RS485 section of the TITAN-based device. We will configure it as shown in the following figure:

4. Configuring the Titan to Read Modbus Devices

Click on the link: “External Devices > Modbus Devices” and configure the screen as shown below:



We want to read registers 10,11,12,13,14 from PLC 2. All we need to do is enter register 10 in the “Start” field and 5 in the “Number Words” field (as we want to read 5 registers, from 10 to 14).

PLC1 is more complex, since we have a non-consecutive register map. The different blocks of registers we need to read will therefore be separated by “;” (semicolons). This means if we want to read registers:

1;10;11;12;55;56;69;70;72;73;74;75;76;77;78;79;80;100;101;102;103;104;105;106;107;108;109;120;121;122;123;124;130;131;132;133;152;153;154;160;161;162;163;164;165;166;170

We must enter the following in the Start field (the initial register of each block):

1;10;55;69;72;100;120;130;152;160;170

And, in the "Number Words" field (the number of registers to be read from each block):

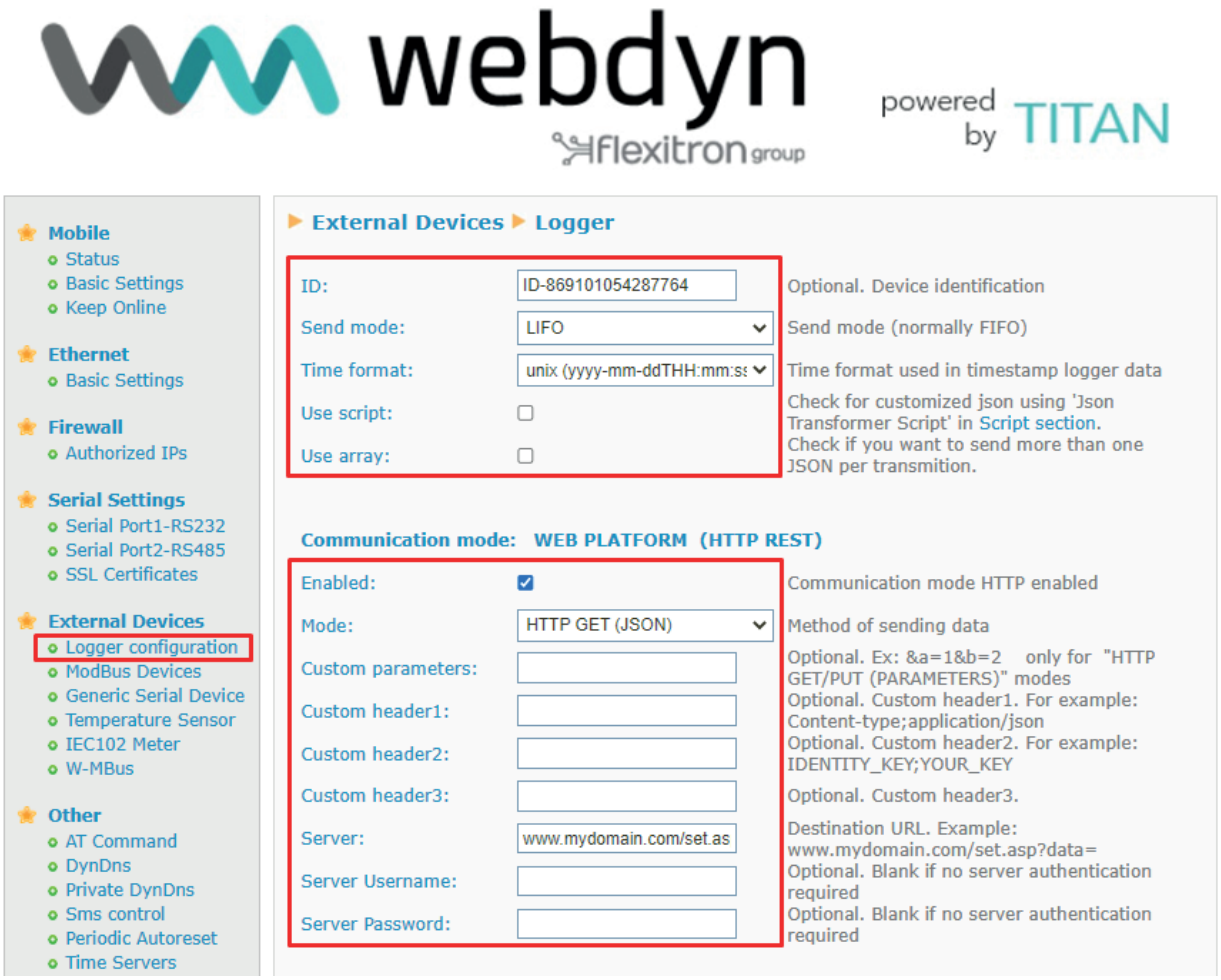
1;3;2;2;9;10;5;4;3;7;1

5. Configuring the Logger (communication with an HTTP server)

The next step is to configure the Logger. This is the data storage and transmission system used by the TITAN-based device itself. In this example we are going to configure the TITAN-based device to send the data to an HTTP server (HTTPS, FTP and MQTT / MQTTS can also be used).

The HTTP server has the URL http://www.mydomain.com/set.asp?data= in which the "data" variable will receive the data in JSON format for each reading taken (timestamp, device ID, etc.).

As can be seen in the following figure, we will access it through the “External Devices > Logger Configuration” menu and configure the section as follows:



6. Other Considerations

- After configuring the TITAN-based device we will need to reset it so that the new configuration takes effect and it starts reading and sending.
- Each time the TITAN-based device sends a measurement to an HTTP server, it does so using a JSON object of the following type:

Example JSON for PLC 2

{“IMEI”:” 869101054287764”,”TYPE”:”MODB”,”TS”:”2022-07-15T10:14:02Z”,”ID”:”2”,”A”:”2”,”ST”:”10”,”N”:”5”,”V”:[10,11,12,0,0],”P”:”ID- 869101054287764”}

Example JSON for PLC 1

{“IMEI”:” 869101054287764”,”TYPE”:”MODB”,”TS”:”2022-07-15T10:09:01Z”,”ID”:”1”,”A”:”1”,”STX”:[1,10,55,69,72,100,120,130,152,160,170],”NX”:[1,3,2,2,9,10,5,4,3,7,1], “PX”:[0,1,4,6,8,17,27,32,36,39,46], “V”:[1,10,11,12,55,56,69,70,72,73,74,75,76,77,78,79,80,100,101,102,103,104,105,106,107,108,109,120,121,122,123,124,130,131,132,133,152,153,154,160,161,162,163,164,165,166,170], “P”:”ID- 869101054287764”}

Where:

IMEI: is the unique identifier for the modem

TYPE: indicates the type of data (MODB = Modbus reading)

TS: is the Timestamp (the time the reading was read)

ID: name or identifier of the Modbus device

A: Modbus device address

ST: the address of the first Modbus register read

STX: an array that indicates the address of the first Modbus registers when reading groups of registers

N: indicates the number of words read

NX: an array that indicates the number of words read when reading groups of registers

PX: an array indicating the position of the initial register of each block within V

V: An array containing the data read

P: the ID field configured in the Logger

It should be noted that there are significant differences between the data sent by PLC1 and that sent by PLC2. PLC1 has groups, meaning that the ST and N fields are replaced by STX and NX in the JSON, these are the arrays in which the initial registers and the number of registers in each block are stored. Similarly, the PX register indicates the initial position of the group within the array V (PX is not really necessary as it can be calculated, but it is included to facilitate decoding of the operation on the server).