www.rexcontrols.com/rex

Getting started with REX and Monarco HAT

User guide

REX Controls s.r.o.

Version 2.50.1
2016-11-07
Plzeň (Pilsen), Czech Republic

Contents

Chapter 1

Introduction

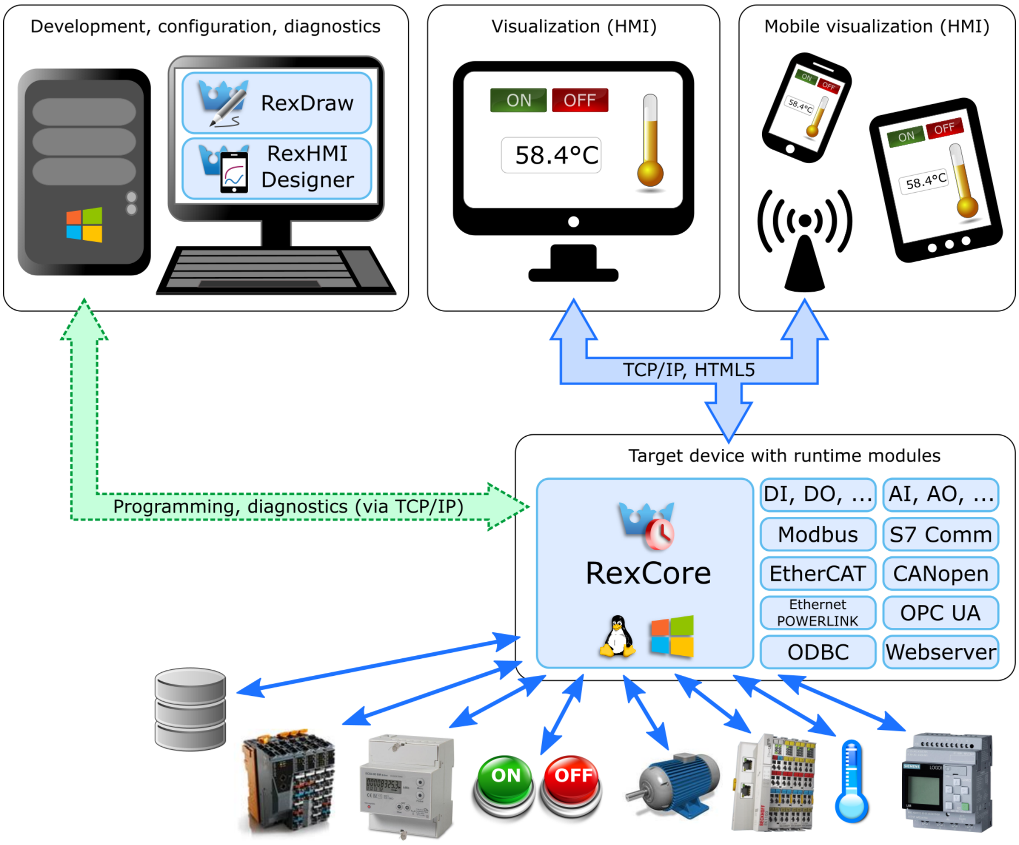
The REX Control System is a family of software products for automation projects. You can use it in all fields of automation, robotics, measurements and feedback control.

The runtime core of the REX Control System turns your Raspberry Pi into a programmable device which will run your algorithms.

1.1 Features of the REX Control System

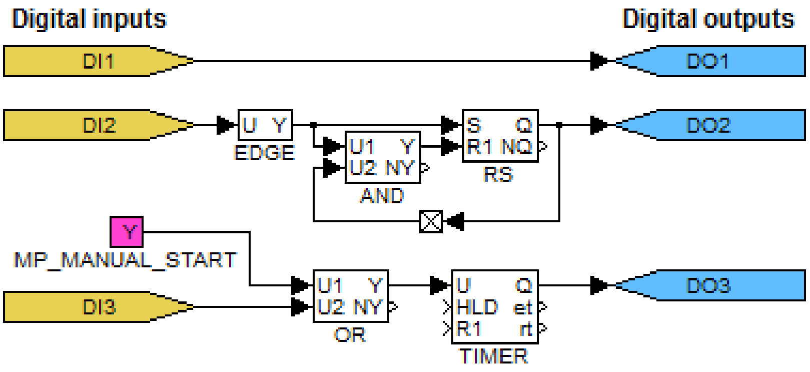
- Graphical programming without hand-coding
- Programming control units on a standard PC or laptop
- User interface for desktop, tablet and smartphone (HMI)
- Wide family of supported devices and input-output units (including Monarco HAT)
- Industry-proven control algorithms
- Easy integration into business 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. You create the algorithms using the function block library[1] 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 can compile and run your algorithms on your Raspberry Pi.

Once running, you can watch your algorithm in real-time. Just select the signals and function blocks of your interest. 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 Raspberry Pi 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 called 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 (Raspberry Pi). 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 important tool for commissioning and diagnostics of control algorithms. You obtain detailed hierarchical information about the running control algorithm. You can connect via local network or over the Internet.

[1] The 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 Vista/7/8/10 operating systems.

2.1 Windows Vista/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 Control System on Raspberry Pi

It is necessary to have a correctly configured Internet connection on your Raspberry Pi running the so-called Raspbian distribution of GNU/Linux[1] prior to using this guide. Visit www.raspberrypi.org for more information.

3.1 Installation of components on Raspberry Pi

1. On your Raspberry Pi, install GIT
   `sudo apt install git`

2. Go to your home directory
   `cd ~`

3. Download the latest revision of installation scripts
   `git clone https://github.com/rexcontrols/rex-install-rpi.git`

4. Change the working directory
   `cd rex-install-rpi`

5. Depending on your hardware, run **ONLY ONE** of the following installation scripts

   - For bare Raspberry Pi:
     `sudo bash install-rex.sh`
   - For Raspberry Pi with the Monarco HAT:
     `sudo bash install-rex-monarcohat.sh`
   - For Raspberry Pi with UniPi extension board:
     `sudo bash install-rex-unipi.sh`

[1] RexCore should also work on any other distribution based on Debian (e.g. Ubuntu).

- For Raspberry Pi with PiFace Digital extension board:
  `sudo bash install-rex-pifacedigital.sh`

6. Perform reboot if you are asked for it and you are done.

Right after the installation the RexCore runtime module is started automatically in the background as a daemon and it is possible to establish connection between the host PC and the Raspberry Pi using the RexDraw and/or RexView programs. RexCore also automatically runs upon system (re)start.

3.2 Installed files and folders

Please refer to the RexCore user guide [2] for detailed information about the files and folders of the REX Control System runtime modules.

3.3 Uninstall procedure

Use the `sudo apt-get remove rexcore` command to remove RexCore from the operating system.

Chapter 4

Configuration, compilation and execution

The process of creating the control algorithm will be demonstrated on a very simple example with two Boolean variables representing manual switches, which will be later replaced by physical inputs of the Monarco HAT. A timer will be used for measuring the time when both variables are true (i.e. both switches are in the ON position). A Boolean signal will be used for signalling 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).

Standard approach:

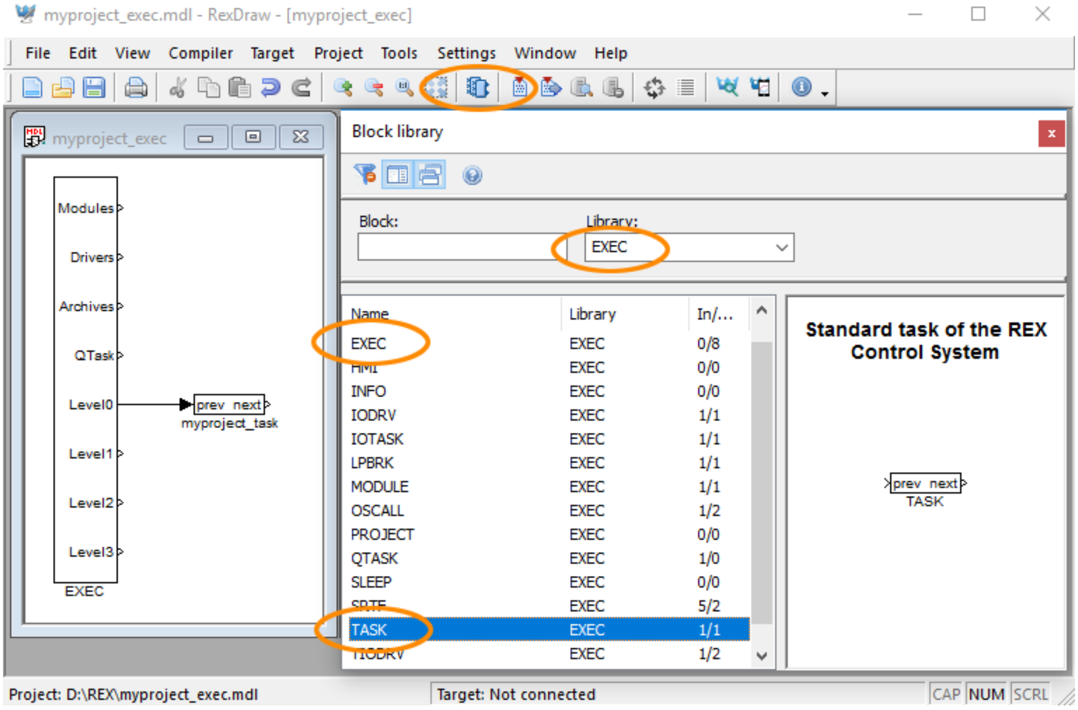
1. Run the RexDraw program, start with an empty project and save the new file as e.g. `myproject_exec.mdl`.

2. Open `Block Library`, choose *View/Block Library* in the menu or use the icon from the toolbar.

3. Drag&drop the main block called `EXEC` and one `TASK` block to the `myproject_exec.mdl` file. Both of them are contained in the `EXEC` library. Further the location of blocks within the libraries will be denoted as `library/block`, e.g. `EXEC/TASK`.

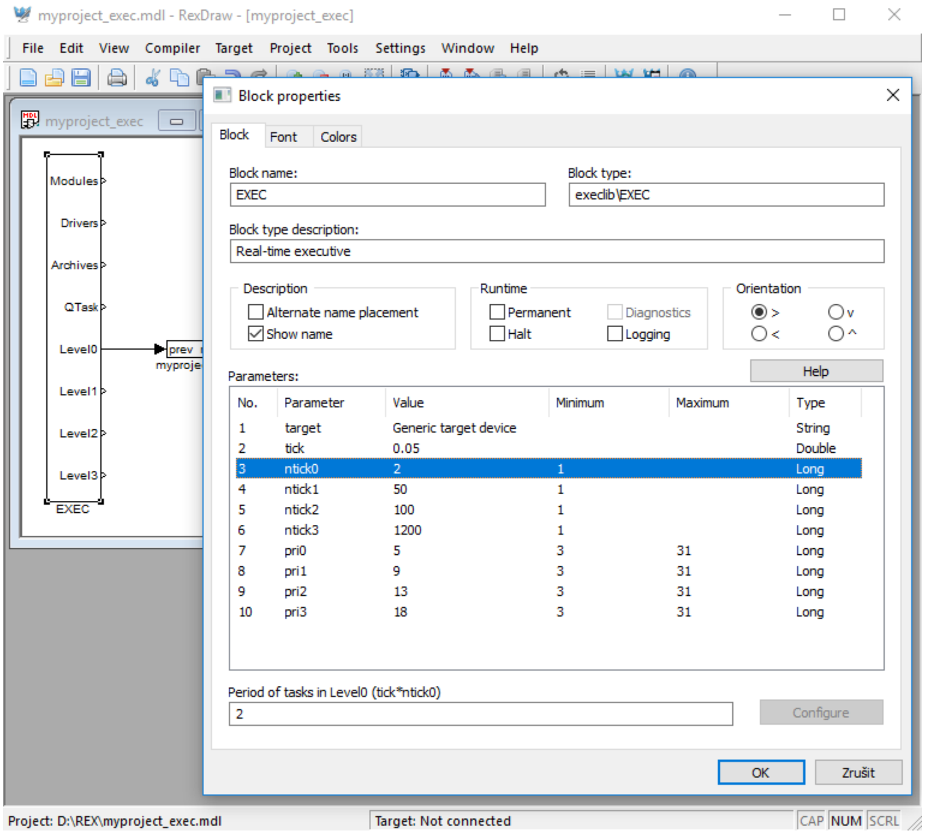
4. Now connect the two blocks, namely the `Level0` output of the `EXEC` block and the `prev` input of the `TASK` block. To connect the blocks, drag the output arrow to the input arrow using the left mouse button. The connection will be established when the line goes bold and green. After releasing the 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.


myproject_exec.mdl - RexDraw - [myproject_exec]
File Edit View Compiler Target Project Tools Settings Window Help
myproject_exec
Modules
Drivers
Archives
QTask
Level0
Level1
Level2
Level3
EXEC
prev next
myproject_task
Block library
Block:
Library:
EXEC
Standard task of the REX Control System
prev next
TASK
Project: D:\REX\myproject_exec.mdl
Target: Not connected
CAP NUM SCRL

Name	Library	In/...
EXEC	EXEC	0/8
HMI	EXEC	0/0
INFO	EXEC	0/0
IODRV	EXEC	1/1
IOTASK	EXEC	1/1
LPBRK	EXEC	1/1
MODULE	EXEC	1/1
OSCALL	EXEC	1/2
PROJECT	EXEC	0/0
QTASK	EXEC	1/0
SLEEP	EXEC	0/0
SRTF	EXEC	5/2
TASK	EXEC	1/1
TIODRV	EXEC	1/2

5. Each 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].



Block name: EXEC

Block type: execlib\EXEC

Block type description: Real-time executive

Description
□ Alternate name placement
☑ Show name

Runtime
□ Permanent □ Diagnostics
□ Halt □ Logging

Orientation
● > ○ v
○ < ○ ^

Parameters:

No.	Parameter	Value	Minimum	Maximum	Type
1	target	Generic target device			String
2	tick	0.05			Double
3	ntick0	2	1		Long
4	ntick1	50	1		Long
5	ntick2	100	1		Long
6	ntick3	1200	1		Long
7	pri0	5	3	31	Long
8	pri1	9	3	31	Long
9	pri2	13	3	31	Long
10	pri3	18	3	31	Long

Period of tasks in Level0 (tick*ntick0)

2

Project: D:\REX\myproject_exec.mdl   Target: Not connected   CAP NUM SCRL

6. Set `ntick0` $= 2$ in the block parameters dialog of the `EXEC` block, i.e. the task connected to the `Level0` output will be executed each `tick*ntick0`$=$0.1 seconds.

7. Rename the `TASK` block to `myproject_task` (the `Block name` field in the block parameters dialog). By this, we define the file associated with the `TASK` block. The `myproject_task.mdl` file will be created later.

8. Save the `myproject_exec.mdl` file. At this moment it is necessary to define the main file of the project. Pick *File/Set as main* in the menu. The window title changes to `myproject_exec - RexDraw [myproject_exec]`.

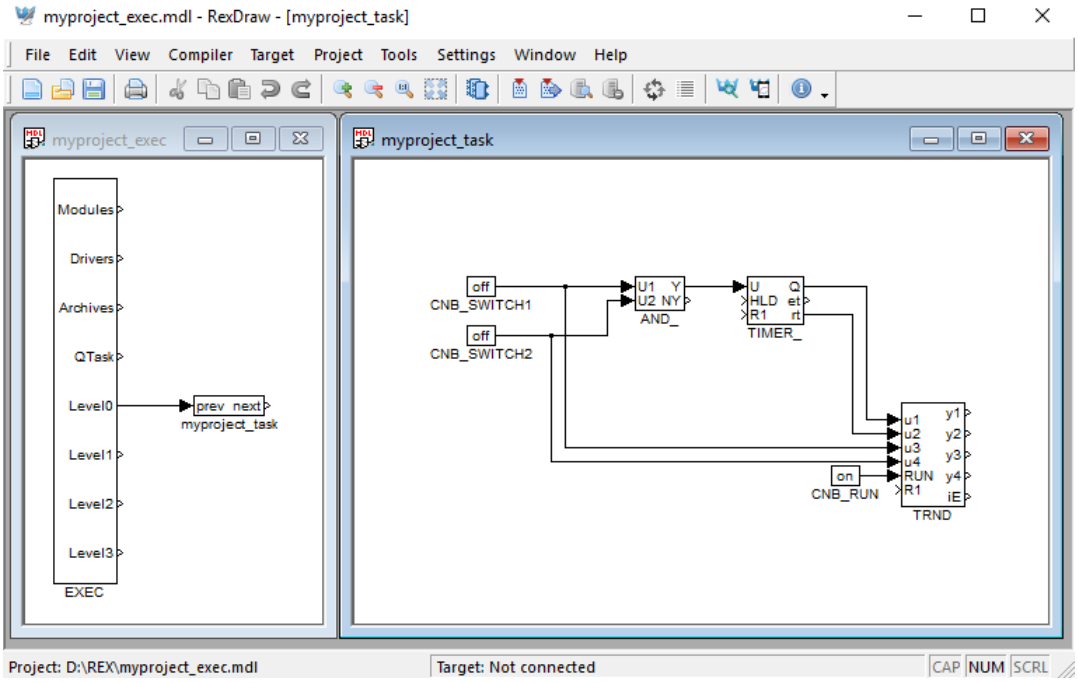
9. Open a new file and save it as `myproject_task`. Use the following function blocks from `Block Library`:

   - `MATH/CNB` – constant of type `Boolean`, change name to `CNB_SWITCH1`, set parameter `YCN` $=$ `off`,
   - `MATH/CNB` – constant of type `Boolean`, change name to `CNB_SWITCH2`, set parameter `YCN` $=$ `off`,
   - `LOGIC/AND_` – logical AND,
   - `MATH/TIMER_` – a timer block, set parameter `mode` $= 2 : \mathit{DelayedON}$, `pt` $= 5$,

- ARC/TRND – real-time recording, set parameters l = 500, Title = My timer, leave the default values otherwise
- MATH/CNB – constant of type Boolean, change name to CNB_RUN, set parameter YCN = on,

Connect the blocks as shown below.

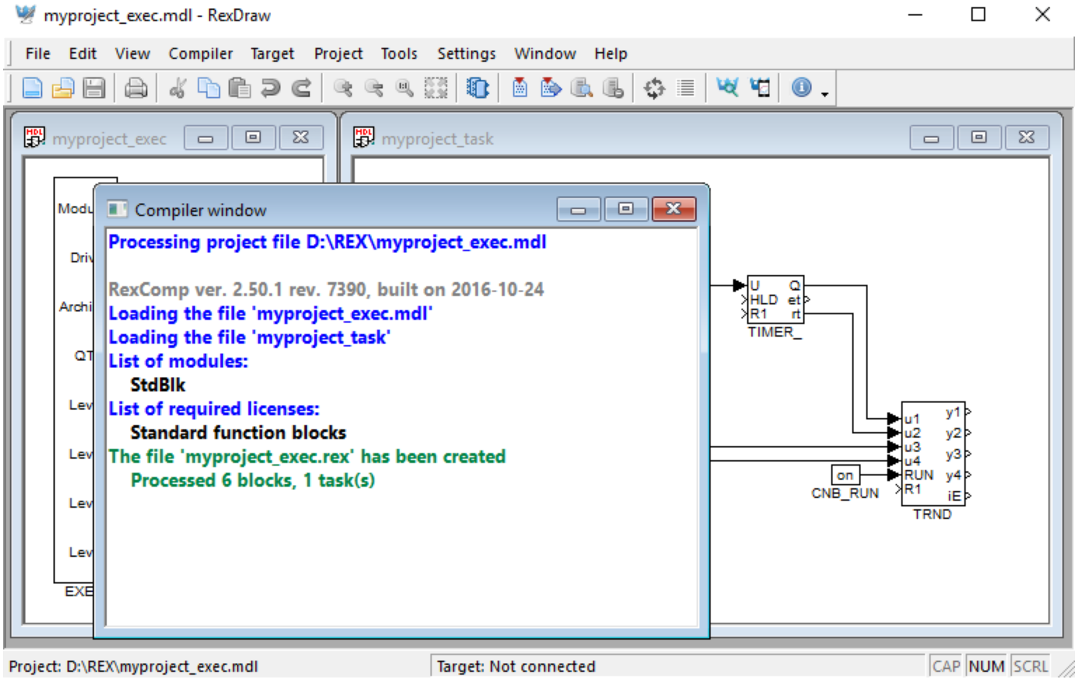
New branch of a line can be created by right mouse button dragging.



At this moment the executive myproject_exec.mdl and the corresponding file with the algorithm myprojec_task.mdl are ready for compilation.

4.2 Compiling and running a project

The developed algorithm must be compiled to binary form prior to deploying. Pick *Compiler/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.

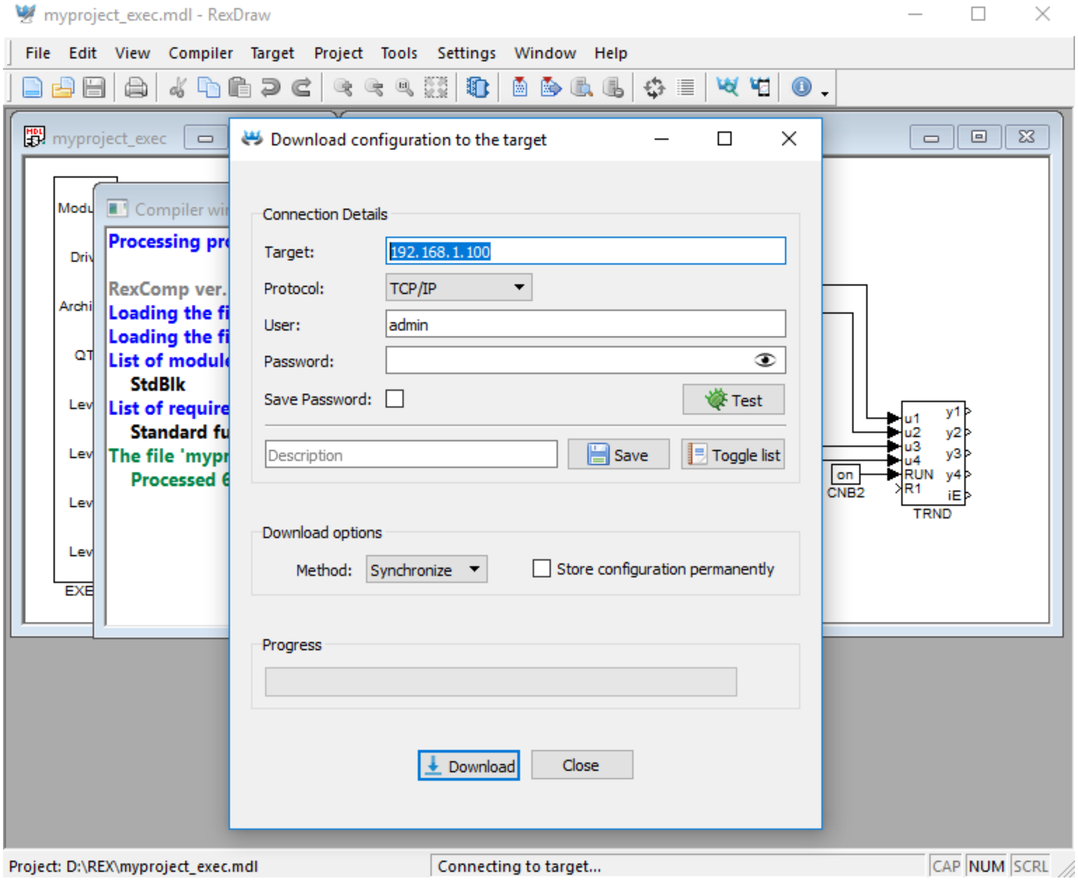


```
Processing project file D:\REX\myproject_exec.mdl

RexComp ver. 2.50.1 rev. 7390, built on 2016-10-24
Loading the file 'myproject_exec.mdl'
Loading the file 'myproject_task'
List of modules:
   StdBlk
List of required licenses:
   Standard function blocks
The file 'myproject_exec.rex' has been created
   Processed 6 blocks, 1 task(s)
```

Project: D:\REX\myproject_exec.mdl

Target: Not connected

At this moment it is possible to deploy the control algorithm to the target platform. Use the `Compile and Download` icon for this purpose. A dialog for defining the target device appears upon successful compilation.


myproject_exec.mdl - RexDraw

Download configuration to the target

Connection Details

Target: 192.168.1.100

Protocol: TCP/IP

User: admin

Password:

Save Password: □

Download options

Method: Synchronize

□ Store configuration permanently

Progress

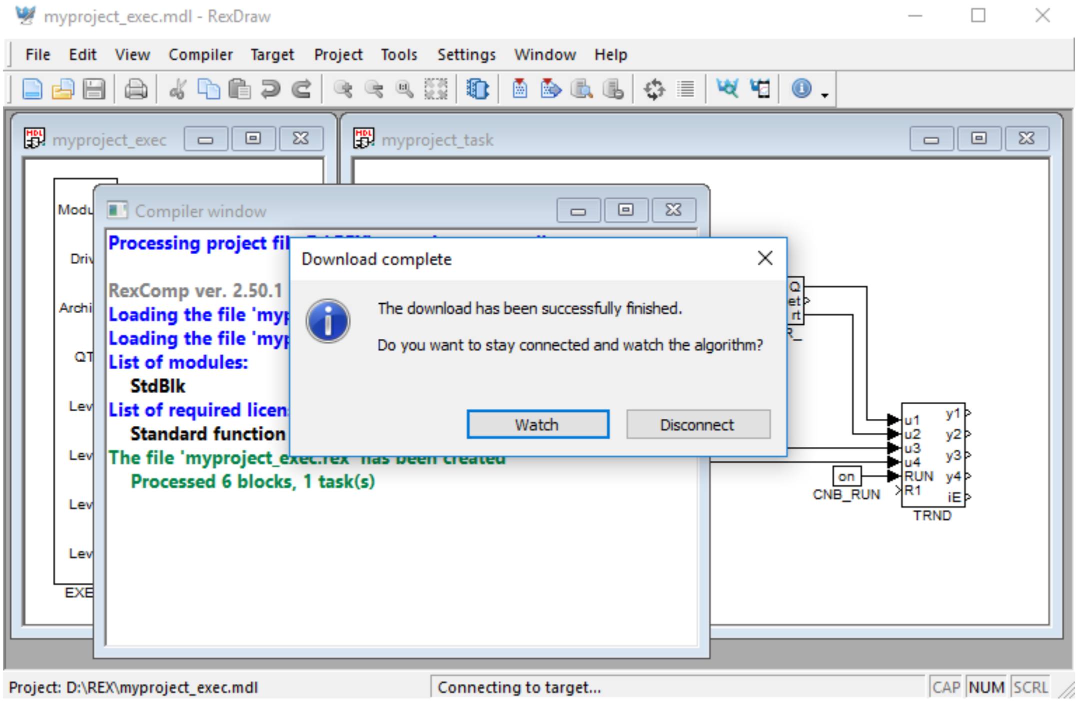
Project: D:\REX\myproject_exec.mdl

Connecting to target...

Enter the IP address of the Raspberry Pi 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 Raspberry Pi, you need to get one first. See chapter 6 for details and come back afterwards.

As soon as the download is complete it is possible to connect to the target device and watch the control algorithm in action – click `Watch`.



```
Processing project fil
RexComp ver. 2.50.1
Loading the file 'myp
Loading the file 'myp
List of modules:
   StdBlk
List of required licen
   Standard function
The file 'myproject_exec.rex' has been created
   Processed 6 blocks, 1 task(s)
```

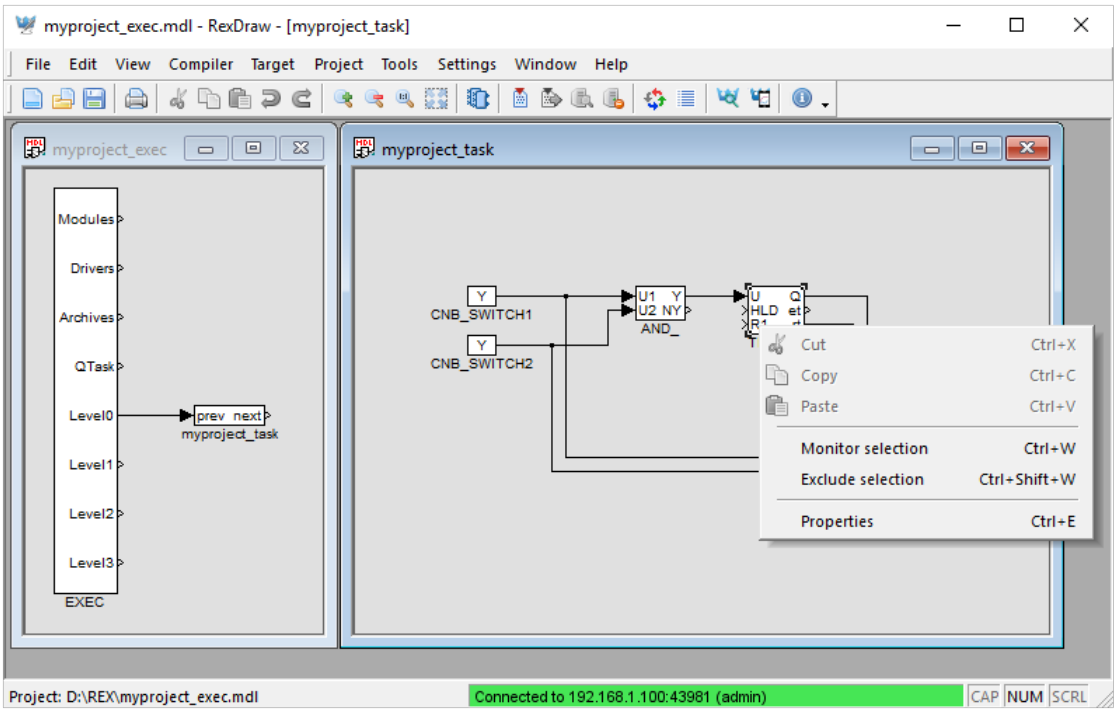
Download complete

The download has been successfully finished.

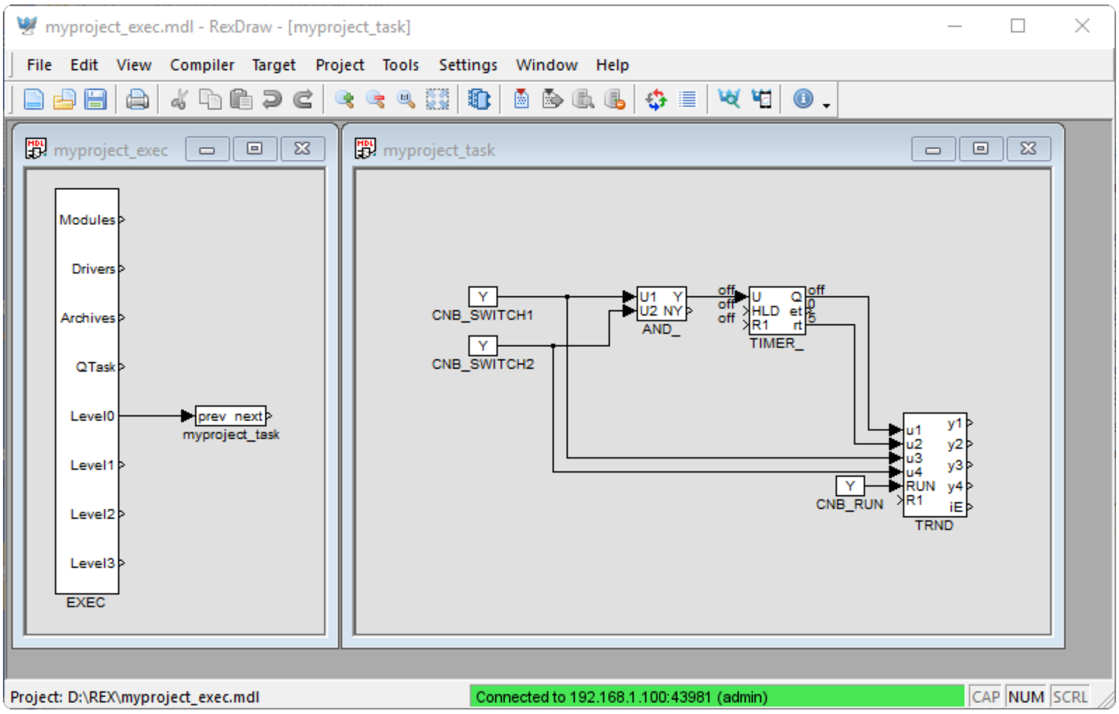
Do you want to stay connected and watch the algorithm?

Project: D:\REX\myproject_exec.mdl | Connecting to target...

The RexDraw program now works in the so-called online mode. Right-click the TIMER_ block and select Monitor selection in the menu to watch the inputs and outputs of the timer.



Project: D:\REX\myproject_exec.mdl | Connected to 192.168.1.100:43981 (admin)



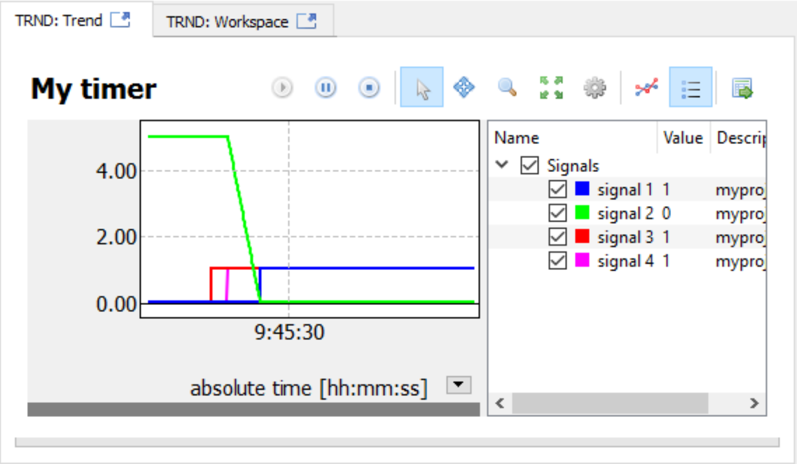
Project: D:\REX\myproject_exec.mdl

Connected to 192.168.1.100:43981 (admin)

You can do the same with the CNB 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_SWITCH2 block, the AND_ block output Y goes on and the TIMER_ starts to count down (observe the rt output). Once the timer reaches zero, its output Q is set to on and it remains on as long as the U input is on (both the switches are on).

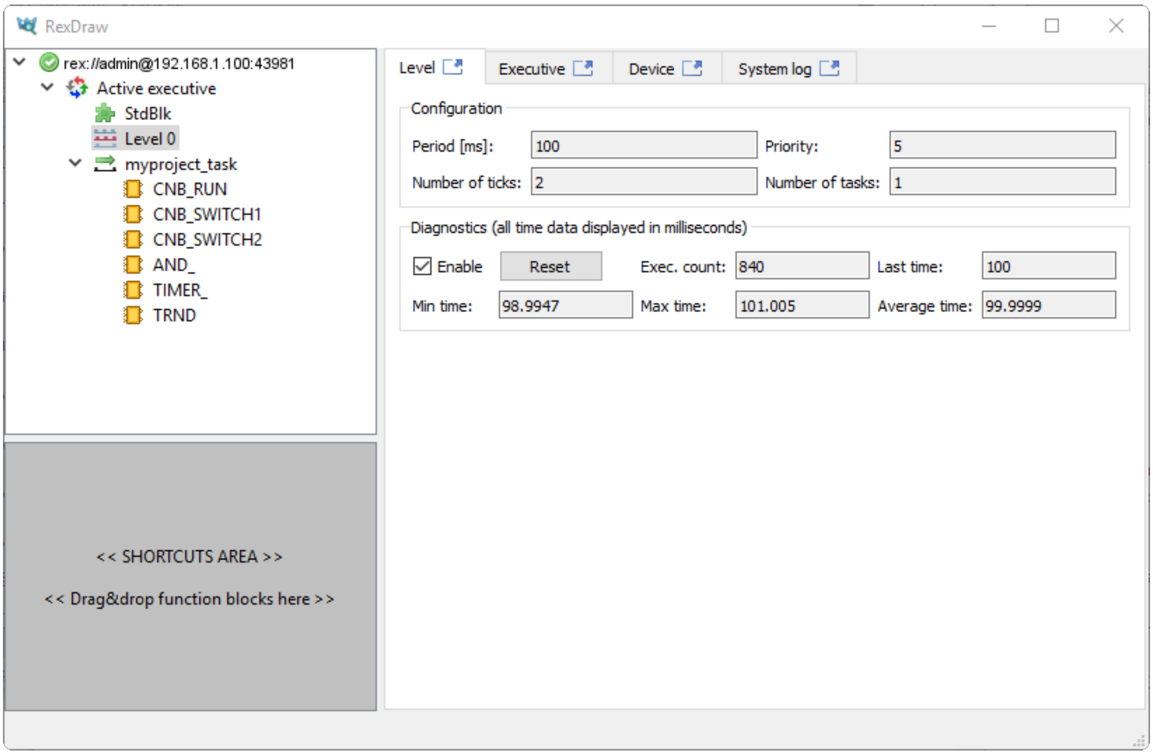
You can double-click the TRND block to see the signals in a real-time graph. Red line is the first switch, magenta line is the second switch, green line is the remaining time of the timer and blue line is the output of the timer.



Name	Value	Descrip
☑ Signals		
☑ 🟦 signal 1	1	mypro
☑ 🟩 signal 2	0	mypro
☑ 🟥 signal 3	1	mypro
☑ 🟪 signal 4	1	mypro

Try turning the `CNB` blocks `off`, change the `pt` parameter of the `TIMER_` block, engage the switches 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. Click the and you will see the algorithm in a tree view which allows you to monitor the control algorithm in full detail. You can also adjust parameters of individual function blocks, which has the same effect as modifying them directly in the *Block properties* dialog.



Configuration

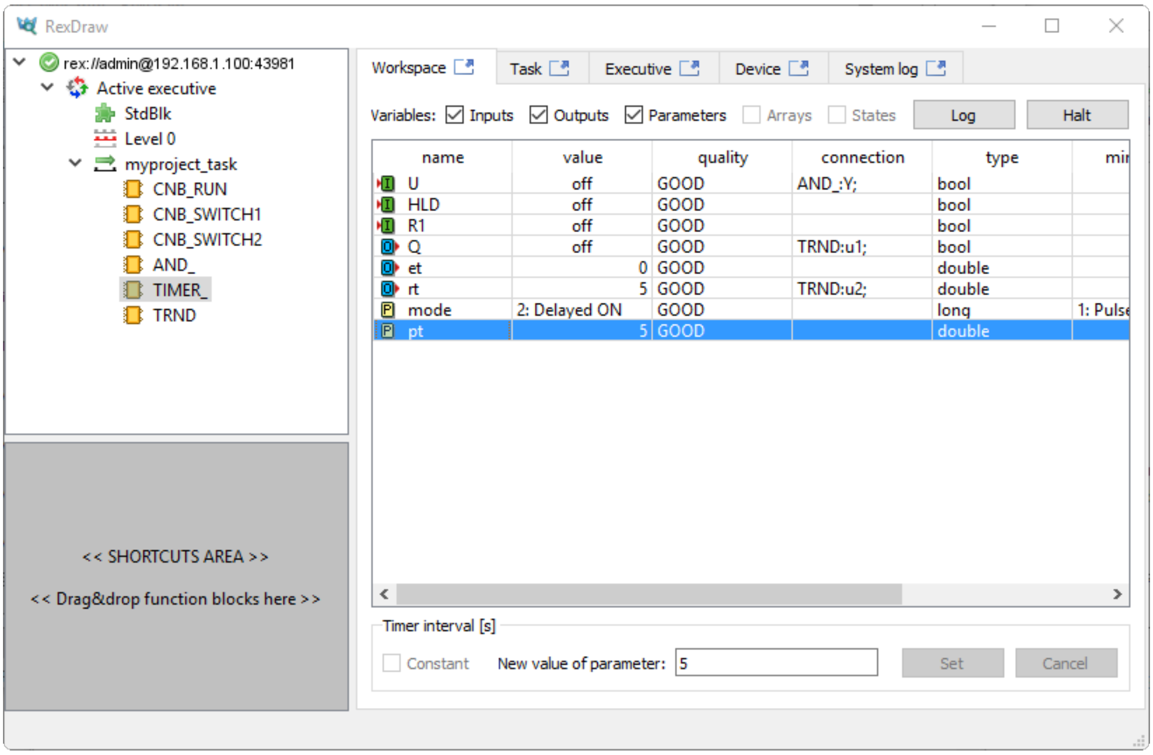
Period [ms]: 100 Priority: 5

Number of ticks: 2 Number of tasks: 1

Diagnostics (all time data displayed in milliseconds)

☑ Enable Exec. count: 840 Last time: 100

Min time: 98.9947 Max time: 101.005 Average time: 99.9999



- rex://admin@192.168.1.100:43981
  - Active executive
    - StdBlk
    - Level 0
    - myproject_task
      - CNB_RUN
      - CNB_SWITCH1
      - CNB_SWITCH2
      - AND_
      - TIMER_
      - TRND

Variables: ☑ Inputs ☑ Outputs ☑ Parameters □ Arrays □ States

name	value	quality	connection	type	mi
U	off	GOOD	AND_:Y;	bool	
HLD	off	GOOD		bool	
R1	off	GOOD		bool	
Q	off	GOOD	TRND:u1;	bool	
et	0	GOOD		double	
rt	5	GOOD	TRND:u2;	double	
mode	2: Delayed ON	GOOD		long	1: Puls
pt	5	GOOD		double	

Timer interval [s]

□ Constant New value of parameter: 5

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

Now you can disconnect RexDraw from the target device using the *Disconnect* icon. The RexDraw program offers synchronization of the changed parameters with the source file of the project, choose `No` at this moment.

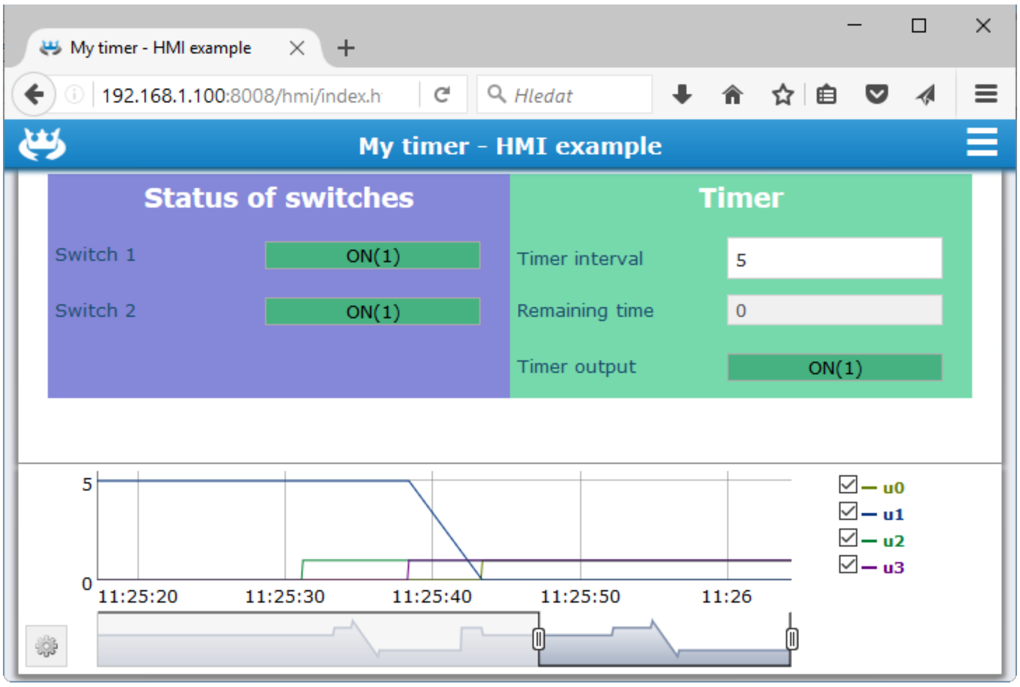
All changes made in online mode are not permanently stored in the target device (unless you decide so, see [3]). Upon restart of RexCore runtime module the algorithm will start with the parameters previously compiled and downloaded to the target device. To store the changes permanently, tick them when disconnecting and transfer the changes to the source files. Now *Compile and download* the project one more time and the changes become permanent.

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 who is unfamiliar 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 thus accessible via web browser on desktop PC, tablet or smartphone.

In this tutorial, only a simple HMI called WebBuDi will be created. It provides very simple indicators and input fields to interact with the control algorithm via a web page. See [4] for information on creating more advanced graphical HMI screens.

This is how the HMI will look like and the steps to create it are below.



My timer - HMI example

Status of switches

Switch 1 ON(1)

Switch 2 ON(1)

Timer

Timer interval 5

Remaining time 0

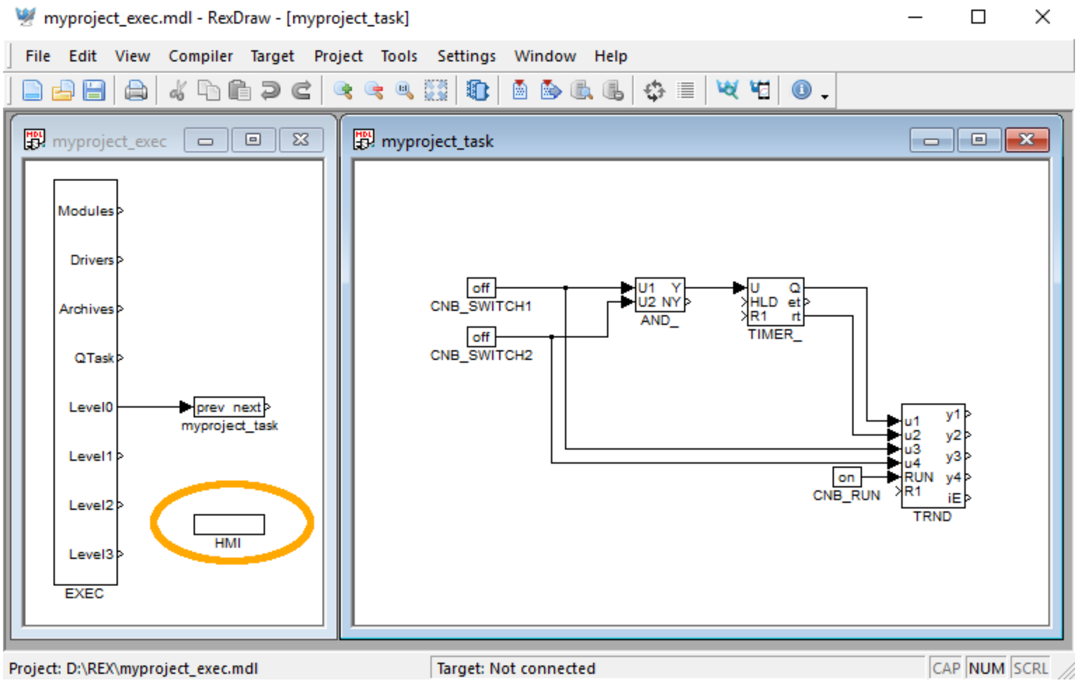
Timer output ON(1)

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

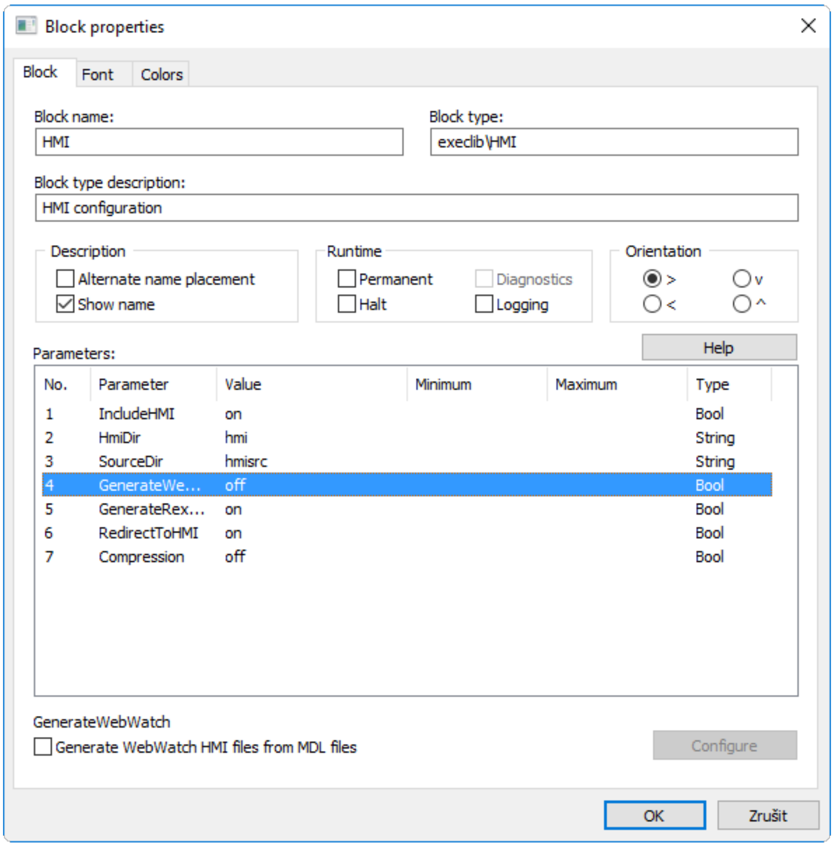
```
REX.HMI.init = function(){

//Indicators displaying the status of switches
  var switches = {
    column: 1,
    title: 'Status of switches',
    rows: [
      {alias: 'switch1', desc: 'Switch 1',
       cstring: 'myproject_task.AND_:U1', type: 'DR'},
      {alias: 'switch2', desc: 'Switch 2',
       cstring: 'myproject_task.AND_:U2', type: 'DR'},
      {type: 'ES'}
    ]
  };
  REX.WebBuDi.addSection(switches);
```

```
 //Timer settings and status
 var timer = {
  column: 2,
  title: 'Timer',
  rows: [
   {alias: 'interval', desc: 'Timer interval',
    cstring: 'myproject_task.TIMER_:pt', type: 'AW'},
   {alias: 'rt', desc: 'Remaining time',
    cstring: 'myproject_task.TIMER_:rt', type: 'AR'},
   {alias: 'timerQ', desc: 'Timer output',
    cstring: 'myproject_task.TIMER_:Q', type: 'DR'}
  ]
 };
 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');
}
```

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.



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.



Block name: HMI

Block type: execlib\HMI

Block type description: HMI configuration

Description
☐ Alternate name placement
☑ Show name

Runtime
☐ Permanent ☐ Diagnostics
☐ Halt ☐ Logging

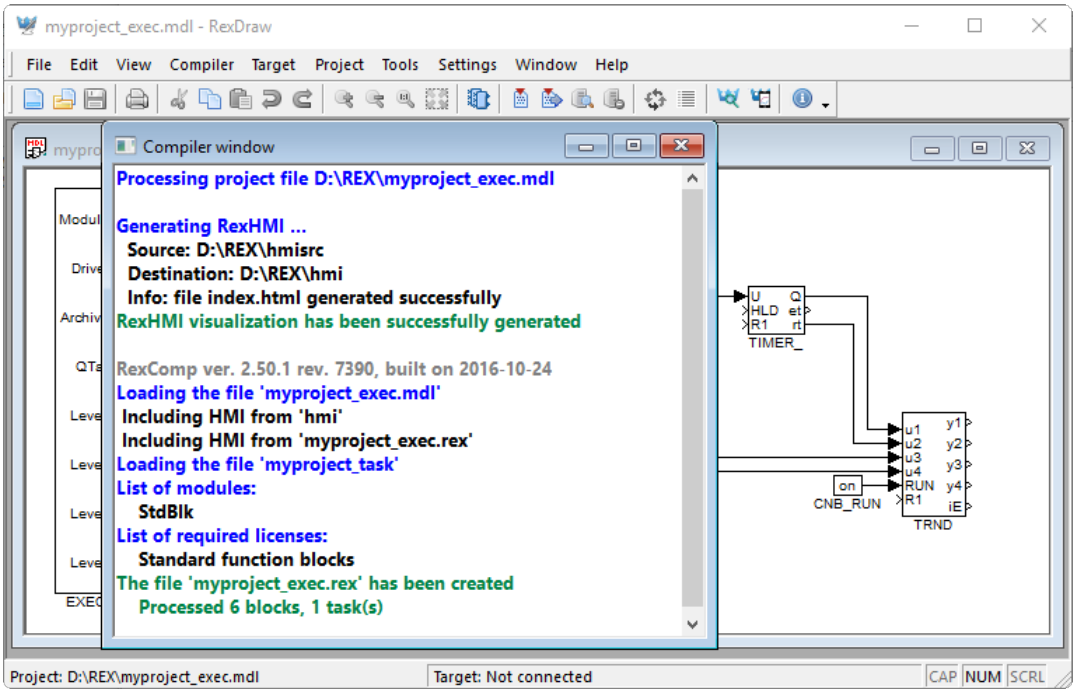
Orientation
● > ○ v
○ < ○ ^

Parameters:

No.	Parameter	Value	Minimum	Maximum	Type
1	IncludeHMI	on			Bool
2	HmiDir	hmi			String
3	SourceDir	hmisrc			String
4	GenerateWe...	off			Bool
5	GenerateRex...	on			Bool
6	RedirectToHMI	on			Bool
7	Compression	off			Bool

GenerateWebWatch
☐ Generate WebWatch HMI files from MDL files

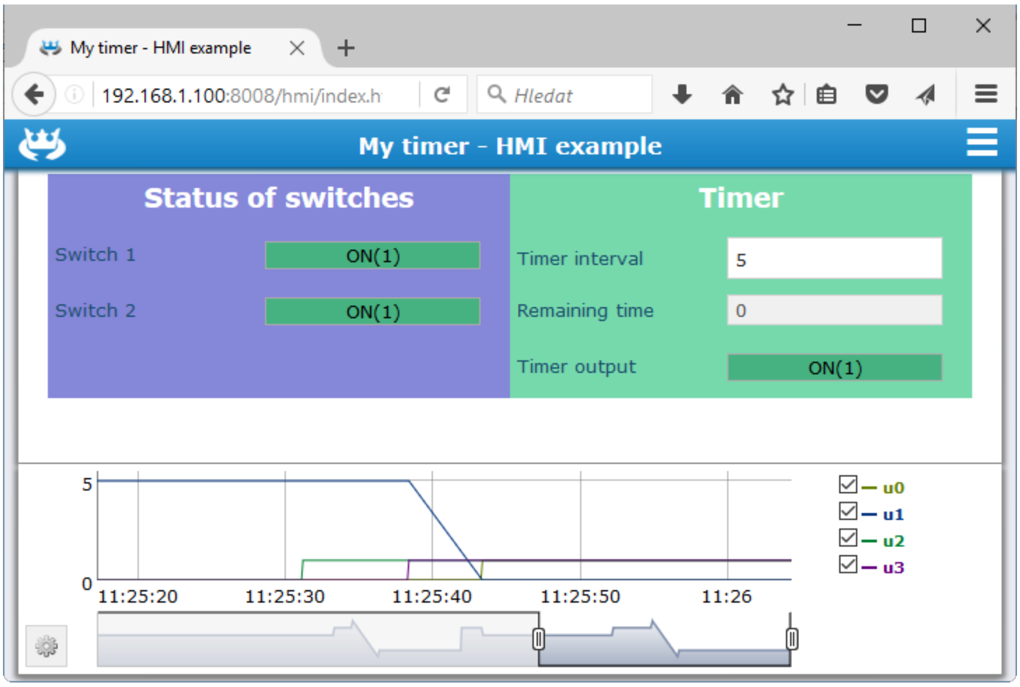
Once you compile the project again, you will see that the compile log contains more information. The HMI files were generated into the `hmi` subfolder and included in the resulting binary `myproject_exec.rex` file.



```
Processing project file D:\REX\myproject_exec.mdl

Generating RexHMI ...
 Source: D:\REX\hmisrc
 Destination: D:\REX\hmi
 Info: file index.html generated successfully
RexHMI visualization has been successfully generated

RexComp ver. 2.50.1 rev. 7390, built on 2016-10-24
Loading the file 'myproject_exec.mdl'
 Including HMI from 'hmi'
 Including HMI from 'myproject_exec.rex'
Loading the file 'myproject_task'
List of modules:
   StdBlk
List of required licenses:
   Standard function blocks
The file 'myproject_exec.rex' has been created
   Processed 6 blocks, 1 task(s)
```

After you download the project to the target device, you can access the HMI via web browser. The address is `http://192.168.1.100:8008/hmi/index.html` (use the IP address of your Raspberry Pi). There are only indicators of the Boolean values (physical switches) in this simple example. Creating virtual buttons is also possible, see [4] for details.

The `index.html` file is generated from the source `index.hmi.js` file. The default port of the webserver (8008) can be changed in RexCore settings. See [2] for details.



My timer - HMI example

192.168.1.100:8008/hmi/index.h

Hledat

My timer - HMI example

Status of switches

Switch 1 ON(1)

Switch 2 ON(1)

Timer

Timer interval 5

Remaining time 0

Timer output ON(1)

See also the documentation of RexHMI Designer, which allows you to create graphical user interfaces

4.4 Ready for interaction with the outer world

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.

Chapter 5

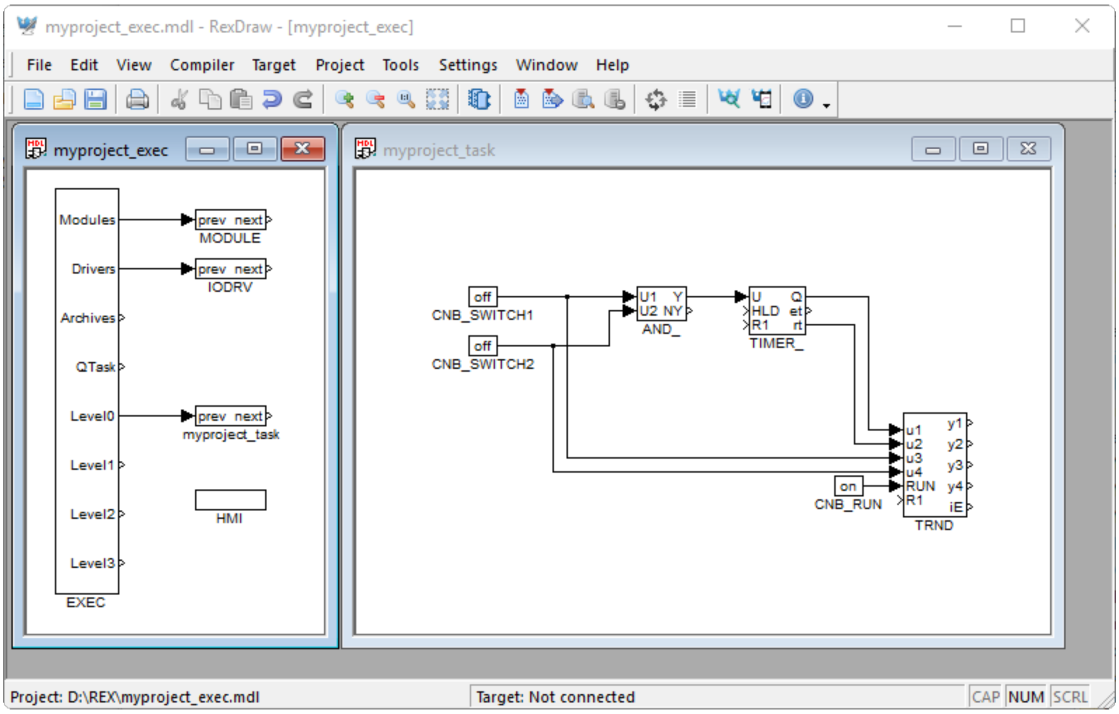
IO configuration for the Monarco HAT

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 Monarco HAT for interaction with the real world.

5.1 Adding inputs and outputs to the project

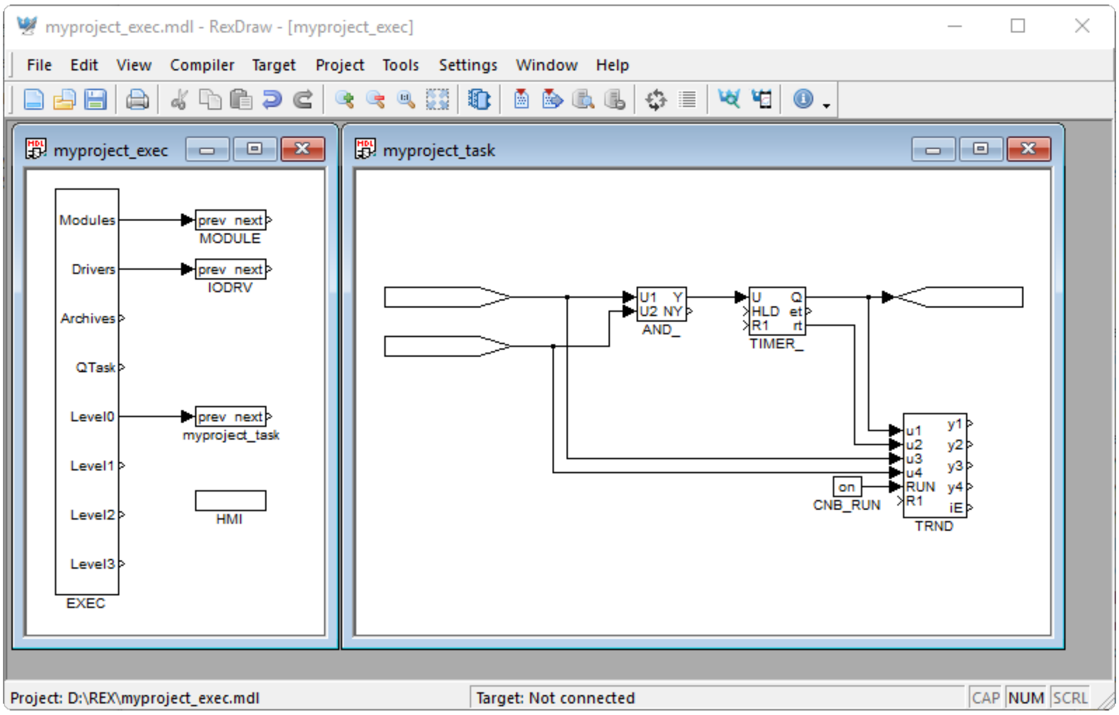
You need to expand your project main file with 2 additional function blocks to access the inputs and outputs from the control algorithm in your project. Insert the `EXEC/MODULE` and `EXEC/IODRV` blocks from the *Block library* and attach them to the `EXEC` block as shown below.



Project: D:\REX\myproject_exec.mdl

Target: Not connected

In the task file, delete the CNB_SWITCH1 and CNB_SWITCH2 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?



5.2 Working with I/O signals

5.2.1 Modifications in the project main file

Now we tell the compiler to use the Monarco HAT I/O driver. This is what needs to be done:

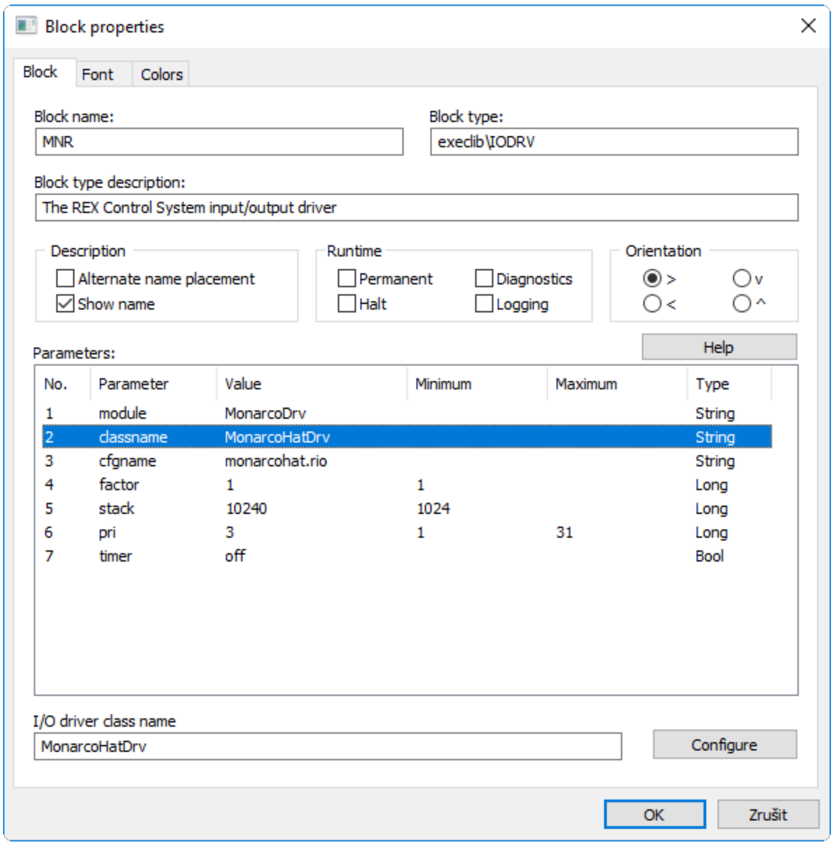
- Rename the `MODULE` block to `MonarcoDrv` – CASE SENSITIVE!

Further link the `IODRV` block with the `MonarcoDrv` module by setting

- `module` = `MonarcoDrv` – CASE SENSITIVE!

For Monarco HAT set the following:

- `classname` = `MonarcoHatDrv` – CASE SENSITIVE!
- `cfgname` = `monarcohat.rio`
- `factor` = `1`
- Leave the other parameters intact.



Block name: MNR

Block type: execlib\IODRV

Block type description: The REX Control System input/output driver

Description: □ Alternate name placement ☑ Show name

Runtime: □ Permanent □ Diagnostics □ Halt □ Logging

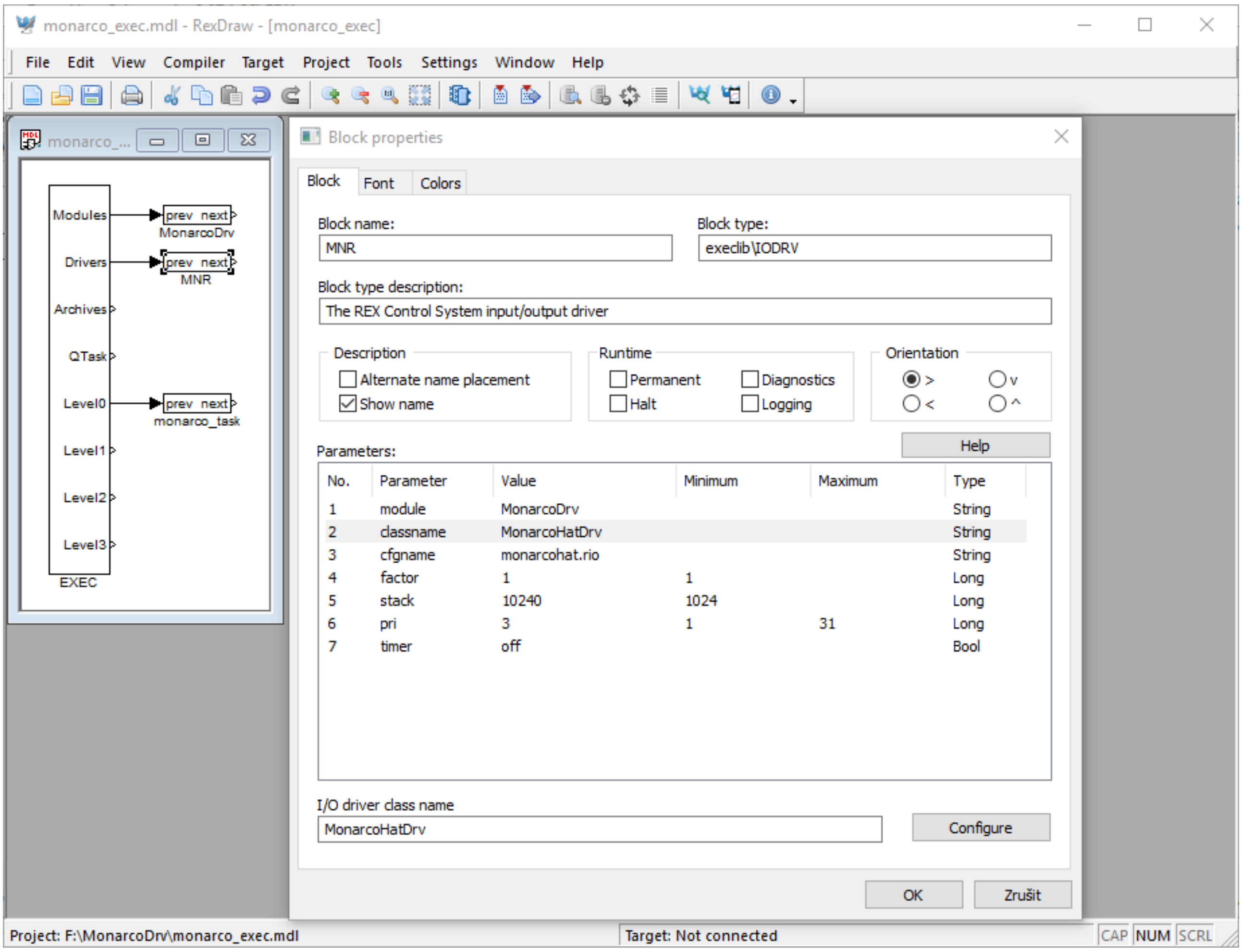
Orientation: ● > ○ v ○ < ○ ^

Parameters:

No.	Parameter	Value	Minimum	Maximum	Type
1	module	MonarcoDrv			String
2	classname	MonarcoHatDrv			String
3	cfgname	monarcohat.rio			String
4	factor	1	1		Long
5	stack	10240	1024		Long
6	pri	3	1	31	Long
7	timer	off			Bool

I/O driver class name: MonarcoHatDrv

Remember to click the *Configure* button. This will create a default I/O driver configuration file (.rio). Leave the default values and close the dialog.



Block name: MNR

Block type: execlib\IODRV

Block type description: The REX Control System input/output driver

Description: ☐ Alternate name placement ☑ Show name

Runtime: ☐ Permanent ☐ Diagnostics ☐ Halt ☐ Logging

Orientation: ● > ○ v ○ < ○ ^

Parameters:

No.	Parameter	Value	Minimum	Maximum	Type
1	module	MonarcoDrv			String
2	classname	MonarcoHatDrv			String
3	cfgname	monarcohat.rio			String
4	factor	1	1		Long
5	stack	10240	1024		Long
6	pri	3	1	31	Long
7	timer	off			Bool

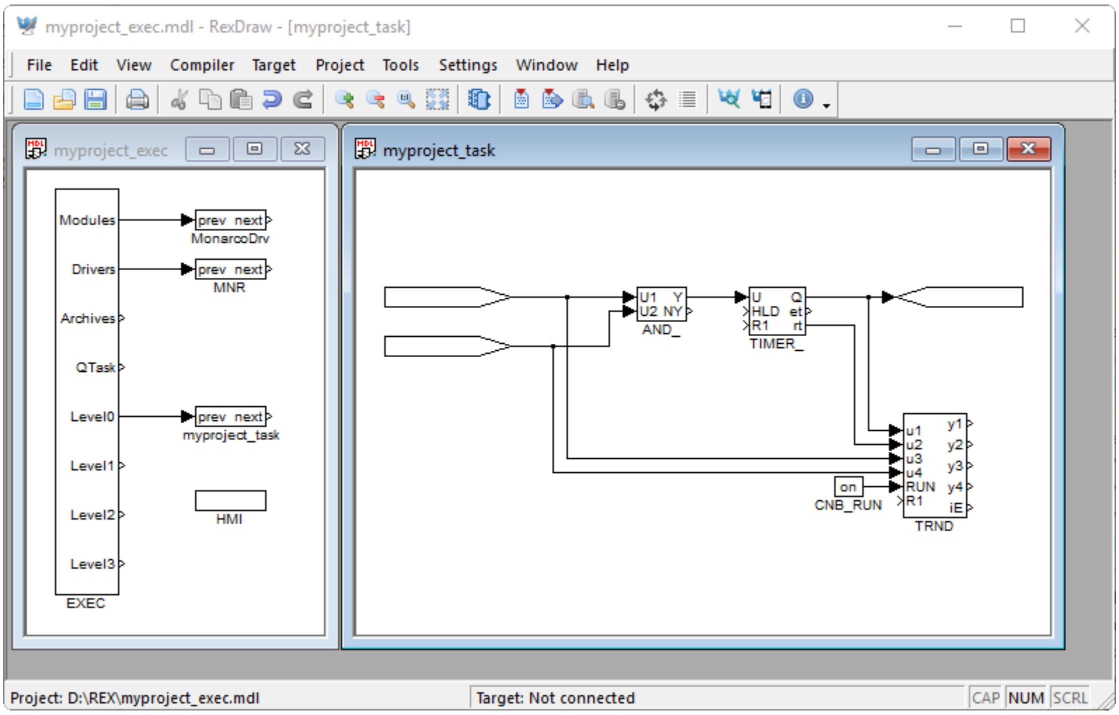
I/O driver class name: MonarcoHatDrv

Project: F:\MonarcoDrv\monarco_exec.mdl

Target: Not connected

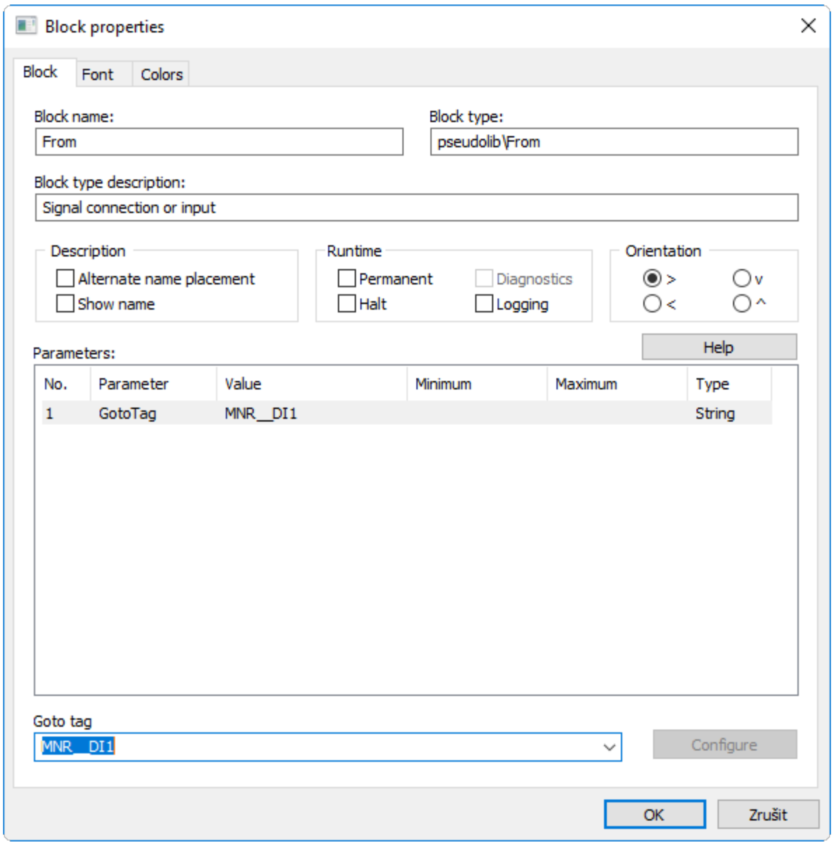
As the final step, rename the IODRV block to MNR, which will serve as a prefix for all I/O signals of this driver.

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



5.2.2 Modifications in the task

In the task double-click the input flag and set `GotoTag` = `MNR__DI1`.Note the MNR prefix and **two underscore characters**. The first physical switch will be connected to DI1.



Block name: From

Block type: pseudolib\From

Block type description: Signal connection or input

Description: ☐ Alternate name placement ☐ Show name

Runtime: ☐ Permanent ☐ Diagnostics ☐ Halt ☐ Logging

Orientation: ● > ○ v ○ < ○ ^

Parameters:

No.	Parameter	Value	Minimum	Maximum	Type
1	GotoTag	MNR__DI1			String

Goto tag: MNR__DI1

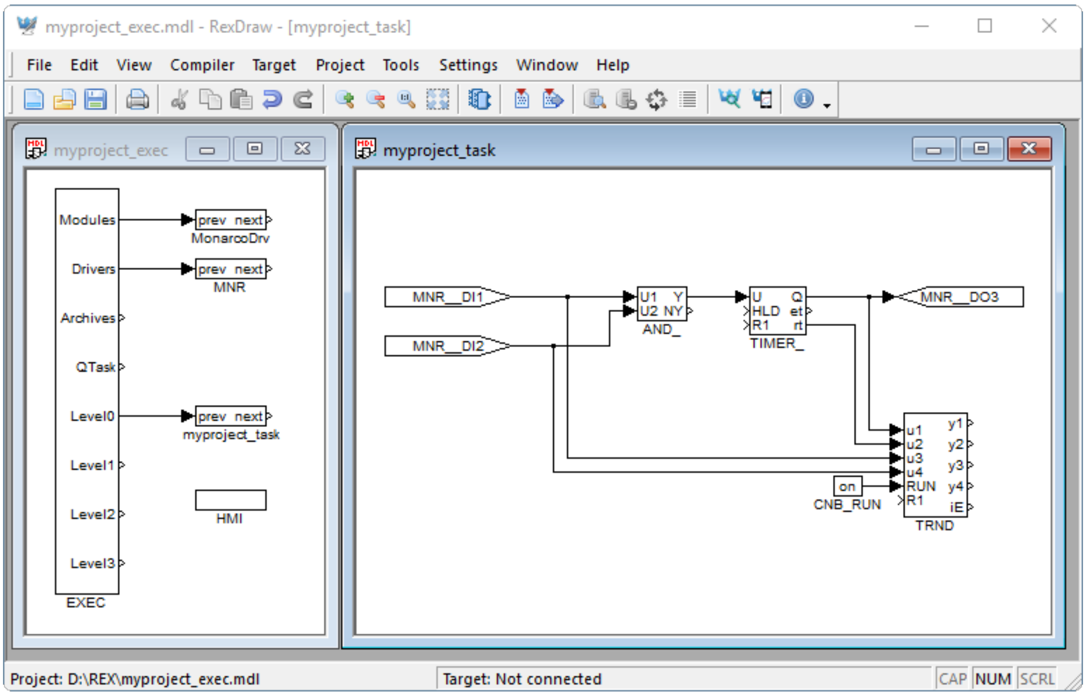
The second physical switch will be connected to DI2 (`MNR__DI2`) and DO1 will serve as the output signal (`MNR__DO1`).

Similarly for other pins we could use the following flags:

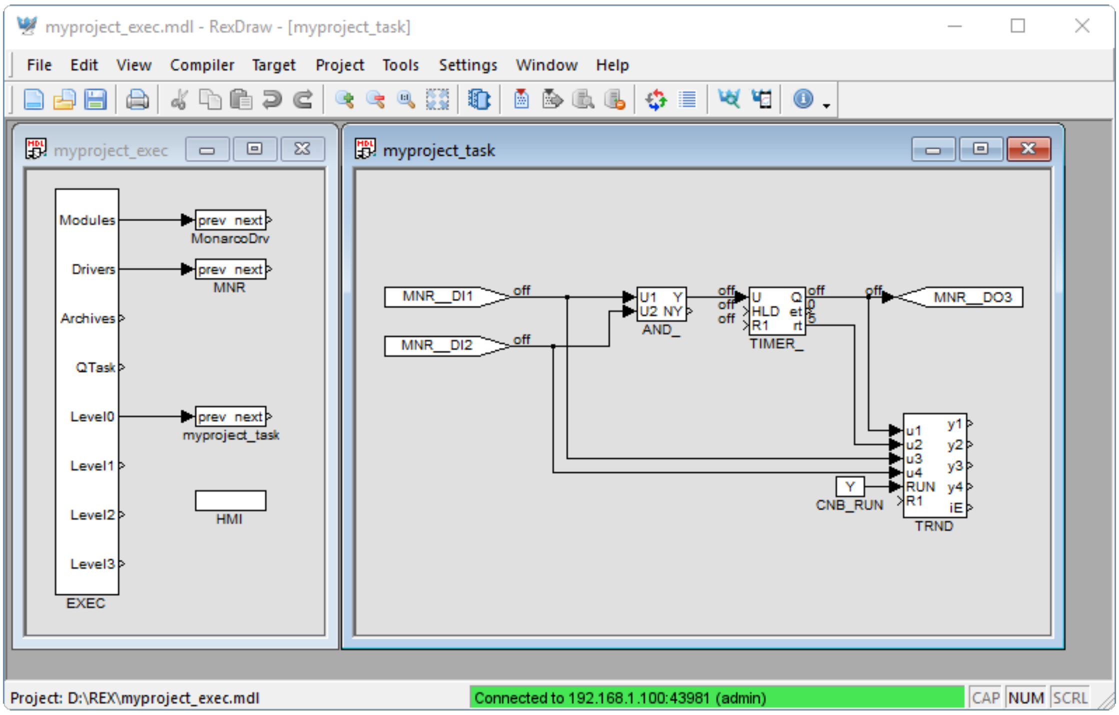
- `Goto`, `MNR__DO4` – digital output 4
- `From`, `MNR__DI3` – digital input 3
- `From`, `MNR__AI1` – analog input 1

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

Your project should now look like this:



After compiling the project and downloading it to the Raspberry Pi the control algorithm interacts with the physical world. Again it is possible to switch to online mode and watch the signals in real-time or analyze the trends of signals. Flip the physical switches and watch the signals.



Project: D:\REX\myproject_exec.mdl

Connected to 192.168.1.100:43981 (admin)

5.3 Additional information

5.3.1 Detailed description of the driver

A detailed description of the IO driver for Monarco HAT is available in a separate manual [5].

5.3.2 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 `Project` → `New project from template` and select one of the Monarco HAT examples.

Chapter 6

Licensing of the REX Control System

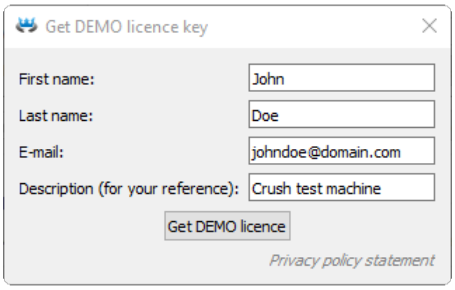
The licensing model of the REX Control System is quite simple:

- 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 Monarco HAT. There are DEMO licences available at no cost and there are permanent licences which you can purchase. Each Monarco HAT needs an individual licence.

6.1 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`).

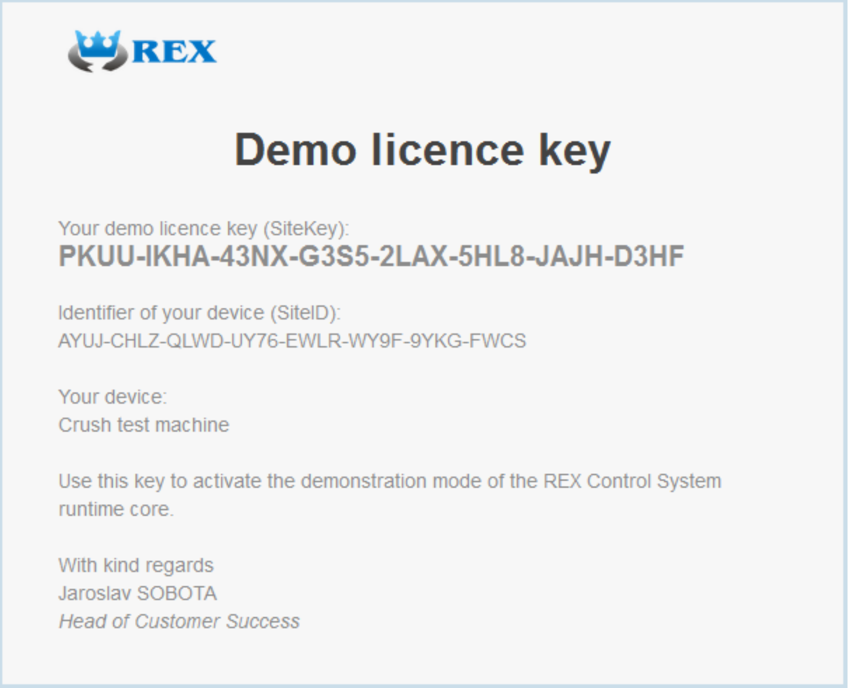


First name: John

Last name: Doe

E-mail: johndoe@domain.com

Description (for your reference): Crush test machine



Demo licence key

Your demo licence key (SiteKey):
PKUU-IKHA-43NX-G3S5-2LAX-5HL8-JAJH-D3HF

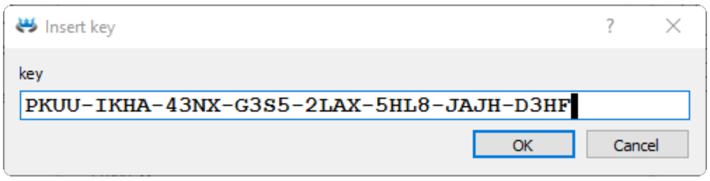
Identifier of your device (SiteID):
AYUJ-CHLZ-QLWD-UY76-EWLR-WY9F-9YKG-FWCS

Your device:
Crush test machine

Use this key to activate the demonstration mode of the REX Control System runtime core.

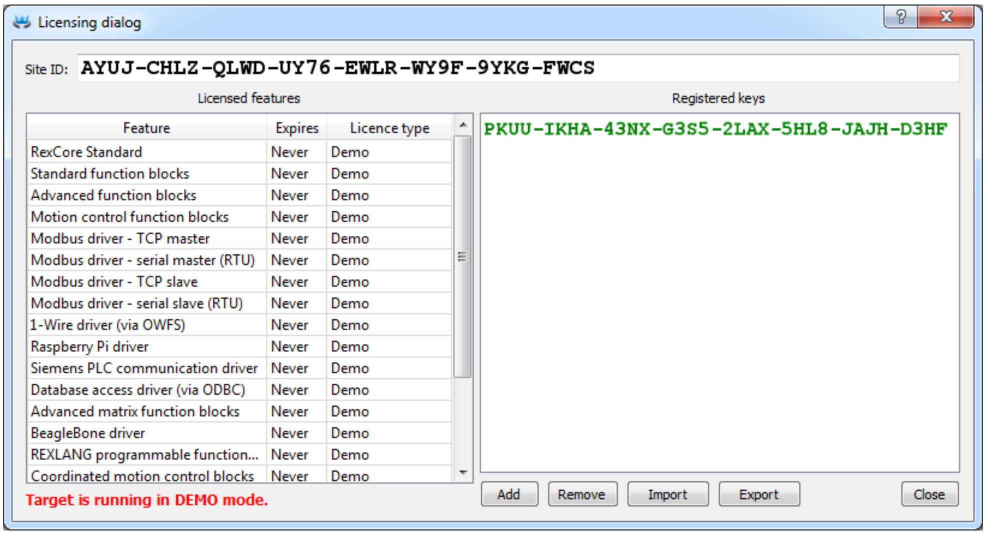
With kind regards
Jaroslav SOBOTA
Head of Customer Success

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



PKUU-IKHA-43NX-G3S5-2LAX-5HL8-JAJH-D3HF

Once applied, you will see all the modules.



Site ID: AYUJ-CHLZ-QLWD-UY76-EWLR-WY9F-9YKG-FWCS

Feature	Expires	Licence type
RexCore Standard	Never	Demo
Standard function blocks	Never	Demo
Advanced function blocks	Never	Demo
Motion control function blocks	Never	Demo
Modbus driver - TCP master	Never	Demo
Modbus driver - serial master (RTU)	Never	Demo
Modbus driver - TCP slave	Never	Demo
Modbus driver - serial slave (RTU)	Never	Demo
1-Wire driver (via OWFS)	Never	Demo
Raspberry Pi driver	Never	Demo
Siemens PLC communication driver	Never	Demo
Database access driver (via ODBC)	Never	Demo
Advanced matrix function blocks	Never	Demo
BeagleBone driver	Never	Demo
REXLANG programmable function...	Never	Demo
Coordinated motion control blocks	Never	Demo

PKUU-IKHA-43NX-G3S5-2LAX-5HL8-JAJH-D3HF

Target is running in DEMO mode.

Now you can run your algorithms on your Monarco HAT.

Evaluation version of the RexCore runtime is functional for 2 hours. It is possible to run your algorithm on the Monarco HAT but you cannot store it permanently, it 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.

6.2 Obtaining a permanent licence

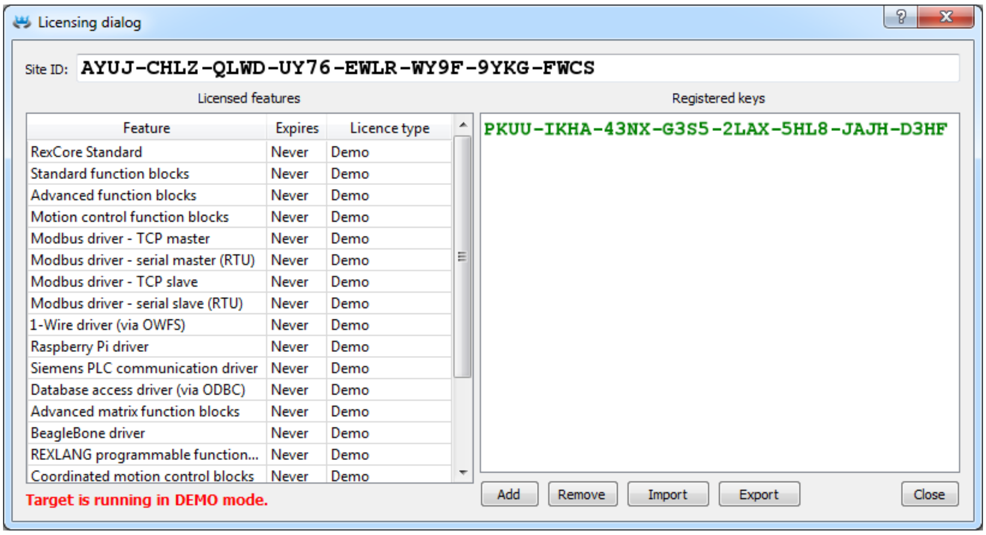
It is necessary to activate the RexCore runtime module and optional additional modules for permanent operation. This can be done using the licence, which you can obtain in an e-shop at

www.rexcontrols.com/e-shop

6.2.1 Activation of the permanent licence

The purchased licence must be associated with the hardware device.

Each device running the RexCore runtime module is identified by the so-called `SiteID` tag. You can get it in RexDraw or RexView when connected to the device. After connecting to the target device, go to menu *Target* → *Licensing...*. A dialog pops up and you can copy the `SiteID` identifier.



Site ID: AYUJ-CHLZ-QLWD-UY76-EWLR-WY9F-9YKG-FWCS

Licensed features

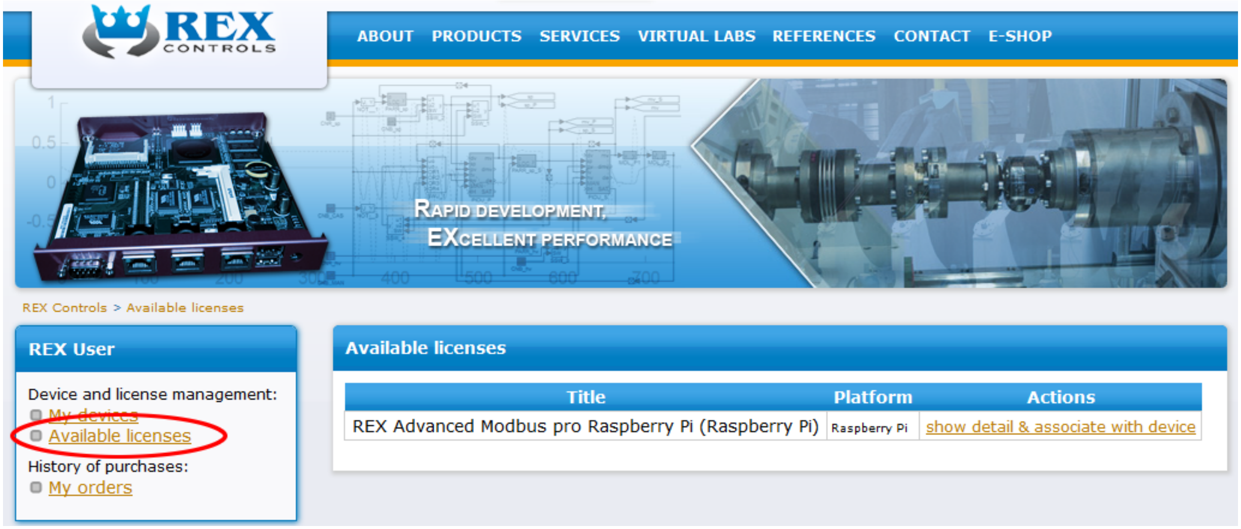
Feature	Expires	Licence type
RexCore Standard	Never	Demo
Standard function blocks	Never	Demo
Advanced function blocks	Never	Demo
Motion control function blocks	Never	Demo
Modbus driver - TCP master	Never	Demo
Modbus driver - serial master (RTU)	Never	Demo
Modbus driver - TCP slave	Never	Demo
Modbus driver - serial slave (RTU)	Never	Demo
1-Wire driver (via OWFS)	Never	Demo
Raspberry Pi driver	Never	Demo
Siemens PLC communication driver	Never	Demo
Database access driver (via ODBC)	Never	Demo
Advanced matrix function blocks	Never	Demo
BeagleBone driver	Never	Demo
REXLANG programmable function...	Never	Demo
Coordinated motion control blocks	Never	Demo

Registered keys

PKUU-IKHA-43NX-G3S5-2LAX-5HL8-JAJH-D3HF

Target is running in DEMO mode.

Afterwards, login to www.rexcontrols.com using your username and password and list the available licences which you possess.



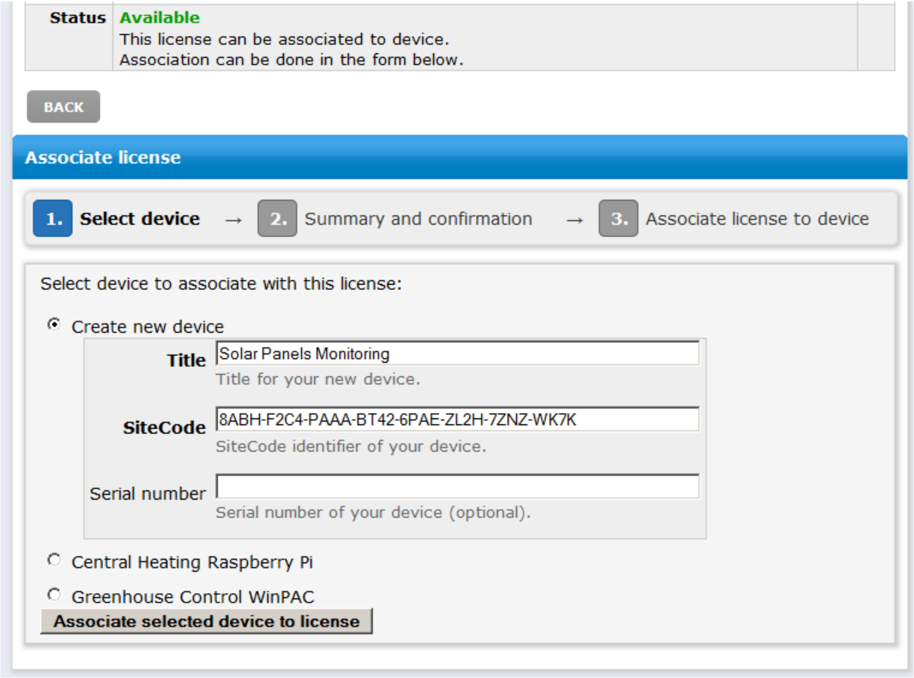
Device and license management:

History of purchases:

Available licenses

Title	Platform	Actions
REX Advanced Modbus pro Raspberry Pi (Raspberry Pi)	Raspberry Pi	show detail & associate with device

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
Associate selected device to license

Select device to associate with this license:

● Create new device

Title Solar Panels Monitoring
Title for your new device.

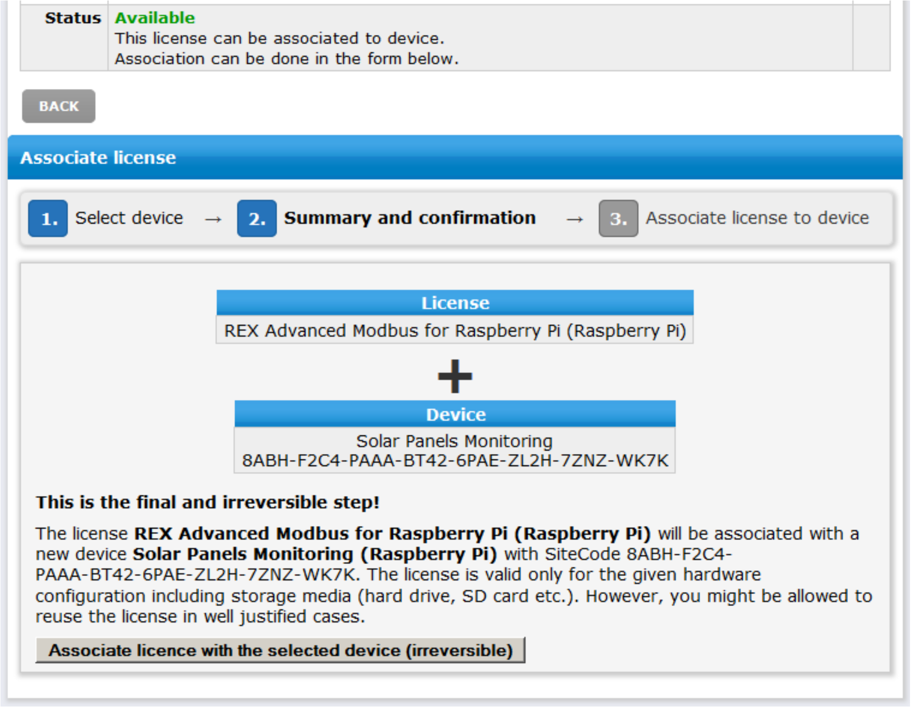
SiteCode 8ABH-F2C4-PAAA-BT42-6PAE-ZL2H-7ZNZ-WK7K
SiteCode identifier of your device.

Serial number
Serial number of your device (optional).

○ Central Heating Raspberry Pi

○ Greenhouse Control WinPAC

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



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

Associate license

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

License

REX Advanced Modbus for Raspberry Pi (Raspberry Pi)

+

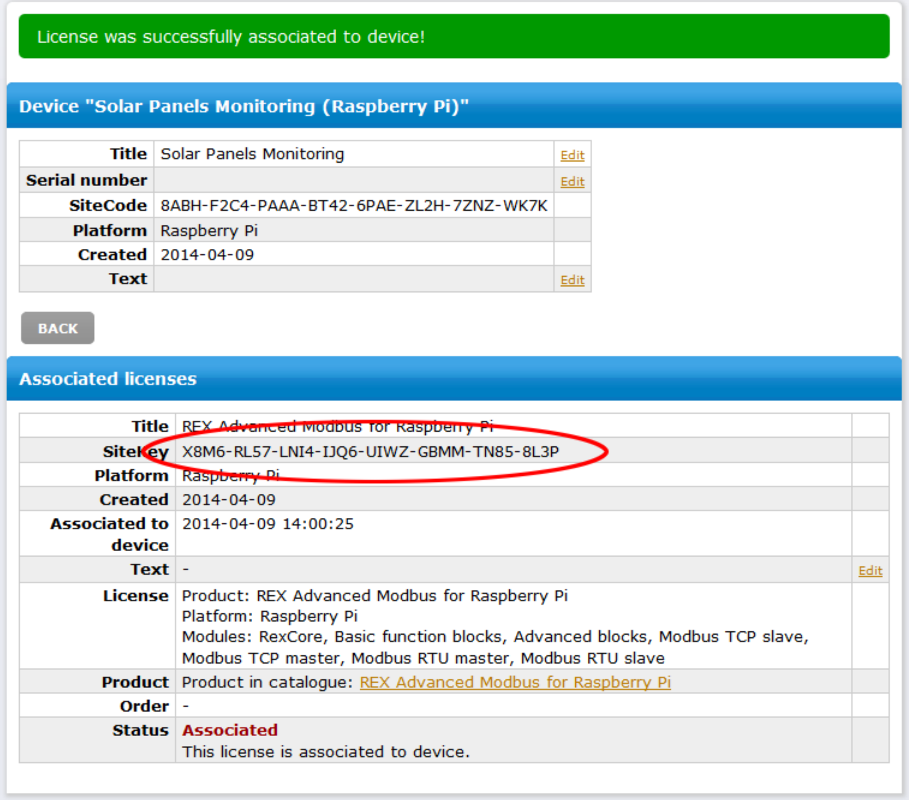
Device

Solar Panels Monitoring
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.

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



License was successfully associated to device!

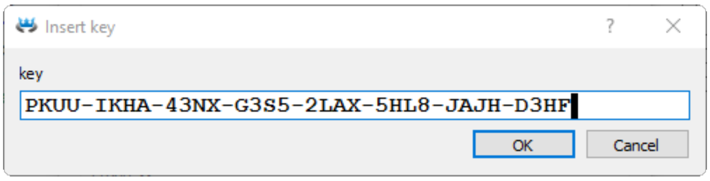
Device "Solar Panels Monitoring (Raspberry Pi)"

Title	Solar Panels Monitoring	Edit
Serial number		Edit
SiteCode	8ABH-F2C4-PAAA-BT42-6PAE-ZL2H-7ZNZ-WK7K	
Platform	Raspberry Pi	
Created	2014-04-09	
Text		Edit

Associated licenses

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

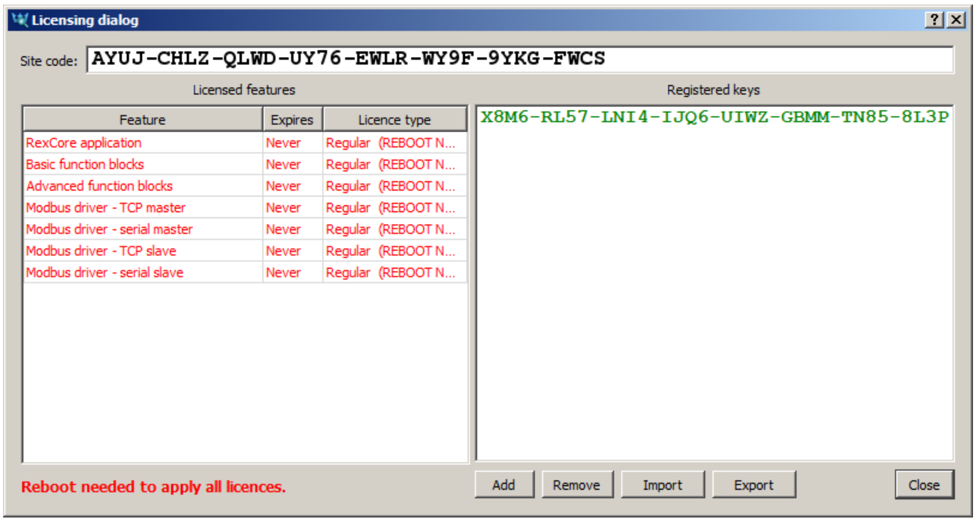
This key will allow permanent operation of the runtime core, but it must be applied to the target device. To do that, open the licensing dialog in RexDraw or RexView again and apply the `SiteKey` using the Add button.



key

PKUU-IKHA-43NX-G3S5-2LAX-5HL8-JAJH-D3HF

If the key is valid, the activated modules appear. The Monarco HAT must be restarted afterwards.



Site code: AYUJ-CHLZ-QLWD-UY76-EWLR-WY9F-9YKG-FWCS

Licensed features

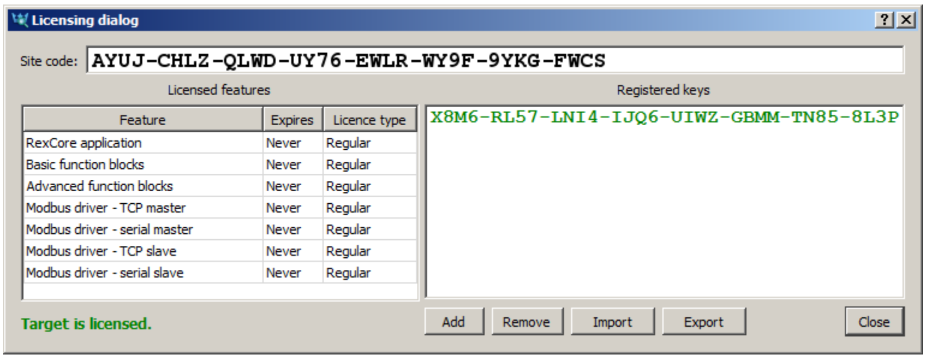
Feature	Expires	Licence type
RexCore application	Never	Regular (REBOOT N...
Basic function blocks	Never	Regular (REBOOT N...
Advanced function blocks	Never	Regular (REBOOT N...
Modbus driver - TCP master	Never	Regular (REBOOT N...
Modbus driver - serial master	Never	Regular (REBOOT N...
Modbus driver - TCP slave	Never	Regular (REBOOT N...
Modbus driver - serial slave	Never	Regular (REBOOT N...

Registered keys

X8M6-RL57-LNI4-IJQ6-UIWZ-GBMM-TN85-8L3P

Reboot needed to apply all licences.

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



Site code: AYUJ-CHLZ-QLWD-UY76-EWLR-WY9F-9YKG-FWCS

Licensed features

Feature	Expires	Licence type
RexCore application	Never	Regular
Basic function blocks	Never	Regular
Advanced function blocks	Never	Regular
Modbus driver - TCP master	Never	Regular
Modbus driver - serial master	Never	Regular
Modbus driver - TCP slave	Never	Regular
Modbus driver - serial slave	Never	Regular

Registered keys

X8M6-RL57-LNI4-IJQ6-UIWZ-GBMM-TN85-8L3P

Target is licensed.

Bibliography

[1] REX Controls s.r.o.. *Function blocks of the REX Control System – reference manual*, 2016.

[2] REX Controls s.r.o.. *RexCore – User manual*, 2016.

[3] REX Controls s.r.o.. *RexDraw – User manual*, 2016.

[4] REX Controls s.r.o.. *RexHMI – User manual*, 2016.

[5] REX Controls s.r.o.. *MonarcoDrv driver of the REX Control System – user guide*, 2016.

Documentation reference number: 7567