Leonis3.5 Configurator instruction

Table of contents

1 Common....2
1.1 Brief description....2
1.2 Project creation overview....3
1.3 Connection to Leonis3.5....5
1.4 Main device settings....6
1.5 User variables....7
2 UI design....9
2.1 Editor functions....9
2.2 Screens....10
2.3 Text widgets....11
2.4 Supported special symbols....14
2.5 Icon widgets....14
2.6 Button, slider and switch widgets....16
3 Logic programming....16
3.1 Introduction....16
3.2 Analog I/O....19
3.3 Buttons....19
3.4 Compare....20
3.5 Control....20
3.6 Device....22
3.7 Digital I/O....23
3.8 Logic....23
3.9 Math....26
3.10 Modbus RTU....26
3.11 Regulation....28
3.12 Script....29
3.13 Time....31
3.14 Tools....34
3.15 Values....37
4 Simulation....38
5 Work with Leonis3.5 device....40
5.1 Firmware update and checking software and configuration version....40
5.2 EEPROM erase....42
5.3 Read / write device date time....42
5.4 Read device configuration....42
5.5 CO2 sensor calibration....43
6 Leonis3.5 device information....44
6.1 Settings screens....44

1 Common

1.1 Brief description

Leonis3.5 is a Modbus RTU master and slave room device with temperature and humidity sensors, optional CO2 sensor, 2*RS485 ports (master and slave), RTC with backup battery, USB device port and 3.5" touch screen. Modbus registers mapping with description can be found in Leonis3.5_registermapping.xlsx document.

There can be the following optional I/O: 2*DI, 1*PT1000 or 1*0-10V input, 2*OC or 2*0-10V output.

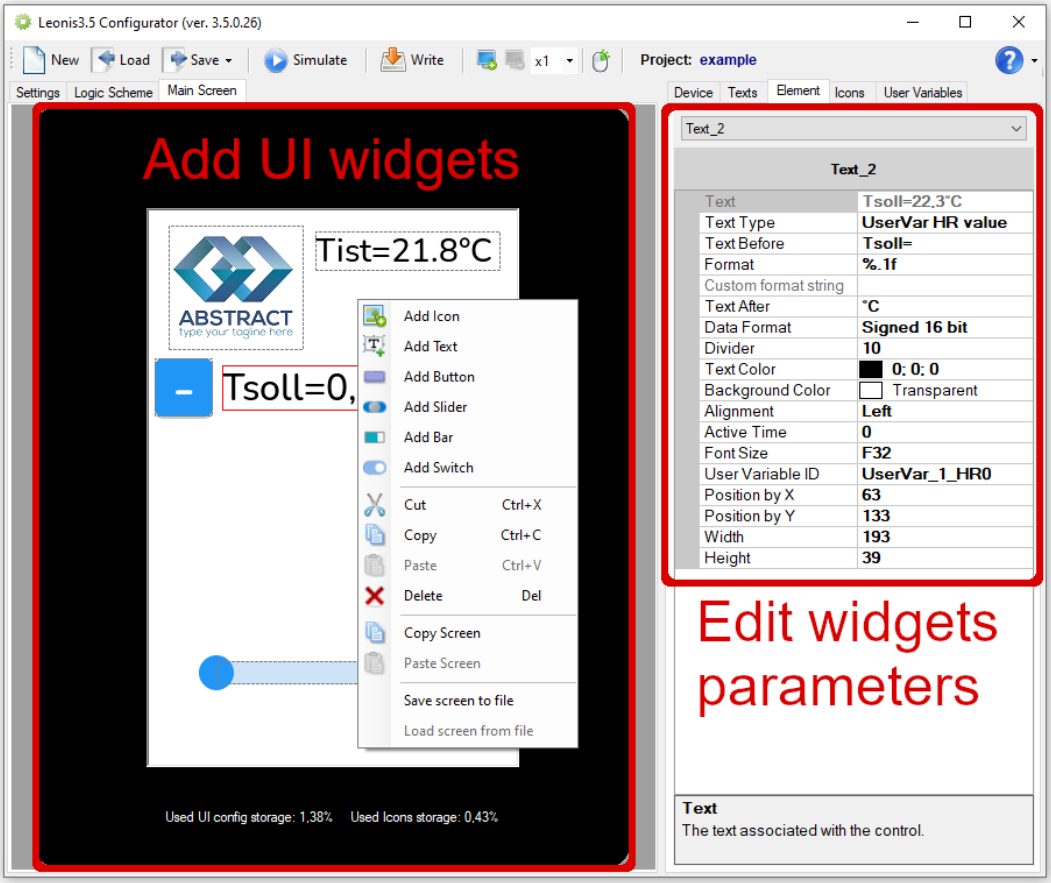
Leonis3.5 also acts as a configurable regulator and can be used for different regulation applications (temperature regulation, light control, jalousie control, fan coil regulator, etc).

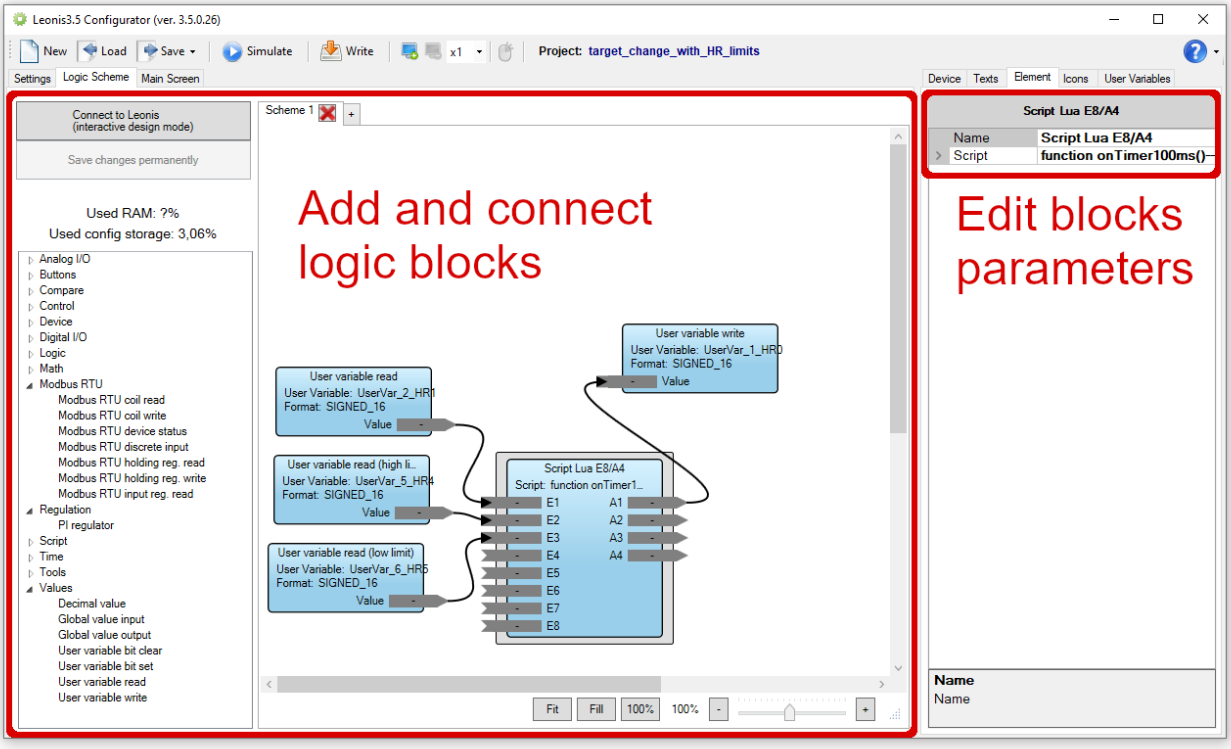
Leonis3.5 configurator PC tool can be used for creating configurations for Leonis3.5 device. User can add widgets to screen, configure device settings and define device regulation behaviour. Each configuration includes common device settings, screens / widgets and regulator logic. Configuration is stored in *.lmcfg file and icons in *.bfs file.

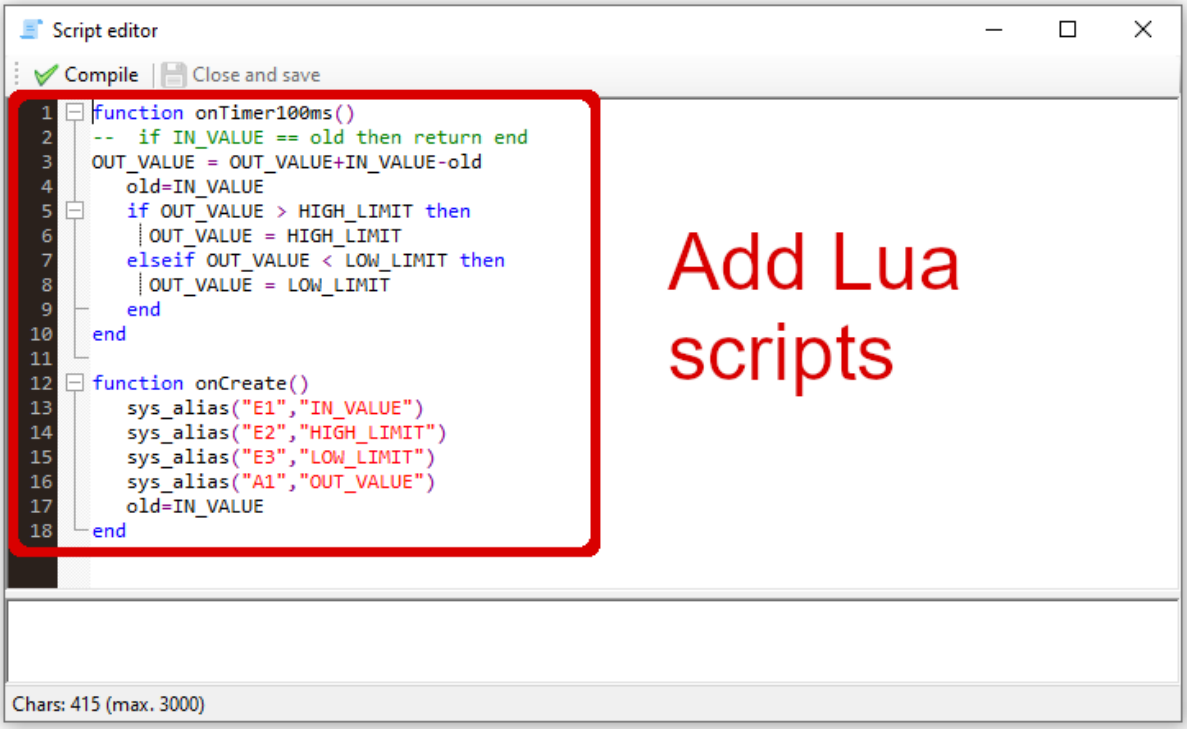
Regulator logic can be created in a PC tool with visual programming by adding and connecting basic blocks. Also Lua language scripting is supported for custom blocks behaviour creation.

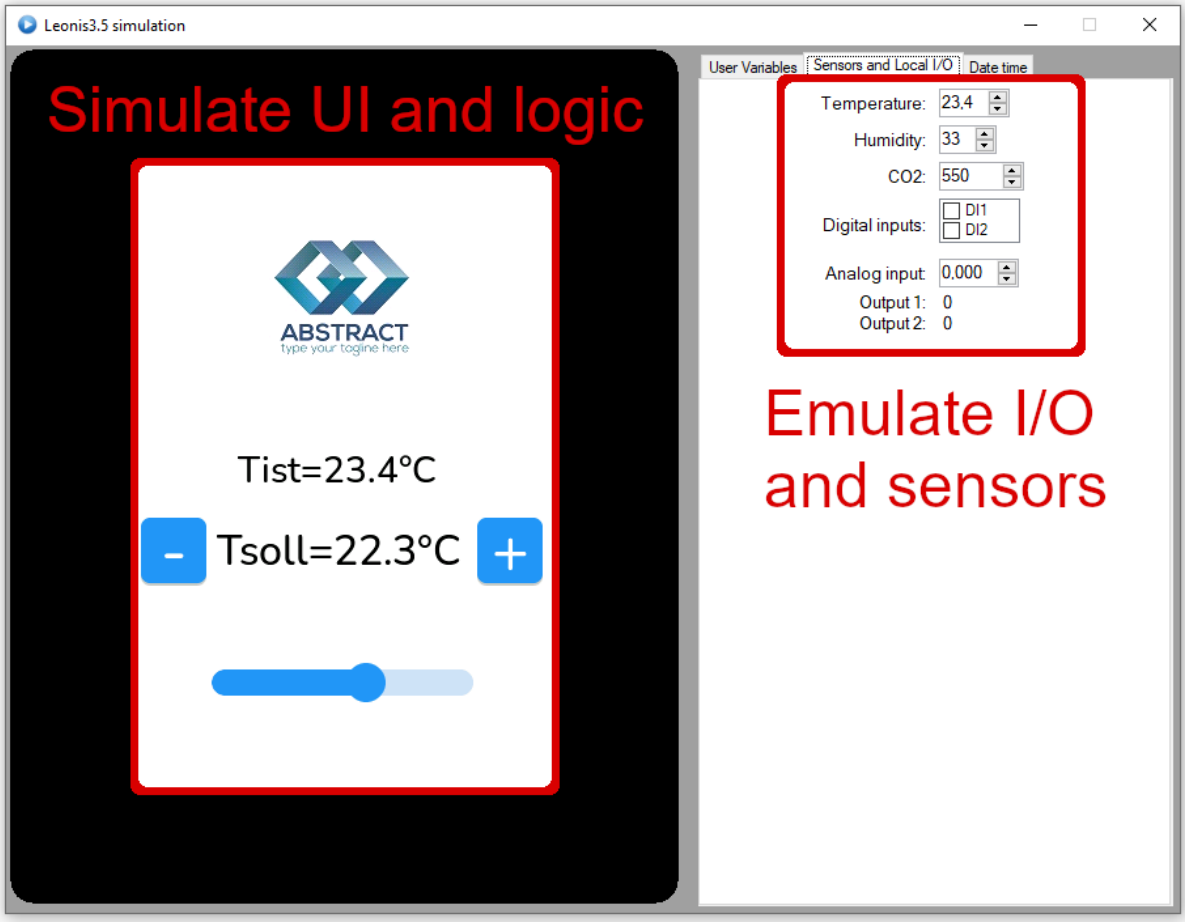
The PC tool is able to load configuration to the device via slave RS485 port (with USB-RS485 converter) or USB port (virtual COM port). USB port is much faster and there also slave port parameter can be changed.

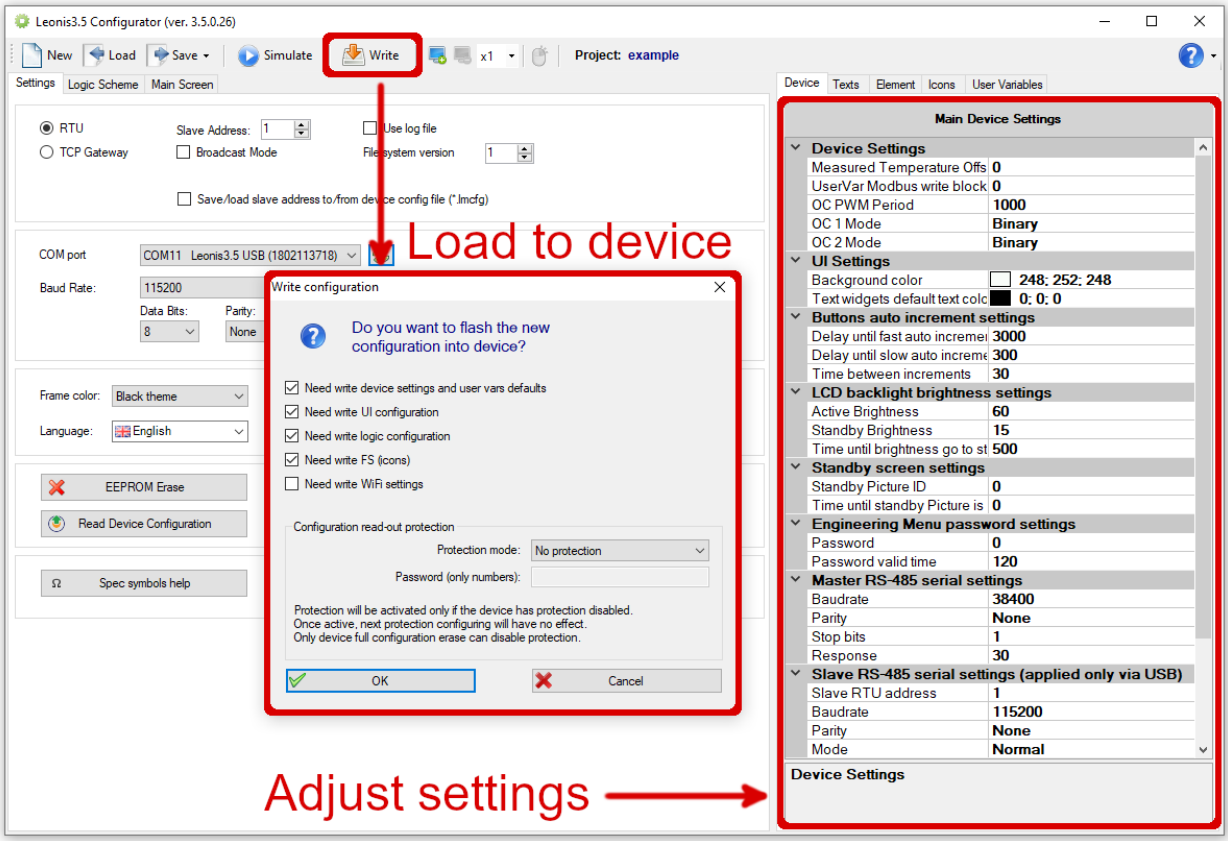
1.2 Project creation overview





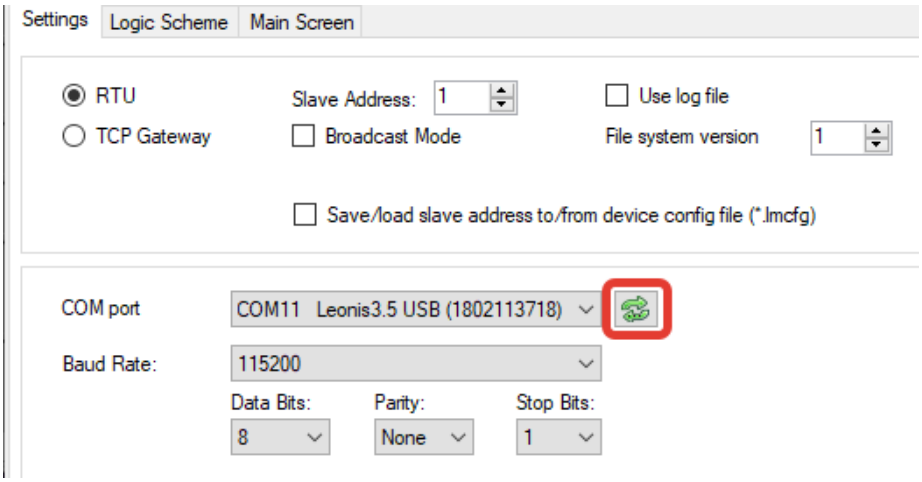

Script editor
Compile
Close and save
function onTimer100ms()
-- if IN_VALUE == old then return end
OUT_VALUE = OUT_VALUE+IN_VALUE-old
old=IN_VALUE
if OUT_VALUE > HIGH_LIMIT then
OUT_VALUE = HIGH_LIMIT
elseif OUT_VALUE < LOW_LIMIT then
OUT_VALUE = LOW_LIMIT
end
end
function onCreate()
sys_alias("E1","IN_VALUE")
sys_alias("E2","HIGH_LIMIT")
sys_alias("E3","LOW_LIMIT")
sys_alias("A1","OUT_VALUE")
old=IN_VALUE
end
Add Lua scripts
Chars: 415 (max. 3000)


Leonis3.5 simulation
Simulate UI and logic
ABSTRACT
type your tagline here
Tist=23.4°C
-
Tsoll=22.3°C
+
User Variables
Sensors and Local I/O
Date time
Temperature: 23,4
Humidity: 33
CO2: 550
Digital inputs: DI1 DI2
Analog input: 0,000
Output 1: 0
Output 2: 0
Emulate I/O and sensors



1.3 Connection to Leonis3.5

There are two possibilities of connection to device: RS485 slave port via USB-RS485 adapter or direct USB connection with PC. In case of direct USB connection, device will create virtual COM port, which can be found on Settings tab by pressing Refresh button.



In case of USB connection Slave address and serial port parameters are not important.

1.4 Main device settings

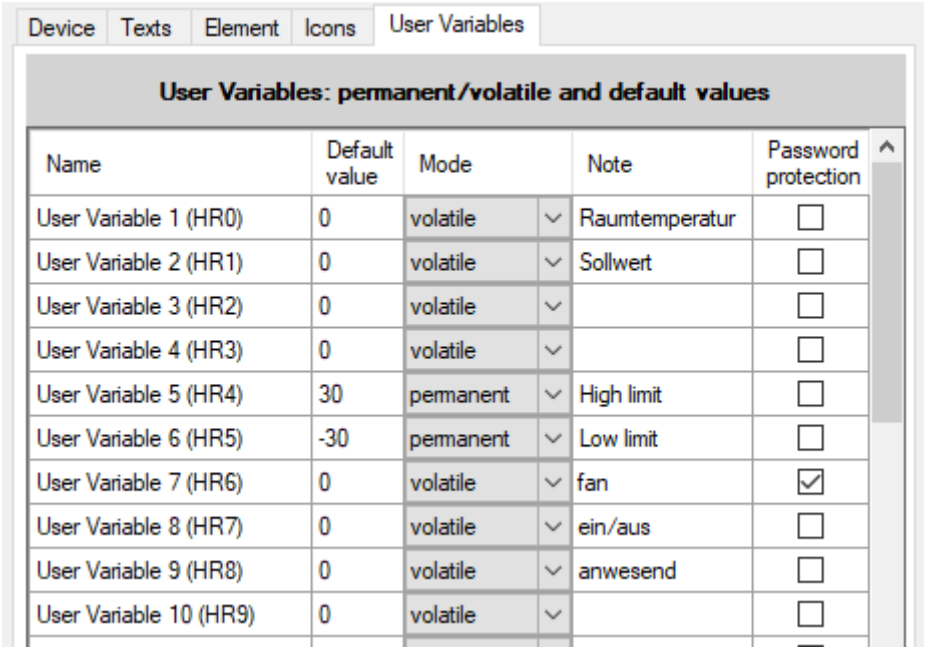
There are a lot of device settings available for configuring on Device tab, like Active / Standby brightness, Button widgets auto increment settings, RS-485 Master and Slave ports serial settings and others.
Slave RS-485 serial settings will be applied by device only if connection is done via USB.

Main Device Settings	
Device Settings	
Measured Temperature Offset	**0**
UserVar Modbus write block time after change by widget	**0**
OC PWM Period	**1000**
OC 1 Mode	**Binary**
OC 2 Mode	**Binary**
UI Settings	
Background color	**248; 252; 248**
Text widgets default text color	**0; 0; 0**
Buttons auto increment settings	
Delay until fast auto increment start	**3000**
Delay until slow auto increment start	**300**
Time between increments	**30**
LCD backlight brightness settings	
Active Brightness	**60**
Standby Brightness	**15**
Time until brightness go to standby value	**500**
Standby screen settings	
Standby Picture ID	**0**
Time until standby Picture is shown	**0**
Engineering Menu password settings	
Password	**0**
Password valid time	**120**
Master RS-485 serial settings	
Baudrate	**38400**
Parity	**None**
Stop bits	**1**
Response	**30**
Slave RS-485 serial settings (applied only via USB)	
Slave RTU address	**1**
Baudrate	**115200**
Parity	**None**
Mode	**Normal**

1.5 User variables

Modbus registers from special range (HR0..HR99) are called user variables. These registers can be linked to widgets using the User Variable ID parameter. User variables allow access to UI widgets via Modbus RTU protocol. Also with user variables it is possible to define reaction on widgets events, for example, change target temperature on the screen with a slider. To achieve this, the same user variable should be assigned for both Slider and Text widget. At the same time it is possible to perform remote read of target temperature value using Modbus RTU read requests of corresponding user variable holding register.

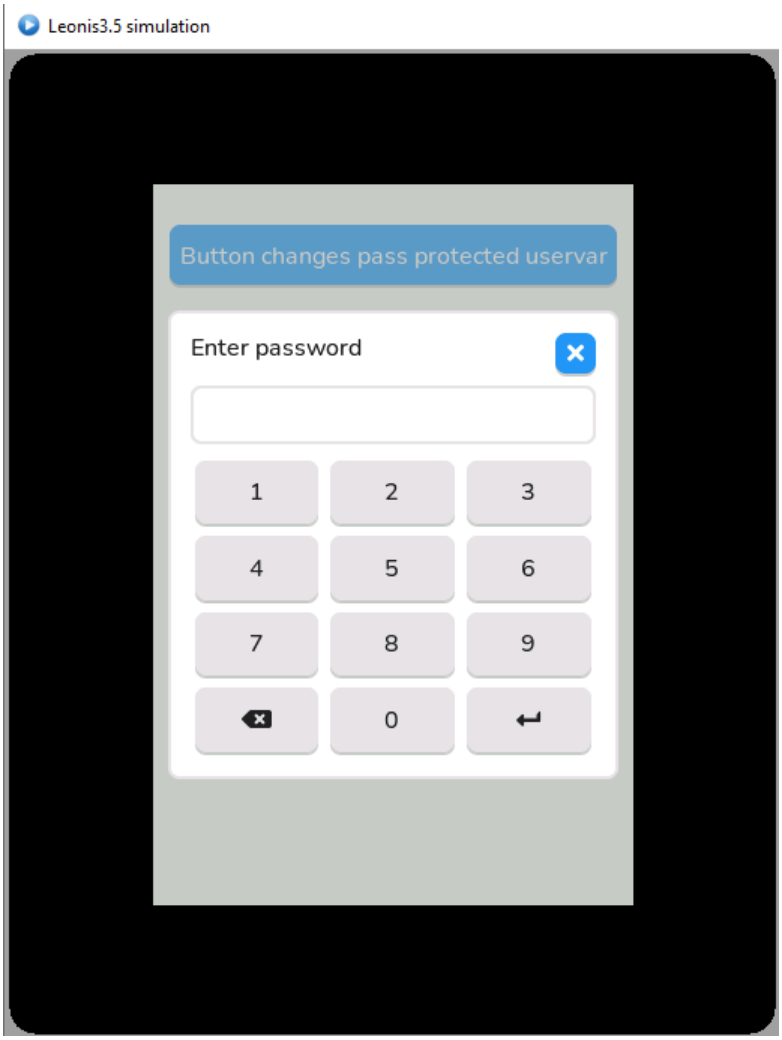
Each user variable can be configured as volatile or permanent. Unlike volatile, permanent mode preserves value during Leonis3.5 device power off periods. So for a target temperature example, it can be useful to set permanent mode for the correspondent user variable to avoid loss of selected temperature in case of power off/on cycles. In case of volatile mode, the user variable will be set to default value after each power on. Default values can be assigned to each user variable on the User Variables tab.



Note text (“Note” column) is very helpful for user variables distinguishing, it is can be a good idea to comment all used variables.

It is also possible to protect user variable value change with password. This function can be useful, for example, when a user variable defines the mode of operation of some remote device and only authorized staff can change this mode. With password protection enabled, password enter screen will appear when the user will try to change a variable using a button or slider. In case of successful password enter, user variable change will be allowed during time defined in Password valid time device parameter with prolongation on

each new screen touch. To set a password, the engineering menu Password parameter from the main device settings tab should be used (0 – protection off).



Last user variable (HR99) has a special purpose: get / set active screen (0 - first, 1 - second ...).

There is an important feature regarding Modbus RTU write requests to user variables linked with button / slider / switch. If **UserVar Modbus write block time after change by widget** device parameter has non-zero value, after the user variable changes by touch, Modbus write to corresponding register will be ignored for defined time. This is needed for easy several Leonis3.5 devices synchronization while working with PLC and controlling the same thing, e.g. for dimming same lamp. While user change lamp brightness with slider on one Leonis3.5, PLC read brightness value from all devices, control lamp with actual value and write this value to all devices including user one. Write ignore period allows user to comfortably select needed value without overwriting with older value from PLC.

2 UI design

2.1 Editor functions

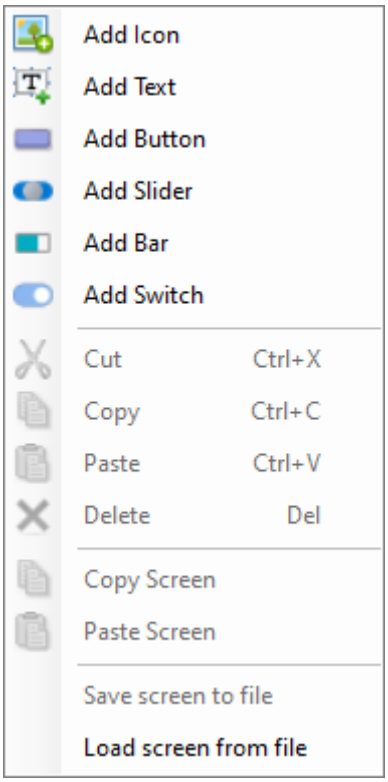
Leonis3.5 configurator PC tool is WYSIWYG editor (What You See Is What You Get), so UI elements (widgets) in design view app have exactly the same appearance as on real Leonis3.5 device.

UI configuration stored in *.lmcfg files (together with device settings and logic configuration). Icons images data stored in *.bfs files.

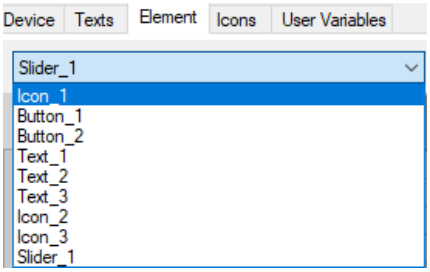
Different design area scale modes are supported (x1, x2, x3).



Right click on widget pop-up menu allows to change widget Z-order (move foreground/background) and also cut, copy or delete widget or several widgets in case of group selection. It is also possible to copy/paste between two PC configurator instances using Windows clipboard. Load/Save separate screen from/to *.lmscr file is supported.



For widget selection, Element tab drop-down list can be used or keyboard: TAB (next widget) / SHIFT+TAB (previous widget). This can be useful when some widget is placed under another, e.g. for virtual button and text widget under.



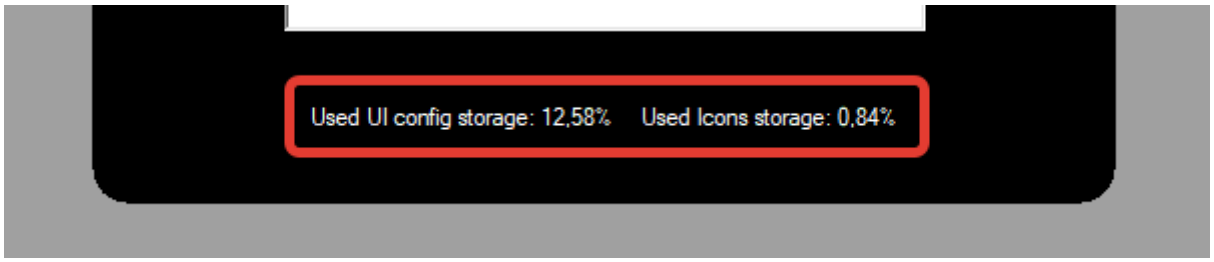
Also keyboard arrows (↑ ↓ → ←) can be used for precise widgets positioning.

Used config storage displayed at the bottom of the frame indicates in percent how large is current UI configuration compared to max possible and supported by Leonis3.5.

UI configuration includes: screens with widgets and text strings.

UI configuration not includes: logic configuration and icons images.

Also Iconts storage utilization displayed.



2.2 Screens

There is 1 main screen and up to 19 additional (auxiliary) screens. So it is possible to have overall 20 screens. To add or delete additional screens, the Add Aux Screen or Delete Aux Screen button from the main toolbar should be used accordingly.



Switching screens can be performed by changing the last user variable (User Variable 100, HR99) via Modbus or two Button widgets (go to previous / next screen) assigned to this variable configured with increment/decrement button type. Also touch gestures (horizontal swipes) are supported for changing screens.
There is a possibility to copy an already created widgets to another screen. Steps for screen copy:

1. Right click on source screen -> Copy Screen.

2. Select the destination screen tab (press Add Aux Screen button to create a new one if needed).

3. Right click on destination screen -> Paste Screen.

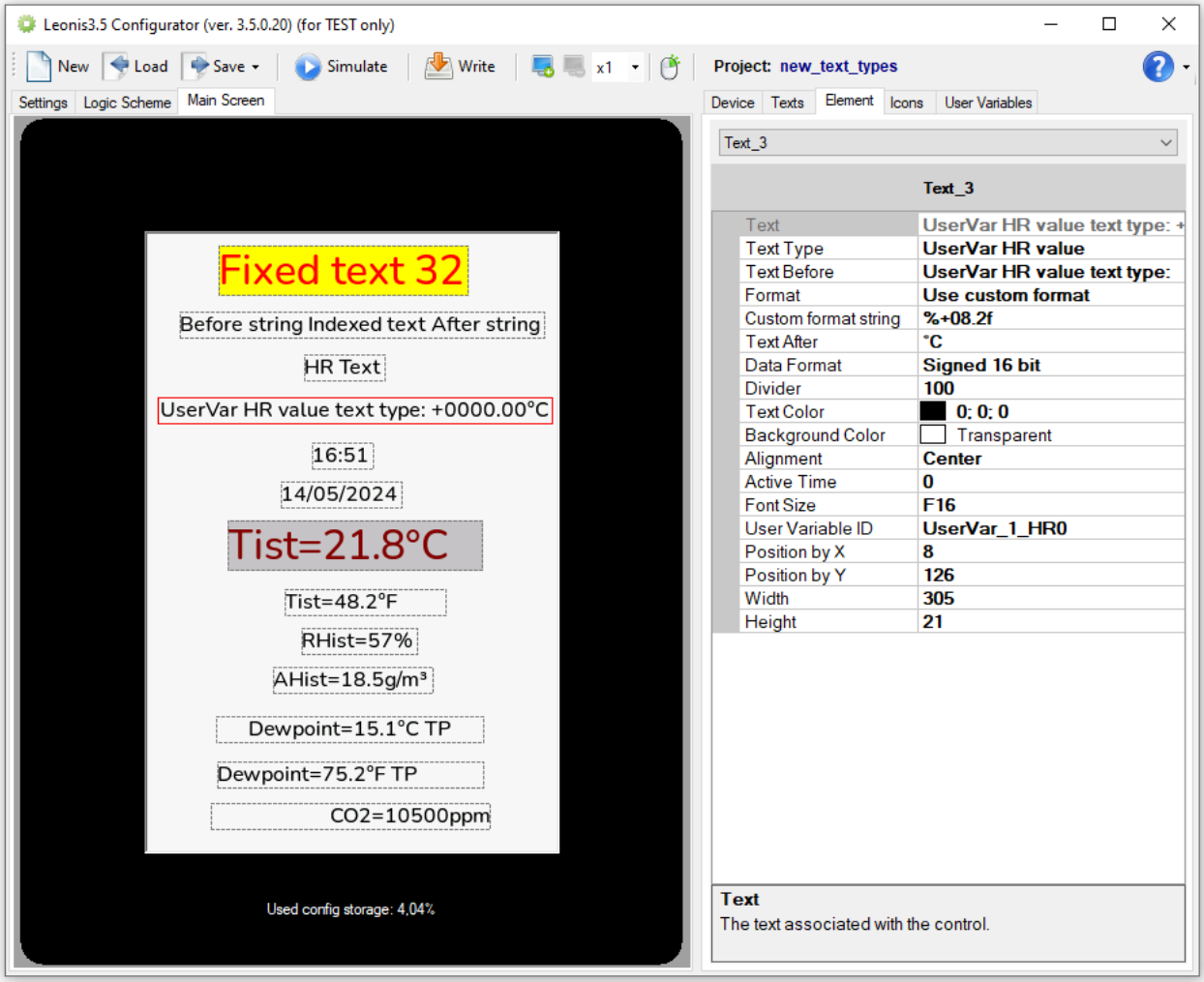
It is also possible to copy / paste a single or group of widgets using the right-click popup menu.

2.3 Text widgets

To display some value or text on the screen Text widget should be used.

Important parameter is User Variable ID. User Variable allows linking Text widget to one of Modbus holding registers from special range (HR0..HR99). Registers from this range are called user variables. Change of linked register value will cause change of Text widget value on the screen (exact behaviour depends from type of Text widget). Availability of User Variable ID parameter depends on Text type. Some Text types are intended to display internal measured values, for example temperature, humidity. User Variable ID parameter is unavailable for them.

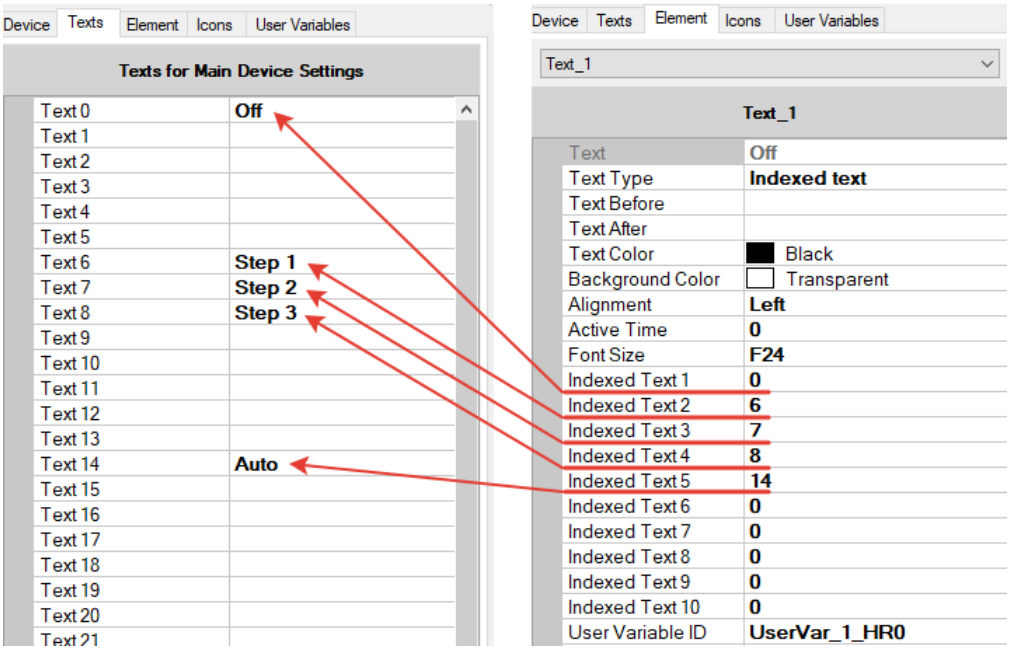
Text widget has a text type parameter that defines its behaviour. E.g. UserVar HR value type allows to display value from one of the user variables using predefined or custom formatting (Format / Custom format string parameters). Text before / Text after parameters can be used to add some text before / after value. Data format parameter defines internal interpretation of value, e.g. signed 16-bit or unsigned 16-bit. Divider parameter allows value scaling by dividing it to 1 / 10 / 100 / 1000.



User can store up to 40 predefined strings for each screen. Such strings can be used only for Indexed text type of Text widgets. With Indexed text type it is possible to assign predefined strings to a special set of string indexes (up to 10) to be able to select an active string from a set with Modbus RTU requests or other widgets like Buttons.

Predefined strings

Indexed type Text widget parameters

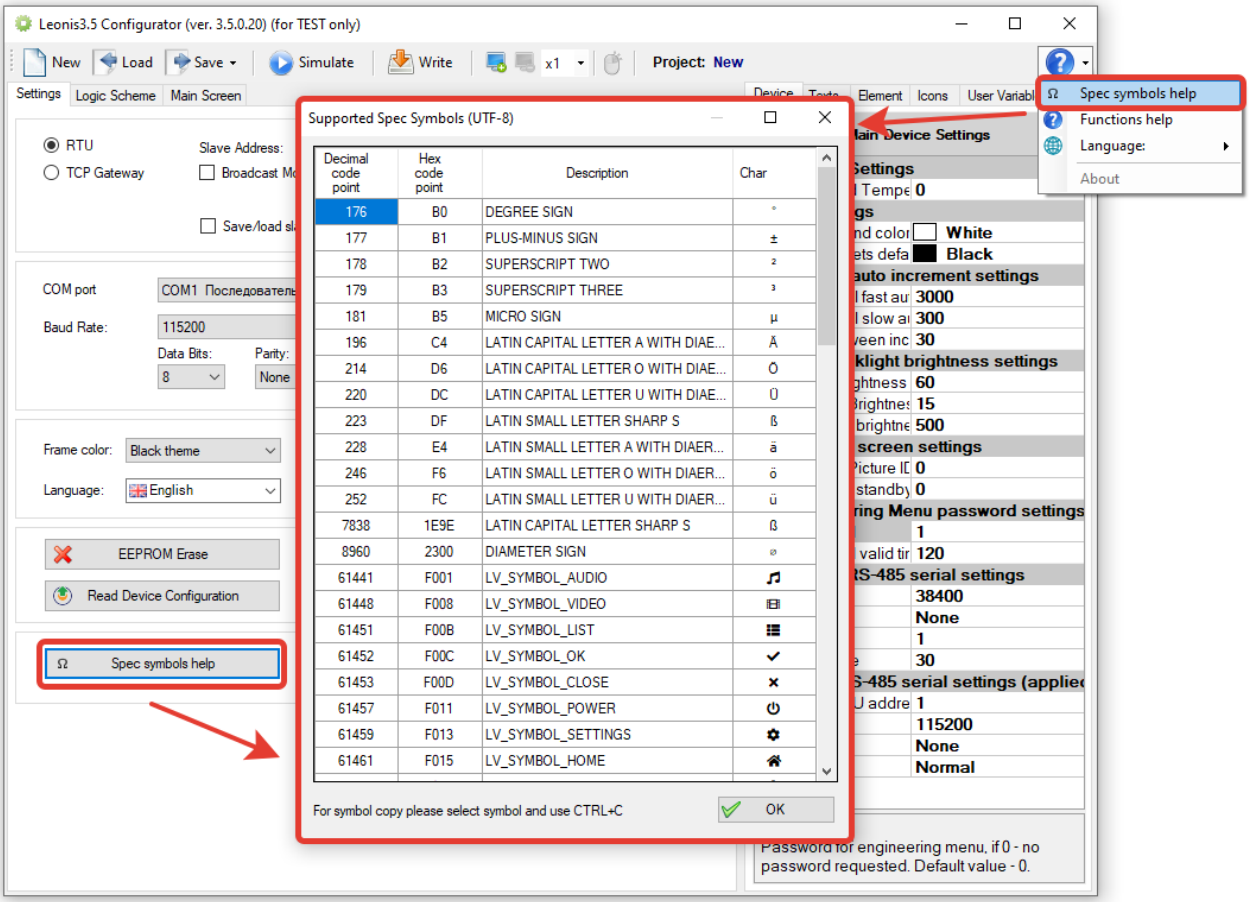


Simulation with different UserVar 1 values (0,1,2,3,4)



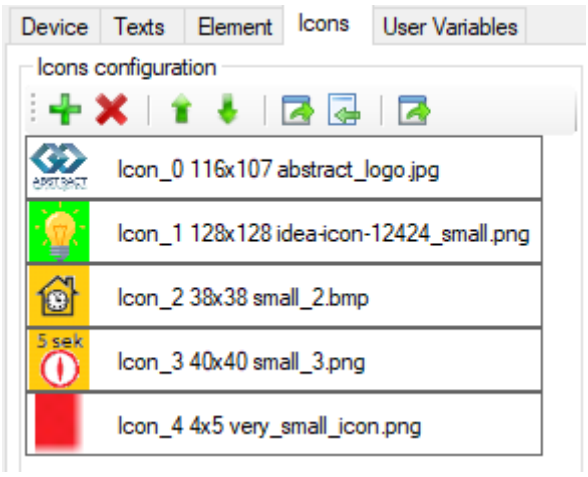
2.4 Supported special symbols

It is possible to copy needed symbol using CTRL+C combination.



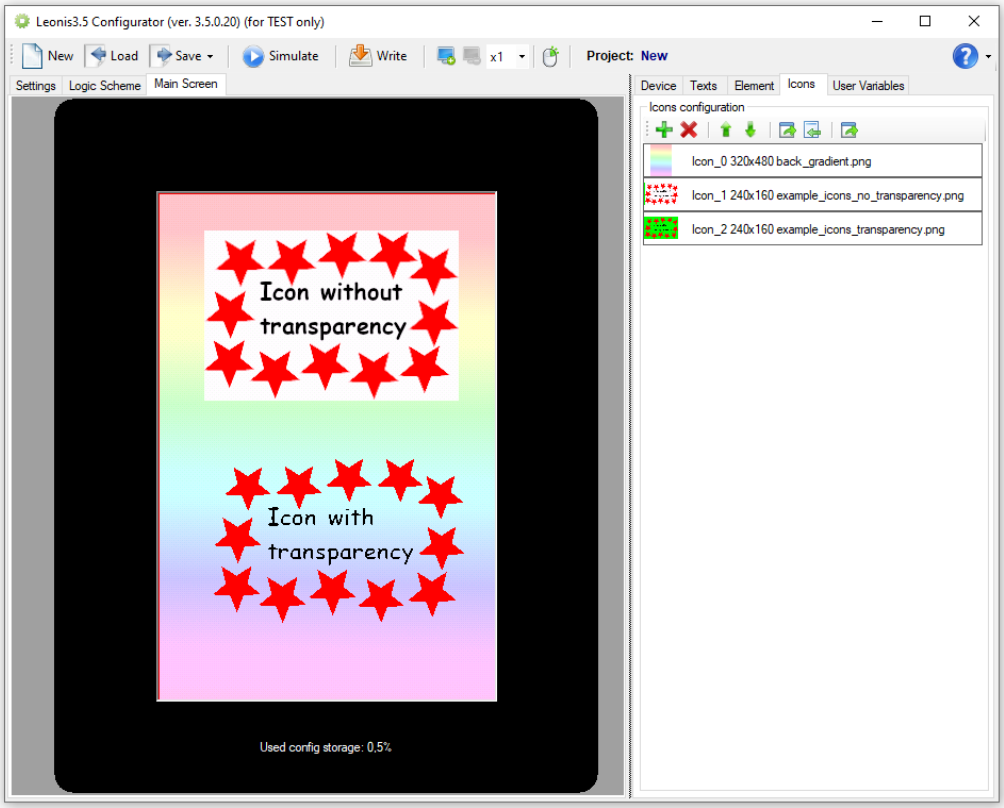
2.5 Icon widgets

To display some icon (picture) on the screen an Icon widget should be used. Picture ID parameter allows assigning exact image. Icons have type parameter, which defines the behaviour of the icon. For example, it can be Fixed or Indexed icon (works similar to Indexed text type, see above).



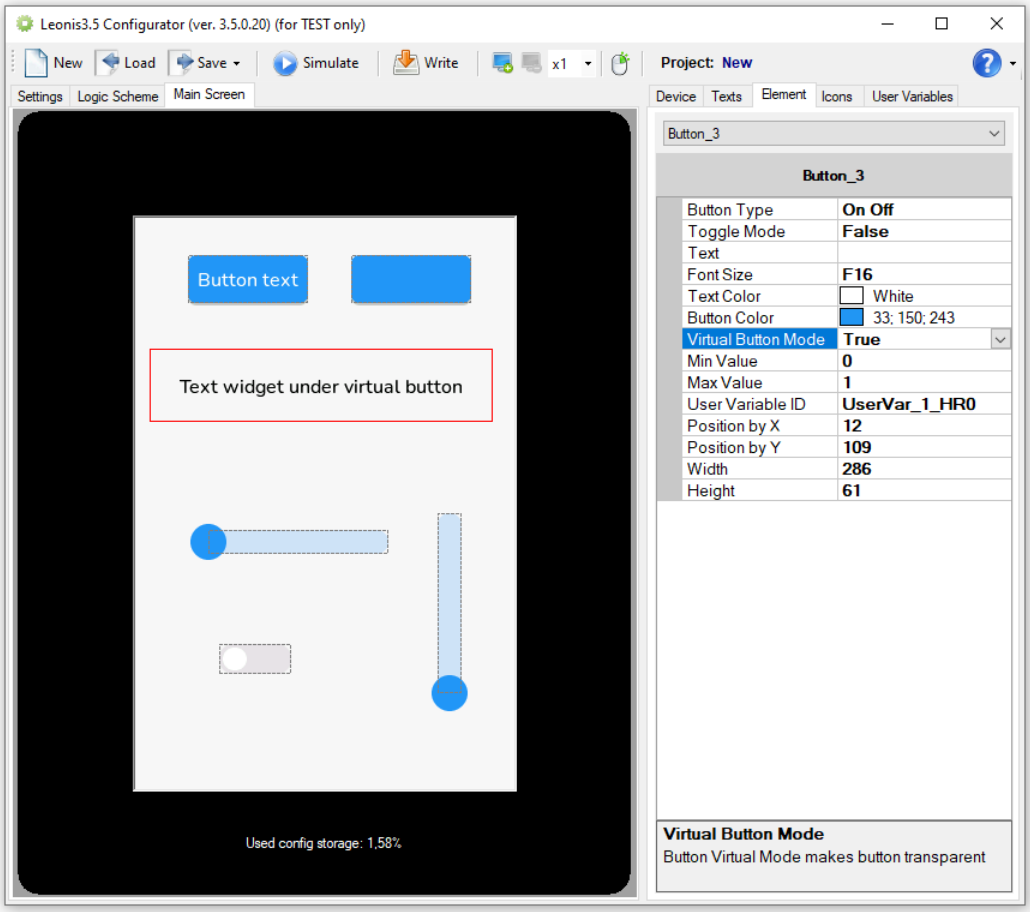
On Icons tab all project icons are listed. Up to 255 icons can be stored. Max possible icon size: 320x480. Max possible size of all icons (*.bfs file size): 6 MB (only uncompressed icons are stored, auto conversion to uncompressed format is always done for all icons loaded by user).

Transparent icons are supported: user can load png pictures with transparency (alpha channel) and it will be auto converted into transparency format supported by device (such icons transparent areas will be filled with bright green color on small previews at Icons tab).



2.6 Button, slider and switch widgets

Such widgets can change the assigned user variable on widget touch (exact behaviour depends from type of widget). Button widget supports Virtual Button Mode, in this mode button itself is not visible, so it can be placed over some other widget like Text or Icon to achieve some reaction on press, e.g. change string / picture on each press for indexed type Text or Icon widget.

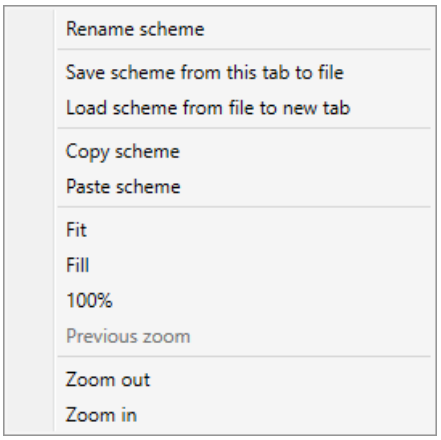


3 Logic programming

3.1 Introduction

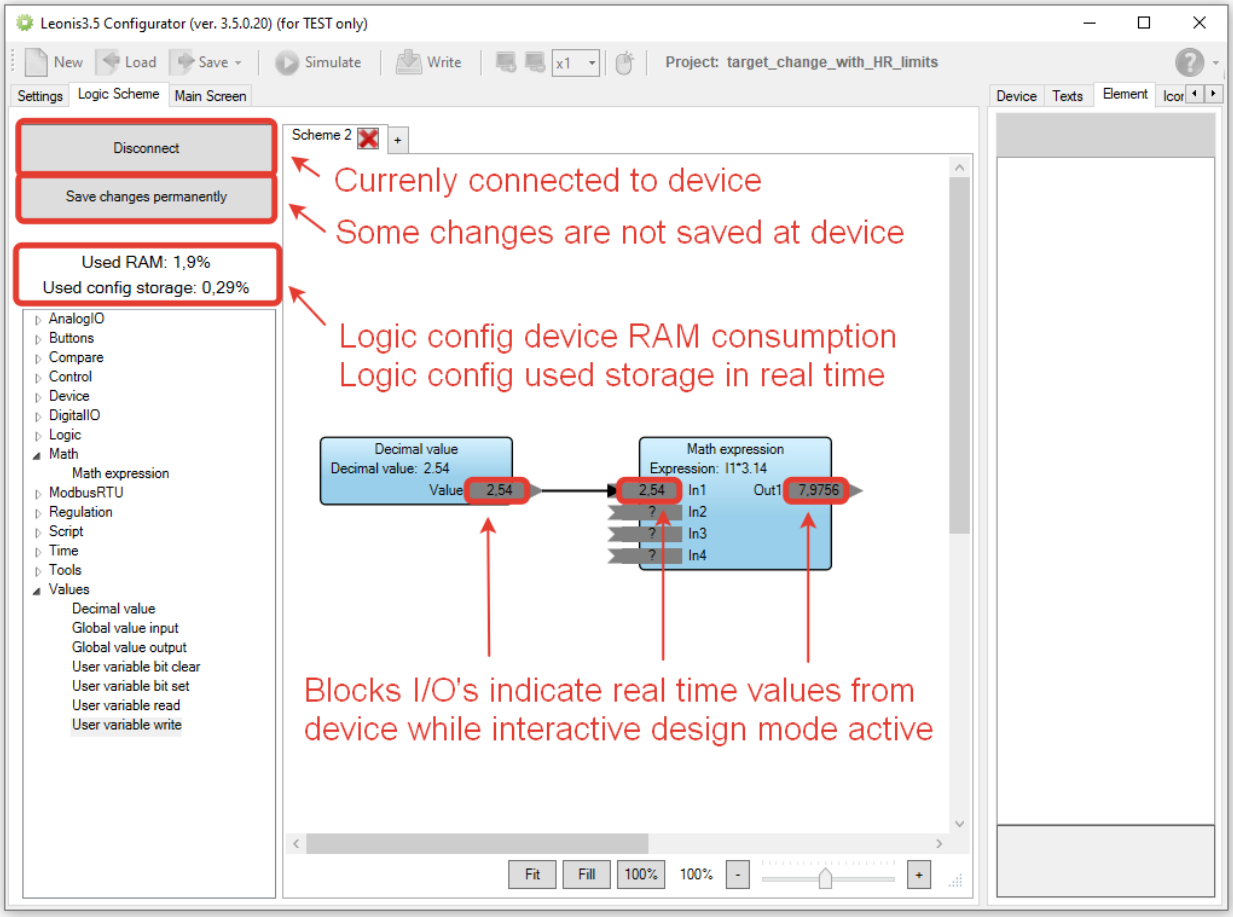
It is possible to create up to 30 logic schemes using the PC tool. Each scheme can include different blocks and connections between them. All schemes are stored together with UI (user interface) configuration in *.lmcfg file. Using right-click pop-up menu it is possible to copy existing scheme and paste it into other empty scheme sheet. Scheme name can be

changed in pop-up menu. Also each scheme can be saved / loaded to / from *.lmsch file separately from whole logic configuration (this can be useful to create typical schemes with some finished function and load them to different projects).



Max blocks number: 200-300 for all created schemes, depending from blocks complexity and number of connections between blocks. For example: in case of 10 active schemes it can be 20-30 blocks per each scheme and in case of 30 active schemes it can be 6-10 blocks per scheme.

Two approaches can be used for logic scheme creating: offline creation or online creation in interactive design mode. In offline mode device logic should be firstly created on PC and then loaded to device together with UI configuration. In online mode it is possible to create logic regulation configuration in real time: add/delete blocks, connect/disconnect wires or change blocks parameters actions will be immediately applied to the device.



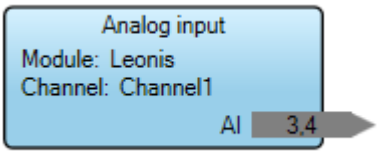
Also blocks I/O values will be displayed, so logic configuration debug can be performed. Slave RS485 is used for online mode. After creation is finished it is possible to save the logic scheme permanently to the device.

After interactive design mode, in offline mode, if some change will be done with logic scheme (e.g. change position of block) PC tool will not be able to start online mode again because local logic scheme differs from device logic scheme. So for starting online mode, local PC tool config should be exactly the same as device config (or both empty).

All internal signals (blocks connection wires) are float values. Even if signal can only take two values as for DIGITAL INPUT: 0 (OFF, low state) and 1 (ON, high state), it stored as float. So any block output can be connected to any other block input. Some blocks works with float values as is, without any conversion, for example LIMITER block. Other blocks designed to work with integer numbers only, for example all Modbus RTU/TCP write blocks, will truncate float input value fractional part. Also some blocks (LOGIC, JALOUSIE, DIMMER blocks) works with only 2 values (0 and 1), for such blocks input float values conversion take place: negative numbers, zero, range from 0.0 to 0.999 = 0 (OFF), all other values = 1 (ON), for example: -0.7 = 0, 0.0 = 0, 0.5 = 0, 1.0 = 1, 3.1 = 1.

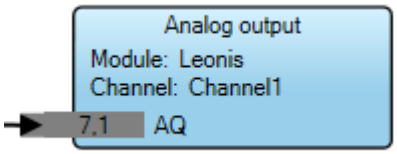
3.2 Analog I/O

Analog input



Description: selected module analog input channel real time value.
Parameters: module, channel.

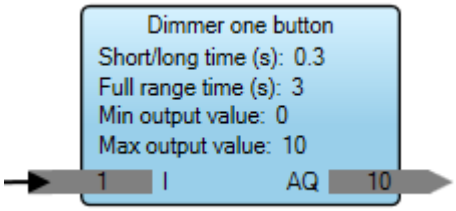
Analog output



Description: input value set to selected module analog output channel.
Parameters: module, channel.

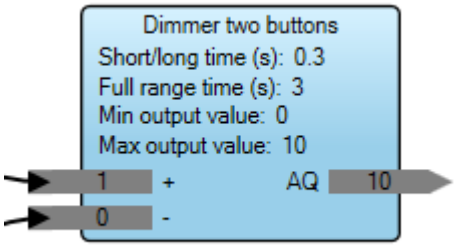
3.3 Buttons

Dimmer one button



Description: input short press set output to 0 or old value (depends of previous state), input long press increase/decrease output value from zero to maximum (or vise versa) for full range time. Direction is change to opposite after each long press.
Parameters: time for short/long detection threshold (seconds, 0.1-1000000), full range time (seconds, 0.1-1000000), min output value, max output value.

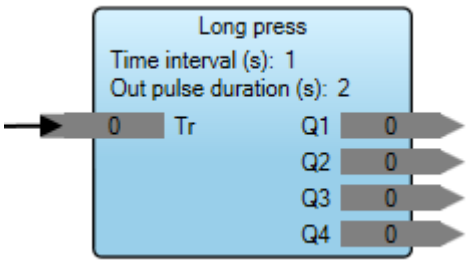
Dimmer two buttons



Description: minus input short press set output to 0, plus input short press set output to old value. Plus and minus inputs long press respectively increase/decrease output value from zero to maximum (or vise versa) for full range time.

Parameters: time for short/long detection threshold (seconds, 0.1-1000000), full range time (seconds, 0.1-1000000), min output value, max output value.

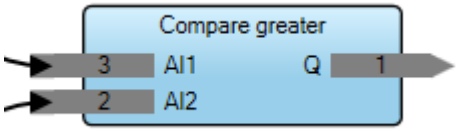
Long press



Description: selection one of four outputs depends from the time of the input ON state. Selected output generates pulse.
Parameters: press time interval for output selection (seconds, 0.1-1000000), out pulse duration (seconds, 0.1-1000000).

3.4 Compare

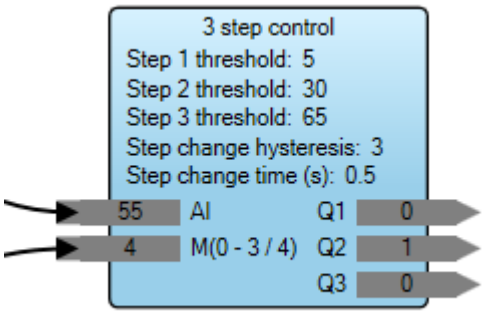
Compare greater



Description: output is ON if AI1 input value greater than AI2 input value and OFF if equal or less.

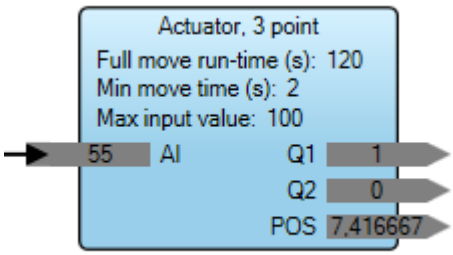
3.5 Control

3 step control



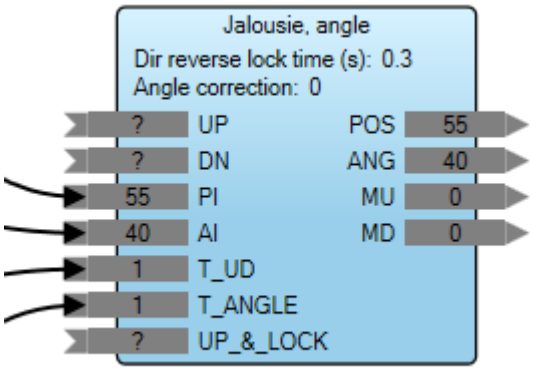
Description: converting AI input signal range (e.g. 0-100 %) to 3 step output according to thresholds with hysteresis. While change step, outputs turns OFF for step change time. M input allows manually set step (0 – all outputs OFF; 1, 2, 3 – output Q1/Q2/Q3 respectively) or enable auto conversion (4 or disconnected).
Parameters: step 1 threshold, step 2 threshold, step 3 threshold, step change hysteresis, step change time (seconds, 0.1-1000000).

Actuator, 3 point



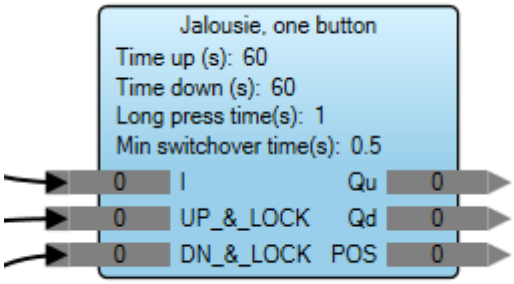
Description: converting AI input signal range (e.g. 0-100 %) to control position of 3-point actuator via Q1, Q2 outputs. POS output provides simulated actuator position.
Parameters: full move run-time of actuator (seconds, 1-1000000), min move time (seconds, 1-1000000), max input value.

Jalousie, angle



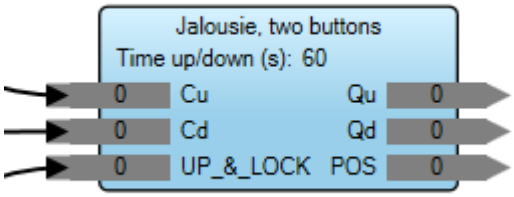
Description:
UP and DN inputs respectively controls MU and MD outputs (manual mode).
If both UP and DN are ON, auto mode is active (also auto mode can be activated if UP and DOWN inputs are not connected, while PI input is connected).
PI input (0..100) is desired position value input for auto mode.
AI input (0..100) is desired angle value input for auto mode.
T_UD controls up/down time in seconds (1-1000000, if nothing connected, default is 60s).
T_ANGLE controls angle change time in seconds (0-1000000, if nothing connected, default is 0).
On UP_&_LOCK input ON state active, jalousie goes up ignoring UP,DN input states and ignoring auto mode.
If T_ANGLE is set to non zero value, before position change, angle change occurs for both manual and auto mode.
POS output (0..100) calculated actual position based on up/down time.
0 – full dow, 100 – full up.
ANG output (0..100) calculated actual angle based on angle change time.
0 – full closed (light blocked), 100 – full opened (transparent).
Parameters:
dir reverse lock time (seconds, 0.1-1000000).
angle correction (value for angle correction on each activation), for example: T_ANGLE = 1 sec., angle correction = 20, AI change from 0 to 100 will take 1.2 sec. (1.0 change itself + 0.2 correction), AI change from 0 to 30 will take 0.5 sec (0.3 change itself + 0.2 correction);

Jalousie, one button



Description: I input short press start jalousie up/down process, input long press control jalousie up/down while pressed. Direction is change to opposite after each short/long press.
On UP_&_LOCK and DN_&_LOCK inputs ON state active, jalousie goes up and down accordingly ignoring I input state.
POS output (0..100) calculated actual position based on up/down time.
0 – full dow, 100 – full up.
Parameters: time up (seconds, 1-1000000), time down (seconds, 1-1000000), time for long detection threshold (seconds, 0.1-1000000), min switchover time (seconds, 0.1-1000000).

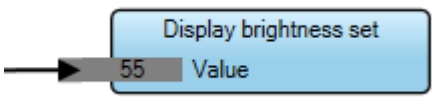
Jalousie, two buttons



Description: Cu and Cd inputs respectively controls Qu and Qd outputs. If input ON time more than 1 seconds and less than 3 seconds, respective output turns ON for up/down time.
On UP_&_LOCK input ON state active, jalousie goes up ignoring I input state.
POS output (0..100) calculated actual position based on up/down time.
0 – full dow, 100 – full up.
Parameters: time up/down (seconds, 1-1000000).

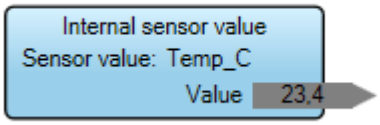
3.6 Device

Display brightness set



Description: allows to set display brightness in 0-100% range.

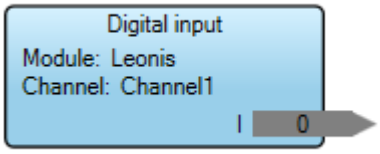
Internal sensors value



Description: selected internal sensor channel real time value.
Parameters: internal sensor channel.

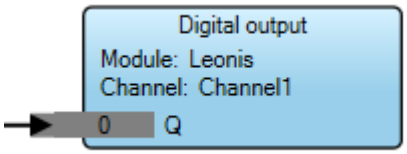
3.7 Digital I/O

Digital input



Description: selected module digital input channel real time value.
Parameters: module, channel.

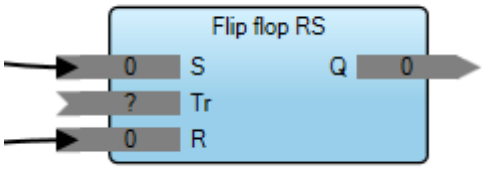
Digital output



Description: input value turn ON/OFF selected module digital output channel (0 and less = OFF, above zero value = ON).
Parameters: module, channel.

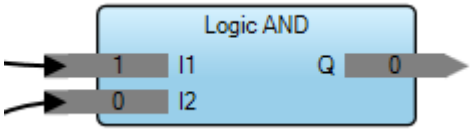
3.8 Logic

Flip flop RS



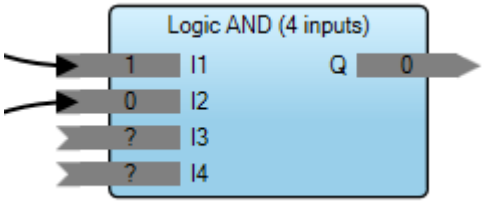
Description: S input ON value turn output ON. ON state will active until R input become ON. The R input is dominant. Trigger Tr input switch output on and off with each rising edge (S and R should be 0 or disconnected).

Logic AND



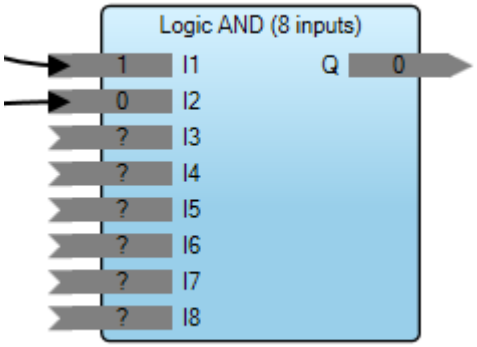
Description: output ON if both inputs connected and both inputs are ON, output OFF in other cases.

Logic AND (4 inputs)



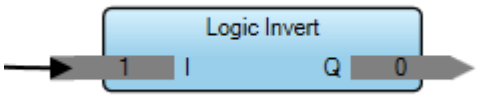
Description: output ON if connected more than one input and all of the connected inputs are ON, output OFF in other cases.

Logic AND (8 inputs)



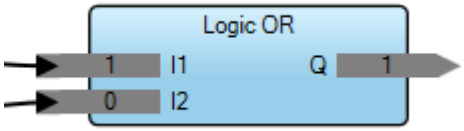
Description: output ON if connected more than one input and all of the connected inputs are ON, output OFF in other cases.

Logic INVERT



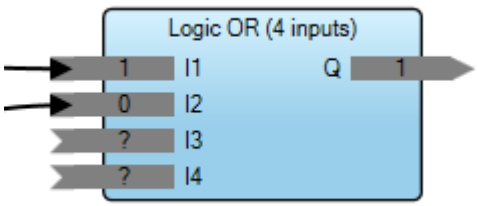
Description: output ON if connected input is OFF and OFF in other cases.

Logic OR



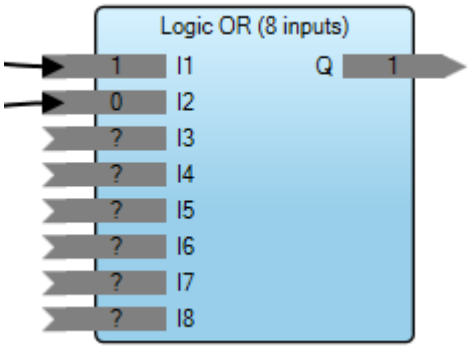
Description: output ON if both inputs connected and any input is ON, output OFF in other cases.

Logic OR (4 inputs)



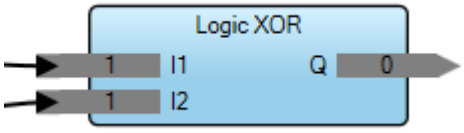
Description: output ON if connected more than one input and any connected input is ON, output OFF in other cases.

Logic OR (8 inputs)



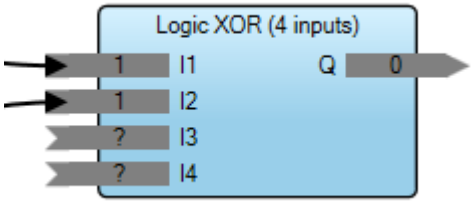
Description: output ON if connected more than one input and any connected input is ON, output OFF in other cases.

Logic XOR



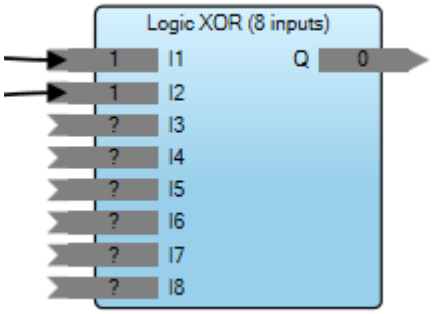
Description: output ON if both inputs connected and only one connected input is ON, output OFF in other cases.

Logic XOR (4 inputs)



Description: output ON if connected more than one input and only one connected input is ON, output OFF in other cases.

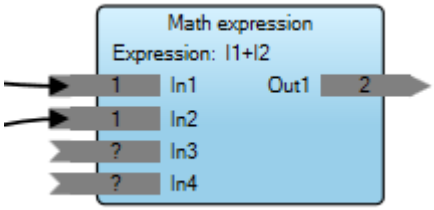
Logic XOR (8 inputs)



Description: output ON if connected more than one input and only one connected input is ON, output OFF in other cases.

3.9 Math

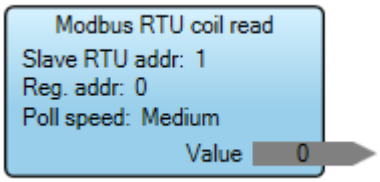
Math expression



Description: calculation result of expression where up to four inputs (I1, I2, I3, I4) can be used as variables.
Parameters: expression, e.g.: ((I1+I2)/I3)*I4 or sqrt(pow(I1,2)+(pi*I2*e)). Supports: addition (+), subtraction (-), multiplication (*), division (/), exponentiation (^), modulus (%). C style math functions: abs (calls to fabs), acos, asin, atan, atan2, ceil, cos, cosh, exp, floor, ln (calls to log), log (calls to log10), log10, pow, sin, sinh, sqrt, tan, tanh. Also following constants are available: pi, e.

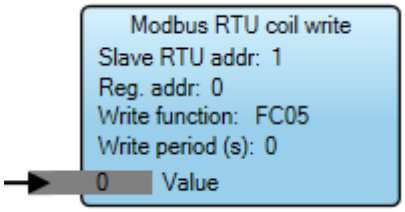
3.10 Modbus RTU

Modbus RTU coil read



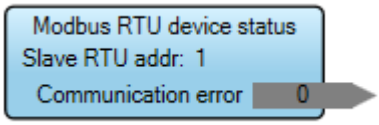
Description: modbus RTU slave device 1-bit coil value.
Parameters: slave device address (1-247), register address (0-65535), poll speed (slowest/slow/medium/fast/fastest).

Modbus RTU coil write



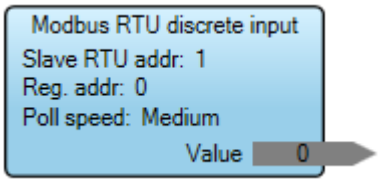
Description: input value write to modbus RTU slave device 1-bit coil.
Parameters: slave device address (1-247), register address (0-65535), write function (FC05 – single coil or FC15 – multiply coils), write period (seconds, 0.1-1000000).
Write occurs if input value changed or if write period time is come (write period should have non zero value for periodic write).

Modbus RTU device status



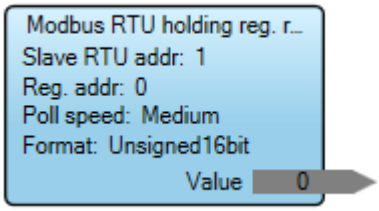
Description: modbus RTU slave device communication status (0 = no communication error, 1 = communication error).
Parameters: slave device address (1-247).

Modbus RTU discrete input



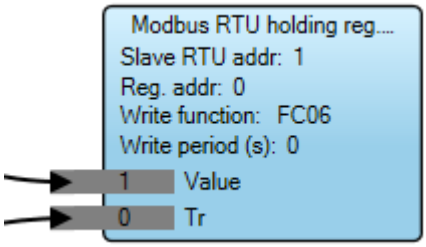
Description: modbus RTU slave device 1-bit discrete input value.
Parameters: slave device address (1-247), register address (0-65535), poll speed (slowest/slow/medium/fast/fastest).

Modbus RTU holding reg. read



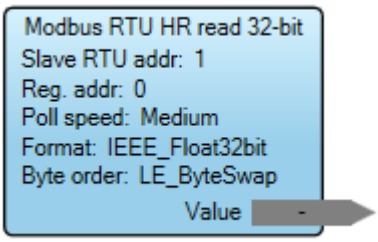
Description: modbus RTU slave device 16-bit holding register value.
Parameters: slave device address (1-247), register address (0-65535), poll speed (slowest/slow/medium/fast/fastest), format (signed 16-bit, unsigned 16-bit).

Modbus RTU holding reg. write



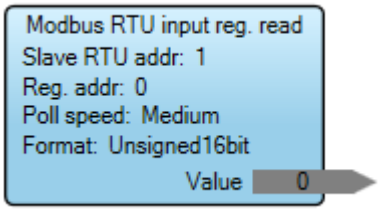
Description: input value write to 16-bit modbus RTU slave device holding register.
Parameters: slave device address (1-247), register address (0-65535), write function (FC06 – single register or FC16 – multiply registers), write period (seconds, 0.1-1000000). Write period should have non zero value for periodic write.
Write occurs:
1) Tr input disconnected: on input value change or on write period time come.
2) Tr input connected: on write period time come or on Tr (trigger input) pulse (0→1).

Modbus RTU HR read 32-bit



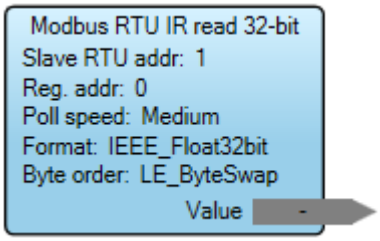
Description: modbus RTU slave device 32-bit value read (two holding registers read).
Parameters: slave device address (1-247), register address (0-65534), poll speed (slowest/slow/medium/fast/fastest), format (IEEE_Float32bit, signed 16-bit, unsigned 16-bit), byte order (BE, LE, BE_ByteSwap, LE_ByteSwap).

Modbus RTU input register



Description: modbus RTU slave device 16-bit input register value.
Parameters: slave device address (1-247), register address (0-65535), poll speed (slowest/slow/medium/fast/fastest), format (signed 16-bit, unsigned 16-bit).

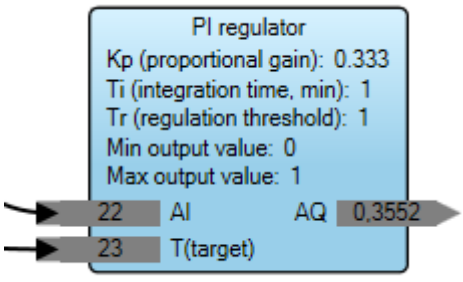
Modbus RTU IR read 32-bit



Description: modbus RTU slave device 32-bit value read (two input registers read).
Parameters: slave device address (1-247), register address (0-65534), poll speed (slowest/slow/medium/fast/fastest), format (IEEE_Float32bit, signed 16-bit, unsigned 16-bit), byte order (BE, LE, BE_ByteSwap, LE_ByteSwap).

3.11 Regulation

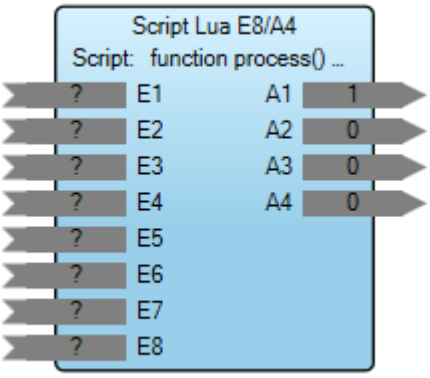
PI regulator



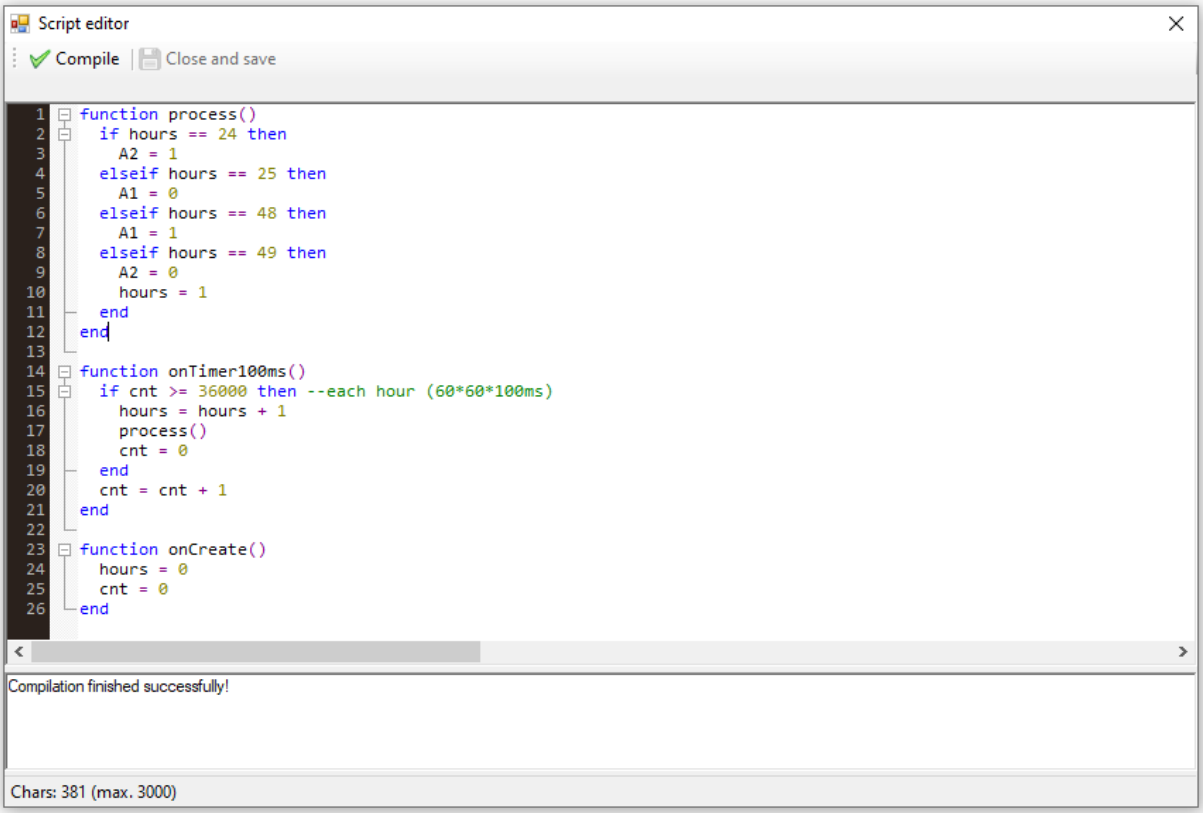
Description: proportional and integral controller for reaching and maintaining target value (e.g.: temperature). Output value depends from error between current value (AI input) and target value (T input).
Parameters: Kp (proportional gain), Ti (integration time, minutes), Tr (regulation threshold, no regulation if error lower than threshold), minimum output value, maximum output value.

3.12 Script

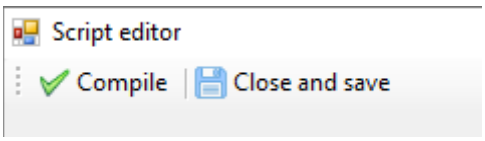
Script Lua E8/A4



Description: Lua programming language script with 8 inputs and 4 outputs.
Parameters: script (Lua source code).
Double click on block to open script editor window.



Lua (www.lua.org) is used as interpreter.
Lua version 5.3 is supported.
Compile button can be used for checking script correctness.
Close and save button should be pressed for script save (script editor close will not save changes).



Max possible script length is 3000 chars.

There are following internally called functions, these should be presented in each script:
onTimer100ms() → called each 100 ms, whole logic should be done here.
onCreate() → called on start once, should be used for init and for block aliases.

Also following system functions available to use:
sys_get_hour() → returns current hour 0..23
sys_get_minute() → returns current minute 0..59
sys_get_second() → returns current second 0..59
sys_get_weekday() → returns current weekay 1 → Monday, 2 → Tuesday...
sys_get_year() → returns current year 0..99
sys_get_month() → returns current month 1..12
sys_get_day() → returns current day 1..31
sys_uservar_read(n, [f]) → read UserVar value; parameters:
n – uservar number: 1..100 (mandatory)
[f] – conversion format, interpret uservar as: 0 → unsigned, 1 → signed (optional, signed if not specified)
return – uservar value in range 0..65535 or -32768..32767 depending on conversion format

```
function onTimer100ms()
  A1 = sys_uservar_read(1)
  A2 = sys_uservar_read(2, 0)
end
```

Also it is possible to use **sys_uv_read()** instead of **sys_uservar_read()**

sys_uservar_write(n, v) → write UserVar value; parameters:
n – uservar number: 1..100 (mandatory)
v – integer value to write in range -32768..65535 (mandatory)

```
function onTimer100ms()
  sys_uservar_write(1, E1)
end
```

Also it is possible to use **sys_uv_write()** instead of **sys_uservar_write()**

sys_alias(x, y) → creates alias; parameters:
x – block input or output, e.g. "E1" or "A1"
y – alias for block input or output, e.g. "CTRL" or "FAN"
I/O aliases allow to use convenient names for work with blocks I/O's instead of Ex and Ax.
Example:

```
function onCreate()
  sys_alias("E1","CTRL")
  sys_alias("A1","FAN")
  FAN = 1
end
```

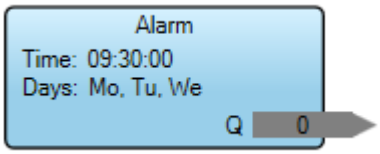
Following Lua libraries are available:
math (abs, acos, asin, atan, ceil, cos, deg, exp, tointeger, floor, fmod, ult, log, max, min, modf, rad, random, randomseed, sin, sqrt, tan, type, pi, huge, maxinteger, mininteger)

Example:

```
A1 = math.max(E1,E2)
```

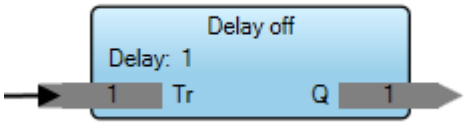
3.13 Time

Alarm



Description: output pulse (duration 2 seconds) occurs if current weekday and time equal to alarm time and weekday.
Parameters: alarm time in 24 hour format, weekdays to alarm.

Delay off



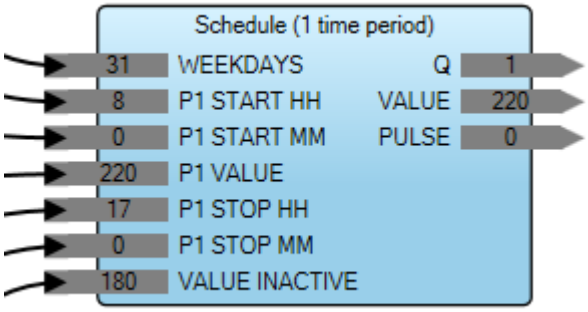
Description: output stay ON for delay value time, after input become OFF.
Parameters: delay value (seconds, 0.1-1000000).

Delay on



Description: output is ON, if input is ON longer than delay value.
Parameters: delay value (seconds, 0.1-1000000).

Schedule (1 time period)

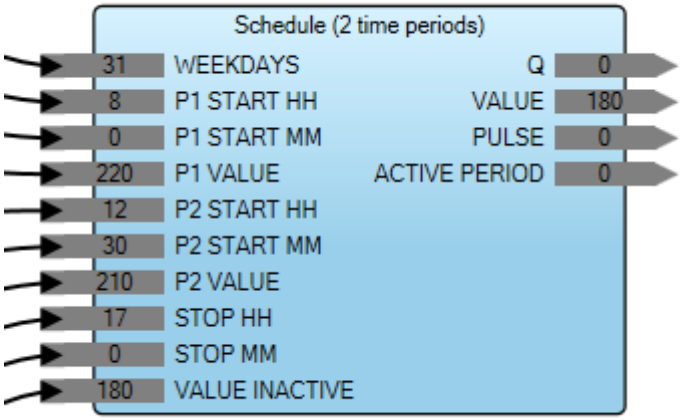


Description: output Q is ON, if current weekday & time is inside specified WEEKDAYS & P1 START HH:MM...P1 STOP HH:MM time period and OFF if outside.
WEEKDAYS input value bits defines active weekdays (bit0 → Monday, bit1 → Tuesday,...).
VALUE outputs:
P1 VALUE if currently inside specified period,
VALUE INACTIVE if outside.

PULSE produces pulse (duration 2 seconds) when specified period enter/exit.

All days are active if WEEKDAYS input is not connected.
P1 START HH default = 00 if input is not connected (same for others HH:MM inputs).
If P1 STOP HH:MM = 00:00 (or STOP inputs are not connected) period will be ended at midnight.
P1 VALUE default = 0 if input is not connected (same for VALUE INACTIVE input).

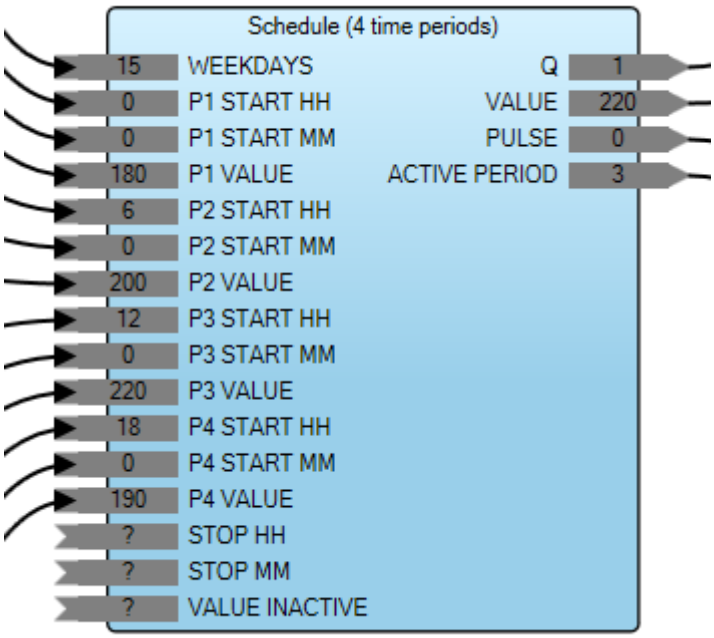
Schedule (2 time periods)



Description: output Q is ON, if current weekday & time is inside specified WEEKDAYS &
P1 START HH:MM...P2 START HH:MM,
P2 START HH:MM...STOP HH:MM
time periods and OFF if outside.
WEEKDAYS input value bits defines active weekdays (bit0 → Monday, bit1 → Tuesday,...).
VALUE outputs:
P1 VALUE if currently inside P1 period,
P2 VALUE if currently inside P2 period,
VALUE INACTIVE if outside.
PULSE produces pulse (duration 2 seconds) when some of the specified periods enter/exit.
ACTIVE PERIOD outputs:
0 if outside of any period,
1 if currently inside P1 period,
2 if currently inside P2 period.

All days are active if WEEKDAYS input is not connected.
P1 START HH default = 00 if input is not connected (same for others HH:MM inputs).
If STOP HH:MM = 00:00 (or STOP inputs are not connected) P2 period will be ended at midnight.
P1 VALUE default = 0 if input is not connected (same for P2 VALUE, VALUE INACTIVE inputs).

Schedule (4 time periods)



Description: output Q is ON, if current weekday & time is inside specified WEEKDAYS & P1 START HH:MM...P2 START HH:MM,
P2 START HH:MM...P3 START HH:MM,
P3 START HH:MM...P4 START HH:MM,
P4 START HH:MM...STOP HH:MM
time periods and OFF if outside.
WEEKDAYS input value bits defines active weekdays (bit0 → Monday, bit1 → Tuesday,...).
VALUE outputs:

- P1 VALUE if currently inside P1 period,
- P2 VALUE if currently inside P2 period,
- P3 VALUE if currently inside P3 period,
- P4 VALUE if currently inside P4 period,
- VALUE INACTIVE if outside.

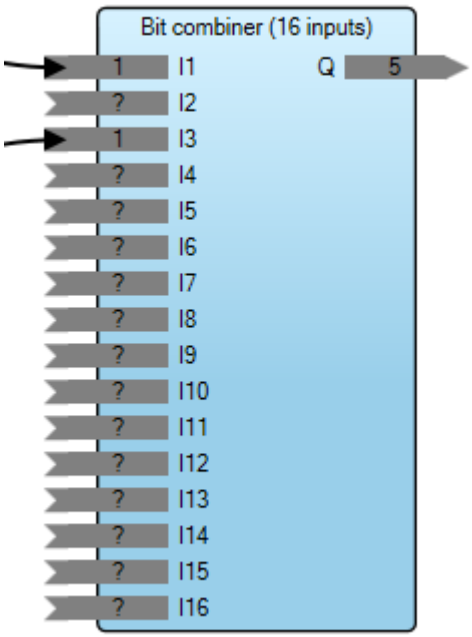
PULSE produces pulse (duration 2 seconds) when some of the specified periods enter/exit.
ACTIVE PERIOD outputs:

- 0 if outside of any period,
- 1 if currently inside P1 period,
- 2 if currently inside P2 period,
- 3 if currently inside P3 period,
- 4 if currently inside P4 period.

All days are active if WEEKDAYS input is not connected.
P1 START HH default = 00 if input is not connected (same for others HH:MM inputs).
If STOP HH:MM = 00:00 (or STOP inputs are not connected) P4 period will be ended at midnight.
P1 VALUE default = 0 if input is not connected (same for P2 VALUE, P3 VALUE, P4 VALUE, VALUE INACTIVE inputs).

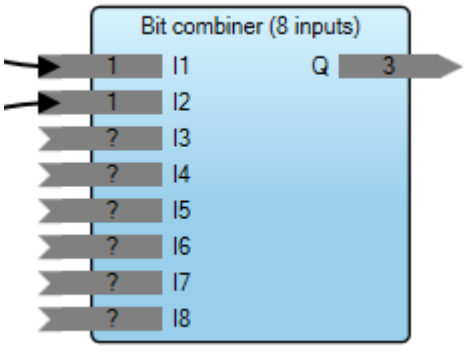
3.14 Tools

Bits combiner (16 inputs)



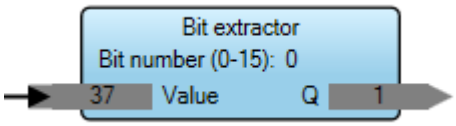
Description: outputs number with bits set according to 16 inputs values (0 and less – corresponding bit clear, 1 and above – corresponding bit set). Unconnected input clears corresponding bit.

Bits combiner (8 inputs)



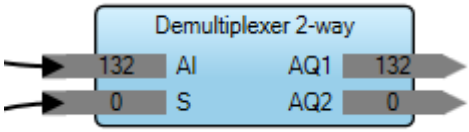
Description: outputs number with bits set according to 8 inputs values (0 and less – corresponding bit clear, 1 and above – corresponding bit set). Unconnected input clears corresponding bit.

Bit extractor



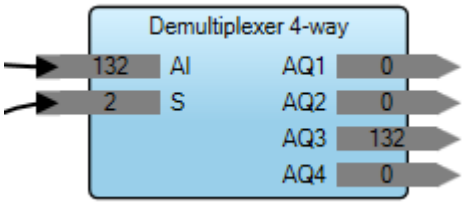
Description: output is ON if selected bit of input value is set.
Parameters: bit number (0-15).

Demultiplexer 2-way



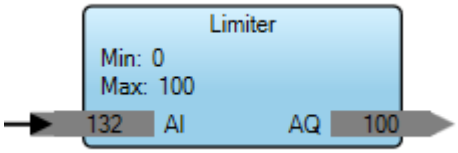
Description: S input value selects output for AI value (0 = AQ1, 1 = AQ2).

Demultiplexer 4-way



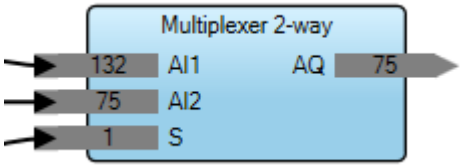
Description: S input value selects output for AI value (0 = AQ1, 1 = AQ2, 2 = AQ3, 3 = AQ4).

Limiter



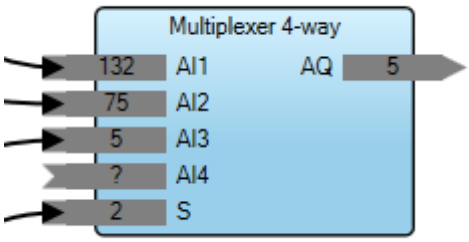
Description: outputs limited input value.
Parameters: minimum possible value, maximum possible value.

Multiplexer 2-way



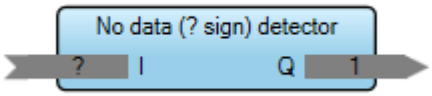
Description: if S input OFF, output = AI1 input, if S input ON, output = AI2 input.

Multiplexer 4-way



Description: S input value selects one of 4 inputs as source for output (0 = AI1, 1 = AI2, 2 = AI3, 3 = AI4).

No data (? sign) detector



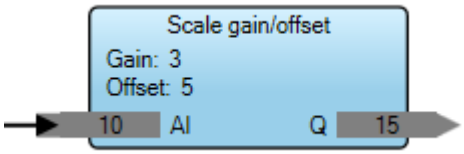
Description: Q output is ON when ? (no date) value connected to I input.

Reset device



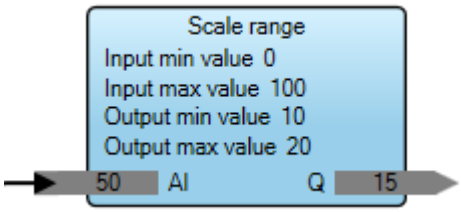
Description: resets device if I input value not equal to 0.

Scale gain/offset



Description: output value = (input value - offset) * gain.
Parameters: offset, gain.

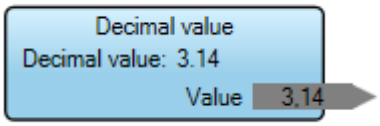
Scale range



Description: output value = scaled to a different range input value.
Parameters: input min value, input max value, output min value, output max value.

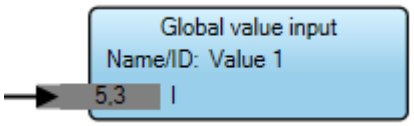
3.15 Values

Decimal value



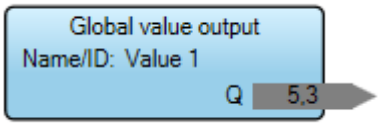
Description: output decimal value.
Parameters: decimal value.

Global value input



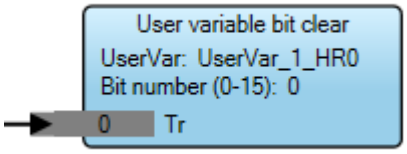
Description: input for global value output block with the same name/id.
Parameters: global value textual name/identifier.

Global value output



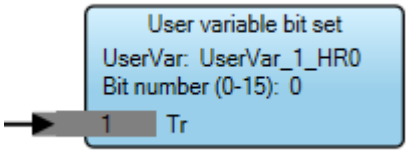
Description: output from global value input block with the same name/id.
Parameters: global value textual name/identifier.

User variable bit clear



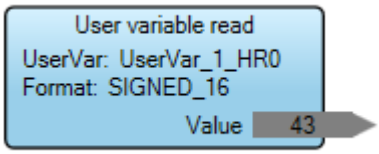
Description: Clears defined user variable bit if I input value not equal to 0.
Parameters: user variable ID, bit number (0-15).

User variable bit set



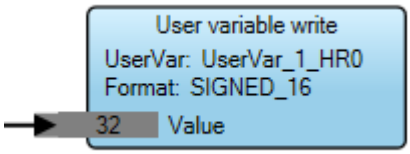
Description: Sets defined user variable bit bit if I input value not equal to 0.
Parameters: user variable ID, bit number (0-15).

User variable read



Description: output value from selected user variable.
Parameters: user variable ID, Format – how to interpret UserVar HR value (Unsigned 16-bit, Signed 16-bit, FLOAT32_BE, FLOAT32_LE, FLOAT32_BE_ByteSwap, FLOAT32_LE_ByteSwap) (all FLOAT32 options uses two uservars: specified and next one).

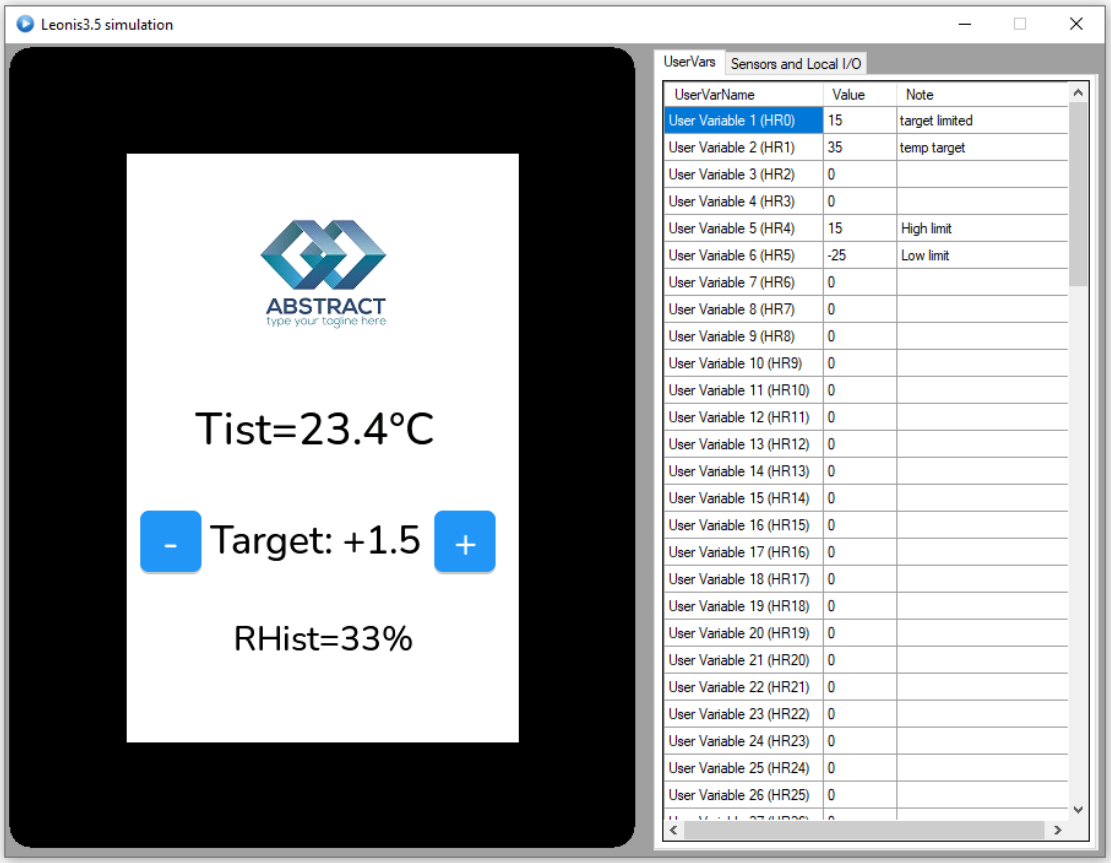
User variable write



Description: input for value to write into selected user variable.
Parameters: user variable ID, Format – how to interpret UserVar HR value (Unsigned 16-bit, Signed 16-bit, FLOAT32_BE, FLOAT32_LE, FLOAT32_BE_ByteSwap, FLOAT32_LE_ByteSwap) (all FLOAT32 options uses two uservars: specified and next one).

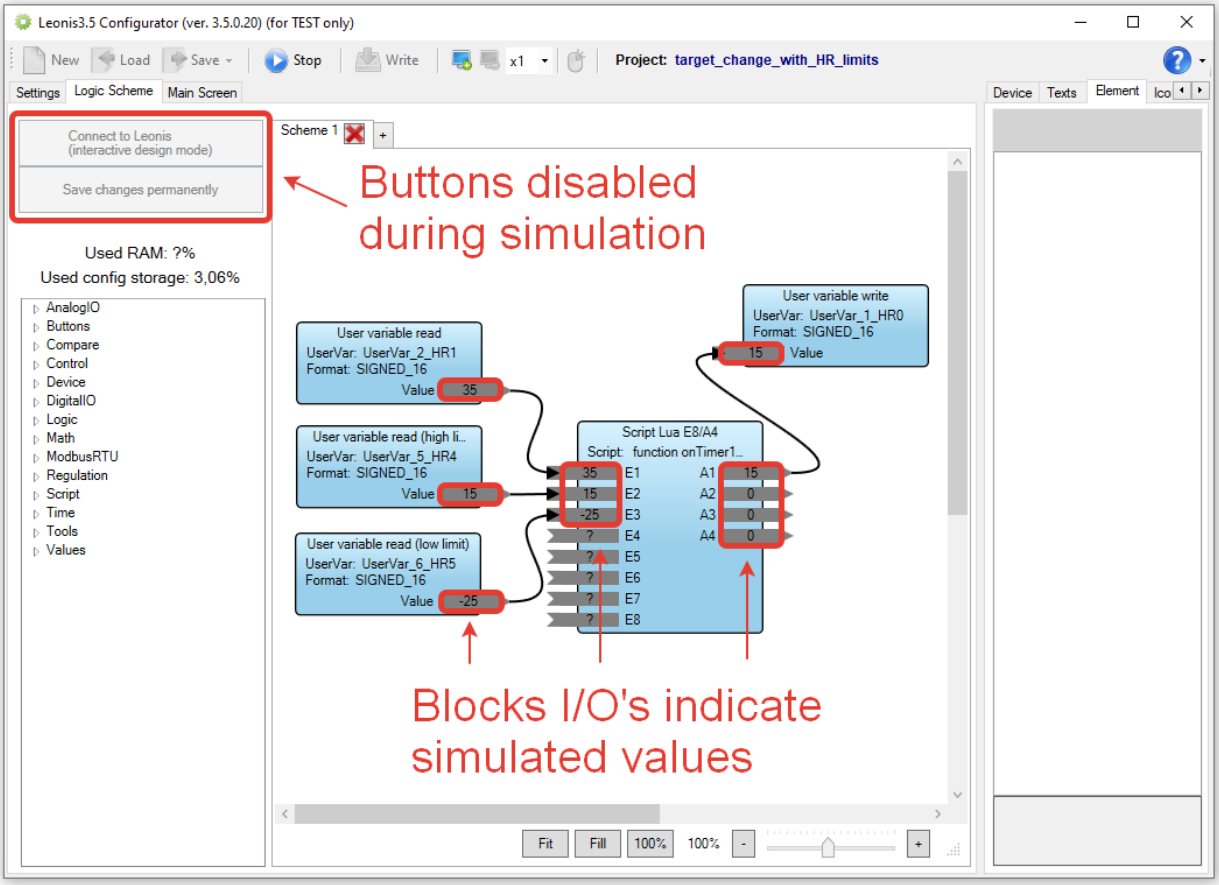
4 Simulation

Simulation mode can be helpful for checking configuration behaviour without loading to device. To start simulation, a “Simulate” button should be pressed, separate simulation windows will be shown.



Full device behaviour simulation is supported including UI and logic. It is possible to view / change user variables during simulation. Also user can define internal sensor values (Temperature, Humidity, CO2) and local device I/O.

While simulation active regulation logic blocks I/O values displayed at the Logic Scheme tab of the configurator main window.

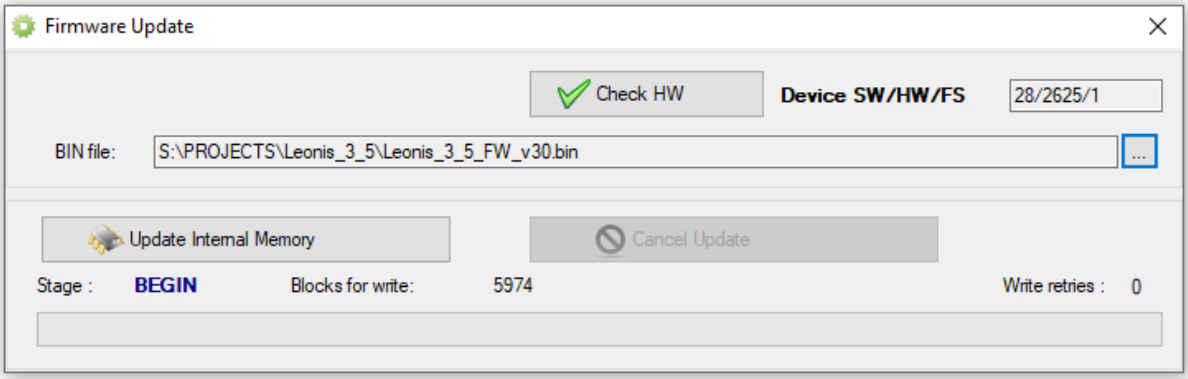


It is possible to keep the simulation window open and continue to create UI and regulation logic, all changes will be applied to the simulation window.

5 Work with Leonis3.5 device

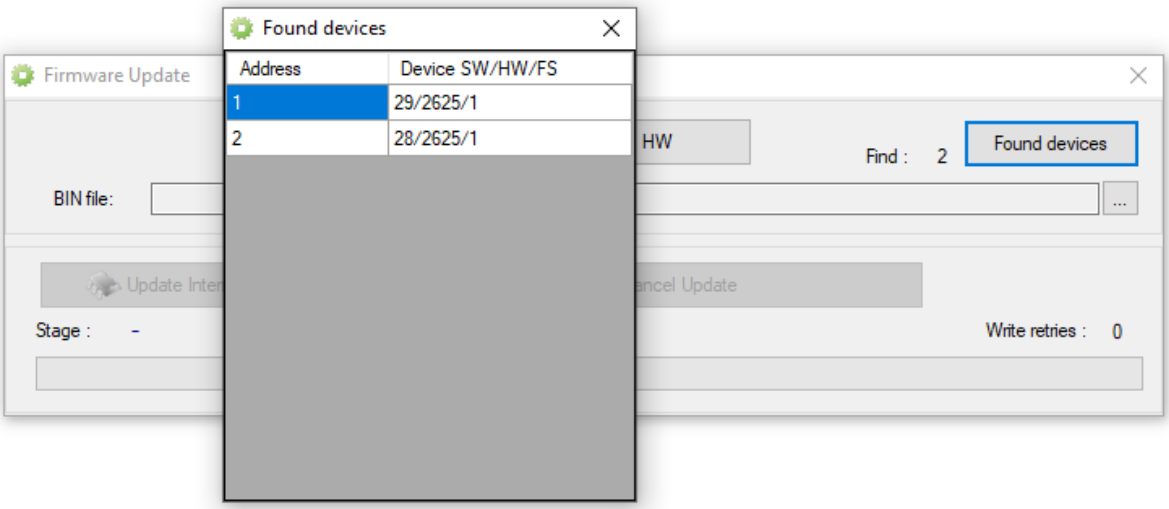
5.1 Firmware update and checking software and configuration version

Remote device FW update can be performed using the firmware update window (button on Settings tab). Update is possible via RS485 or USB connection. For the firmware update the appropriate BIN file must be selected and then Update of the internal memory must be pressed.



Also it is possible to check connection to the device by pressing the Check HW button. In case of a successful connection, Leonis3.5 software version (SW), hardware version (HW) and file system version (FS) will be displayed.

To update several devices at the same time via RS485, the broadcast mode can be set. Then "Check HW" is mandatory. If broadcast mode is used, the configurator will scan 1..247 address range for devices. If at least one device is found, Found device button will become active. By pressing it, the table with found devices will be shown. It can be useful to check SW versions of found devices to ensure that broadcast FW update was successful. If some devices have not been updated, they should be updated individually in the addressed mode (without broadcast). Also after broadcast update of device configurations, found FS versions information can help to check if all configurations were successfully loaded.

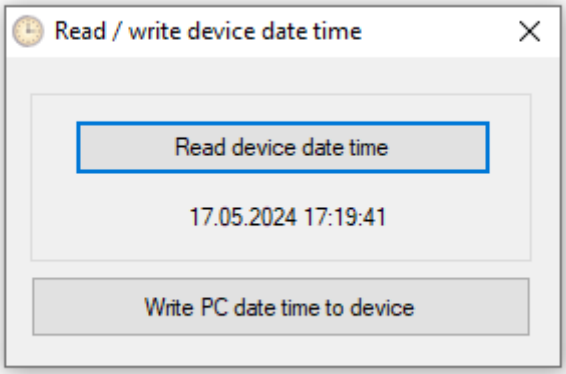


5.2 EEPROM erase

Whole device configuration (Main device settings + UI + logic + Icons images data) can be erased using the EEPROM erase button on Settings tab.

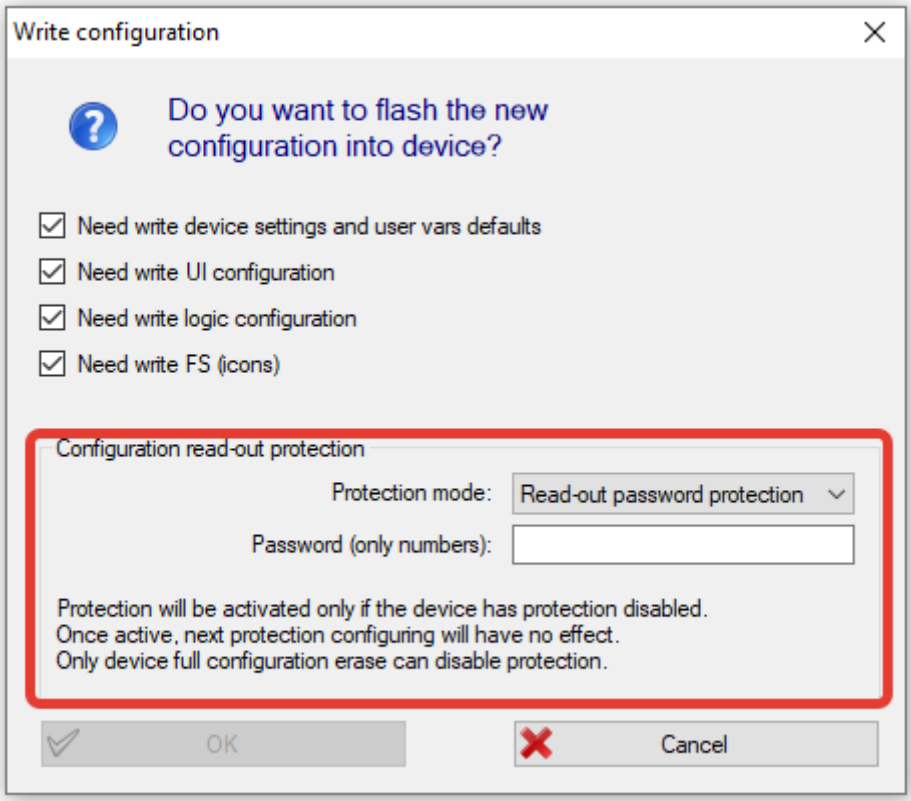
5.3 Read / write device date time

It is possible to read or write device date and time using the corresponding button on Settings tab.

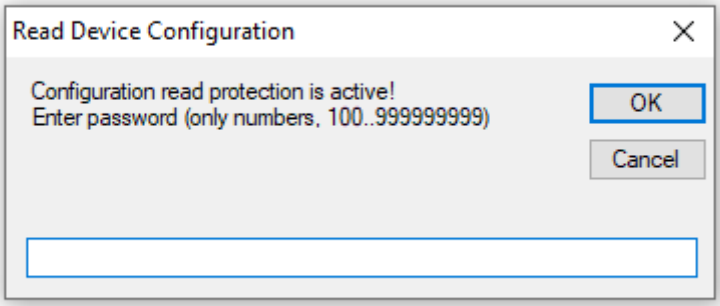


5.4 Read device configuration

Whole device configuration (Main device settings + UI + logic + Icons images data) can be read from device using the read device configuration button on Settings tab.
It is possible to protect configuration read-out by setting protection mode while new configuration write. Supported modes: no protection, read-out password protection, complete read-out lock. Protection can be activated only if there is no already active protection on device. In other case protection mode selection drop-down will be unavailable.
There is no any configuration write restrictions in case of active read-out protection.
It is possible to use number in range 100..999999999 as a password.

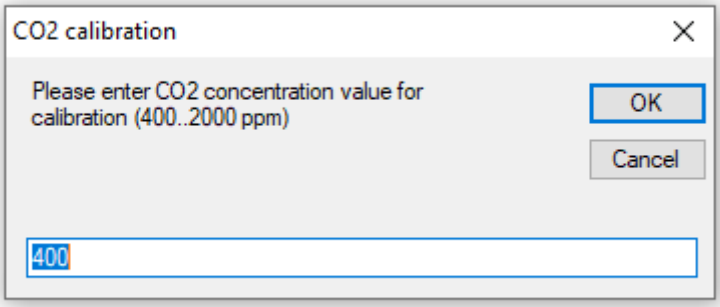


In case of active password protection on device read try password entry window will be displayed.



5.5 CO2 sensor calibration

For CO2 sensor calibration the corresponding button on Settings tab should be used.



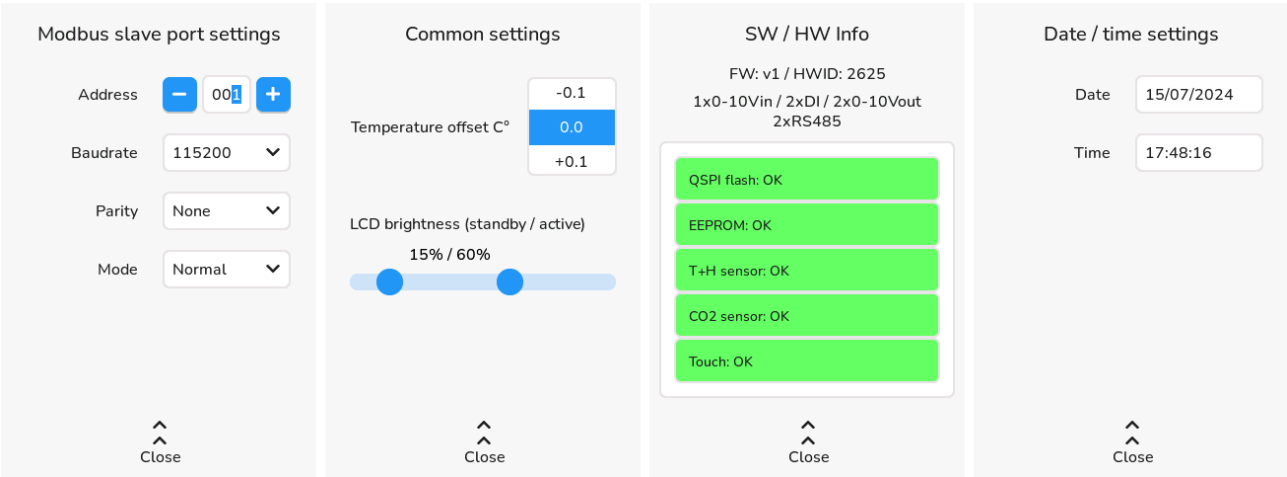
6 Leonis3.5 device information

6.1 Settings screens

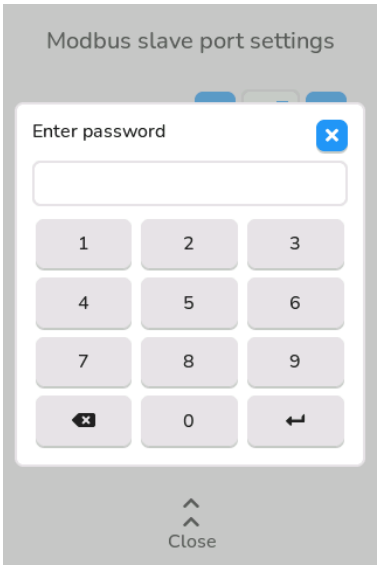
Commons settings screen can be used for configure active / standby brightness and T measured correction directly on Leonis3.5 device. To enter common settings, 3 vertical swipes in the top screen area should be done.

Modbus slave port settings screen is used for configuring Modbus RTU slave port parameters (address, parity, baudrate). This screen can be opened with left to right horizontal swipe in the bottom area of the common settings screen.

SW / HW info screen can be opened with right to left horizontal swipe in the bottom area of the common settings screen.



It is possible to protect change of Modbus slave port settings with password. Password can be defined in the settings tab of the configurator. In case of active engineering menu password (password value different from 0), password enter screen will appear on change try.


Modbus slave port settings
Enter password
1
2
3
4
5
6
7
8
9
0
Close