



www.rexcontrols.com/rex

Getting started with REX and WAGO PFC100/200

User guide

REX Controls s.r.o.

Version 2.50.5

2017-09-06

Plzeň (Pilsen), Czech Republic

Contents

1	Introduction	3
1.1	Features of the REX Control System	3
1.2	Structure of the REX Control System	4
1.3	Programming in the REX Control System	4
1.4	Main components of the REX Control System	5
1.4.1	RexDraw – the development environment	5
1.4.2	RexHMI Designer – the user interface	5
1.4.3	RexComp – the compiler	5
1.4.4	RexCore – the runtime core	5
1.4.5	RexView – the diagnostic tool	5
2	Installation of the development tools	6
2.1	Windows 7/8/10	6
2.1.1	Installation procedure	6
2.1.2	Uninstall procedure	7
3	Installing the runtime modules of the REX system on WAGO PFC100/200	8
3.1	Install using operating system image	8
3.2	Installation into an existing operating system	9
3.3	Installed files and folders	10
4	Configuration, compilation and execution	11
4.1	Creating a new project	11
4.2	Compiling and running a project	16
4.3	Adding a user interface (HMI)	22
4.4	Ready for interaction with the outer world	27
5	I/O configuration for the WAGO PFC100/200	28
5.1	Physical connections	28
5.2	Adding inputs and outputs to the project	30
5.3	Working with I/O signals	30
5.3.1	Modifications in the project main file	30
5.3.2	Modifications in the task	32
5.4	Updating the HMI	34

5.5	Additional information	35
5.5.1	Detailed description of the driver	35
5.5.2	Examples	36
6	Summary	37
A	Graphical HMI with RexHMI Designer	38
A.1	Initializing the HMI design	38
A.2	Adding the first HMI components	40
A.3	Linking HMI components with the running algorithm	43
A.4	Adding more HMI components	47
A.5	Additional links to the running algorithm	48
A.6	The final steps	49
B	Licensing of the REX Control System	52
B.1	Obtaining a DEMO licence	52
B.2	Obtaining a permanent licence	54
B.2.1	Activation of the permanent licence	54
	Bibliography	58

Chapter 1

Introduction

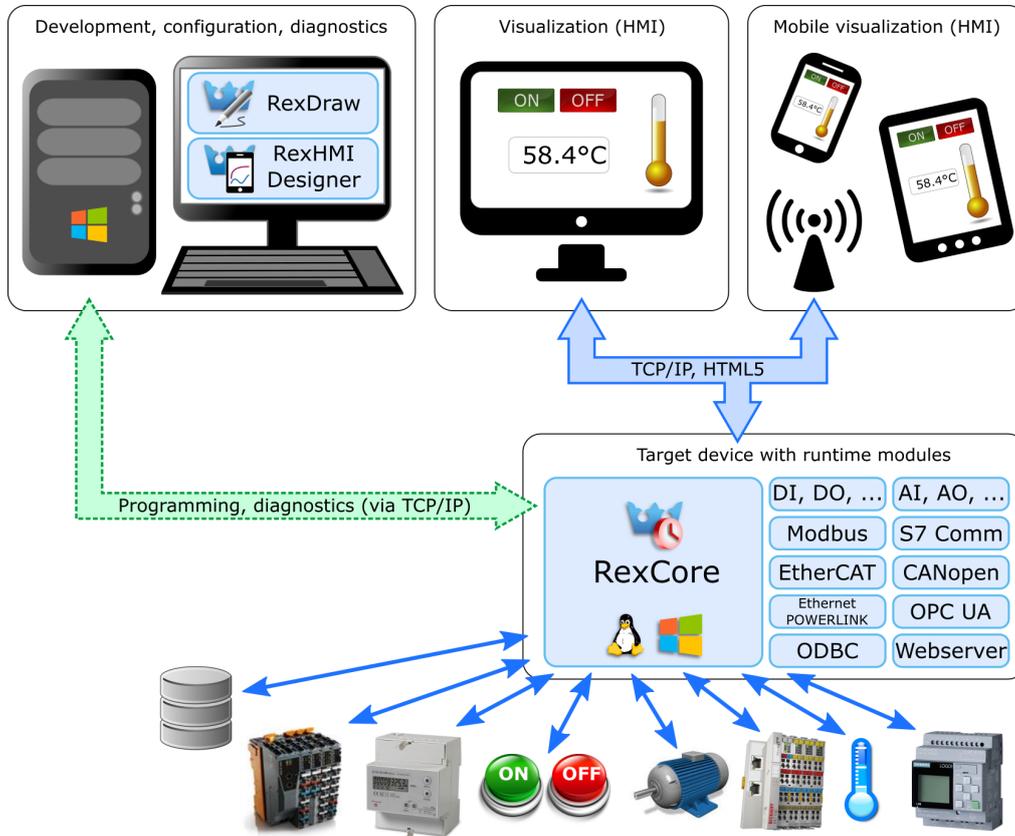
The REX Control System is a family of software products for automation projects. You can use it in all fields of automation, robotics, measurements and feedback control.

The runtime core of the REX Control System turns your WAGO PFC100/200 into a programmable device which will run your algorithms.

1.1 Features of the REX Control System

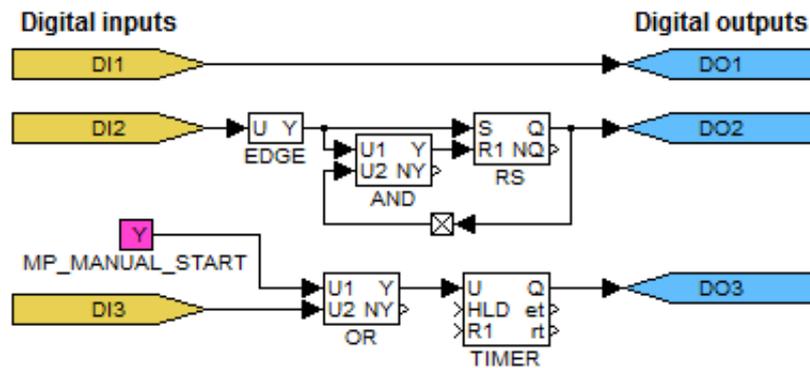
- Graphical programming without hand-coding
- Programming control units on a standard PC or laptop
- User interface for desktop, tablet and smartphone (HMI)
- Wide family of supported devices and input-output units (including WAGO PFC100/200)
- Industry-proven control algorithms
- Easy integration into business IT infrastructure (ERP/BMS)
- REST API for seamless integration into Industry 4.0 and (I)IoT solutions

1.2 Structure of the REX Control System



1.3 Programming in the REX Control System

The REX Control System offers a graphical development environment for programming the algorithms. You can use standard desktop or laptop PC. You create the algorithms from the so-called function blocks. The library includes countless items (timers, comparators, filters, PID controllers and many more).



1.4 Main components of the REX Control System

1.4.1 RexDraw – the development environment

The RexDraw graphical environment is a developer's tool which runs on a standard PC with Windows. You create the algorithms using the function block library¹ of the REX Control System [1]. The library contains simple comparators and timers as well as advanced blocks for signal processing and feedback control (PID controllers etc.). You compile your project on your PC and run it on your WAGO PFC100/200.

Once running, you can watch your algorithm in real-time. Just select the signals and function blocks of your interest. You can also obtain detailed hierarchical information about the running algorithm and diagnose the runtime core and execution of your algorithm. You can connect via local network or over the Internet.

1.4.2 RexHMI Designer – the user interface

The RexHMI Designer is another developer's tool intended for designing graphical user interface (or HMI, Human Machine Interface, if you prefer) for your algorithms. The user interface is included in the project and it is copied to your WAGO PFC100/200 along with the algorithm.

1.4.3 RexComp – the compiler

The RexComp compiler converts your algorithms into binary code of the REX Control System. The compiler is almost invisible for the user, it is invoked from the RexDraw development environment. The compiler detects and reports possible errors in your algorithms.

1.4.4 RexCore – the runtime core

The RexCore runtime core runs on the target device (WAGO PFC100/200). It handles timing and execution of your algorithms and provides various services. The individual tasks are prioritized and executed using preemptive multitasking.

RexCore further contains an integrated webserver providing user interface (HMI) and REST API.

1.4.5 RexView – the diagnostic tool

With RexView you can diagnose the runtime core and execution of your algorithm. It is an alternative standalone tool for commissioning and diagnostics of control algorithms. All the diagnostic functions are included in RexDraw, therefore RexView can be considered obsolete. You can connect via local network or over the Internet.

¹The IEC 61131-3 standard defines Function Block Diagram (FBD) as one of the PLC programming techniques.

Chapter 2

Installation of the development tools

This chapter describes the steps to install and uninstall the development tools of the REX Control System on Windows 7/8/10 operating systems.

2.1 Windows 7/8/10

The installation package of the REX Control System development tools contains the RexDraw and RexHMI Designer development environments, the RexComp compiler and the RexView diagnostic tool. It also includes the RexCore runtime module for developing and testing purposes.

The installation package can be downloaded from <https://www.rexcontrols.com/software-download>.

The installation process requires the administrator rights on your PC.

2.1.1 Installation procedure

1. Run the `REX-X.XX.XX.XXXX-PPP.exe` downloaded from <https://www.rexcontrols.com/software-download>.
2. Select the language and follow the installation wizard.
3. Select the target installation folder, the default is `C:\Program Files (x86)\REX Controls\REX X.XX.XX.XXXX`.
4. Afterwards you can select the components to install. The requirements are quite low (approx 400 MB disk space) therefore the `Full install` option is recommended.
5. The following steps are standard and do not require further explanation.

There is no need to restart the system after installation.

2.1.2 Uninstall procedure

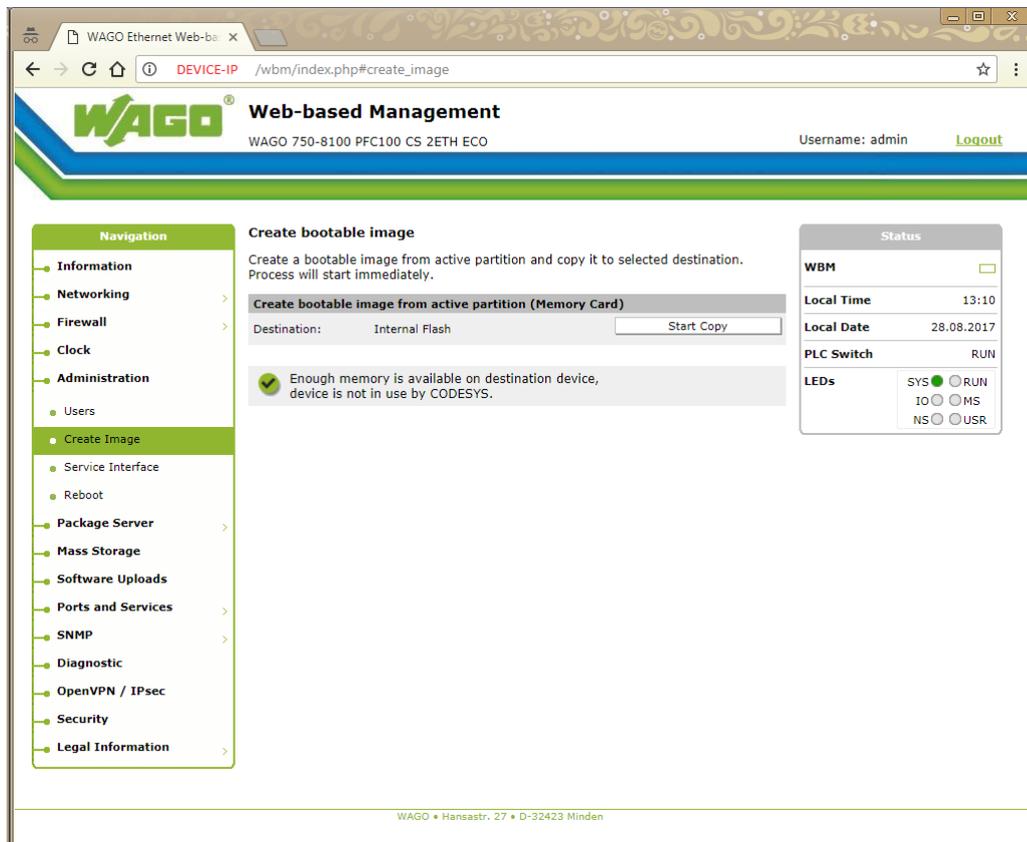
The common procedure can be used to uninstall the REX Control System – go to *Control panel* and choose *Install/Uninstall programs*.

Chapter 3

Installing the runtime modules of the REX system on WAGO PFC100/200

3.1 Install using operating system image

1. It is highly recommended to use the official REX OS image for WAGO PFC100/200 which you can download from <http://download.rexcontrols.com/files/wago/2.50.5/rex-2.50.5-sd.img>
2. After you download the file, flash it onto a micro SD card. On Windows machines you can use e.g. the Win32DiskImager or similar tool. On Linux machines, use the dd command.
3. Afterwards insert the micro SD card into your WAGO PFC100/200 and power-up the device. By default, WAGO PLCs boot from SD card if present. So the new operating system is immediately started. The image already contains all the runtime modules of the REX system.
4. Connect to the WAGO web configuration interface on address <http://device-IP-address/wbm>
Click on the "login" link, the default account is "admin" and password "wago".
5. Go to *Administration* -> *Create Image* to transfer the operating system image to the internal storage.
6. Press the *Start Copy* button and save the current configuration.



7. Power-off the Wago PLC, remove the SD card and power it on again.

Now the RexCore runtime module will be started automatically in the background as a daemon whenever you power up your WAGO PFC100/200. This allows you to establish connection between the host PC and the WAGO PFC100/200 using the RexDraw program and develop your automation solutions.

3.2 Installation into an existing operating system

Proceed as follows to install REX runtime on a running PFC100/200 platform:

1. Connect to the WAGO web configuration interface on address `http://device-ip-address/wbm`
Click on the "login" link, the default account is "admin" and password "wago".
2. In the *PLC Runtime -> General Configuration section*, set the runtime to "None".
3. Connect via SSH (PuTTY) to the device (user "root", password "wago") and install REX using the following commands.
4. Add REX installation repository to the IPKG packaging system

- ```
echo src rex http://download.rexcontrols.com/repositories/wago-rex-2.50
 » /etc/ipkg.conf
```
5. Update cache of the IPKG packaging system

```
ipkg update
```
  6. Install the runtime module. The rexcore package contains all standard components of the REX runtime.

```
ipkg install rexcore
```
  7. Update openssl:

```
ipkg install openssl
```
  8. Install bzip2 and lighttpd:

```
ipkg install bzip2
ipkg install lighttpd
```
  9. Restart the device:

```
reboot
```
  10. The REX runtime is started automatically after reboot is complete and you can connect to the device with RexDraw [2].

### 3.3 Installed files and folders

Please refer to the RexCore user guide [3] for detailed information about the files and folders of the REX Control System runtime modules.

## Chapter 4

# Configuration, compilation and execution

The process of creating a control algorithm will be demonstrated on a very simple example with four Boolean variables representing manual switches. In Chapter 5.3, two of them will be replaced by physical inputs of the WAGO PFC100/200. A software timer will be used for measuring the time when the variables are true (i.e. the switches are in the ON position). A Boolean signal will indicate that the interval of predefined length has elapsed.

### 4.1 Creating a new project

The project configuration is created using the `RexDraw` program. Each project consists of at least two `.mdl` files. The first file is the main file of the project, which is used for configuration of tasks, drivers, priorities and timing. The other file(s) contain the individual control algorithms (tasks).

First we'll create the example 0101-01 from scratch<sup>1</sup>.

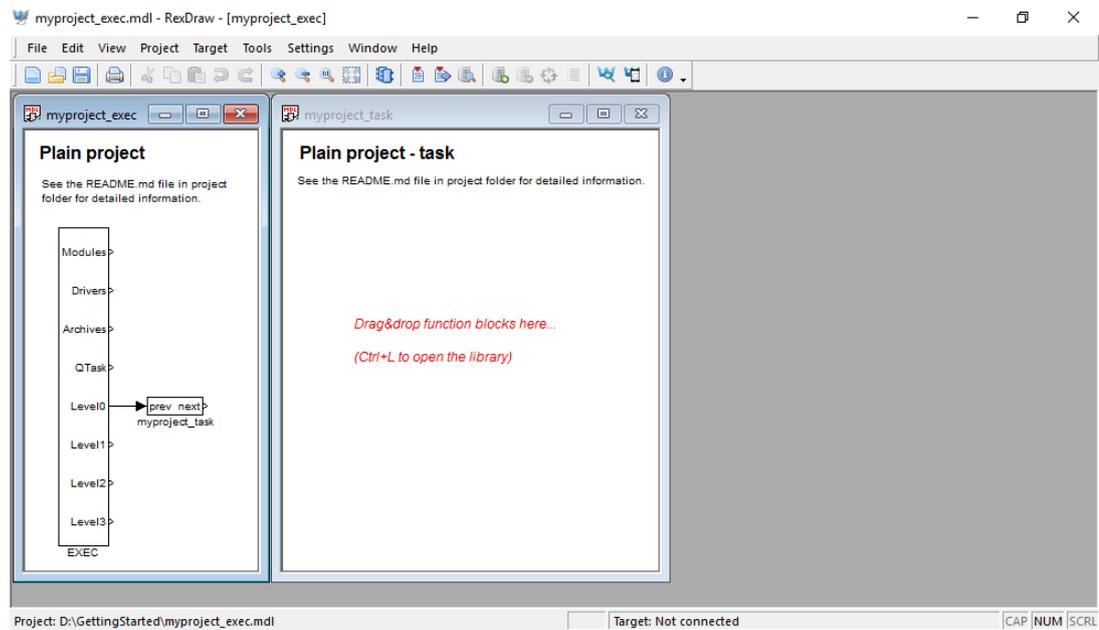
Standard approach:

1. Run the `RexDraw` program. Start with a plain project and select a folder to save the project files in (e.g. `D:\GettingStarted`).
2. The folder will contain two important files:
  - `myproject_exec.mdl`
  - `myproject_task.mdl`
3. The `myproject_exec.mdl` is the project main file. It contains one `EXEC` block from the `EXEC` library. The other block is the `TASK` block from the same library

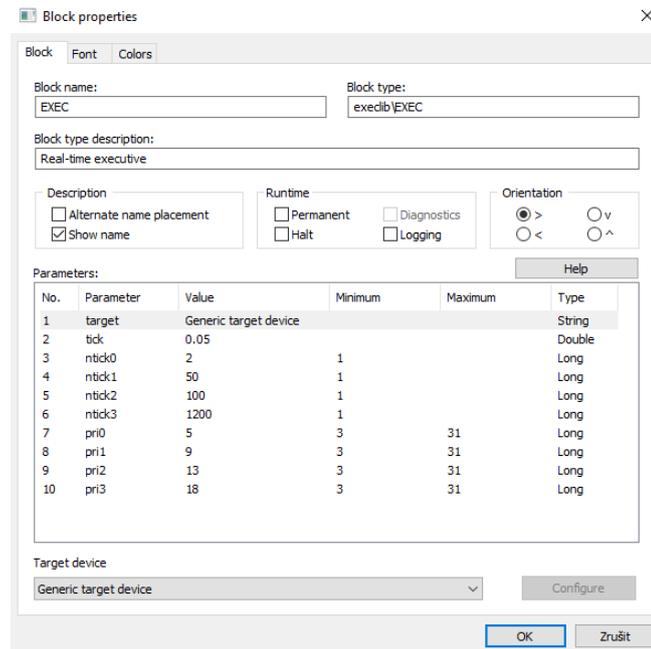
---

<sup>1</sup>All examples which are part of the installation are marked by an ExampleID. The most up-to-date examples are available at <https://github.com/rexcontrols/REXexamples/archive/v2.50.zip>

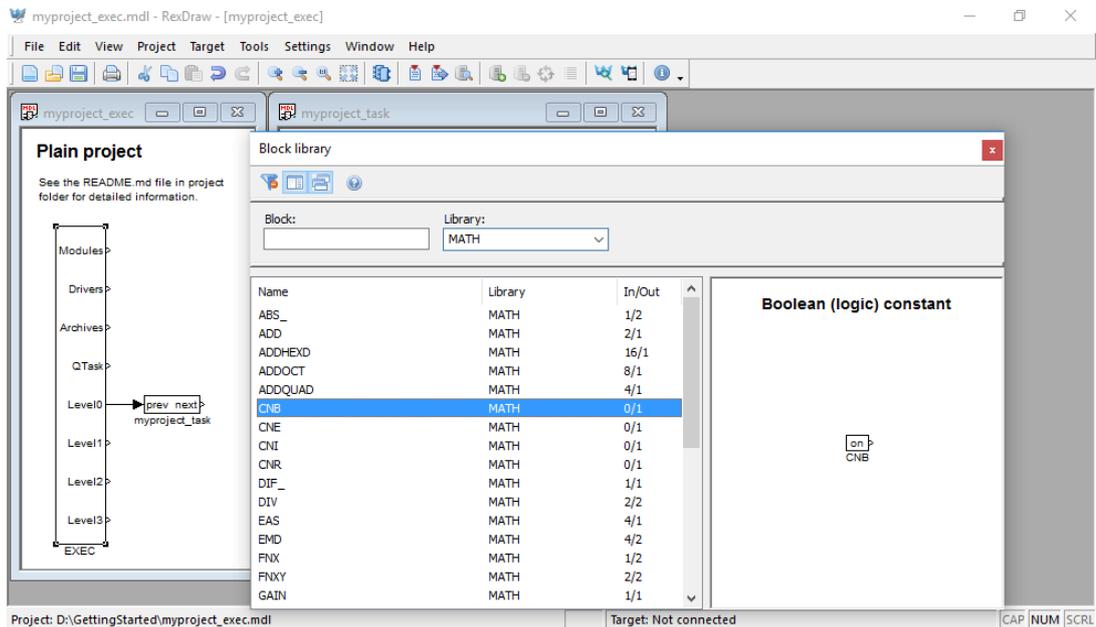
and it is renamed to `myproject_task` to reference the second file of the project (`myproject_task.mdl`), which will contain the algorithm (the so-called task).



4. The task is connected to the `Level0` output of the `EXEC` block and therefore its timing is defined by `tick` and `ntick0` parameters of the `EXEC` block.
5. The `EXEC` block (and any other block) can be configured by double-clicking on it. A block parameters and properties dialog appears. The parameters of all blocks of the REX Control System are described in the Help (press the F1 key) and in the Block reference manual [1].

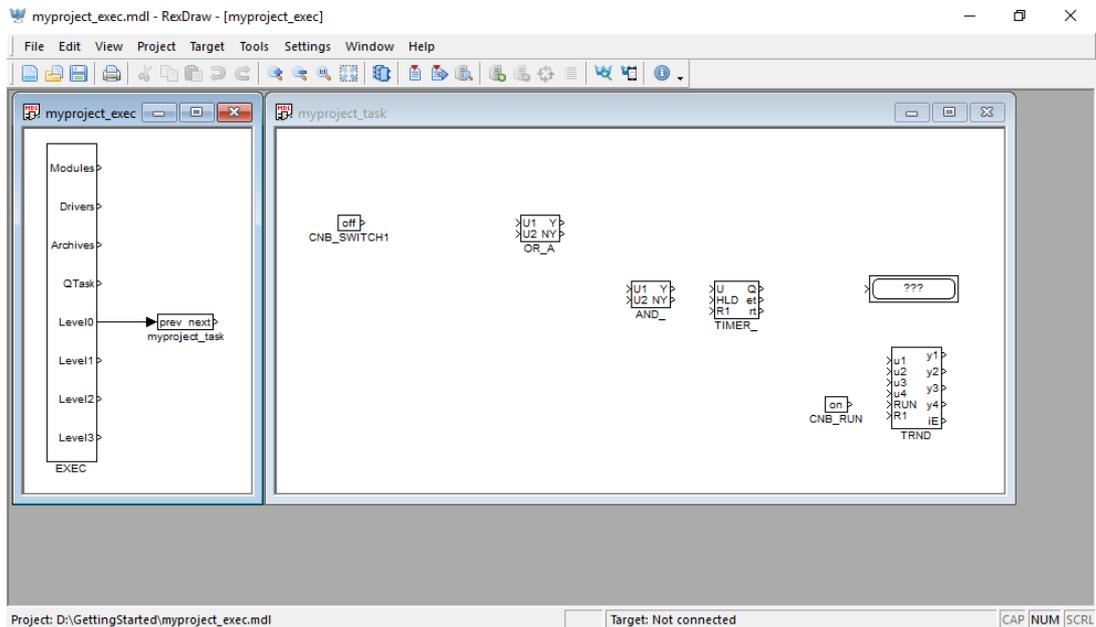


6. Note that `tick=0.05` and `ntick0=2`, therefore the task will run each 100 milliseconds ( $0.05 \cdot 2 = 0.1s$ ). There is no need to change any parameter at the moment. Close the dialog.
7. You can delete all the descriptive texts in project files. These have no effect on the functionality and can be considered programmer's comments.
8. Open **Block Library**, choose *View/Block Library* in the menu or use the  icon from the toolbar.
9. The blocks are ordered in alphabetical order. Further the blocks are also organized in sublibraries and the location is always denoted as **sublibrary/block**, e.g. **LOGIC/AND** for the logical AND block in the LOGIC sublibrary.

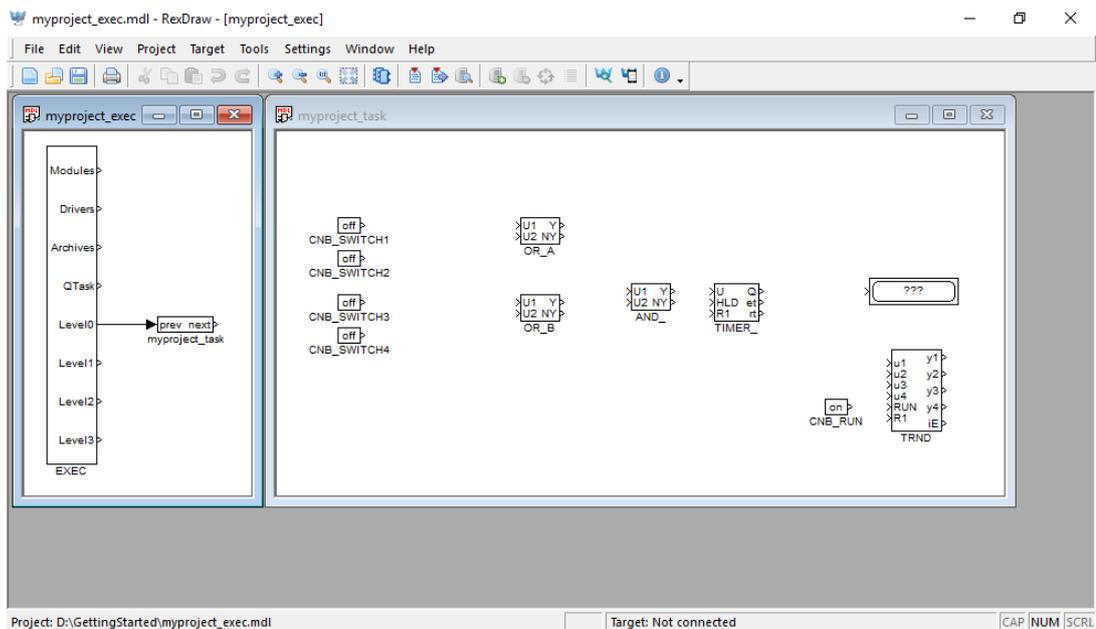


10. Locate the following function blocks in the Block Library and drag them to the task file:

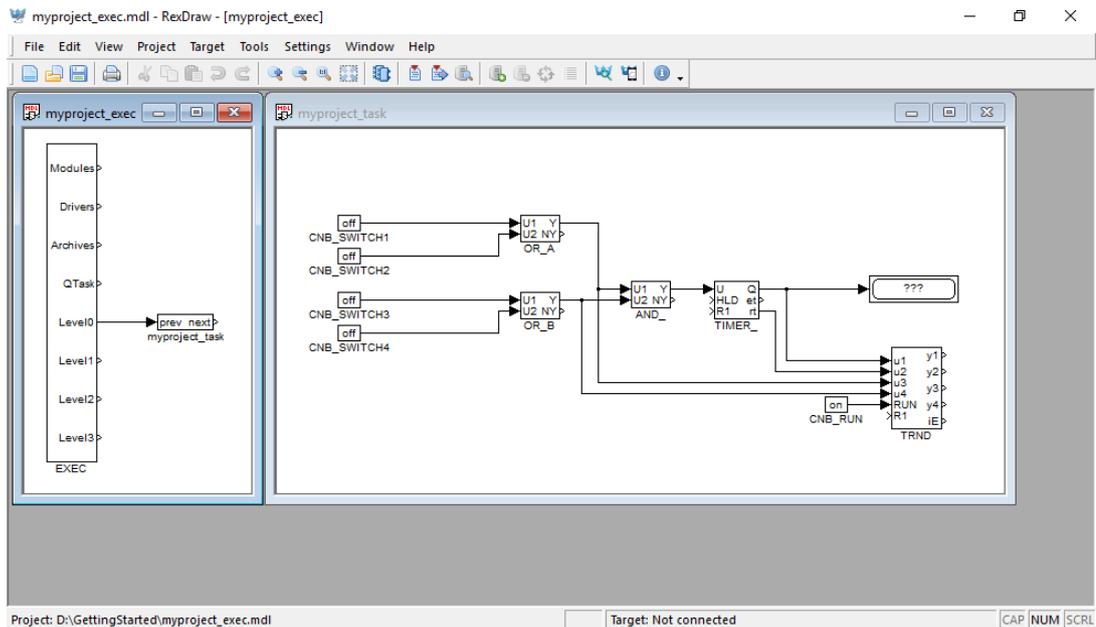
- MATH/CNB – constant of type Boolean. Once dropped, double-click its name and change it to CNB\_SWITCH1. Double-click the block and set parameter YCN = off.
- LOGIC/OR\_ – logical OR. Rename it to OR\_A.
- LOGIC/AND\_ – logical AND.
- LOGIC/TIMER\_ – a timer block. Set parameter mode = 2: Delayed ON, pt = 5.0.
- INOUT/Display – a display to show values in real-time.
- ARC/TRND – real-time recording. Set parameters l = 2000, Title = My timer, leave the default values otherwise.
- MATH/CNB – constant of type Boolean, change name to CNB\_RUN, set parameter YCN = on.



11. Duplicate the CNB\_SWITCH1 block with right mouse button dragging. Or simply Copy&Paste the block.
12. Duplicate the block 2 more times.
13. Duplicate also the OR\_A block. Rename the duplicate to OR\_B.



- Connect the blocks as shown below. To connect the blocks, drag the output arrow of one block to the input arrow of the other block using the left mouse button. The connection will be established when the line goes bold and green. After releasing the mouse button you can recognize a successfully connected line by its style. A full line terminated by a full arrow at the input of the connected block indicates a valid connection. New branch of an existing line can be created by dragging an existing line with the right mouse button.

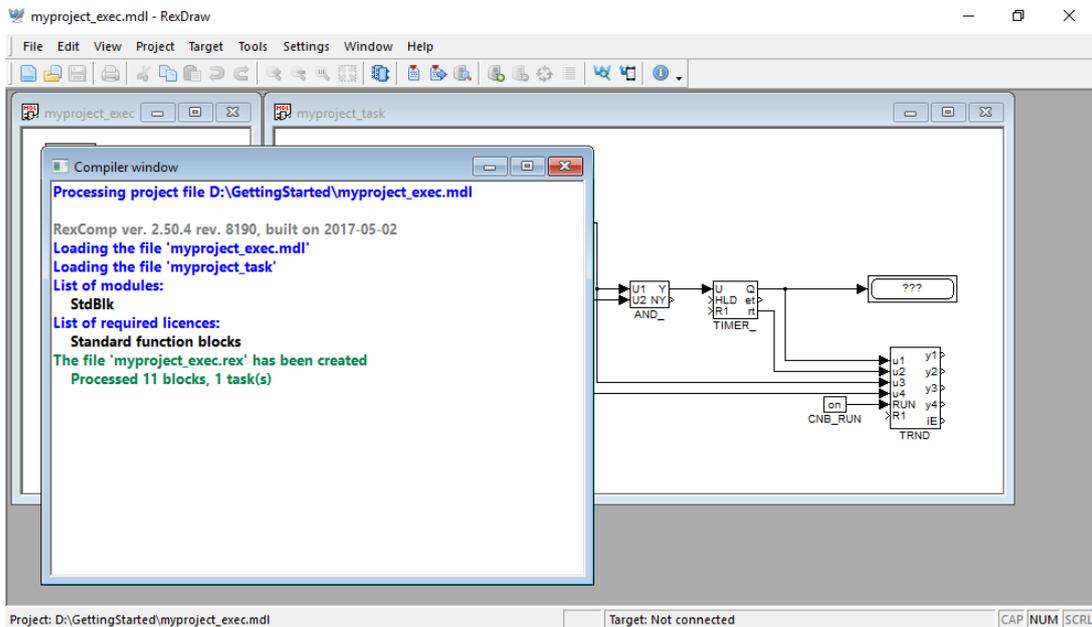


At this moment the executive configuration `myproject_exec.mdl` and the corresponding `myproject_task.mdl` file with the algorithm are ready. The algorithm will be evaluated in the direction of the arrows, starting from the source CNB blocks, passing through the OR\_, AND\_ and TIMER\_ blocks and finishing at the Display and TRND blocks.

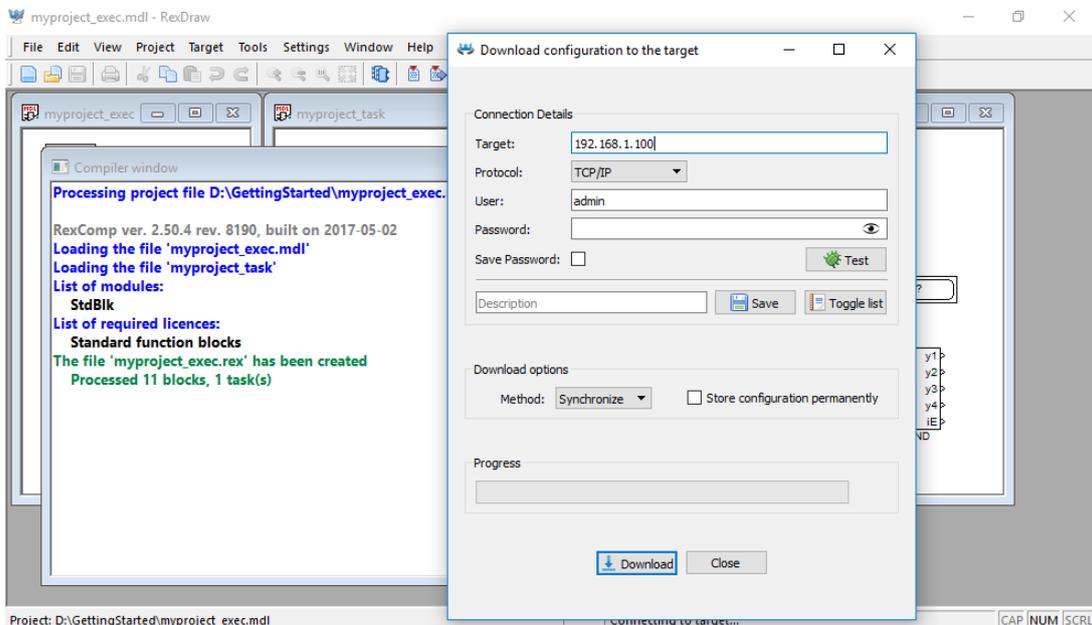
Congratulations, your first project is ready for compilation!

## 4.2 Compiling and running a project

The developed algorithm must be compiled to binary form prior to deploying. Pick *Project/Compile* from the menu or use the  icon from the toolbar. The compiler output is displayed in the **Compiler** window. If no error is found, the `myproject_exec.rex` file is created.



At this moment it is possible to deploy the control algorithm to the target platform. Use *Project/Compile and Download* in the menu or click the **Compile and Download** icon  for this purpose. A dialog for defining the target device appears upon successful compilation.

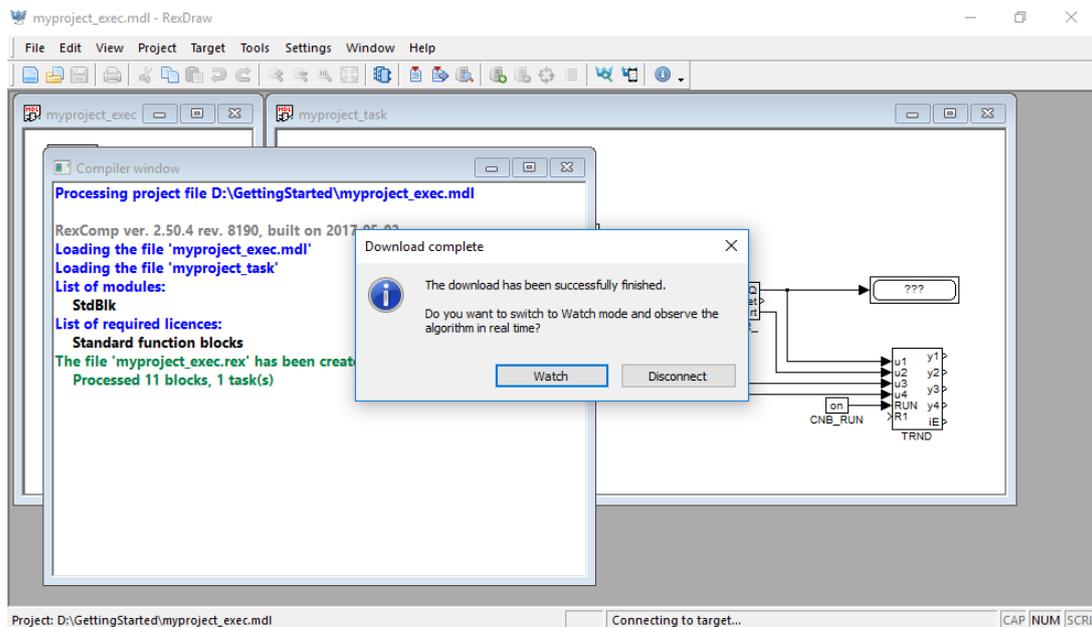


Enter the IP address of your WAGO PFC100/200 in the **Target** field. **The default user is admin and there is no password by default.** Leave the other elements

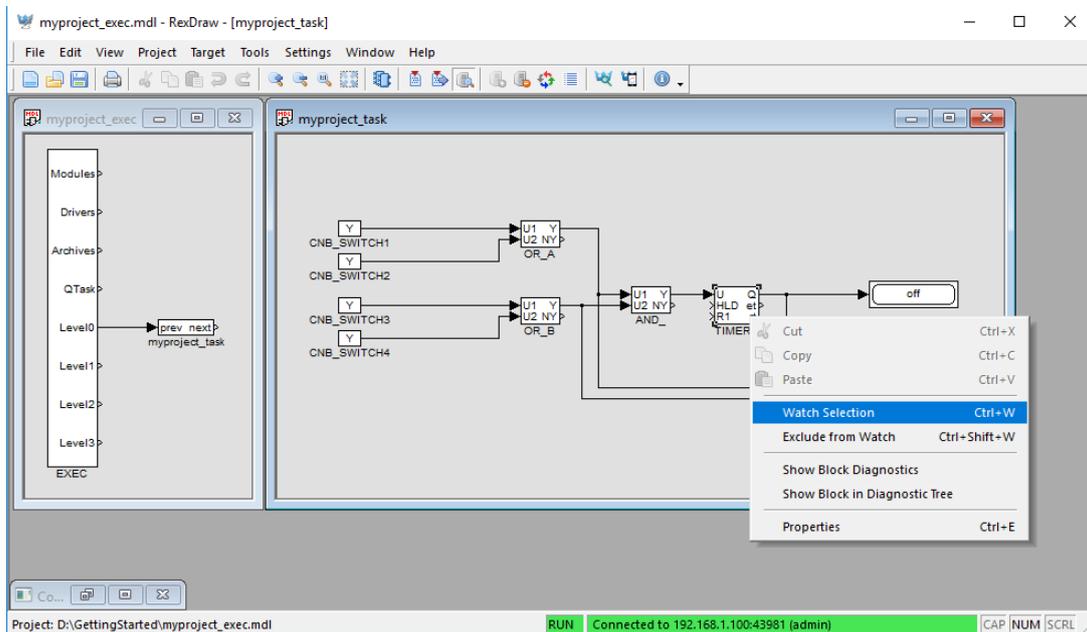
intact and click Download.

If there is no licence on your WAGO PFC100/200, you need to get one first. You can get a DEMO licence for free. See Appendix B for details and come back afterwards.

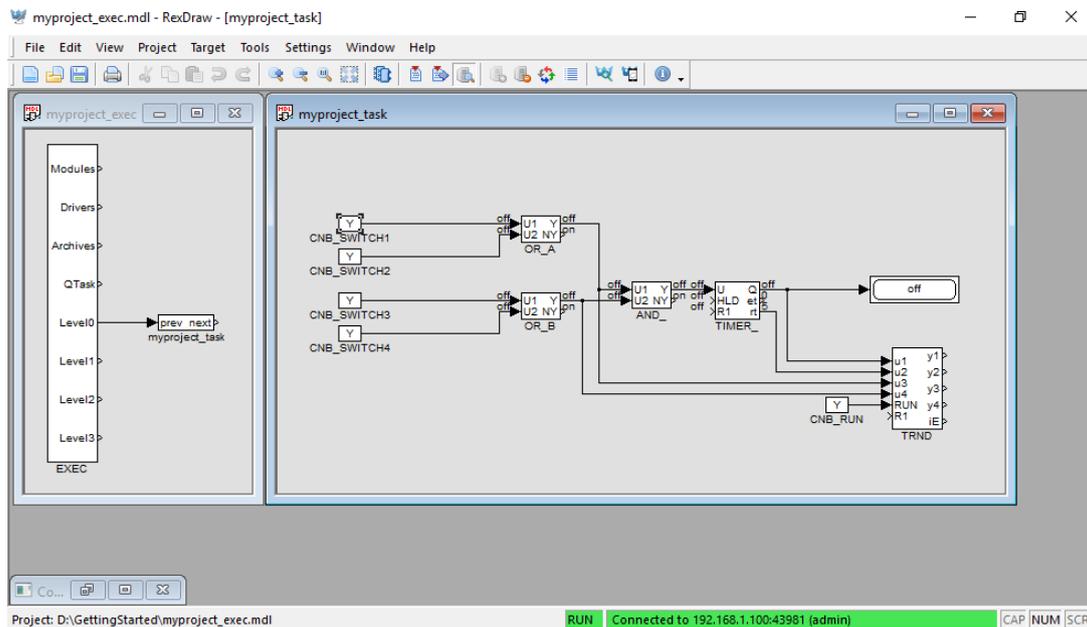
As soon as the download is complete it is possible to switch the RexDraw application to the so-called *Watch mode* and watch the control algorithm in real-time – click Watch.



In the *Watch mode*, the background of all files goes gray and you cannot move or delete any blocks or connections. Right-click the `TIMER_` block and select *Watch selection* in the menu to watch the inputs and outputs of the timer.



You can do the same with the OR\_ and AND\_ blocks (or any other selection).

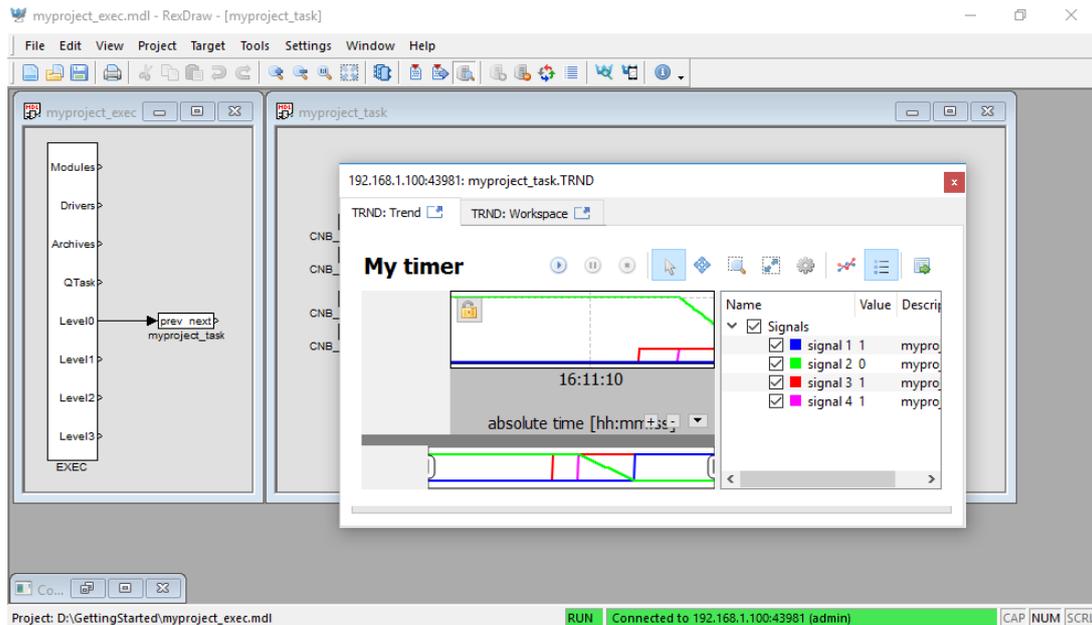


Now it is possible to double-click the CNB\_SWITCH1 block and change the Boolean variable to YCN = on (tick the checkbox and click OK). Once you do the same with the CNB\_SWITCH3 block, the outputs of both OR blocks are on and the Y output of the AND\_ block goes on and the TIMER\_ starts to count down. Observe the `rt` output.<sup>2</sup> Once the

<sup>2</sup>Do not get confused by the default 1 second refresh rate of the *Watch mode*. The algorithm on the target device runs each 100 milliseconds as mentioned earlier.

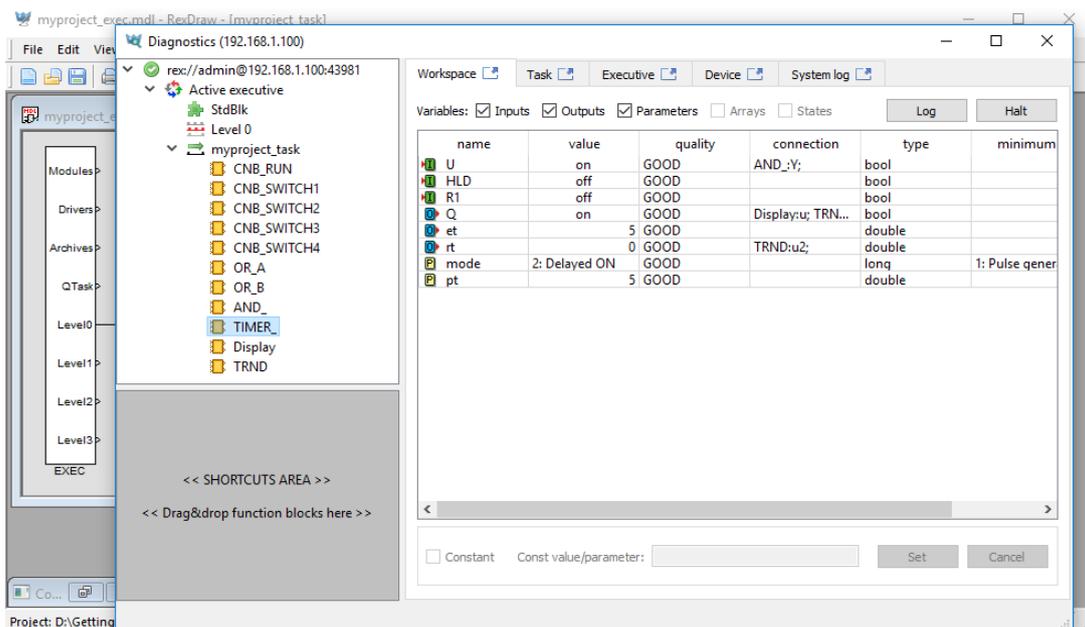
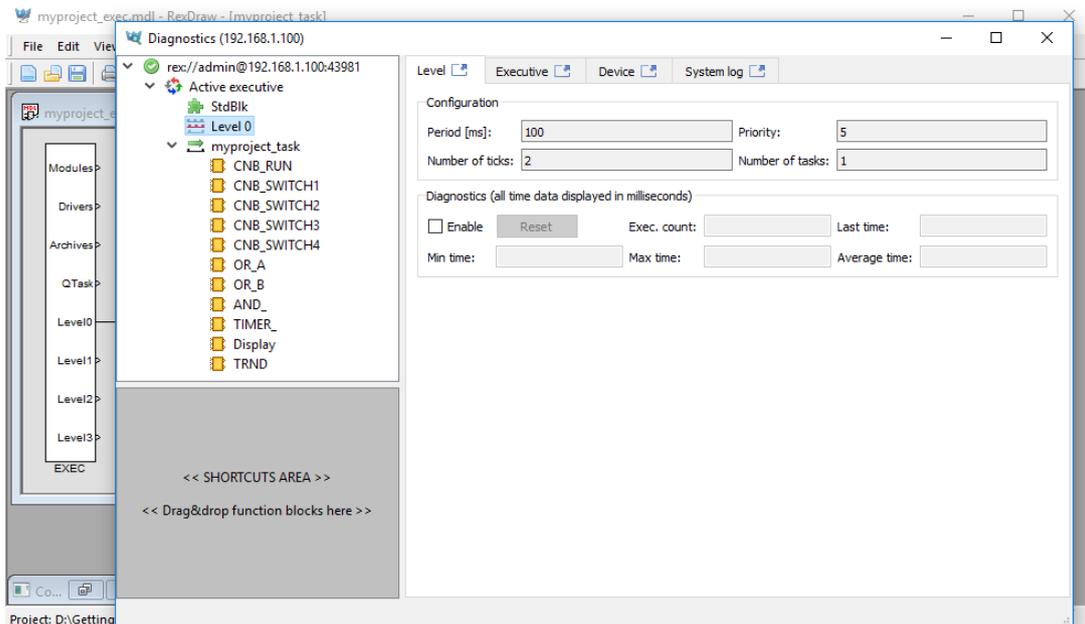
timer reaches zero, its output  $Q$  is set to on and it remains on as long as the  $U$  input is on.

You can double-click the TRND block to see the signals in a real-time graph. The red line is the output of the OR\_A block, the magenta line is the output of the OR\_B block, the green line is the remaining time of the timer and the blue line is the Boolean output of the timer.



Try turning the CNB blocks off and change the  $pt$  parameter of the  $TIMER_$  block. Afterwards turn the CNB blocks on again and observe the signals in the TRND block again. As you can see, you can modify any parameter in real-time, which allows you to fine tune your algorithm.

It is also possible to open a **Diagnostics** view of the algorithm. Pick *Target/Diagnostics* from the menu or click the  icon and you will see the algorithm in a tree view which allows you to monitor the control algorithm in full detail. You can verify that the sampling rate of your algorithm is indeed 100 milliseconds. You can also adjust parameters of individual function blocks, which has the same effect as modifying them directly in the *Block properties* dialog.



*Note:* There is also a standalone diagnostics program called RexView. Click the RexView icon  and confirm the IP address of the target platform.

Now you can switch RexDraw back to the *Development mode*. You can do so by deactivating the *Watch mode* (use the  icon). You are offered synchronization of the changed parameters with the source files of the project, choose *No* at this moment.

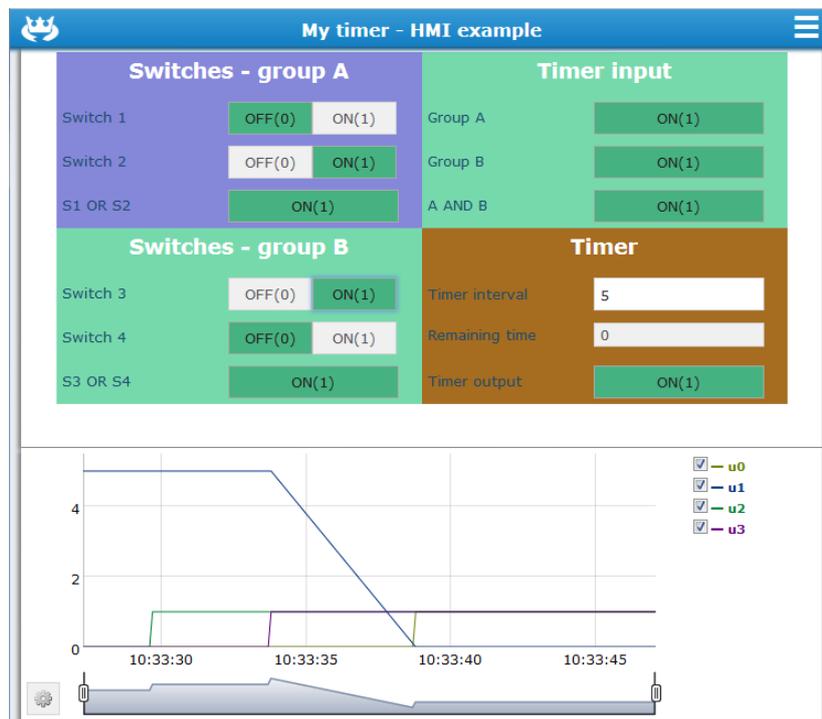
All changes made while in the *Watch mode* are not permanently stored in the target device (unless you decide so, see [2]). Upon restarting the *RexCore* runtime module the algorithm will start with the parameters defined in the project source files, which were valid when compiling and downloading the algorithm to the target device. To apply the changes permanently, you have to transfer the changes to the source files and *Compile and download* the project one more time which defines new startup values.

### 4.3 Adding a user interface (HMI)

The next step in developing a control algorithm is its user interface, or HMI, Human-Machine-Interface. It allows anyone (even those who are not familiar with the REX Control System) to interact with the algorithm. The HMI of the REX Control System relies on modern web-based technology and the HMI is therefore accessible via web browser on desktop PC, tablet or smartphone.

In this tutorial, a simple HMI will be created using the so-called WebBuDi technology. It provides very simple indicators and input elements to interact with the control algorithm via a web page (Web Buttons and Displays).

The steps to create the HMI are described below. This is how the HMI will look like in the end:



1. In the folder with the project files, create a `hmisrc` subfolder. Inside this folder, create a file named `index.hmi.js` and edit it with your favorite text editor. The content should be the following:

```

REX.HMI.init = function(){

//Indicators and virtual switches - group A
var switchesA = {
 column: 1,
 title: 'Switches - group A',
 rows: [
 {type: 'DW', alias: 'switch1', desc: 'Switch 1',
 cstring: 'myproject_task.CNB_SWITCH1:YCN'},
 {type: 'DW', alias: 'switch2', desc: 'Switch 2',
 cstring: 'myproject_task.CNB_SWITCH2:YCN'},
 {type: 'DR', alias: 'S1orS2', desc: 'S1 OR S2',
 cstring: 'myproject_task.OR_A:Y'},
]
};
REX.WebBuDi.addSection(switchesA);

//Indicators and virtual switches - group A
var switchesB = {
 column: 1,
 title: 'Switches - group B',
 rows: [
 {type: 'DW', alias: 'switch3', desc: 'Switch 3',
 cstring: 'myproject_task.CNB_SWITCH3:YCN'},
 {type: 'DW', alias: 'switch4', desc: 'Switch 4',
 cstring: 'myproject_task.CNB_SWITCH4:YCN'},
 {type: 'DR', alias: 'S3orS4', desc: 'S3 OR S4',
 cstring: 'myproject_task.OR_B:Y'},
]
};
REX.WebBuDi.addSection(switchesB);

//Timer input
var timerInput = {
 column: 2,
 title: 'Timer input',
 rows: [
 {type: 'DR', alias: 'inputA', desc: 'Group A',
 cstring: 'myproject_task.AND_:U1'},
 {type: 'DR', alias: 'inputB', desc: 'Group B',
 cstring: 'myproject_task.AND_:U2'},
 {type: 'DR', alias: 'AandB', desc: 'A AND B',
 cstring: 'myproject_task.AND_:Y'},
]
}

```

```

};
REX.WebBuDi.addSection(timerInput);

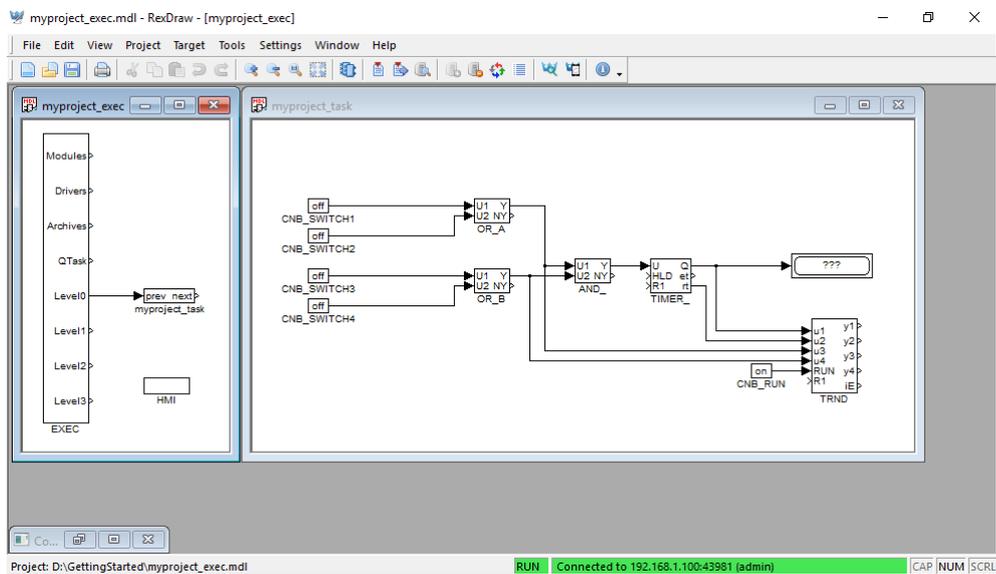
//Timer settings and status
var timer = {
 column: 2,
 title: 'Timer',
 rows: [
 {type: 'AW', alias: 'interval', desc: 'Timer interval',
 cstring: 'myproject_task.TIMER_pt'},
 {type: 'AR', alias: 'rt', desc: 'Remaining time',
 cstring: 'myproject_task.TIMER_rt'},
 {type: 'DR', alias: 'timerQ', desc: 'Timer output',
 cstring: 'myproject_task.TIMER_Q'},
]
};
REX.WebBuDi.addSection(timer);

//Add real-time trend
REX.HMI.Graph.addTrend({cstring: 'myproject_task.TRND'});
REX.HMI.Graph.setMaxBufferSize(200);

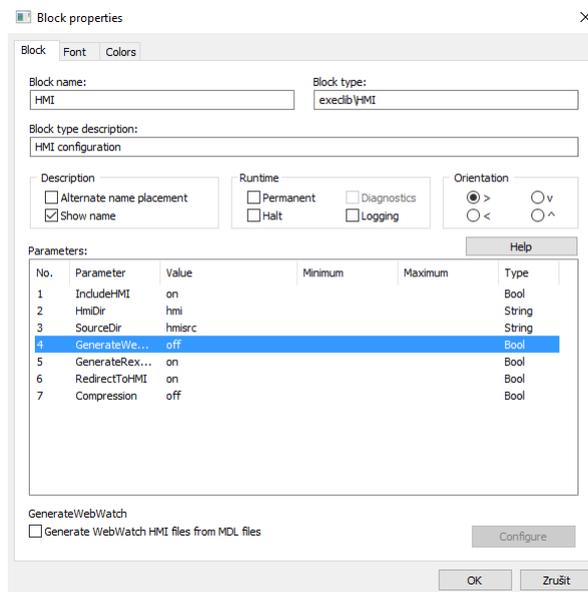
// Change title of the page
REX.HMI.setTitle('My timer - HMI example');
}

```

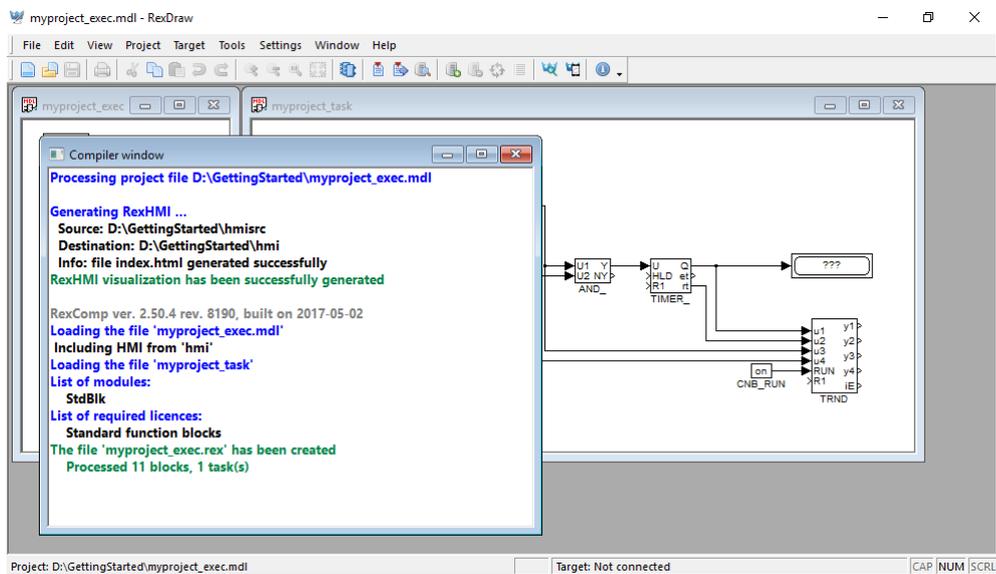
2. This file will be processed when compiling the project. However, it is necessary to add the EXEC/HMI block into the project main file first.



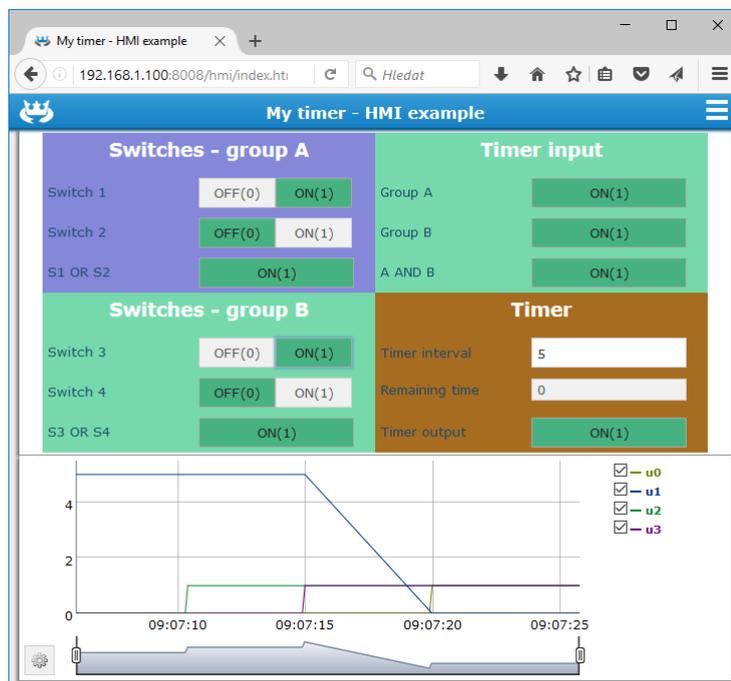
3. Double-click the HMI block to edit its parameters. Set **GenerateWebWatch** = **off** and confirm. WebWatch is another type of HMI, which you do not need at the moment. See [4] for details, it is a very interesting tool for developers and technicians.



4. The HMI is now an integral part of your project. Compile the project again and you will see that the compile log contains more information. The `index.html` file is generated from the source `index.hmi.js` file. All the HMI files are generated into the `hmi` subfolder (HTML, JS and CSS files) and included in the resulting binary `myproject_exec.rex` file.



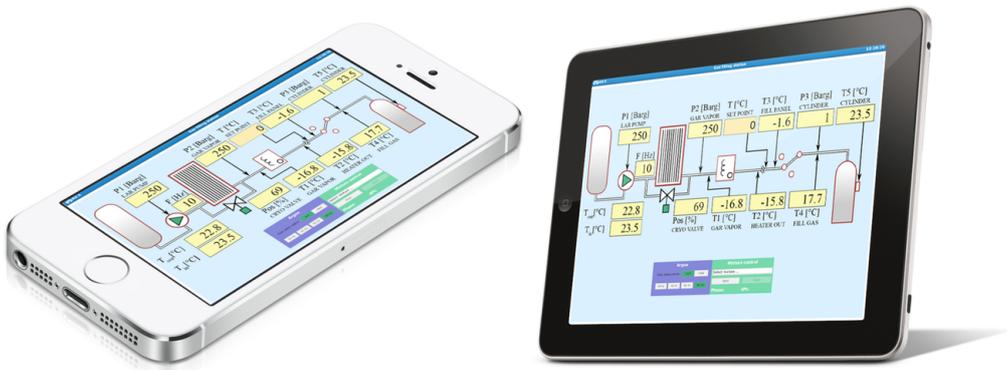
- After you download the project to the target device, you can access the HMI via web browser. Go to menu *Target/Web Interface* which will open the webpage. Remember the default login credentials: admin with no password.



- You can toggle the switches and observe the results. The virtual switches are linked to the individual CNB function blocks therefore the effect is the same as toggling the values directly in RexDraw.

7. You can also change the timer setting and shorten or lengthen the interval.
8. See [4] for detailed information about WebBuDi elements and possible customization (colors, backgrounds etc.).
9. The default URL address is <http://192.168.1.100:8008/hmi/index.html>.
10. The default port of the webserver (8008) can be changed in RexCore settings. See [3] for details.

Please note that there is also a program called RexHMI Designer, which allows you to create graphical user interfaces. Appendix A of this guide shows the steps to create a basic graphical interface. A more complex HMI is shown below solely for inspiration purposes.



## 4.4 Ready for interaction with the outer world

Well done! You have created the example 0101-01 from scratch<sup>3</sup>. You have learned the basic workflow for developing and running your algorithms using the REX Control System, which is the same for all platforms. Now it's time to add the so-called *input-output drivers* so that the algorithm can interact with sensors, actuators and external data.

---

<sup>3</sup>All examples which are part of the installation are marked by an ExampleID. The most up-to-date examples are available at <https://github.com/rexcontrols/REXexamples/archive/v2.50.zip>

## Chapter 5

# I/O configuration for the WAGO PFC100/200

The previous chapter illustrated the process of creating a control algorithm in the REX Control System and deploying it to the target device. But so far, the algorithm does not interact with the outer world, it is not connected to any physical signal (or external data).

We will use the input and output signals of the WAGO PFC100/200 for interaction with the real world.

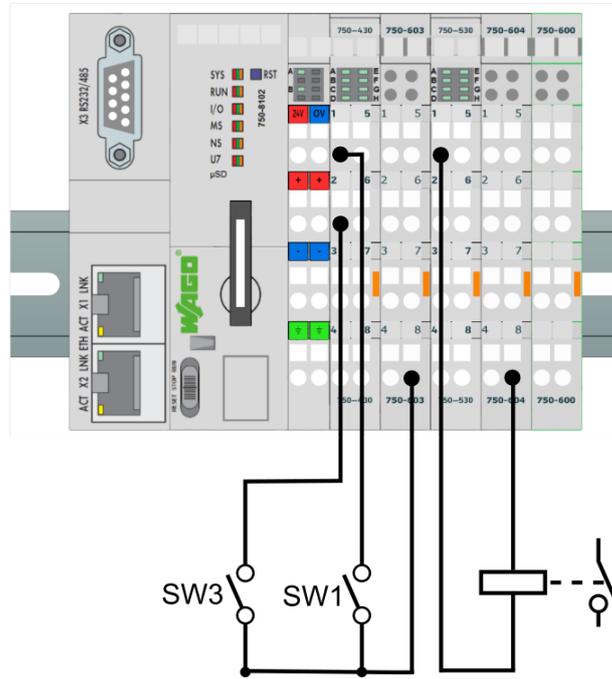
As mentioned in Chapter 4, two physical switches will be connected as inputs and a software timer will control one output signal. In other words, this chapter is devoted to advancing from the generic example 0101-01 to example 0102-01 for the WAGO PFC100/200.

### 5.1 Physical connections

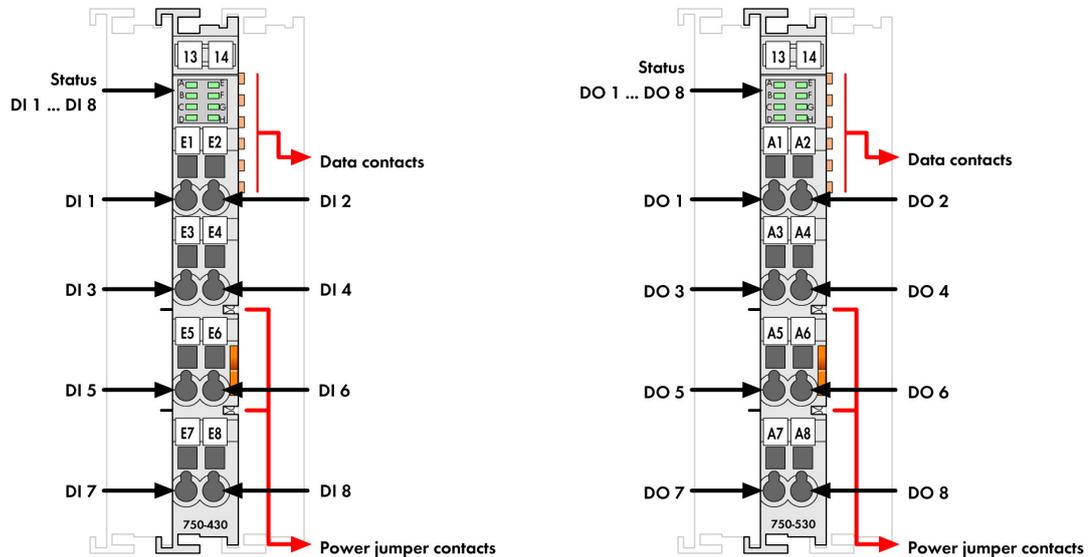
For this example project a simple setup is assumed:

- 1x 750-8120 PFC controller with power supply
- 1x 750-430 module (8 digital inputs)
- 1x 750-603 module (24V potential distribution)
- 1x 750-530 module (8 digital outputs)
- 1x 750-604 module (0V potential distribution)
- 1x 750-600 mandatory end module

Connect the switches and an external relay as shown below. The relay is optional, the onboard LED will indicate the status of the output.



IMPORTANT: Note that numbering of the terminals on the WAGO modules (above) does not comply with channel numbering (below)! Always follow the DI1..8, DO1..8, AI1..4, etc. notation as per WAGO documentation.

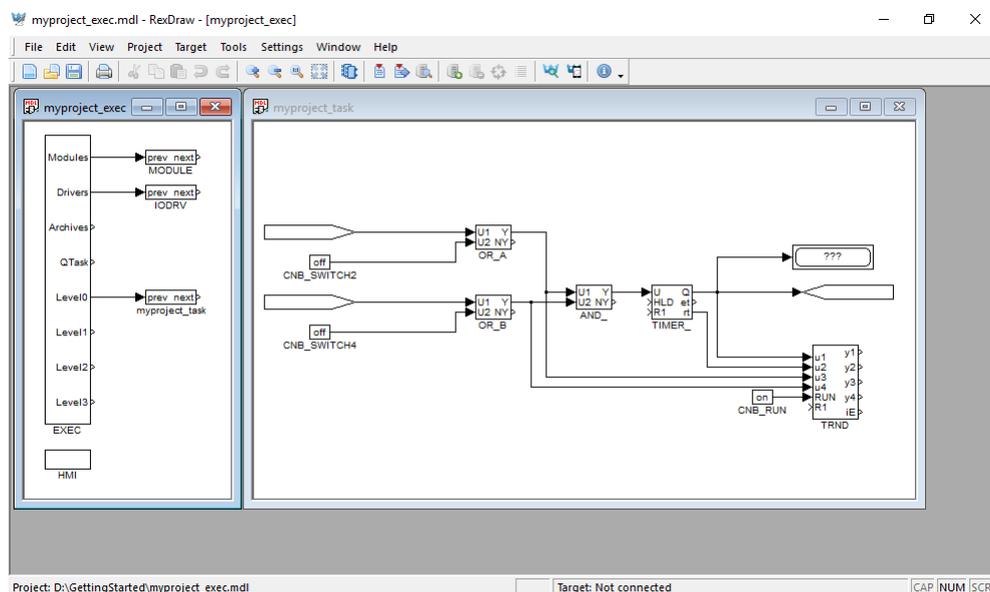


For details and wiring examples for WAGO PFC100/200 refer to documentation of the WAGO 750/753 modular I/O system.

## 5.2 Adding inputs and outputs to the project

Now when we have everything wired up, it's time to include the physical signals into the algorithm. You need to expand your project main file with 2 additional function blocks to access the inputs and outputs from the control algorithm in your project. Insert the **EXEC/MODULE** and **EXEC/IODRV** blocks from the *Block library* and attach them to the **EXEC** block as shown below.

In the task file, delete the **CNB\_SWITCH1** and **CNB\_SWITCH3** blocks and replace them with **INOUT/From** blocks. These will be the input signals. Also add one **INOUT/Goto** block, which will serve as an output and which will be controlled by the timer. You already know that a new branch of a line is created by right-button dragging, don't you?



## 5.3 Working with I/O signals

### 5.3.1 Modifications in the project main file

Now we tell the compiler to use the WAGO PFC100/200 I/O driver. This is what needs to be done:

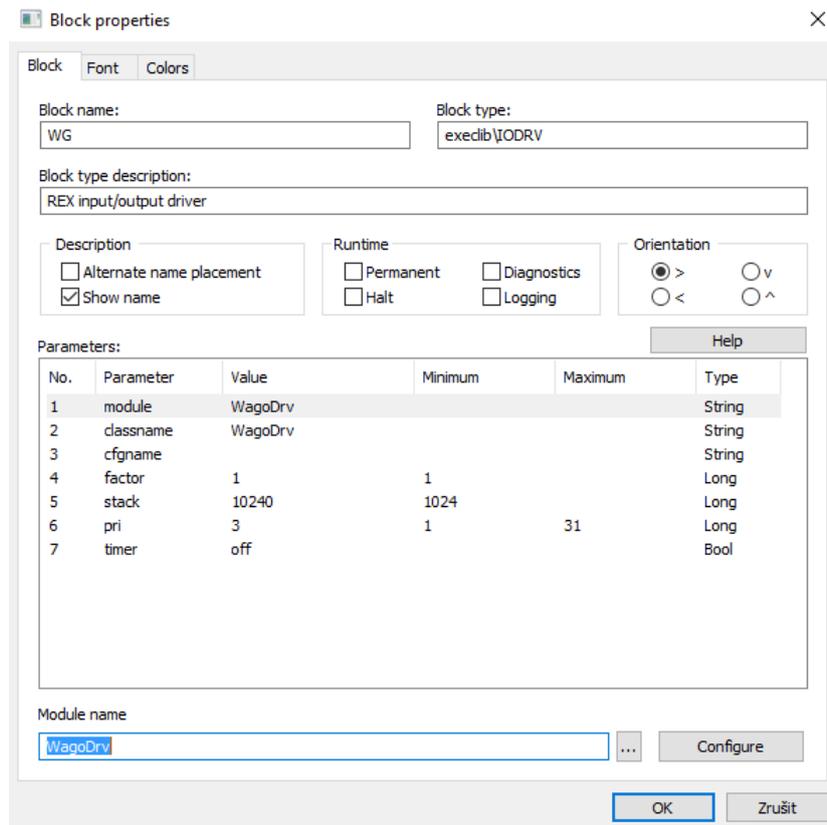
- Rename the **MODULE** block to **WagoDrv** (beware, it's CASE SENSITIVE!)

Next edit the **IODRV** block parameters. Link the **IODRV** block with the **WagoDrv** module by setting

- **module = WagoDrv** (beware, it's CASE SENSITIVE!)

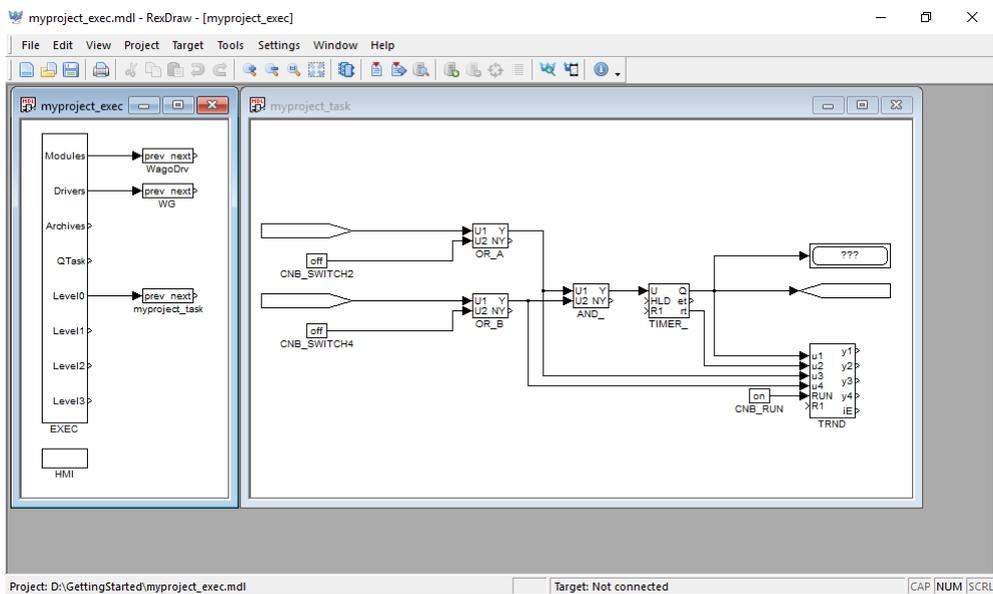
Continue editing the **IODRV** block parameters. For WAGO PFC100/200 set the following:

- `classname = WagoDrv` (beware, it's CASE SENSITIVE!)
- Leave the `cfgname` parameter blank.
- `factor = 1`
- Leave the other parameters intact.



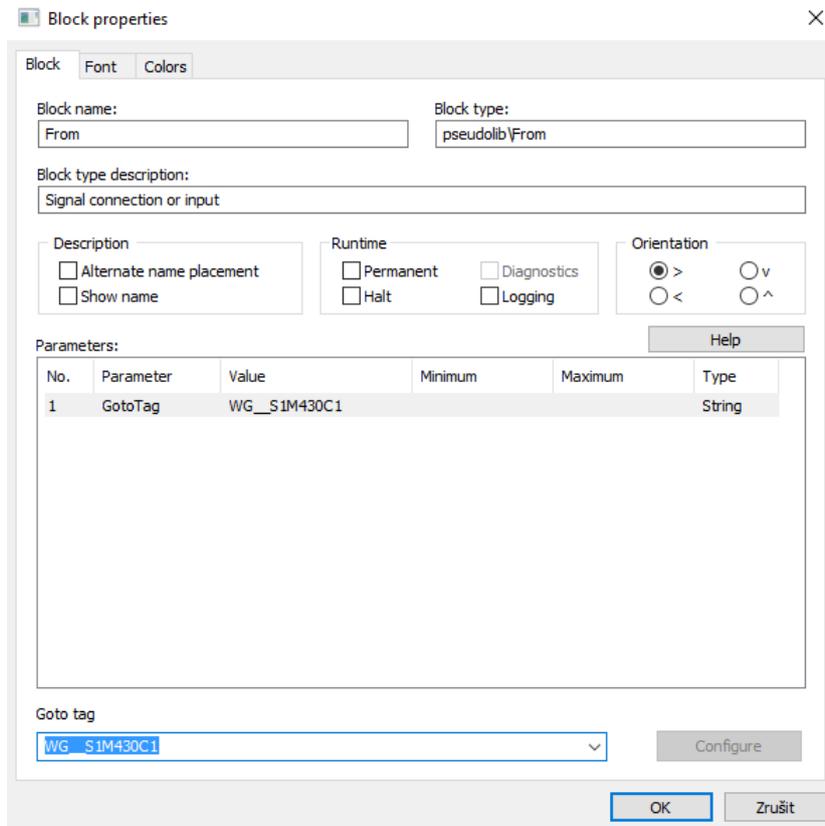
As the final step, rename the IODRV block to WG, which will serve as a prefix for all I/O signals of this driver.

The executive of the REX Control System is configured, your project should look like this:



### 5.3.2 Modifications in the task

In the task double-click the input flag and set `GotoTag = WG__S1M430C1`. Note the WG prefix and **two underscore characters**. The first physical switch (SW1) is connected to channel 1 (DI1) of the 750-430 module, which is in memory slot 1.



The other physical switch (SW3) is connected to channel 3 of the same module (WG\_\_S1M430C3). Channel 1 of the 750-530 module in memory slot 2 (WG\_\_S2M530C1) will serve as control signal for the external relay.

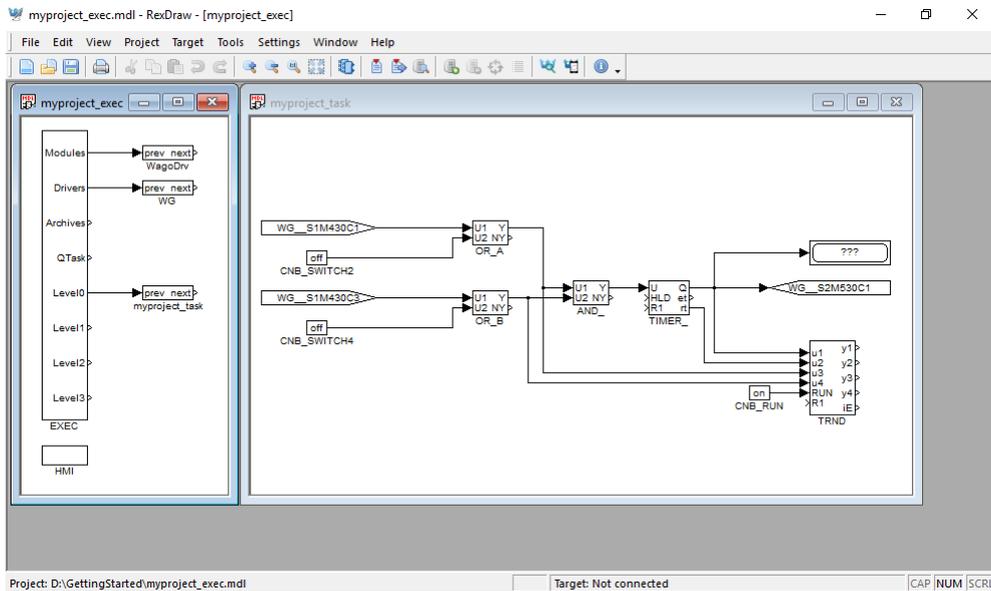
**IMPORTANT:** The potential distribution modules do not provide any data, therefore they do not occupy any memory and do not count when numbering module slots!

Similarly for other I/Os we could use the following flags:

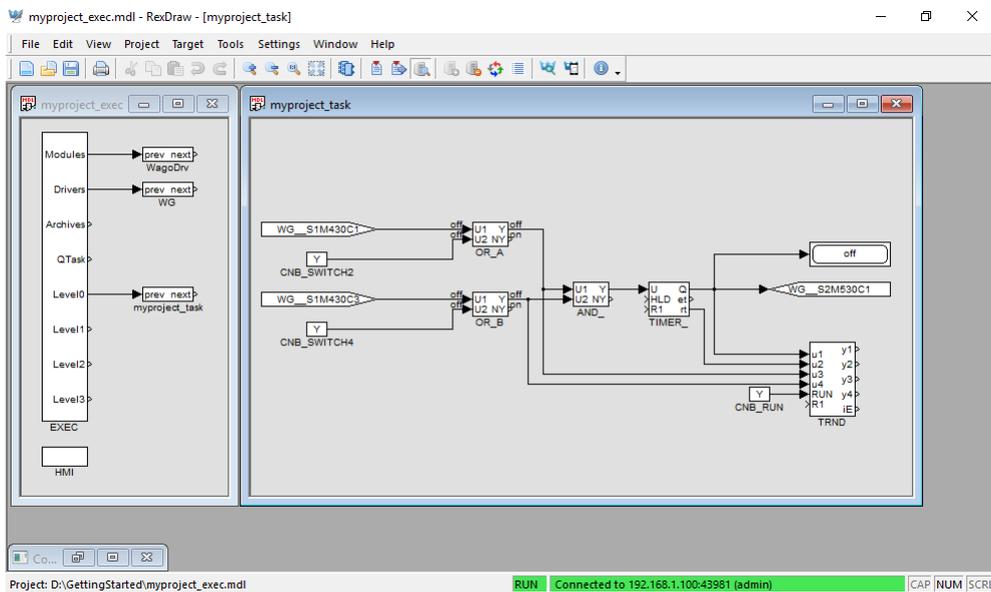
- Goto, WG\_\_S2M530C8 – channel 8 of the 750-530 module in slot 2
- From, WG\_\_S3M555C2 – channel 2 of the 750-555 module in slot 3 (not present in this example)
- etc.

A detailed description of the I/O driver for WAGO PFC100/200 is available in a separate manual [5].

Your project should now look like this:



After compiling the project and downloading it to the WAGO PFC100/200 the control algorithm interacts with the physical world. Again it is possible to switch to Watch mode and observe the signals in real-time or analyze the trends of signals. Flip the physical switches and watch the signals.



## 5.4 Updating the HMI

It is also necessary to update the HMI. The `CNB_SWITCH1` and `CNB_SWITCH3` blocks are no longer present in the algorithm. Moreover, we need to replace virtual input elements (DW) with indicators (DR). Therefore open the `index.hmi.js` file and replace

```
{type: 'DW', alias: 'switch1', desc: 'Switch 1',
 cstring: 'myproject_task.CNB_SWITCH1:YCN'},
```

with

```
{type: 'DR', alias: 'switch1', desc: 'Switch 1',
 cstring: 'myproject_task.OR_A:U1'},
```

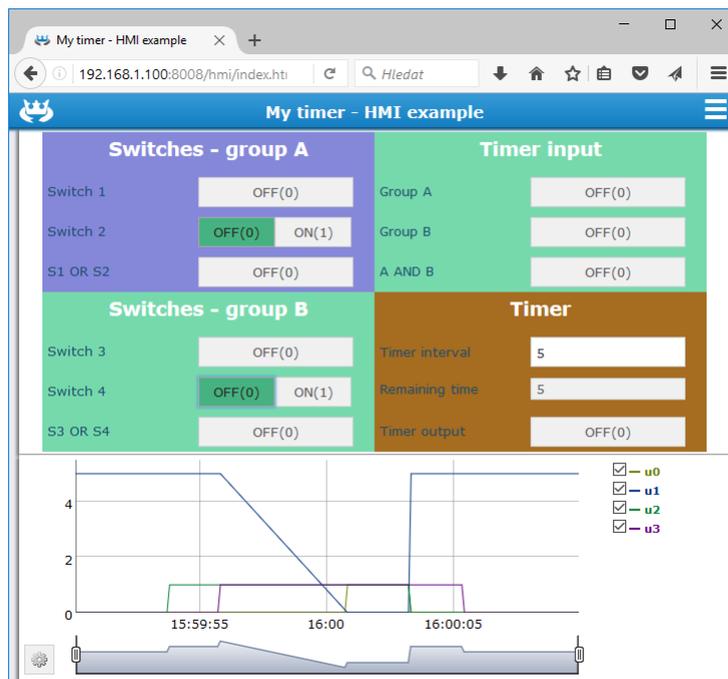
Similarly for switch no. 3, replace

```
{type: 'DW', alias: 'switch3', desc: 'Switch 3',
 cstring: 'myproject_task.CNB_SWITCH3:YCN'},
```

with

```
{type: 'DR', alias: 'switch3', desc: 'Switch 3',
 cstring: 'myproject_task.OR_B:U1'},
```

Save the file, compile and download the project again and open the web interface. Push the two physical switches and wait until the timer triggers the output. Alternatively, you can still use the virtual switches. This demonstrates that you can combine physical and virtual input elements.



## 5.5 Additional information

### 5.5.1 Detailed description of the driver

A detailed description of the IO driver for WAGO PFC100/200 is available in a separate manual [5].

### 5.5.2 Examples

Example projects and a set of all supported I/O flags are included in the installation package of the REX Control System development tools. In **RexDraw**, go to menu **File** → **Start from an Example Project** and select one of the WAGO PFC100/200 examples. As mentioned earlier, the most up-to-date examples are available at <https://github.com/rexcontrols/REXexamples/archive/v2.50.zip>

## Chapter 6

# Summary

Congratulations, you have created the example 0102-01 from scratch! You have learned how to develop, compile and run your algorithms on the WAGO PFC100/200. The interaction with sensors and actuators is provided via input-output driver of the REX Control System, which you have learned to configure and use.

You have achieved quite a lot in a relatively short time, haven't you? The purpose of this guide was to quickly show you the basic steps and tools for developing a project.

Now it's time to focus on your own project and keep learning on the go. There are function blocks which are much more powerful than the ones mentioned in this guide, there are many inspiring example projects, there are additional I/O drivers which you can use to expand the scope of your project, there are many ways to exchange data with external systems and devices, etc.

Remember that whenever you have some achievement to share, we will be happy to hear from you. And whenever you encounter any difficulties, we will be happy to help. You can reach us at [support@rexcontrols.com](mailto:support@rexcontrols.com).

## Appendix A

# Graphical HMI with RexHMI Designer

Chapter 4 describes the creation of a simple WebBuDi user interface. The REX development tools also contain the RexHMI Designer program which is a tool for designing custom graphical visualizations from predefined components<sup>1</sup>. The RexHMI Designer is based on the well-known open-source vector editor Inkscape<sup>TM</sup> <https://inkscape.org/en/>.

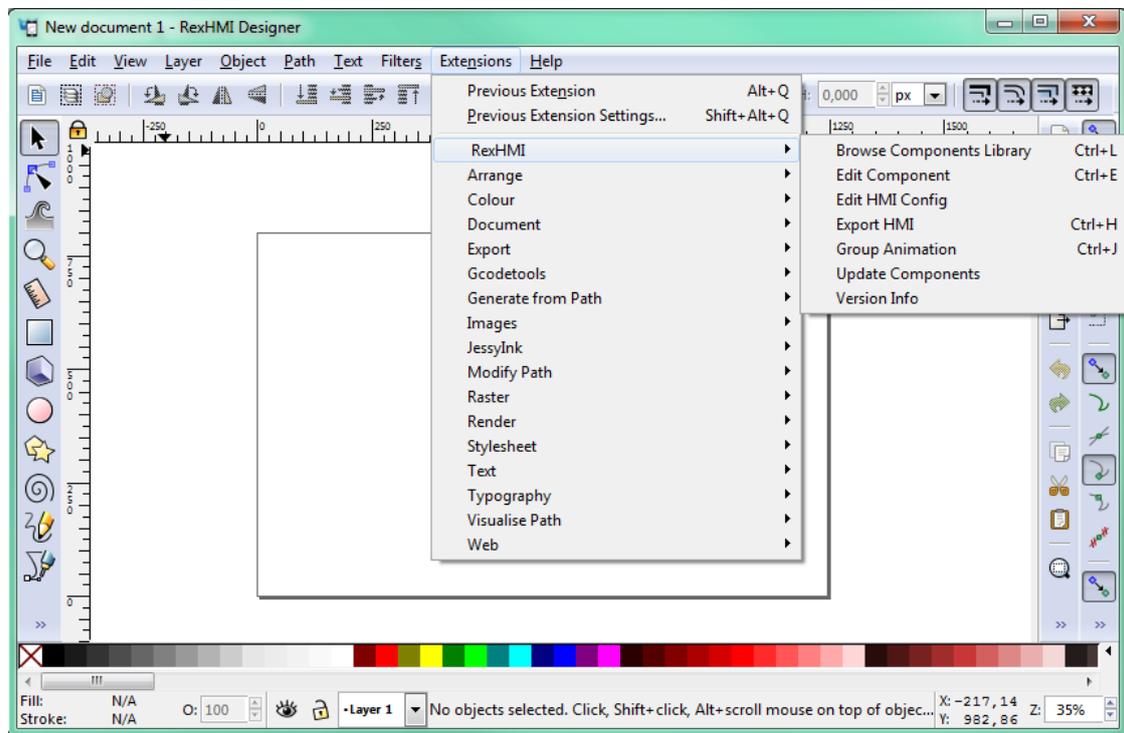
In this chapter we'll develop an alternative HMI for the example 0101-01. Just like in the case of the WebBuDi user interface, the SVG file created with RexHMI Designer will serve as a source file which will become a part of your REX project. During compilation of the project the SVG file will be processed and converted to HTML, JS and CSS files.

### A.1 Initializing the HMI design

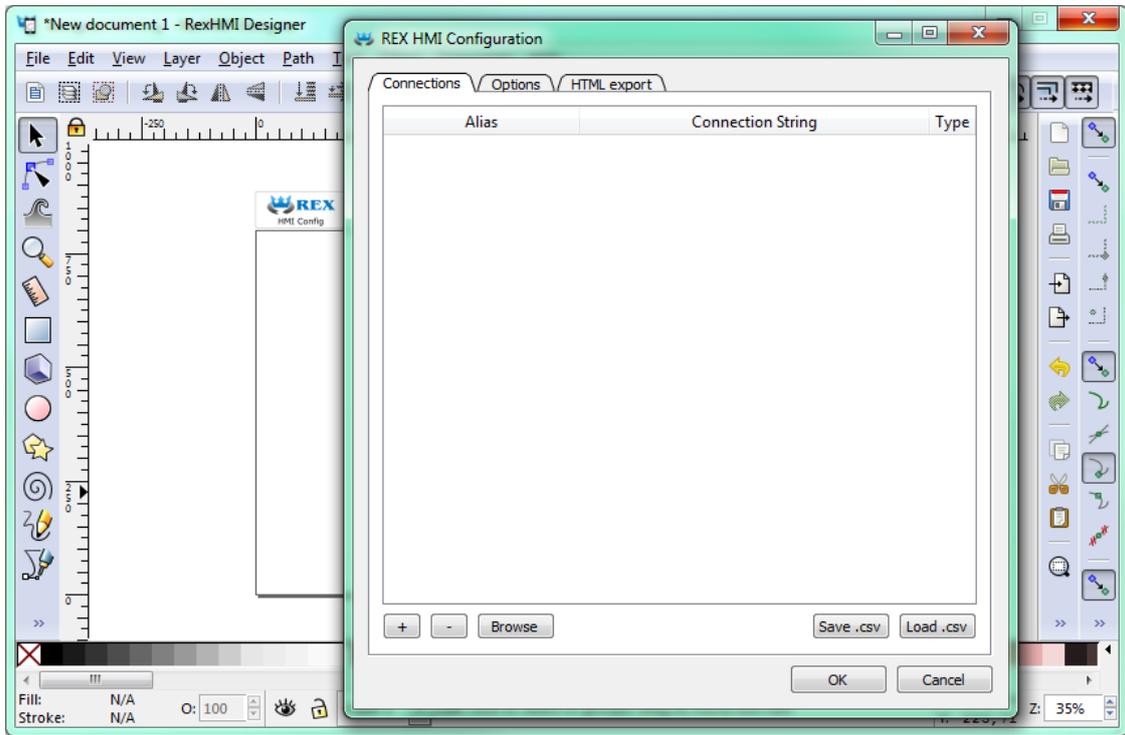
After launching the RexHMI Designer from *Start Menu* you will find a clean page. The first thing to do is to initialize the new visualization. The HMI is configured via the *RexHMI* extensions. Navigate to *Extensions* → *RexHMI* → *Edit HMI Config* in the menu.

---

<sup>1</sup>Definition of custom components is also possible but it requires a bit of Javascript coding.



This extension adds a special component which contains general settings of the HMI. Close it with the OK button for now.

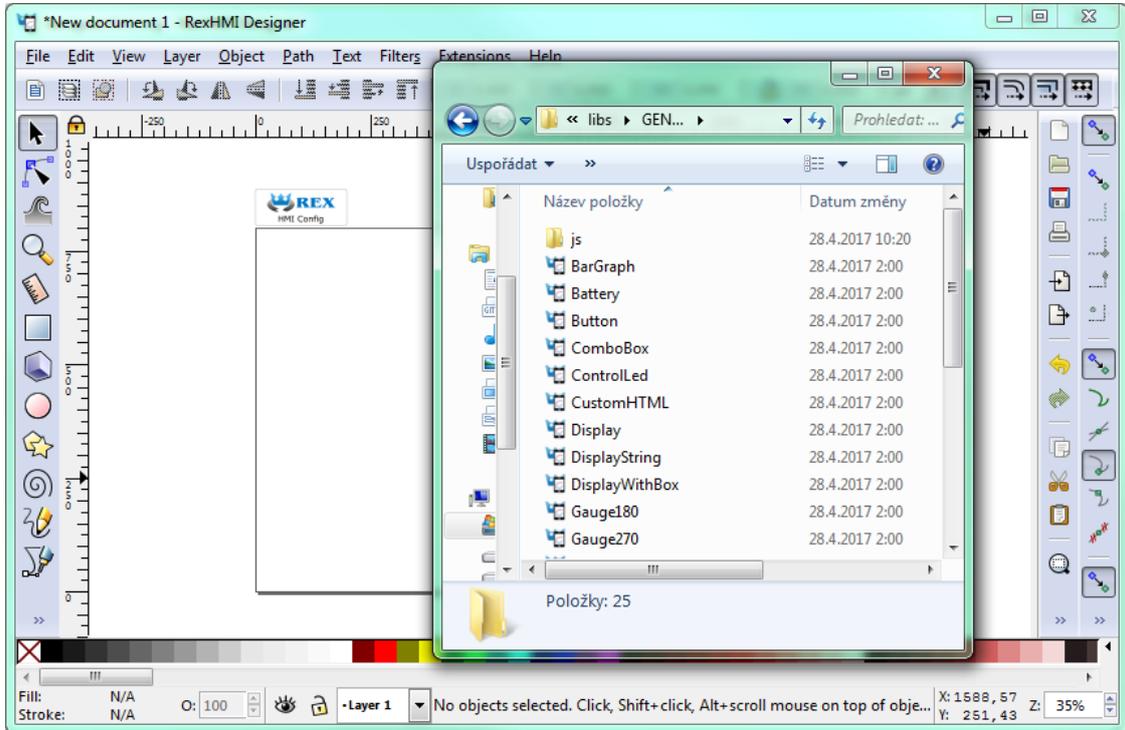


In order to include the HMI during the REX project compilation, the file name has to end with `.hmi.svg`. Save the file as e.g. `designer.hmi.svg` to the `hmisrc` subfolder of your project. Use the standard *File* → *Save as* menu.

*Note: If you want to replace the WebBuDi interface with the RexHMI Designer interface just delete the `index.hmi.js` and save the HMI as `index.hmi.svg`.*

## A.2 Adding the first HMI components

Now we will add some displays and inputs. The RexHMI Designer contains a library of components which you can use to build your HMI. The library is available through *Browse Components Library* extensions. Navigate to *Extensions* → *RexHMI* → *Browse Components Library* (Ctrl+L). It will open the explorer window with several folders. Open the **GENERAL** folder and drag&drop the **Display** to the drawing. The display will be used as an indicator of the remaining time in the **TIMER** function block.



For configuration of the display settings select the display by mouse click on top of it and use the *Edit Component* extension from *Extensions* → *RexHMI* → *Edit Component* (Ctrl+E). When the configuration dialog is opened you can change the *Title* to *Display\_remaining*. The Edit Component dialog has two tabs: *Data points* and *Options*.

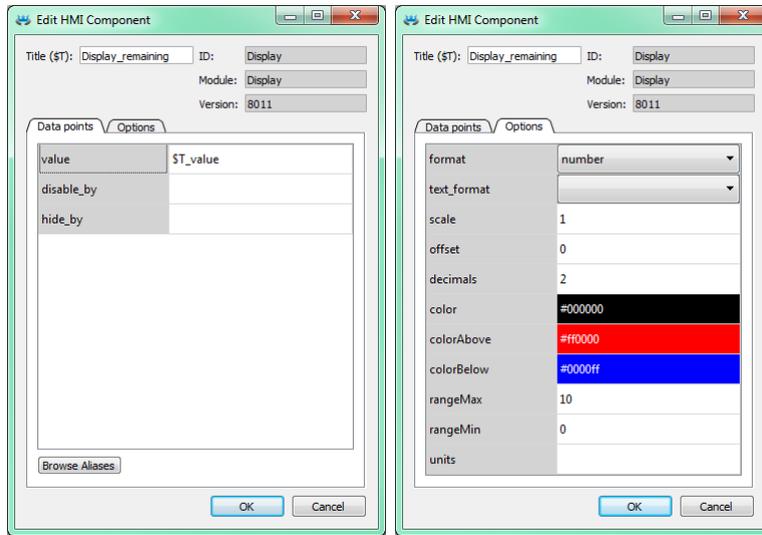
The *Data points* tab contains three items defining the behavior and animations of the component. Each data point contains an alias, which is in fact a connection to live data from the REX algorithm.

- *value* – The value to display.
- *disable\_by* – If true the display is disabled and data are no longer updated.
- *hide\_by* – If true the display is hidden.

The *value* property contains  $\$T\_value$ . The  $\$T$  will be later automatically substituted by the *Title* of the component, resulting in the *Display\_remaining\_value* alias. The *disable\_by* and *hide\_by* data points are optional. Leave them blank at the moment.

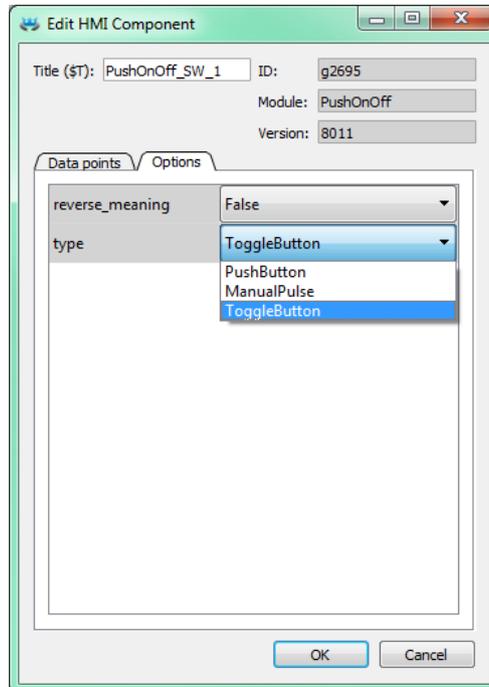
The *Options* tab contains several properties which are specific for the Display component. You can find the description of each component and property in [4]. Leave the default values for now and press OK.

*Note: Each component is in fact one SVG group with unique content. You can copy the components all over the screen using copy (Ctrl+C) and paste (Ctrl+V) approach.*



Now we will add controls for all the switches (CNB blocks). All of them will be controlled using the *PushOnOff* components. Add them from the library. Select the first *PushOnOff* and open the editor dialog *Extensions* → *RexHMI* → *Edit Component* (*Ctrl+E*). Change the title to *PushOnOff\_SW\_1* and select the *Options* tab. Select the *ToggleButton* item in the *type* property list. Close the *Edit Component* dialog using OK button. Copy-paste the button three times and remember to change the titles to *PushOnOff\_SW\_2*, *PushOnOff\_SW\_3*, *PushOnOff\_SW\_4*).

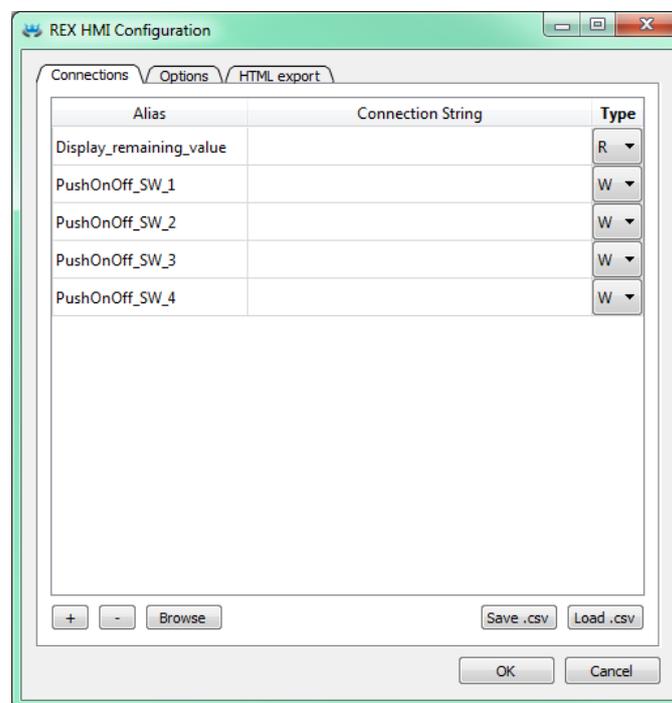
*Note: Throughout the RexHMI Designer you can use **double click** in Options tab to open the corresponding configuration dialog (color picker, number input, etc.).*



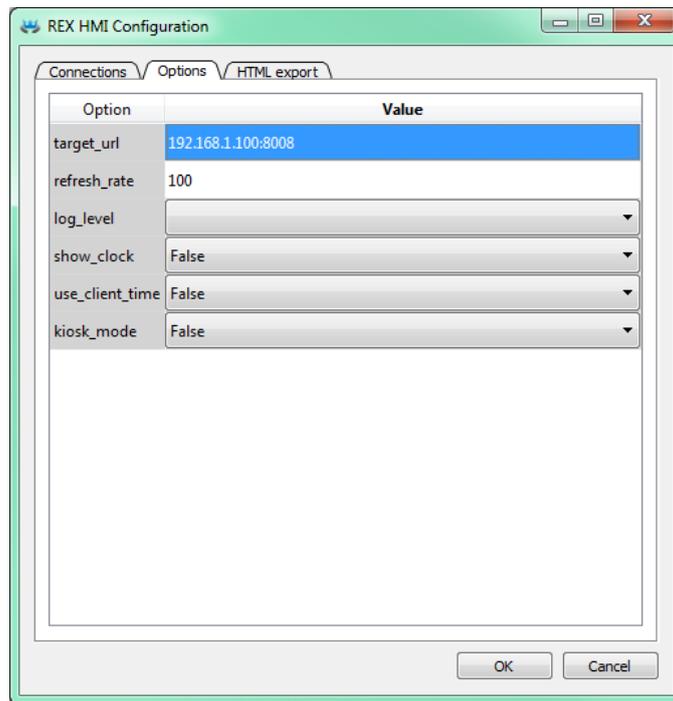
Now we have four *PushOnOff* buttons and one *Display* and we want to link all components with live data from the target device.

### A.3 Linking HMI components with the running algorithm

Open the *HMI Configuration* dialog either using *Extensions* → *RexHMI* → *Edit HMI Config* or just unselect all components in the drawing (click outside any component) and press *Ctrl+E*. The configurator parses all components and creates a list of used *Aliases*. Each of them should be linked with one signal in the running algorithm. You can either fill in each connection string manually or you can use the *Browse* function.

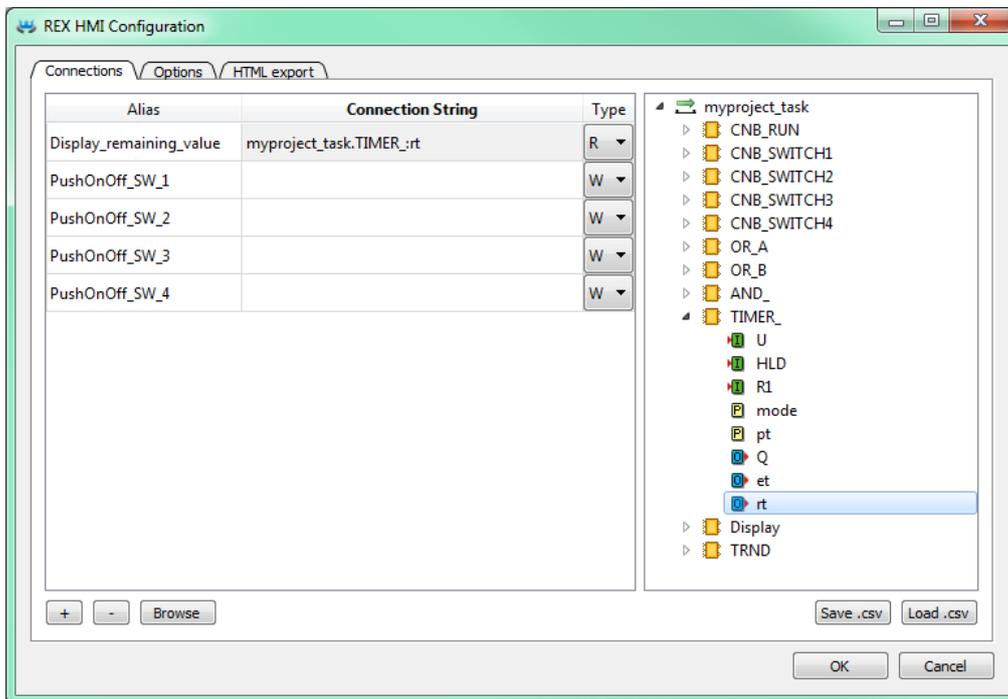


The *Browser* requires a target device with running algorithm. Make sure the algorithm is running, see Section 4.2. Also the target URL must be set. Switch to the *Options* tab and double click on the right side of the `target_url` property. Change it to `192.168.1.100:8008`. Then set the `refresh_rate` property to 100 (default is 500 ms) for more frequent display updates.

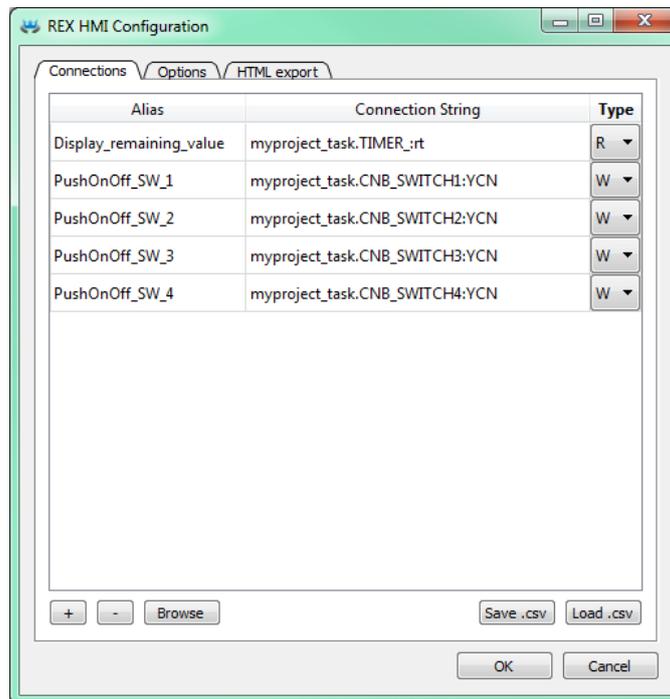


Return back to *Connections* tab and press *Browse* button. The login dialog will be opened. Unless you changed the login credentials, use the default username **admin** with an empty password. After a successful login the connection tab is expanded with a tree-view of the running algorithm (you have already seen this tree-view in algorithm diagnostics).

Select *Connection String* field of the `Display_remaining_value` item and afterwards browse the tree to the `TIMER_` block and **double-click** the `rt` parameter. The connection string of the parameter is copied to the `Display_remaining_value` alias, which is shown in the next figure.

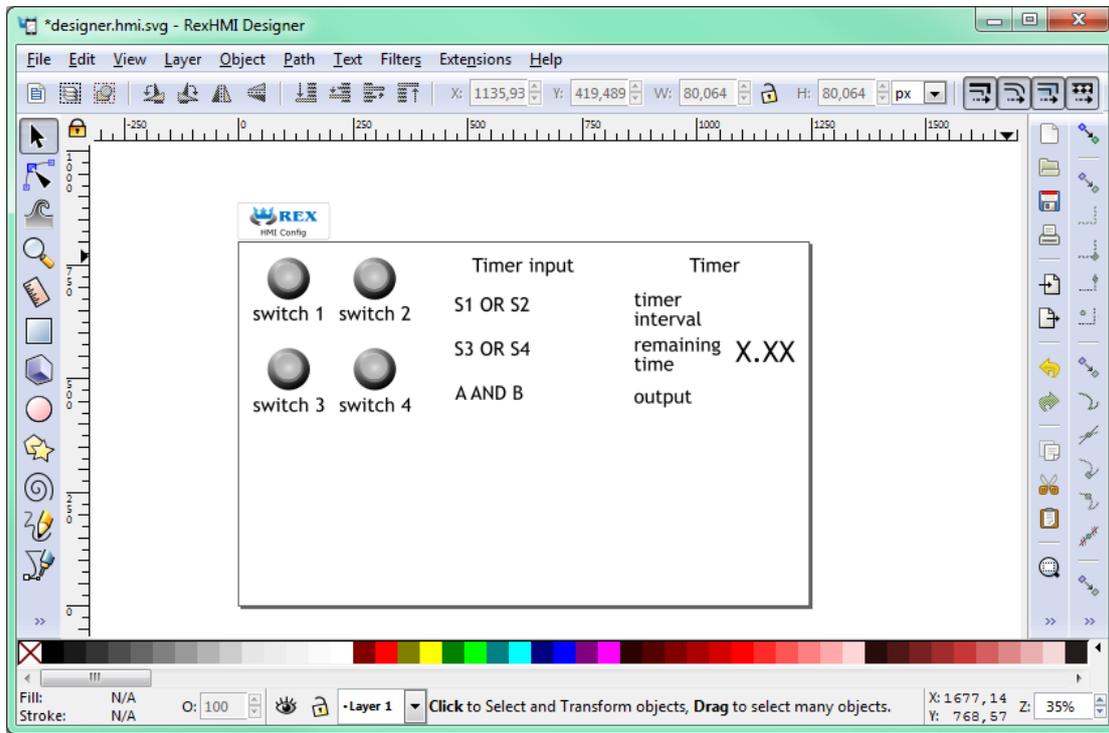


Once the display is linked we will also link the switches. Just browse the tree to CNB\_SWITCH1, select the *Connection String* field of the *PushOnOff\_SW\_1* alias and double-click the YCN parameter. Repeat this for the remaining connection strings. Afterwards press OK to save the settings and close the dialog.



The interactive components in RexHMI Designer are just parts of the drawing. The user can position the components arbitrarily and add as many decorative static components to as needed. We will add some text descriptions to distinguish individual buttons. Use the *Text tool* (F8), click anywhere in an empty space and start typing. Pick the *Select and Transform tool* (F1) afterwards and move the texts and buttons.

*Note: More information about custom drawing can be found in Inkscape tutorials (See the Help → Tutorials → Inkscape: Basic)*

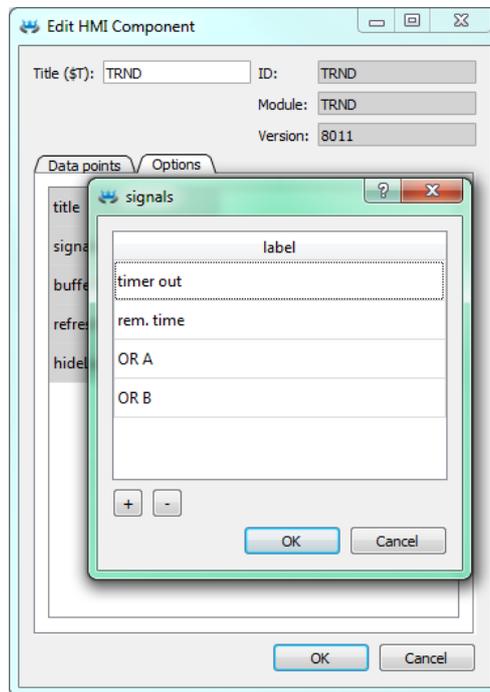


## A.4 Adding more HMI components

Next we will add more components to control the timer and show the status of the OR and AND blocks. Open the Elements library (Ctrl+L) and add one *Input* and four *Led* components. The LEDs will show the status of Boolean values and the *Input* will be used for changing the default timer interval value.

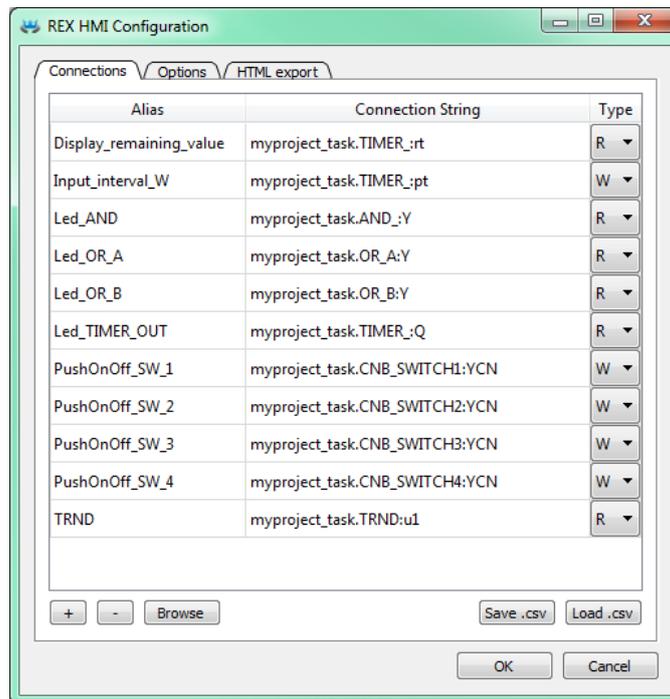
In the *Led* components just edit the *Title* via the *Edit Component* extension (select the component by single click and press Ctrl+E). The titles should be *Led\_OR\_A*, *Led\_OR\_B*, *Led\_AND*, and *Led\_TIMER\_OUT* respectively. Finally edit the *Input* component by changing the *Title* to *Input\_interval*.

The last component we need is a graph showing the time-plot of data from the TRND block. Use the components library (Ctrl+L) and add a *TRND* component. You can adjust its size to fit the desired position. Edit the component (Ctrl+E) and change the *Title* to *TRND* and switch to the *Options* tab. Double-click the *signals* property. Add the following labels using the plus (+) button: *timer out*, *rem. time*, *OR A*, *OR B*. These labels will be shown in the legend of the graph.



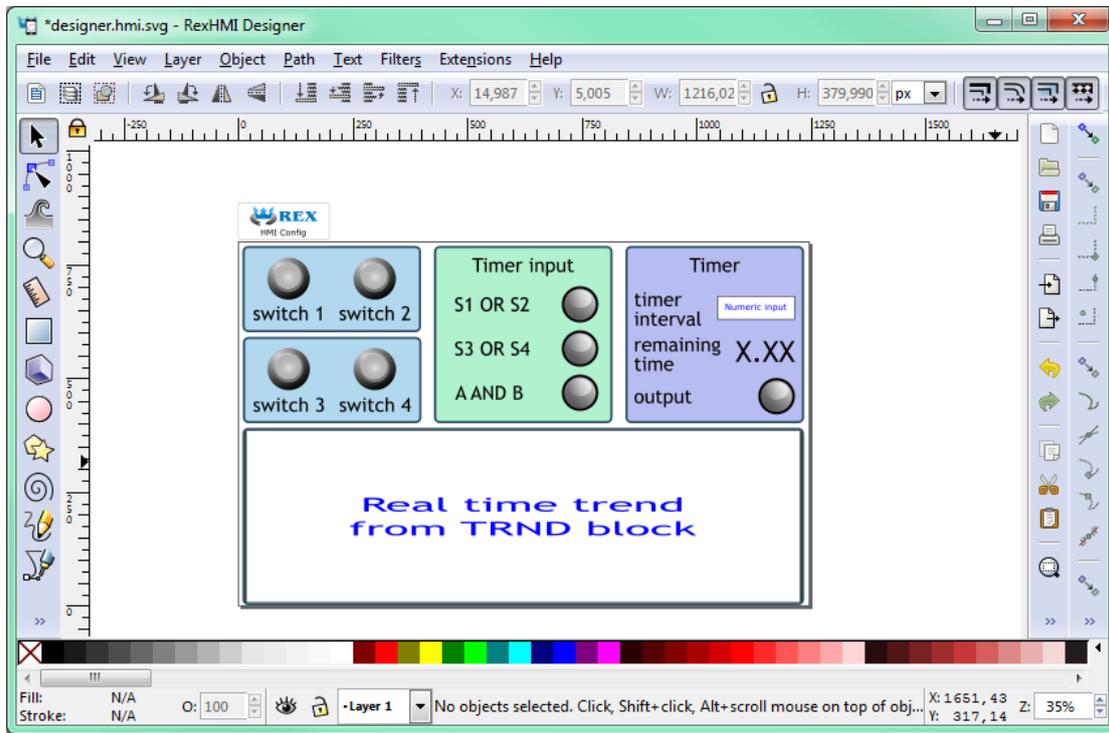
## A.5 Additional links to the running algorithm

When all the components are in place we link them to the running algorithm again. Just repeat the procedure described in Section A.3, open the HMI Configuration dialog (*Extensions* → *RexHMI* → *Edit HMI Config*) and browse the running algorithm to pair the remaining aliases with corresponding connection strings. The list is shown in the following image.



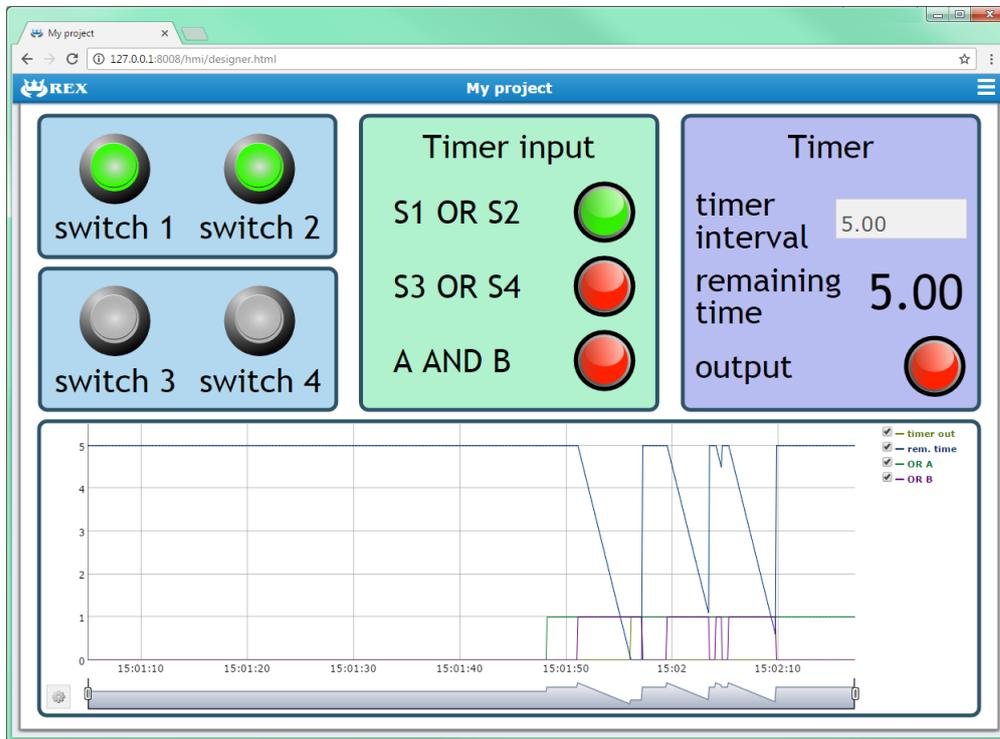
## A.6 The final steps

Congratulations, your first graphical HMI is almost ready! Add a few rectangles which will visually divide the HMI into individual sections. Use the *Squares and rectangles tool* (F4), draw the rectangle, pick a color from the palette and send the rectangle to the background using the End key. Do not forget to save the drawing.



As mentioned earlier, the *\*.hmi.svg* files in the `hmisrc` folder are automatically processed while the project is compiled in the RexDraw application. The project main file must contain the HMI block with `GenerateRexHMI` parameter enabled. This was already covered in Section 4.3 so you should have everything ready.

Once you compile the project again and download it to your WAGO PFC100/200, the HMI will be accessible via a web browser. Navigate to `http://192.168.1.100:8008/hmi/designer.html`. You will see your HMI with live data.



This tutorial covers only the very basic components. If you want to get more information about additional components, see [4].

## Appendix B

# Licensing of the REX Control System

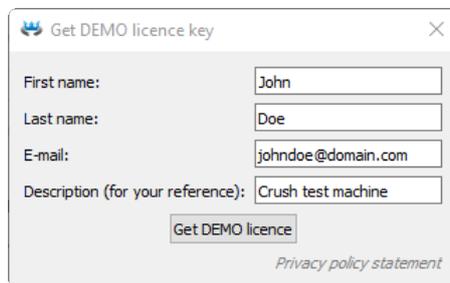
The licensing model of the REX Control System is quite simple:

- The development tools are free to use, you can install it on as many computers as you want.
- The RexCore runtime module always needs a licence to run on your WAGO PFC100/200. There are DEMO licences available at no cost and there are permanent licences which you can purchase. Each WAGO PFC100/200 needs an individual licence.

### B.1 Obtaining a DEMO licence

The DEMO licence is intended for evaluating, testing and educational purposes. Feel free to experiment with the DEMO licence as long as you need. Commercial use of the DEMO licence is not allowed.

When you try to run your algorithm on a device which does not have a licence, you are offered a chance to get a DEMO licence. Identify yourself and you'll receive a DEMO licence via e-mail (the so-called **SiteKey**).



Get DEMO licence key

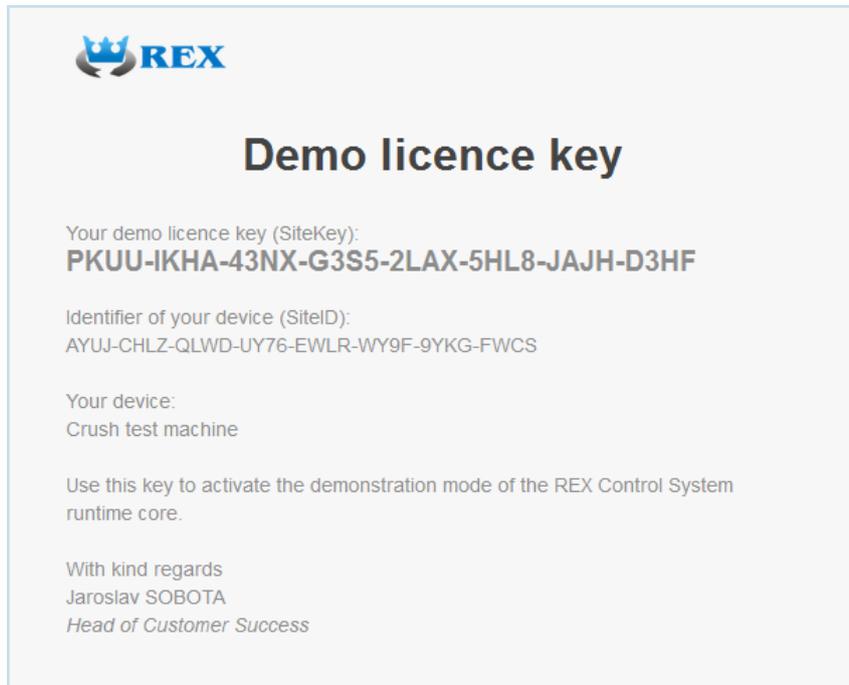
First name:

Last name:

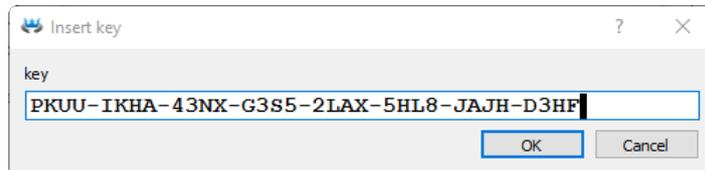
E-mail:

Description (for your reference):

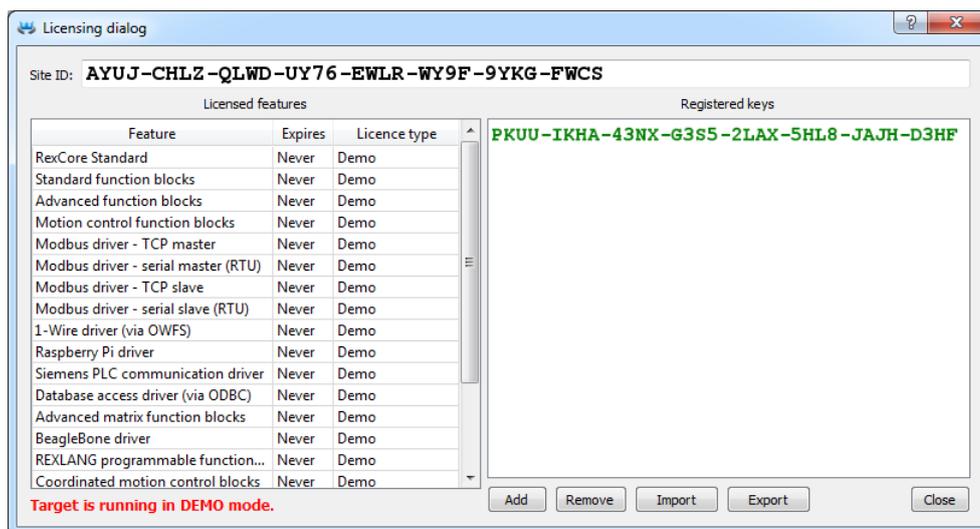
[Privacy policy statement](#)



Try downloading your algorithm once again and apply the DEMO licence (SiteKey).



Once applied, you will see all the modules.



Now you can run your algorithms on your WAGO PFC100/200.

Evaluation version of the RexCore runtime core is functional for 2 hours. It is possible to run your algorithm on the WAGO PFC100/200 but you cannot store it permanently. The algorithm resides only in the memory. You can use almost all function blocks, see [1]. The RexCore runtime core on the target device is terminated after 2 hours of operation in demo mode without any warning. After a restart, you have another 2 hours for your experiments.

## B.2 Obtaining a permanent licence

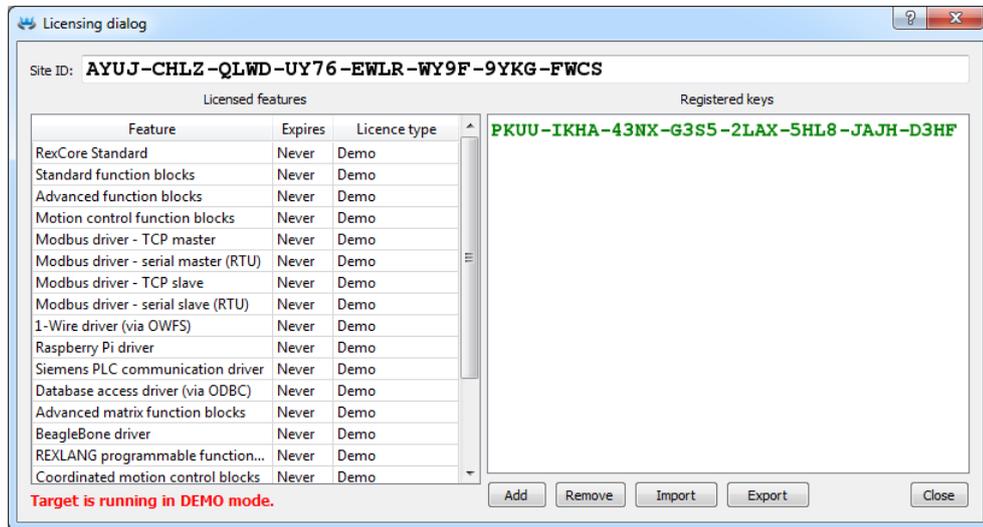
It is necessary to activate the RexCore runtime module and optional additional modules for permanent operation. This can be done using the licence, which you can obtain at

[www.rexcontrols.com/e-shop](http://www.rexcontrols.com/e-shop)

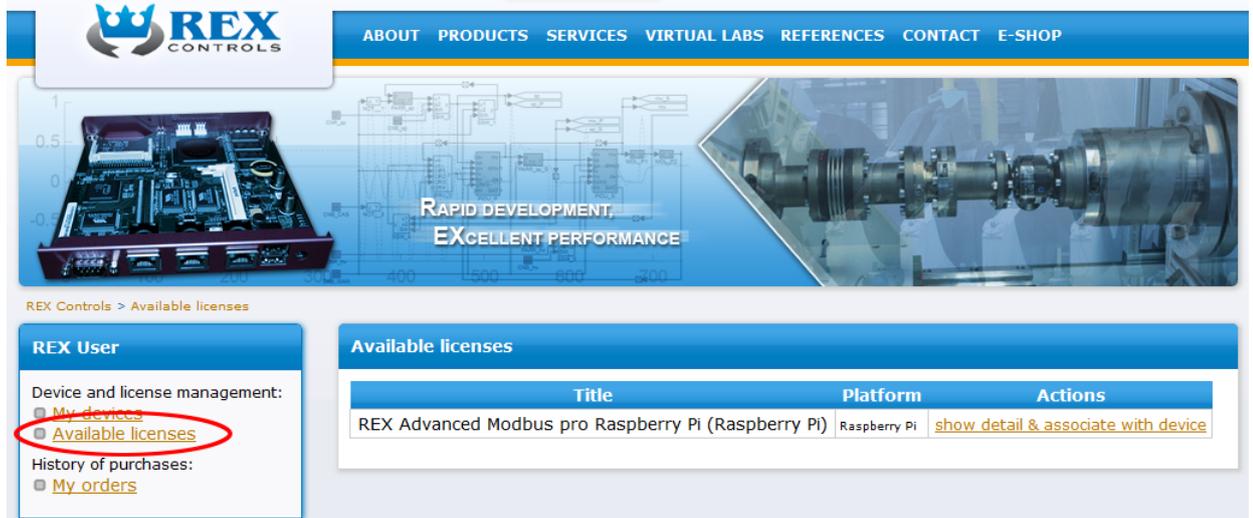
### B.2.1 Activation of the permanent licence

Each device running the RexCore runtime module is identified by the so-called SiteID tag. The purchased licence must be associated with the hardware device, i.e. with the SiteID tag.

1. You can get the SiteID identifier in RexDraw. Connect to the device and go to menu *Target* → *Licensing...*. A dialog pops up and you can copy the SiteID identifier.



2. Afterwards, login to [www.rexcontrols.com](http://www.rexcontrols.com) using your username and password and list the available licences which you possess.



- Use the SiteID identifier to associate the licence with the hardware device.

**Status** Available  
This license can be associated to device.  
Association can be done in the form below.

[BACK](#)

**Associate license**

1. Select device → 2. Summary and confirmation → 3. Associate license to device

Select device to associate with this license:

Create new device

**Title** Solar Panels Monitoring  
Title for your new device.

**SiteCode** 8ABH-F2C4-PAAA-BT42-6PAE-ZL2H-7ZNZ-WK7K  
SiteCode identifier of your device.

**Serial number**  
Serial number of your device (optional).

Central Heating Raspberry Pi

Greenhouse Control WinPAC

**Associate selected device to license**

- You are asked to confirm the association – this is the last and irreversible step.

|               |                                                                                                             |
|---------------|-------------------------------------------------------------------------------------------------------------|
| <b>Status</b> | <b>Available</b><br>This license can be associated to device.<br>Association can be done in the form below. |
|---------------|-------------------------------------------------------------------------------------------------------------|

BACK

**Associate license**

1. Select device → 2. **Summary and confirmation** → 3. Associate license to device

|                                                                    |
|--------------------------------------------------------------------|
| <b>License</b>                                                     |
| REX Advanced Modbus for Raspberry Pi (Raspberry Pi)                |
| +                                                                  |
| <b>Device</b>                                                      |
| Solar Panels Monitoring<br>8ABH-F2C4-PAAA-BT42-6PAE-ZL2H-7ZNZ-WK7K |

**This is the final and irreversible step!**

The license **REX Advanced Modbus for Raspberry Pi (Raspberry Pi)** will be associated with a new device **Solar Panels Monitoring (Raspberry Pi)** with SiteCode 8ABH-F2C4-PAAA-BT42-6PAE-ZL2H-7ZNZ-WK7K. The license is valid only for the given hardware configuration including storage media (hard drive, SD card etc.). However, you might be allowed to reuse the license in well justified cases.

**Associate licence with the selected device (irreversible)**

5. The so-called SiteKey activation key is generated upon associating the licence.

License was successfully associated to device!

**Device "Solar Panels Monitoring (Raspberry Pi)"**

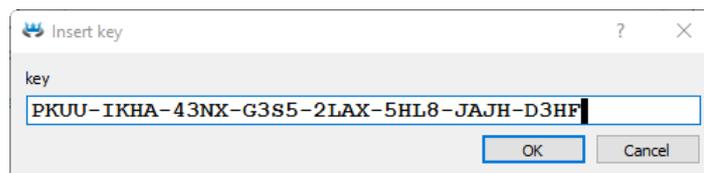
|                      |                                         |                      |
|----------------------|-----------------------------------------|----------------------|
| <b>Title</b>         | Solar Panels Monitoring                 | <a href="#">Edit</a> |
| <b>Serial number</b> |                                         | <a href="#">Edit</a> |
| <b>SiteCode</b>      | 8ABH-F2C4-PAAA-BT42-6PAE-ZL2H-7ZNZ-WK7K |                      |
| <b>Platform</b>      | Raspberry Pi                            |                      |
| <b>Created</b>       | 2014-04-09                              |                      |
| <b>Text</b>          |                                         | <a href="#">Edit</a> |

[BACK](#)

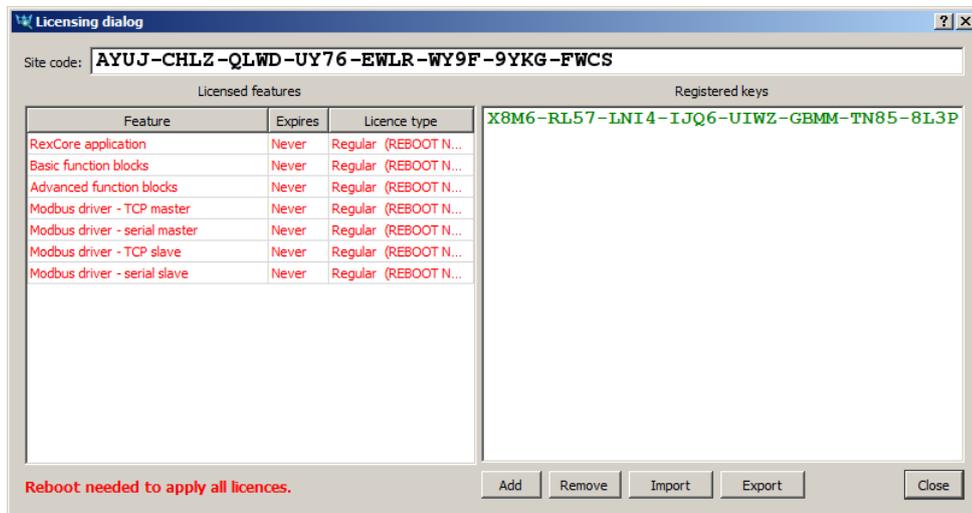
**Associated licenses**

|                             |                                                                                                                                                                                                               |                      |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| <b>Title</b>                | REX Advanced Modbus for Raspberry Pi                                                                                                                                                                          |                      |
| <b>SiteKey</b>              | X8M6-RL57-LNI4-IJQ6-UIWZ-GBMM-TN85-8L3P                                                                                                                                                                       |                      |
| <b>Platform</b>             | Raspberry Pi                                                                                                                                                                                                  |                      |
| <b>Created</b>              | 2014-04-09                                                                                                                                                                                                    |                      |
| <b>Associated to device</b> | 2014-04-09 14:00:25                                                                                                                                                                                           |                      |
| <b>Text</b>                 | -                                                                                                                                                                                                             | <a href="#">Edit</a> |
| <b>License</b>              | Product: REX Advanced Modbus for Raspberry Pi<br>Platform: Raspberry Pi<br>Modules: RexCore, Basic function blocks, Advanced blocks, Modbus TCP slave, Modbus TCP master, Modbus RTU master, Modbus RTU slave |                      |
| <b>Product</b>              | Product in catalogue: <a href="#">REX Advanced Modbus for Raspberry Pi</a>                                                                                                                                    |                      |
| <b>Order</b>                | -                                                                                                                                                                                                             |                      |
| <b>Status</b>               | <b>Associated</b><br>This license is associated to device.                                                                                                                                                    |                      |

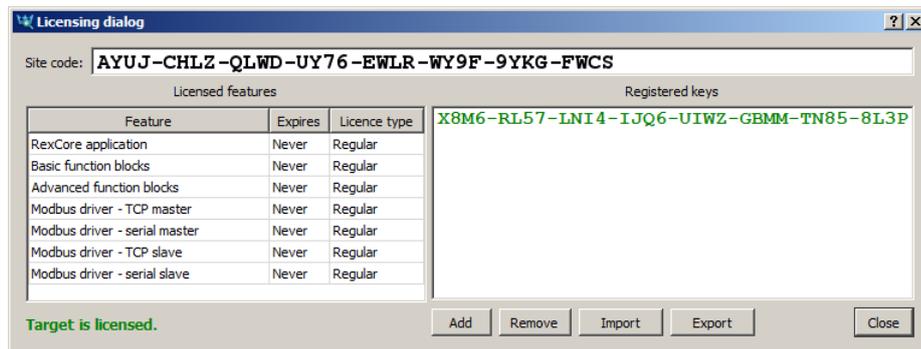
6. This key will allow permanent operation of the runtime core, but it must be applied to the target device. To do that, open the licensing dialog in RexDraw again.
7. **Delete any DEMO licence keys if present.**
8. Afterwards apply the SiteKey using the Add button.



9. If the key is valid, the activated modules appear. The WAGO PFC100/200 must be restarted afterwards.



- After reboot, check that the licence was applied correctly and that the RexCore runtime module no longer runs in demo mode.



# Bibliography

- [1] REX Controls s.r.o.. *Function blocks of the REX Control System – reference manual*, 2017.
- [2] REX Controls s.r.o.. *RexDraw – User manual*, 2017.
- [3] REX Controls s.r.o.. *RexCore – User manual*, 2017.
- [4] REX Controls s.r.o.. *RexHMI – User manual*, 2017.
- [5] REX Controls s.r.o.. *WagoDrv driver of the REX Control System for Wago PFC100/PFC200 – user guide*, 2017.