

Titan Router

V6 Firmware

Performing actions on Modbus registers
through SMS commands

•Scenario Details

TITAN routers 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 additional features is the ability to run custom scripts within the router itself. In this application note, we will teach you how to program a short script that will allow you to manage the Modbus devices through simple SMS commands. You will use the Titan-based device, Webdyn-Easy-Tunnel, for this application.

Example scenario: a small PLC is available that enables Modbus RTU communications through its RS485 port. This PLC has several sensors connected to it, including a temperature sensor. The PLC also has several relay outputs. The aim is to create a control application through simple SMS commands. To do so, the following must apply:

- 1.- You need a special system administrator's telephone number.
- 2.- You must have up to 9 authorised telephone numbers to be able to send SMS commands.
- 3.- The administrator's phone number must be able to add or remove these 9 authorised phone numbers. To add an authorised telephone number, the administrator's telephone will need to send an SMS command with the text "AUTH,x,yyyyyyyyyy", where "x" is a number from 2 to 10 and "yyyyyyyyyy" is the telephone number to be added to the list of authorised numbers. An example command would be: "AUTH,2,+34666123456".
- 4.- The administrator's phone number must be able to reboot the Titan-based device (this is necessary, for example, so that the Titan-based device uses the new settings once SMS-authorised phone numbers have been added or changed). The SMS command to reboot the Titan-based device shall be called "REBOOT".
- 5.- All telephone numbers must be able to send the SMS command "TEMP". The Titan-based device, upon receiving the "TEMP" command, must read, from the PLC (which will have the Modbus address @1 and whose RS485 port is set to 9600,8,N,1), the Modbus register @30005, which will contain the temperature value. This register value will be divided by 10 to obtain the temperature in degrees to 1 decimal place. The temperature value will be sent to the phone number that sent the "TEMP" message via SMS.
- 6.- All telephone numbers must be able to send the SMS command "GET,xxxxx". This will allow you to read any Modbus register of the PLC, where xxxxx is the address of the Modbus register to be read. Once the Modbus register has been read, the Titan-based device will respond by sending an SMS with this reading to the phone number that sent the "GET,xxxxx" command. For example, the command "GET,30010" could be used to read register 30010.
- 7.- All telephone numbers must be able to send the SMS command "SET,xxxxx,yyyyy". This will allow you to write to any Modbus register of the PLC, where "xxxxx" is the register address and "yyyyy" is the value to be written. The Titan-based device will respond by sending an SMS with the register writing result to the phone number that sent the SMS command. It will also send an additional SMS message to the administrator's phone number with the executed command, as well as the phone number that performed the change action.



1. WAN mobile configuration

In this application, IP communications will not be necessary, as everything will be done via SMS commands. Therefore, the configuration of this section should be the default configuration of Titan-based devices in Disabled mode (only SMS and CSD). Configure the "Mobile > Basic Settings" menu as follows, as well as the APN, username and password if necessary, as required by 4G networks.

The screenshot shows the 'Mobile > Basic Settings' configuration page. The left sidebar lists various settings categories. The main content area is titled 'Mobile > Basic Settings'. A red box highlights the 'Mobile WAN' section, which includes the following settings:

- Mobile WAN: Disabled (only SMS and CSC) (dropdown menu)
- Sim Mode: SIM1 (dropdown menu)
- SIM1 APN: movistar.es (text input)
- SIM1 Username: MOVISTAR (text input)
- SIM1 Password: ***** (masked text input)
- SIM1 Pin: (empty text input)
- SIM1 Auth: None (dropdown menu)

Other visible settings include SIM2 APN, SIM2 Username, SIM2 Password, SIM2 Pin, SIM2 Auth, and Network selection (Auto 4G/3G/2G).

2. Configuring RS485 Serial communications

As indicated early in this application note, the PLC has RS485 serial communications at 9600,8,N,1, so you must use the same configuration for the RS485 serial port of the Titan-based device from the "Serial Settings > Serial Port2-RS485" menu.

The screenshot shows the 'Serial Gateway > Com2 Settings' configuration page. The left sidebar lists various settings categories. The main content area is titled 'Serial Gateway > Com2 Settings'. A red box highlights the 'Baudrate' section, which includes the following settings:

- Baudrate: 9600 (dropdown menu)
- Data bits: 8 (dropdown menu)
- Parity: none (dropdown menu)
- Stop bits: 1 (dropdown menu)
- Timeout ms: 50 (text input)

Below this, there are three checkboxes for AT commands and a 'Function' dropdown menu. The 'Function' dropdown is highlighted with a red box and shows 'Function: Nothing or used by External Device or Script' selected.

3. Configuring Modbus communications

PLC communications will be carried out via Modbus, more specifically, through the RS485 port of the Titan-based device, so the Modbus service must be linked to the RS485 port of the Titan-based device. You need to do this from the “External Devices > Modbus Devices” menu:

External Devices > ModBus RTU / TCP

Enabled: ☒ Enable Modbus Devices

Serial Port: Select the connected serial port if needed

Logger: ☐ Check if logger must be used
Please, configure logger before using this option

Dev. name / ID	Addr.	Command	Start @	Num word/bit	Reg Type	Period
----------------	-------	---------	---------	--------------	----------	--------

Device name / ID: Insert the device name or ID

Address: Modbus RTU address or IP:port address

4. Configuring SMS messages and the administrator's phone number

You must configure the administrator's telephone number. This phone number will later be used to add authorised phone numbers to the list. It is also extremely important to set the "AT header" (we can think of it as a password) and keep it secret only for the administrator. This way, only the administrator will be able to send special AT commands to the Titan-based device. In this example, AT commands can only be executed (with a password) to numbers that are authorised (they should only be those indicated in the configuration). In other words, the settings in the "Other > SMS control" section will appear as shown in the screenshot below, where, for example, "+34666123456" is the administrator's phone number and "mypass" is the password required to execute special AT commands on the Titan-based device.

Other > SMS control

SMS function

AT : ☒ enabled Send AT Commands by SMS allowed (you can reboot the device, get IP Wan, get GSM RSSI, change configuration, ...)

AT header: Header of at commands

Authorized phone numbers: ☐ all phones All Phones are allowed

Authorized number 1

Authorized number 2

Authorized number 3

Authorized number 4

Authorized number 5

Authorized number 6

Authorized number 7

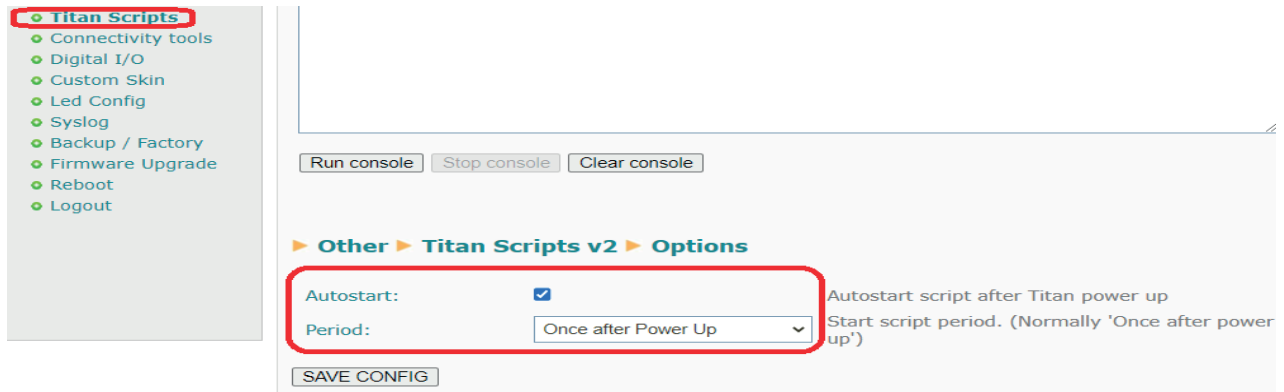
Authorized number 8

Authorized number 9

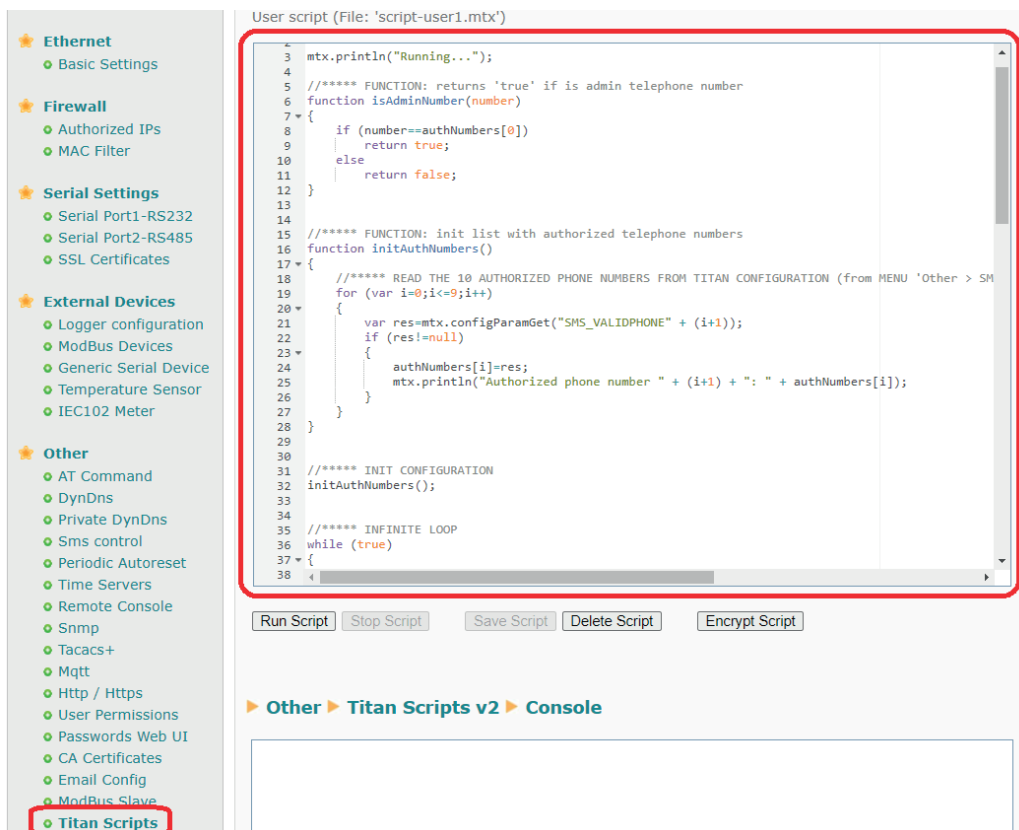
Authorized number 10

5. Script programming

Now, the only thing left for you to do is to program the script that will allow you to perform the described application. This programming must be done from the “Other > Titan Scripts” menu. Once the script is programmed, the "Once after Power Up" option must be selected so that the script runs every time the Titan-based device is started (e.g. after a reboot or a power restart).



The script will be programmed in the section shown below:



The example script code could be as follows (to be used and modified at your own risk):

```
var authNumbers=[];
mtx.println("Running...");

//***** FUNCTION: init list with authorized telephone numbers
function initAuthNumbers()
{
//***** READ THE 10 AUTHORIZED PHONE NUMBERS FROM TITAN CONFIGURATION (from MENU
'Other > SMS control')
    for (var i=0;i<=9;i++)
    {
        var res=mtx.configParamGet("SMS_VALIDPHONE" + (i+1));
        if (res!=null)
        {
            authNumbers[i]=res;
            mtx.println("Authorized phone number " + (i+1) + ": " + authNumbers[i]);
        }
    }
}

//***** FUNCTION: returns 'true' if is admin telephone number
function isAdminNumber(number)
{
    if (number==authNumbers[0])
        return true;
    else
        return false;
}
```

```

//***** INIT CONFIGURATION

initAuthNumbers();

//***** INFINITE LOOP

while (true)
{
    var smsMessage=mtx.smsRead()

    //***** IF THERE IS A NEW SMS ...
    if (smsMessage!=null)
    {
        mtx.println("Received SMS. Number: " + smsMessage[0] + ". Message: " + smsMessage[1]);

        var messageToken=smsMessage[1].split(",");

        //***** REBOOT COMMAND
        if (messageToken[0].toUpperCase()=="REBOOT")
        {
            //***** ONLY ADMIN CAN EXECUTE THIS COMMAND
            if (isAdminNumber(smsMessage[0]))
                mtx.atSend("AT^MTXTUNNEL=REBOOT", 2);
        }

        //***** EXAMPLE, "AUTH,2,+34666223344" ADD/CHANGE THE AUTH NUMBER 2 TO
        +34666223344
        else if (messageToken[0].toUpperCase()=="AUTH")
        {
            //***** ONLY ADMIN CAN EXECUTE THIS COMMAND
            if (isAdminNumber(smsMessage[0]))
                var res=mtx.configParamSet("SMS_VALIDPHONE" + messageToken[1], messageToken[2]);
        }
    }
}

```

```

//***** READ TEMPERATURE COMMAND
else if (messageToken[0].toUpperCase()=="TEMP")
{
//***** READING REGISTER 5 (TEMPERATURE)
    var res=mtx.modbusRTUGetWords(1,3,30005,1);
    //***** IF THE MODBUS READING WAS CORRECT ...
    if (res!==null)
    {
        //***** SHOW THE MODBUS VALUE BY CONSOLE
        mtx.println("Temperature: " + res[0]/10);
var smsOUT=mtx.smsSend(smsMessage[0],"TEMPERATURE: " + res[0]/10 );
    }
}

//***** READ MODBUS REGISTER COMMAND. EXAMPLE COMMAND: "GET,30000"
else if (messageToken[0].toUpperCase()=="GET")
{
    var res=mtx.modbusRTUGetWords(1,3,messageToken[1],1);
    //***** IF THE MODBUS READING WAS CORRECT ...
    if (res!==null)
    {
        mtx.println("The value of register is:" + res[0]);
//***** SENDING A SMS WITH THE READ VALUE
        var smsOUT=mtx.smsSend(smsMessage[0],"Value of register " + messageToken[1] + ":"
+ res[0]);
    }
}

//***** SET MODBUS COMMAND. EXAMPLE COMMAND: "SET,30005,12"
else if (messageToken[0].toUpperCase()=="SET")
{

```



```

modbusRTUSetWords(1,6,parseInt(messageToken[1]),[parseInt(messageToken[2])]);

//***** IF THE MODBUS WRITE WAS CORRECT ...
if (res)
{
    mtx.println("The value of register " + parseInt(messageToken[1]) + " changed to value: " +
    parseInt(messageToken[2]));

    //***** SENDING A SMS TO USER WITH THE RESULT
    var smsOUT=mtx.smsSend(smsMessage[0],"Register " + messageToken[1] + " changed to " +
    parseInt(messageToken[2]) + " -> OK");

    //***** SENDING A SMS TO ADMIN WITH THE ACTION AND RESULT
    var smsOUTAdmin=mtx.smsSend(authNumbers[0],"Register " + messageToken[1] + " changed to " +
    parseInt(messageToken[2]) + " by " + smsMessage[0] + " -> OK");
}
}

//***** 1 SECOND PAUSE
mtx.pause(1000);
}

```

Here is an explanation of certain parts of the script:

The `initAuthNumbers()` function reads and stores the list of authorised telephone numbers in the `authNumbers[]` array. These authorised phone numbers are those found in the "Other >SMS control" section of the Titan-based device. This is done by reading the configuration parameters `SMS_VALIDPHONE1` ... `SMS_VALIDPHONE10`. Consider the phone number of the `SMS_VALIDPHONE1` parameter to be the administrator's phone number.

```

1  var authNumbers=[];
2
3  mtx.println("Running...");
4
5  /***** FUNCTION: init list with authorized telephone numbers
6  function initAuthNumbers()
7  {
8      /***** READ THE 10 AUTHORIZED PHONE NUMBERS FROM TITAN CONFIGURATION (from MENU 'Other > SM
9      for (var i=0;i<=9;i++)
10     {
11         var res=mtx.configParamGet("SMS_VALIDPHONE" + (i+1));
12         if (res!=null)
13         {
14             authNumbers[i]=res;
15             mtx.println("Authorized phone number " + (i+1) + ": " + authNumbers[i]);
16         }
17     }
18 }
19

```

Essentially, the isAdminNumber function returns "true" if the phone number set as a parameter is that of the administrator. This function will be used to enable the execution of SMS commands that only the administrator should be able to execute.

```

20 /***** FUNCTION: returns 'true' if is admin telephone number
21 function isAdminNumber(number)
22 {
23     if (number==authNumbers[0])
24         return true;
25     else
26         return false;
27 }
28
29
30 /***** INIT CONFIGURATION
31 initAuthNumbers();
32

```

This section shows the main loop of the script. In the screenshot below, a check is being performed to see if a new SMS message has been received. If a new SMS has been received, the telephone number that sent the SMS will appear in smsMessage[0], while the text of the SMS message will be displayed in smsMessage[1]. Please note that the SMS message (smsMessage[1]) is split into tokens using the split() method. For example, if the SMS message has been received: SET,30005,12 will appear in the messageToken array after executing the command var messageToken=smsMessage[1].split(","), while the value "SET" (i.e. the command) will appear in messageToken[0], the value 30005 will be displayed in messageToken[1] and the value 12 can be seen in messageToken[2].

In the fragment shown below, you will be able to see if the "REBOOT" command has been received and determine whether the number that sent the SMS is the administrator's number. The AT^MTXTUNNEL=REBOOT command will be executed internally and reboot the Titan-based device.

```

33
34 /***** INFINITE LOOP
35 while (true)
36 {
37     var smsMessage=mtx.smsRead()
38
39     /***** IF THERE IS A NEW SMS ...
40     if (smsMessage!=null)
41     {
42         mtx.println("Received SMS. Number: " + smsMessage[0] + ". Message : " + smsMessage[1]);
43
44         var messageToken=smsMessage[1].split(",");
45
46         /***** REBOOT COMMAND
47         if (messageToken[0].toUpperCase()=="REBOOT")
48         {
49             /***** ONLY ADMIN CAN EXECUTE THIS COMMAND
50             if (isAdminNumber(smsMessage[0]))
51                 mtx.atSend("AT^MTXTUNNEL=REBOOT", 2);
52         }
53     }
54 }
55

```

In the following fragment, you will be able to see if the SMS command "AUTH" has been received and whether this SMS message has been sent from the administrator's phone number. If so, a new authorised phone number will be added/modified. For example, if the SMS command has been received: AUTH,2,+34666223344 will change the configuration parameter of the Titan-based device SMS_VALIDPHONE2 to +34666223344. If, after sending the SMS, you consult the "Other > SMS Control" menu, you will notice that the phone number has been added. Remember to send a "REBOOT" SMS command to ensure the Titan-based device incorporates the new configuration.

```

//***** EXAMPLE, "AUTH,2,+34666223344" ADD/CHANGE THE AUTH NUMBER 2 TO +34666223
else if (messageToken[0].toUpperCase()=="AUTH")
{
    //***** ONLY ADMIN CAN EXECUTE THIS COMMAND
    if (isAdminNumber(smsMessage[0]))
    {
        var res=mtx.configParamSet("SMS_VALIDPHONE" + messageToken[1], messageTok
    }
}

```

The following code fragment will allow you to execute the "TEMP" command to read the temperature:

Once the TEMP command has been received, the command to read the Modbus register 30005 from the PLC with address @1, and by using Modbus command 3, will be executed. If the Modbus reading is successful, a message will be returned to the phone number that sent the message (which is stored in the variable smsMessage[0]). The value of the register read (stored in res[0]) will be divided by 10 to obtain the temperature value in decimal format.

```

//***** READ TEMPERATURE COMMAND
else if (messageToken[0].toUpperCase()=="TEMP")
{
    //***** READING REGISTER 5 (TEMPERATURE)
    var res=mtx.modbusRTUGetWords(1,3,30005,1);
    //***** IF THE MODBUS READING WAS CORRECT ...
    if (res!=null)
    {
        //***** SHOW THE MODBUS VALUE BY CONSOLE
        mtx.println("Temperature: " + res[0]/10);
        var smsOUT=mtx.smsSend(smsMessage[0],"TEMPERATURE: " + res[0]/10 );
    }
}

```

The next command is "GET". For example, if the command sms GET,30000 has been received, as we saw before when explaining the .split() method, the value 30000 will appear in messageToken[1]. This register will then be read and sent by SMS to the telephone number that sent the command.

```

//***** READ MODBUS REGISTER COMMAND. EXAMPLE COMMAND: "GET,30000"
else if (messageToken[0].toUpperCase()=="GET")
{
    var res=mtx.modbusRTUGetWords(1,3,messageToken[1],1);
    //***** IF THE MODBUS READING WAS CORRECT ...
    if (res!=null)
    {
        mtx.println("The value of register is:" + res[0]);
        //***** SENDING A SMS WITH THE READ VALUE
        var smsOUT=mtx.smsSend(smsMessage[0],"Value of register " + messageToken[1] + " is " + res[0]);
    }
}

```

Finally, the script handles the "SET" command, which allows the value of a PLC modbus register to be changed, in this case using the Modbus 6 command.

If the Modbus write action is successful, then the result of the action is sent to the phone number that sent the SMS command, as well as to the administrator's phone number.

```

88 //***** SET MODBUS COMMAND. EXAMPLE COMMAND: "SET,30005,12"
89 else if (messageToken[0].toUpperCase()=="SET")
90 {
91     var res=mtx.modbusRTUSetWords(1,6,parseInt(messageToken[1]),[parseInt(messageToken[2]),parseInt(messageToken[3]),parseInt(messageToken[4]),parseInt(messageToken[5]),parseInt(messageToken[6]),parseInt(messageToken[7])]);
92     //***** IF THE MODBUS WRITE WAS CORRECT ...
93     if (res)
94     {
95         mtx.println("The value of register " + parseInt(messageToken[1]) + " changed to " + parseInt(messageToken[2]) + " " + parseInt(messageToken[3]) + " " + parseInt(messageToken[4]) + " " + parseInt(messageToken[5]) + " " + parseInt(messageToken[6]) + " " + parseInt(messageToken[7]));
96         //***** SENDING A SMS TO USER WITH THE RESULT
97         var smsOUT=mtx.smsSend(smsMessage[0],"Register " + messageToken[1] + " changed to " + parseInt(messageToken[2]) + " " + parseInt(messageToken[3]) + " " + parseInt(messageToken[4]) + " " + parseInt(messageToken[5]) + " " + parseInt(messageToken[6]) + " " + parseInt(messageToken[7]));
98         //***** SENDING A SMS TO ADMIN WITH THE ACTION AND RESULT
99         var smsOUTAdmin=mtx.smsSend(authNumbers[0],"Register " + messageToken[1] + " changed to " + parseInt(messageToken[2]) + " " + parseInt(messageToken[3]) + " " + parseInt(messageToken[4]) + " " + parseInt(messageToken[5]) + " " + parseInt(messageToken[6]) + " " + parseInt(messageToken[7]));
100     }
101 }
102

```

Any questions?

Please direct your enquiries to iotsupport@mtxm2m.com