

YATUN Modbus PLC driver for Control4

[Introduction](#)

[Compatibility](#)

[Requirements](#)

[Limitations](#)

[Test the driver before installing it on site](#)

[Installation](#)

[Basic process of integration](#)

[Modbus PLC protocol driver](#)

[Modbus memory space and plc_map.csv file](#)

[Driver settings](#)

[Modbus TCP PLC: network connection setting](#)

[Modbus RTU PLC: serial port connection setting](#)

[Manual call of Modbus commands via Programming](#)

[Write Single Coil Command](#)

[Write Multiple Coils Command](#)

[Write Single Register Command](#)

[Mask Write Register Command](#)

[Write Multiple Registers Command](#)

[Read Coils Command](#)

[Read Discrete Inputs Command](#)

[Read Input Registers Command](#)

[Read Holding Registers Command](#)

[Slave drivers](#)

[Slave Driver Number Variable](#)

[Adding the driver](#)

[Settings](#)

[Variables](#)

[Troubleshooting and problems](#)

[I can not load the plc_map.csv file to controller](#)

[There are no connections in Modbus PLC driver](#)

[Error reporting](#)

[Licensing](#)

[Annex 1 - Creating the plc_map.csv driver](#)

[Rules](#)

[Example of creating plc_map.csv driver](#)

[Annex 2 – List of drivers](#)

Introduction

The driver allows you to connect devices using Modbus protocol to Control4 system. Implemented two way communication with connected device makes possible control of the device through standard controls from Control4 user interface. The driver operates as a Modbus master and two versions of the driver are available, one for the Modbus TCP (connection via TCP/IP) and second for the Modbus RTU (connection via serial link) protocol. More Modbus devices can be connected, one driver for each device has to be added to Control4 project.

Modbus is a communication protocol which supports large range of devices from home and industrial automation, especially different PLC units, ventilation and heat recovery devices and many others. Usually these have nontrivial functions for which drivers created using the Driver Wizard are insufficient.

Compatibility

The drivers have been tested on following versions:

Control4

OS:	2.5.2, 2.5.3
-----	--------------

We have successfully tested driver communication with PLC units Amit, GFR and Wago, controller Danfoss ECL Comfort 310, IO modules Moxa ioLogik E1200 Series, ventilation system Atria using Carel PCOS004850 module and RTD-NET device (used to control Daikin Ventilation units [see web](#)).

Requirements

The driver works on HC-800 and on HC-250 with sufficient power reserve. If more drivers are used in one project using the HC-800 may be inevitable. In case you connect Modbus device to RS485 bus do not forget you are going to need RS232-RS485 convertor to be able connect it to a Control4 controller.

Limitations

The driver works in the Modbus master mode only, the slave mode is not supported and the support is not planned. Currently the transmission of text strings is not supported but in a future we plan to add it. Following Modbus protocol functions are supported:

- Read coils (FC 1)
- Read discrete inputs (FC 2)
- Read holding registers (FC 3)
- Read input registers (FC 4)
- Write single coils (FC 5)
- Write single register (FC 6)
- Write multiple coils (FC 15)
- Write multiple registers (FC 16)
- Mask write register (FC 22)

The driver is classified as Modbus device Class 1 with exception of unimplemented function Read exception status (FC 7). Functions FC 15, FC 16 a FC 22 are implemented as an extra.

Test the driver before installing it on site

We strongly recommend you test the features and limitations of this driver package before you attempt to install the drivers for your customer.

For this reason the protocol driver has a “trial” mode which allows you to test the driver for six hours at no cost without enter the licence key.

If you need a license key for your showroom or workbench, please send us an email to c4drivers@yatun.cz.

Installation

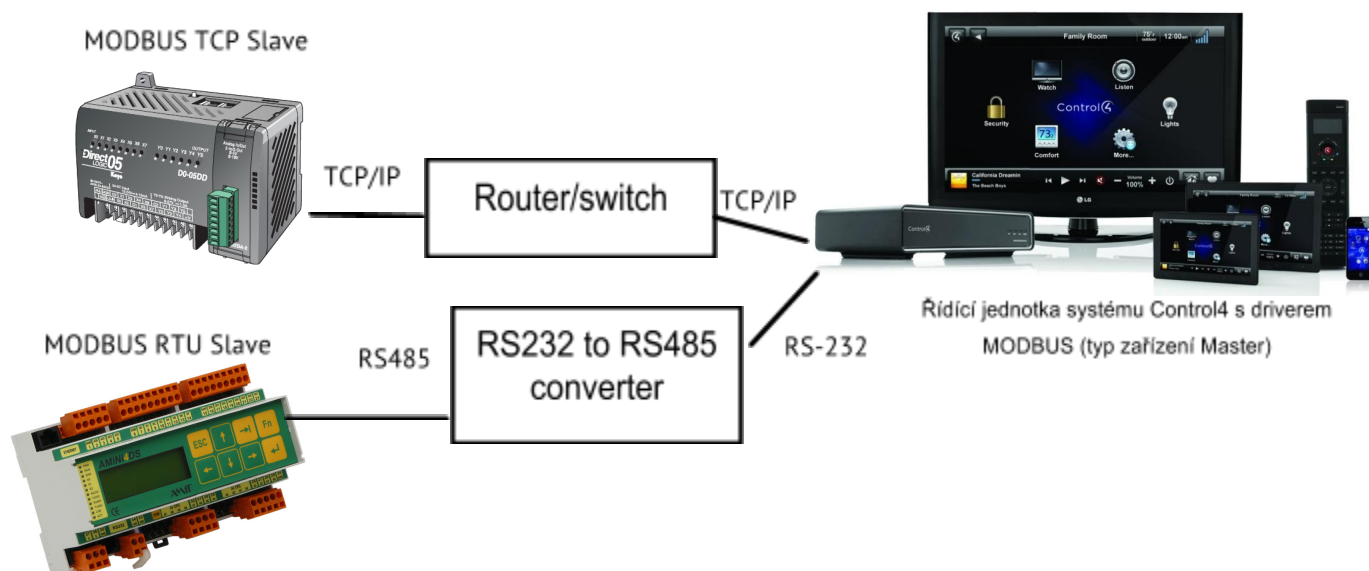
You can download the current version of the driver package at <http://www.yatundev.eu/>.

Unpack the driver files (c4i extension) and move/copy them to your Composer Pro drivers folder – usually *Documents\Control4\Drivers* and restart Composer Pro.

Never copy any third-party drivers to the Composer Pro installation folder (usually *C:\Program Files*).

Basic process of integration

1. Prepare the list of elements in connected Modbus device, its types and modbus addresses. You will have to use the manufacturer's documentation. It is preferable to prepare the list in MS Excel or LibreOffice Calc.
2. Based on the list create the configuration file *plc_map.csv* which describes all elements you want to integrate with Control4. If you have written them in a spreadsheet (in the first step) systematically you can just export it to csv format.
3. Choose Modbus RTU (for RS485) or Modbus TCP (for IP) to be used.
4. Set the required communication speed in Modbus RTU (c4i) driver if needed.
5. Add the chosen Modbus driver to the Control4 project in *Composer Pro – System Design*.
6. License the driver or use it in the trial mode.
7. Set the used serial port in *Connections – Control AV* for Modbus RTU driver or set the IP address in *Connections – Network* for Modbus TCP of connected Modbus device.
8. Upload the configuration file (*plc_map.csv*) to the location shown in driver *Properties* .
9. In *System Design – Actions* click on *Process Config File*. This action will create driver connections for slave drivers.
10. Add a slave driver for each element to Control4 project. Connect each slave driver with Modbus protocol driver in *Composer Pro – Connections*.



Connection diagram - Control4 with PLC units supporting Modbus protocol

Modbus PLC protocol driver

The driver periodically reads the memory space of connected device when properly configured. It transforms these binary data according to the description in the `plc_map.csv` file to the numeric or other values and send them to control and visualisation elements in Control4 system using messages through connections (these are created by processing `plc_map.csv` file). When visualisation element in Control4 changes its state the message is send back to Modbus PLC driver through a connection and Modbus PLC driver overwrites the value in the appropriate memory place in the connected Modbus device. You have to add a slave driver to Control4 project for each controlled or visualized value. Modbus PLC protocol driver is meant for connecting one device which uses Modbus communication protocol. If you want to have more devices connected via Modbus protocol in one project add for each of them its own driver.

Modbus memory space and `plc_map.csv` file

When configuring the driver the most important thing to do is to prepare `plc_map.csv` file. The manufacturer of the device usually let you know of modbus addresses of useful variables and their types in the documentation. If you don't have this information you have to ask for it. It also beneficial to know how modbus protocol addresses works. For detail information you can read the Modbus for Field Technicians document (http://www.modbusbacnet.com/includes/pdf/MODBUS_2010Nov12.pdf) or visit <http://www.simplymodbus.ca>. In abbreviated form it is as follows:

1. Four memory sections exist – *Coils* (1-bit value, read/write, a relay originally), *Discrete Inputs* (1-bit value, read only, digital inputs originally), *Input Registers* (16-bit value, read only, analog input originally), *Holding Registers* (16-bit value, read/write, analog output originally)

- Addresses of each elements are separately numbered in each memory section. Originally the range was 1-9999, but today most of devices uses 1 – 65535. In some user guides are numbers shifted by one down (0 means 1, 1 means 2, etc.). Usually the number of memory section is placed before the number of address (0 – Coils, 1 – Discrete Inputs, 3 – Input Registers, 4 – Holding Registers) but it is not always.
- The original intention of simply addressing the relay, contacts and analog inputs and outputs are complicated by the transfer of numbers with more digits and decimal point (they can occupy 2 or 4 registers). Some PLCs map relays to individual bits in chosen Holding Registers instead of Coils.

Everyone has its own way and therefore you have to find out from documentation which way is used for particular device. You have to specify at least the name of each element (column *NAME*), the memory section (column *LOCATION*), the address (column *ADDRESS*) and the type (column *TYPE*) for the `plc_map.csv` file. The name can be chosen freely but the best way is to make the name of the element easily recognizable. The memory section and the address correspond to the modbus memory section and address. The address should be in the format as if the address range starts with 1 (if the producer states that range starts with 0, add plus 1 to address). The type defines the Modbus driver behavior towards element and how it interprets the value of the element. The memory section does not have to be stated in `plc_map.csv` file if it is the same as an implicit memory section for the type. There are available following types in the current version :

- LIGHT: light control (1 bit size, coils implicit memory section).
- RELAY: relay (1 bit, coils).
- CONTACT: binary input (1 bit, discrete inputs).
- DIMMER: dimming light, assumed to be an integer in 0 – 100 range (1 register, holding registers).
- TEMP: temperature, integer (1 register, INT without sign, holding registers).
- TEMP_10: temperature with an accuracy of one decimal number multiplied by ten, integer (1 register, holding registers).
- TEMP_100: temperature with an accuracy of two decimal numbers multiplied by 100, integer (1 register, holding registers).
- SETPOINT: temperature, integer (1 register, INT without sign, holding registers).
- SETPOINT_10: temperature with an accuracy of one decimal number multiplied by ten, integer (1 register, holding registers).
- SETPOINT_100: temperature with an accuracy of two decimal numbers multiplied by 100, integer (1 register, holding registers).
- VAR_FLOAT: variable, decimal number, float accuracy (2 registers, holding registers).
- VAR_DOUBLE: variable, decimal number, double accuracy (4 registers, holding registers).
- VAR_UINT16: variable, nonnegative integer unsigned short (1 register, holding registers).
- VAR_INT16: variable, integer short (1 register, holding registers).
- VAR_UINT32: variable, nonnegative integer unsigned int (2 registers, holding registers).
- VAR_INT32: variable, integer int (2 registers, holding registers).
- VAR_UINT64: variable, nonnegative integer unsigned long (4 registers, holding registers).
- VAR_INT64: variable, integer long (4 registers, holding registers).

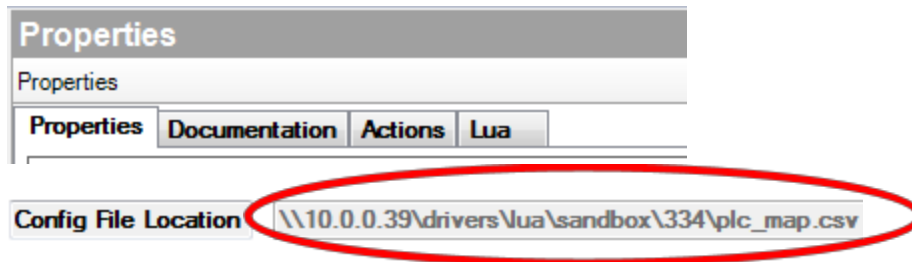
In the case of 1-bit element placed to registers (input or holding) you have to put down the number of the bit representing the element into BIT column.

Example of `plc_map.csv` file:

```
NAME, TYPE, ADDRESS, BIT, LOCATION
Lamp_at_the_gate, LIGHT, 101, 0, 4 // 1-bit value in register 101, bit 0
Light_entrance_hall, LIGHT, 10, , // switch light in coil 10
Dimmer_garage, DIMMER, 102, , // dimmer
Relay_gate, RELAY, 101, 1, 4 // 1-bit value in register on address 102, bit 1
Temperature, TEMP_10, 106, , // temperature sensor
AnalogInput, VAR_FLOAT, 104, , // value from light intensity sensor, as a variable
Humidity_outdoor, VAR_INT16, 114, , // value from humidity sensor, as a variable
```

You can find detailed rules how to create `plc_map.csv` file in Annex 1.

After you manage to create `plc_map.csv` file upload it with a file manager to address stated in driver Properties.



Driver settings

There are many parameters available in *System Design – Properties* which influence driver functions and communication with device.

Address is an identification number of the connected Modbus slave device (UID – unit ID, Slave ID), range is 1 - 255. It is usually ignored in Modbus TCP, you can put in anything. For Modbus RTU it must be set to the right value.

TCP Port (Modbus TCP only) is the number of TCP port where device awaits communication. Usually you do not have to change its default value of 502.

Modbus Read Delay [ms] (0 – 3600000) is a delay between reads of each data block in the address space of connected Modbus device. When `plc_map.csv` file is loaded the driver automatically divides reserved memory space according to *Modbus Read Blocks* setting. Then these blocks are read repeatedly with the delay set by this parameter. If the value is set to 0, the automatic reading of blocks from the device is stopped. We suggest to slightly experiment with this parameter to find the value which is the best for the device. Too high a number means slow update of value in Control4, too low a number can be taxing for Control4 or the device.

Modbus Read Blocks determine which ways are created the blocks for automatic value reading from the device. The *Can include space* value tries to create blocks as large as possible so there would be a minimum amount of blocks and the reading is as fast as possible. However, this option results in errors for some devices. The *Precise* value makes blocks that contain only elements defined in the `plc_map.csv` file. Generally this means more smaller blocks and therefore slower value reading from device.

PLC: Register Endianness defines the order of registers for values which are over many registers. Big Endian means that the most significant register is the one with the lowest address.

PLC: Byte Endianness sets the order of bytes in the register. Big Endian means that the most significant byte is the one in bits 0-7 of a register.

(Yes, there are devices with different endianness of bytes and registers out there.)

Modbus fields allow you to set whether the device supports each of the Modbus functions. These settings serve as a basis for decision which kind of commands to use for reading and writing values into device. Some commands are fully or partially replaceable (e.g. can cause a lower speed), others disable some function to be used.

Field name	Modbus function
Modbus: Read coils (FC 1)	Reading the state of binary outputs (Coils memory section).
Modbus: Read discrete inputs (FC 2)	Reading the state of binary inputs (Discrete Inputs memory section).
Modbus: Read holding registers (FC 3)	Reading register values from Holding registers memory section.
Modbus: Read input registers (FC 4)	Reading register values from Input registers memory section.
Modbus: Write single coil (FC 5)	Setting the required value of one binary output.
Modbus: Write single register (FC 6)	Setting the required value of one register in Holding registers memory section. If the device does not support FC 22 this function can be used for writing one-bit value placed in registers too. This can negatively affect the system behaviour predictability, because the whole register is written in the state which driver currently knows (in limiting cases this state does not have to correspond with the actual).
Modbus: Write multiple coils (FC 15)	Setting the required value of more binary outputs at once.
Modbus: Write multiple registers (FC 16)	Setting the required value of more registers in Holding Registers memory section at once.
Modbus: Mask write register (FC 22)	Setting the required output register value using bit masks AND and OR. Allows to set one specific register bit and it is used by driver for writing one bit types (if they are saved in registers).

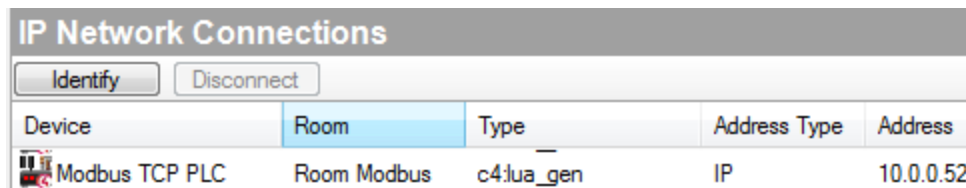
Note: The whole field description shows when you click on it and move the cursor to the end by pressing right arrow key.


Warning: When content of binary outputs and registers are changed at the same time (e.g. if their values are changed from Control4 driver and almost simultaneously from a Modbus device) the last value written in them remains.

If connected device does not support Mask write register (FC 22) command and if device has mapped 1-bit types in registers, the Write single register (FC 6) command is used to write 1-bit values into registers. This command does not have masking and therefore writes the whole register at once. If there is a nearly simultaneous change of value in the register carried out from the driver and from the device (both devices are changing the content of the same register in quick sequence) you may lose some of written information. The driver does not register a change in the bits of the register done just before it sends the write command and therefore the register is overwritten with older data.

Modbus TCP PLC: network connection setting

Choose the driver line in *Composer Pro – Connections* in *Network* tab and click on Identify for an input field to be shown where you can enter the MODBUS server unit IP address.



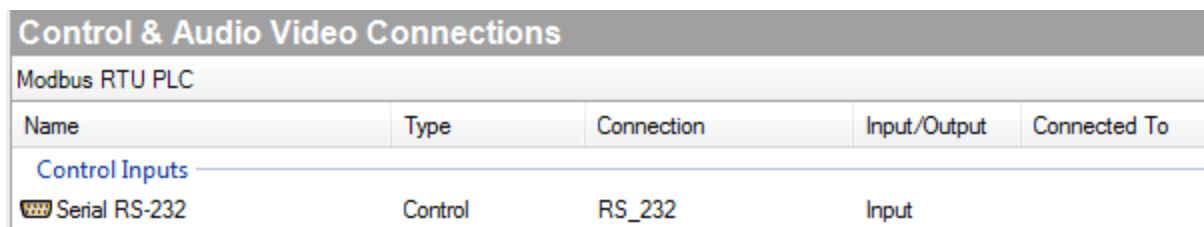
IP Network Connections				
Identify		Disconnect		
Device	Room	Type	Address Type	Address
 Modbus TCP PLC	Room Modbus	c4lua_gen	IP	10.0.0.52


From this moment the driver is active and communicates with Modbus TCP device.

Note: If the standard Modbus TCP protocol port (502) is not being used you have to set the number of the actual TCP port used in *Properties*.

Modbus RTU PLC: serial port connection setting

If Modbus RTU PLC driver is being used, after adding it to the project you have to set the used serial RS232 port where the Modbus device is connected (usually via RS232 – RS485 converter).



Control & Audio Video Connections				
Modbus RTU PLC				
Name	Type	Connection	Input/Output	Connected To
Control Inputs				
 Serial RS-232	Control	RS_232	Input	

The connected device may require different communication parameters of serial port from default parameters used by driver. The only way how to change these parameters is to edit the driver file. Copy the driver file and name it properly (for example automation_232_modbus_plc_57600_8_even_none.c4i) so you would recognize it. Open the new file in a text editor (e.g. Notepad in Windows) and search for serial settings string, correct values to required and save the file. Before adding driver to the project restart *Composer Pro*.

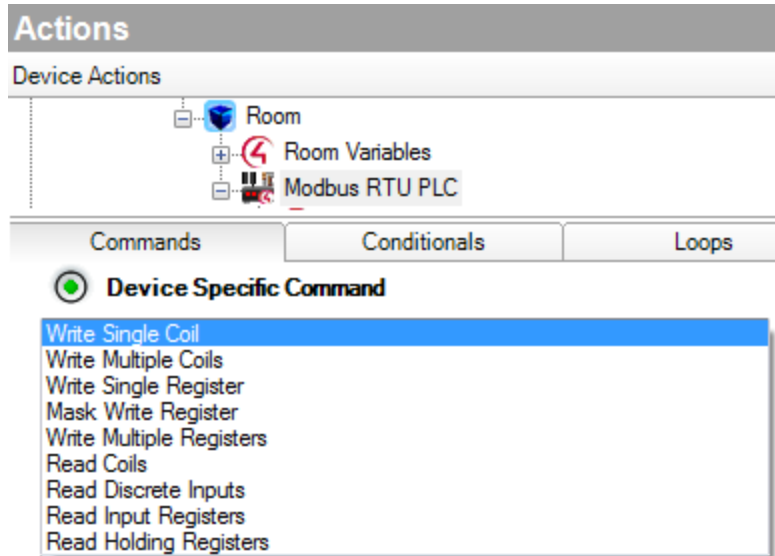
Default parameters values of serial port used by driver are: 38400, 8 databit, even parity, 1 stop bit, no flow control, which corresponds to the following sections in XML driver:

```
<capabilities>
```

```
<serialsettings>38400 8 even 1 none</serialsettings>
</capabilities>
```

Manual call of Modbus commands via Programming

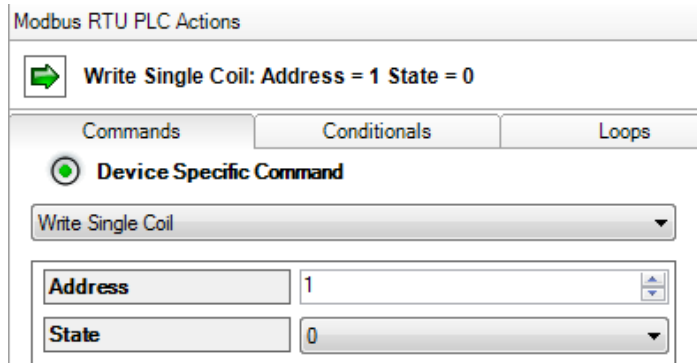
Modbus PLC driver allows to send Modbus commands directly in addition to the automatic exchange of values between Modbus devices and Control4. It is a low-level option and we recommend using it only if you know what you are doing. You can find them in *Device Specific Commands*.



Write Single Coil Command

Function: Writes a value to one digital output.

Parameters: *Address* – an address in memory section *Coils*, *State* – new value (0 or 1).



Write Multiple Coils Command

Function: Writes values of elements to the *Coils* address section.

Parameters: *Address* – an address of the first element, *States* – a list of values to write separated by a comma (the value can be 0 or 1).

Write Single Register Command

Function: Write a value to one register in *Holding Registers* memory section.

Parameters: *Address* – an address of the register, *Value* – a value to write (16-bit value expressed as a number between 0 – 65535).

Mask Write Register Command

Function: Allows to alter a value in *Holding Registers* memory section using AND and OR masks. resulting register value = (original register value AND AndMask) OR (OrMask AND (NOT AndMask)).

Parameters: *Address* – an address of the register, *AndMask* – AND mask, *OrMask* – OR mask. Values of masks are entered as a number between 0 – 65535.

Write Multiple Registers Command

Function: Writes values into consecutive registers in *Holding Registers* memory section at once.


Parameters: *Address* – an address of the first register, *Values* – a list of values separated by commas. Values are 16-bit, stated as a number in the range 0 – 65535.

Read Coils Command


Function: Reads values from *Coils* memory section. Read values updated states of elements created by plc_map.csv. The value of element in the first address is also available in the variable RESULT READ COIL.

Parameters: *Address* – an address of the first element, *Count* – a number of elements to read.

Modbus TCP PLC Actions

 Read Coils: Address = 1 Count = 1

Commands Conditionals Loops

 Device Specific Command

Read Coils

Address	1
Count	1

Read Discrete Inputs Command

Function: Reading values from *Discrete Inputs* memory space. Read values update states of elements created by plc_map.csv.

Parameters: *Address* – an address of the first element, *Count* – a number of elements to read.

Read Input Registers Command

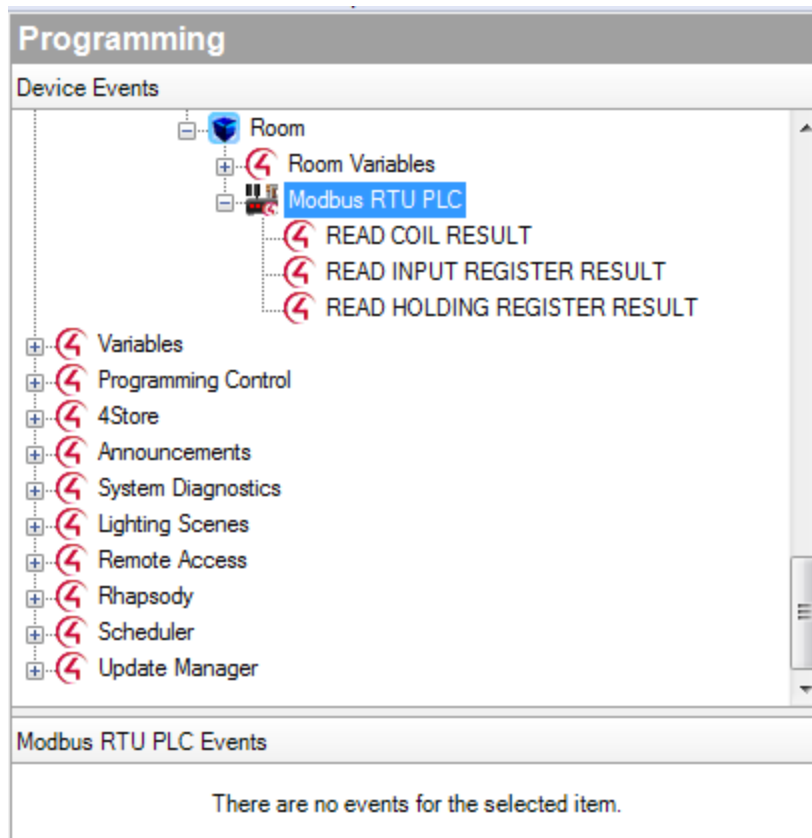
Function: Reads values from *Input Registers* memory space. Read values update states of elements created by plc_map.csv. The value from the first address is also available in READ INPUT REGISTER RESULT value.

Parameters: *Address* – an address of the first element, *Count* – a number of elements to read.

Read Holding Registers Command

Function: Reading of entered values from *Holding Registers* memory space. Readed values update states of elements created by *plc_map.csv*. The value from the first address is also available in READ HOLDING REGISTER RESULT value.

Parameters: *Address* – the address of the first read element, *Count* – the number of elements which will be readed.



Slave drivers

Modbus PLC protocol driver provides communication with devices using Modbus protocol. To reflect an element in Control4 user interface you have to add a slave driver for each element of the device in *Composer Pro – System Design*. You have to connect these slave drivers properly with Modbus PLC in *Composer Pro – Connections*. After uploading of *plc_map.csv* file to controller and calling the *Process Config File* action in *Composer Pro – System Design – Actions* the connections will appear in Modbus PLC driver.

Type in <i>plc_map.csv</i>	Connection class	Subordinate driver
CONTACT	CONTACT_SENSOR	Anything from <i>My Drivers – Sensors</i>
RELAY	RELAY	Anything from <i>My Drivers – Motorization</i>

LIGHT	GEN_SWITCH	Yatun Switching Light (light_generic.c4i)
DIMMER	GEN_DIMMER	Yatun Dimming Light (light_gen_dimmer.c4i)
VAR_*	NUMBER_VARIABLE	Number Variable (variable_number.c4i)
TEMP, TEMP_10,		
TEMP_100	THERMOMETER	Software Thermostat, Temperature Consolidator
SETPOINT, SETPOINT_10,		
SETPOINT_100	THERMOMETER	Software Thermostat (Current Heat/Cool Setpoint)

The example of Modbus driver window in *Composer Pro – Connections*, which serve to connect slave drivers.

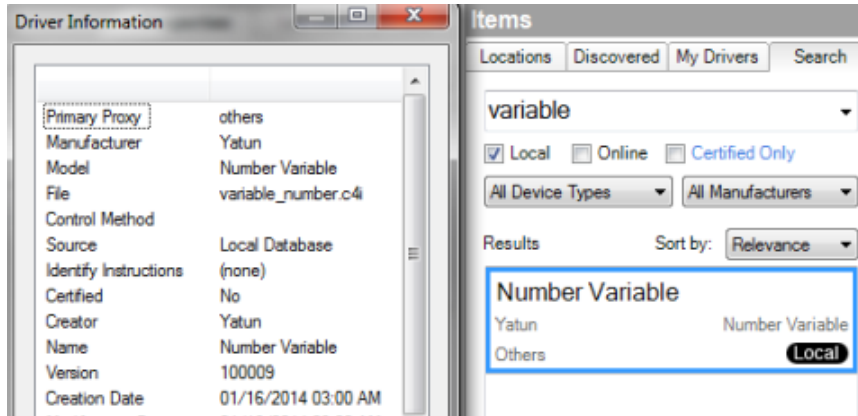
Control & Audio Video Connections				
Modbus TCP PLC				
Name	Type	Connection	Input/Output	Connected To
Control Outputs				
Lamp_at_the_gate	Control	GEN_SWITCH	Output	
Relay_gate	Control	RELAY	Output	
Temperature	Control	THERMOMETER	Output	
AnalogInput	Control	NUMBER_VARIABLE	Output	
Humidity_outdoor	Control	NUMBER_VARIABLE	Output	
NUMBER_VARIABLE Input Devices				
Device	Name	Location	Connections	
Number Variable 1	Number Variable	Room Modbus		
Number Variable 2	Number Variable	Room Modbus		

Slave Driver Number Variable

The driver is intended to work with variables which are transferred from Modbus system while they do not have direct use in visualisation. The driver allows to correct the transferred value using the mathematical formula and the result is available as a numerical (VALUE) and a text (FORMATTED VALUE) value for further use in *Programming*. The value transfer works both ways – when the value of the numerical value VALUE is changed, it is transferred to connected Modbus device.

Adding the driver

Look up the driver in *Composer Pro – System Design* in the Search tab (see the picture) and add it to the list of devices in the project. For every variable which you want to transfer from Modbus device you need to add one Number Variable driver.



After adding it to the project connect the Number Variable driver with Modbus PLC controller in *Composer Pro – Connections*.

Settings

The driver allows you to modify the received value from Modbus. The method of modification is defined in *Composer Pro – Properties*. You can choose the way how to process it in *Transformation*:

- *None* – the value is not changed.
- *Gain* – the value is multiplied by the constant stated in the Gain field. The Gain field appear after choosing Gain in Transformation field and an confirmation by click on the Set button.

The Gain value is entered as a text which is automatically transferred to a number (before the calculation). In this way you can enter a non decimal number in Composer Pro too.

Transformation	Gain	Set
Gain	7.0	
Format	Value: %f	
Value	Value: 210.000000	

- *Formula* – the value received from Modbus device is processed according to the formula which is entered in the Formula field. The value is represented by „x“ variable in the formula. Be careful – the letter x must be entered in lowercase. The Lua syntax is applied for the formula writing.

The Formula field is shown after the choosing Formula in Transformation field and an confirmation by click on the Set button.

Transformation	Formula	
Formula	(x-32)*5/9	Set
Format	Value: %f	
Value	Value: 0.000000	

You can use all operators and functions which are supported by Lua in Director (version 5.1+).

Operators: +, -, *, /, % (modulo), ^ (square)

Functions: math.abs, math.acos, math.asin, math.atan, math.atan2, math.ceil, math.cos, math.cosh, math.deg, math.exp, math.floor, math.fmod, math.frexp, math.huge, math.ldexp, math.log, math.log10, math.max, math.min, math.modf, math.pi, math.pow, math.rad, math.random, math.randomseed, math.sin, math.sinh, math.sqrt, math.tanh, math.tan.

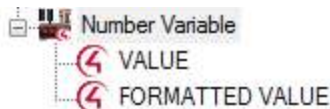
(For function descriptions see <http://lua-users.org/wiki/MathLibraryTutorial>.)

You can set the formatting of the result number to the text variable FORMATTED VALUE in Format. The formatting of output string is the same as the string.format command in Lua language which approximately corresponds with printf in C language:

<http://www.cplusplus.com/reference/cstdio/printf/>. The description of formatting exception in Lua: <http://pgl.yoyo.org/luai/i/string.format>.

Variables

The driver contains two variables.



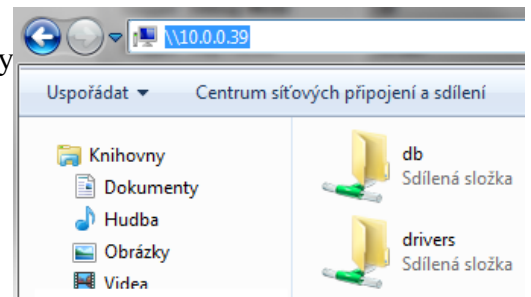
The numerical variable VALUE contains value received from Modbus device transformed according to *Properties*. When the VALUE variable is changed in Control4 the new value is send to Modbus device and similarly if a change is detected in Modbus device the VALUE variable is automatically updated. To make creation of formulas easier the value of variable VALUE is shown in *Properties* as the Value item.

The text value FORMATTED VALUE contains the number from numerical variable converted to text according to formatting rules stated in Format option in *Properties*. The formatting of the text variable FORMATTED VALUE is processed only when the new value is received from Modbus device. When the value of the VALUE variable is changed from the Control4 Programming script the content of the text variable FORMATTED VALUE does not change.

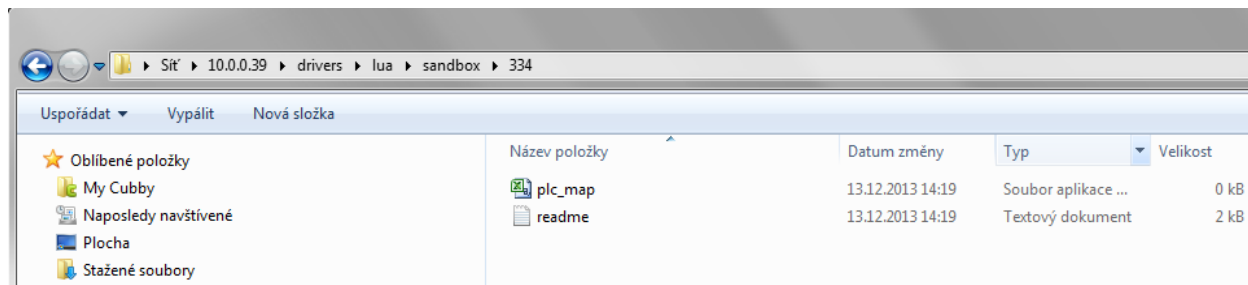
Troubleshooting and problems

I can not load the plc_map.csv file to controller

1. Click on Windows icon then choose the „Computer“ item.
2. After the Windows explorer opens enter the path of the directory where the configuration file should be saved to address field. The path can be find in driver *Properties* in *Config File Location* field.
3. Copy the `plc_map.csv` configuration file to this directory
4. Then do the restart of the controller/the Control4 system director.



Warning: When an IP address is entered in Windows explorer, you must enter two backslashes before the IP address, or the controller system file won't show (only the Control4 logo)!



There are no connections in Modbus PLC driver

Check if you created the `plc_map.csv` file correctly and if it does not contain a typo. Check if the file is loaded in proper directory which is stated in driver *Properties*. Click to *Process Config File* in *System Design – Actions* button.

Error reporting

Please contact YATUN support at podpora@yatun.cz with any driver-related support questions.

We kindly ask you to provide detailed reports:

- Clear description of the problem with steps to reproduce.
- Control4 hardware & software used (e.g. HC-800 running Control4 OS 2.5.2).
- Hardware & software version info related to Modbus devices used.
- Control4 project backup (without media and personal information).
- The `plc_map.csv` file.
- Driver versions (from the *Properties* or *Documentation* tab).
- Debug logs of *Debug* or *Fine* level (if applicable).
- Some drivers allow you to generate a *Startup Log* or *Debug Dump* which can also be useful.
- If it's a visual problem (in navigators) please make screenshots or a short video.

Every bit of information helps. In some cases we will ask you to arrange us remote access to the system so we can debug remotely.

Licensing

Without entered licence key the driver works for six hours in trial mode which allows free testing of its functionality. To activate additional six hours for testing you have to restart the director. Following information about licensing is available in *Composer Pro – Properties*:

Licence: Key – purchased licence key has to be entered in it. This licence key activates the driver to its fully function state. The entered licence key has to be confirmed by clicking to Set button.

Licence: Status – indicates if the driver is fully functioning or in the trial mode or if the trial mode expired. It shows for how long can the unlicensed driver be used in testing trial mode.

Licence: Director MAC – the MAC address of the controller where the Director module runs. This address has to be stated when the licence is being purchased.

Licence :Director Version – the firmware version of the controller where the Director module runs. It is suitable to state it when errors are reported.

Licence: Driver Version – the number of the driver version. It is suitable to state it when errors are reported.

Annex 1 - Creating the plc_map.csv driver

Rules

1. The name of file has to be plc_map.csv.
2. The end of line can be unix (LF) or windows (CR+LF), file coding UTF-8.
3. Columns are separated by a comma.
4. The first row contains headers – names of each column. Allowed column names in the header are: NAME, TYPE, ADDRESS, BIT and LOCATION.
5. Comments in the configuration file – anything between /* and */ or (* a *) is omitted, anything after // or -- is skipped.
6. Parameters in columns NAME, TYPE and ADDRESS are required!
7. NAME is the name of the element which will be transferred to Control4 (e.g. Hallway light, return water temperature ...) as the connection name.
8. TYPE – allowed data types are: LIGHT, RELAY, CONTACT, DIMMER, TEMP, TEMP_10, VAR_FLOAT, VAR_DOUBLE, VAR_UINT16, VAR_INT16, VAR_UINT32, VAR_INT32, VAR_UINT64, VAR_INT64.
9. Column ADDRESS – the address of the element in memory map of connected Modbus device (the number between 1 – 65535).
10. Column LOCATION (optional) is being used for the adjustment of the method of storing data in the connected device. As was mentioned in the chapter *Modbus memory space and plc_map.csv file* sometimes each types can be stored in different memory sections than usual or even advisable. By entering the LOCATION the default location of a given type can be changed and the driver will try to deal with it the best way considering supported modbus

functions in device. Default Location settings:

TYPE	Default Location
LIGHT, RELAY	= 0 (Coils)
CONTACT	= 1 (Digital Inputs)
DIMMER, TEMP, TEMP_10	= 4 (Holding Registers)

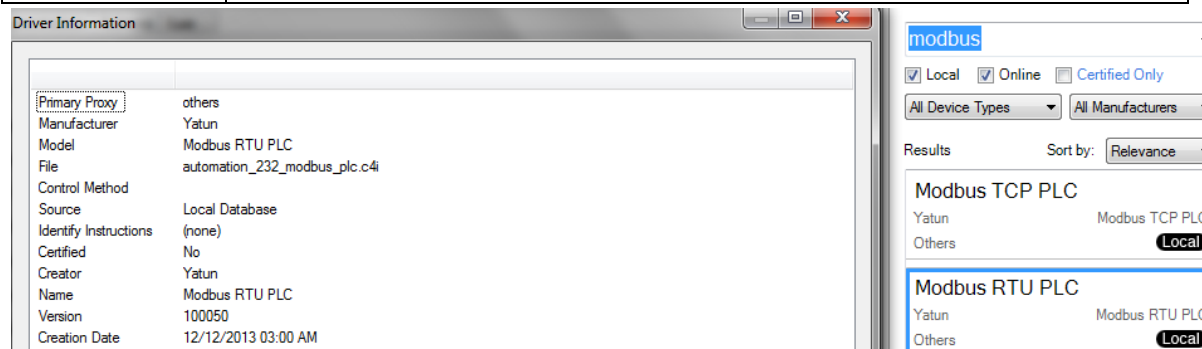
11. BIT parameter has to be used if bit types (RELAY, LIGHT, CONTACT) are stored in registers. Specifies the position of the bit value in the register. The value range is 0 – 15.

Example of creating plc_map.csv driver

http://www.moxa.com/doc/man/ioLogik_E1200_Series_Users_Manual_v10.pdf, chapter A-1.

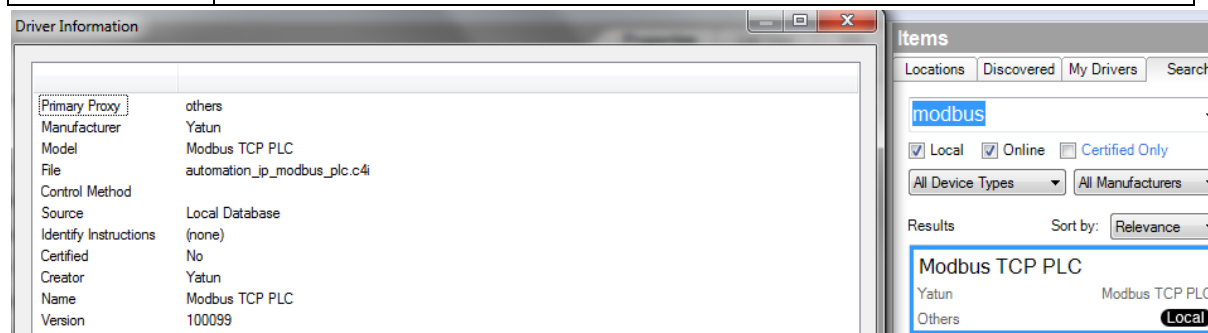
Annex 2 – List of drivers

File	automation_232_modbus_plc.c4i
Name	Modbus RTU PLC
Device Type	--others--
Manufacturer	Yatun
Description	Protocol driver for communication with devices using Modbus protocol serial link.



File	automation_ip_modbus_plc.c4i
------	------------------------------

Name	Modbus TCP PLC
Device Type	--others--
Manufacturer	Yatun
Description	Protocol driver for communication with devices using Modbus TCP protocol connected through computer network.



File	variable_number.c4i
Name	Number Variable
Device Type	--others--
Manufacturer	Yatun
Description	Driver allows to work (transformation etc.) with variables from devices connected into Control4 system via Modbus protocol.

