REX www.rexcontrols.com/rex

RexHMI – A Web-based HMI for REX

User guide

REX Controls s.r.o.

Version 2.50.5
2017-09-06
Plzeň (Pilsen), Czech Republic

Contents

Chapter 1

Introduction

RexHMI covers all tools and libraries necessary for creating human-machine interfaces (visualizations) for the REX control system. There are three different types of visualization the WebWatch, the WebBuDi and the one created by RexHMI Designer.

- WebWatch(Chapter 2) is an auto-generated HMI from the RexDraw development tool during project compilation. It has similar look, attributes and functions as the Watch mode of the RexDraw development tool. The WebWatch is a perfect tool for instant creation of HMI that is suitable for system developers or integrators. It provides a graphical interaction with almost all signals in the control algorithm.

- RexHMI Designer(Chapter 3) creates a standard SVG file with the *RexHMI* extensions. The RexHMI Designer is a great tool for creating graphical HMI that is suitable for operators and other end users.

- WebBuDi(Chapter 4), which is an acronym for Web Buttons and Displays, is a simple JavaScript file with several declarative blocks that describe data points which the HMI is connected to and assemble a table in which all the data is presented. It provides a textual interaction with selected signals and is suitable for system developers and integrators or may serve as a fall-back mode HMI for non-standard situations.

All the tools result to the HTML5 web page served from the internal REX web server. The HMI can be accessed using desktop, tablets and also mobile devices. Recommended web browsers are **Google Chrome** and **Mozilla Firefox**.

Usually the HMI is downloaded to the target device using *HMI* block in the *exec.mdl* file. In the project directory create *exec.mdl* file (or use predefined templates from the Start-up wizard). Add the *HMI* block to the executive, set the *IncludeHMI* parameter and then enable *WebWatch* visualization (*GenerateWebWatch* parameter) or create your custom one. Once configured the visualization is downloaded using *Compile and Download* function in the RexDraw design tool to the target device. The HMI is accessible from the internal server of the target device for example at `http://127.0.0.1:8008/hmi`.

The HMI uses HTML, CSS3, JavaScript and WebSockets.

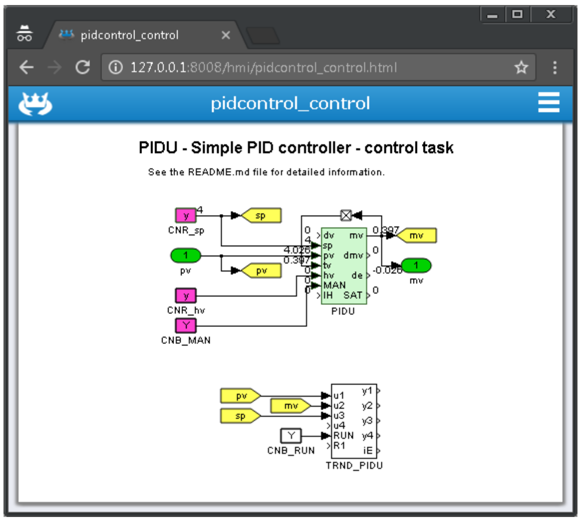
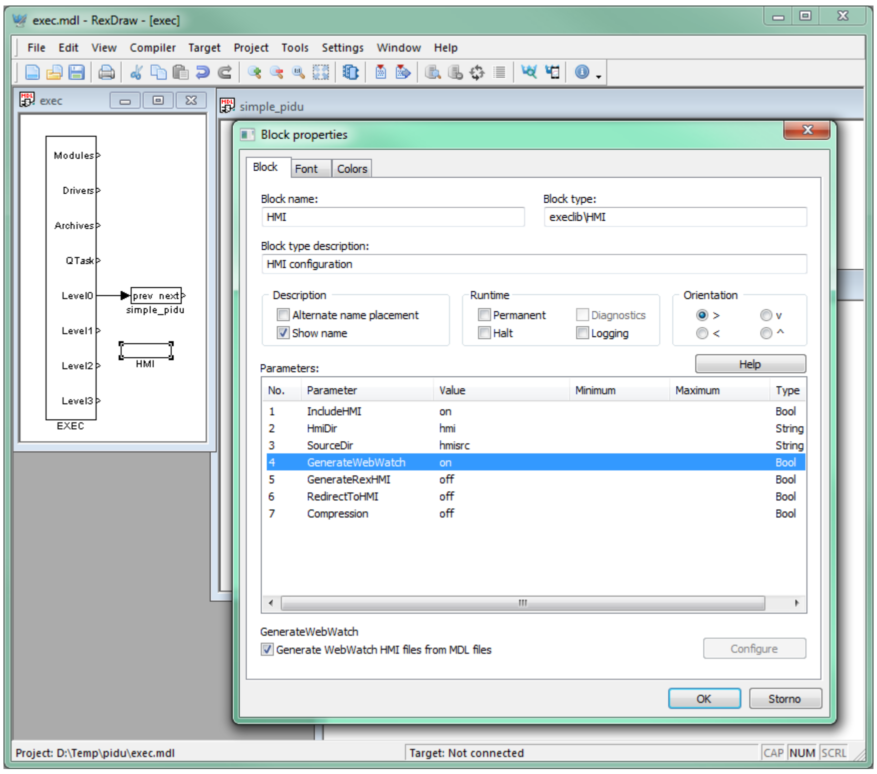

pidcontrol_control
127.0.0.1:8008/hmi/pidcontrol_control.html
PIDU - Simple PID controller - control task
See the README.md file for detailed information.
CNR_sp
pv
CNR_hv
CNB_MAN
PIDU
mv
CNB_RUN
TRND_PIDU

Figure 1.1: The example of *WebWatch* visualization

For the special cases user can create its own HMI based on the JavaScript and custom HTML or SVG components. For that purpose the *REX.HMI* function library is available. Chapter 5 describes all functions with parameters.



Block name: HMI

Block type: execlib\HMI

Block type description: HMI configuration

Description: Alternate name placement [ ], Show name [✓]

Runtime: Permanent [ ], Halt [ ], Diagnostics [ ], Logging [ ]

Orientation: [✓ >] < v ^

Parameters:

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

GenerateWebWatch

[✓] Generate WebWatch HMI files from MDL files

Project: D:\Temp\pidu\exec.mdl

Target: Not connected

CAP NUM SCRL

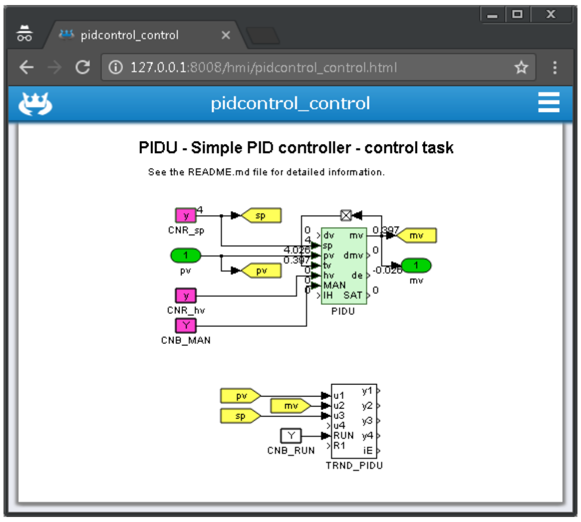
Figure 1.2: Configuration of the *HMI* block in RexDraw

Chapter 2

WebWatch

2.1 Automatic Generation of HMI from RexDraw

WebWatch is automatically generated HMI based on the project structure. It is similar to the *Watch mode* in RexDraw. The whole scheme is generated to web page. User can monitor all signals from selected blocks, change block parameters and read the data from TRND blocks. The WebWatch is generated automatically using *HMI* block in project main file.


pidcontrol_control

127.0.0.1:8008/hmi/pidcontrol_control.html

pidcontrol_control

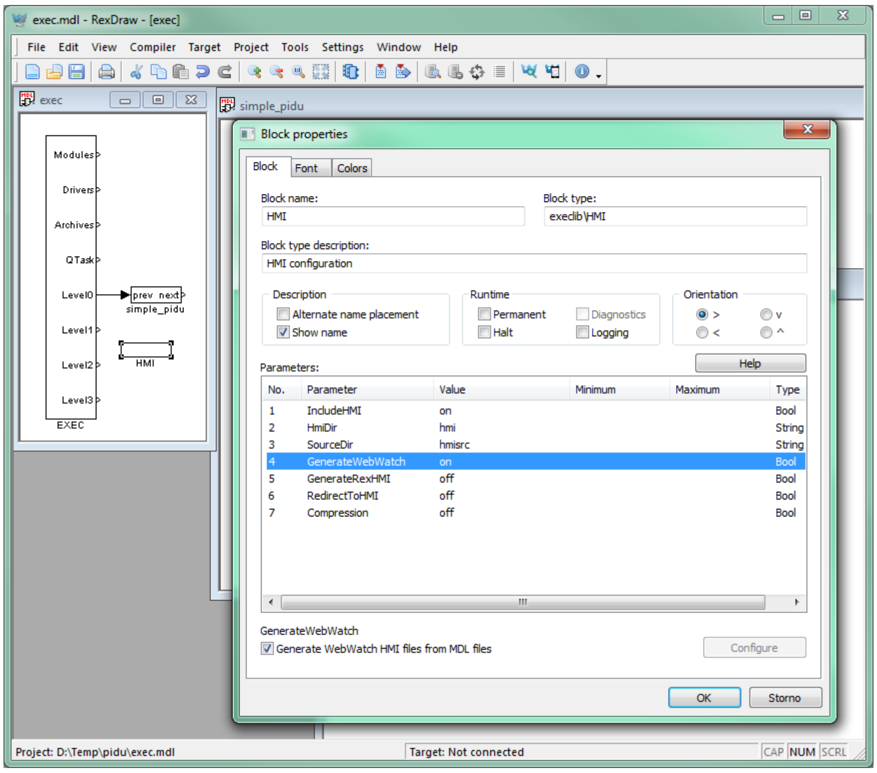
PIDU - Simple PID controller - control task

See the README.md file for detailed information.

Figure 2.1: The example of WebWatch visualization

1. Insert HMI block to the *exec.mdl* file of your project
2. Check `GenerateWebWatch` and `IncludeHMI` to enable WebWatch generation
3. Run *Compile and Download*

4. Open the web browser on your target device eg. `http://127.0.0.1:8008/hmi`.
5. Use *left mouse click (touch)* for changing the block parameters and *right mouse click (long touch)* for enabling the block monitoring.



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	GenerateWebWatch	on			Bool
5	GenerateRexHMI	off			Bool
6	RedirectToHMI	off			Bool
7	Compression	off			Bool

GenerateWebWatch

[✓] Generate WebWatch HMI files from MDL files

Project: D:\Temp\pidu\exec.mdl

Target: Not connected

CAP NUM SCRL

Figure 2.2: Check GenerateWebWatch to create WebWatchHMI.

2.2 Advanced Usage

The auto-generated scheme can be adjusted manually. [1]

Edit the `REX.HMI.init = function(){}` function in the selected `*.html` file. There are only few functions for the WebWatch HMI, but you can use all the functions described in chapter 5.

REX.WebWatch.enableMonitoring(blockIDs) – Enable monitoring of the selected blocks from the beginning (after web page is loaded).

[1] Note: If the *GenerateWebWatch* in HMI block is enabled the HMI is generate whenever the scheme is compiled. So for manual adjustment disable the *GenerateWebWatch* option in *HMI* block.

Param	Type	Description
blockIDs	Array .<String>	List of all blocks, described by connection string (eg.[“task.block1”,“task.block2”])

Example

```
// Enable monitoring of selected blocks
REX.HMI.init = function(){
  REX.WebWatch.enableMonitoring(["pidcontrol_control.CNR_sp","
      pidcontrol_control.PIDU"]);
}
```

REX.WebWatch.disableHint() – Disable hint after page is loaded

REX.WebWatch.showHint() – Show hint

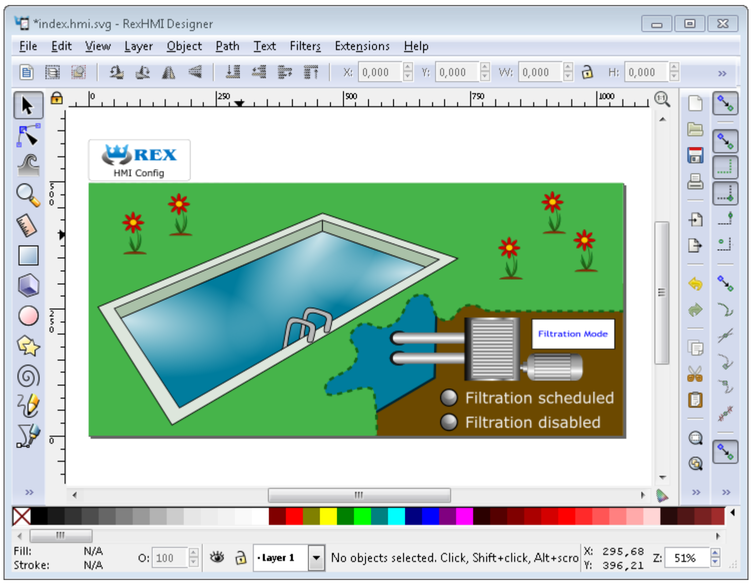
Chapter 3

RexHMI Designer

3.1 A Graphical Designer of Web HMI

RexHMI Designer[1] is tool for designing the custom visualizations using predefined components. The whole HMI is stored in the SVG (Scalable Vector Graphics) file with REX specific extensions.

When the visualization is ready the HTML page with all necessary libraries is generated. Such a webpage can be easily generated and downloaded to the target device from *./hmi* directory using *Compile and Download* function of the RexDraw development tool.


*index.hmi.svg - RexHMI Designer
File Edit View Layer Object Path Text Filters Extensions Help
X: 0,000 Y: 0,000 W: 0,000 H: 0,000
REX
HMI Config
Filtration Mode
Filtration scheduled
Filtration disabled
Fill: N/A
Stroke: N/A
O: 100
Layer 1
X: 295,68 Y: 396,21
Z: 51%

No objects selected. Click, Shift+click, Alt+scro

Figure 3.1: The example of RexHMI Designer visualization

[1]The RexHMI Designer is build on the well-known SVG editor Inkscape^{TM} https://inkscape.org/en/.

Each HMI scheme composes from several components which are connected to the signals from the control system. These components are organized in libraries. The example HMI on fig. 3.1 contain one *Button*, two *LEDs* and several *GeneralComponents* for pool and flower animations.

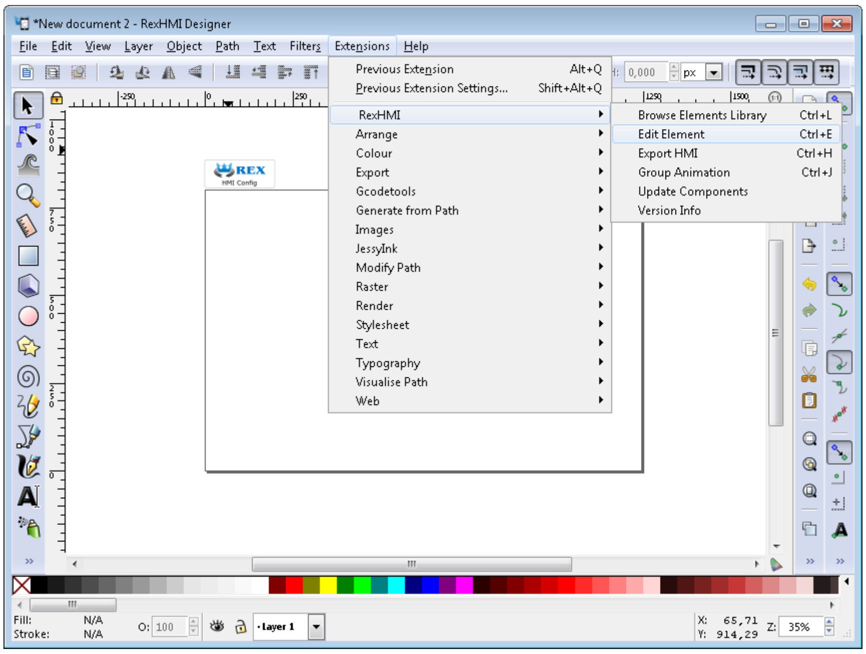


Figure 3.2: The list of RexHMI extensions

The HMI components are controlled via extensions (on fig. 3.2) which edit the RexHMI specific parameters and allow the export of the final HMI.

- **Browse Components Library** (Ctrl + L) – Open the folder with all available components. Each component is stored as a SVG file which can be drag&dropped to the current HMI
- **Edit Component** (Ctrl + E) – Open edit window for the selected component or for the whole project if nothing is selected.
- **Export HMI** (Ctrl + H) – Exports the content of the SVG file to the HTML5 webpage with all necessary libraries. This extension is configured via parameters of the whole project.
- **Group Animation** (Ctrl + J) – Enable animation of the transformations (translation, rotation, scale), opacity or color of the selected object (SVG group) based on the values from the control system.
- **Update Components** – Extension for updating the schema to new version

- **Version Info** – Show current version of the RexHMI Designer and RexHMI tools.

Follow the tutorials on www.rexcontrols.com to know how to use the extensions for creating your custom HMIs.

3.2 List of available components

There are several predefined HMI components in the component library. Every component is represented as one SVG file which can be drag&drop to the current visualization scheme. The library is divide to three main groups:

- `GENERAL` - Set of common components like buttons, inputs, sliders, gauges for visualization of basic variables.
- `HOME` - Set of components for home automation, like various pipes, valves, radiators, boilers and more.
- `HVAC` - Specialized library for heating, ventilation and air conditioning vizualizations.

Every component has some data points which provide the interaction between live data and state of the HMI. Mostly you can find:

- `value` - the main data point where value is read or written,
- `disable_by` - the component is disabled when connected to non-zero value,
- `hide_by` - the component is hidden when connected to non-zero value.

and various options which are described in the following text. The components are sort alphabetically.

3.2.1 HMIConfig

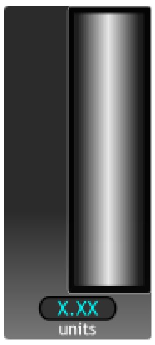
Global RexHMI configuration component. User can change project options, configure the HTML export and define list of connections between REX target and HMI.

Options	Type	Description

`target_url`	str	The URL of the target device. So far only the WebSocket protocol is supported. The URL must also contain the PORT, but the protocol can be omitted. Default is empty string which means ws://127.0.0.1:8008 Examples: 192.168.1.100:8008 or wss://192.168.1.100:8009 *Note: If the HMI is hosted on the internal web server this option is ignored and HMI connects to the hostname / IP you use in the webbrowser URL.*
`refresh_rate`	int	Changes the default read refresh rate of the whole HMI. Default 500ms
`log_level`	enum	Set the global log level for whole RexHMI. The log messages are written both to the right upper corner menu and to the web browser console. Default level is INFO.
`show_clock`	bool	If True the clock are shown in the right upper corner of the template instead of blinking square. If the target is connected the clock are updated if connection get error o lost the time stops.
`use_client_time`	bool	If True the `show_clock` display the time from client computer rather then target. This can be use when target device does not have NTP synchronization or RTC chip.
`kiosk_mode`	bool	If True the links from header (from logo and header) are removed and user can not change the refresh rate and target.

3.2.2 GENERAL

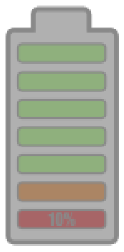
BarGraph



Bargraph component which draw scale on the left side and contain color zones and current value display

Data points	Optional	Description
`value`		Displayed value
`disable_by`	True	If non-zero the component is disabled.
`hide_by`	True	If non-zero the component is hidden.
Options	**Type**	**Description**
`rangeMin`	float	Lower limit for the bargraph value
`rangeMax`	float	Upper limit for the bargraph value
`tickStep`	float	Value for small tick step between main tick steps.
`mainTickStep`	float	Value for main axis tick steps
`decimals`	float	How many decimal figures are displayed
`units`	str	Text which is displayed on the bottom of the bargraph. Usually the name of units.
`colorZones`	table	User can define several color zones which can represent some level of warning. The zones are displayed on the right side of the bargraph. Each has a custom color and range defined by `startValue` and `endValue`. Just double click the `colorZones` and change them in table.
`colorOffLimits`	color	The border color of the bargraph when the value is of `rangeMin` and `rangeMax` limits.
`levelColor1`	color	Change the main bargraph gradient color
`levelColor2`	color	Change the supplementary bargraph gradient color

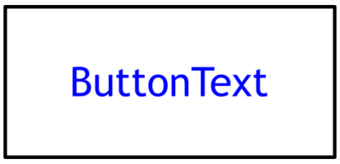
Battery



Show battery level from 0 to 100

Data points	Optional	Description
`value`		Battery level from 0 to 100
Options	**Type**	**Description**

Button



The Button component changes the boolean value. Normally the ON state means TRUE value and OFF mean FALSE value. If needed the parameter `reverseMeaning` switches the written values.
There are three different modes of operation:
`PushButton` when press or touched the ON state is written to the target `value` datapoint. When released the OFF state is written.
`ManualPulse` always write TRUE value after button is pressed, release is ignored.(Suitable for MP block control)
`ToggleButton` when button is pressed the value is toggled (ON -> OFF -> ON).

Data points	Optional	Description
`value`		The main datapoint where boolean value is written.
`refresh_from`	True	If set the state of the component is updated based on the value read from this datapoint. Otherwise the value from main datapoint is used.
`disable_by`	True	If non-zero the component is disabled.
`hide_by`	True	If non-zero the component is hidden.
Options	**Type**	**Description**
`fontScale`	float	Use this parameter to scale font because the button is using HTML button component which is not binded with SVG.
`reverseMeaning`	bool	If True the meaning of the value is switched. So the ON state is False and OFF state is True.
`text`	str	Text written on the button.
`type`	enum	Changes the type of the Button. `PushButton` when press or touched the ON state is written to the target `value` datapoint. When released the OFF state is written. `ManualPulse` always write TRUE value after button is pressed, release is ignored.(Suitable for MP block control) `ToggleButton` when button is pressed the value is toggled (ON -> OFF -> ON).

ComboBox

Combobox allow to choose item (value) from predefine list. If the value in the control system is not in the list it is automatically added.

Data points	Optional	Description
`value`		Datapoint where values are written after select.
`refresh_from`	True	If set the state of the component is updated based on the value read from this datapoint. Otherwise the value from main datapoint is used.
`disable_by`	True	If non-zero the component is disabled.
`hide_by`	True	If non-zero the component is hidden.
Options	**Type**	**Description**
`values`	table	Define value:desc pairs for the combobox list. `Value` is number or string and `desc` is it's string description.
`showValue`	bool	If True the values are shown in the list eg. `0:OK`
`valueType`	enum	Choose if the value is read as a number or string.
`fontScale`	float	Use this parameter to scale the font because the Combobox is using HTML select component which is not binded with SVG.

ControlLed

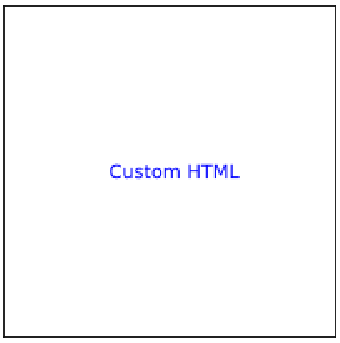
Simple boolean state indicator. Empty or Zero mean OFF, otherwise ON.

Data points	Optional	Description
`value`		Datapoint from which the LED state is read.

`disable_by`	True	If non-zero the component is disabled.
`hide_by`	True	If non-zero the component is hidden.

Options	Type	Description
`reverseMeaning`	bool	If True the meaning of the value is switched. So the ON state is False and OFF state is True.
`color_true`	color	The ON state color.
`color_false`	color	The OFF state color.

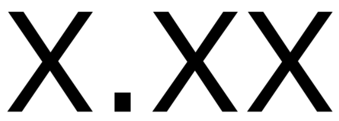
CustomHTML



CustomHTML component allows inserting custom HTML to the HMI. It creates empty DIV element which is filled with content from the `html` parameter. It can be used for example for including the live camera feed.

Options	Type	Description
`html`	str	Custom HTML content which is insert directly to the HMI.

Display



Display component shows value from the target system as a string. It can be either `number`, `date`, `datetime` or simple value as a `text` string depending on the `format` property. Each display can have `units` which are shown at the end of the displayed string.

Data points	Optional	Description
`value`		Datapoint from which the value is read.
`disable_by`	True	If non-zero the component is disabled and data are no longer updated.

`hide_by`	True	If non-zero the component is hidden.

Options	Type	Description
`format`	enum	`number` - If selected the read value is converted to the number either integer or float. Also the parameters `scale`,`offset` and `decimals` are applied during conversion. The color of the text depends on the `color` property if the value is higher than `rangeMax` the `colorAbove` is used likewise if the value is lower than `rangeMin` the `colorBelow` is applied. `time` and `datetime` - The display will convert read value in milliseconds from REX Epoch to the current time or datetime. The property `text_format` is applied in this context where the format of the displayed time can be selected. `text` - Simple show the value in the format as was received. Suitable for short strings.
`text_format`	enum	Select the format of the time string for `time` and `datetime`.
`scale`	float	Scale the read value. Applicable for `number` format only.
`offset`	float	Add some offset to the read value. Applicable for `number` format only.
`decimals`	int	Round the read value to defined number of decimals. Applicable for `number` format only.
`color`	color	Normal font color of the display. Applicable for `number` format only.
`colorAbove`	color	Font color of the display when the read value is above `rangeMax`. Applicable for `number` format only.
`colorBelow`	color	Font color of the display when the read value is below `rangeMin`. Applicable for `number` format only.
`rangeMax`	float	When the read value is above `rangeMax` the `colorAbove` color is applied to the font. Applicable for `number` format only.
`rangeMin`	float	When the read value is below `rangeMin` the `colorBelow` color is applied to the font. Applicable for `number` format only.
`units`	str	Append some string to the end of the displayed value. Usually the name of the units.

DisplayString

XXX DISPLAY STRING XXX

The Display string component reads the value from the target and display it as a text.

It can simply display the read value `format`: `text` or lookup the value in the table `alt` (alternative lookup table) of descriptions.
Note:If the value contains newline sign the text is wrapped.

Data points	Optional	Description
`value`		Datapoint from which the value is read.
`disable_by`	True	If non-zero the component is disabled and data are no longer updated.
`hide_by`	True	If non-zero the component is hidden.
Options	Type	Description
`format`	enum	`text` - show the read value as is. `alt` - Lookup the description for the read value in the `values` list. If not found display the value as is.
`values`	table	List of value:desc pairs where the description is looked up when the value is read.
`showValue`	bool	If True prepend the value in front of the description separated by colon. Example: `0: Status OK`

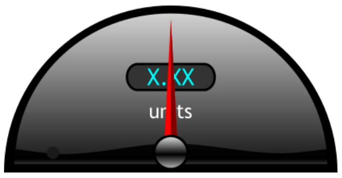
DisplayWithBox

Display component shows value from the target system as a string. It can be either `number`, `date` or `datetime` string depending on the `format` property. Each display can have `units` which are shown at the bottom of the box.

Data points	Optional	Description
`value`		Datapoint from which the value is read.
`disable_by`	True	If non-zero the component is disabled and data are no longer updated.
`hide_by`	True	If non-zero the component is hidden.
Options	Type	Description

`format`	enum	`number` - If selected the read value is converted to the number either integer or float. Also the parameters `scale`,`offset` and `decimals` are applied during conversion. The color of the text depends on the `color` property if the value is higher than `rangeMax` the `colorAbove` is used likewise if the value is lower than `rangeMin` the `colorBelow` is applied. `time` and `datetime` - The display will convert read value in milliseconds from REX Epoch to the current time or datetime.
`scale`	float	Scale the read value. Applicable for `number` format only.
`offset`	float	Add some offset to the read value. Applicable for `number` format only.
`decimals`	int	Round the read value to defined number of decimals. Applicable for `number` format only.
`color`	color	Normal font color of the display. Applicable for `number` format only.
`colorAbove`	color	Font color of the display when the read value is above `rangeMax`. Applicable for `number` format only.
`colorBelow`	color	Font color of the display when the read value is below `rangeMin`. Applicable for `number` format only.
`rangeMax`	float	When the read value is above `rangeMax` the `colorAbove` color is applied to the font. Applicable for `number` format only.
`rangeMin`	float	When the read value is below `rangeMin` the `colorBelow` color is applied to the font. Applicable for `number` format only.
`units`	str	Append some string to the end of the displayed value. Usually the name of the units.

Gauge180



A Gauge displays the read value on the scale using the arm. The range of the scale is limited by `rangeMin` and `rangeMax` parameters. The scale is divided to some main steps using `mainTickStep` parameter and more using `tickStep`. Also the read value is diplayed as a number rounded to `decimals`. If the read value is beyond the limits the Gauge's border changes to `colorOffLimits` color. The Gauge also contain special kind of color scale which is defined by `colorZones` table. For example it can be use for defining the normal, warning and error levels.

Data points	Optional	Description
`value`		Datapoint from which the value is read.
`disable_by`	True	If non-zero the component is disabled and data are no longer updated.
`hide_by`	True	If non-zero the component is hidden.
Options	Type	Description
`rangeMin`	float	The minimum value for the Gauge scale.
`rangeMax`	float	The maximum value for the Gauge scale.
`tickStep`	float	Draw tick every `tickStep` value to the scale.
`mainTickStep`	float	Draw main tick every `mainTickStep` value to the scale with text description.
`decimals`	float	Round the read value to defined number of decimals.
`units`	str	Name of the units or just text description.
`colorZones`	table	Defines the special scale with color segments starting from `startValue` and ending at `endValue` colored by `color`.
`colorOffLimits`	color	Border color when read value is of the range.

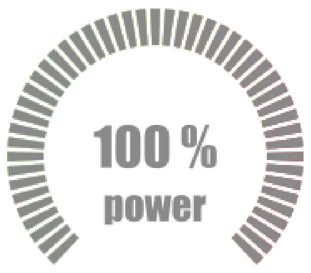
Gauge270

A Gauge displays the read value on the scale using the arm. The range of the scale is limited by `rangeMin` and `rangeMax` parameters. The scale is divided to some main steps using `mainTickStep` parameter and more using `tickStep`. Also the read value is diplayed as a number rounded to `decimals`. If the read value is beyond the limits the Gauge's border changes to `colorOffLimits` color. The Gauge also contain special kind of color scale which is defined by `colorZones` table. For example it can be use for defining the normal, warning and error levels.

Data points	Optional	Description

`value`		Datapoint from which the value is read.
`disable_by`	True	If non-zero the component is disabled and data are no longer updated.
`hide_by`	True	If non-zero the component is hidden.
Options	Type	Description
`rangeMin`	float	The minimum value for the Gauge scale.
`rangeMax`	float	The maximum value for the Gauge scale.
`tickStep`	float	Draw tick every `tickStep` value to the scale.
`mainTickStep`	float	Draw main tick every `mainTickStep` value to the scale with text description.
`decimals`	float	Round the read value to defined number of decimals.
`units`	str	Name of the units or just text description.
`colorZones`	table	Defines the special scale with color segments starting from `startValue` and ending at `endValue` colored by `color`.
`colorOffLimits`	color	Border color when read value is of the range.

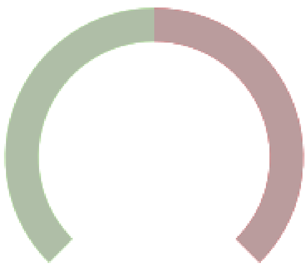
GaugeBars



The GaugeBars component show the current value as a number of highlighted ticks. The value should be between `rangeMin` and `rangeMax`. Also the read value is diplayed as a number rounded to `decimals` with possible `units` string.

Data points	Optional	Description
`value`		Datapoint from which the value is read.
Options	Type	Description
`rangeMin`	float	The minimum value for the Gauge.
`rangeMax`	float	The maximum value for the Gauge.
`decimals`	int	Round the read value to defined number of decimals.
`units`	str	Name of the units or just text description.

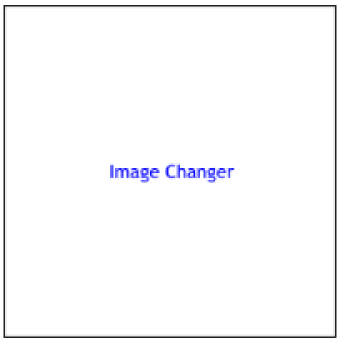
GaugeGradient



The GaugeGradient component show the current value as a fill of the arc. The value should be between `rangeMin` and `rangeMax`. The min value means whole arc is red the max value is whole arc is green.

Data points	Optional	Description
`value`		Datapoint from which the value is read.
Options	Type	Description
`rangeMin`	float	The minimum value for the Gauge.
`rangeMax`	float	The maximum value for the Gauge.

ImageChanger

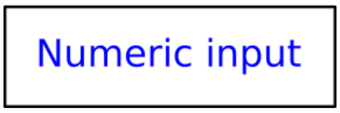


ImageChange component load image (*.png,*.jpg,*.bmp,*.svg etc.) and display it inside its boundaries. The image is refreshed when the `refresh_from` datapoint changes. The `imagePath` should be relative to the root of the webserver. If the `appendKey` property is True the an `imagePath` contain `0` sign which is replaced using the read value.
Example of the `imagePath` '/hmi/images/img-0.png' and value can change from 0 to 10, etc.

Data points	Optional	Description
`refresh_from`		When the value is changed the image is updated.
Options	Type	Description

`imagePath`	str	Relative path from the webserver root where the image is located.
`appendKey`	bool	If True the `0` is replaces with current read value in the `imagePath` property.

Input



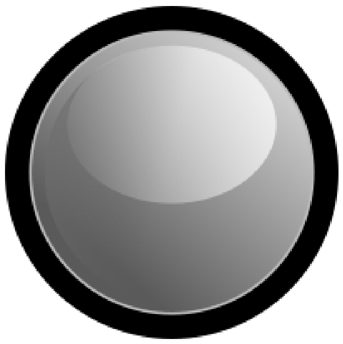
Input component allow to write numbers, texts, dates or times to the target. If the `number` format is selected the value can be converted using `scale` and `offset` adn rounded to number of `decimals`. The Input allows to limit user input to the range defined by `min` and `max` parameters. There is `virtualKeyboard` available for kiosk mode devices. The `date`, `time` and `datetime` format has a special widget to enter the dates. The value from this format is converted to the number of milliseconds from REX epoch.
Note:The value is written to the `value` datapoint, if necessary it can be refresh from different location defined in the `refresh_from` datapoint. It can be used if the value is limited by saturation.

Data points	Optional	Description
`value`		Datapoint where the value is written.
`refresh_from`	True	Optional datapoint where the value is updated.
`disable_by`	True	If non-zero the component is disabled and data are no longer updated.
`hide_by`	True	If non-zero the component is hidden.

Options	Type	Description
`format`	enum	`number` - If selected the value is converted to the number either integer or float. Also the parameters `scale`,`offset` and `decimals` are applied during conversion. `time` and `datetime` - The input will convert time or datetime to the number of milliseconds from REX Epoch. `text` - Simple enter the value in the format as is. Suitable for short strings.
`scale`	float	Scale the value. Applicable for `number` format only.

`fontScale`	float	The Input is using HTML input component which is not binded with SVG. Use this parameter to scale font.
`offset`	float	Add some offset to the value. Applicable for `number` format only.
`decimals`	int	Round the value to defined number of decimals. Applicable for `number` format only.
`min`	float	The minimum value user can enter. Applicable for `number` format only.
`max`	float	The maximum value user can enter. Applicable for `number` format only.
`virtualKeyboard`	bool	If True the virtual keyboard is show for data input.

Led



Boolean state indicator. Empty or Zero mean OFF,FALSE, otherwise ON, TRUE. This can be switched using `reverseMeaning` parameter.

Data points	Optional	Description
`value`		Datapoint from which the value is read.
`disable_by`	True	If non-zero the component is disabled and data are no longer updated.
`hide_by`	True	If non-zero the component is hidden.
Options	Type	Description
`reverseMeaning`	bool	If True the meaning of the value is switched. So the ON state is False and OFF state is True.
`color_true`	color	LED background color when the signal is ON, ENABLED, ...
`color_false`	color	LED background color when the signal is OFF, DISABLED, ...

LedLabel

Boolean state indicator with text label. Empty or Zero mean OFF,FALSE (**OK** text) otherwise ON, TRUE (**NG** text). This can be switched using `reverseMeaning` parameter.

Data points	Optional	Description
`value`		Datapoint from which the LED state is read.
`disable_by`	True	If non-zero the component is disabled and data are no longer updated.
`hide_by`	True	If non-zero the component is hidden.
Options	Type	Description
`reverseMeaning`	bool	If True the meaning of the value is switched. So the ON state is False and OFF state is True.
`label_false`	string	The label for OFF, FALSE state.
`color_false`	color	The color of OFF, FALSE state.
`color_true`	color	The color of ON, TRUE state.
`label_true`	string	The label for ON, TRUE state.

PushOnOff

The PushOnOff button component changes the boolean value. Normally the ON state means TRUE value and OFF mean FALSE value. If needed the parameter `reverseMeaning` switches the written values.
There are three different modes of operation:
`PushButton` when press or touched the ON state is written to the target `value` datapoint. When released the OFF state is written.
`ManualPulse` always write TRUE value after button is pressed, release is ignored.(Suitable for MP block control)
`ToggleButton` when button is pressed the value is toggled (ON -> OFF -> ON).

Data points	Optional	Description
`value`		Datapoint where boolean value is written.
`refresh_from`	True	If set the state of the component is updated based on the value read from this datapoint. Otherwise the value from main datapoint is used.
`disable_by`	True	If non-zero the component is disabled and data are no longer updated.
`hide_by`	True	If non-zero the component is hidden.
Options	**Type**	**Description**
`reverse_meaning`	bool	If True the meaning of the value is switched. So the ON state is False and OFF state is True.
`type`	enum	Changes the type of the Button. `PushButton` when press or touched the ON state is written to the target `value` datapoint. When released the OFF state is written. `ManualPulse` always write TRUE value after button is pressed, release is ignored.(Suitable for MP block control) `ToggleButton` when button is pressed the value is toggled (ON -> OFF -> ON).

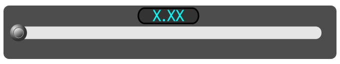
SimpleLogger

Simple logger component write list of separated lines / logs. It can be used in two modes based on the `format` parameter. Each line can be prepend with `timestamp` using local (client) time. Whole log is limited to the max number of `lines`. Parameter `format` have to options:
`text` - Simply log when the datapoint `value` changes.
`alt` - When datapoint `value` changes the description for the value is look up in `texts` table. If not found the value is logged. The value is prepend to description if `showValue` is enabled.

Data points	Optional	Description
`value`		The value from the datapoint will be logged in the logger.
`disable_by`	True	If non-zero the component is disabled and data are no longer updated.

`hide_by`	True	If non-zero the component is hidden.
Options	Type	Description
`format`	enum	The format of the logged message. Either pure `text` or `alt` analog value lookup table.
`texts`	table	The lookup table with `value` and description (`desc`) pairs. Available for `alt format`.
`lines`	number	Maximum number of row / lines in the logger.
`timestamp`	bool	If True the local client time timestamp will be shown.
`showValue`	bool	If True the value is prepend before description from `texts` table.
`css`	object	Custom CSS style for the HTML logger as a JSON object. For example {'background-color':'#00ffff'}

SliderHorizontal

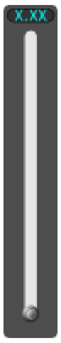


Slider sets integer or floating value to the target. The range of the slider is limited between the `min` and `max` value. Slider move is split to several `steps`. The `value` can be converted using `scale` and `offset` and rounded to chosen number of `decimals`. The `fontScale` applies to the displayed value. Each slider has a `label`.

Data points	Optional	Description
`value`		The main datapoint where numeric value is written.
`refresh_from`	True	If set the state of the component is updated based on the value read from this datapoint. Otherwise the value from main datapoint is used.
`disable_by`	True	If non-zero the component is disabled and data are no longer updated.
`hide_by`	True	If non-zero the component is hidden.
Options	Type	Description
`min`	float	The minimum slider value.
`max`	float	The maximum slider value.
`step`	float	The size of step (increment) when slider moves.
`label`	str	The text description.
`fontScale`	float	The fontScale of the displayed value. If set to 1 the font size is default. The floating number above 1 increase the font size.
`scale`	float	Scale the number when read. The equation is scale*val+offset. The number is converted back when written.

`offset`	float	Offset the number when read. The equation is scale*val+offset. The number is converted back when written.
`decimals`	int	Round the number when read.

SliderVertical

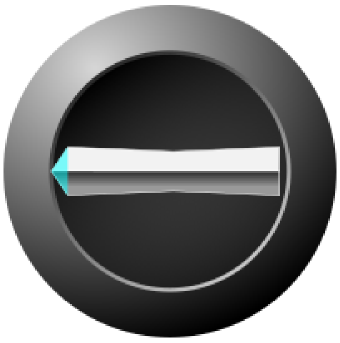


Slider sets integer or floating value to the target. The range of the slider is limited between the `min` and `max` value. Slider move is split to several `steps`. The `value` can be converted using `scale` and `offset` and rounded to chosen number of `decimals`. The `fontScale` applies to the displayed value. Each slider has a `label`.

Data points	Optional	Description
`value`		The main datapoint where numeric value is written.
`refresh_from`	True	If set the state of the component is updated based on the value read from this datapoint. Otherwise the value from main datapoint is used.
`disable_by`	True	If non-zero the component is disabled and data are no longer updated.
`hide_by`	True	If non-zero the component is hidden.
Options	Type	Description
`min`	float	The minimum slider value.
`max`	float	The maximum slider value.
`step`	float	The size of step (increment) when slider moves.
`label`	str	The text description.
`fontScale`	float	The fontScale of the displayed value. If set to 1 the font size is default. The floating number above 1 increase the font size.
`scale`	float	Scale the number when read. The equation is scale*val+offset. The number is converted back when written.
`offset`	float	Offset the number when read. The equation is scale*val+offset. The number is converted back when written.

`decimals`	int	Round the number when read.

Switch



Switch changes the values from the predefine list. Each position write defined value to the target device. Click left or right mouse button to change position clock / counter-clockwise.

Data points	Optional	Description
`value`		Datapoint where value is written.
`refresh_from`	True	If set the state of the component is updated based on the value read from this datapoint. Otherwise the value from main datapoint is used.
`disable_by`	True	If non-zero the component is disabled and data are no longer updated.
`hide_by`	True	If non-zero the component is hidden.
Options	Type	Description
`positions`	table	List of switch positions with values.

SwitchOnOff

Data points	Optional	Description

`value`		Datapoint where boolean value is written.
`refresh_from`	True	If set the state of the component is updated based on the value read from this datapoint. Otherwise the value from main datapoint is used.
`disable_by`	True	If non-zero the component is disabled and data are no longer updated.
`hide_by`	True	If non-zero the component is hidden.
Options	Type	Description
`reverse_meaning`	bool	If True the meaning of the value is switched. So the ON state is False and OFF state is True.

TRND

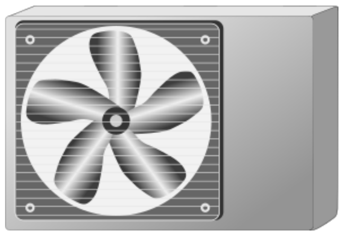
The TRND component is used for displaying the signals from the TRND blocks. Simply just set `TRND` datapoint to any parameter of the TRND block eg. `taskname.TRND:u1`. All signals will be displayed in the trend. Each signal can be labeled using `signals` table.

Data points	Optional	Description
`TRND`		Set the datapoint to any parameter of the TRND block eg. `taskname.TRND:u1` to enable reading of all signals.
`disable_by`	True	If non-zero the component is disabled and data are no longer updated.
`hide_by`	True	If non-zero the component is hidden.
Options	Type	Description
`title`	str	Trend title
`signals`	table	Table of signal labels. Write labels in the same order as you connect the signals to the TRND block.
`bufferSize`	number	The size of the buffer in the client webbrowser. *Note: if the buffer is larger than the buffer on the device the history is lost when the page is refreshed. Note: Large buffer can slow your device especially on slower CPUs like RaspberryPi*

`refreshRate`	number	Custom refresh rate of the trend. Normally the trend is refreshed with the HMI main refresh rate period. If needed each trend can have custom one.
`hideLegend`	bool	Hide the legend of the graph.

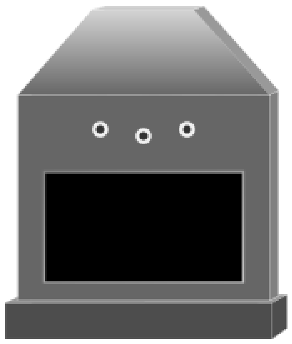
3.2.3 HOME

AirCirculator



Data points	Optional	Description
`POWER`		If non-zero the component is enabled.
Options	Type	Description
`colorRun`	color	Component color when enabled / running.
`colorStop`	color	Component color when disabled / stopped.

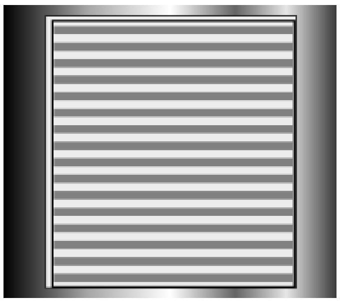
Boiler



The boiler show progress of heating. If the `currentTemp` is less than `setTemp` the flames are visible.

Data points	Optional	Description
`POWER`		If non-zero the component is enabled.
`setTemp`		Boiler temperature set-point.
`currentTemp`		Current temperature.
Options	Type	Description

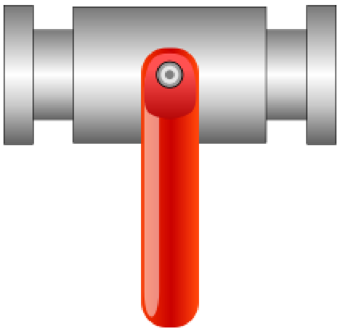
Filter



The filter show if filtration is in progress.

Data points	Optional	Description
`POWER`		If non-zero the component is enabled.
Options	Type	Description
`colorOn`	color	Component color when enabled / running.
`colorOff`	color	Component color when disabled / stopped.

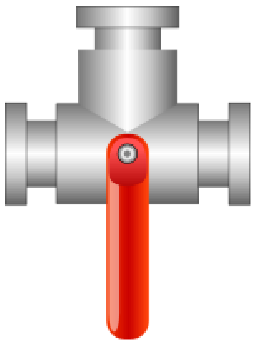
HandleValve



The valve controller. It switch the value when clicked.

Data points	Optional	Description
`FLOW`		State of the valve.
`refresh_from`	True	If set the state is read from this datapoint.
Options	Type	Description

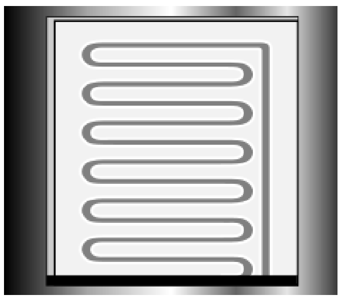
HandleValveT



The three state valve controller. When clicked it toggle two boolean values.

Data points	Optional	Description
`FLOW1`		State of the valve.
`FLOW2`		State of the valve.
`refresh_from`	True	If set the state is read from this datapoint. It expect integer number *0 - unknown, 1 - True-False state, 2 - False-True state.*
Options	Type	Description

Heater



The heater show when the heating is active and what is the current temperature. The color changes from `colorMin` when the value is less than `colorSignalMin` and `colorMax` when the value is greater than `colorSignalMax`.

Data points	Optional	Description
`POWER`		If non-zero the heater is active.
`currentTemp`		Current temperature.
Options	Type	Description
`heatingColor`	color	The color of the heating spiral when active.
`colorMin`	color	The color of the medium when the current temperature is low.

`colorMax`	color	The color of the medium when the current temperature is high.
`colorSignalMin`	float	The minimum level of the value.
`colorSignalMax`	float	The maximum level of the value.

Motor

The motor status indicator.

Data points	Optional	Description
`POWER`		If non-zero the motor is active.
Options	**Type**	**Description**
`colorRun`	color	The color when motor is enabled / running.
`colorStop`	color	The color when motor is disabled / stopped.

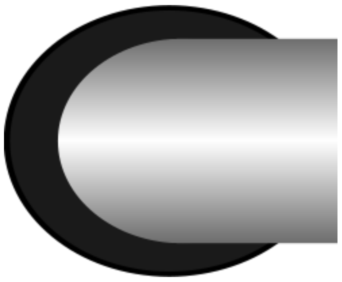
Pipe

The pipe changes the color based on the `COLOR` signal.

Data points	Optional	Description
`COLOR`		The value for color change (e.g. temperature of the pipe)
Options	**Type**	**Description**
`colorMin`	color	The color when the current temperature is bellow min.
`colorMax`	color	The color when the current temperature is above max.

`colorSignalMin`	float	The minimum temperature.
`colorSignalMax`	float	The maximum temperature.

Pipe



The pipe changes the color based on the `COLOR` signal.

Data points	Optional	Description
`COLOR`		The value for color change (e.g. temperature of the pipe)
Options	**Type**	**Description**
`colorMin`	color	The color when the current temperature is bellow min.
`colorMax`	color	The color when the current temperature is above max.
`colorSignalMin`	float	The minimum temperature.
`colorSignalMax`	float	The maximum temperature.

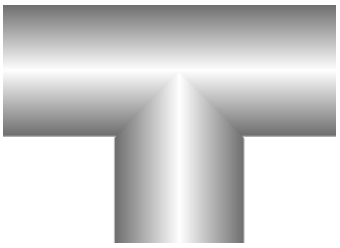
Pipe

The pipe changes the color based on the `COLOR` signal.

Data points	Optional	Description
`COLOR`		The value for color change (e.g. temperature of the pipe)
Options	**Type**	**Description**
`colorMin`	color	The color when the current temperature is bellow min.
`colorMax`	color	The color when the current temperature is above max.

`colorSignalMin`	float	The minimum temperature.
`colorSignalMax`	float	The maximum temperature.

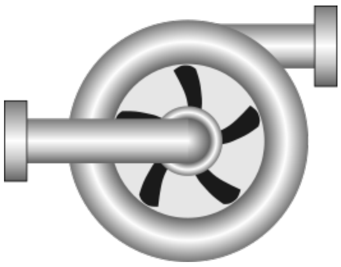
Pipe



The pipe changes the color based on the `COLOR` signal.

Data points	Optional	Description
`COLOR`		The value for color change (e.g. temperature of the pipe)
Options	Type	Description
`colorMin`	color	The color when the current temperature is bellow min.
`colorMax`	color	The color when the current temperature is above max.
`colorSignalMin`	float	The minimum temperature.
`colorSignalMax`	float	The maximum temperature.

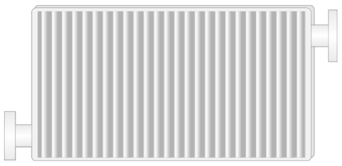
PumpRotation



The pump status indicator.

Data points	Optional	Description
`POWER`		If non-zero the pump is active.
Options	Type	Description
`colorRun`	color	The color when pump is enabled / running.
`colorStop`	color	The color when pump is disabled / stopped.

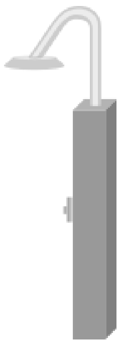
Radiator



The radiator status indicator.

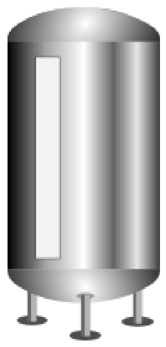
Data points	Optional	Description
`POWER`		If non-zero the heating is active.
`currentTemp`		Current temperature.
Options	**Type**	**Description**
`inOutColor`	color	The color of input and output when heating is active.
`colorMin`	color	The color of the radiator when `currentTemp` is less than `colorSignalMin`.
`colorMax`	color	The color of the radiator when `currentTemp` is more than `colorSignalMax`.
`colorSignalMin`	float	Minimum value of the temperature.
`colorSignalMax`	float	Maximum value of the temperature.

Shower



Options	Type	Description

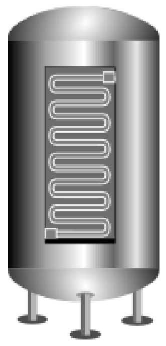
Tank



Show the fill of the tank.

Data points	Optional	Description
`level`		The level of the liquid in the tank.
Options	Type	Description
`capacity`	float	The maximum capacity.
`colorOfLevel`	color	The color of the content.

WaterBoiler



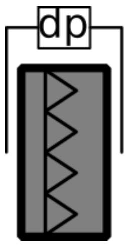
The water boiler component show progress of both heating and cooling. The state depend on `power` value:<br/> 1 - heating
0 - none
-1 - cooling

Data points	Optional	Description
`power`		The state of the water boler: 1 - heating 0 - none -1 - cooling

Options	Type	Description
heatingColor	color	The color of the spiral when boiler is heating.
coolingColor	color	The color of the spiral when boiler is cooling.
colorMax	color	The color when the current temperature is above max.
colorMin	color	The color when the current temperature is bellow min.
colorSignalMax	float	The maximum temperature.
colorSignalMin	float	The minimum temperature.

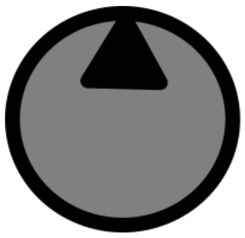
3.2.4 HVAC

AirFilter

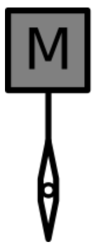


Data points	Optional	Description
DIRTY		If non-zero the Air filter is dirty.
Options	**Type**	**Description**

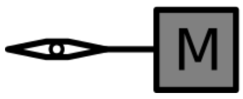
Circulator



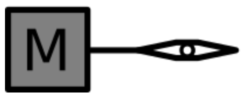
ClosingDamper



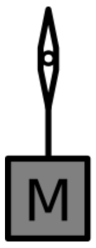
ClosingDamper



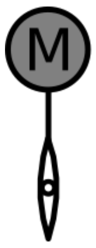
ClosingDamper



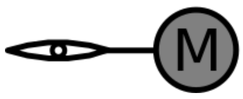
ClosingDamper



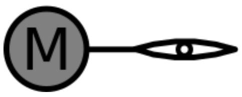
Damper



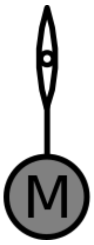
Damper



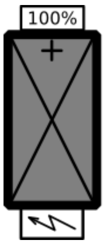
Damper



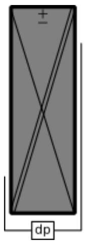
Damper



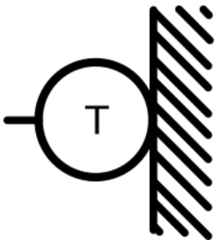
ElAirHeater



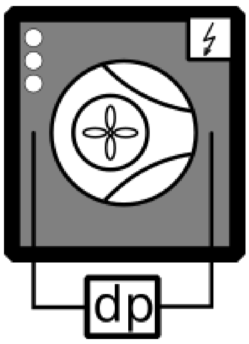
Exchanger



ExternalThermometer

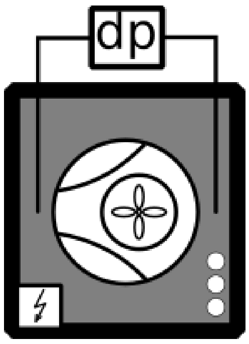


Fan



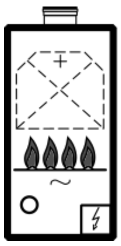
Options	Type	Description

Fan

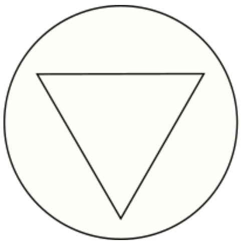


Options	Type	Description

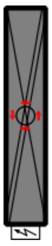
GasBoiler



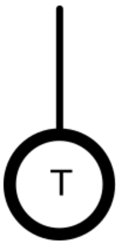
Pump



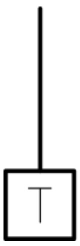
RotaryExchanger



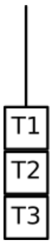
Thermometer



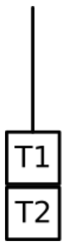
Thermostat



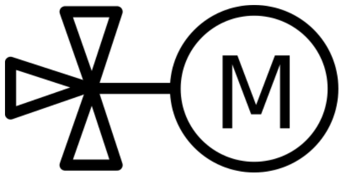
ThermostatThree



ThermostatTwo



TValve



Chapter 4

WebBuDi

4.1 Simple Buttons and Displays on the Web

WebBuDi is an acronym for Web Buttons and Displays, is a simple JavaScript file with several declarative blocks that describe data points which the HMI is connected to and assemble a table in which all the data is presented. It provides a textual interaction with selected signals and is suitable for system developers and integrators or may serve as a fall-back mode HMI for non-standard situations.

WebBuDi is composed from several `rows` (a graphical components with pre-defined function and look) connected to a single item in the control system (specified by an `alias` or a `cstring` property). There are different rows according to the type they are changing (for boolean, numbers, dates, etc.). All `rows` are organized in *sections* (colored blocks which can have a heading). The sections are then organizes in several columns.

The configuration is done using JavaScript objects. See 4.1 for more details.

Example

```
REX.HMI.init = function(){
// Simple PID controller example

// Optional - Add items first
REX.WebBuDi.addItems([
{alias: 'PID_MAN', cstring: 'pidcontrol_control.CNB_MAN:YCN', write: true
    }
]);

// Add WebBuDi section
REX.WebBuDi.addSection({
column: 1,
title: 'User controls',
rows: [
{alias: 'PID_MAN', desc:'Controller mode', type: 'DW', label_false: 'AUT'
    , label_true:'MAN'},
{type:'ES'},
{alias: 'SP_AW', cstring: 'pidcontrol_control.CNR_sp:ycn', desc:'Setpoint
    ', type: 'AW'}
]
});

// Show graph from TRND block
REX.HMI.Graph.addSignal({cstring: 'pidcontrol_control.TRND_PIDU', labels:
    ['Process value','Manipulated variable','Setpoint']});

// Set different target address
// REX.HMI.setTargetUrl('ws://127.0.0.1:8008/rex');

// Set refresh rate (Default: 500 ms)
REX.HMI.setRefreshRate(100);

// Change title of the page
REX.HMI.setTitle('Simple PID controller');
}
```

.WebBuDi : object

- .addSection(opt) => REX.HMI.WebBuDi
- .add() => REX.HMI.WebBuDi
- .addItems(items) => Array.<REX.WS.Item>

REX.WebBuDi.addSection(opt) => REX.HMI.WebBuDi The `addSection` adds new rows / HMI components to the web page.

The section contains all components defined in `rows` array. It can have `title` shown in header. The whole section can be controlled via `disable_by` and `hide_by` item. Finally the section is placed to the `column` (index based).

The function calls can be chained or called via alias `REX.WebBuDi.add()`.

Kind: static method of WebBuDi

Param	Type	Default	Description
opt			The main configuration object
opt.rows	Array.< RowOption>		Definition of all HMI components / rows. See the list for more details
[opt.title]	String		Title of the section shown in the header.
[opt.column]	number	1	Index of the column (starts from 1).
[opt.background_color]	String		Custom background color of the section.
[opt.text_color]	String		Custom text color of the section header
[opt.disable_by]	String | Object		If defined by 'alias' or object `{alias:"XXX", cstring:"XXX", reverse_meaning:false}` the state of the component changes (enabled / disabled).
[opt.hide_by]	String | Object		If defined by 'alias' or object `{alias:"XXX", cstring:"XXX", reverse_meaning:false}` the visibility of the row changes.

Param	Type	Default	Description
[opt.customDivID]	String	"content"	The ID of the element where all the columns / sections will be appended.

Example

```
// Simple HMI for PIDU_Simple_PID_Controller

REX.WebBuDi.addSection({
column: 1,
title: 'Controls',
rows: [
// Digital write
{alias: 'PID_MAN', desc:'Controller mode', type: 'DW', label_false: 'AUT'
    , label_true: 'MAN'},
// Analog write
{alias: 'SP', desc:'Setpoint', cstring: 'pidcontrol_control.CNR_sp:ycn',
    type: 'AW'},
{type: 'ES'}, // Empty space
],
hide_by:"",
disable_by:""
});
```

REX.WebBuDi.add() => REX.HMI.WebBuDi Shortcut for `REX.WebBudi.addSection` function.

Kind: static method of WebBuDi

REX.WebBuDi.addItems(items) => Array.<REX.WS.Item> Add several items at once. This is useful way how to define aliases and use them in various rows

Kind: static method of WebBuDi

Param	Type	Description
items	Array.<Object>	Array of items to register. Shortcut for `REX.HMI.addItems()`.
items.alias	String	
items.cstring	String	
items.write	boolean	Set true if item is writable

Example

```
REX.WebBuDi.addItems([
{alias: 'PID_MAN', cstring: 'pidcontrol_control.CNB_MAN:YCN', write: true
    },
{alias: 'SP_AW', cstring: 'pidcontrol_control.CNR_sp:ycn', write: true},
{alias: 'HV', cstring: 'pidcontrol_control.PIDU:hv'}
]);
```

4.2 Available Rows and Components

- AnalogLookupTable ('ALT')
- Analog Write ('AW')
- Analog Read ('AR')
- Digital Read ('DR')
- Digital Write ('DW')
- Empty Space ('ES')
- Link Button ('LINK')
- Manual Pulse ('MP')
- Push Button ('PB')

General row options - Every row is configured with common and row-specific properties. The following object represents the common part.

Properties

Name	Type	Description
type	String	The type of the component / row (eg. "AR")
alias	alias	UNIQUE identification of the item from control system (eg. 'SP'). Must NOT contain spaces or diacritics
desc	String	Name of the component shown in the description (eg. "Set point"). If the `alias` is not defined the `desc` is used instead. The `desc` is converted to lowercase, spaces are substitute with underscore and all non-ascii letters omitted or replaced with ascii equivalent.
cstring	String	Connection string which contain whole path to the target device (eg. "task.CNR:ycn")
disable_by	String | Object	If defined by 'alias' or object `{alias:"XXX", cstring:"XXX", reverse_meaning:false}` the state of the component changes (enabled / disabled).
hide_by	String | Object	If defined by 'alias' or object `{alias:"XXX", cstring:"XXX", reverse_meaning:false}` the visibility of the row changes.
refresh_from	String	If defined by 'alias' or object `{alias:"XXX", cstring:"XXX"}` the value for the item is read from different location than written. It is aplicable on for WRITE components

```
// All properties
{alias:"SP", desc:"Set point", cstring:"task.CNR:ycn", disable_by:null,
    hide_by:null refresh_from:null}
```

AnalogLookupTable ('ALT') - Select with list of options. Used for user define enums.

Param	Type	Default	Description
opt	RowOption		General configuration for row extended with following properties
[opt.show_key]	boolean	false	Append keys to the list of options
opt.values	Object	{}	List of all values which can be selected. It is a Object with key-value pairs (e.g. `{"1":"Options 1", "2": "Options 2"}`) *

Example

```
{type:'ALT', values:{"1":"Options 1", "2":"Options 2"}, show_key: false}
```

Analog Read ('AR') - Periodical reading of selected value (`date`,`time`,`datetime`,`text`,`number`).
`date`, `time`, `datetime` - Show date calculated from seconds from REX Epoch timestamp.
`text` - Show the value without any transformation (suitable for string values).
`number` - Show number transformed by `scale`, `offset` and round to number of `decimals`.

Param	Type	Default	Description
opt	RowOption		General configuration for row extended with following properties
[opt.format]	boolean	number	One of the following `date`,`time`,`datetime`,`text`,`number`
[opt.scale]	number	1	Scale factor
[opt.offset]	number	0	Offset for the displayed value
[opt.decimals]	number	4	Number of decimals
[opt.convert]	function		If defined, the `format='number'` value is transformed using convert function eg. `convert=function(val){return val+1;}`

Example

```
// Show number rounded to 4 decimal places
{type:'AR'}

// Show date
{type:'AR', format:'date'}

// Show string values
{type:'AR', format:'text'}

// Convert radians to degrees
{type:'AR', format:'number', scale:(Math.PI/180), offset=0, decimals=0}
```

Analog Write ('AW') - Set `date`,`time`,`datetime`,`text` or `number` to the control system.
`date`, `time`, `datetime` - R/W date calculated from seconds from REX Epoch timestamp
`text` - Write value without any transformation (suitable for string values)
`number` - R/W number transformed by `scale`, `offset` and rounded to number of `decimals`.

Param	Type	Default	Description
opt	RowOption		General configuration for row extended with following properties
[opt.format]	boolean	number	One of the following `date`,`time`,`datetime`,`text`,`number`
[opt.scale]	number	1	Scale factor
[opt.offset]	number	0	Offset for the displayed value
[opt.min]	number		Limit the minimum value
[opt.max]	number		Limit the maximum value
[opt.decimals]	number	4	Number of decimals
[opt.set_on_blur]	boolean	false	If set the value is written when the input is blured. Use *ESC* to cancel changes
[opt.convert]	function		If defined, the `format='number'` value is transformed using convert function eg. `convert=function(val){return val+1;}`
[opt.convertW]	function		If defined, the `format='number'` value is transformed before write by convert function eg. `convertW=function(val){return val+1;}`

Example

```
// Change number rounded to 4 decimal places
{type:'AW'}

// Set date
{type:'AW', format:'date'}

// Set string values
{type:'AW', format:'text'}

// Display degrees, read and write as radians
{type:'AW', format:'number', scale:(Math.PI/180), offset=0, decimals=0}

// Default options
{type:'AW', format:'number',
scale:1, offset:0, min: -Number.MAX_VALUE, max: Number.MAX_VALUE,
set_on_blur:false, convert:null, convertW:null}
```

Digital Read ('DR') - Periodical reading of boolean value.

Param	Type	Default	Description
opt	RowOption		General configuration for row extended with following properties
[opt.label_false]	String	"OFF"	Label for the false / off / disable value.
[opt.label_true]	String	"ON"	Label for the true / on / enable value.
[opt.reverse_meaning]	boolean	false	If set the '0' (zero) means enables / ON and '1' disabled / OFF
[opt.color_false]	String	""	Change color of FALSE button when active
[opt.color_true]	String	""	Change color of TRUE button when active

Example

```
{alias: 'DR', desc:'Controller mode', type: 'DR', label_false: 'AUT',
    label_true: 'MAN'}

// All options
{type: 'DR', label_false: 'OFF(0)', label_true: 'ON(1)', reverse_meaning:
    false, color_false:'', color_true:''}
```

Digital Write ('DW') - Set boolean value

Param	Type	Default	Description
opt	RowOption		General configuration for row extended with following properties
[opt.label_false]	String	"OFF"	Label for the false / off / disable value.
[opt.label_true]	String	"ON"	Label for the true / on / enable value.
[opt.reverse_meaning]	boolean	false	If set the '0' (zero) means enables / ON and '1' disabled / OFF
[opt.color_false]	String	""	Change color of FALSE button when active
[opt.color_true]	String	""	Change color of TRUE button when active
[opt.flip]	boolean	false	If set the position of the TRUE/FALSE buttons is flipped.

Example

```
{alias: 'DW', desc:'Controller mode', type: 'DW', label_false: 'AUT',
    label_true: 'MAN'}

// All options
{type: 'DW', label_false: 'OFF(0)', label_true: 'ON(1)', reverse_meaning:
    false, color_false:'', color_true:'', flip:false}
```

Empty Space ('ES') - Creates empty row to fill gaps

Link Button ('LINK') - Create link to different page

Param	Type	Default	Description
opt	Object		LINK configuration object
[opt.target_url]	String	””	URL to which the link leads to.
[opt.desc]	String	””	Description of the link
[opt.label]	String	””	Button label

Example

```
{alias: 'LINK', target_url:'/hmi/index.html', desc: 'Go to index page ...
    ', label: 'To index'}
```

Manual Pulse (’MP’) - Manual Pulse controller (for MP block)

Param	Type	Description
opt	RowOption	General configuration for row apply

Example

```
{alias: 'MP_RUN', cstring:"task.MP_RUN:BSTATE" type: 'MP'}
```

Push Button (’PB’) - One button for setting different values on press and release.

Param	Type	Default	Description
opt	RowOption		General configuration for row extended with following properties
[opt.label_false]	String	”OFF”	Label for the false / off / disable value.
[opt.label_true]	String	”ON”	Label for the true / on / enable value.
[opt.reverse _meaning]	boolean	false	If set the ’0’ (zero) means enables / ON and ’1’ disabled / OFF
[opt.color_false]	String	””	Change color of FALSE button when active
[opt.color_true]	String	””	Change color of TRUE button when active
[opt.value_release]	number | String	0	Set the value which should be set on release (`reverse_meaning do not apply`).
[opt.value_press]	number | String	1	Set he value which should be set on press (`reverse_meaning do not apply`).

Example

```
// Write 'true' on press and 'false' on release
{type: 'PB'}

// Write float value
{type: 'PB', value_release: 0, value_press: 0.1}
```

```

// All options
{type: 'PB', label_false: 'OFF(0)', label_true: 'ON(1)', reverse_meaning:
    false,
color_false:'', color_true:'', flip:false, value_release: 0, value_press:
     1}}s
```

Chapter 5

REX.HMI library

5.1 How to Use the Library

The user can program the HMI over REX.HMI interface. It contain several public methods for reading and writing items from REX targets. To create and build your custom HMI based on the RexHMI library follow these steps:

1. Create project directory with *exec.mdl*. Add the *HMI* block to the executive (see the [1] for more details)
2. Create **.hmi.js* file in *./hmisrc* directory with `REX.HMI.init = function(){}` function inside.
3. Start writing your script

When the script is ready, enable generation of RexHMI using *GenerateRexHMI* parameter in the *HMI* block of the executive. Then run *Compile and Download* function in the RexDraw. The content of the **.hmi.js* file is inserted to the RexHMI template with all the necessary scripts and libraries, also all other content of the *hmisrc* directory is copied to the target *./hmi* folder. The HMI is then downloaded to the target and available from the internal webserver.

The `REX.HMI` interface is described in following sections. Each function contain short example with common parameters.

```
// Common usage
// Append this function to the HTML document or use predefined
    placeholder in the template
REX.HMI.init = function(){
// Change some basic settings

}
```

5.2 Reference Guide for REX.HMI

The main entry-point for all RexHMI visualization. This class exposes all methods necessary for reading and writing variables in control scheme over WebSockets.

Emits: event:time, event:online, event:offline

Properties

Name	Type	Description
REX.HMI.kioskMode	boolean	Set true to enable kioskMode od the HMI
REX.HMI.disableAutoReload	boolean	Set to true if autoreload of the web page should be disabled Autoreload is call when the REX executive or HMI changes. This can be useful if general signals are read to Graph and one do not want to loose history

List of all available functions

- .init()
- .connect() =>Promise
- .disconnect()
- .getTarget(url) =>WSTarget
- .setTargetUrl(url, force) =>Promise
- .setRefreshRate(period)
- .addItem(opt) =>Item
- .addItems(items) =>Array.<Item>
- .removeItem(alias) =>Promise
- .get(alias) =>Item
- .$i(alias) =>Item
- .addGroup(g)
- .removeGroup(g)
- .addTrend(t)
- .removeTrend(t)
- .writeValues(aliases, values) =>Promise

- .setTitle(title)
- .setHeaderTitle(title)
- .showHeartBeatClock(show)
- .useClientTime(use)
- .getItemsEventSynchonizer(aliases, events) =>EventSync

REX.HMI.init() This method can be override by the user. The *init* is called when the websocket connection is opened and one can add own items for RW operations. The method can be called either synchronously or asynchronously with callback. Example of the *REX.HMI.init* function is part of the HTML template

Example

```
// Synchronous version
REX.HMI.init = function(){
REX.HMI.addItem({alias:"SP", cstring:"task.block:param"});
}

// Init with callback
REX.HMI.init = function(done){
done();
}
```

REX.HMI.connect() =>Promise Connect the RexHMI to the target. This function is called automatically

REX.HMI.disconnect() Disconnect all items from the target. Stop reading and dispose connections to all targets.

REX.HMI.getTarget(url) =>WSTarget Return REX target base on the given URL. If URL is null (the most common case) then it returns the default target.

Param	Type	Description
url	String	URL of the requested target

Example

```
// Retrieve version of the default target
REX.HMI.getTarget().getVersion().then((data)=>{console.log(data)})
```

REX.HMI.setTargetUrl(url, force) =>Promise Sets the new default target URL. When the page is served from server (not localhost) and the `location.hostname` is set the *setTargetURL* function sets NULL to use default targetURL. So the target connects to the location which the page is served from.

This behaviour can be changed by setting the `force` parameter tu true. Than the 'url' will be used on any occasion.

Param	Description
url	URL of the target
force	set true if the URL should be set even run from server with hostname

Example

```
// The most common usage
// If run locally from file:// connect to IP, when uploaded to server (
    RexCore)
// then connect to location.hostname
REX.HMI.setTargetUrl("ws://192.168.0.100:8008");

// Always connect to the localhost
REX.HMI.setTargetUrl("ws://127.0.0.1:8008", true);

// Always use secure WebSocket
REX.HMI.setTargetUrl("wss://192.168.0.100:8008", true);
```

REX.HMI.setRefreshRate(period) Change the default refresh rate (how fast the data from RexCore will be read)

Param	Type	Default	Description
period	number	500	New refresh period [ms]

Example

```
// Change default refresh period to 1000 ms (1s)
REX.HMI.setRefreshRate(1000);
```

REX.HMI.addItem(opt) =>Item Register new *Item* defined by `alias` and `cstring` for periodical reading and asynchronous writing.

Returns: Item - - Registered item

Param	Type	Description
opt	Object	Main configuration object
opt.alias	string	Alias for the connection string
opt.cstring	string	Connection string of the signal from REX
[opt.url]	string	URL of the target, if NULL the default is used

Param	Type	Description
[opt.period]	number	Item refresh period [ms]
[opt.writeCString]	string	If defined the value of the item will be written to this *cstring*

Example

```
// The most common usage
var sp = REX.HMI.addItem({alias:"SP", cstring:"task.block:param"});
sp.on('change',(data)=>{console.log(data)});

// Different location of reading and writing (eg. Write before saturation
   and read after)
REX.HMI.addItem({alias:"SP", cstring:"task.SAT:y", writeCString:"task.CNR
   :ycn"});
```

REX.HMI.addItems(items) =>Array.<Item> Add several Items at once. See `HMI#addItem` for more details

Returns: Array.<Item> - - Return array of added items

Param	Type	Description
items	Array. <Object>	An array of items

REX.HMI.removeItem(alias) =>Promise Remove an Item based on its `alias`

Param	Description
alias	Items `alias` used during registration

REX.HMI.get(alias) =>Item Find Item using its `alias`

Param	Description
alias	Items `alias` used during registration

REX.HMI.$i(alias) =>Item Find Item using its `alias`

Param	Description
alias	Items `alias` used during registration

REX.HMI.addGroup(g) Register custom group of items for R/W operations

Param	Type	Description
g	REX.WS.Group	Group for registering

Example

```
// Create group see REX.WS.Group
var g = new REX.WS.Group({id:"group1", period:100, url:""});

// Add some items
g.addItem(new REX.WS.Item({id:"ITEM-1", cstring:"task.block:param", url:"
    "}));

// Register events
g.on('read',(data)=>{console.log(data)});

// Register group
REX.HMI.addGroup(g);
```

REX.HMI.removeGroup(g) Unregister custom group

Param	Type	Description
g	REX.WS.Group	Instance of group which will be unregistered

REX.HMI.addTrend(t) Unlike REX.WS.Group the Trend reads data from TRND* blocks. These blocks store several signal with buffered data. Once registered the user can process the data from TRND* blocks.

Param	Type	Description
t	REX.WS.Trend	Trend which will be registered

Example

```
// Create new trend
var trend = REX.WS.Trend({cstring:"task.TRND", id:"TRND-1", period:500,
    readWholeBuffer:true});

// Register
REX.HMI.addTrend(trend);
// Add event handlers
trend.on('read',(data)=>{console.log(data)});
```

REX.HMI.removeTrend(t) Unregister trend

Param	Type
t	REX.WS.Trend

REX.HMI.writeValues(aliases, values) =>Promise Write one or several values to the control system. Using already registered items (aliases).

Param	Type	Description
aliases	Array.<String>	An array of already registered aliases
values	Array	An array of values to be written

Example

```
// Register some items
REX.HMI.addItems([{alias:"A1",cstring:"task.A1:ycn"},{alias:"A2",cstring:
    "task.A2:ycn"}]);

// Write values
REX.HMI.writeValues(["A1","A2"],[0.5, 0.7]);
```

REX.HMI.setTitle(title) Change bot title in header and title of the webpage

Param	Type	Description
title	String	New title for header and webpage

Example

```
REX.HMI.setTitle("My HMI");
```

REX.HMI.setHeaderTitle(title) Change the header title only

Param	Type	Description
title	String	New title for the header

REX.HMI.showHeartBeatClock(show) If true, the template will display CLOCK in upper right corner of the main screen. When the update of the time stops, the default target is disconnected and the HMI is not updated

Param	Type	Description
show	Boolean	True to show the clock

REX.HMI.useClientTime(use) Set to True if the time should be displayed in client time not target time. When the target is not able to synchronize with some time server is it possible to use and display times in a client time

Param	Type	Description
use	Boolean	True to use client time instead of target one

REX.HMI.getItemsEventSynchonizer(aliases, events) =>EventSync Return an EventSync object which emits events when all registered items have emitted the same event.

Param	Type	Description
aliases	Array | String	Array of item aliases or one alias as a string
events	Array | String	Array of events which will be monitored

Example

```
/ Register some items
REX.HMI.addItems([{alias:"A1",cstring:"task.A1:ycn"},{alias:"A2",cstring:
    "task.A2:ycn"}]);

var sync = REX.HMI.getItemsEventSynchonizer(["A1","A2"],["read"]);

sync.on("read",()=>{console.log("All read events emitted")});
```

5.3 Reference Guide for REX.HMI.Graph

Time-based graph component which is shown on the bottom of the web page. Graph can read arbitrary signal connected via `alias` and `cstring` or all signals from TRND* blocks. The Graph is shown when first signal is added over `addSignal` or `addTrend` function.

The REX.HMI.Graph has following functions:

- .resume()
- .pause()
- .show()
- .hide()
- .addSignal(opt)
- .addTrend(opt)
- .setSize(size)
- .setRefreshRate(period)
- .hideLegend()

REX.HMI.Graph.resume() Resume redrawing the graph
Example

```
REX.HMI.Graph.resume()
```

REX.HMI.Graph.pause() Pause redrawing the graph
Example

```
REX.HMI.Graph.pause()
```

REX.HMI.Graph.show() Show graph
Example

```
REX.HMI.Graph.show()
```

REX.HMI.Graph.hide() Hide graph
Example

```
REX.HMI.Graph.hide()
```

REX.HMI.Graph.addSignal(opt) Add arbitrary signal from the REX executive to the trend. Warning! The data are stored inside the web page once refreshed all the data will be lost.

Param	Type	Description
opt	Object	Main configuration object
opt.alias	string	Alias for the connection
[opt.cstring]	string	Connection string of the signal from REX
[opt.desc]	Array.<string>	Signal's description
[opt.period]	number	Signal refresh period [ms]

Example

```
REX.HMI.Graph.addSignal({alias:"Signal-1", cstring:"task.CNR:ycn", desc:"
    Set point", period:500});
```

REX.HMI.Graph.addTrend(opt) Add signals from TRND* blocks to the common graph in HMI

Param	Type	Description
opt	Object	Main configuration object
opt.cstring	string	Connection string for TRND* block

Param	Type	Description
[opt.labels]	Array.<string>	Array of signal labels
[opt.period]	number	Graph redraw period

Example

```
REX.HMI.Graph.addTrend({cstring:"task.TRND", labels:["signal-1","signal-2
   "], period:500});
```

REX.HMI.Graph.setSize(size) Change size of the graph. The size is in percent of the page.

Param	Type	Default	Description
size	number	0.39	Size of the graph in percents <0;1>

Example

```
REX.HMI.Graph.setSize(0.39); // Default
```

REX.HMI.Graph.setRefreshRate(period) Change refresh rate of all signals and trends.

Param	Description
period	Refresh period [ms]

Example

```
REX.HMI.Graph.setRefreshRate(250);
```

REX.HMI.Graph.hideLegend() Hide legend of the graph
Example

```
REX.HMI.Graph.hideLegend();
```

Bibliography

[1] REX Controls s.r.o.. *Function blocks of the REX Control System – reference manual*, 2017.

Documentation reference number: 8614