

TITAN ROUTER

Nota de aplicación 79

Configuración de un Router Titan como gateway
DLMS/COSEM - MODBUS

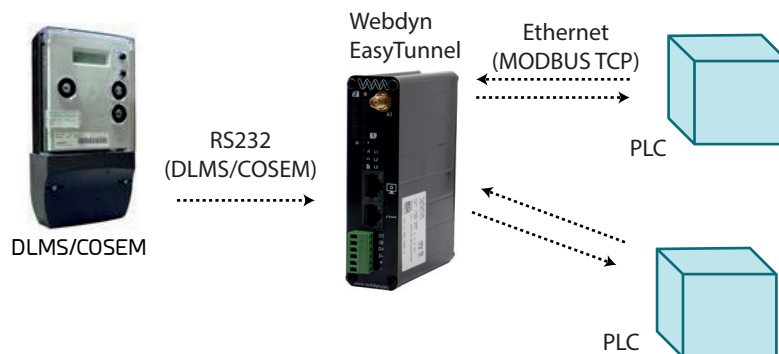
Detalles del escenario

Los router Titan disponen de todas las funcionalidades típicas de un router 4G/3G/2G pero además cuentan con una serie de prestaciones adicionales que lo convierten en uno de los routers con más prestaciones del mercado. Una de las prestaciones adicionales es la capacidad de leer de crear un gateway DLMS/COSEM a MODBUS, lo que permite a dispositivos externos que no tengan implementado el protocolo DLMS/COSEM (como podría ser un PLC), acceder vía modbus a los datos dispositivos DLMS/COSEM, como puede ser un contador eléctrico.

1. Descripción del escenario de ejemplo.

- Se dispone de un Contador Eléctrico ITRON SL7000 con protocolo DLMS/COSEM, con puerto serie RS232 y configuración 9600,8,N,1.

- Se pretende configurar un router Titan (en el caso se utilizará un modelo Webdyn-EasyTunnel al que se hará referencia como "router Titan") para crear una pasarela DLMS/COSEM a MODBUS que permita leer ciertos valores instantáneos del contador (Energía Activa Positiva, Energía Activa Negativa, Energía Reactiva Inductiva, Energía Reactiva Capacitiva, Potencia Activa Positiva, Potencia Activa Negativa, Potencia Reactiva Activa, Potencia Reactiva Negativa, Voltaje Fase I, Voltaje Fase II, Voltaje Fase III) desde dos PLCs conectados. Un PLC conectado al router Titan por el puerto Ethernet (vía MODBUS TCP) y otro conectado por el puerto RS485 (vía Modbus RTU). Es decir, que el router Titan debe comportarse como un Slave Modbus tanto TCP como RTU.



2. Configuración del Puerto Serie RS232.

Este ejemplo utilizará el puerto RS232 del router Titan como puerto serie que estará conectado al Contador Eléctrico DLMS/COSEM. Para configurarlo hay que acudir al menú “Serial Settings → Serial Port1-232” y configurar los valores apropiados. Estos valores apropiados son aquellos que coincidan con la configuración del puerto serie del Contador Eléctrico que en el caso de este ejemplo será 9600,8,N,1.

También debe configurarse una pasarela TCP Server. Utilizaremos como puerto LOCAL el puerto TCP 20010. Este puerto TCP será utilizado por el SCRIPT de conversión DLMS/COSEM a Modbus para la comunicación local con el contador (utilizará internamente la pasarela TCP-RS232)

The screenshot displays the configuration page for the Titan router's serial gateway. The left sidebar shows the navigation menu with 'Serial Settings' and 'Serial Port1-RS232' highlighted. The main content area is titled 'Serial Gateway > Com1 Settings'. It features several configuration fields: Baudrate (9600), Data bits (8), Parity (none), Stop bits (1), Flow Control (none), and Timeout ms (50). Below these fields are three unchecked checkboxes: 'Allow local embedded AT commands', 'Allow remote embedded AT commands', and 'Allow incoming GSM call (CSD Data Call)'. The 'Function' section has 'Serial - IP Gateway (TCP Server)' selected. This section includes fields for 'TCP Local Port' (20010), 'Timeout' (300), and 'TCP Local Priority Port' (20011). The 'Other' section has 'Temporal client RS232' unchecked.

3. Configuración del Puerto Serie RS485

El segundo PLC, con comunicación Modbus RTU, estará conectado al router Titan mediante el bus RS485. Por ello también debemos configurar el puerto RS485 del router Titan. Para configurarlo hay que acudir al menú “Serial Settings → Serial Port2-485” y configurar los valores apropiados. Estos valores apropiados son aquellos que coincidan con la configuración del puerto serie RS485 del PLC, que para este ejemplo consideraremos también 9600,8,N,1.

Serial Gateway ▶ Com2 Settings

Baudrate: 9600 Baudrate of serial port

Data bits: 8 Number of data bit

Parity: none Parity

Stop bits: 1 Number of stop bits

Timeout ms: 50 msec without serial data before sending (default: 50)

Allow local embedded AT commands Ex.: <MTXTUNNEL>AT</MTXTUNNEL>

Allow remote embedded AT commands Ex.: <MTXTUNNELR>AT</MTXTUNNELR>

Allow incoming GSM call (CSD Data Call) Only TCP Server and TCP Client functions or Nothing. 2G (CSD) network required.

Function: Nothing or used by External Device or Script

4. Configuración como Modbus Slave

Tal y como se indicó anteriormente, el router Titan debe configurarse como un esclavo Modbus que permita comunicaciones tanto TCP como RTU. Para configurar ese comportamiento debe acudir al menú “Other → Modbus Slave” y activar ambas opciones. Al servicio Modbus RTU le asociaremos el puerto RS485 (Serial Port 2), y como dirección modbus RTU del router Titan, la 1.

Other ▶ ModBus TCP Slave

Enabled: Enable Titan router as Modbus TCP Slave

ModBus TCP Port: 502 Router waits for connections at this TCP port

Other ▶ ModBus RTU Slave

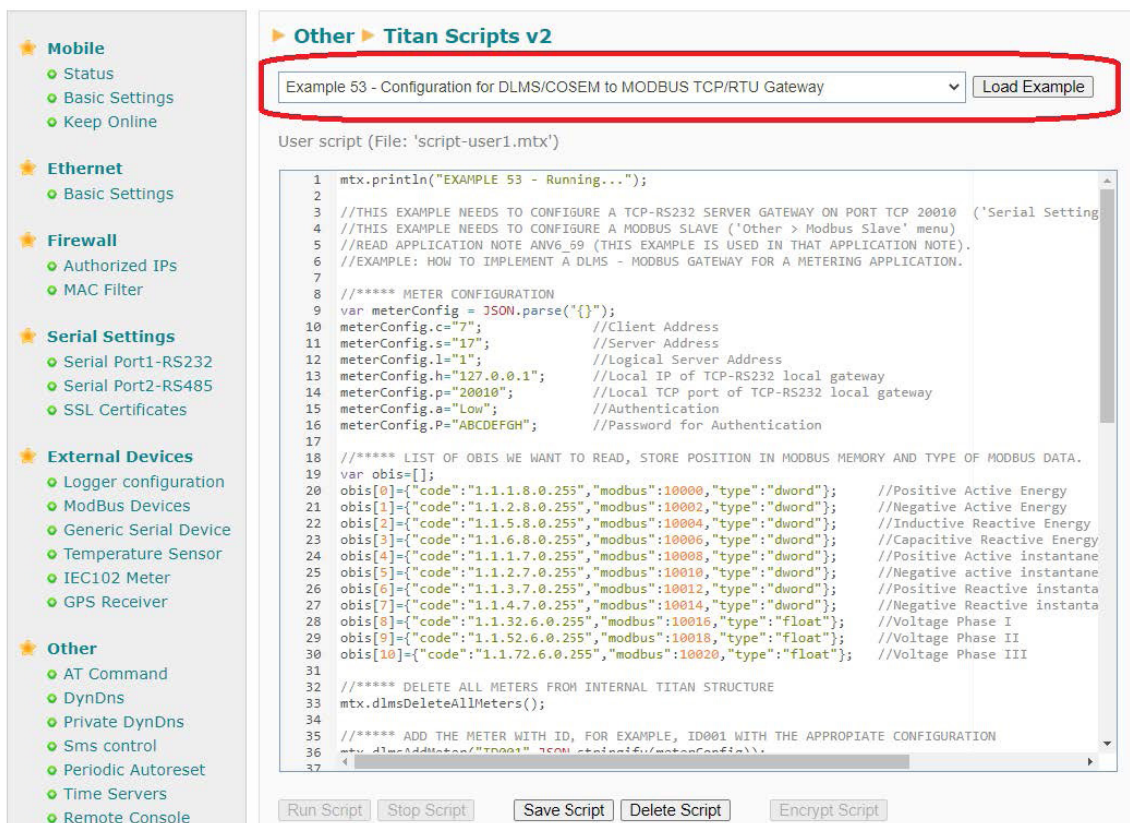
Enabled: Enable Titan router as Modbus RTU Slave

ModBus RTU address: 1 Modbus RTU for Titan Router (1 ... 254)

ModBus COM Port: Serial Port 2 In the COM port configuration, select function "none" or external device.

5. Configuración del SCRIPT

El último paso es programar el script que realice la lectura de los registros del contador DLMS/COSEM y almacene los valores en su memoria interna, en la tabla de registros modbus para que puedan ser leídos externamente. Para programar este script debe acudir a la sección “Other → Titan Scripts”. En este caso no será necesario escribir manualmente todo el código del ejemplo, pues éste forma parte del ejemplo 53 (titulado “Configuration for DLMS/COSEM to MODBUS TCP/RTU Gateway”). Es decir, basta con seleccionar el ejemplo del desplegable y pulsar el botón “Load Example” para cargarlo en memoria. Después bastará con ajustar las configuraciones necesarias para comunicarse con el contador.



The screenshot shows the Titan Scripts v2 interface. On the left is a sidebar with categories: Mobile (Status, Basic Settings, Keep Online), Ethernet (Basic Settings), Firewall (Authorized IPs, MAC Filter), Serial Settings (Serial Port1-RS232, Serial Port2-RS485, SSL Certificates), External Devices (Logger configuration, ModBus Devices, Generic Serial Device, Temperature Sensor, IEC102 Meter, GPS Receiver), and Other (AT Command, DynDns, Private DynDns, Sms control, Periodic Autoreset, Time Servers, Remote Console). The main area is titled 'Other ▶ Titan Scripts v2' and contains a dropdown menu with 'Example 53 - Configuration for DLMS/COSEM to MODBUS TCP/RTU Gateway' selected and a 'Load Example' button. Below this is a text area for the user script (File: 'script-user1.mtx') with the following code:

```
1 mtx.println("EXAMPLE 53 - Running...");
2
3 //THIS EXAMPLE NEEDS TO CONFIGURE A TCP-RS232 SERVER GATEWAY ON PORT TCP 20010 ('Serial Setting
4 //THIS EXAMPLE NEEDS TO CONFIGURE A MODBUS SLAVE ('Other > Modbus Slave' menu)
5 //READ APPLICATION NOTE ANV6_59 (THIS EXAMPLE IS USED IN THAT APPLICATION NOTE).
6 //EXAMPLE: HOW TO IMPLEMENT A DLMS - MODBUS GATEWAY FOR A METERING APPLICATION.
7
8 //***** METER CONFIGURATION
9 var meterConfig = JSON.parse("{}");
10 meterConfig.c="7"; //Client Address
11 meterConfig.s="17"; //Server Address
12 meterConfig.l="1"; //Logical Server Address
13 meterConfig.h="127.0.0.1"; //Local IP of TCP-RS232 local gateway
14 meterConfig.p="20010"; //Local TCP port of TCP-RS232 local gateway
15 meterConfig.a="Low"; //Authentication
16 meterConfig.P="ABCDEFGH"; //Password for Authentication
17
18 //***** LIST OF OBIS WE WANT TO READ, STORE POSITION IN MODBUS MEMORY AND TYPE OF MODBUS DATA.
19 var obis=[];
20 obis[0]={"code":"1.1.1.8.0.255","modbus":10000,"type":"dword"}; //Positive Active Energy
21 obis[1]={"code":"1.1.2.8.0.255","modbus":10002,"type":"dword"}; //Negative Active Energy
22 obis[2]={"code":"1.1.5.8.0.255","modbus":10004,"type":"dword"}; //Inductive Reactive Energy
23 obis[3]={"code":"1.1.6.8.0.255","modbus":10006,"type":"dword"}; //Capacitive Reactive Energy
24 obis[4]={"code":"1.1.1.7.0.255","modbus":10008,"type":"dword"}; //Positive Active Instantane
25 obis[5]={"code":"1.1.2.7.0.255","modbus":10010,"type":"dword"}; //Negative active instantane
26 obis[6]={"code":"1.1.3.7.0.255","modbus":10012,"type":"dword"}; //Positive Reactive Instanta
27 obis[7]={"code":"1.1.4.7.0.255","modbus":10014,"type":"dword"}; //Negative Reactive Instanta
28 obis[8]={"code":"1.1.32.6.0.255","modbus":10016,"type":"float"}; //Voltage Phase I
29 obis[9]={"code":"1.1.52.6.0.255","modbus":10018,"type":"float"}; //Voltage Phase II
30 obis[10]={"code":"1.1.72.6.0.255","modbus":10020,"type":"float"}; //Voltage Phase III
31
32 //***** DELETE ALL METERS FROM INTERNAL TITAN STRUCTURE
33 mtx.dlmsDeleteAllMeters();
34
35 //***** ADD THE METER WITH ID, FOR EXAMPLE, ID001 WITH THE APPROPRIATE CONFIGURATION
36 mtx.dlmsAddMeter("ID001",JSON.stringify(meterConfig));
37
```

A continuación se describirá el código del ejemplo. El siguiente fragmento de código define el JSON de configuración del contador. Esta configuración es necesaria pasarla como parámetro de la función `mtx.dlmsAddMeter()`, como se verá más adelante.

```
8 //***** METER CONFIGURATION
9 var meterConfig = JSON.parse("{}");
10 meterConfig.c="7"; //Client Address
11 meterConfig.s="17"; //Server Address
12 meterConfig.l="1"; //Logical Server Address
13 meterConfig.h="127.0.0.1"; //Local IP of TCP-RS232 local gateway
14 meterConfig.p="20010"; //Local TCP port of TCP-RS232 local gateway
15 meterConfig.a="Low"; //Authentication
16 meterConfig.P="ABCDEFGH"; //Password for Authentication
17
```

En el siguiente fragmento de código, en los parámetros “code” del array de json “obis”, se indican los códigos OBIS que se pretenden leer del contador DLMS/COSEM. En el parámetro “modbus” se indica la dirección de memoria interna del router Titan donde se almacenará cada valor del registro OBIS leído (puede consultarse esa tabla de memoria de usuario en el menú “Other → Modbus Slave”) y en el parámetro “type” se especifica el tipo de datos que tendrá el registro modbus almacenado en memoria. Por ejemplo, en obis[0] se indica que debe leerse el registro OBIS “1.1.1.8.0.255” (que corresponde a la Energía Activa Positiva), que dicho valor debe almacenarse en la dirección de registro modbus del router Titan “10000” y que el tipo de dato del registro modbus almacenado será de tipo “dword” (double word).

```

17
18 //***** LIST OF OBIS WE WANT TO READ, STORE POSITION IN MODBUS MEMORY AND TYPE OF MODBUS DATA.
19 var obis=[];
20 obis[0]={"code": "1.1.1.8.0.255", "modbus":10000, "type": "dword"}; //Positive Active Energy
21 obis[1]={"code": "1.1.2.8.0.255", "modbus":10002, "type": "dword"}; //Negative Active Energy
22 obis[2]={"code": "1.1.5.8.0.255", "modbus":10004, "type": "dword"}; //Inductive Reactive Energy
23 obis[3]={"code": "1.1.6.8.0.255", "modbus":10006, "type": "dword"}; //Capacitive Reactive Energy
24 obis[4]={"code": "1.1.1.7.0.255", "modbus":10008, "type": "dword"}; //Positive Active instantane
25 obis[5]={"code": "1.1.2.7.0.255", "modbus":10010, "type": "dword"}; //Negative active instantane
26 obis[6]={"code": "1.1.3.7.0.255", "modbus":10012, "type": "dword"}; //Positive Reactive instanta
27 obis[7]={"code": "1.1.4.7.0.255", "modbus":10014, "type": "dword"}; //Negative Reactive instanta
28 obis[8]={"code": "1.1.32.6.0.255", "modbus":10016, "type": "float"}; //Voltage Phase I
29 obis[9]={"code": "1.1.52.6.0.255", "modbus":10018, "type": "float"}; //Voltage Phase II
30 obis[10]={"code": "1.1.72.6.0.255", "modbus":10020, "type": "float"}; //Voltage Phase III
31

```

En el siguiente fragmento de código se borran todos los contadores DLMS/COSEM que hubieran en la estructura inicialmente y se añade el único contador que se va a leer en este ejemplo y que identificaremos con el identificador “ID001” como primer parámetro de la función. Como segundo parámetro pasaremos la configuración de conexión con el contador que comentamos anteriormente y que está almacenada en la variable “meterConfig”.

```

32 //***** DELETE ALL METERS FROM INTERNAL TITAN STRUCTURE
33 mtX.dlmsDeleteAllMeters();
34
35 //***** ADD THE METER WITH ID, FOR EXAMPLE, ID001 WITH THE APPROPRIATE CONFIGURATION
36 mtX.dlmsAddMeter("ID001",JSON.stringify(meterConfig));

```

En el siguiente fragmento de código básicamente se realiza, de forma cíclica y periódica, la conexión con el contador DLMS/COSEM con la función mtX.dlmsConnect(). Si la conexión es correcta se realiza la lectura del valor de cada registro OBIS (concretamente se lee el índice 2, donde se encuentra el valor del registro) mediante la función mtX.dlmsGetAttributeValue(). Una vez leído el valor del registro OBIS, éste se almacenará en la memoria interna del router Titan mediante la función mtX.modbusSaveFloat() o mtX.modbusSaveDWord() en función del tipo de datos del registro. Una vez leídos todos los OBIS del contador DLMS/Cosem se realizará la desconexión mediante la función mtX.dlmsDisconnect().

```

38 var running=true;
39
40 while (running)
41 {
42   try
43   {
44     //***** CONNECT WITH THE DLMS METER
45     var resConnect=mtx.dlmsConnect("ID001");
46     if (resConnect)
47     {
48       //***** FOR EACH OBIS ...
49       for (var i=0;i<obis.length;i++)
50       {
51         //***** GET THE VALUE OF OBIS (ATTRIBUTE NUMBER 2)
52         var value=mtx.dlmsGetAttributeValue(obis[i].code,2);
53         //***** IF READING WAS OK ...
54         if (value[0].length>0)
55         {
56           //***** PRINT THE READ VALUE BY CONSOLE
57           mtx.println("OBIS (" + obis[i].code + "): " + value[0]);
58
59           //***** IF TYPE OF OBIS IS FLOAT, THEN SAVE DATA IN MODBUS MEMORY AS FLOAT
60           if (obis[i].type=="float")
61             mtx.modbusSaveFloat(obis[i].modbus, value[0]);
62           //***** IF TYPE OF OBIS IS DOUBLE WORD, THEN SAVE DATA IN MODBUS MEMORY
63           else if (obis[i].type=="dword")
64             mtx.modbusSaveDWord(obis[i].modbus, value[0]);
65         }
66         //***** IF READING WAS ERROR ...
67         else
68           mtx.println("OBIS (" + obis[i].code + "): ERROR: " + value[1]);
69       }
70     }
71     mtx.dlmsDisconnect();
72   }
}

```

6. Probando el escenario.

Por último queda comprobar el correcto funcionamiento del sistema. Para ello podemos ejecutar manualmente el Script pulsando el botón “Run Script” y observar el resultado en la salida de consola. Si todo funciona correctamente se podrá visualizar el valor de cada uno de los registros OBIS leídos.

The screenshot shows the Titan Scripts v2 interface. On the left is a navigation menu with 'Titan Scripts' highlighted. The main area displays a script with a 'Run Script' button circled in red. Below the script, the console output is shown, also circled in red, displaying the following data:

```

EXAMPLE 54 - Running...
OBIS (1.1.1.8.0.255): 0
OBIS (1.1.2.8.0.255): 0
OBIS (1.1.5.8.0.255): 0
OBIS (1.1.6.8.0.255): 0
OBIS (1.1.1.7.0.255): 0
OBIS (1.1.2.7.0.255): 0
OBIS (1.1.3.7.0.255): 0
OBIS (1.1.4.7.0.255): 0
OBIS (1.1.32.6.0.255): 242
OBIS (1.1.52.6.0.255): 0.0
OBIS (1.1.72.6.0.255): 0.6
JSON DATA:{"PAV":0,"NAC":0,"IRE":0,"CRA":0,"PAIP":0,"NAIP":0,"PEIP":0,"NEIP":0,"VP1":242,"VP2":0.8,"VP3":0.6}

```

Una vez veamos que los registros son leídos, también podemos conectarnos vía Modbus TCP al router Titan y comprobar la lectura de los registros Modbus asociados a cada uno de los OBIS.

Modbus Poll - Mbpoll1

File Edit Connection Setup Functions Display View Window Help

05 06 15 16 17 22 23 TC

Mbpoll1

Tx = 39406: Err = 7558: ID = 1: F = 03: SR = 1000ms

	Alias	10000	Alias	10010	Alias	10020
0		0		0		0.6
1		--		--		--
2		0		0		0
3		--		--		--
4		0		0		0
5		--		--		0
6		0		242		0
7		--		--		0
8		0		0.8		0
9		--		--		0

For Help, press F1. [192.168.1.2]: 502

¿Más dudas?

Escríbenos tus consultas a iotsupport@mtxm2m.com