

Manual

# PB610-B Panel Builder 600 Programming Software for CP600-eCo Control Panels

Power and productivity  
for a better world™





# Contents

<b>1</b>	<b>Getting started</b>	<b>1</b>	FreeType font rendering	45
	Assumptions	2	Behavior	45
	Installing the application	2	Events	48
<b>2</b>	<b>Runtime</b>	<b>5</b>	<b>6 The HMI simulator</b>	<b>49</b>
	Runtime modes	6	Data simulation methods	50
	HMI device basic settings	6	Simulator settings	50
	Context menu options	7	Launching and stopping the simulator	51
	Built-in SNTp service	10	<b>7 Transferring the project to HMI device</b>	<b>53</b>
<b>3</b>	<b>My first project</b>	<b>11</b>	Download to HMI device	54
	The workspace	12	Update package	57
	Creating a project	12	The Runtime loader	59
	Communication protocols	14	Upload projects	60
	Designing a page	15	<b>8 System Variables</b>	<b>63</b>
	The Widget Gallery	17	Alarms variables	65
	Adding tags	19	Buzzer variables	65
	Exporting tags	21	Communication variables	66
	Importing tags	21	Daylight Saving Time variables	66
	Attaching widget to tags	24	Device variables	67
	Dialog pages	26	Dump information variables	68
<b>4</b>	<b>Programming concepts</b>	<b>27</b>	Keypad variables	69
	Data types	28	Network variables	69
	"Attach to" parameters	28	Printing variables	70
	Events	32	Remote Client variables	70
	Widgets positioning	35	Version variables	71
	Managing overlapping widgets	36	Screen variables	71
	Grouping widgets	36	SD card variables	71
	Changing multiple widgets properties	37	Server variables	72
<b>5</b>	<b>Project properties</b>	<b>39</b>	Time variables	72
	Project properties pane	40	Touch screen variables	73
	Developer tools	42	USB drive variables	74

User management variables .....	74	Exporting alarm configuration .....	121
<b>9 Actions .....</b>	<b>75</b>	<b>13 Recipes .....</b>	<b>125</b>
Alarm actions .....	76	Managing recipes .....	125
Event actions .....	76	Configuring a recipe widget .....	128
MultiLanguage actions .....	77	Recipe status .....	129
Keyboard actions .....	77	Uploading/downloading a recipe .....	129
Page actions .....	79	Backup and restore recipes data .....	130
Print actions .....	82	<b>14 Trends .....</b>	<b>131</b>
Recipe actions .....	84	Data logging .....	132
Remote Client actions .....	87	Exporting trend buffer data .....	133
System actions .....	88	Trend widgets .....	134
Tag actions .....	93	History trends .....	136
Trend actions .....	95	Trend widget properties .....	137
User management actions .....	98	Values outside range or invalid .....	138
Widget actions .....	100	Showing trend values .....	139
<b>10 Using the Client application .....</b>	<b>103</b>	Scatter diagram widget .....	140
The Client application toolbar .....	104	<b>15 Data transfer .....</b>	<b>143</b>
Workspace .....	104	Data transfer editor .....	144
Settings and time zone options .....	104	Exporting data to .csv files .....	146
Transferring files to a remote HMI device ....	105	Data transfer limitations and suggestions ...	146
<b>11 Using the integrated FTP server .....</b>	<b>107</b>	<b>16 Offline node management .....</b>	<b>149</b>
FTP settings .....	107	Offline node management process .....	150
<b>12 Alarms .....</b>	<b>109</b>	Manual offline node management process ...	150
Alarms Editor .....	110	Manual offline configuration .....	150
Remote alarms acknowledge .....	112	Automatic offline node detection .....	151
Alarm state machine .....	112	<b>17 Multi-language .....</b>	<b>153</b>
Setting events .....	113	The Multi-language editor .....	155
Active Alarms widget .....	115	Changing language .....	156
Alarms History widget .....	119	Multi-language widgets .....	156
Managing alarms at run time .....	119	Exporting/importing multi-language strings ..	158
Enable/disable alarms at run time .....	119	Changing language at run time .....	160
Displaying live alarm data .....	120	Limitations in Unicode support .....	160
Exporting alarm buffers to .csv files .....	121	<b>18 Scheduler .....</b>	<b>163</b>



Creating a schedule .....	164	<b>24 Keypads .....</b>	<b>199</b>
HighResolution schedule .....	164	Creating and using custom keypads .....	200
Recurring schedule .....	164	Deleting or renaming custom keypads .....	202
Configuring location for schedules .....	166	Keypad type .....	202
Configuring the Scheduler widget .....	167	Keypad position .....	203
Scheduling events at run time .....	168	<b>25 External keyboards .....</b>	<b>205</b>
<b>19 User management and passwords .....</b>	<b>171</b>	Search and filter .....	207
Enable/disable security management .....	172	Displayed keys .....	207
Configuring groups and authorizations .....	172	Removing action associations .....	207
Modifying access permissions .....	173	Keyboard layout .....	208
Assigning widget permissions from page view .....	177	Enable/disable keyboard .....	208
Configuring users .....	178	Associating actions to keys .....	208
Default user .....	179	<b>26 Tag cross reference .....</b>	<b>211</b>
Managing users at run time .....	179	Updating data in the Tag Cross Reference pane .....	212
Force remote login .....	180	<b>27 Indexed addressing .....</b>	<b>215</b>
<b>20 Audit trails .....</b>	<b>181</b>	Creating an indexed addressing set .....	216
Enable/disable audit trail .....	182	Using indexed tag set in pages .....	219
Configure audit events .....	182	<b>28 Special widgets .....</b>	<b>221</b>
Configure tags for audit trail .....	183	DateTime widget .....	222
Configure alarms for audit trail .....	184	Multistate Image widget .....	222
Configure recipes for audit trail .....	184	Multistate Image Multilayer widget .....	223
Configure login/logout details .....	185	Combo Box widget .....	225
Exporting audit trail as .csv files .....	185	Consumption Meter widget .....	226
Viewing audit trails .....	186	RSS Feed widget .....	228
<b>21 Reports .....</b>	<b>187</b>	Scrolling RSS Feed widget .....	229
Adding a report .....	188	IPCamera widgets .....	229
Configuring text reports .....	188	Browser widget .....	232
Configuring graphic reports .....	189	Control list widgets .....	233
Print triggering events .....	190	Variables widget .....	235
Default printer .....	191	<b>29 Custom widgets .....</b>	<b>239</b>
<b>22 Screen saver .....</b>	<b>195</b>	Creating a custom widget .....	240
<b>23 Backup/restore of Runtime and project .....</b>	<b>197</b>	Adding properties to a custom widget .....	240

Editing custom widgets properties .....	242	System Mode .....	286
<b>30 Sending an email message .....</b>	<b>243</b>	<b>33 Updating system components in HMI devices .....</b>	<b>289</b>
Configuring the email server .....	244	Display information on connected devices ..	290
Configure emails .....	244	List of upgradable components .....	290
<b>31 JavaScript .....</b>	<b>247</b>	Update of system components from the application .....	291
JavaScript editor .....	249	Update system components via USB .....	293
Execution of JavaScript functions .....	249	<b>34 Protecting access to HMI devices .....</b>	<b>295</b>
Events .....	251	Changing password .....	296
Widget events .....	252	Changing password on HMI device .....	296
Page events .....	254	Ports and firewalls .....	297
System events .....	255	<b>35 Factory restore .....</b>	<b>299</b>
Objects .....	257	<b>36 Tips and tricks to improve performance .....</b>	<b>301</b>
Widget class objects .....	257	Static Optimization .....	302
Widget properties .....	258	FAQ on Static Optimization .....	305
Widget methods .....	260	Page caching .....	306
Page object .....	262	Image DB .....	306
Page object properties .....	262	Precaching .....	306
Page object methods .....	263	FAQ on precaching .....	306
Group object .....	265	<b>37 FAQ .....</b>	<b>309</b>
Group object methods .....	265	Changing fill color property according to tag values .....	309
Project object .....	266	<b>38 CP600-eCo products .....</b>	<b>311</b>
Project object properties .....	266	The Runtime loader .....	312
Project object methods .....	266	Limitations .....	314
State object .....	275	Converting projects between different HMI types .....	317
State object methods .....	276	<b>39 Communication protocols .....</b>	<b>319</b>
Keywords .....	277	ABB CODESYS Ethernet .....	320
Global functions .....	277	ABB Mint Controller HCP .....	328
Handling read/write files .....	278	ABB Modbus RTU .....	332
Limitations in working with widgets in JavaScript .....	280	ABB Modbus TCP .....	339
Debugging of JavaScript .....	281	ABB Pluto .....	345
<b>32 System Settings tool .....</b>	<b>285</b>		
User Mode .....	285		

BACnet .....	351
ABB CODESYS Serial .....	360
CODESYS V2 Ethernet .....	367
Modbus RTU .....	377
Modbus RTU Server .....	387
Modbus TCP .....	393
Modbus TCP Server .....	404



# 1 Getting started

---

PB610-B Panel Builder 600 is a software application designed to create graphical HMI pages. PB610-B Panel Builder 600 has a drag-and-drop interface that makes it easy to create complex pages. Many of the features found in common Windows applications are also available in PB610-B Panel Builder 600.

This document is divided into chapters that describe the key functions of PB610-B Panel Builder 600 and explain how to use them. Each chapter is presented in a standalone manner, allowing you to jump from chapter to chapter, depending on the task at hand.

---

<b>Assumptions</b>	<b>2</b>
<b>Installing the application</b>	<b>2</b>

# Assumptions

We assume that readers of this manual are using the PB610-B Panel Builder 600 software to design control panel applications that run on CP600 panels and on computers running Windows.

We also assume that readers have a basic understanding of computers, Microsoft Windows, and the specific network environment where the application will run.

## Installing the application

PB610-B Panel Builder 600 installation contains:

- PB610-B Panel Builder 600: an application for designing custom HMI projects in a user-friendly manner, along with a variety of objects in its built-in library, the Widget Gallery.
- HMI Client: a light-weight application that can be used on Windows computers to remotely view and manage a project running on an HMI device.
- HMI Runtime: a standalone application that runs on the HMI devices. The HMI Runtime is installed via PB610-B Panel Builder 600.

## PB610-B Panel Builder 600 system requirements

PB610-B Panel Builder 600 has the following system requirements:

<b>Operating System</b>	Windows XP (SP2 or SP3)
	Windows Vista Business/Ultimate
	Windows 7
	Windows 8
<b>Storage</b>	500 MB Minimum
<b>RAM</b>	512 MB
<b>Other</b>	One Ethernet connection

## Installing multiple versions of PB610-B Panel Builder 600

You may install different instances of PB610-B Panel Builder 600 on the same computer. Each installation has its own settings and can be uninstalled individually.

Three installation scenarios are possible:

Installation scenario	Results
<b>First installation of PB610-B Panel Builder 600 in the system</b>	Software is installed in the specified destination folder
<b>System with only one instance of PB610-B Panel Builder 600 already installed</b>	Current version can be replaced or maintained.
<b>System with multiple instances of PB610-B Panel Builder 600 already installed</b>	Last version installed can be replaced or maintained.

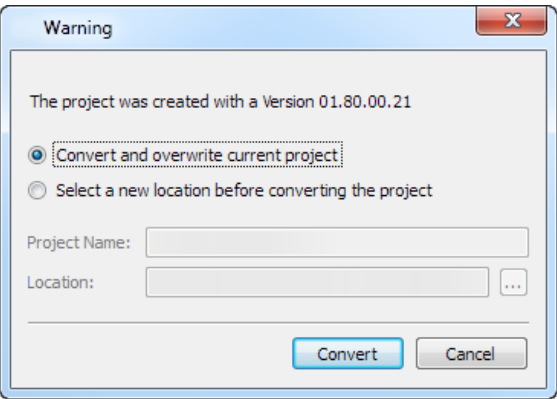
If you try to install a second instance of an already installed version of PB610-B Panel Builder 600, a warning message is displayed.

Multiple PB610-B Panel Builder 600 installations share a common workspace folder, each sub-folder includes the version number, for example *C:\Program Files\ABB\Panel Builder 600 Basic Suite1.90*. Each installed version has its ID and can therefore be removed individually.

Each installation is listed separately in the Windows **Start** menu.

## Opening older projects

When opening a PB610-B Panel Builder 600 project (.jpr file) created with an older version of the software PB610-B Panel Builder 600 asks to convert the project to the current version:



Option	Description
Convert and overwrite current project	The project is converted without a backup copy of the original version
Select a new location before converting the project	The project is converted and the older version is copied to the specified folder.



**WARNING:** Do not edit projects with a version of PB610-B Panel Builder 600 older than the version used to create them. This will damage the project and may cause runtime instability.

## Multilanguage for PB610-B Panel Builder 600

PB610-B Panel Builder 600 is available in multiple languages. All languages are installed by default as part of PB610-B Panel Builder 600.

The default language is English. To change it go to **Help > Change Language**.

## Crash reports

A crash report dialog appears whenever PB610-B Panel Builder 600 freeze or crash.



**Important:** Always save crash report files since they may contain useful information for technical support.



**Note:** Crash reports are unavailable in Windows XP.





# 2 Runtime

---

HMI Runtime is designed to support different platforms and different operating systems.

---

<b>Runtime modes</b> .....	<b>6</b>
<b>HMI device basic settings</b> .....	<b>6</b>
<b>Context menu options</b> .....	<b>7</b>
<b>Built-in SNTP service</b> .....	<b>10</b>

# Runtime modes

The HMI Runtime is composed of two logic units:

- **Server:** run communication protocols, collect data, monitor alarms, drive trend buffer sampling.
- **Client:** display data collected by server.

The server unit is responsible for handling the HMI services such as the communication protocols, performing data acquisition, driving trend buffer sampling activities, monitoring alarms, and so on.

The client unit is the part which is responsible for the visualization process: use the data collected by the server to render it on the display as graphical information.

The server unit works in two operating modes:

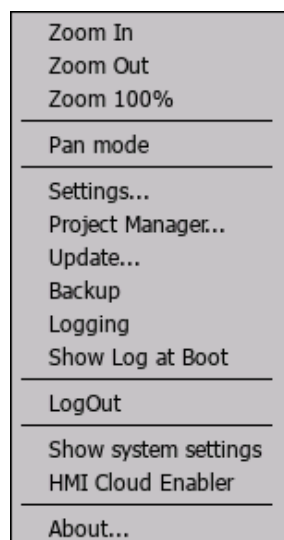
- **Configuration mode:** server is idle (for example when no project is loaded on the device or some system files are missing).
- **Operation mode:** server is operating according to the settings defined by the system files and by the loaded application project.



Note: Data on client may be displayed even if no activity is running on the server.

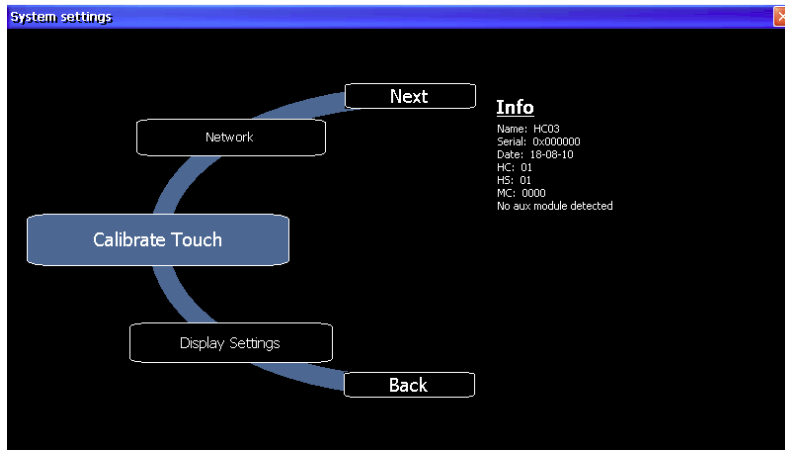
## HMI device basic settings

On the HMI device press and hold on an empty area of the screen for a few seconds to display the context menu.




If no runtime is installed on the device click the dedicated button on the device when in loader mode. See "[The Runtime loader](#)" on page 59 for details.

1. From the context menu, select **Show system settings**: the System settings menu is displayed.



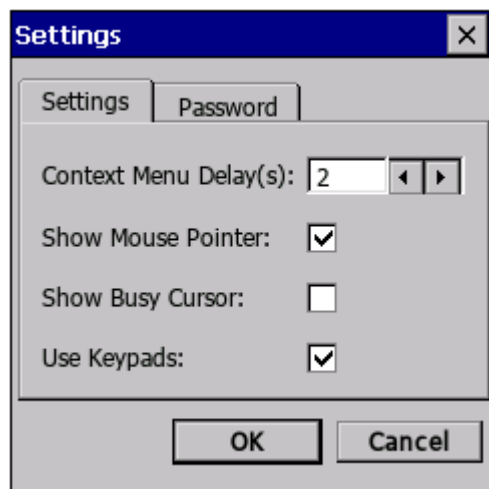
2. Click **Back/Next** to scroll the rotating menu.

Menu	Function
<b>Calibrate Touch</b>	Calibrate the touch screen when needed
<b>Display settings</b>	Control backlight inactivity timeout and brightness
<b>Time</b>	Set HMI device date and time manually or configure NTP servers
<b>BSP Settings</b>	Display operating system version and unit operating timers to control buzzer and battery led.
<b>Network</b>	Set IP address
<b>Plug-in List</b>	List the plug-in modules installed and recognized by the system.  Note: this option may not be supported by all platforms and all versions.

## Context menu options

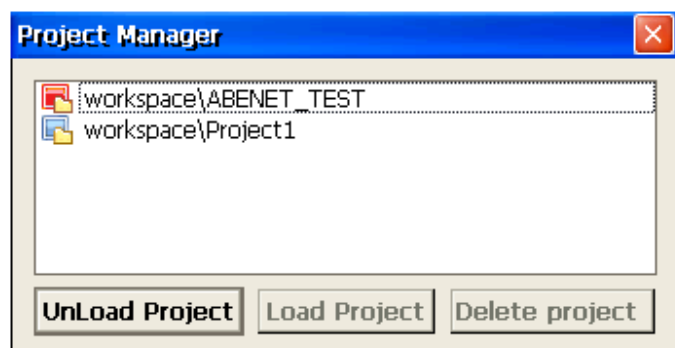
Option	Function
<b>Zoom In/Out/100%</b>	Select view at run time
<b>Pan Mode</b>	Enables/disables pan mode after a zoom in

## Runtime settings



Main parameters	Description
<b>Context Menu Delay</b>	Context menu activation delay. Range: 1–60 seconds.
<b>Show Busy Cursor</b>	Display an hourglass when the system is busy
<b>Use keypads</b>	Display keypads when user touches a data entry field. Set to <b>disable</b> when an external USB keyboard is connected to the device.
<b>Password</b>	Define password protected operations amongst the following: <ul style="list-style-type: none"> <li>• Download Project/Runtime</li> <li>• Upload project</li> <li>• Board management (BSP Update)</li> </ul> See " <a href="#">Protecting access to HMI devices</a> " on page 295 for details.

## The Project Manager



This tool allows you to:

- unload the current project
- load another project
- delete a project.

When you load a new project, the current project is automatically unloaded. You must unload a project before you can delete it.

## Update

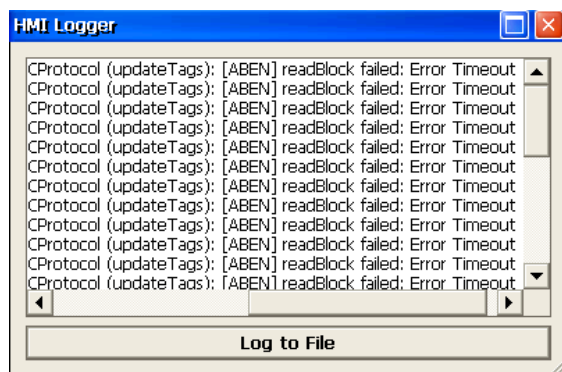
This function loads update packages from an external USB drive. See ["Update system components via USB" on page 293](#) for details.

## Backup

You can create a backup copy of the Runtime and of the project.

## Logging

This function displays a log of system operations.



In this example the log reports a communication error.

Click **Log to file** to save data: a logger.txt file is saved to the ...\\var\\log folder.

This file can be retrieved using an FTP Client and forwarded to technical support.



Note: Once enabled, logging is maintained after power cycles and must be manually disabled.

## Show log at boot

This function enables the logger at start up. If the **Log to file** option has been enabled, log files are saved from startup.

## Developer tools

Utility functions for debugging at run time.

## About

This function shows information about the Runtime version.



**WARNING:** Context Menu action has no effect if executed from a dialog page.

## Built-in SNTP service

The HMI device features an integrated SNTP that synchronizes the internal real-time clock panel whenever the predefined server is available.

The system searches for the following servers when turned on, or once a week if the HMI device is not turned off:

- time.windows.com
- tock.usno.navy.mil



**Important:** Server addresses are hard-coded and cannot be changed by the user.

## Customizing SNTP servers

*Path: from the context menu > **System Settings**> **Time**> **SNTP***

*Availability: BSP v1.76 ARM / 2.79 MIPS or higher*

You can customize up to two SNTP servers.



**Note:** This function is not available in Configuration Mode (ConfigOS).

# 3 My first project

---

This section describes how to create a simple PB610-B Panel Builder 600 project.

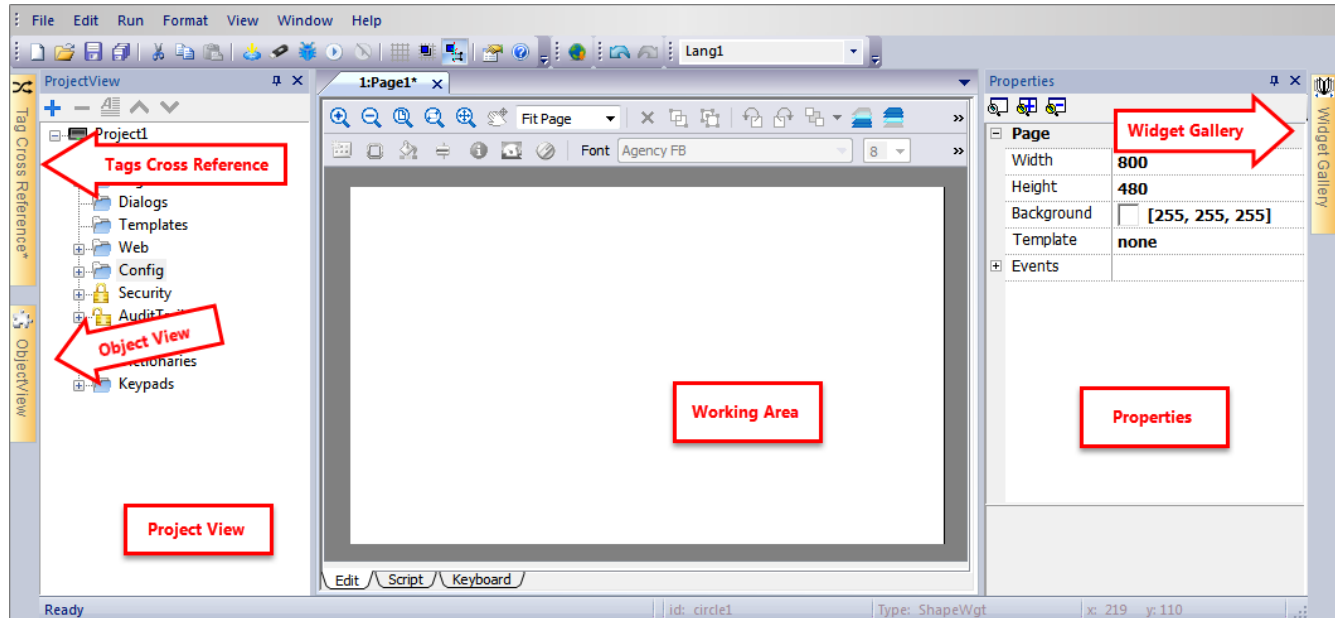
---

<b>The workspace .....</b>	<b>12</b>
<b>Creating a project .....</b>	<b>12</b>
<b>Communication protocols .....</b>	<b>14</b>
<b>Designing a page .....</b>	<b>15</b>
<b>The Widget Gallery .....</b>	<b>17</b>
<b>Adding tags .....</b>	<b>19</b>
<b>Exporting tags .....</b>	<b>21</b>
<b>Importing tags .....</b>	<b>21</b>
<b>Attaching widget to tags .....</b>	<b>24</b>
<b>Dialog pages .....</b>	<b>26</b>


# The workspace

## Workspace areas

PB610-B Panel Builder 600 workspace is divided into the following main areas:



Area	Description
<b>Project View</b>	Project elements in hierarchical project tree.
<b>Object View</b>	Tree view of widgets organized by page.
<b>Working Area</b>	Space where pages are edited. Tabs at the top of the area show all open pages.
<b>Properties</b>	Properties of selected object.
<b>Widget Gallery</b>	Library of graphic objects and symbols.
<b>Tag cross reference</b>	List of locations where a given tag is referenced.

 Note: The workspace layout can be changed at any time, changes are saved and maintained through working sessions.

## Resetting the workspace layout

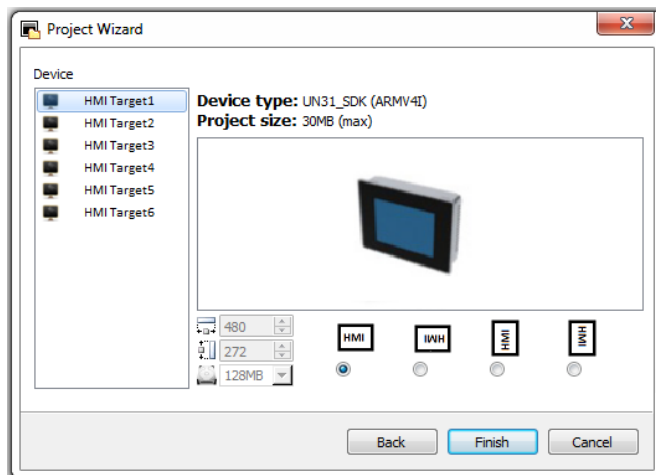
To restore the default layout, use the **File > Reset and Restart** function.

## Creating a project

*Path: File> New Project*



1. In the **Project Wizard** dialog enter a name for the project and the storage location.
2. Click **Next**: the HMI device selection dialog is displayed.



3. Choose one device from the list of the available models.
4. Choose device orientation.
5. Click **Finish** to complete the Wizard.

## Portrait rotation exceptions

The following elements are not rotated in portrait mode.

Element	Description
Operating system dialogs	System settings and system dialog
ContextMenu and related dialogs	Project Manager, About, Settings, Logging, Backup
Video	Analog Video Input, IPCamera, MediaPlayer
JavaScript	Alert and Print function
Dialog pages	“Title” of dialog pages
Scheduler	Dialogs for data entry
Macro	ShowMessage, LunchApplication, LunchBrowser
External applications	

## Changing the device model

Once you have developed your project you can still change the device model, from the Project Properties pane. This will not resize the widgets, but will relocate them on the screen. A warning will be displayed if some objects cannot be relocated.

Project Widget : Project1	
Id	Project1
Full Path	
Version	
Context Menu	on delay
Developer Tools	false
Buzzer on touch	false
Buzzer duration (ms)	200
Image DB Enable	true
Plug-in	
Behavior	
Home Page	Page1.jmx +
PageWidth	800
PageHeight	480
Display Mode	Landscape +
Project Type	HMI +
Panel Memory	128MB +
PageRequest	+ +

## Copying, moving, renaming a project

PB610-B Panel Builder 600 projects folder contain all the files of the project: to move, copy or backup a project, move or copy the project folder to the desired location.

To rename a project use the **File > Save Project As** function: this operation might take a few minutes.



**WARNING:** Do not rename the project folders manually.

## Communication protocols

*Path: ProjectView> Config > Protocols*

Device communication drivers are configured in the **Protocol Editor**. You can add up to the maximum number of protocols as specified in Table of functions and limits. Variable and System Variables are not counted as protocols.

See "[Communication protocols](#)" on page 319 for more details.



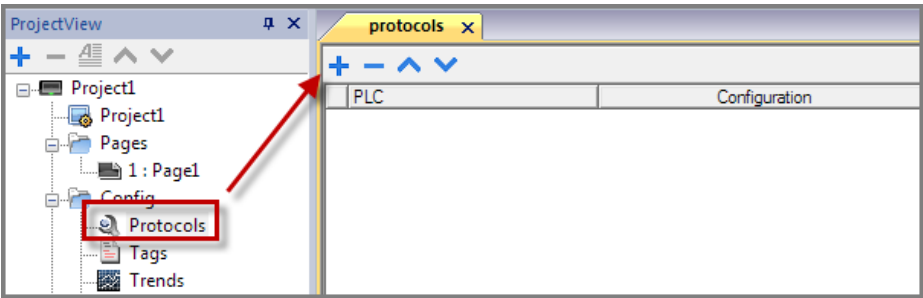
Note: you can run different Ethernet protocols over the same physical Ethernet port, but you cannot run different serial protocols using the same serial port. Some serial protocols support access to multiple controllers, but this option is set within the protocol itself which is still counted as one protocol.

## Adding a protocol



Note: Refer to CP600 operating instructions manual in case you need cables information.

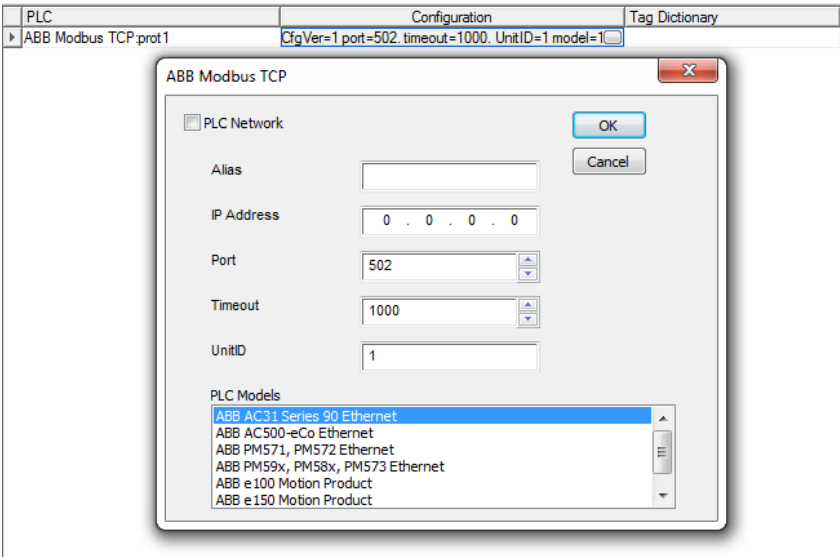
1. Click +.



2. Select the protocol from the PLC list and enter the required values.

## Changing protocol settings

To change configuration parameters, click the browse button in the **Configuration** column.



## Protocol parameters

Click **Show Advanced Properties** icon to see all parameters.

Parameter	Description
Dictionaries	Tags imported for the protocol. See <a href="#">"Importing tags" on page 21</a> for details.
Enable Offline AlgorithmOffline Retry Timeout	See <a href="#">"Automatic offline node detection" on page 151</a> for details.
Version	Protocol version available in PB610-B Panel Builder 600 for selected HMI device.

## Designing a page

Path: **ProjectView** > **Pages**

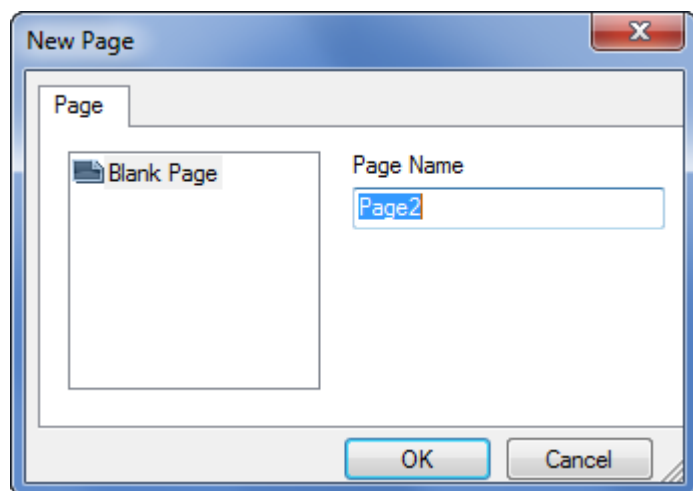
When a project is created, the first page is automatically added and shown in the **Page Editor**.

## Adding objects to a page

Drag and drop objects from **Widget Gallery** to the page.

## Adding a page

1. Right click the **Pages** node from the project tree and select **Insert new page**.
2. Type a name for the new page.



## Importing a page

When importing a page PB610-B Panel Builder 600 will import the page layout and the page widgets without importing the actions and data links attached to widgets. You can choose between two different behavior:

- importing only the pages and the widgets: in this case all actions and data link have to be defined
- importing pages with references to actions and data links: used tags must be present in the project for these elements to work properly



Note: Page import can only be performed between projects made using the same software version. Save the older project as the newer version, then try again.

1. Right click the **Pages** node from the project tree and select **Import page**.
2. Choose the page to be imported from the desired project then click **OK**: a warning message is displayed.
3. Click **Yes** to remove all the links to data and actions. Click **No** to maintain the reference to data links and actions. Tags need to be available in the new project.

## Group of pages

You can group similar pages for easier maintenance. Grouping pages does not affect how pages appears at run time. To create a group of pages:

1. In **ProjectView** right click **Pages** node and select **Create Group**: a new folder is added
2. To move a page to a group, right click a page and select **Groups > groupName**.

# The Widget Gallery

**Path:** *View> Toolbars and Docking Windows> Widget Gallery*

HMI objects required to build an application are available in the **Widget Gallery**. The gallery is divided into several categories, each containing a collection of widgets.



## Adding a widget to a page

1. Select the widget from the **Widget Gallery**.
2. Drag and drop it on the page.

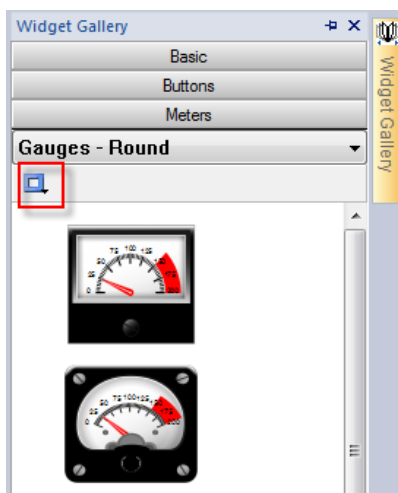
## Changing the appearance of a widget

All widgets have properties (**Properties** pane) that can be changed. Some widgets are presented in various styles. You can click the buttons in each category to see available styles.

### Example

To set the widget style for round gauges:

1. Click the style button to display the available styles for the widget.



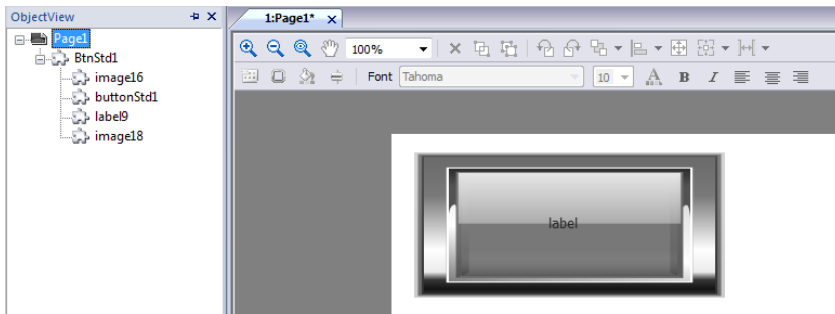
2. Select one of the available styles from the toolbar: depending on the selected widget, different options are available.



## Complex widgets

Some widgets are composed of many sub widgets. For example, a button is a complex widget composed by a button widget and a label. The structure of widgets can be seen in the **ObjectView** when the widget is selected.

You can select a sub-widget, such as the label in a button, from the **ObjectView** and modify it without ungrouping the whole widget.



## Adding tags

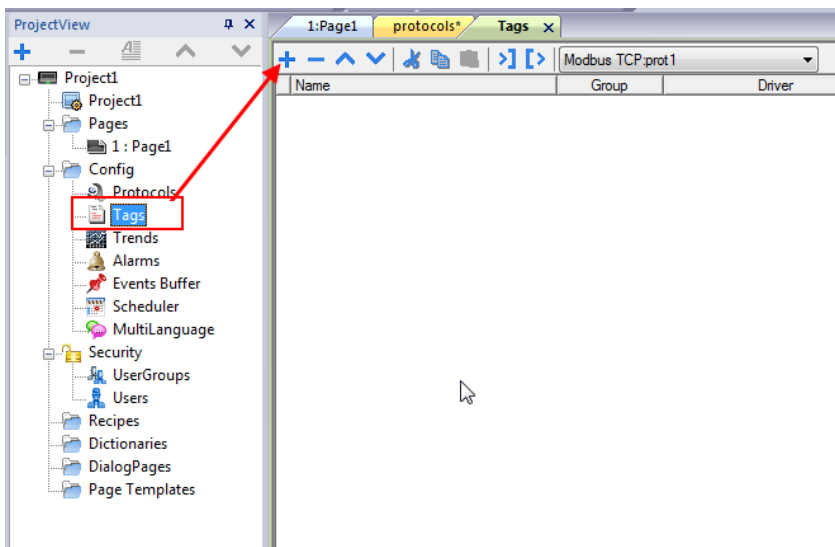
PB610-B Panel Builder 600 uses tag names to access all device data. All fields and reference locations in the device need to be assigned a tag name to be used in the HMI project.

Tag Editor can be used to create and manage tags. After the tags have been defined, they can be used in the project by attaching them to widgets' properties.

See [""Attach to" parameters" on page 28](#) for details.

## Tag editor

**Path:** *ProjectView > Tags*







## Adding a tag

1. Click + and enter the required data.
2. Select the Address from the communication protocol address dialog: new tags are named Tag1, Tag2, ....
3. Click on the tag name to rename it.

## Tag properties

See specific protocol documentation for details.

Property	Description
<b>Name</b>	<p>Unique tag name at project level. Primary key to identify information in the runtime tag database.</p> <p> <b>WARNING: Duplicate tag names are not allowed.</b></p>
<b>Group</b>	Group name associated to a tag
<b>Driver</b>	Communication protocol
<b>Address</b>	<p>Controller memory address.</p> <p>To edit click on the right side of the column to get the dialog box where you can enter the address information.</p>
<b>Encoding</b>	Encoding type for string data type (UTF-8, Latin1, UTF-2 and UTF-16)
<b>Comment</b>	Tag description
<b>Rate (ms)</b>	<p>Tag refresh time. Default: 500ms.</p> <p> <b>WARNING: Tags refresh rate is the maximum refresh rate. Actual refresh rate depends on: communication type (serial, fieldbus, Ethernet), protocol, amount of data exchanged.</b></p>
<b>R/W</b>	<p>R/W tag attribute (R/W, R or W).</p> <p> Note: The content of Write Only tags is always written and never read. When communication is not active, the content of these tags may not be available in widgets.</p>
<b>Active</b>	<p>Update mode.</p> <p><b>false</b> = tags are read from controller only when required by the HMI device.</p> <p><b>true</b> = tags are continuously read even if not required by the displayed page.</p> <p> <b>Important: Leave this value set to false for higher communication performance.</b></p>
<b>Simulator</b>	Tag behavior during simulation. Several profiles are available.
<b>Scaling</b>	<p>Conversion applied to tag before database storage.</p> <p><b>By formula</b> = defined as a linear transformation.</p> <p><b>By range</b> = defined as a range conversion.</p>
<b>PLC Tag Name</b>	<p>Original PLC tag name, used to match tags used by HMI application (Tag Name) and tags exported from PLC</p> <p>R/W only in advanced view to allow for adjustments in case tag import errors.</p>



## Managing tag names

Tag names must be unique at project level. If the same tags, from the same symbol file have to be used for two different controllers, use the “Alias” feature to add a prefix to the imported tags and make them unique at project level.



Note: Not all protocols support the “Alias” feature.

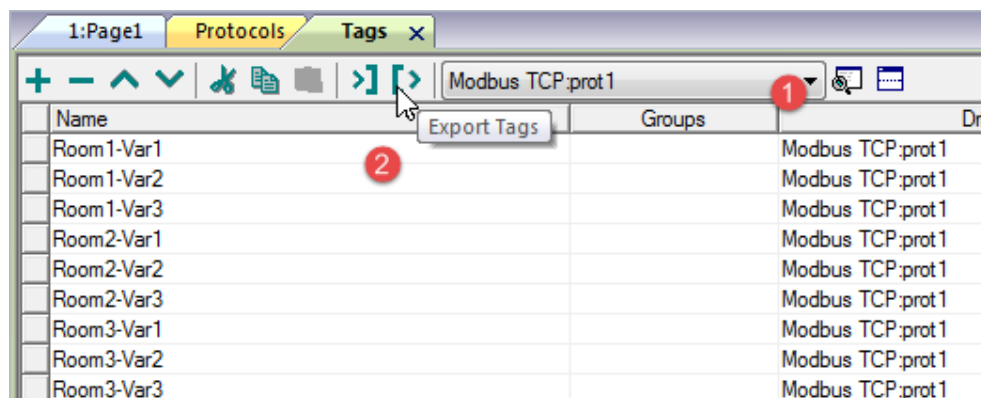
See ["Communication protocols" on page 319](#) for details.

## Managing tag groups

Tags used in each page are identified as part of a group, so that requests made by the communication protocol to the connected controller(s) can be processed faster: only the tags included in the displayed page are polled from the controller.

## Exporting tags

Path: **ProjectView > Tags**



1. Select the protocol for the tags you want to export.
2. Click the **Export Tags** button: all the tags configurations for the selected protocols are exported into an .xml file.

You can edit the resulting .xml file using third part tools (for example, Microsoft Excel) and then re-import the modified file (see ["Importing tags" below](#) for details).

## Importing tags

### Introduction

Some protocols allow you to import tags stored in a comma separated file (.csv or other formats). Refer to the Tag Import section of each protocol for details (see ["Communication protocols" on page 319](#)).

Importing is a two step process:

1. Import of the tag definition into a dictionary
2. Import tags from the dictionary to the project



**WARNING:** Special characters in tag names such as “&” character cause communication errors. See ["Limitations in Unicode support" on page 160](#)



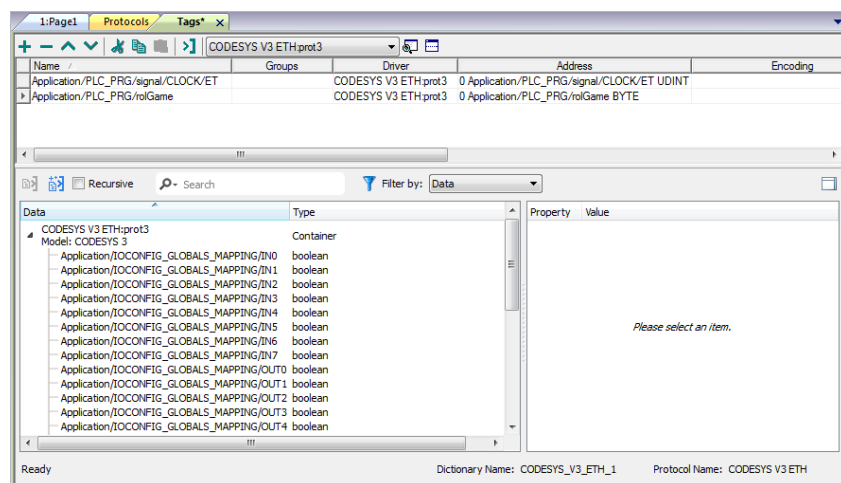
Note: When importing tags, character "." in tag names is replaced with "/" . The protocol will use the correct syntax when communicating to the PLC.

## Dictionaries

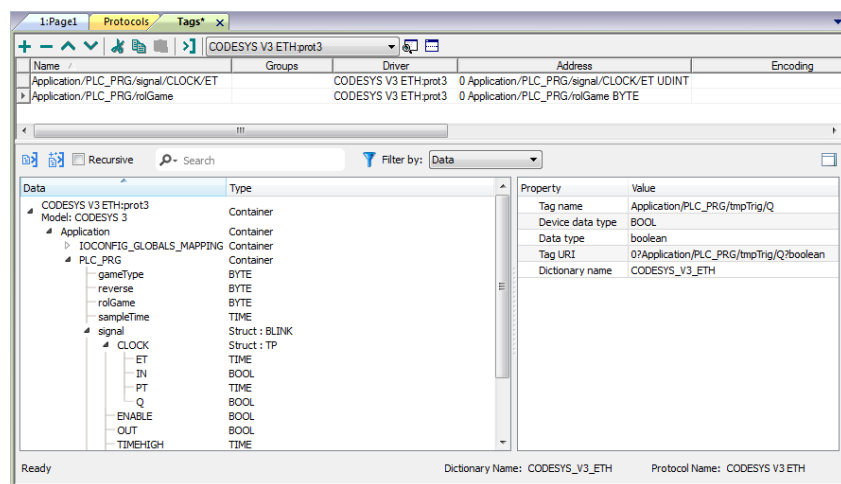
Path: **ProjectView > Dictionaries**

A dictionary is a list of tags imported in the Tag Editor for a specific protocol. Depending on the protocol type, tags are shown in linear view or in hierarchical view.

### Linear view



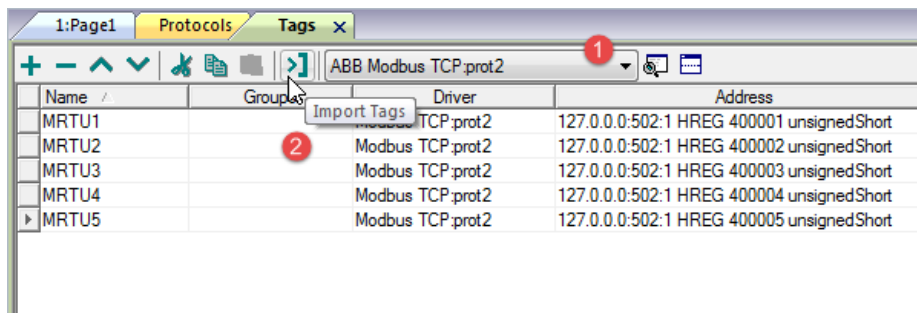
### Hierarchical view



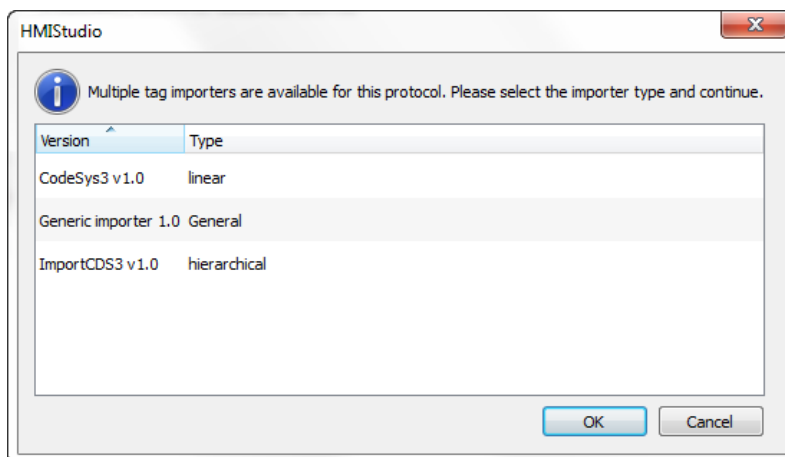
## Importing tags

To import tags from an external file:

1. In **ProjectView, Tags** select the protocol from the filter list.



2. Click the **Import Tags** button: the select file dialog appears. A dialog to choose the importer type appears.



3. Select the file: a list of tags is shown in a linear or hierarchical view.
4. To import tags, select one or more tags or a node (hierarchical view only) and click the **Import tag** button: tags are copied to the project and listed in the upper window section.

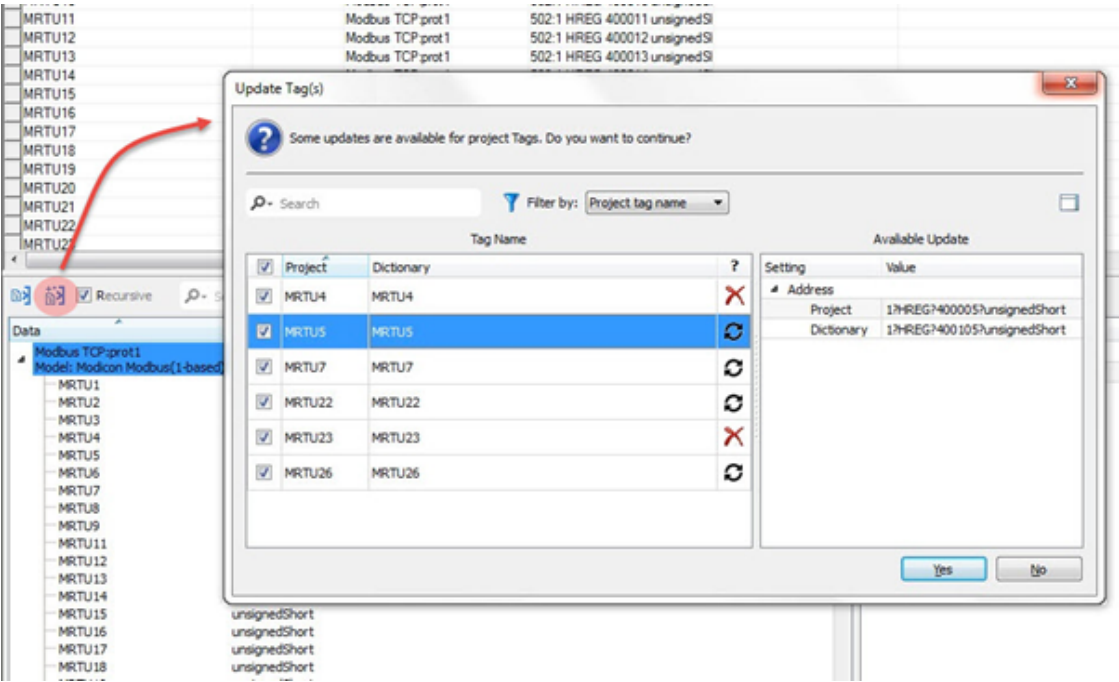
Parameter	Description
<b>Recursive</b>	All elements of the structure are imported into separate tags.



Note: When the project is configured to use a protocol network you must also select the protocol node where tags are to be imported. You can import the same tags on multiple protocols. When the tags file contains the node information, you can choose to use the information to filter the tags and import only those matching with the selected nodes.

## Updating the imported tags

Using the Update Tag(s) command you can re-import tags. A dialog allows you to select the tags to be reimported:



These tags need to be updated. A list of differences between project and dictionary is displayed.



These tags are no longer available in the dictionary. If updated, these tags will be removed from the project.

# Attaching widget to tags

To control a widget and animate it through live data it is possible to bind a specific property to different data sources. For example it is possible to bind the gauge **Value** property to a probe temperature tag, or the **Display** property to a recipe data

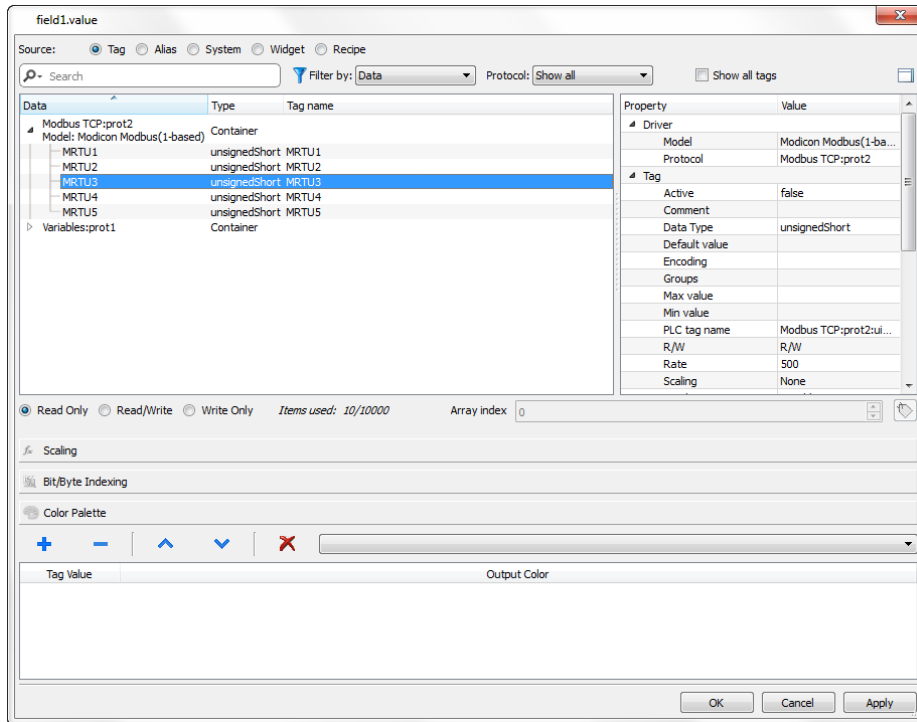
## Data sources

Elements to which an object property can be attached:

Data source	Description
Tag	Tag defined in the Tag Editor
Alias	Indexed tag address
System	Predefined system tags (see "System Variables" on page 63)
Widget	Connect to a widget property (for example, value of a slider widget)
Recipe	Data from the Recipe Manager (see "Recipes" on page 125)

## Attaching a property to a tag

1. Click **+** in the **Properties** pane.
2. In **Source** choose the data source, in the list choose a protocol and the tag. Use the **Search** box to filter tags.



3. Set the access type (for example **Read Only**). The **Array Index** field appears when the selected tag is an array to identify the element of the array to use. The indirect index mode, through an additional tag, is supported.
4. Click **OK** to confirm.

See [""Attach to" parameters" on page 28](#) for details.

## Communication Error

Two icons may appear close to widgets that have an attached tag.



- : communication error
- : data not yet available (slow communication protocol)

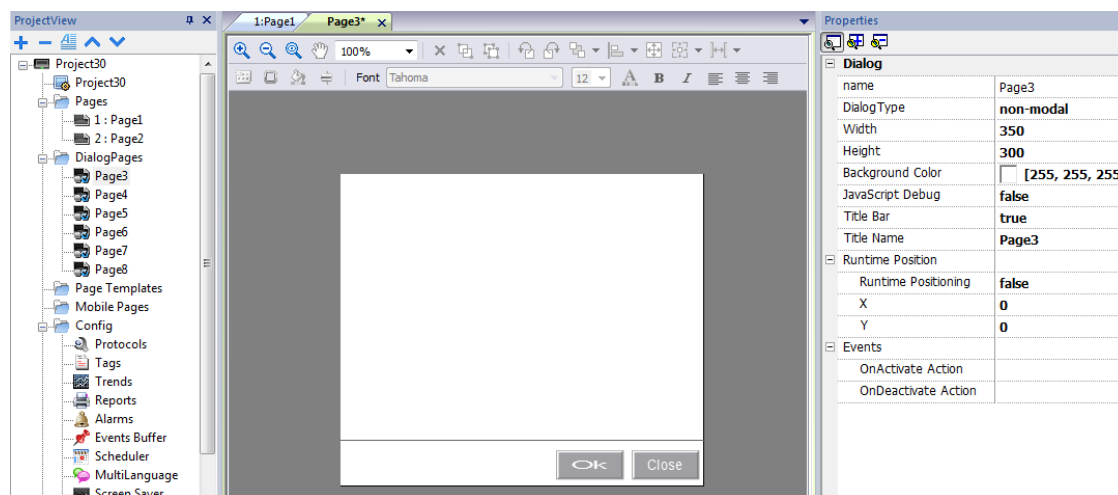
# Dialog pages

Path: **ProjectView** > **Web** > **Dialogs**

Dialog pages are opened at run time on top of the current page on project request. They are used to notify alarms, errors or to require user action.

## Main dialog properties

Property	Description
<b>Dialog Type</b>	<b>modal</b> = user cannot return to main project window/page until dialog is closed. <b>non-modal</b> = user can continue to use main project window (or other non-modal dialogs) while a dialog is shown on top of it.
<b>Title Bar</b>	<b>true</b> = dialog title displayed <b>false</b> = no dialog title displayed
<b>Title Name</b>	Dialog title. Only if <b>Title Bar</b> =true.
<b>Runtime Position</b>	Dialog fixed position <b>false</b> = Dialog will be placed centered on the screen <b>true</b> = Dialog will be placed with upper-left corner at position X and Y



## Maximum number of dialogs

When the maximum number of open dialogs is reached, the oldest dialog is closed to open the new one.

# 4 Programming concepts

---

Programming for PB610-B Panel Builder 600 is based on a few basic concepts and behaviors.

---

<b>Data types</b> .....	<b>28</b>
<b>"Attach to" parameters</b> .....	<b>28</b>
<b>Events</b> .....	<b>32</b>
<b>Widgets positioning</b> .....	<b>35</b>
<b>Managing overlapping widgets</b> .....	<b>36</b>
<b>Grouping widgets</b> .....	<b>36</b>
<b>Changing multiple widgets properties</b> .....	<b>37</b>

# Data types

When creating a tag you have to specify its properties. Data type are specific to PB610-B Panel Builder 600, memory type are specific to the selected protocol. Choose the value according to the internal representation you need for the selected controller address.



Note: arrays type use the same data type followed by "[ ]" (i.e.: boolean [ ])

Data Type	Description
<b>boolean</b>	One bit data (0..1)
<b>byte</b>	Signed 8 bit data (-128..127)
<b>double</b>	IEEE double-precision 64-bit floating point type (2.2e-308 ... 1.79e308)
<b>float</b>	IEEE single-precision 32-bit floating point type (1.17e-38 ... 3.40e38)
<b>int</b>	Signed 32 bit data (-2.1e9 ... 2.1e9)
<b>short</b>	Signed 16 bits data (-32768..32767)
<b>string</b>	Characters coded according to selected format
<b>time</b>	Time data
<b>unsignedByte</b>	Unsigned 8 bit data (0..255)
<b>unsignedInt</b>	Unsigned 32 bit data (0 ... 4.2e9)
<b>unsignedShort</b>	Unsigned 16 bit data (0..65535)
<b>uint64</b>	Unsigned 64 bit data (0...264 – 1)

## "Attach to" parameters

### Object properties

In PB610-B Panel Builder 600 the properties of an object placed on a page can be set at programming time or configured to be dynamic. To change a property at programming time use the page toolbar or the property pane. Select the object first to see its properties displayed.



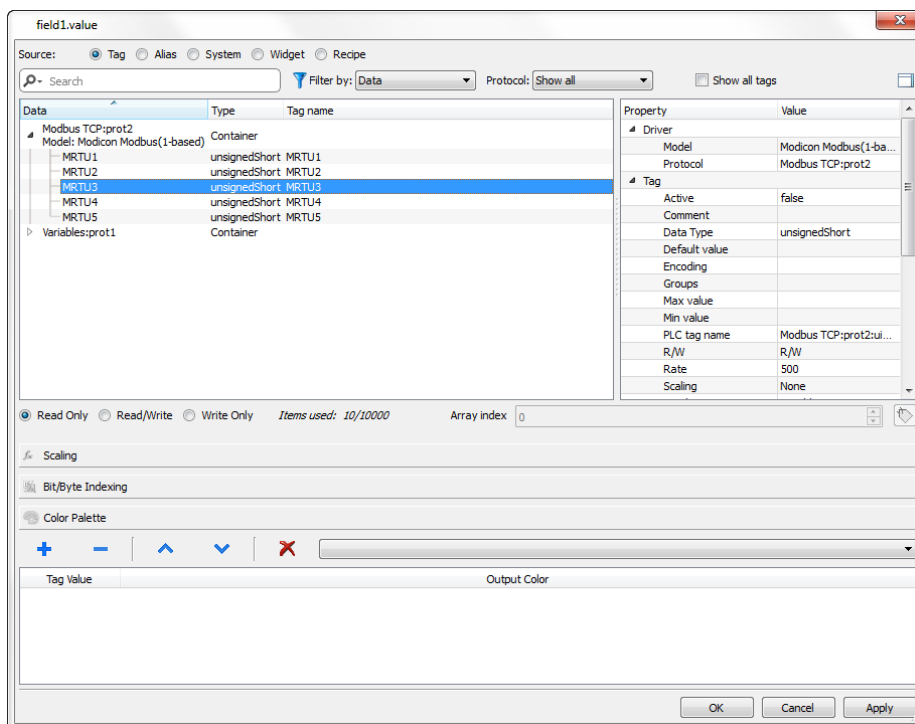


The page toolbar shows only the most common object properties, while the property pane shows all the properties in a basic or advanced view.

To change a property value dynamically you can attach it to tags or variables.

## Attaching a property to a tag

1. Click **+** in the **Properties** pane.
2. In **Source** choose the data source, in the list choose a protocol and the tag. Use the **Search** box to filter tags.



3. Set the access type (for example **Read Only**). The **Array Index** field appears when the selected tag is an array to identify the element of the array to use. The indirect index mode, through an additional tag, is supported.
4. Click **OK** to confirm.

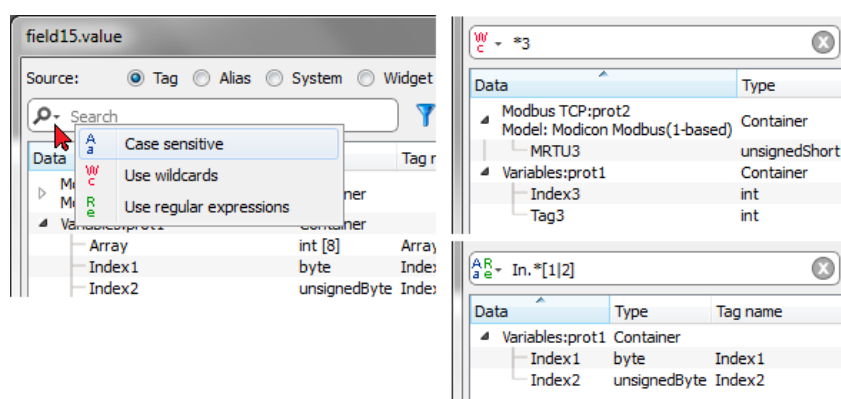
## Data sources

Elements to which an object property can be attached:

Data source	Description
<b>Tag</b>	Tag defined in the Tag Editor
<b>Alias</b>	Indexed tag address
<b>System</b>	Predefined system tags (see <a href="#">"System Variables" on page 63</a> )
<b>Widget</b>	Connect to a widget property (for example, value of a slider widget)
<b>Recipe</b>	Data from the Recipe Manager (see <a href="#">"Recipes" on page 125</a> )

## Advanced search

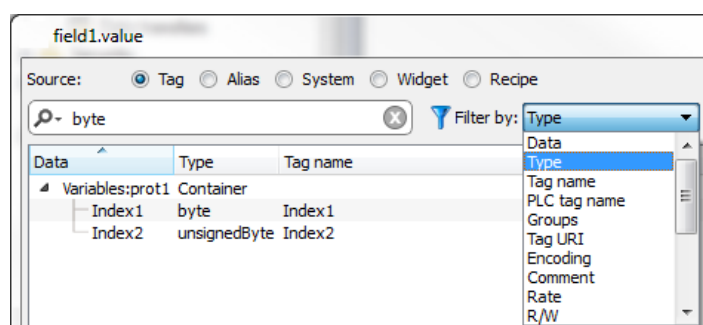
Various syntax options can be applied to search box:



Main options	Function
<b>WildChars</b>	Search using simple wildcards matching . Character '?': matches any single character. Character '*': matches zero or more of any characters. "[...]": sets of characters can be represented in square brackets.
<b>Regular Expression</b>	Describes character pattern. See <a href="http://www.regular-expressions.info/">http://www.regular-expressions.info/</a>

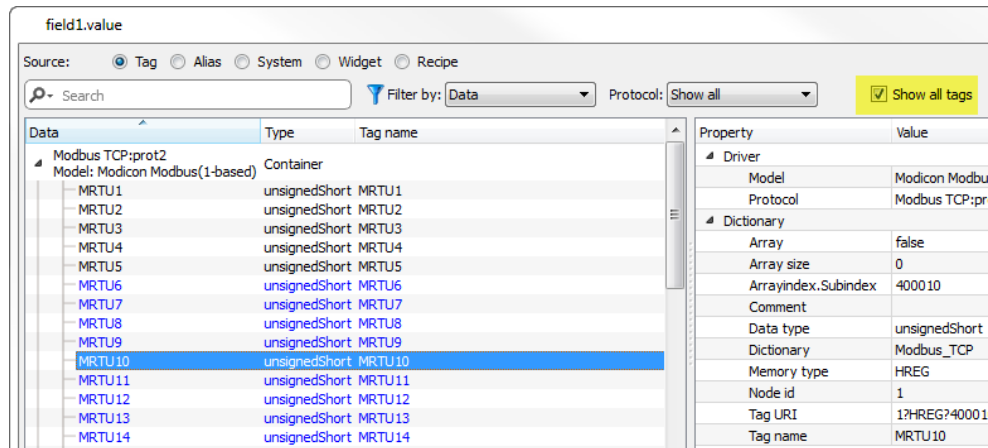
## Filtering tags

Choose various tag filter criteria:

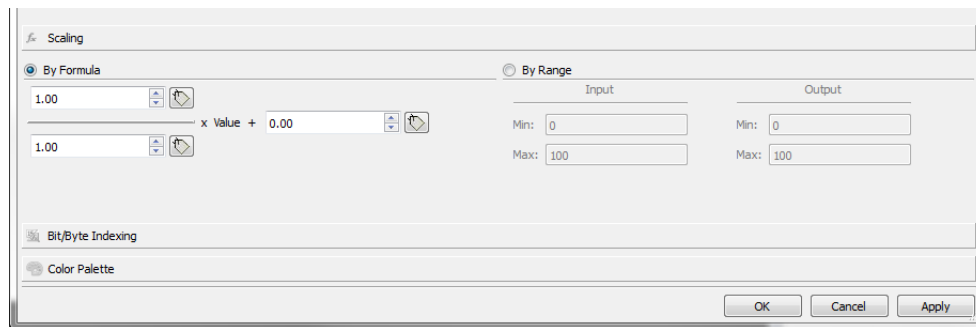


## Showing dictionary tags

When **Show all tags** is checked, tags that belong to one dictionary but have not been imported yet, appear in blue color. You can select and double-click a tag to import it into the project.



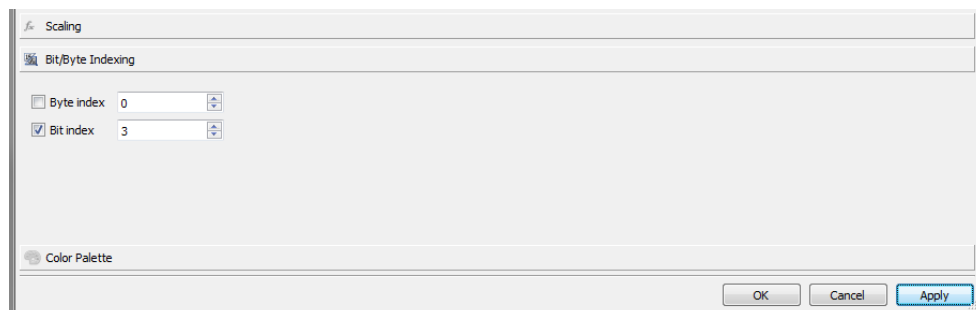
## Converting tag value



**Scaling** tab converts the tag value. In **By Range** section set the input and output range: the system will automatically calculate the scaling factors.

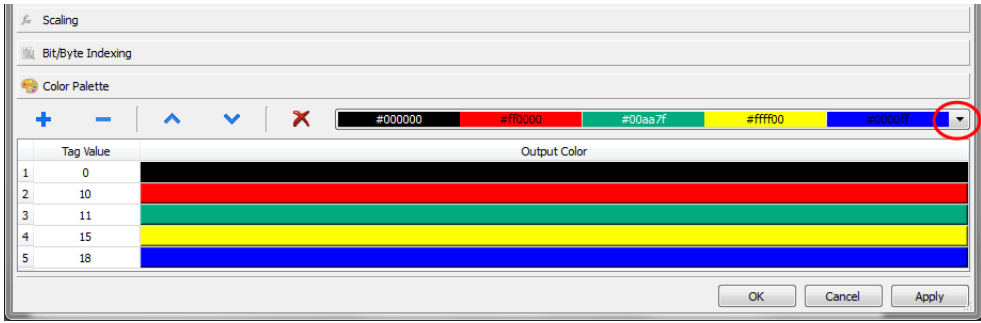
## Extract tag bit/byte based on index

Allows extracting a single bit or byte content from a word depending on the specified bit or byte number



## Mapping tag values to color

Allows you mapping numeric tag values to colors. You can use this option to change the color of a button.



Section	Function
	From the toolbar add/remove or move up/down the colors lines. The tag value is editable and you can modify the sequence values.
	Last defined color combination is saved automatically and can be retrieved from the color toolbar.

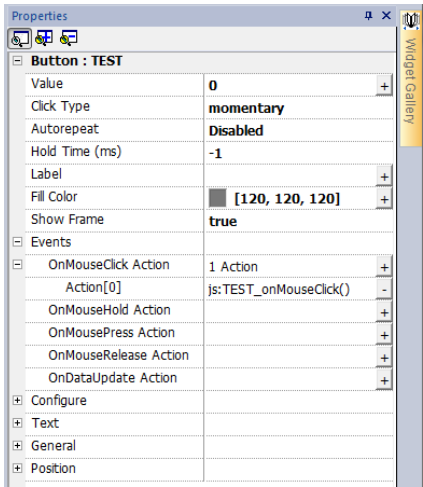
# Events

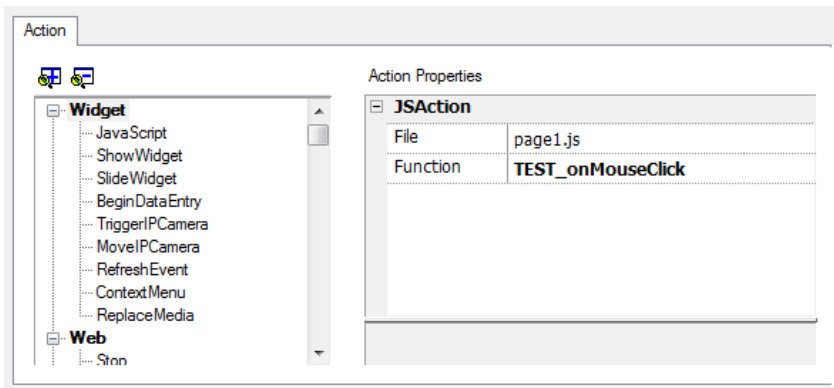
Events are used to trigger actions at project level and can be associated to:

- buttons / touch (click, press, release)
- external input devices like keyboards and mouse (click, press, hold, release, wheel)
- data changes (OnDataUpdate)
- switch of pages (OnActivate, OnDeactivate)
- alarms
- scheduler

You can attach one or more actions to an event, so that they will be executed whenever the event occurs.

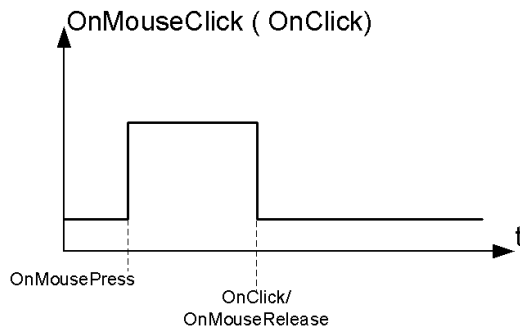
This example shows a JavaScript action activated by pressing a button.





## OnClick / OnMouseClicked

Triggers the event when the button/key is pressed and released quickly.



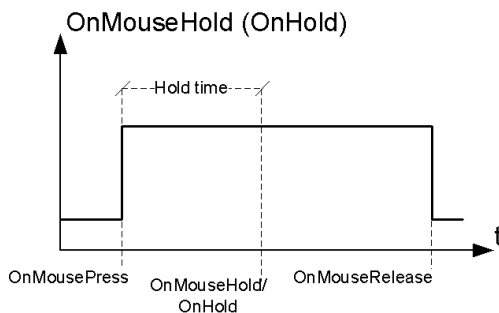
## OnHold/OnMouseHold

Triggers the event when the button/key is pressed and held pressed for a certain time set as **Hold Time** in the widget properties. Actions programmed for this event will be executed only after the hold time has expired.

The default **Hold Time** is configured in Project properties but can be redefined for each button/key. See "[Project properties](#)" on page 39.



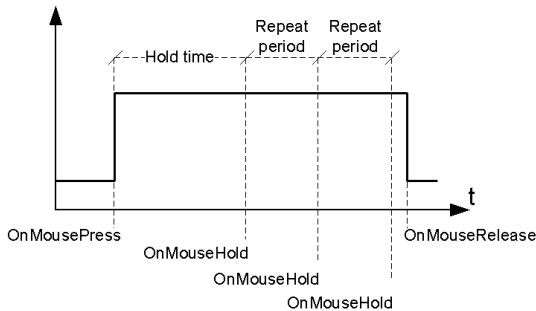
Note: If **Hold Time** is set to -1 for the widget, the project **Hold Time** value will be used.



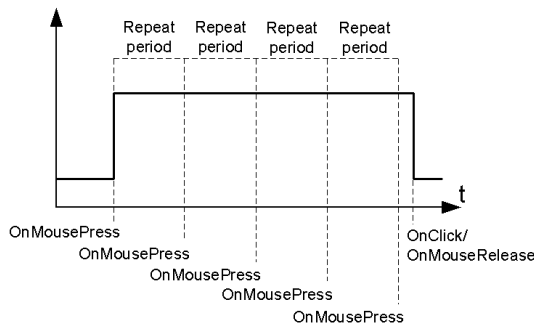
## Autorepeat

Enables auto repeat for a press or hold event of button or key. **Autorepeat Time** is specified in the Project properties but can also be redefined for each button or key

OnMouseHold (OnHold) and Autorepeat



OnMousePress and Autorepeat



## OnWheel

Triggers the event when a wheel (for example a USB mouse wheel) value changes. A wheel usually is used to increase/-decrease values in a text box or attached to a tag.

## OnActivate

Triggers the event when a page is loaded. The event starts before widgets in the page are initialized.

## OnDataUpdate

Triggers the event when the tag value changes. The update moment depend on the time needed by the protocol to finish the update process. For example the **OnDataUpdate** event can be triggered or not, depending on whether data becomes available from protocol respectively after or before widgets being initialized for the first time. In particular, page change notifications are more likely to happen with slow protocols and remote clients.



Note: The value read during **OnActivate** can be the same obtained from a subsequent **OnDataUpdate** event, since **OnDataUpdate** notifications are sent asynchronously.

## Widgets positioning

You can position widgets in the page using two methods:

- Snap to Grid
- Snap to Object

To display the grid, on the **View** menu, click **Show Grid**.

### Snap to Grid

*Path: View> Snap to Grid*

When you move or re-size an object, its top left corner will align with the nearest intersection of lines in the grid, even if the grid is not visible.

### Setting grid properties

*Path: View> Properties*

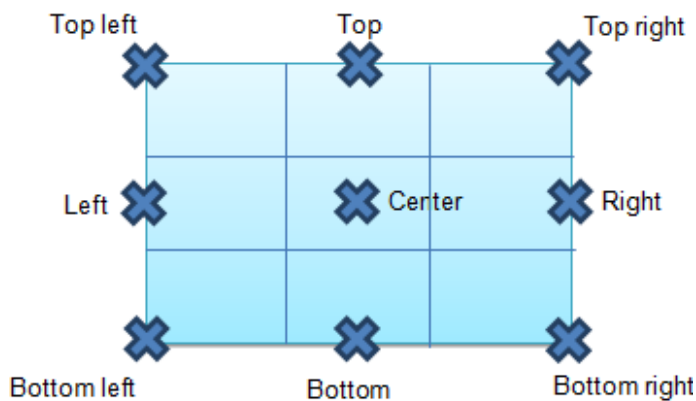
Parameter	Description
Spacing X	Space in pixel between two lines/dots on the X axis
Spacing Y	Space in pixel between two lines/dots on the Y axis
Type	Grid type (dot or line)
Color	Grid color

### Snap to Object

*Path: View> Snap to Object*

When you move an object, it will align with other objects on the page.

When you select an object, one of the following hot points is selected as the source of the snap point, depending on the area you pressed: top, top left, top right, bottom, bottom left, bottom right, left, right, center:

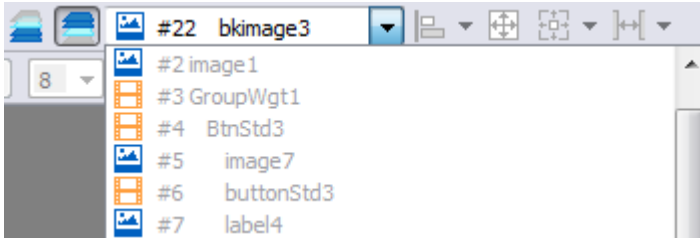


An algorithm finds a matching hot point among the near widgets hot points matching either the x or the y coordinates of the source snap point. For line widgets, the source snap points are the terminal points of the line.

# Managing overlapping widgets

When one or more widgets on the page overlap, you can manage their order so that one is displayed on top of the other.

The order of the widget on the page is shown in the combo box. A widget with greater z-order number is in front of an element with a lower z-order number. A picture icon identifies static objects, a movie frame icon identifies dynamic objects.



**Important: Correct ordering of widgets is essential for run time performance since overlapping dynamic widgets can invalidate static optimization and reduce performance of HMI applications.**

## Hiding/showing widget on z-order

To hide widgets above a selected widget:

- On the toolbar click  and select a widget: all widgets above this one are hidden

To hide widgets below a selected widget:

- On the toolbar click  and select a widget: all widgets below this one are hidden

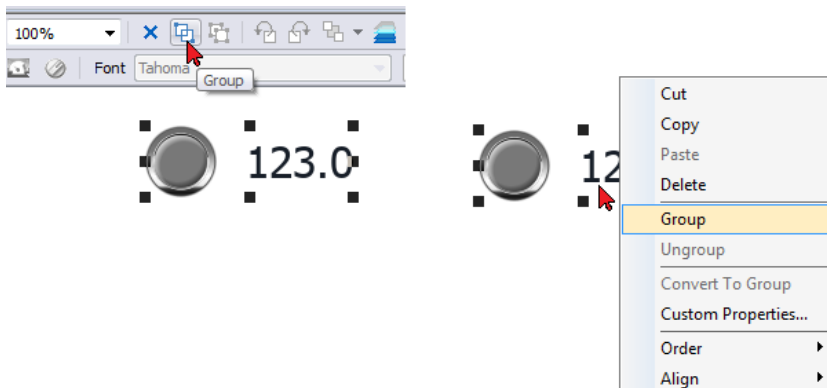
The toolbar allows to:

- hide widgets stacked above and/or below selected widgets
- work on different widgets using the combo box which lists all the widgets in their z-order.

## Grouping widgets

To group widgets:

- Select all the widgets to group.
- Right-click and then click **Group**.

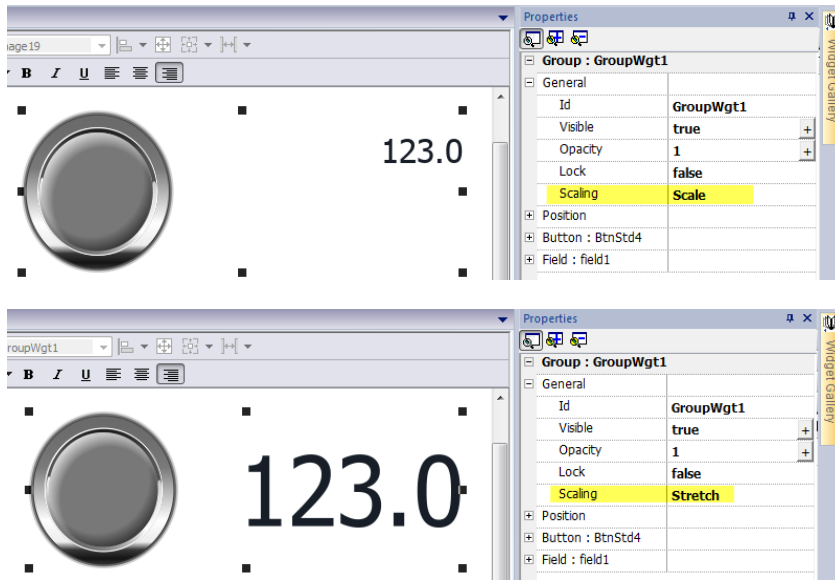




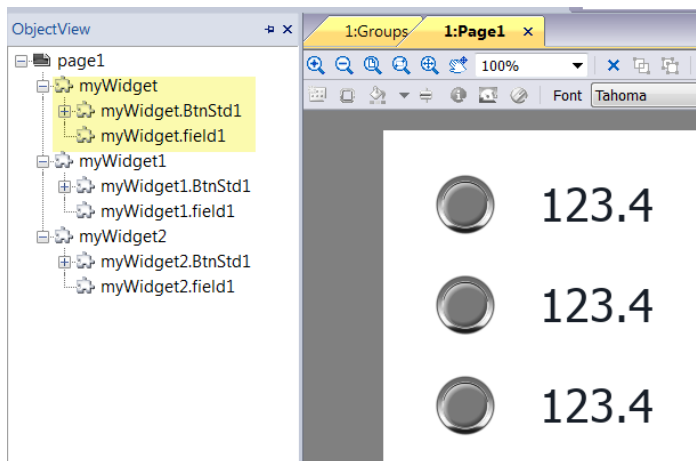
## Resizing grouped widgets

You can define how object reacts when re-sized. Use the **Scaling** property in **General** section:

- **Scale**: object and text are not re-sized proportionally
- **Stretch**: object and text are re-sized proportionally



Tip: Rename the components of a group of object using the same prefix followed by a point character (for example, **myWidget.**). This because when a group of objects with the same prefix is found, PB610-B Panel Builder 600 replicates the same prefix when the object is copied. This is very useful when the object needs to be referenced from JavaScript code.

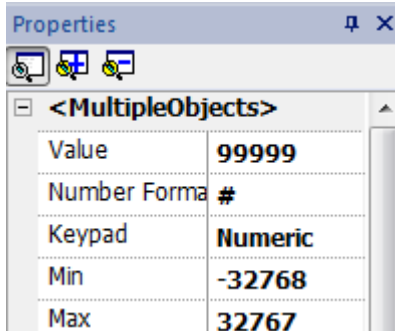


## Changing multiple widgets properties

You can set the properties of more widgets of the same category (label, field, gauge and so on) all at once.

To change properties:

1. Select widgets.
2. Set common properties from **Properties** pane.
3. When multiple widgets are selected, the Properties pane title changes to **<MultipleObjects>**: all changes will be applied to all selected widgets.



Note: Not all properties can be modified for multiple widgets simultaneously and must therefore be modified individually.

# 5 Project properties

---

Project properties contain settings for the project.

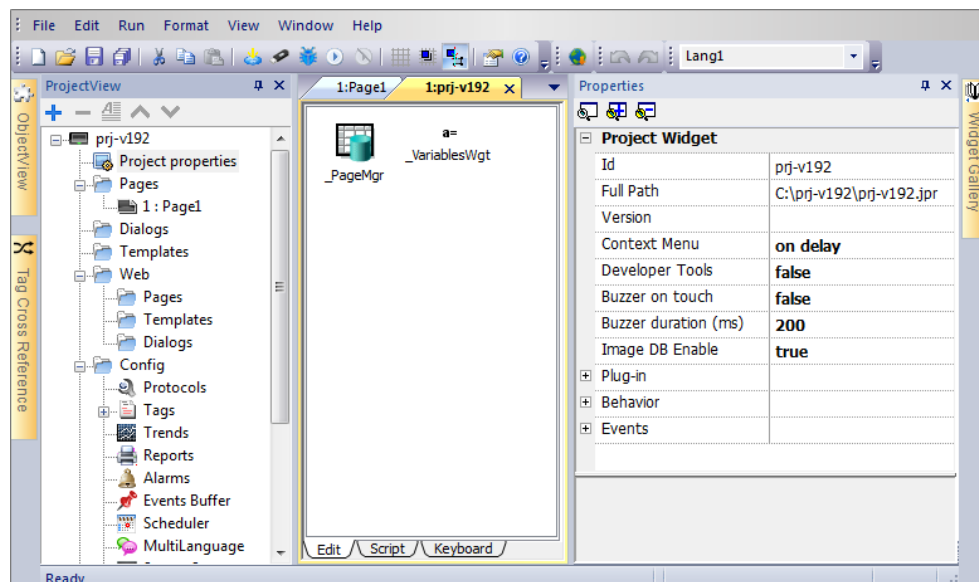
---

<b>Project properties pane</b>	<b>40</b>
<b>Developer tools</b>	<b>42</b>
<b>FreeType font rendering</b>	<b>45</b>
<b>Behavior</b>	<b>45</b>
<b>Events</b>	<b>48</b>

# Project properties pane

Path: **ProjectView**> double-click **Project properties**> **Properties** pane

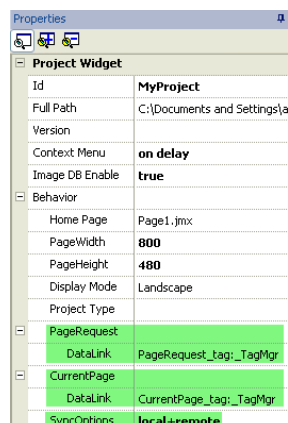
The project **Properties** pane contains a list of project level user-configurable data.



## Basic and advanced properties

To view all project properties:




- Click **Show Advanced Properties** button to expand the property view in the **Properties** pane.



## Main properties description



Note: Some properties are displayed only in advanced mode.

Property	Description
<b>Version</b>	OS and Runtime version.  Version of Main OS, for example, UN30HSxx60M0166. Version of Runtime, for example, 1.90 (0) – Build (682)
<b>Context Menu</b>	Define how context menu should appear in the HMI project.  <b>on delay</b> = context menu appears touching/pressing and holding for a few seconds an empty area of the runtime screen, or via <b>Context menu</b> action  <b>on macro command</b> = context menu appears only via <b>Context menu</b> action.  See " <a href="#">Widget actions</a> " on page 100 for details.
<b>Developer Tool</b>	Enable/disables a collection of runtime debugging utility tools.
<b>Buzzer on Touch</b>	Enables buzzer when touching the runtime screen. It can be used as an alternative to the touch buzzer feature available in Windows CE side that gives feedback when the user touches anywhere on the touch-screen. Buzzer on touch is supported also by Win32 Runtime.  <i>Available for Windows CE from v1.76 ARM / 2.79 MIPS.</i>
<b>Buzzer duration</b>	Default 200 ms
<b>Keyboard</b>	Enables the use of keyboard macros at run time when using external keyboards.
<b>JavaScript Debug</b>	Enables the JavaScript debugger at run time for the current project.   Note: For UN20 HMI devices (Windows CE MIPS HMI devices), local debugger is disabled. However, a remote JavaScript debugger is available to be used from a computer connected to the HMI device via Ethernet.
<b>Allow JS Remote Debugger</b>	Enables JavaScript remote debugger for current project.   Note: Remote debugging not supported on HMI Client.
<b>Image DB enable</b>	Activates an engine used by the Runtime to optimize project performance.   <b>WARNING: This property should only be disabled by technical support for debugging purposes since this might reduce performance at run time.</b>
<b>FreeType Font Rendering</b>	Switches to FreeType the font rendering used by PB610-B Panel Builder 600 and runtime.
<b>Behavior</b>	These properties define different aspects of page behavior. See " <a href="#">Behavior</a> " on page 45

## Buzzer

A buzzer can be associated to the following widgets:

- buttons
- hotspots
- needles
- fields
- external keys
- combo boxes
- tables items
- control list items

## Developer tools

Collection of runtime debugging functions that can be enabled or disabled.

### Enabling developer tools

1. In **Properties** pane, set **Developer Tools** to **true**.
2. Download the project.
3. Open context menu.
4. Select **Developer tools**.

### Developer tool list

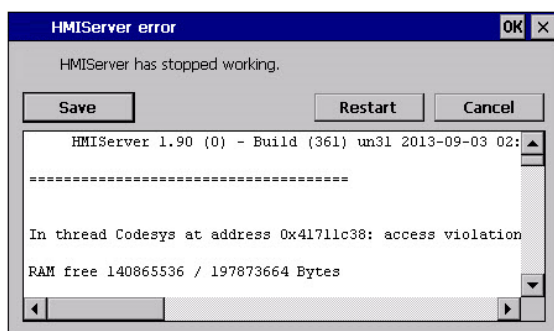
Tool	Description
<b>Show/Hide all</b>	Shows a dialog containing information about device status like CPU load, memory usage, event queues.
<b>CPU statistics</b>	Shows information on CPU load. See <a href="#">"CPU Statistics" on the facing page</a> .
<b>Memory statistics</b>	Shows information about system RAM . A negative value indicates that free memory is decreasing.
<b>Event queues</b>	Shows information on event queues (size, maximum achieved size, number of processed events, last and maximum processing time). Timing statistics are only available for non-UI queue.
<b>Timelog summary</b>	Show page loading time.
<b>Embed window</b>	Allows embedding in runtime the scene or leave the developer tool window as a standalone window (dialog).
<b>Reset queue stats</b>	Resets statistical information on event queues.
<b>Disable watchdog</b>	Disable the watchdog function and prevents system restart in case of freeze or crash of services.
<b>Ignore</b>	Disables crash report function, exceptions are not saved in the crash report window.

Tool	Description
exceptions	
Profiling	Measures the time spent for loading/rendering the active page. See <a href="#">"Profiling" on the next page</a>

## Watchdog

This feature allows you to disable the watchdog. This way you can avoid system restart in case of a runtime crash and have the time to save the crash report or check system status information (for example, memory available, CPU load, events queue size and so on).

The crash report dialog is displayed automatically in case of a system freeze or crash allowing users to save a log file of crash.



**Important: Save this file for technical support.**

## CPU Statistics

```

2014-04-25 23:02:48, up: 0:08:27, idle: 24 *
Period 2110 ms (overhead 69ms)
  Thread      ID Prio    ms kernel/  user
*      59637774   3    697    0/    697
  Codesys 78839810   0     8    0/     8
Other threads < 5ms
RAM free 125833216 / 194211840 Bytes (diff: 0)
ImageDB size ~2MB, free 4MB / RAMSIZE-76MB)
Page Preload 56MB free / RAMSIZE-64MB)
Page Cache 80MB free / RAMSIZE-40MB)
Storage free 45 / 92 MB

  EvQueue  Size  MaxSize  Evts    ms  max(ms)
  EvMgr    0      0        0     0      0
  ActionMgr 0      1      61    22    189
  AlnMgr    0      0        0     0     0
  MODR     0      0    122    11    15
  UI        0     11    270    --    --

Timelog is disabled!
(Tap-tap to change position)

```

On the top row the current machine time is shown along with the total device uptime.

CPU statistics are collected with a frequency of 2000 milliseconds. The actual period and the overhead required to collect and visualize statistics are displayed as well. The more the actual period is far from the nominal 2000 milliseconds the higher is the system load. CPU consumption of threads is listed reporting the name of the thread (if available, main thread is marked with a \*), the thread ID, the thread priority and CPU time spent during the 2000 milliseconds period, divided in user and kernel time.

## Profiling


Profiling allows you to check time spent for loading/rendering the active page. Profiling will start from the next page load and will be active only for the first painting of the page to the screen (the configuration is retained).

```
2014-04-25 23:27:19, up: 0:32:58, idle: 36 *
Period 2053 ms (overhead 47ms)

Page "Alarms.jmox":
      START      dT (ms/cpuMs)
Time parsing    : +    5      45/    45
Time unloading  : +   54      6/     6
Time lst update : +  195      3/     0
Time gfx creation: +  198    300/   133
      OnLoad    :      241/    94
Time rendering  : +  535    390/   387
ImageDB cache 15 hit/0 miss(0 ms, cpu: 0 ms)

Page "TemplatePagel.jmox":
Time init/start : +   60    133/    86
Time lst update : +  195      2/     0
Time gfx creation: +  459    27/    27
      OnLoad    :      9/     9
ImageDB cache 28 hit/0 miss(0 ms, cpu: 0 ms)

(Tap-tap to change position)
```

Profiling option	Description
<b>Save timelog to file</b>	<p>Saves a report of profile details and the time spent loading a project and its pages into a .txt file. This file can be exported and shared for further analysis.</p> <p> <b>Important: The execution of this function may reduce page change performance.</b></p>
<b>Timelog summary</b>	Data on the current page and template, if any, is displayed using the context menu. See <a href="#">"Timelog data" below</a> .
<b>Overlay OnLoad/ Rendering times</b>	This view allows displaying time spent on single widgets and is available only for the rendering and OnLoad steps. The view gives an immediate feeling of where time is spent. Red zones represent the most time critical zones. Detailed widget times are visualized by a tooltip window (on Win32 platform attached to mouse over event, on Windows CE press drag and release over the region of interest). In case of out-of-the-scene widgets some arrows allow to navigate to these areas and hovering on them the tooltip will show the area summary

## Timelog data

Data	Description
<b>Time parsing</b>	Time spent parsing current page. Depends on page complexity/number of widgets.
<b>Time gfx creation</b>	Time spent for image rendering. Mainly related to the <i>Onload</i> method.
<b>Time rendering</b>	Time spent rendering the page.
<b>Time unloading</b>	Time spent unloading the page, if current page depends from another page.



Times are provided in couples: wall time/CPU time. Wall time is the absolute time required by this part which can be higher than the actual CPU time required since higher priority threads are also running (for instance protocols). The start time column refers to the page load start time. It can be used to track the actual time required to load a page, since partial times only refer to the most time critical functions and do not include other times that often contribute significantly to the total time.

For example, the actual total wall time required to load a page is rendering (which is the last step) start time + rendering wall time.

## FreeType font rendering

All projects created with PB610-B Panel Builder 600 v1.90 (b608) or newer use the FreeType font engine as default. Projects created with older versions of PB610-B Panel Builder 600 use an older font engine also after project conversion to avoid any backward compatibility issue.



Switch to FreeType whenever possible for better page rendering.

Once you have switched to the new font rendering, save the project and verify that all texts are displayed correctly in all project pages.

### Font rendering issues

When switching to the FreeType font engine a project created with the older font engine, you may experience the following problems:

- text requires more/less pixels for rendering thus changing text layout
- widgets are resized to accommodate text
- better rendering can be obtained using antialiasing. Antialiasing is a text widget property and it can be disabled from v1.90 onwards.

## Behavior

These properties define various elements of page behavior.

### Home Page

The first page loaded at run time (after log-in page if security is enabled in project).

When security is enabled, you can specify a different homepage for each groups of users. In this case this setting is ignored. See ["User management and passwords" on page 171](#) for details.

### Page Width/Page Height

Defines the default size in pixel of an HMI page. Default is the display resolution of the HMI device model selected when creating the project.

### Display Mode

Defines HMI device orientation.

## Project Type

Defines HMI device type for the project. According to the model, some project features and properties are automatically adjusted.



**WARNING:** Starting from v2, the HMI Runtime will check if the selected project type is matching with the HMI device model and will advise with a message when the selected type is not matching: "HMI Type mismatch. Convert project and download again."

## Panel Memory

Size of the available internal panel memory.

## PageRequest, CurrentPage and SyncOptions

It is possible to have HMI Runtime exchange devices information on the page shown by the HMI. You can synchronize pages shown on the HMI device and on HMI Client or to control an HMI project from a controller such as a PLC.

The following properties can be customized:

Property	Description
<b>PageRequest</b>	Page to be shown on the HMI device and on HMI Client. Attached tag must contain an integer value within the range of the available project pages and must be available at least as a Read resource.
<b>CurrentPage</b>	Page number displayed on the HMI device or on HMI Client or on both. Attached tag must be available at least as a Write resource and must have integer data type.
<b>SyncOptions</b>	Synchronization of project pages with the value contained into the <b>CurrentPage</b> property. Options can be: <ul style="list-style-type: none"> <li>• <b>disable</b>: page number value is ignored,</li> <li>• <b>local</b>: page number displayed on HMI,</li> <li>• <b>remote</b> : page number displayed on HMI Client.</li> <li>• <b>local + remote</b>: page number displayed on HMI and on HMI Client, if different pages are displayed the last page loaded is considered.</li> </ul>

### Example: forced page change from controller/PLC to HMI device and HMI Client

Set properties as follows:

<b>PageRequest</b>	attached to tag "A"
<b>CurrentPage</b>	empty
<b>SyncOptions</b>	disable

Set value of tag "A" to display the requested page on HMI device and HMI Client.

### Example: forced page change from controller/PLC to HMI and HMI Client. Read current page loaded on HMI

Set properties as follows:

<b>PageRequest</b>	attached to tag "A"
<b>CurrentPage</b>	attached to a tag "B" as read/write
<b>SyncOptions</b>	local

Set value of tag "A" to display the requested page on HMI device and HMI Client. Tag "B" will contain the number of page currently shown by the device.

**Example: forced page change from controller/PLC to HMI device and HMI Client. Read current page loaded on HMI Client.**

Set properties as follows:

<b>PageRequest</b>	attached to tag "A"
<b>CurrentPage</b>	attached to a tag "B" as read/write
<b>SyncOptions</b>	remote

Set value of tag "A" to display the requested page on HMI and HMI Client. Tag "B" will contain the number of page currently shown by HMI Client.

**Example: forced page change from controller/PLC to HMI device and HMI Client. Force HMI Client page synchronization with HMI device (not vice versa).**

Set properties as follows:

<b>PageRequest</b>	attached to a tag "A" as Read/Write
<b>CurrentPage</b>	attached to the same tag "A" as per <b>PageRequest</b>
<b>SyncOptions</b>	local

Set value of tag "A" to display the requested page on HMI and HMI Client. Change page on HMI to display the same page on HMI Client.

**Example: forced page change from controller/PLC to HMI device and HMI Client. Force HMI page synchronization with HMI Client (not vice-versa).**

Set properties as follows:

<b>PageRequest</b>	attached to a tag "A" as read/write
<b>CurrentPage</b>	attached to the same tag "A" as per <b>PageRequest</b>
<b>SyncOptions</b>	remote

Change value of tag "A" to display the requested page on HMI and HMI Client. Change page on HMI Client to display the same page on HMI.

**Example: synchronize displayed page between HMI device and on HMI Client**

Set properties as follows:

<b>PageRequest</b>	attached to a tag "A" as read/write
<b>CurrentPage</b>	attached to the same tag "A" as per <b>PageRequest</b>
<b>SyncOptions</b>	local+remote

Changing page on HMI device, same page will be shown on HMI Client and vice-versa.

## Hold Time/Autorepeat Time

Defines the values for hold time and autorepeat time for buttons and external keyboards.



Note: These properties can be redefined for each button or key in their widget property table.

## HMI device Zoom Factor

It is the zoom factor of the HMI device that will be applied when project is loaded at run time.

<b>Range</b>	0.3–2.9
<b>Default value</b>	1 = no zoom

# Events

## OnWheel

Used only in conjunction with wheel input devices. Normally the wheel is used to increase/decrease the value of a tag without an external keyboard device.

Attach this property to a change of wheel event and use an action like **StepTag** to increase/decrease a tag value.

# 6    The HMI simulator

---

HMI simulator allows you testing projects before downloading it to the HMI device. It may be used to test the project when no HMI device is available and to speed up development and debugging activities.

The HMI simulator supports:

- online simulation - in communication with real devices (only for protocols with Ethernet or RS-232 communication),
- offline simulation - simulating tag behavior

The data simulation method is set in the **Simulator** column of the Tag Editor.

---

<b>Data simulation methods .....</b>	<b>50</b>
<b>Simulator settings .....</b>	<b>50</b>
<b>Launching and stopping the simulator .....</b>	<b>51</b>

# Data simulation methods

Set tag simulation behavior in the **Simulator** field of Tag Editor.

Method	Description
<b>Variables</b>	Data is stored in a simulator variable. This variable holds the value of the tag so you can read and write the value.
<b>SawTooth</b>	A count value is incremented from <b>Offset</b> to <b>Amplitude + Offset</b> value with a <b>Period</b> of 60..3600 seconds. When the counter reaches <b>Amplitude + Offset</b> , the value is reset to <b>Offset</b> and the counter restarts.
<b>Sine Wave</b>	A sine wave value is generated and written to the tag value. <b>Min</b> , <b>Max</b> and <b>Period</b> values can be defined for each tag.
<b>Triangle Wave</b>	A triangle wave value is generated and written to the tag value. <b>Min</b> , <b>Max</b> and <b>Period</b> values can be defined for each tag.
<b>Square Wave</b>	A square wave value is generated and written to the tag value. <b>Min</b> , <b>Max</b> and <b>Period</b> values can be defined for each tag.

See ["Adding tags" on page 19](#) for details.

## Simulator settings

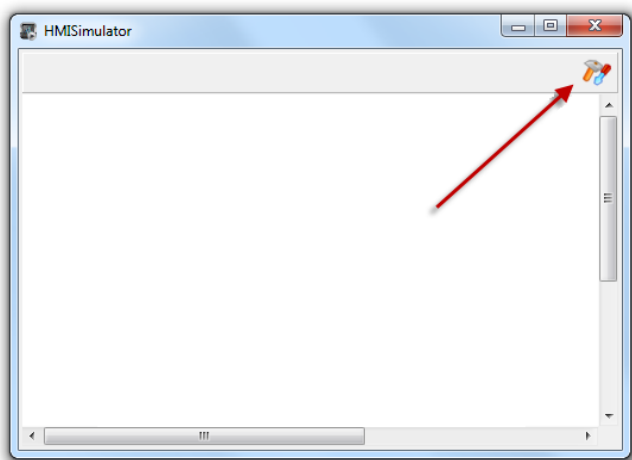
The Simulator works by default with simulated protocols. It can also work with real protocols (Ethernet or serial protocols)



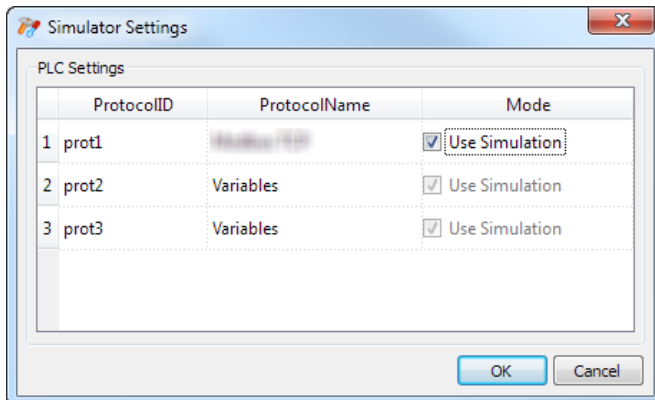
Note: For protocols not supporting communication with external devices, such as the Variables protocol, this option is always disabled.

## Changing simulated protocols

1. Click the simulator **Settings** icon.



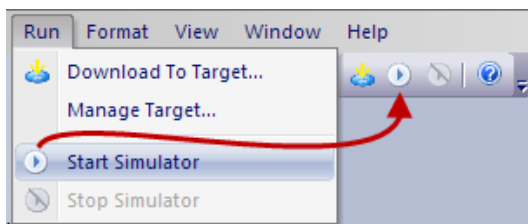
2. Select **Use Simulation** to use simulated protocols, otherwise real protocols will be used for communication with external devices.



## Launching and stopping the simulator

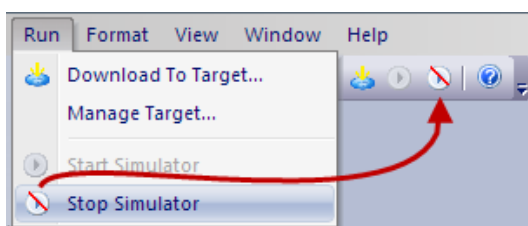
To launch the simulator:

1. On the **Run** menu, click **Start Simulator**: the Simulator runs on the computer in the same way as the server would run on the HMI device.



To stop the simulator:

1. On the **Run** menu, click **Stop Simulator** or on the simulated page double-click the **Exit** button.







# 7 Transferring the project to HMI device

---

To transfer the PB610-B Panel Builder 600 project to the target HMI device you can use:

- function **Run > Download to Target**
- function **Run > Update Package** with the use of a USB device


---

<b>Download to HMI device</b> .....	<b>54</b>
<b>Update package</b> .....	<b>57</b>
<b>The Runtime loader</b> .....	<b>59</b>
<b>Upload projects</b> .....	<b>60</b>

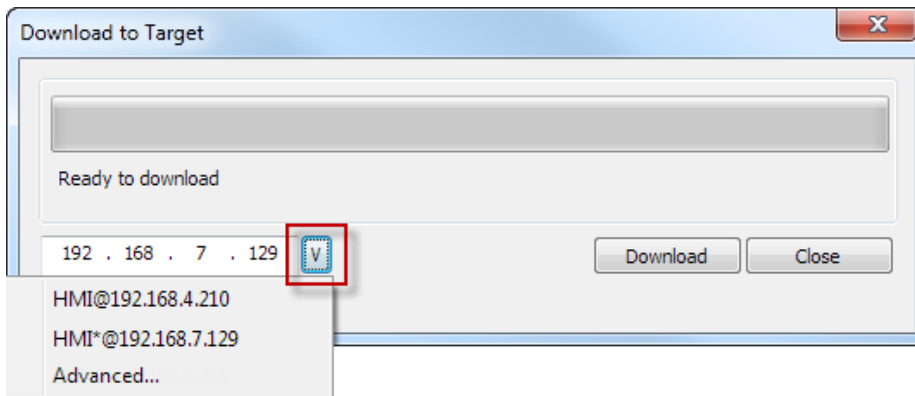
# Download to HMI device

**Path:** *Run* > *Download to Target*

This function transfers project and HMI Runtime via Ethernet .

 Note: The HMI device must have a valid IP address. See "[HMI device basic settings](#)" on page 6 for details on how to assign an IP address.

1. Click the discovery button: a list of the detected IP addresses is displayed.
2. Select the HMI device IP address.





3. Click **Download**: PB610-B Panel Builder 600 will switch the HMI device to Configuration Mode and transfer the files.

When the download operation is completed, the HMI device is automatically switched back to Operation Mode and the project is started.

## Advanced options



Option	Description
<b>Download only changes</b>	Transfers to the HMI device only the modified project files.
<b>Binary Format</b>	Download files using binary format.
<b>Delete Dynamic Files</b>	<p>Modified configuration of recipes, users, schedulers, etc. done at run time will be deleted and overwritten by the configuration defined in the project.</p> <p> <b>CAUTION: This operation cannot be undone, deleted dynamic files cannot be restored.</b></p> <p> <b>CAUTION: Dynamic files are not deleted if stored on external devices (USB or SD Cards).</b></p>

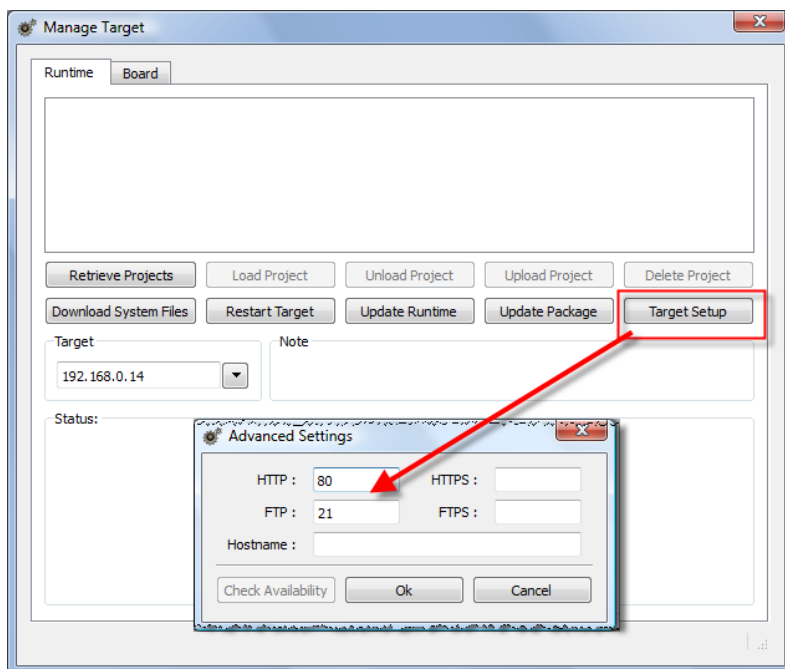
When transferring a project, PB610-B Panel Builder 600 uses a combination of HTTP and FTP connections:

- HTTP connection - issues the commands to switch to transfer mode or to unload running project,
- FTP session - transfers the files to the flash memory in the HMI device.

## Changing HMI device connection settings

Path: **Run> Manage Target**

1. Click **Target Setup**: the **Advanced Settings** dialog is displayed. Default port for HTTP connections on the HMI device is port 80.

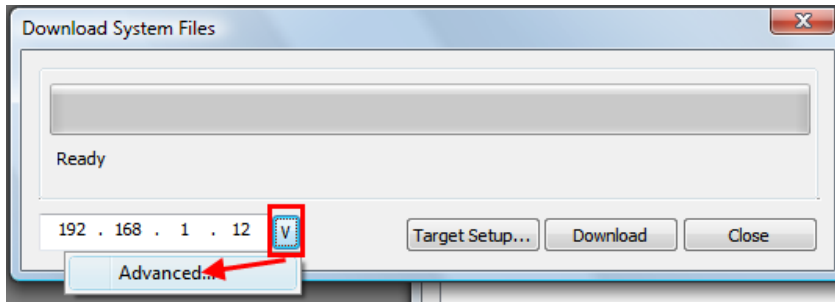


2. Set correct HTTP, FTP or HTTPS, FTPS ports for the HMI device.
3. Specify **Hostname** to easily identify each device in a network where multiple devices are available. The default host-name is "HMI" for all devices.

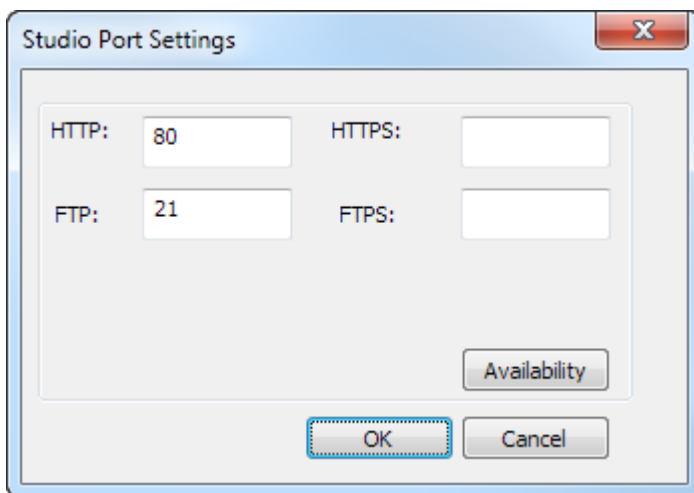
4. Click **Download System Files**. At the next download the new ports will be used in the HMI device and new host-name will appear in the drop-down list

## Changing system connection settings

1. In the **Download System Files** dialog, click **Advanced**.



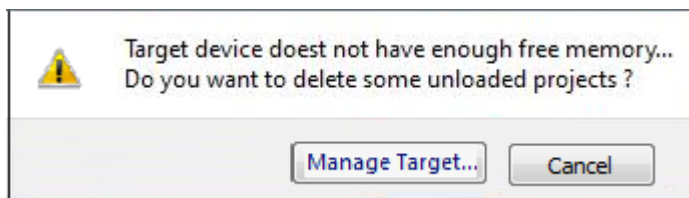
2. Set correct HTTP, FTP or HTTPS, FTPS ports for the HMI device.



These are the ports used by the system to connect to the HMI device and may need to be modified when default ports are used by other services or applications or if the local network requires specific settings.

## Managing big projects

For successful download the project size should be at least 2 MB smaller than the available memory. If not, you run out of flash memory in the HMI device and a warning message is displayed.



To free more memory:

1. Click **Manage Target**.
2. Delete the projects you no longer need to make more memory available.

## Update package

To install or update HMI Runtime and project you may create a package to be loaded via USB.



**Important:** Always include both project and the Runtime in the update packages.

If you need to use an old project with the latest Runtime version, convert the project first. See ["Installing the application" on page 2](#) for details.

### Creating an update package

*Path: Run> Update Package*

Option	Description
Target	HMI device type. Selected automatically if the project is open.
Project	Adds open project to update package.

Option	Description
<b>HMI Runtime &amp; Plug-In</b>	HMI Runtime is added to the update package. If the project is open the required plugins are also added to update package.
<b>Binary Format</b>	Download files using binary format.
<b>Set Target Password</b>	Sets password to perform critical tasks (for example, project download/upload , board management)  See " <a href="#">Protecting access to HMI devices</a> " on page 295.
<b>User Files</b>	Selects files to be copied to the QTHM folder of HMI device. Max size 5 MB
<b>Encrypted</b>	Enables encryption of update package so that it can only be unzipped by the HMI Runtime.
<b>Location</b>	Location of update package.

### Example of user's file location

Computer:

*C:\Users\Username\Desktop\myFolder*

*subFolder1/file1*

*file2*

HMI device:

*\Flash\QtHmi (on HMI device)*

*subFolder1/file1*

*file2*

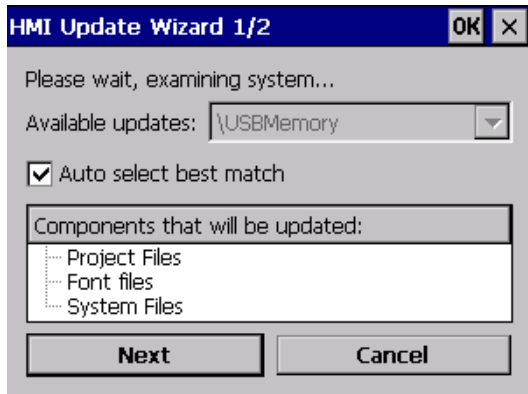


Note: User files copy is available only from the USB key.

## Loading an update package

*Path: from the context menu > **Update***

1. Assuming you have stored the package in the root folder of a USB drive, remove the drive from the computer, plug it in the HMI device, display the context menu by holding your finger for a few seconds on the screen and select **Update**.
2. The system will check for the presence of the update package in the USB drive root and ask confirmation to proceed with the update.



3. Select **Auto select best match** and click **Next**: the procedure is completed automatically

## The Runtime loader

HMI devices are delivered from factory without Runtime.

When you power up the device for the first time, the Runtime Loader window is displayed.



The Runtime Loader presence depends on the device Operating System and may not be available on all the units.

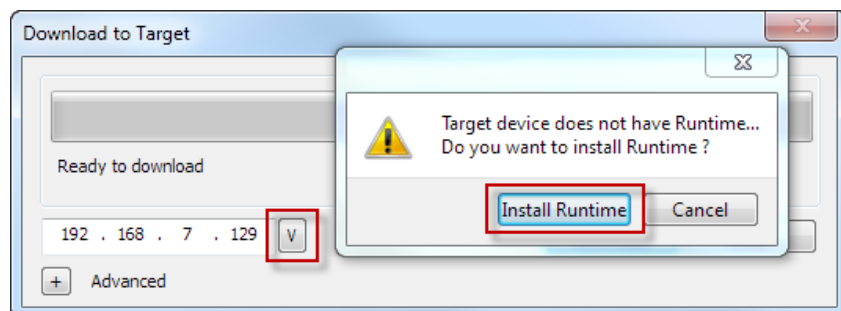


*Important: Old versions of HMI devices may not include the Runtime Loader. Contact technical support if you need further information.*

## Installing Runtime with a project

1. Click **System settings**: the **System** menu is activated in user mode.
2. Enter the IP address for the HMI device. See "[System Settings tool](#)" on page 285 for details.
3. Download a project with PB610-B Panel Builder 600 to install the Runtime.

When you download a project the Runtime is automatically installed if needed.



See ["Transferring the project to HMI device" on page 53](#) for details.

4. Click **Install Runtime**: the procedure is run automatically.

## Installing Runtime from a USB drive

1. Prepare the Update Package as described in ["Transferring the project to HMI device" on page 53](#)
2. Plug the USB drive in the device and click **Transfer from disk**.
3. Follow the instructions displayed.



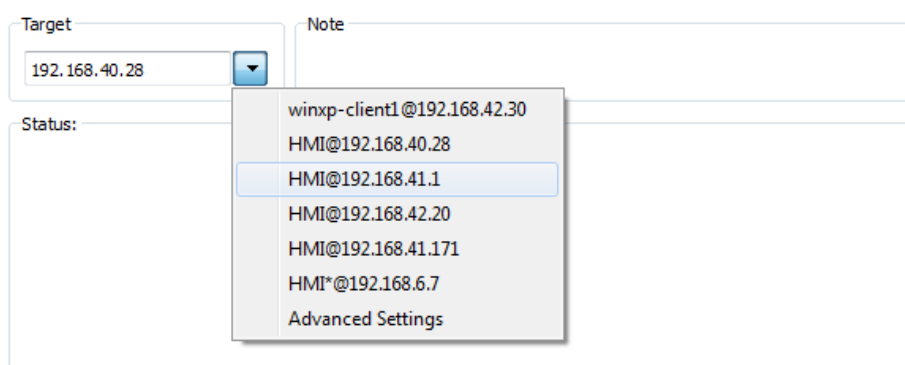
*Note: Old versions of HMI devices may not support automatic installation of Runtime. Contact technical support for more information.*

## Upload projects

**Path:** *Run* > *Manage Target*

You can copy a project from the Runtime to the computer where PB610-B Panel Builder 600 is running.

1. In the **Runtime** tab, select the IP address of the device from the drop-down list **Target**.



2. Click **Retrieve Projects**: a list of all the projects available is displayed.
3. Select project to upload
4. Click **Upload Project**: a password is required to proceed.



*Note:* From PB610-B Panel Builder 600 v1.90 (build 608) upload is password protected. See ["Protecting access to HMI devices" on page 295](#) for details.

5. Enter password: the upload process starts.



A copy of the project is saved in:

*C:\Users\username\Documents\PB610-B Panel Builder 600\workspace\Uploaded\Runtime\IPAddress\workspace\ProjectName*



Note: If the upload operation fails, check firewall settings the computer where PB610-B Panel Builder 600 is running.



# 8 System Variables

Path: **Source**> **Attach to**

System variables are special tags containing information about the HMI runtime.



Note: System Variables are available also as a standard protocol in the Protocol Editor. Use System Variables as a protocol when you have to transfer data between system variables and tags from devices, or to select custom refresh rate for a system variable.

<b>Alarms variables</b>	<b>65</b>
<b>Buzzer variables</b>	<b>65</b>
<b>Communication variables</b>	<b>66</b>
<b>Daylight Saving Time variables</b>	<b>66</b>
<b>Device variables</b>	<b>67</b>
<b>Dump information variables</b>	<b>68</b>
<b>Keypad variables</b>	<b>69</b>
<b>Network variables</b>	<b>69</b>
<b>Printing variables</b>	<b>70</b>
<b>Remote Client variables</b>	<b>70</b>

---

<b>Version variables .....</b>	<b>71</b>
<b>Screen variables .....</b>	<b>71</b>
<b>SD card variables .....</b>	<b>71</b>
<b>Server variables .....</b>	<b>72</b>
<b>Time variables .....</b>	<b>72</b>
<b>Touch screen variables .....</b>	<b>73</b>
<b>USB drive variables .....</b>	<b>74</b>
<b>User management variables .....</b>	<b>74</b>

## Alarms variables

Number of alarms of the requested type.

Variable	Description	Data type
<b>Not Triggered Acknowledged</b>	Alarm condition no longer active; alarms already acknowledged	int read only
<b>Not Triggered Not Acknowledged</b>	Alarma condition no longer active; awaiting acknowledgment	int read only
<b>Number of missed alarm events</b>	Alarms exceeding the event queue. Queue length is defined in the <i>engineconfig.xml</i> file.	int read only
<b>Triggered Acknowledged</b>	Alarm condition active; alarms already acknowledged	int read only
<b>Triggered Alarms</b>	Alarm active: acknowledgement not required	int read only
<b>Triggered Not Acknowledged</b>	Alarm condition active; awaiting acknowledgment	int read only

## Buzzer variables

Adjust buzzer behavior.

Variable	Description	Data type
<b>Buzzer Setup</b>	<b>0</b> = disabled <b>1</b> = enabled (buzzer sounds as audible on any touchscreen event) <b>2</b> = buzzer status controlled by <b>Buzzer Control</b> system variable.	int
<b>Buzzer Control</b>	<b>0</b> = buzzer off <b>1</b> = buzzer on <b>2</b> = buzzer blink	int

Variable	Description	Data type
<b>Buzzer Off Time</b>	Duration in milliseconds of off time when blink has been selected. Default = 1000. Range: 100–5000.	int
<b>Buzzer On Time</b>	Duration in milliseconds of on time when blink has been selected. Default = 1000. Range: 100–5000.	int



**WARNING:** Buzzer Setup =1 (on touch) can be overridden by the “Buzzer on Touch” property defined inside the ["Project properties pane" on page 40](#)

## Communication variables

Communication status between HMI device and controllers.

Variable	Description	Data type
<b>Protocol Communication Status</b>	Summarize the status of the communication protocols.  <b>0</b> = No protocol running, protocol drivers might not have been properly downloaded to the HMI device. <b>1</b> = Protocols loaded and started, no communication error. <b>2</b> = At least one communication protocol is reporting an error.	int  Read only
<b>Protocol Error Message</b>	Communication error with error source.  For example: "[xxxx]" where "xxxx" is the protocol abbreviation, the error source.  Multiple acronyms appear in case of multiple error sources. Blank when no errors are reported.	ASCII string  Read only
<b>Protocol Error Count</b>	Number of communication errors occurred since last reset. Reset value with Reset Protocol Error Count action, see <a href="#">"System actions" on page 88</a> .	int  Read only

## Daylight Saving Time variables

Information on the system clock. The variables contain information on the "local" time. Standard Time (solar time) and Day Light Saving time (DST) are available.



Note: All variables are read only; you cannot use them to update the system clock.

Variable	Description
<b>Standard offset</b>	Offset in minutes when standard time is set, with respect to GMT (for example: $-8 \times 60 = -480$ minutes).
<b>Standard week</b>	Week in which the standard time starts (for example: First = 1).
<b>Standard Month</b>	Month in which the standard time starts. Range: 0–11. (for example: November = 10).
<b>Standard Day</b>	Day of week in which the standard time starts (for example: Sunday = 0).
<b>Standard hour</b>	Hour in which the standard time starts (for example: 02 = 2).
<b>Standard minute</b>	Minute in which the standard time starts (for example: 00 = 0).
<b>DST offset</b>	Offset in minutes when DLS time is set, with respect to GMT
<b>DST week</b>	Week in which the DLS time starts
<b>DST Month</b>	Month in which the DLS time starts. Range: 0–11.
<b>DST Day</b>	Day of week in which the DLS time starts
<b>DST hour</b>	Hour in which the DLS time starts
<b>DST minute</b>	Minute in which the DLS time starts

## Device variables

Device settings and operating status information.

Variable	Description	Data type
<b>Available System Memory</b>	Free available RAM memory in bytes.	uint64 read only
<b>Backlight Time</b>	Activation time in hours of the display backlight since production of the device.	unsignedInt read only
<b>Battery LED</b>	Enables/disables the low battery LED indicator (when available).  0 = disabled 1 = enabled	int
<b>Battery Timeout</b>	Reserved	int
<b>Display Brightness</b>	Returns and adjusts brightness level.  When set to a low light level (0..3), the backlight stays lit to a higher level for 8 seconds to allow the user to make the adjustments and then is switched-off.  Even when set to 0, the backlight is still on and the <b>Backlight Time</b> counter increases.	int

Variable	Description	Data type
	Range: 0–255	
<b>External Timeout</b>	Non-operational time after which the display backlight is automatically turned off. The backlight is automatically turned on when the user touches the screen.  -1 = switch off backlight and disable touch (switch display off). <b>Backlight Time</b> counter is stopped.  0 = switch backlight on (switch display on)  1..n = timeout for switch off backlight (screensaver timer)	int
<b>Flash Free Space</b>	Free space left in internal Flash memory.	uint64 read only
<b>System Font List</b>	List of system fonts	string read only
<b>System Mode</b>	Runtime operation status.  1 = booting  2 = configuration mode  3 = operating mode  4 = restart  5 = shutdown	int
<b>System UpTime</b>	Time the system has been powered since production of the unit (hours).	unsignedInt read only

## Dump information variables

Status of the copy process to external drives (USB or SD Card) for trend and event buffers.



Note: If copy time is less than one second, the system variable does not change its value.

Variable	Description	Data type
<b>Dump Trend Status</b>	1 = trend buffer copy in progress	int read only
<b>Dump Archive Status</b>	1 = event buffer copy in progress	int read only



Variable	Description	Data type
<b>Dump Recipe Status</b>	<b>1</b> = recipe buffer copy in progress If the copy duration time is less than 1 second, the system variable does not change its value	int read only
<b>Reset Recipe Status</b>	<b>1</b> = recipe buffer reset in progress If the reset duration time is less than 1 second, the system variable does not change its value	int read only
<b>Dump Recipe Status</b>	Returns information during the copy process of recipes. If the copy duration time is less than 1 second, the system variable does not change its value. <b>0</b> = initial default state <b>1</b> = operation triggered <b>2</b> = operation complete successfully <b>3</b> = operation completed with errors	int read only

## Keypad variables

Keypad status.

Variable	Description	Data type
<b>Is keypad open</b>	<b>0</b> = no keypad open <b>1</b> = keypad open	int read only

## Network variables

Device network parameters.

Variable	Description	Data type
<b>Gateway</b>	Gateway address of the main Ethernet interface of device	string read only
<b>IP Address</b>	IP address of the main Ethernet interface of device	string read only
<b>Mac ID</b>	MAC ID of the main Ethernet interface of device	string read only
<b>Subnet Mask</b>	Subnet Mask of the main Ethernet interface of device	string read only

## Printing variables

Information on printing functions.

Variable	Description	Data type
<b>Completion percentage</b>	Percentage of completion of current print job. Range: 0–100	read only
<b>Current disk usage</b>	Folder size in bytes where PDF reports are stored. If <i>Flash</i> has been selected as <i>Spool media type</i> , this value corresponds to <i>reportspool</i> .	read only
<b>Current job</b>	Name of the report the job is processing. Current job is the following: <ul style="list-style-type: none"> <li>• [report name] for a <b>Graphic Report</b></li> <li>• [first line of text] for a <b>Text Report</b></li> </ul>	read only
<b>Current RAM usage</b>	Size in bytes of the RAM used to process the current job	read only
<b>Disk quota</b>	Maximum size in bytes of the folder where PDF reports are stored	read only
<b>Graphic job queue size</b>	Number of available graphic jobs in the printing queue	read only
<b>RAM quota</b>	Maximum size in bytes of the RAM used to generate reports	read only
<b>Status</b>	Printing system status. Values: <ul style="list-style-type: none"> <li>• <b>idle</b></li> <li>• <b>error</b></li> <li>• <b>paused</b></li> <li>• <b>printing</b></li> </ul>	string read only
<b>Text job queue size</b>	Number of available text jobs in the printing queue	read only

## Remote Client variables

The following system variables are associated to the transferring files to a remote HMI device.

Variable	Description	Data type
<b>DownloadFromHMIError</b>	Error description	ASCII string read only
<b>DownloadFromHMIPercentage</b>	Download progress (0→100)	read only
<b>DownloadFromHMIStatus</b>	<b>0</b> = idle, action is not in use or completed	int (32 bit)

Variable	Description	Data type
	<b>1</b> = file download in progress <b>2</b> = error	read only
<b>uploadToHMIError</b>	Error description	ASCII string read only
<b>uploadToHMIPercentage</b>	Upload progress (0→100)	read only
<b>uploadToHMIStatus</b>	<b>0</b> = idle, action is not in use or completed <b>1</b> = file upload in progress <b>2</b> = error	int (32 bit) read only

## Version variables

Operating System and runtime version.

Variable	Description	Data type
<b>Main OS Version</b>	Version of Main OS, for example, UN30HSxx60M0166.	string
<b>Runtime Version</b>	Version of runtime, for example, 1.90 (0) – Build (682)	string

## Screen variables

Screen status.

Variable	Description
<b>Time remaining to unlock</b>	Time remaining to unlock screen (see <b>LockScreen</b> action, <a href="#">"Page actions" on page 79</a> )
<b>X Screen resolution</b>	Display horizontal screen size in pixel
<b>Y Screen resolution</b>	Display vertical screen size in pixel

## SD card variables

Information on the external SD card.

Variable	Description	Data type
<b>SD Card FreeSpace</b>	Available space on card in bytes	uint64 read only
<b>SD Card Name</b>	Name of SD card	string

Variable	Description	Data type
		read only
<b>SD Card Size</b>	Size in bytes of the card plugged in the slot	uint64 read only
<b>SD Card Status</b>	Status of SD card	int read only

## Server variables

Server status.



**Important: All variables refer to server, not to HMI Client.**

Variable	Description	Data type
<b>Current page</b>	Name of current page	string
<b>Current project</b>	Name of current project	string
<b>Operating mode time</b>	Seconds elapsed since device started operating mode	uint64
<b>Project load time</b>	Date when the project was loaded on the HMI Runtime as in <b>System Date</b> format (milliseconds).	uint64

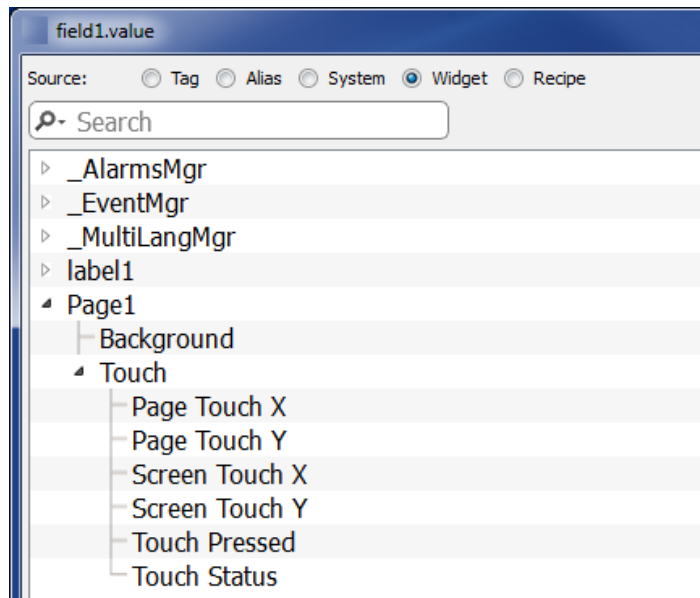
## Time variables

System time expressed in UTC format.

Variable	Description	Data type
<b>Day Of Month</b>	Range: 1–31	int
<b>Day of Week</b>	Range: 0 = Sunday, .. , 6 = Saturday	int
<b>Hour</b>	Range: 0–23	int
<b>Minute</b>	Range: 0–59	int
<b>Month</b>	Range: 1–12	int
<b>Second</b>	Range: 0–59	int
<b>System Time</b>	The same as UTC time. It can also be set as date/time for this variable.	unsignedInt
<b>Year</b>	Current Year	int

## Touch screen variables

Cursor status and position on the touchscreen. These are properties of the active page and can be selected in the **Widget** section.



Note: Page size can be different than HMI device display size.

Variable	Description	Java Script
<b>Page Touch X</b>  <b>Page Touch Y</b>	Cursor position related to page	page.primaryTouch.x page.primaryTouch.y
<b>Screen Touch X</b>  <b>Screen Touch Y</b>	Cursor position related touchscreen	page.primaryTouch.screenX page.primaryTouch.screenY
<b>Touch Press</b>	<b>0</b> = screen not pressed <b>1</b> = screen pressed	page.primaryTouch.pressed
<b>Touch Status</b>	<p>Generic touch screen changes. This variable contains the concatenation of <b>Screen Touch X</b>, <b>Screen Touch Y</b> and <b>Touch Press</b> values (for example, "924,129,0").</p> <p>The main usage of this variable is to trigger an event, using the OnDataUpdate feature, when something (x, y or click) is changed.</p>	page.primaryTouchStatus

## USB drive variables

Information on the external USB drive connected to the device.

Variable	Description	Data type
<b>USB Drive free space</b>	Available space in bytes	uint64 read only
<b>USB Drive Name</b>	Name of USB device	string read only
<b>USB Drive Size</b>	Size in bytes of the device plugged in the USB port	uint64 read only
<b>USB Drive Status</b>	Status of USB device	int read only

## User management variables

Information on users and groups.

Variable	Description	Data type
<b>No of Remote-Clients Alive</b>	Number of HMI Clients connected to the server	short read only
<b>This Client Group-Name</b>	Group of currently logged user	string read only
<b>This Client ID</b>	Only for HMI Clients. Local and remote clients connected to the same server (for example, runtime) get a unique ID.	short read only
<b>This Client User-Name</b>	Name of the user logged to the client where the system variable is displayed.	string read only

# 9 Actions

---

Actions are functions used to interact with the system and are normally executed when events are triggered.

Events can be triggered by various widgets, for example on press and on release of a button. Not all actions are available for all the events of an object.

Actions are linked to widgets in the **Event** section of the Property pane (Page Editor).

---

<b>Alarm actions</b> .....	<b>76</b>
<b>Event actions</b> .....	<b>76</b>
<b>MultiLanguage actions</b> .....	<b>77</b>
<b>Keyboard actions</b> .....	<b>77</b>
<b>Page actions</b> .....	<b>79</b>
<b>Print actions</b> .....	<b>82</b>
<b>Recipe actions</b> .....	<b>84</b>
<b>Remote Client actions</b> .....	<b>87</b>
<b>System actions</b> .....	<b>88</b>
<b>Tag actions</b> .....	<b>93</b>
<b>Trend actions</b> .....	<b>95</b>
<b>User management actions</b> .....	<b>98</b>
<b>Widget actions</b> .....	<b>100</b>

## Alarm actions

Used to acknowledge or reset alarms.

### SelectAllAlarms

Selects all alarms in the alarm widget.

### AckAlarm

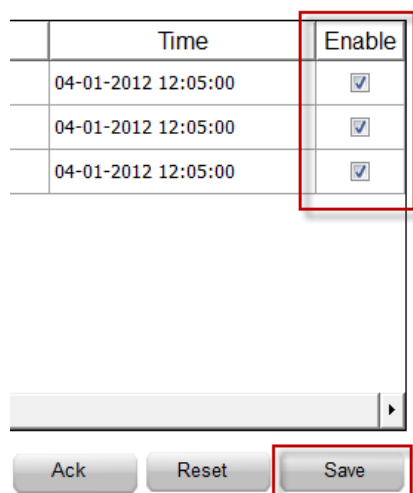
Acknowledges selected alarms.

### ResetAlarm

Resets selected acknowledged alarms.

### EnableAlarms

Saves changes made in the **Enable** column in the alarm widget. This action is used with the **Save** button in the alarm widget.



## Event actions

Used by Alarm History widget to scroll events/alarms backward/forward in table view (event buffer widget).

### ScrollEventsBackward

Scrolls events/alarms backward in table view (event buffer widget).

### ScrollEventsForward

Scrolls events/alarms forward in table view (event buffer widget).



## MultiLanguage actions

Selects the application language.

### SetLanguage

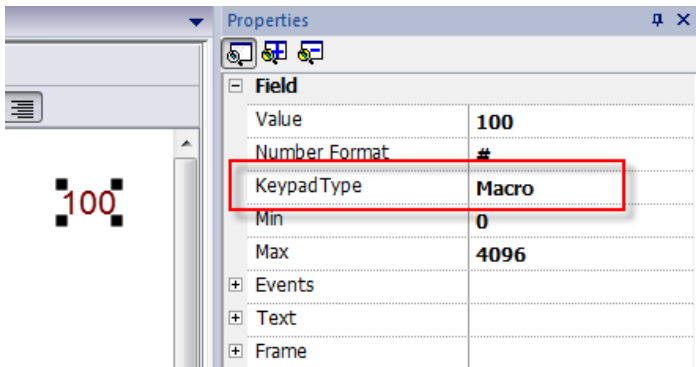
Sets the language used. The selected language will be applied at run time to all applicable widgets.

## Keyboard actions

Changes the use of keypads.

### SendKey

Sends one character to a numeric widget. The **KeypadType** property of the numeric widget must be set as **Macro**.

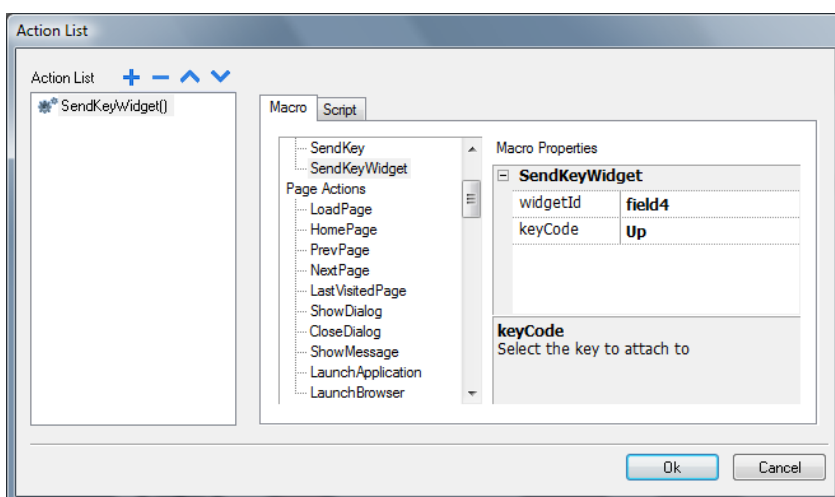
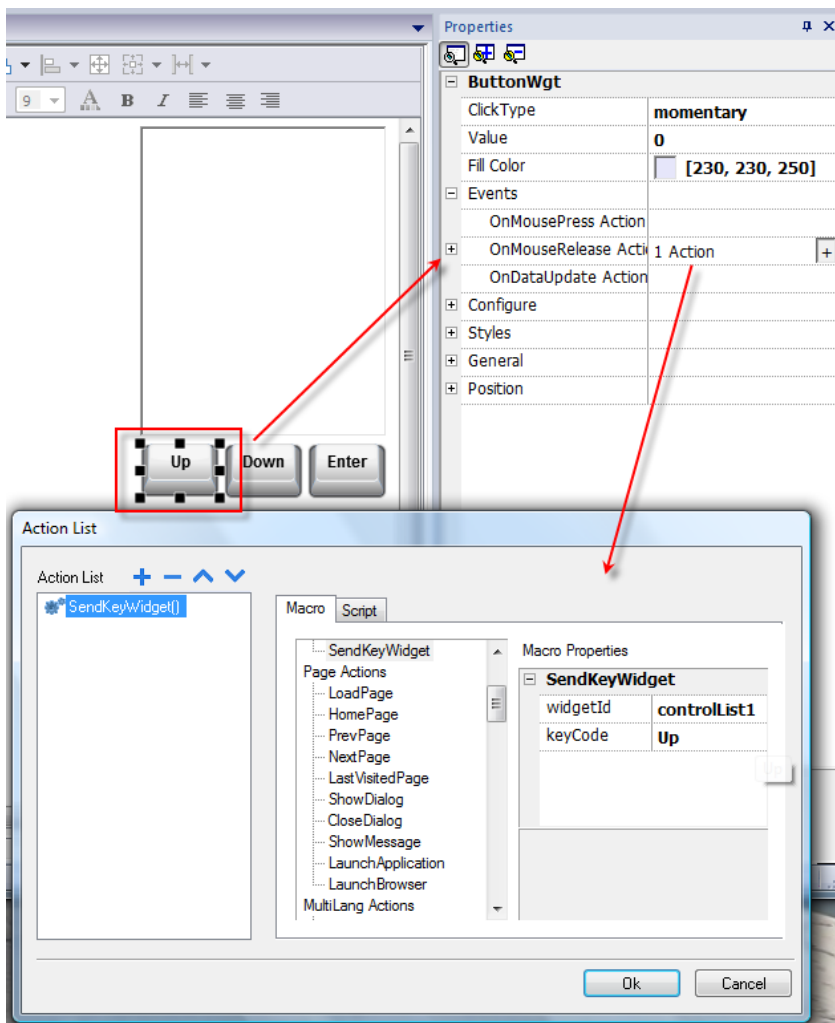


### SendKeyWidget

Sends one character to a specific widget.

#### Example

The **Up** and **Down** buttons use the **SendKeyWidget** action in association with the **Control List Widget**.



## ShowKeyPad

Shows the default operating system touch keypad.

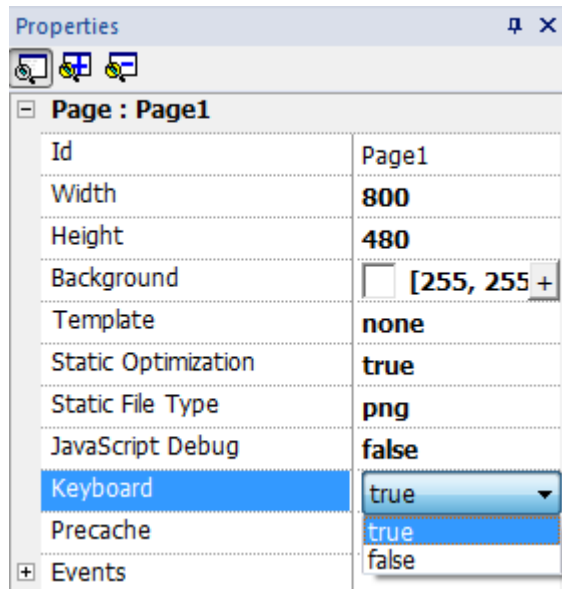


Note: might not be supported by all operating systems.

## Keyboard

Enables/disables the use of actions when using external keyboards. Action execution can be enabled/disabled both at project and at page level.

The effect is equivalent to the use of the property Keyboard for project and page.



## Page actions

Page navigation. Page actions can be used with the following events:

- OnMouseClicked,
- OnMouseRelease,
- OnMouseHold
- OnActivate
- OnDeactivate
- Alarms
- Schedulers.

### LoadPage

Go to the selected page of the project.

### HomePage

Go to the home page.

You can set the home page in the **Behavior** section of the **Project Widget**, see ["Behavior" on page 45](#)

### PrevPage

Go to the previous page.

## NextPage

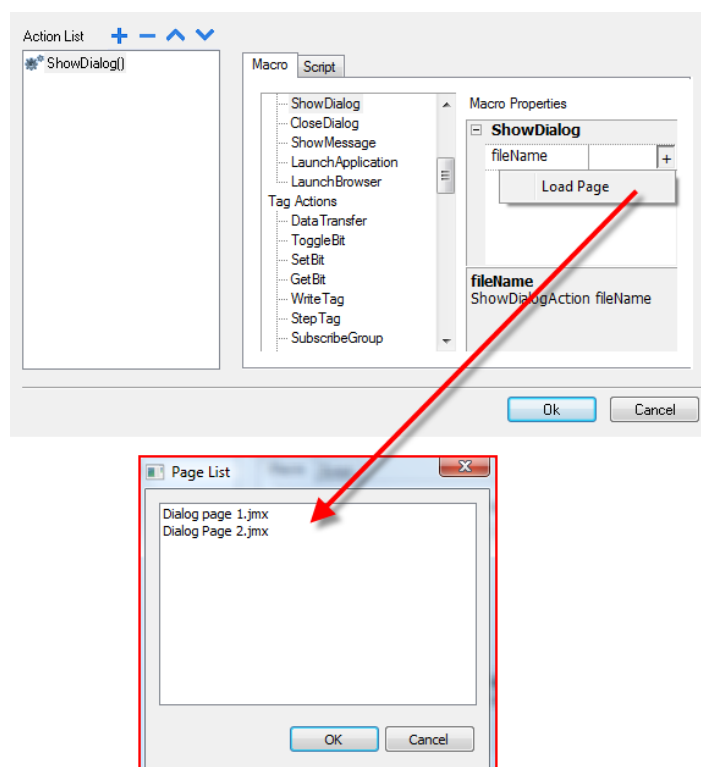
Go to the next page.

## LastVisitedPage

Go to the previously displayed page

## ShowDialog

Opens a dialog page defined in the project.

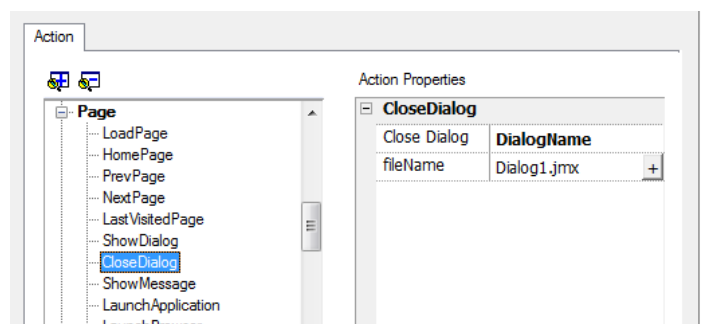


## CloseDialog

Close dialog pages.



Note: This action is applicable only to dialog pages.



## CloseDialog options

Option	Description
<b>All</b>	Closes all open dialogs
<b>Selected</b>	Closes only active dialog
<b>DialogName</b>	Closes dialog specified as <b>fileName</b> property

## JavaScript Interface

`project.closeDialog(DialogID);`

Where *DialogID*:

<b>All</b>	Closes all open dialogs
<b>Selected</b>	Closes only active dialog
<b>DialogName.jmx</b>	Closes dialog specified as <b>fileName</b> parameter

## Examples

Example	Behavior
<code>project.closeDialog("All");</code>	All open dialogs are closed
<code>project.closeDialog("Selected");</code>	The selected dialog is closed
<code>project.closeDialog("Dialog1.jmx");</code>	All instances of Dialog1 are closed

The function `project.closeDialog()`; without parameter works as `project.closeDialog("Selected");`.

## ShowMessage

Displays a popup message. Enter the text of the message to be displayed.

## LaunchApplication

Launches an external application.

Parameter	Description
<b>App Name</b>	Executable name with extension (for example, "notepad.exe" to run Notepad)
<b>Path</b>	Application path. In Windows CE platforms: <code>\flash\qthmi</code> .
<b>Arguments</b>	Application specific arguments (for example, <code>\flash\qthmi\Manual.pdf</code> to open the document "Manual.pdf")
<b>Single Instance</b>	Argument to start the application in a single instance or multiple instances.  When single instance is selected, the system first verifies whether the application is already running; if so, then the application is brought to the foreground, if not, then the application is launched.



Note: Arguments with spaces must be quoted (for example, "\Storage Card\Manual.pdf")

## LaunchBrowser

Opens the default web browser. You can define URL address as argument.



Note: Only works on platforms having a native web browser (for example, on Windows CE PRO with Internet Explorer enabled).

## LaunchUpdater

Updates project and/or runtime from an external device.

Use **Path** parameter to specify folder.

### Examples

- \USBMemory (for USB devices in Windows CE)
- \Storage Card (for SD devices in Windows CE)



Note: Not supported in devices based on Win32.

## JavaScript Interface

*project.launchUpdater(strPath)*

### Examples

```
project.launchUpdater("\\USBMemory")
```

## LockScreen

Temporarily locks the touch screen. Allows cleaning the touch screen.

The system variable **Time remaining to unlock** displays the time remaining to unlock. See ["Screen variables" on page 71](#)

# Print actions

Manages print tasks.

## PrintGraphicReport

Prints a graphic report.

Parameter	Description
<b>reportName</b>	Assigns a name to the report
<b>silent</b>	<b>false</b> = allows to set printer properties at run time

## PrintText

Prints a string.

Parameter	Description
<b>text</b>	String to be printed
<b>silent</b>	<b>false</b> = allows to set printer properties at run time

This action works in line printing mode and uses a standard protocol common to all printers that support it. Text is printed immediately line by line or after a timeout custom for each printer model.



Note: printing could a few minutes for models not designed for line printing.

No custom driver is required.

## PrintBytes

Prints an hexadecimal string representing data to print (for example, "1b30" to print < ESC 0 >).

Parameter	Description
<b>bytes</b>	Exadecimal string to print
<b>silent</b>	<b>false</b> = allows to set printer properties at run time

This action works in line printing mode and uses a standard protocol common to all printers that support it. Text is printed immediately line by line or after a timeout custom for each printer model.



Note: printing could a few minutes for models not designed for line printing.

No custom driver is required.

## EmptyPrintQueue

Flushes the current printing queue. If executed while executing a job, the queue is cleared at the end of the job.

## PausePrinting

Puts the current printing queue on hold. If executed while executing a job, the queue is paused at the end of the job.

## ResumePrinting

Restarts a queue previously put on hold.

## AbortPrinting

Stop the execution of the current job and removes it from the queue. If the queue has another job, then, after aborting, the next job starts.

# Recipe actions

Used to program recipe management.

## DownloadRecipe

Copy recipe data from HMI device flash memory to the controller (e.g. PLC, local variable, depending on the protocol).

Parameter	Description
<b>RecipeName</b>	Name of recipe to download
<b>RecipeSet</b>	Number of recipe set to copy. <b>curSet</b> = download currently selected recipe set

## UploadRecipe

Saves recipe data from the controller (e.g. PLC, local variable, depending on the protocol) to the device Flash Memory.

Parameter	Description
<b>RecipeName</b>	Name of recipe to upload
<b>RecipeSet</b>	Number of recipe set to copy. <b>curSet</b> = upload currently selected recipe set

## WriteCurrentRecipeSet

Sets the selected recipe as current recipe set.

Parameter	Description
<b>RecipeName</b>	Name of recipe to set as current recipe
<b>RecipeSet</b>	Recipe set to define as current recipe set

## DownloadCurRecipe

Downloads current set of recipe data to the controller.

No parameter is required.

## UploadCurRecipe

Uploads set of controller data to current recipe set.

No parameter is required

## ResetRecipe

Restores factory settings for recipe data. Original recipe data will overwrite uploaded recipes

Select the recipe that you want to reset to factory data.



## DumpRecipeData

Dumps recipe data to internal or external storage. Data is saved in .csv format.

Define the location where to save the file.



Note: supported formats are FAT or FAT32. NTFS format is not supported.

Parameter	Description
<b>DateTimePrefixFileName</b>	<b>true</b> = the dumped file will have date and time as prefix to its name (for example D2012_01_01_T10_10_recipe1.csv)
<b>TimeSpec</b>	Time format: <ul style="list-style-type: none"> <li>• <b>Local</b> = the time values exported are the time of the HMI device.</li> <li>• <b>Global</b> = the time values exported are in UTC format.</li> </ul>
<b>FileName</b>	Tag that specifies a filename.

## RestoreRecipeData

Restores previously saved recipe data.

Enter the file full path of the Recipe files in any external storage like USB, SD or network paths.

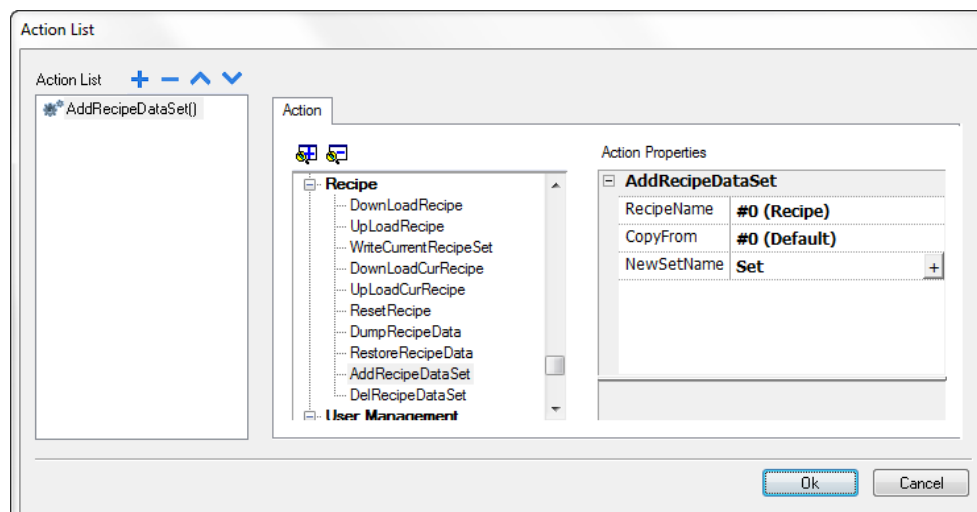


Note: supported formats are FAT or FAT32. NTFS format is not supported.

Parameter	Description
<b>FileName</b>	Attached tag from which read the file name at run time.
<b>BrowseForFile</b>	<b>true</b> = shows the Open dialog to browse the file to read. <b>false</b> = no dialog is shown,

## AddRecipeDataSet

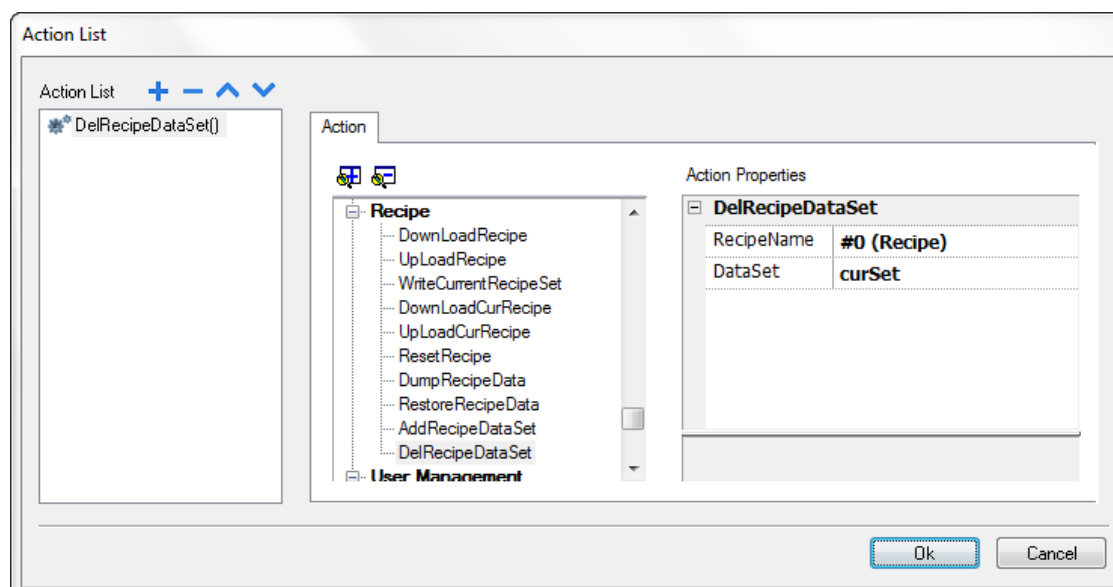
Adds a new dataset to the selected recipe. The new dataset is appended at the end of the already defined datasets.



Parameter	Description
<b>RecipeName</b>	Recipe where the dataset is added.
<b>CopyFrom</b>	Dataset from where parameters values are copied from to initialize the new dataset
<b>NewSetName</b>	Name of new dataset. Here you can use a tag reference.

## DelRecipeDataSet

Deletes a dataset from the selected recipe. Deleting a dataset will rearrange the position number of the datasets that follow.



Parameter	Description
<b>RecipeName</b>	Recipe where the dataset is to be deleted.
<b>DataSet</b>	Dataset to be deleted.

## Remote Client actions

Used to upload and download files to and from a remote HMI device. These actions can only be used from a remote HMI Client to access remote files via FTP.



**Important: Enable FTP support and give all necessary user rights to the folders used to transfer files.**

### UploadToHMI

Opens a file Open dialog to select a file to be uploaded to the remote HMI device.

Parameter	Description
<b>Destination</b>	Destination path on HMI device for file upload
<b>Filter</b>	File extensions of the files to be displayed separated by commas (for example, *.txt)

### DownloadFromHMI

Opens a file Open dialog to select a file to be downloaded from the remote HMI device.



**Note:** Only files matching the set filter are displayed and can be downloaded.

Parameter	Description
<b>Source</b>	Source path on the HMI device for file download
<b>Filter</b>	File extensions of the files to be displayed separated by commas (for example, *.txt)

### JavaScript Interface

```
boolean project.uploadToHMI(dirPath, strFilter);
```

```
boolean project.downloadFromHMI(dirPath, strFilter);
```

Parameter	Description
<b>dirPath</b>	Source path on the HMI device for file download/upload
<b>strFilter</b>	File extensions of the files to be displayed separated by commas (for example, *.txt)

Return values:

<b>True</b>	Transfer successful
<b>False</b>	Transfer failed



Note: When transferred, system variables are updated with the status of ongoing operations.

## System actions

Used to manage system properties.

### Restart

Restarts the runtime.

### DumpTrend

Stores historical trend data to external drives (USB drive or SD card).

Parameter	Description
<b>TrendName</b>	Name of historical trend to store
<b>FolderPath</b>	Destination folder: USB drive = <code>\USBMemory</code> SD Card = <code>\Storage Card</code>
<b>FileFormat</b>	<p><b>Binary</b> = the buffer is dumped in binary format (a .dat file and .inf file). Both these files are then required to convert data in .csv format by an external utility.</p> <p><b>Compatibility CSV</b> = the buffer is dumped to the specified location as a .csv file format compatible with versions 1.xx</p> <p><b>Compact CSV</b> = the buffer is dumped to the specified location as a .csv file using a newer format</p> <p>See <a href="#">"Exporting trend buffer data" on page 133</a></p>
<b>DateTimePrefixFileName</b>	<b>true</b> = the dumped file will have date and time as prefix to its name (for example D2012_01_01_T10_10_Trend1.csv)
<b>timeSpec</b>	Time format: <ul style="list-style-type: none"> <li>• <b>Local</b> = the time values exported are the time of the HMI device.</li> <li>• <b>Global</b> = the time values exported are in UTC format.</li> </ul>



Note: execution of the DumpTrend action will automatically force a flush to disk of the data temporarily maintained in the RAM memory. See ["History trends" on page 136](#) for details on how to save sampled data to disk.



Note: external drives connected to USB port must have format FAT or FAT32. NTFS format is not supported.

### To convert binary dump files to .csv

The TrendBufferReader.exe tool is stored in the *Utils* folder of the PB610-B Panel Builder 600 installation folder.

Use the following syntax:

```
TrendBufferReader -r Trend1 Trend1.csv 1
```

where:

`Trend1` = name of the trend buffer without extension resulting from the dump (original file name is trend1.dat)

`Trend1.csv` = name for the output file.

### .csv file structure

The resulting .csv file has five columns

Column	Description
<b>Data Type</b>	Data type of sampled tag: 0 = empty 1 = boolean 2 = byte 3 = short 4 = int 5 = unsignedByte 6 = unsignedShort 7 = unsignedInt 8 = float 9 = double
<b>Value</b>	Value of the sample
<b>Timestamp (UTC)</b>	Timestamp in UTC format
<b>Sampling Time(ms)</b>	Sampling interval time in milliseconds
<b>Quality</b>	Tag value quality. Information coded according the OPC DA standard and stored in a byte data (8 bits) defined in the form of three bit fields; Quality, Sub status and Limit status.  The eight quality bits are arranged as follows: QQSSSSL. For a complete and detailed description of all the single fields, please refer to the OPC DA official documentation.

### Commonly quality values

The most commonly used quality values returned by the HMI acquisition engine are:

Quality Code	Quality	Description
0	BAD	The value is bad but no specific reason is given
4	BAD	Specific server problem with the configuration. For example, the tag has been deleted from the configuration file (tags.xml).
8	BAD	No value may be available at this time, for example the value has not been provided by the data source.
12	BAD	Device failure detected
16	BAD	Timeout before device response.
24	BAD	Communication failure
28	BAD	No data found for upper or lower bound value Trend interface specific flag.
32	BAD	No data collected (for example, archiving not active. Trend interface specific flag. This value is also used to indicate a temporary offline status (for any condition where sampling was stopped).
64	UNCERTAIN	No specific reason.
65	UNCERTAIN	No specific reason. The value has 'pegged' at some lower limit.
66	UNCERTAIN	No specific reason. The value has 'pegged' at some higher limit.
67	UNCERTAIN	No specific reason. The value is a constant and cannot move.
84	UNCERTAIN	Returned value outside its defined limits defined. In this case the <b>Limits</b> field indicates which limit has been exceeded but the value can move farther out of this range.
85	UNCERTAIN	Returned value outside its defined limits defined. In this case the <b>Limits</b> field indicates which limit has been exceeded but the value can move farther out of this range. The value has 'pegged' at some lower limit.
86	UNCERTAIN	Returned value outside its defined limits defined. In this case the <b>Limits</b> field indicates which limit has been exceeded but the value can move farther out of this range. The value has 'pegged' at some higher limit

Quality Code	Quality	Description
87	UNCERTAIN	Returned value outside its defined limits defined.  In this case the <b>Limits</b> field indicates which limit has been exceeded but the value can move farther out of this range.  The value is a constant and cannot move.
192	GOOD	-


## DeleteTrend

Deletes saved trend data.

Define the name of the trend from which you want to delete logs.

## DumpEventArchive

Stores historical alarm log and audit trail data to external drives, such as USB memory or SD card.

Parameter	Description
<b>EventArchive</b>	Name of buffer to dump data
<b>FolderPath</b>	Destination folder: <ul style="list-style-type: none"> <li>• USB drive = <i>\USBMemory</i></li> <li>• SD Card = <i>\Storage Card</i></li> </ul>  Note: supported formats are FAT or FAT32. NTFS format is not supported.
<b>DumpConfigFile</b>	Enables conversion to .csv file
<b>DumpAsCSV</b>	<b>true</b> = the buffer is dumped to the specified location as a .csv file  <b>false</b> = the buffer is dumped in binary format (a .dat file and .inf file). Both these files are then required to convert data in .csv format by an external utility.
<b>DateTimePrefixFileName</b>	<b>true</b> = the dumped file will have date and time as prefix to its name (for example D2012_01_01_T10_10_alarmBuffer1.csv)  Note: option only available when exporting directly in .csv format.
<b>timeSpec</b>	Time format: <ul style="list-style-type: none"> <li>• <b>Local</b> = the time values exported are the time of the HMI device.</li> <li>• <b>Global</b> = the time values exported are in UTC format.</li> </ul>

## Example

When exporting Event buffers in binary format and **DumpConfigFile** is set to true (recommended settings), there are two folders:

- **data**, containing data files,
- **config**, containing configuration files for .csv conversion.

Once the two folders are copied from the USB drive to the computer disk, the folder structure will be:

```
\config\
    alarms.xml
    eventconfig.xml
\data\
    AlarmBuffer1.dat
    AlarmBuffer1.inf
\
AlarmBufferReader.exe
```

### To convert dump files to .csv

The AlarmBufferReader.exe tool is stored in the *Utils* folder of the PB610-B Panel Builder 600 installation folder.

Use the following syntax:

```
AlarmBufferReader AlarmBuffer1 FILE ./AlarmBuffer1.csv
```

where:

AlarmBuffer1 = name of the dumped .dat without extension

AlarmBuffer1.csv = name for the output file.

The utility AuditTrailBufferReader.exe is available for Audit Trail buffers.



Note: set DumpConfigFile to **true**.

The result of the dump is a folder structure similar to the one generated for Events.

Use the following syntax:

```
AuditTrailBufferReader AuditTrail FILE ./AuditTrail.csv
```

where:

AuditTrail = name of the dumped buffer without extension and

AuditTrail1.csv = name for the output file.

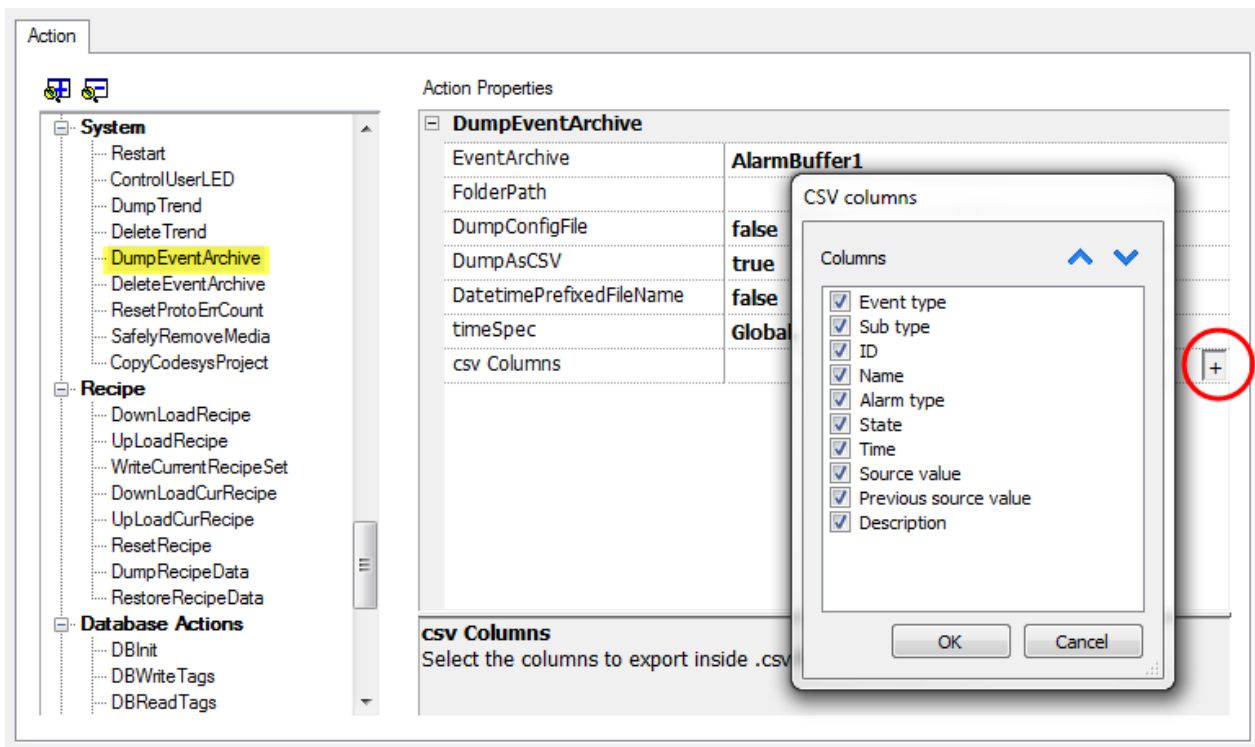
### csv Columns



Note: available only for Alarms buffers.

For Alarms buffers, additional columns can be included in the dump .csv file.





## DeleteEventArchive

Deletes saved Event buffers log data.

Specify the name of Event buffer to delete from the Event logs.

## ResetProtoErrCount

Resets the Protocol Error Count system variable.

See ["System Variables" on page 63](#) for details.

## SafelyRemoveMedia

Provides for safe removal of SD card or USB drive from HMI.

# Tag actions

Interacts with tags.

## DataTransfer

Exchanges data between:

- two controllers,
- registers within a controller,
- from system variables to controllers,
- from controllers to system variables

The various tag types include a controller tag, a system variable, a recipe tag and widget property.

## ToggleBit

Toggles a bit value of a tag.

**BitIndex** allows you to select the bit to be toggled: toggling requires a read-modify-write operation; the read value is inverted and then written back to the tag.

## SetBit

Sets the selected bit to "1".

**BitIndex** allows you to select the bit position inside the tag.

## ResetBit

Resets the selected bit to "0"

**BitIndex** allows you to select the bit position inside the tag.

## WriteTag

Writes constant values to the controller memory. Specify tag name and value.

## StepTag

Increments or decrements tag value.

Parameter	Description
TagName	Name of tag to increase/decrease
Step	Step value
Do not step over limit	Enables step limit
Step Limit	Value of step limit, if enabled.

## ActivateGroup

Forces the update of a group of tags.

Tags are updated either when used in the current page or continuously, if defined as active in the Tag Editor. This action forces all the tags of a group to be continuously updated.

## DeactivateGroup

Deactivates a group of tags, that is stops forcing the update of a group of tags.

## EnableNode

Enable/disables action for offline node management. No communication is done with a disabled node.

Parameter	Description
<b>Protocol ID</b>	Unique identifier of selected protocol
<b>NodeID</b>	Node identifier in selected protocol. Can be attached to a tag.
<b>Enable</b>	Node communication status:  <b>False</b> = disabled  <b>True</b> = enabled  When attached to a tag, tag = 0 means <b>False</b>

## Trend actions

Used for Live Data Trends and Historical Trends Widget.

### RefreshTrend

Refreshes the **Trend** window.

It can be used in any Trends/Graphs widgets. Specify the widget as a parameter for the action.

### ScrollLeftTrend

Scrolls the **Trend** window to the left side, by one-tenth (1/10) of the page duration.



Note: with the real-time trends pause the trend using the **PauseTrend** action, or the window will be continuously shifted to the current value.

### ScrollRightTrend

Scrolls the **Trend** window to the right side, by one-tenth (1/10) of the page duration.



Note: with the real-time trends pause the trend using the **PauseTrend** action, or the window will be continuously shifted to the current value.

### PageLeftTrend

Scrolls the **Trend** window by one-page. For example, if the page size is 10 minutes, then use the **PageLeftTrend** action to scroll the trend left for 10 minutes.

### PageRightTrend

Scrolls the **Trend** window by one-page. For example, if the page size is 10 minutes, then use the **PageRightTrend** action to scroll the trend right for 10 minutes.

### PageDurationTrend

Sets the page duration of the **Trend** window.

Define trend name and page duration.



Note: you can set page duration at run time using a combo box widget.

## ZoomInTrend

Reduces page duration.

## ZoomOutTrend

Extends page duration.

## ZoomResetTrend

Reset the zoom level back to the original zoom level.

## PauseTrend

Stops plotting the trend curves in the **Trend** window.

When used with real time trend the plotting stops when the curve reaches the right border of the graph. This action does not stop trend logging.

## ResumeTrend

Resumes trend plotting if paused.

## ShowTrendCursor

Shows value of the curve at a given point on the X axis.

It activates the trend cursor. A cursor (vertical line) will be displayed in the trend widget.

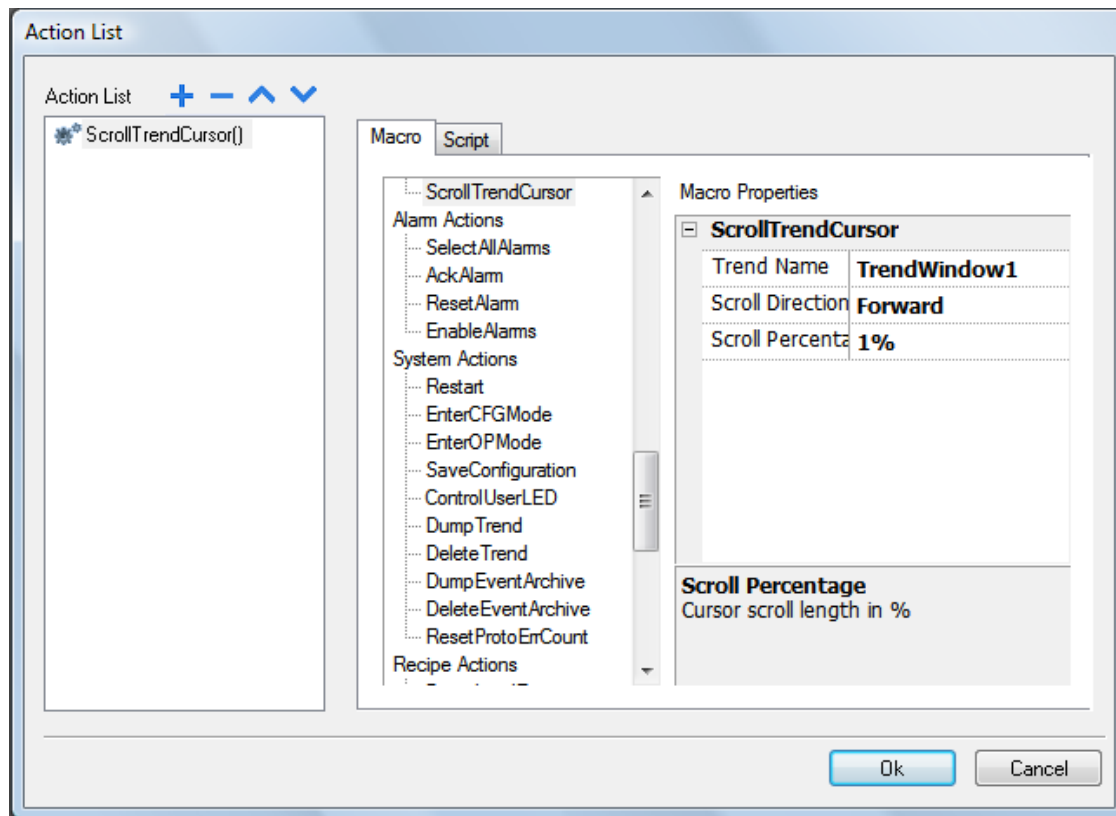
When the graphic cursor is enabled, the scrolling of the trend is stopped.

The **ScrollCursor** action moves the graphic cursor over the curves, or over the entire **Trend** window.

## ScrollTrendCursor

Scrolls the trend cursor backward or forward.

The Y cursor value will display the trend value at the point of the cursor. Scrolling percentage can be set at 1% or 10%. The percentage is calculated on the trend window duration.



## ScrollTrendToTime

Scrolls the **Trend** window to a specified point in time.

Use this action when you need to scroll to a specific position in a trend window when a specific event occurred.

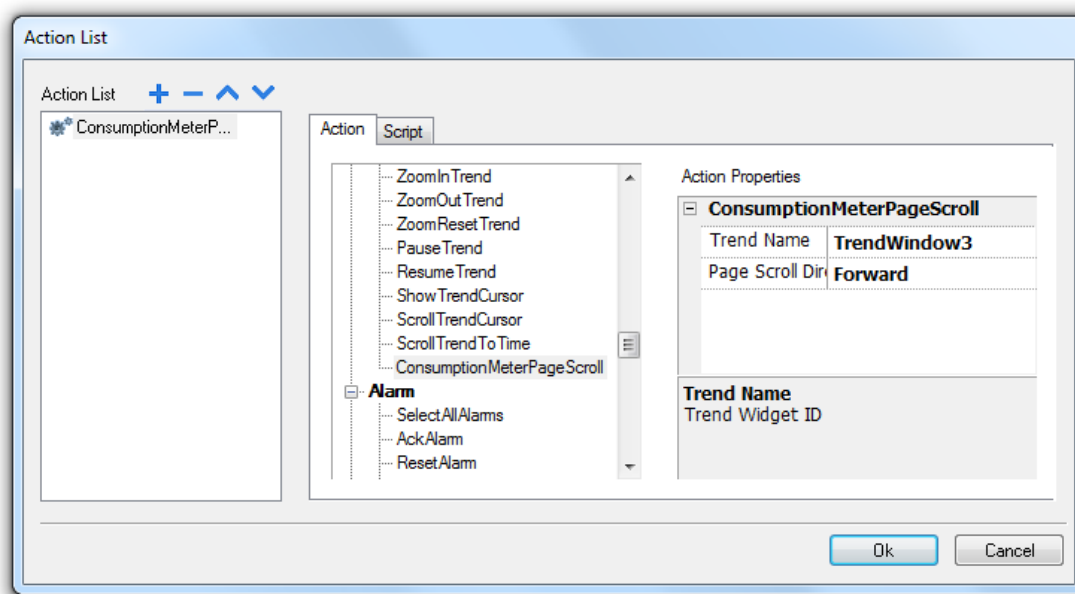
### Example

1. Configure an action for an event (for example, an alarm) that executes a data transfer of the system time into a tag.
2. Select that tag as **ScrollTrendToTime** parameter: the trend windows will be centered at the time when the event was triggered.

## ConsumptionMeterPageScroll

Scrolls the page backward or forward in a Consumption Meter widget.

Parameter	Description
<b>Trend Name</b>	Trend widget ID (for example, TrendWindow3)
<b>Page Scroll Direction</b>	Direction of page scrolling (Forward/backward)



## User management actions

User management and security settings.

### LogOut

Logs off the current user. The default user is then automatically logged in. If no default user has been configured, the logon window is displayed.

### SwitchUser

Switches between two users without logging off the logged user: the user login dialog appears. User can click **Back** to go back to the previously logged user.

User name:

Password:

☐ Show password

The server continues running with the previously logged user, until the next user logs on. One user is always logged onto the system.

### ResetPassword

Restores the original password together with the settings specified in the project for the current user.

No parameter is required.

## AddUser

*Reserved to users with **Can manage other users** property set.*

Adds a user at run time: a dialog appears.

---

User name:	<input type="text" value="user3"/>
Password:	<input type="password" value="*****"/> <input type="checkbox"/> Show password
Group:	<input type="text" value="admin"/>
Comments:	<input type="text"/>

---

Password must contain number:	<input type="checkbox"/>
Password must contain special character:	<input type="checkbox"/>
User must change his initial password:	<input type="checkbox"/>
Enable logoff time:	<input type="checkbox"/>
Inactivity logoff time:	<input type="text" value="0"/> min

---

<input type="button" value="Add"/>	<input type="button" value="Cancel"/>
------------------------------------	---------------------------------------

---

## DeleteUser

*Reserved to users with **Can manage other users** property set.*

Deletes a user at run time: a dialog appears.

No parameter is required.

User name:	<input type="text" value="admin"/>
Group:	<input type="text" value="admin"/>

<input type="button" value="Delete"/>	<input type="button" value="Cancel"/>
---------------------------------------	---------------------------------------

## EditUsers

*Reserved to users with **Can manage other users** property set.*

Edits user settings.

---

User name:

Password:  ☐ Show password

Group:

Comments:

---

Password must contain number: ☐

Password must contain special character: ☐

User must change his initial password: ☐

Enable logoff time: ☒

Inactivity logoff time:  min

---

---

## DeleteUMDynamicFile

Deletes the dynamic user management file. Changes made to users settings at run time are erased. The original settings are restored from the project information.

No parameter is required.

## ExportUsers

Exports user settings to an .xml file (*usermgnt\_user.xml*) in encrypted format to be restored when needed.

Set destination folder for the export file.



**Important: The user file is encrypted and cannot be edited.**



Note: supported formats are FAT or FAT32. NTFS format is not supported.

## ImportUsers

Imports user settings from a previously saved export .xml file (*usermgnt\_user.xml*).

Set source folder for the import file.



Note: supported formats are FAT or FAT32. NTFS format is not supported.

# Widget actions

## ShowWidget

Shows or hides page widgets.



Property	Description
<b>Widget</b>	Widget to show/hide

## SlideWidget

Shows the sliding effect of a widget, or of a widget group.



Note: The widget or grouped widgets can actually be outside of visible part of the page in the project and slide in and out of view.

Property	Description
<b>Widget</b>	Widget to slide
<b>Direction</b>	Sliding direction
<b>Speed</b>	Transition speed of sliding widget
<b>X Distance</b>	Travel distance of X coordinate in pixels
<b>Y Distance</b>	Travel distance of Y coordinate in pixels
<b>Slide Limit</b>	Enable/Disable movement limits of the widget with respect to the x, y coordinates
<b>X Limit</b>	Limit position of slide action for x coordinate
<b>Y Limit</b>	Limit position of slide action for y coordinate
<b>Toggle Visibility</b>	Show/hide widget at the end of each slide action
<b>Image Widget</b>	Image displayed during slide action

## BeginDataEntry

Displays a keypad and starts data entry on a data field without touching the widget itself. This action can be used to activate data entry using a barcode scanner.

### Java Script Interface

```
project.beginDataEntry(wgtName [, pageName])
```

Parameter	Description
<b>wgtNameWidget</b>	Widget name
<b>pageName</b>	Active page for data entry. Optional parameter. Useful to select a data field inside a non-modal active dialog box.

## TriggerIPCamera

Captures an image from an IP Camera. Only works on pages that include an IP Camera widget.

## MoveIPCamera

Sends remote commands to a camera that supports them. See ["IPCamera widgets" on page 229](#) for details. Make sure that the IP Camera supports movement commands.

## RefreshEvent

Refreshes the event buffer for **Alarm History** widget. See ["Alarms History widget" on page 119](#) for details.

## ContextMenu

Displays the context menu.

If **Context Menu** property of Project Widget has been set to **On delay** context menu can appear also touching for a few seconds the background area of the screen. See ["Project properties pane" on page 40](#)

## ReplaceMedia

Replaces existing media files with new files from USB/SD card. Can be used to replace video files of MediaPlayer widgets, or images of project.



Note: New media files must have same name and format of the files to be replaced.

Parameter	Description
<b>Media Type</b>	Type of file to update
<b>Device</b>	Device where new media files are supplied
<b>sourcePath</b>	Folder where new media files are stored (for example, "\USBMemory")
<b>Image Resize</b>	Resizes new images to the size of images to be replaced. Not applicable to video files.
<b>Silent</b>	Replaces media automatically. As default a dialog is displayed for the user to specify file location.

## Java Script Interface

```
void replaceMedia(var sourcePath, var bSilent, var Device, var nMediaType, var bResize)
```

```
project.replaceMedia("Images", true, "\USBMemory", 1, true);
```

# 10 Using the Client application

---

HMI Client is a standalone application which provides remote access to the HMI Runtime, and is included in the PB610-B Panel Builder 600. HMI Client uses the same graphic rendering system as the runtime in the HMI devices, it relies on a specified HMI Runtime as server for live data.

To run the HMI Client application:

1. From the **Start** menu > **PB610-B Panel Builder 600** > **HMI Client**: the client opens in a browser-like style window.
2. Type the server/device IP address in the address bar (for example: <http://192.168.1.12>): HMI Client will connect to the server and the same graphical application running on the device will be loaded in the client window.

HMI Client acts as a remote client and communicates to the server, sharing the local visualization with the tag values that are maintained or updated by the communication protocol.



HMI projects contain properties indicating which page is currently displayed on the HMI and can force the HMI to switch to a specific page. You can use these properties to synchronize pages showed on the HMI device and HMI Client or to control an HMI device with a PLC.

See ["Behavior" on page 45](#) for details.

---

<b>The Client application toolbar .....</b>	<b>104</b>
<b>Workspace .....</b>	<b>104</b>
<b>Settings and time zone options .....</b>	<b>104</b>
<b>Transferring files to a remote HMI device .....</b>	<b>105</b>

# The Client application toolbar

Panel Address :   

Element	Description
HMI server address	HMI device address
Connection status	Network request status. Red during data exchange.
Reload from cache	Reloads project
BookMark	Bookmarks preferred pages and reload them.
Settings	Opens <b>Settings</b> dialog

## Reload options

Option	Description
F5	Reloads project from cache
Shift + F5	Downloads project to client

## Workspace

Project files are uploaded from the device and stored in HMI Client into the following cache folder.

`%appdata%\ABB\[build number]\client\cache`

where:



`[build number]` = folder named as build number, for example 01.90.00.608.

## Settings and time zone options

In the **Settings** dialog you can configure client settings and decide how to display project timestamp information.

### HTTP settings

Parameter	Description
Protocols	Communication protocol used by HMI Client to communicate with an HMI device.
Update Rate	Polling frequency to synchronize data from server. Default = 1 s.
Timeout	Maximum wait time before a request is repeated by the HMI Client. Default = 5 s.
Reuse connection	Enables reuse of the same TCP connection for multiple HTTP requests to reduce network traffic.

Parameter	Description
	 Note: When enabled, this option may cause high latency if the proxy server does not immediately terminate old requests thus saturating connection sockets. This is often the case with 3G connections.
Enable compression	Compresses data to reduce download times. Default = disabled.  <b>CAUTION: enabling this option could causes excessive CPU overhead.</b>
Time Settings	Used by the client to adapt the widget time stamp information.

## FTP settings

Parameter	Description
Port	FTP communication port

## Time settings

Parameter	Description
Use Widget Defaults	Displays time information according to the widget settings.
Local Time	Translates all timestamps in the project into the computer local time where the client is installed.
Global Time	Translates all timestamps in the project into UTC format.
Server Time	Translates all timestamps in the project into the same used by HMI device/server in order to show the same time.



**Important: Make sure you set the HMI RTC correct time zone and DST options.**

## Transferring files to a remote HMI device

You can upload and download files to and from a remote HMI device using two dedicated actions. These actions can only be used from a remote HMI Client and access remote files via FTP.



**Important: Enable FTP support and give all necessary user rights to the folders used to transfer files.**

See ["Remote Client actions" on page 87.](#)

See ["Remote Client variables" on page 70.](#)



# 11 Using the integrated FTP server

HMI Runtime system uses an integrated FTP server.

Connect to the HMI device FTP server using any standard FTP client application. The FTP server responds on the standard port 21 as default.



**Important:** The server supports only one connection at a time; if you are using a multiple connection FTP client disable this feature on the client program or set the maximum number of connections per session to 1.

## FTP settings

### FTP default credentials

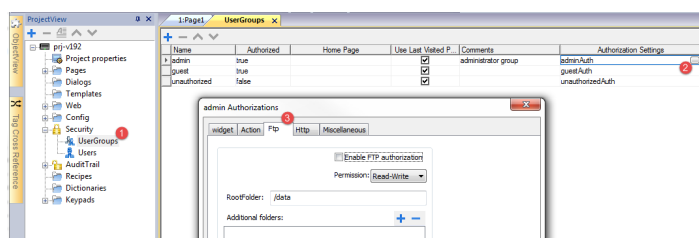
When User Management/Security is disabled use the following credentials for incoming connections:

User name	admin
Password	admin

### Changing FTP settings

**Path:** *ProjectView* > *Security* > *UserGroups* > *Authorization Settings*

You can change FTP permissions and account information in the **Ftp** tab of the **admin authorizations** dialog.



See ["Configuring groups and authorizations"](#) on page 172 for details.





# 12 Alarms

---

The alarms handling system has been designed to provide alerts through pop-up messages, typically to display warning messages indicating any abnormal condition or malfunction in the system under control.

Whenever a bit changes, or the value of a tag exceeds a threshold set in the alarm configuration, a message is displayed. Specific actions can also be programmed to be executed when an alarm is triggered.



**Important: No default action is associated with any alarm.**

You can define how an alarm is displayed on the HMI device, if it requires user acknowledgment, and if and how it is logged into the event list.

Alarms are configured in the Alarms Configuration Editor and, thus, are available for all the pages of the project. An alarm widget can display more than one alarm at a time, if sized appropriately. You can trigger the opening or closing of the Alarm window with an event.

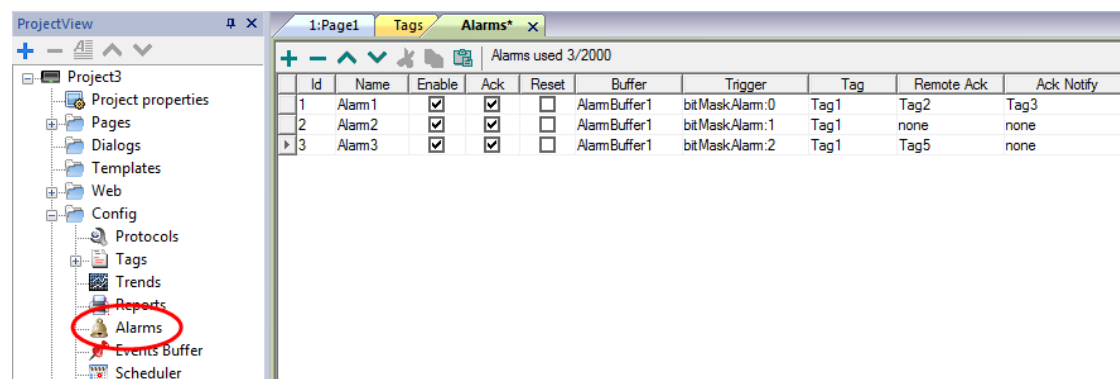
You work with alarms in the same way as you work with any other event. You may not want to display a dialog when an alarm is triggered and you can associate to it any other available action.

---

<b>Alarms Editor</b> .....	<b>110</b>
<b>Remote alarms acknowledge</b> .....	<b>112</b>
<b>Alarm state machine</b> .....	<b>112</b>
<b>Setting events</b> .....	<b>113</b>
<b>Active Alarms widget</b> .....	<b>115</b>
<b>Alarms History widget</b> .....	<b>119</b>
<b>Managing alarms at run time</b> .....	<b>119</b>
<b>Enable/disable alarms at run time</b> .....	<b>119</b>
<b>Displaying live alarm data</b> .....	<b>120</b>
<b>Exporting alarm buffers to .csv files</b> .....	<b>121</b>
<b>Exporting alarm configuration</b> .....	<b>121</b>


# Alarms Editor

Path: **ProjectView** > **Config** > double-click **Alarms**



## Adding an alarm

Click **+** to add an alarm.

Parameter	Description
<b>Name</b>	Name of alarm
<b>Enable</b>	Enable/disable triggering of alarm.  Alarms can be enabled or disabled at run time as well (see <a href="#">"Enable/disable alarms at run time" on page 119</a> for details).
<b>Ack</b>	Enable/disable acknowledgment of alarm, if selected the operator must acknowledge the alarm once triggered to remove it from the <b>Active Alarm</b> widget.
<b>Reset</b>	Used with the <b>Ack</b> option, if selected, acknowledged alarms stay in the alarm list, labeled as <b>Not Triggered Acked</b> , until the operator presses the <b>Reset</b> button in the alarm widget.
<b>Buffer</b>	Buffer file where the alarm history will be saved.
<b>Trigger</b>	Triggering condition depending on alarm type: <ul style="list-style-type: none"> <li><b>limitAlarm</b>: alarm triggered when tag value exceeds its limits. The alarm is not triggered if the value reaches the limits.</li> <li><b>valueAlarm</b> alarm is triggered when tag value is equal to the configured value</li> <li><b>bitMaskAlarm</b>: the bitwise AND operator compares each bit of the bitmask with the tag value corresponding to that Alarm. If both bits are on, the alarm is set to true. You can specify one or more bit positions (starting from 0) inside the tag. The Bit position must be given in decimal format; if more bits are specified, each position must be separated by a ",".</li> <li><b>deviationAlarm</b>: alarm triggered if the percentage of deviation of the tag value from the set point exceeds a set deviation.               <math display="block"> Value_{now} - SetPoint  &gt; \left( \frac{deviation}{100} \times SetPoint \right)</math> </li> </ul>

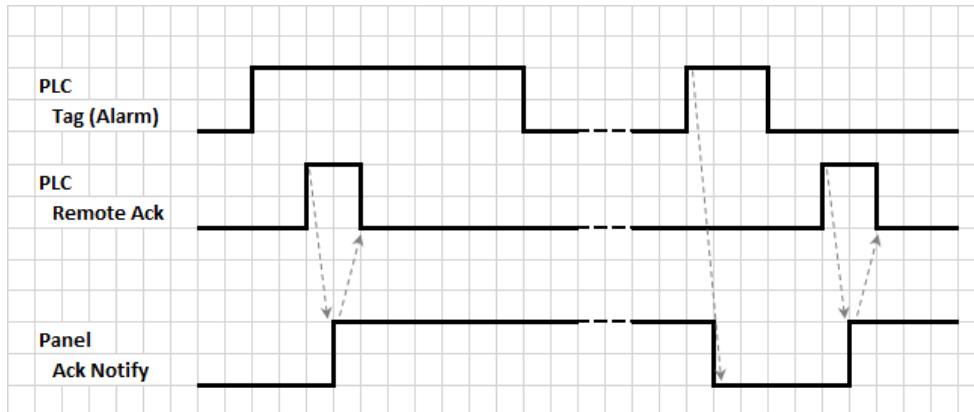
Parameter	Description
<b>Tag</b>	<p>Tag whose value will trigger the alarm when it exceeds the set limits.</p> <p>The alarm can refer to the value of this tag, or to the state of a bit if <b>bitMaskAlarm</b> has been selected as trigger.</p>
<b>Remote Ack</b>	<p>Tag used by the PLC to acknowledge the alarm. A transition of this tag from 0 to a non zero value is considered an acknowledgment request.</p> <p>Leave empty if remote acknowledgment is not required.</p> <p>See <a href="#">"Remote alarms acknowledge" on the next page</a> for details.</p>
<b>Ack Notify</b>	<p>Tag used by the HMI device to notify when the alarm is acknowledged from the device or from the PLC.</p> <p>0 = set to this value when alarm is triggered</p> <p>1 = set to this value when alarm is acknowledged.</p>
<b>Actions</b>	<p>Actions executed when the alarm is triggered. Additional conditions can be specified in the <b>Events</b> column.</p> <p>See <a href="#">"Setting events" on page 113</a> for details.</p>
<b>Description</b>	<p>Alarm description. This text supports the multiple language features and can be a combination of static and dynamic parts, where the dynamic portion includes one or more tag values.</p> <p>See <a href="#">"Displaying live alarm data" on page 120</a> for details.</p>
<b>Color</b>	<p>Foreground and background colors of alarm rows based on the status of alarm.</p>
<b>AckBlink</b>	<p>Blinking for triggered alarms. If selected the alarm rows blinks until acknowledged. Only effective if <b>Ack</b> is selected.</p>
<b>Severity</b>	<p>Severity of the alarm. If multiple alarms are triggered simultaneously, actions will be executed based on severity settings.</p> <p>0 = not important</p> <p>1 = low</p> <p>2 = below normal</p> <p>3 = normal</p> <p>4 = above normal</p> <p>5 = high</p> <p>6 = critical</p>
<b>Events</b>	<p>Conditions in which the alarms are notified, logged or printed.</p> <p>See <a href="#">"Setting events" on page 113</a> for details.</p>

# Remote alarms acknowledge

When the **Remote Ack** parameter is set, an alarm can be acknowledged from a PLC device setting a tag value to a nonzero value. The acknowledged status is notified to the PLC device by the **Ack Notify** flag.

## Alarms acknowledgement process

**Remote Ack** tag is set/reset by the PLC to request the acknowledge, and **Ack Notify** is set/reset by HMI device to notify the execution of the acknowledge.



1. When an alarm condition is detected the HMI device set **Ack Notify** to 0 and all related actions are executed.
2. When the alarm is acknowledged (by HMI device or remotely), **Ack Notify** is set to 1
3. It's up to the controller to set **Remote Ack** to 1 to acknowledge the alarm or reset it to 0 when the HMI device send a notification that the alarm has been acknowledged (**Ack Notify** = 1)



**WARNING:** When an alarm is triggered, some signals need to be update/communicated through the connected devices. We assume the Acknowledge to be a signal pushed from an operator and not released automatically from a controller device. This allows for time required to communicated the original signals.

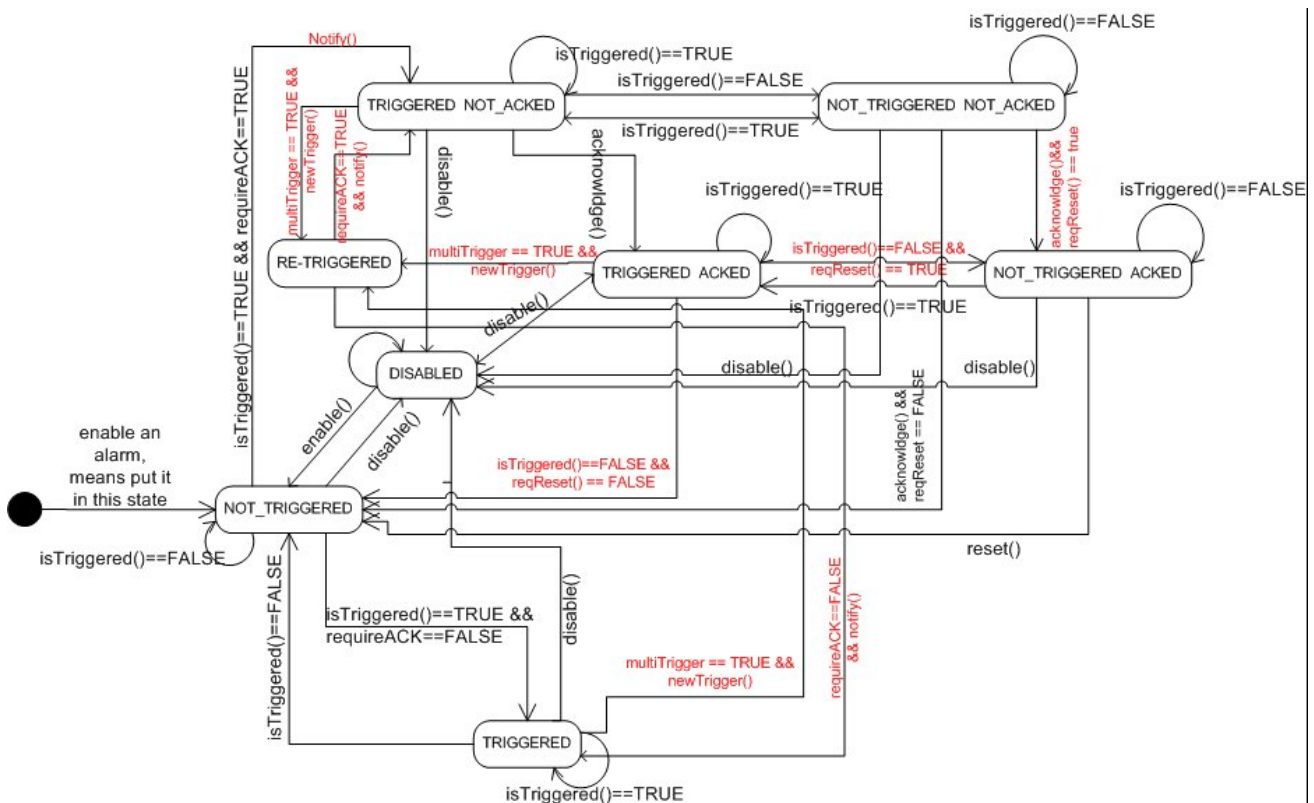


**Tip:** Using the same tag both for **Remote Ack** and **Ack Notify** can connect more devices to the same controller and acknowledge the alarms from any HMI device.

## Alarm state machine

The runtime implements the alarm state machine described in this diagram.

States and transitions between states are described according to the selected options and desired behavior.



## Setting events

Path: **ProjectView**> **Config** > **Alarms** > **Events** column

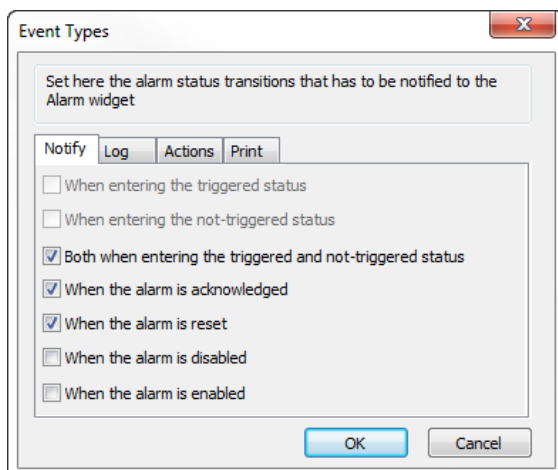
Events are defined using the Alarms Editor.

See "[Alarms Editor](#)" on page 110 for details.

## Notifying events

Path: **ProjectView**> **Config** > **Alarms** > **Events** column > **Notify** tab

Set conditions under which the alarms will be posted in the alarm widget.



Here you define the behavior of the default alarm widget available in the Widget gallery and decide in which cases the widget is updated by a change in an alarm status.



**CAUTION:** Make only the adjustments required by the specific application while leaving all other settings as default.

## Logging events

Path: **ProjectView** > **Config** > **Alarms** > **Events** column > **Log** tab

Set conditions for which you want to store the specific event in an alarm history buffer.

The alarm history is logged in the Event Buffer.

## Executing actions

Path: **ProjectView** > **Config** > **Alarms** > **Events** column > **Actions** tab

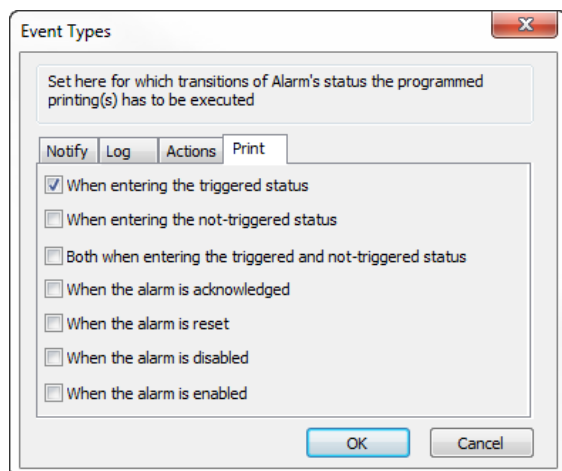
Set conditions under which the action(s), configured for the specific alarm, must be executed.

By default, actions are executed only when the alarm is triggered; other alarm states can also be set to execute actions.

## Print events

Path: **ProjectView** > **Config** > **Alarms** > **Events** column > **Print** tab

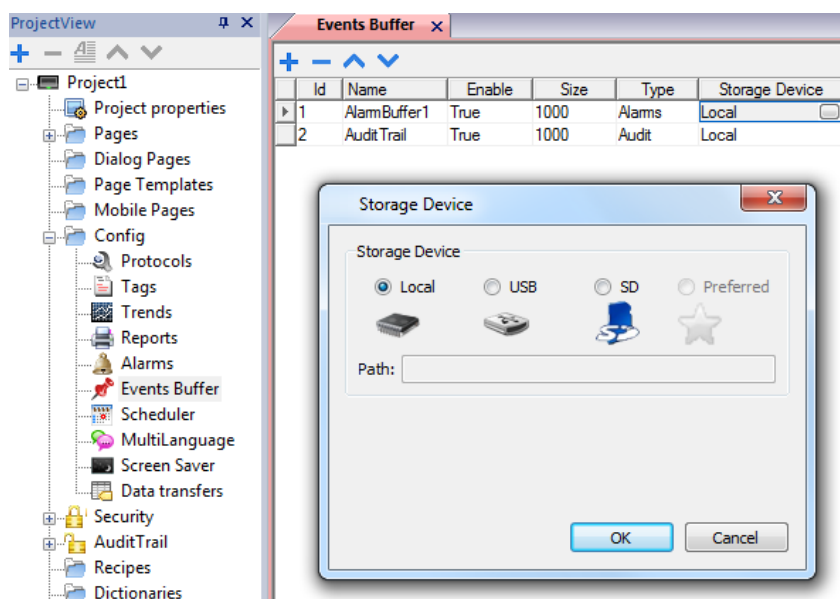
Set conditions for which you want to print the specific event



## Setting storage device

Path: **ProjectView** > **Config** > **Events Buffer** > **Storage Device** tab

1. Open the **Storage Device** dialog.
2. Select a device for event data storage.



Data is automatically saved every five minutes except for alarm data which is saved immediately.

## Active Alarms widget

You can insert the **Active Alarms** widget in a page to display the alarms and to acknowledge, reset or enable/disable them.

Active Alarms

Select	Name	State	Value	Time	Description	Severity	Enable

Filter : Hide Not Triggered

## Alarm filters

*Path: **ActiveAlarm** widget> **Properties** pane> **Filter***

Define filters used to display only some of the configured alarms. Filters are based on alarm fields, which means you can filter alarms according to name, severity, description and so on.

Filter 1 is the default filter. It's managed by the combo box **Filter 1**, and has two options: **Show all alarms** and **Hide Not Triggered** which, when selected, allows to display only active alarms.

Filter 2 is, by default, not configured and available for customization.

Filter's expressions make use of AWK language, the expressions are applied to the data contained in the selected **Filter** column of the Alarm widget.

- Alarms List	
Columns	
Sorting	false
Sort Column	Severity
+ Text	
- Filter	
Filter Column	State
- Filter 1	Hide Not Triggered
DataLink	itemData:Combo2
Filter Column	Select
Filter 2	

## Setting filters

*Path: **ActiveAlarm** widget> **Properties** pane> **Filter***

To set one of the two available filters:

1. Select **Filter Column 1** and choose the value to filter for (e.g.: Name, State, Time)
2. In **DataLink** attach a combo box widget. Use Shift+ left-click to select the combo box.
3. In the **Properties** pane select list property and open dialog to customize combo box values
4. In the combo box configuration dialog, specify **String List** and the regular expression to filter values.

See <http://www.gnu.org/software/gawk/manual/gawk.html> for details on how to use regular expressions.



### Filters first example

You want to show all alarms matching Filter 1 with value equal to 10. Then properties settings: **Filter column 2 = Value**, **Filter 2 = 10**

The screenshot shows the 'Active Alarms' window with a table containing 'State' and 'Value' columns. Below the table are buttons for 'Ack', 'Reset', and 'Save', and a dropdown menu showing 'le Not Triggered'. To the right is the 'Properties' panel for the 'Alarms List'. It includes settings for 'Columns', 'Sorting' (false), 'Sort Order' (Descending), and 'Sort Column' (Severity). Under the 'Filter' section, 'Filter Column 1' is set to 'State' and 'Filter 1' is set to '^((Not Triggered Acked|I'. 'Filter Column 2' is set to 'Value' and 'Filter 2' is set to '10'. These two filter settings are circled in red.

### Filters second example

You want to show all alarms matching a Severity value from 3 to 6 (Normal to Critical). Then properties settings: **Filter column 2 = Severity**, **Filter 2 = [3-6]**

The screenshot shows the 'Active Alarms' window with a table containing 'State' and 'Value' columns. Below the table are buttons for 'Ack', 'Reset', and 'Save', and a dropdown menu showing 'le Not Triggered'. To the right is the 'Properties' panel for the 'Alarms List'. It includes settings for 'Columns', 'Sorting' (false), 'Sort Order' (Descending), and 'Sort Column' (Severity). Under the 'Filter' section, 'Filter Column 1' is set to 'State' and 'Filter 1' is set to '^((Not Triggered Acked|I'. 'Filter Column 2' is set to 'Severity' and 'Filter 2' is set to '[3-6]'. These two filter settings are circled in red.

### Filters third example

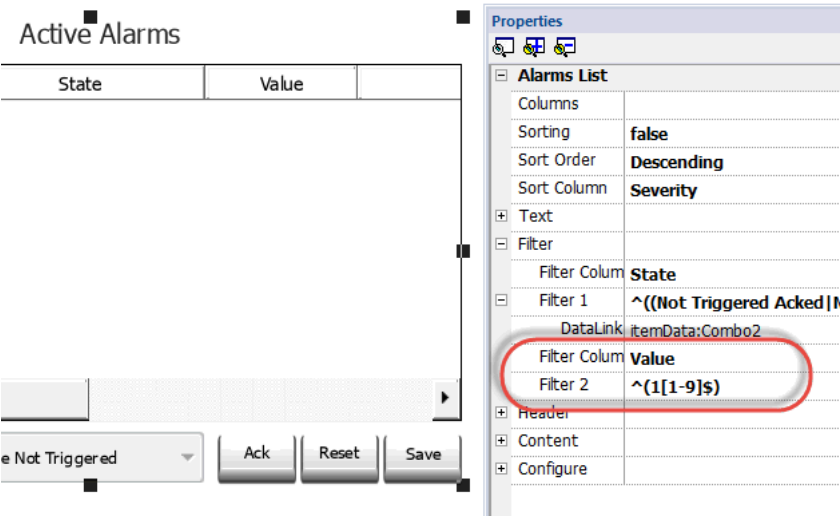
You want to show all alarms matching a value from 11 to 19. Then properties settings: **Filter column 2 = Severity**, **Filter 2 = ^(1[1-9])\$**

Meaning:

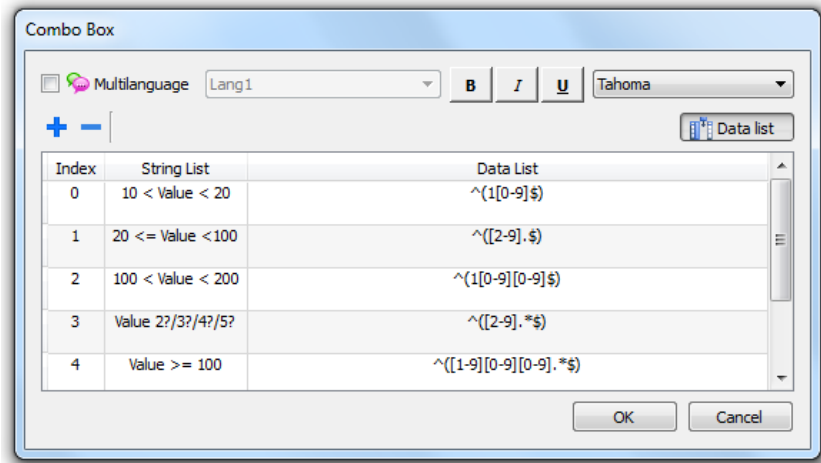
^ = match must starts from the beginning of the string

1[1-9] = first char must be 1 and the second char must be between 1 and 9

\$ = end of the comparison.



Filters expression examples



Filter by	String list	Data list
State	Hide Not Triggered	^((Not Triggered Acked Not Triggered Not Acked Triggered).*\$)
Value	10 < Value < 20	^(1[0-9]\$)
Value	20 <= Value < 100	^([2-9].\$)
Value	100 < Value < 200	^(1[0-9][0-9]\$)
Value	Value 2?/3?/4?/5?	^([2-9].*\$)
Value	Value >= 100	^([1-9][0-9][0-9].*\$)
Value	Value >= 20	^([2-9].*\$ [1-9][0-9][0-9].*\$)

Sorting alarms

Path: **ActiveAlarm** widget> **Properties** pane> **Sorting**

The sorting function allows you to sort alarms at run time in the alarms widget by clicking on the column header.



Note: The severity value displayed here is set in the Alarm Editor.




## Alarms History widget

Logs and display an alarm list if **Buffer** property in Alarms Configuration Editor is set.

Alarms History

From : 09/24/13 - 16:04:49      Duration : 1 Min      Refresh

To : 09/24/13 - 16:04:49

Name	State	Value	Time	Description	Event Type
<div style="text-align: center;">    </div>					

Backward      Forward

### Attaching widget to buffer

Path: **AlarmHistory** widget> **Properties** pane> **Buffer** > **EventBuffer**

In **Properties** pane > **Event** select the **Event Buffer** from which the alarm list is retrieved

## Managing alarms at run time

When an alarm is triggered it is displayed in the Active Alarms widget where you can acknowledge and reset it. You can filter the alarms displayed using several filters, for example you can hide not triggered alarms or show all alarms.

See "[Active Alarms widget](#)" on page 115 for details.



**IMPORTANT:** The Active Alarms widget is not displayed automatically. You must add a dedicated action that will open the page containing the alarm widget when the alarm is triggered.

## Enable/disable alarms at run time

You can enable or disable the alarms at run time.

To enable an alarm select the **Enable** option in the alarm widget.

Disabled alarms are not triggered and therefore not displayed at run time.

Select	Id	Source Value	State	Date	Time	Enable
<input type="checkbox"/>	Alarm1	23	Not Triggered Not Acked	25-01-2011	16:59:31	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Alarm2	23	Not Triggered Not Acked	25-01-2011	16:59:31	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Alarm3	23	Not Triggered Not Acked	25-01-2011	16:59:31	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Alarm4	23	Not Triggered Not Acked	25-01-2011	16:59:31	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Alarm5	23	Not Triggered Not Acked	25-01-2011	16:59:31	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Alarm6	23	Not Triggered Not Acked	25-01-2011	16:59:31	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Alarm7	23	Not Triggered Not Acked	25-01-2011	16:59:32	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Alarm8	23	Not Triggered Not Acked	25-01-2011	16:59:32	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Alarm9	23	Not Triggered Not Acked	25-01-2011	16:59:32	<input checked="" type="checkbox"/>

Check/Uncheck All   Filter: Show All   Ack   Reset   Save

## Displaying live alarm data

Tag values can be included in the alarm description of the event buffer only from version 1.80.

Path: **ProjectView> Config > double-click Alarms**

Both in the Active Alarms and in the History Alarms widget you can set the alarm description to display live tag data.

Id	Name	Enable	Ack	Reset	Tag	Buffer	Trigger	Action	Description
1	Alarm1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Tag1	AlarmBuffer1	bitMaskAlarm:	ShowDialog	Alarm 1 Tag Value is [Tag1]
2	Alarm2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Tag1	AlarmBuffer1	bitMaskAlarm:1	ShowDialog	Alarm 2 Tag Value is [Tag2]
3	Alarm3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Tag1	AlarmBuffer1	bitMaskAlarm:1	ShowDialog	Alarm 3 Tag Value is [Tag3]

To show the tag value, set a placeholder in **Description** entering the tag name in square brackets, for example "[Tag1]". At run time, in **Description** column of Active Alarms widget the current value of the tag will be displayed. In History Alarms widget or in .csv file the value at the time the alarm was triggered is displayed



Use '\ ' before '[' ']' if you want to show the '[' ']' in the description string, for example: `\[Tag\[1\]\]` will display the string "[Tag[1]]".

### Example of Alarm widget

Select	Id	Source Value	State	Description	Date
<input type="checkbox"/>	Alarm1	123	Triggered Not Acked	Alarm 1 Tag value is 123	25-01-2011
<input type="checkbox"/>	Alarm2	1234	Triggered Not Acked	Alarm 2 Tag value is 1234	25-01-2011
<input type="checkbox"/>	Alarm3	456	Triggered Not Acked	Alarm 3 Tag value is 456	25-01-2011
<input type="checkbox"/>	Alarm4	987	Triggered Not Acked	Alarm 4 Tag value is 987	25-01-2011
<input type="checkbox"/>	Alarm5	555	Triggered Not Acked		25-01-2011
<input type="checkbox"/>	Alarm6	1234	Triggered Not Acked		25-01-2011
<input type="checkbox"/>	Alarm7	1234	Triggered Not Acked		25-01-2011

Check/Uncheck All   Filter: Hide Not Triggered   Ack   Reset   Save



Note: The csv file resulting from the dump of the alarm events list will also display the tag value in the description column.

## Exporting alarm buffers to .csv files

To export an event buffer containing an history alarms list, use the **DumpEventArchive** action.

See "System actions" on page 88 for details.



**Note:** Tag values displayed in the alarms description are also included in the buffer. Tags are sampled when the alarm is triggered and that value is logged and included in the description.

## Exporting alarm configuration

Path: **ProjectView> Config > double-click Alarms**

Name	Enable	Ack	Buffer	Trigger	Tag
Alarm1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	AlarmBuffer1	bitMaskAlarm:0	MRTU1
Alarm2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	AlarmBuffer1	deviationAlarm:50.0	MRTU2
Alarm3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	AlarmBuffer1	limitAlarm:10-100	Tag1
Alarm4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	AlarmBuffer1	valueAlarm:30	Tag2
Alarm5	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	AlarmBuffer1	valueAlarm:@Tag4	Tag3
Alarm6	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	AlarmBuffer1	bitMaskAlarm:0	Application/IOCONFIG_GLOBALS_MAPPING/IN0
Alarm7	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	AlarmBuffer1	bitMaskAlarm:0	Application/IOCONFIG_GLOBALS_MAPPING/IN1
Alarm8	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	AlarmBuffer1	deviationAlarm:50.0	Application/PLC_PRG/supercar

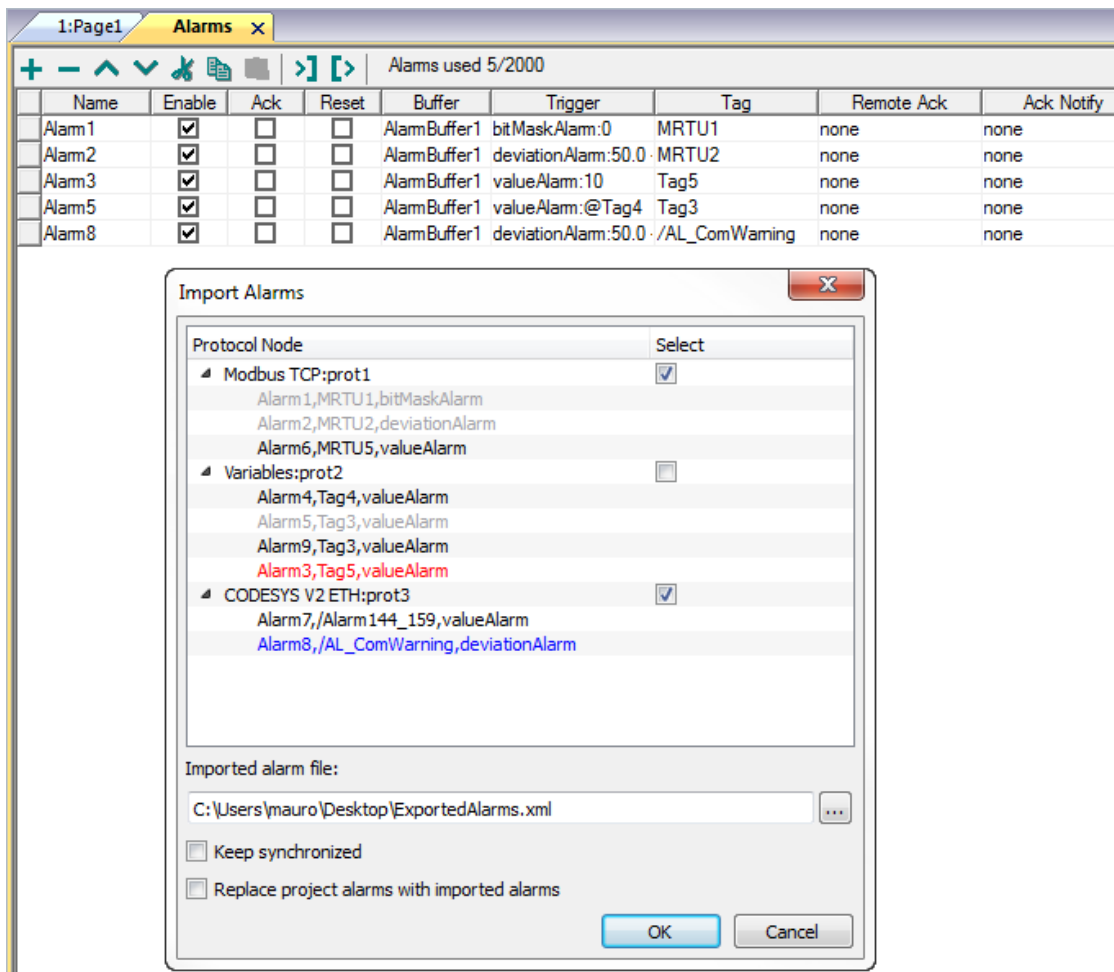
Click the **Export Alarms** button: the alarms configuration table is exported into an .xml file.

You can edit the resulting .xml file using third part tools (for example, Microsoft Excel).

	A	B	C	D	E	F	G
	eventBuffer	logToEventArchive	eventType	subType	storeAlarmInfo	name	source
1	n/a	TRUE	0	0	FALSE	n/a	n/a
2	AlarmBuffer1	TRUE	14	1	TRUE	Alarm1	MRTU1
3	AlarmBuffer1	TRUE	14	1	TRUE	Alarm2	MRTU2
4	AlarmBuffer1	TRUE	14	1	TRUE	Alarm3	Tag1
5	AlarmBuffer1	TRUE	14	1	TRUE	Alarm4	Tag2
6	AlarmBuffer1	TRUE	14	1	TRUE	Alarm5	Tag3
7	AlarmBuffer1	TRUE	14	1	TRUE	Alarm6	Application/IOCONFIG_GLC
8	AlarmBuffer1	TRUE	14	1	TRUE	Alarm7	Application/IOCONFIG_GLC
9	AlarmBuffer1	TRUE	14	1	TRUE	Alarm8	Application/PLC_PRG/supe

## Importing alarm configuration

Path: **ProjectView> Config > double-click Alarms**



1. Click the **Import Alarms** button and select the .xml file from which to import the alarms configuration: the **Import Alarms** dialog is displayed.
2. Select the group of alarms to import and click **OK** to confirm.

Differences are highlighted in the **Import Alarms** dialog using different colors

Color	Description
Black	This is a new alarm and it will be imported
Red	This alarm has not been found and will be removed (only if check "Replace project alarms with imported alarms" is checked)
Blue	This alarm has been modified and will be updated.
Gray	This alarm is already part of the project and will be skipped.

## Automatic synchronization

Select the **Keep synchronized** option in the **Import Alarms** dialog to enable the automatic synchronization of the alarm configuration file.

Whenever changes occur in the alarms configuration, the file will be automatically updated in silent mode.



Tip: Enable this function when the alarm file is managed by a different tool (for example, PLC programming software) as well as by PB610-B Panel Builder 600.





# 13 Recipes

Recipes are collections of tag values organized in sets that satisfy specific application requirements.

For example, if you have to control room variables (temperature and humidity) in the morning, afternoon and evening. You will create three sets (morning, afternoon and evening) in which you will set the proper tag values.

Each element of the recipe is associated to a tag and can be indexed into sets for a more effective use. This feature allows you to extend the capabilities of controllers that have limited memory.

You can add controller data to a page using a recipe widget. Recipe data contains all the controller data items; however data is no longer read directly from the controller but rather from the associated recipe element in the HMI device.

Recipe data is configured in PB610-B Panel Builder 600 workspace; the user can specify default values for each element of the data records. In HMI Runtime, data can be edited and saved to a new data file, any change to recipe data is therefore stored to disk. With the use of a separate data file HMI Runtime ensures that modified recipe values are retained throughout different project updates. In other words, a subsequent project update does not influence the recipe data modified by the user in the HMI Runtime.

See ["Recipe actions" on page 84](#) for details on how to reset recipe data.



Note: Recipe data can be stored on a Flash memory, on a USB drive or on a SD card.

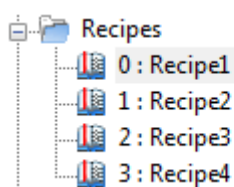
Managing recipes .....	125
Configuring a recipe widget .....	128
Recipe status .....	129
Uploading/downloading a recipe .....	129
Backup and restore recipes data .....	130

## Managing recipes

### Creating a recipe

To create a recipe for your project:

1. In **ProjectView** right-click **Recipes** and select **Insert Recipe**: an empty recipe is added. You create and configure recipes using the Recipe Editor.



## Recipe editor

Path: **ProjectView > Recipes > double-click RecipeName**

index	Element Name	Tag	Fill Tank 1	Fill Tank 3	Fill Tank 5	Fill Tank 7	Fill Tank 1	Empty Tank	Empty Tank	Empty Tank 75	Em
0	Home Valve	Recipe_HomeV: 1	1	1	1	1	0	0	0	0	0
1	Truck Valve	Recipe_TruckV: 0	0	0	0	0	1	1	1	1	1
2	Fill Flow Meter	Recipe_FillFlow: 15	35	50	75	100	75	50	25		15
3	Empty Flow Meter	Recipe_EmptyFl: 0	0	0	0	0	25	50	75		85
4	Chemical1	Recipe_Chemic: 0	0	0	0	0	0	0	0		0
5	Chemical2	Recipe_Chemic: 0	0	0	0	0	0	0	0		0

## Configuring recipe properties

In the **Properties** pane of each recipe you set the following parameters:

Parameter	Description
<b>Recipe Name</b>	Name of the recipe
<b>Number of sets</b>	Number of values sets for each recipe element. Each set has a different configurable name.

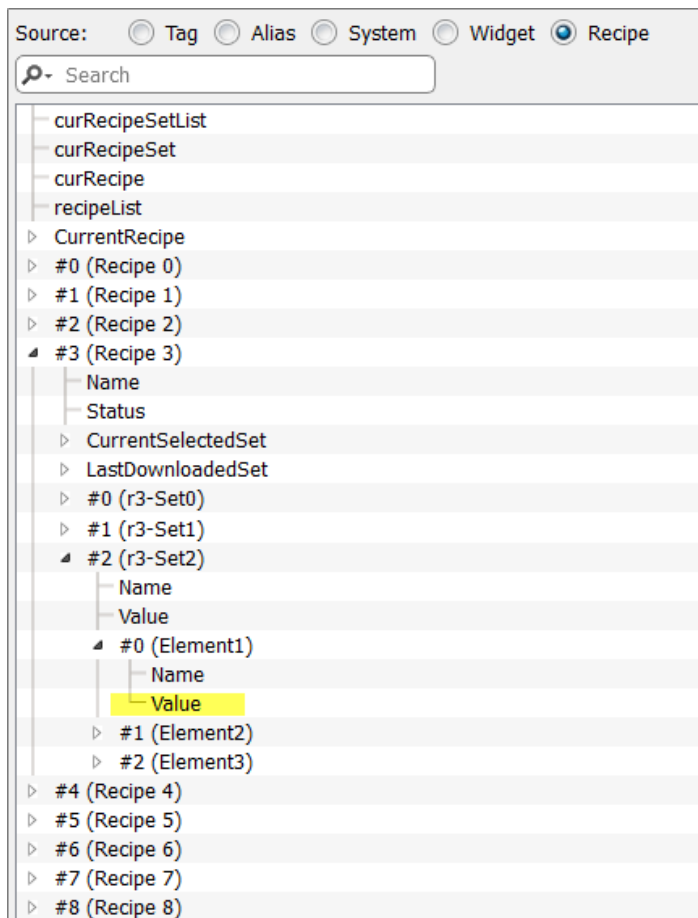
Properties	
Recipe : _RecipeMgr	
Recipe Name	Recipe1
Number of sets	10
Set 0	Fill Tank 15_
Set 1	Fill Tank 35_
Set 2	Fill Tank 50_
Set 3	Fill Tank 75_
Set 4	Fill Tank 100_
Set 5	Empty Tank 25_
Set 6	Empty Tank 50_
Set 7	Empty Tank 75_
Set 8	Empty Tank 90_
Set 9	Empty Tank 100_

## Setting up a recipe

1. Click **+** to add an element of the recipe.
2. Link the tags to each recipe element.

## Defining recipe fields

Create a recipe field in the page using a numeric widget and attaching it to a recipe item after selecting Recipe as the Source.



In the **Attach to** dialog you have the choice of all the different recipe variables, such as:


- Current Recipe > Current Selected Recipe Set > Element > Value
- Selected Recipe > Selected Set0 > Element > Value
- recipeList

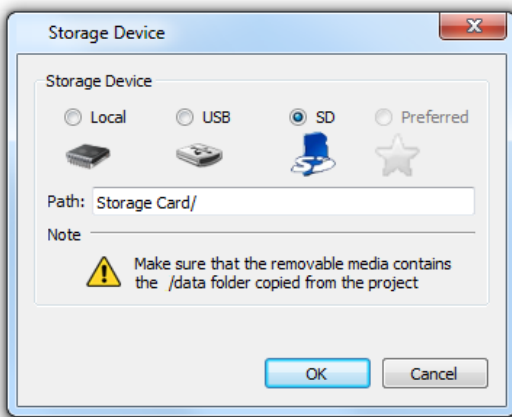
When numeric widgets are defined as read/write, the default recipe data can be edited at run time. These new values are stored in a separate file as modified recipe data.



Note: Since JavaScript API functions are used, the recipe elements can be referenced by name or by position. To avoid ambiguity, the names of the recipe elements must include at least one alphanumeric character.

## Storing recipe data

In the Recipe Editor click the storage type icon  to select where to store recipe data: the **Storage Device** dialog is displayed.



For USB drive and SD card storage you can provide the folder location.



**WARNING:** Recipe configuration files are created automatically when the project is saved and stored in the data subfolder of the project. To use external storage devices, you need to copy this folder into the external device.

Default paths are:

- SD card: */Storage Card/data*
- USB drive: */USBMemory/data*



**Important:** You can add a subfolder but you must not rename the "data" subfolder.

## Configuring a recipe widget

You can choose one of the two recipe widgets available in the **Widget Gallery**:

- **Recipe set:** allows you to select a recipe set for upload or download. See ["Uploading/downloading a recipe" on the facing page](#)
- **Recipe menu:** when more recipes have been created for a project, use this widget to manage all recipes and select the desired sets for each of them.

### Recipe Set

Recipe Set

Download

Upload

### Recipe Menu

Recipe

Recipe Set

Download

Upload

## Configuring the Recipe Set widget

In the **Properties** pane of each **Recipe Set** widget set the following parameter:

Parameter	Description
Recipe Name	Name of the recipe

## Recipe status

After every recipe upload or download, or recipe set modification, the recipe **Status** parameters contain a value with the result of the operation.

Code	Function	Description
0	Set modified	Selected set changed
1	Download triggered	Download request triggered
2	Download Done	Download action completed
3	Download Error	Error during download (for example, unknown set, unknown recipe, controller not ready, Tags write failed etc.)
4	Upload triggered	Upload request triggered
5	Upload done	Upload action completed
6	Upload Error	Error during upload - same as for download
7	General Error	General error (for example, data not available)



Note: On device startup the value of recipe **Status** is 0.

## Uploading/downloading a recipe

### Uploading a recipe

You upload a recipe to an HMI device using a recipe widget and the **UpLoadRecipe**, **UpLoadCurRecipe** action in one of the following ways:

- attach the action to an event of a button or a switch (see [""Attach to" parameters" on page 28](#) for details)
- configure the action in an alarm action list (see ["Alarm actions" on page 76](#) for details)
- configure the action in a scheduler action list (see ["Scheduling events at run time" on page 168](#) for details)

### Downloading a recipe

You download a recipe from an HMI device using a recipe widget and the **DownloadRecipe**, **DownloadCurRecipe** action. See ["Recipe actions" on page 84](#)

# Backup and restore recipes data

The recipe data stored in an HMI device can be exported for backup and later restored. This is done using the **DumpRecipeData** or the **RestoreRecipeData** actions.

See ["Recipe actions" on page 84](#) for details.

# 14 Trends

---

Trends allow you to sample and record the values of specified tags according to specific sampling conditions. The trend function includes trend acquisition and trend display.

Trend acquisition parameters are set in the Trend editor so that data can be stored. Stored data can then be displayed in a graphical format using a trend widget.

---

<b>Data logging</b>	<b>132</b>
<b>Exporting trend buffer data</b>	<b>133</b>
<b>Trend widgets</b>	<b>134</b>
<b>History trends</b>	<b>136</b>
<b>Trend widget properties</b>	<b>137</b>
<b>Values outside range or invalid</b>	<b>138</b>
<b>Showing trend values</b>	<b>139</b>
<b>Scatter diagram widget</b>	<b>140</b>

# Data logging

Data can be logged and stored to HMI memory. Data logging allows you to store the values of a group of tags all at the same time to a buffer. Data logging can be triggered by a timer or by a dedicated tag. Logged data can be exported to a .csv file or displayed using the historical trend widget. Logged data can be saved locally on a USB device or SD card, or on any available custom network folder.



**WARNING:** The operation with removable memory devices (USB Flash drives, SD memory cards) containing a very large number of files may result in a decrease of system performance.



**WARNING:** The max number of files inside a SD memory card depends on the type of formatting (e.g. FAT32 max 65536 files; FAT max 513 files).



**WARNING:** Flash cards support a limited number of write operations. We suggest to use only good quality memory cards; in the case your application use intensively the memory card consider a regular substitution of the memory card.

Storage is based on trend buffers. Trend buffers are organized as a FIFO queue: when the buffer is full, the oldest values are discarded unless you configure your trend to create a backup copy of the buffer.

## Adding a trend buffer



Path: **ProjectView** > **Config** > double-click **Trends**

1. Click **Add** to add a new buffer.
2. Click **+** next to each trend buffer to display all configuration parameters.

The screenshot shows the 'Trends' configuration window in the ProjectView. The left sidebar shows the project structure with 'Trends' selected. The main window displays the configuration for 'Trend1'. The 'Add' button is visible at the top left. The 'Total memory Space' is shown as 6.3%. The 'Storage Device' is set to 'Local'. The 'Path' is 'Data/'. The 'Buffer' section has a checkbox for 'Save a copy when full'. Below the configuration fields is a table with 3 columns: Name, Tag, and Comment.

	Name	Tag	Comment
1	Temperature	Tag1	This is a comment
2	Pressure	Tag2	
3	Umidity	Tag3	



Element	Description
<b>Total memory Space</b>	<p>Memory currently used by the trend buffer.</p> <p>This percentage is calculated as follows:</p> $\text{Total Memory Space} = \frac{\text{Total Number of Samples used in the Project}}{\text{Max Number of Samples allowed for a Project}} * 100$
<b>Trend Name</b>	Name of trend that will be displayed in the window property pane.
<b>Active</b>	<p>When enabled, the trend runs by default at system startup.</p> <p> Note: Trends cannot be activated at run time.</p>
<b>Source</b>	Tags sampled by the trend.
<b>Sampling Time (s)</b>	Sampling interval in seconds.
<b>Trigger</b>	<p>Tag triggering the sample. When the value of this tag changes, a sample is collected.</p> <p> Note: Trigger and Source can refer to the same tag.</p>
<b>Number of Samples</b>	Buffer size.
<b>Storage Device</b>	Where trend buffer data will be stored.
<b>Buffer</b>	If <b>Save a copy when full</b> option is enabled, a backup copy of the buffer data is created before it is overwritten by newer data.
<b>Sampling Filter / Trigger Filter</b>	<p>If triggering condition is time, a new sample is stored when its value, compared with the last saved value, exceeds the specified limits.</p> <p>If triggering condition is a tag, a new sample is stored at each change of the trigger tag value.</p>
<b>Sampled tags table</b>	<p><b>Name:</b> name of trend</p> <p><b>Tag:</b> tag to be sampled.</p> <p><b>Comment:</b> trend description</p>

## Exporting trend buffer data

Use the **DumpTrend** action to export trend buffer data to a .csv file.

Format of trend data exported to a .csv file can be selected from a macro parameter as shown in figure. All tags specified in the trend buffer are exported

Dump normal mode (compatibility mode)

	A	B	C	D	E	F	G	H	I	J	K
1	Type	Value	Time Stamp	Refresh Time	Quality	Type	Value	Quality	Type	Value	Quality
2	4	0	2015-09-18T14:42:22.000Z	1000	192	8	0.00E+00	192	3	0	192
3	4	0	2015-09-18T14:42:23.000Z	1000	192	8	0.00E+00	192	3	0	192
4	4	0	2015-09-18T14:42:24.000Z	1000	192	8	0.00E+00	192	3	0	192
5	4	40	2015-09-18T14:42:25.000Z	1000	192	8	0.00E+00	192	3	0	192
6	4	40	2015-09-18T14:42:26.000Z	1000	192	8	0.00E+00	192	3	0	192
7	4	40	2015-09-18T14:42:27.000Z	1000	192	8	0.00E+00	192	3	0	192
8	4	40	2015-09-18T14:42:28.000Z	1000	192	8	5.00E+01	192	3	0	192
9	4	40	2015-09-18T14:42:29.000Z	1000	192	8	5.00E+01	192	3	0	192
10	4	40	2015-09-18T14:42:30.000Z	1000	192	8	5.00E+01	192	3	0	192

Dump extended mode (compact mode)

	A	B	C	D	E	F	G
1	Timestamp	Tag1	4 Tag2	8 Tag3	3		
2		Value	Quality	Value	Quality	Value	Quality
3	2015-09-18T14:42:22.000Z	0	192	0.00E+00	192	0	192
4	2015-09-18T14:42:23.000Z	0	192	0.00E+00	192	0	192
5	2015-09-18T14:42:24.000Z	0	192	0.00E+00	192	0	192
6	2015-09-18T14:42:25.000Z	40	192	0.00E+00	192	0	192
7	2015-09-18T14:42:26.000Z	40	192	0.00E+00	192	0	192
8	2015-09-18T14:42:27.000Z	40	192	0.00E+00	192	0	192
9	2015-09-18T14:42:28.000Z	40	192	5.00E+01	192	0	192
10	2015-09-18T14:42:29.000Z	40	192	5.00E+01	192	0	192



Note: The first row of the header contains the tags names and tags data types

See "[System actions](#)" on page 88 for details.

## Trend widgets

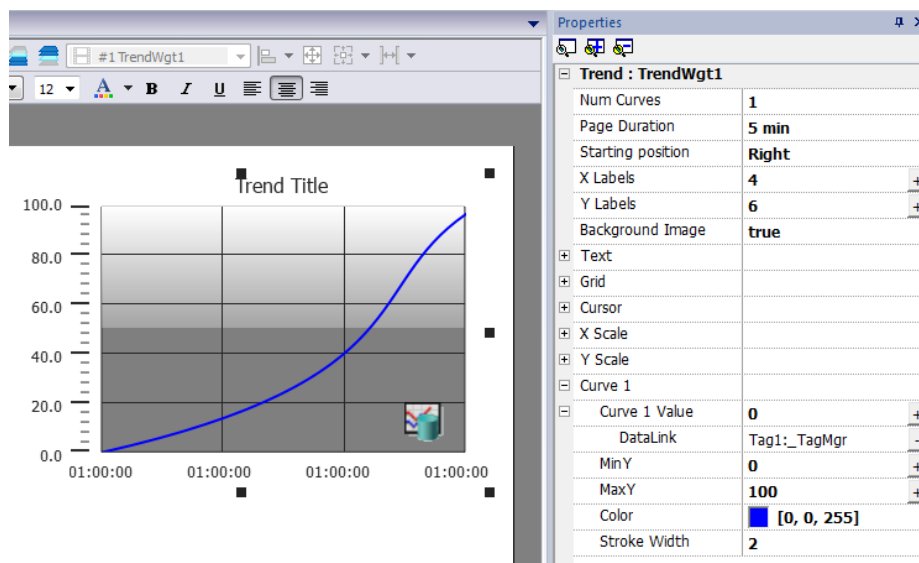
Data logged by the HMI device can be displayed in graphical format using trend widgets.

### RealTime trend widget

The real-time trend widget can be used to display the changes of value of a tag. Data is not stored in a trend buffer and cannot be retrieved for later analysis.


To display a real-time trend:

1. Drag and drop the **RealTime Trend** widget from the widget gallery to the page.



2. Attach the tag that you want to sample to the **Curve n Value**. Data is always plotted against time.

## RealTime trend widget properties

Property	Description
<b>Num Curves</b>	Number of trend curves to be displayed (Max. 5)
<b>Page Duration</b>	Time range of the x-axis.  Tip: You can attach a <b>Date Time</b> combo widget to the <b>Page Duration</b> property and dynamically change page duration at run time.
<b>Starting Position</b>	Specifies the starting point of the curve when the page is opened.
<b>X Labels</b>	Number of ticks on the x-axis scale
<b>Y Labels</b>	Number of ticks in the y-axis scale.
<b>Text</b>	Trend title and font properties (font size, label, etc.)
<b>Grid</b>	Properties of grid presentation (colors)
<b>Cursor</b>	Properties of cursor presentation (enable and color)
<b>X Scale</b>	Properties of X Scale presentation
<b>Y Scale</b>	Properties of Y Scale presentation
<b>Curve "n"</b>	Tag that will be plotted in the trend widget. See " <a href="#">Trend widget properties</a> " on page 137 for details. You can set the minimum and maximum of the curves ( <b>MinY</b> , <b>MaxY</b> ). You can attach a tag to minimum and maximum properties. This enhances the ability to change the minimum and maximum values dynamically at run time.

## Scaling data

Tag values can be scaled using the X Forms in the **Attach to** dialog. See [""Attach to" parameters" on page 28](#) for details.

## History trends

Trend data stored in trend buffers can be analyzed using the **History Trend** widget.

This is a two-step process:

- first you create a trend buffer to collect data for specified tags at specific points in time,
- then you configure a History Trend widget to display the collected data in a graphical format.

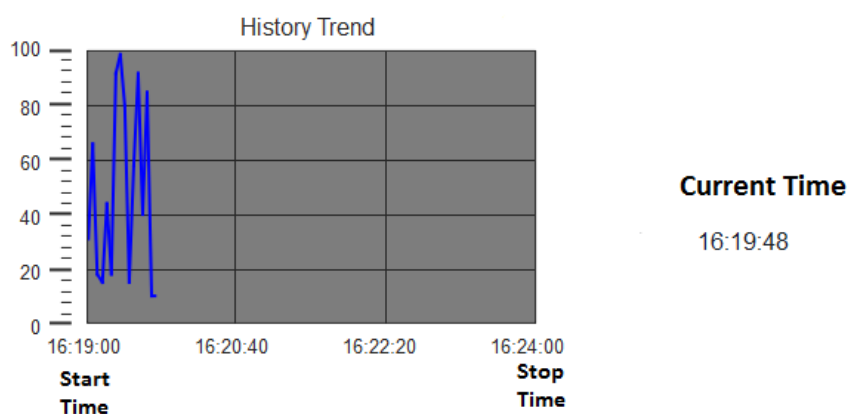
See ["Data logging" on page 132](#) for details on how to create a trend buffer.

### History Trend widget

History Trend widget displays in graphical format the content of a trend buffer.

Start time is the current time and stop time will be the current time plus the duration of the window. The curve starts from the left and progresses to the right, data is automatically refreshed during a certain interval time, until the stop time.

When the curve reaches the stop time, the curve will scroll left and the update of the curve will continue until it again reaches the stop time. At that moment a new scroll is automatically performed and the process repeats.

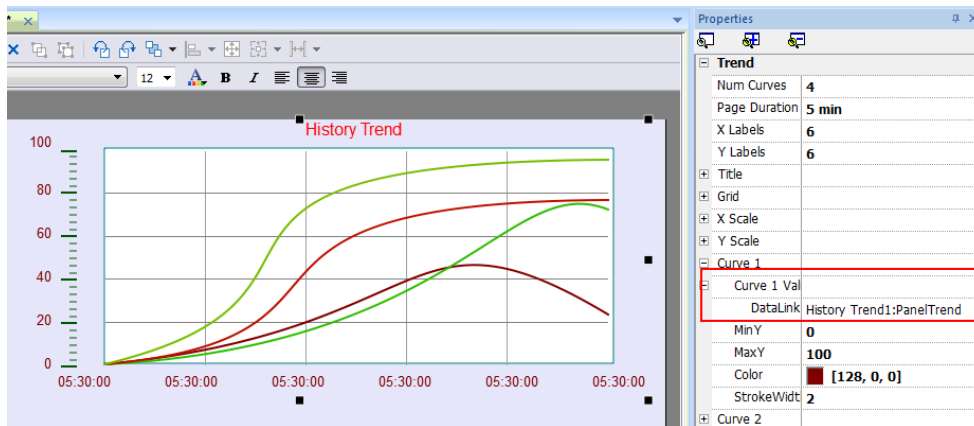


History trends require a proper configuration of trend buffer.

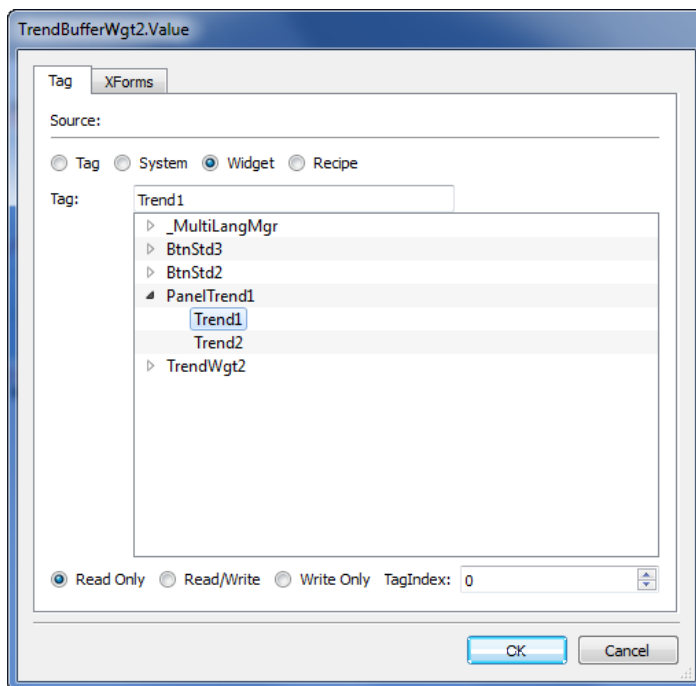
See ["Data logging" on page 132](#) for details on how to work in the Trend editor.

### Configuring the History Trend widget

1. From the **Trends/Graphs** section of the **Widget Gallery**, drag and drop the **History Trend** widget to the page.



2. In the **Properties** pane, attach the trend buffer to be plotted in the widget.



## Trend widget properties

Some Trend widget properties are only available when the Properties pane is in Advanced view.

### Request Samples

**Request Sample** property can be set for each curve and indicates the maximum numbers of samples read by the widget at one time from the trend buffer.

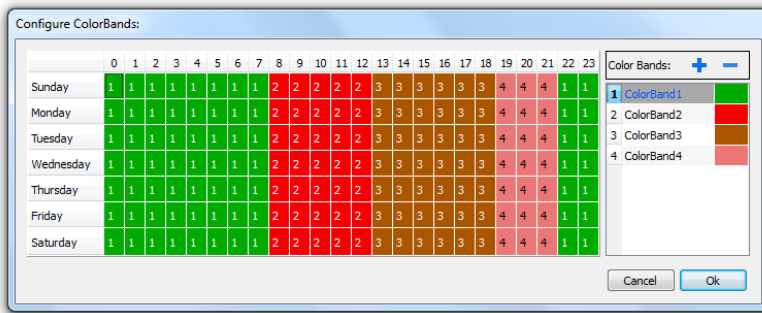


Tip: You normally do not need to modify the default value. Adjust it to fine tune performances in the trend widget refresh, especially when working with remote clients.

## Color bands

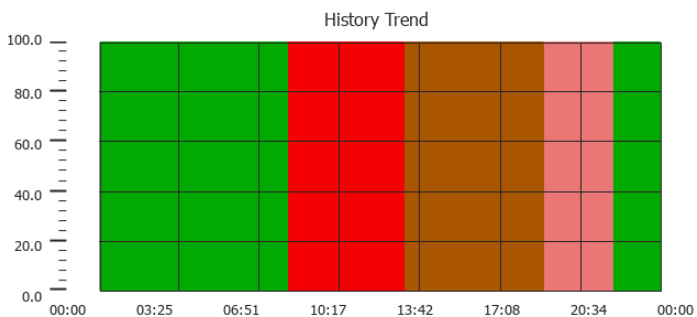
Use the color bands configuration to customize your graphs background, for example to make certain days or hours stand out (weekends, night hours, etc.).

1. In the **Properties** pane, in **Color Bands** property click **+**: the **Configure Bands** window appears.
2. Click **+** to add as many colors you need.
3. Select multiple cells and click on a color band to assign the color to the selected range of cells.



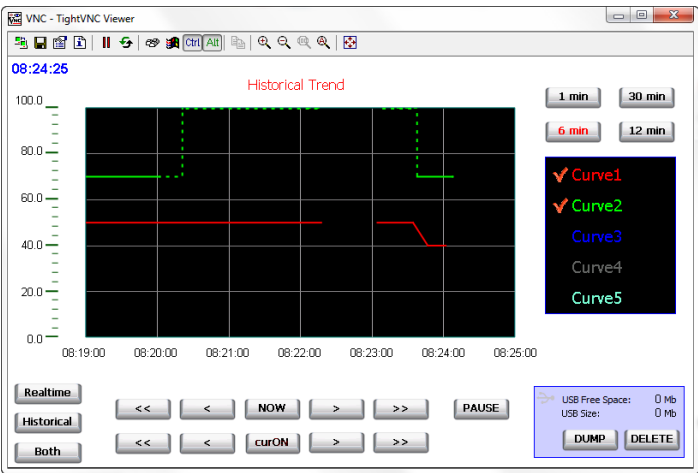
Note: This feature only uses local time in the trend widget, not the global time option.

### Calendar color bands example



## Values outside range or invalid

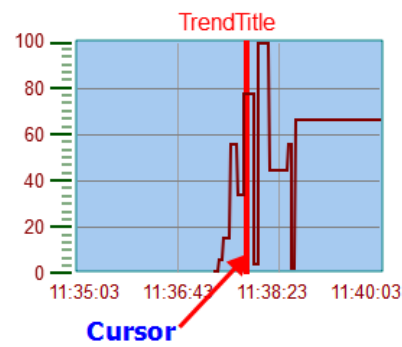
When trend value goes beyond the limits set for the trend widget, a dotted line is displayed. When the value of the tag is not available, for example the controller device is offline, no curve is drawn.



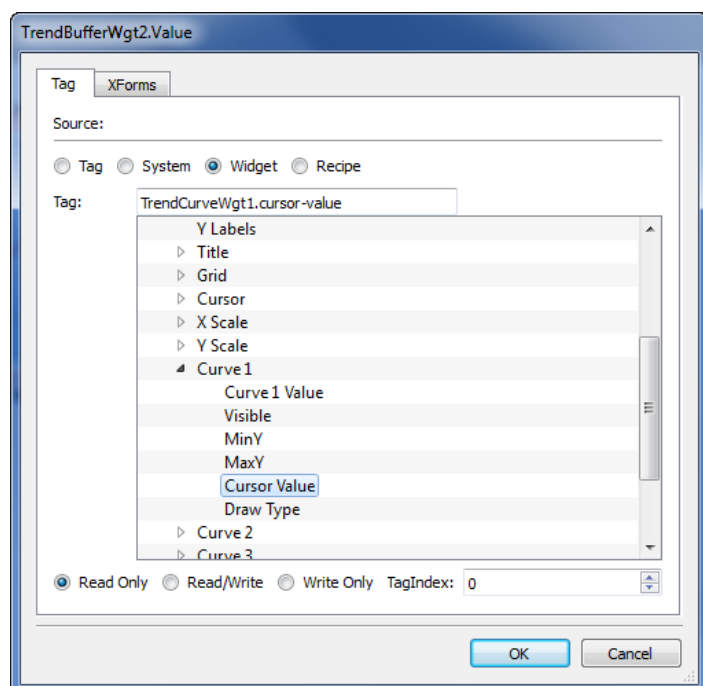
# Showing trend values

Trend cursor displays the trend value at a specific point.

Use the actions **ShowTrendCursor** and **ScrollTrendCursor** to enable the trend cursor and move it to the required point to get the value of the curve at that particular point in time.

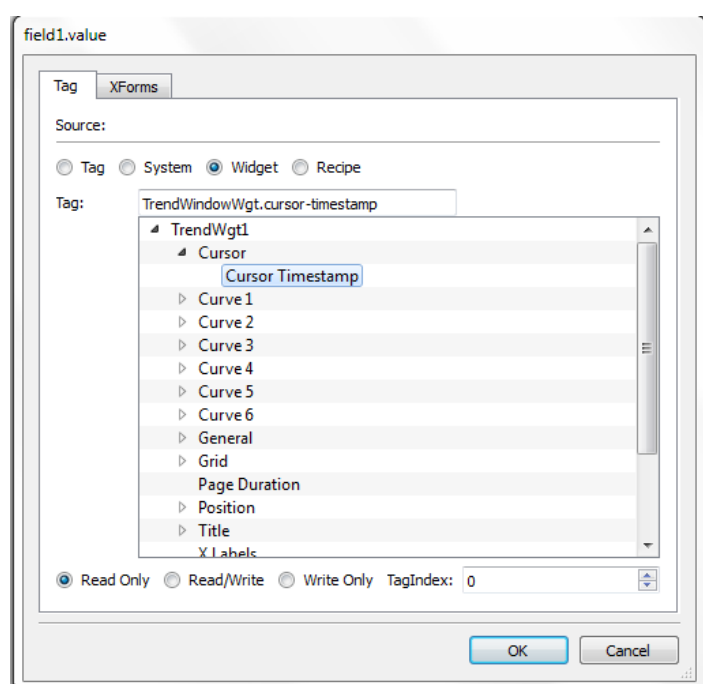


To display the value of the trend cursor on the page, define a numeric field and attach it to the **Cursor Value** widget tag.



In this example the Y axis value of the cursor is displayed.

To display the trend timestamp at the position of the cursor, define a numeric field and attach it to **Cursor Timestamp** widget tag.

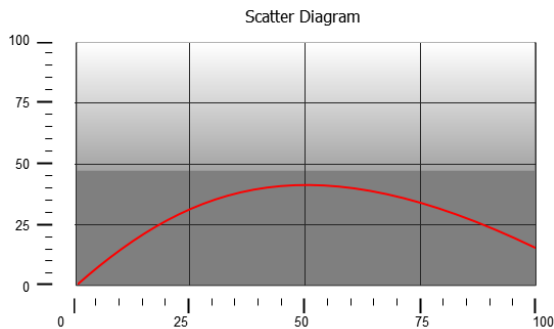


## Scatter diagram widget

A scatter diagram is a type of diagram to display values for two variables from a set of data using Cartesian coordinates. The data is displayed as a collection of points, each having the value of one variable determining the position on the hori-



zontal axis and the value of the other variable determining the position on the vertical axis. For this reason it is often called *XY graph*.



Scatter diagram curves are obtained by a linear interpolation of points. To create a new scatter diagram:

1. Add a **Scatter Diagram** widget to the page.
2. Select the number of curves to show: each curve is named as Graph1, Graph2,...
3. Customize the general graph properties such as **X Min**, **X Max**, **Grid** details.
4. Define the max number of samples/values for each curve by setting the **Max Samples** parameter.

Here you set the max number of values to be displayed in the graph starting from first element in the array.

For example: Tag1[20] and Max Samples = 10 will show just first 10 elements of the Tag1 array.

5. Define for each curve the two tags of type array to be displayed (**X-Tag** and **Y-Tag**).

When the array tags change, you can force a refresh with the **RefreshTrend** action .



Note: Scatter diagrams support only the **RefreshTrend** action.



# 15    Data transfer

---

Data transfer allows you transferring variable data from one device to another. Using this feature an HMI device can operate as a gateway between two devices, even if they do not use the same communication protocol.

---

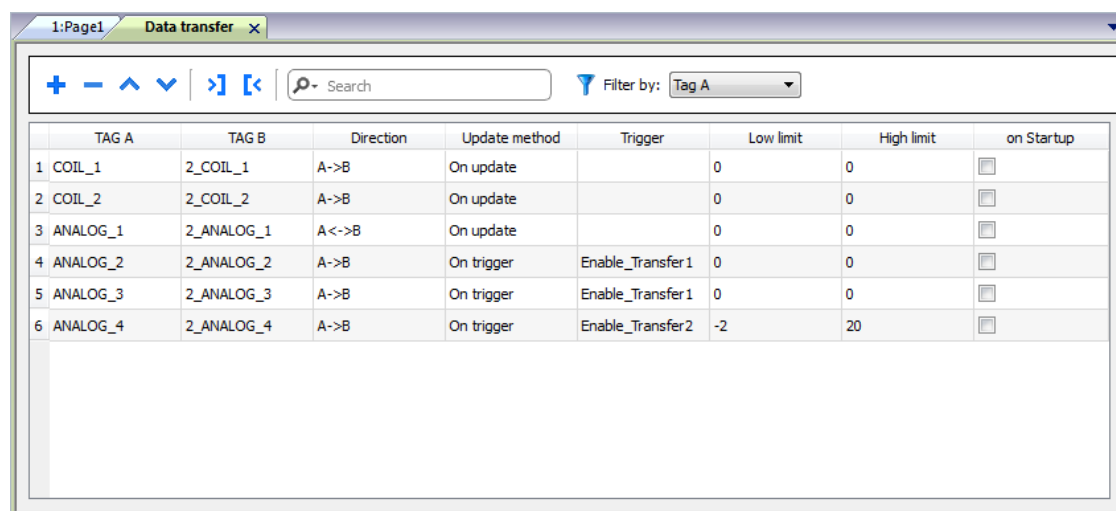
<b>Data transfer editor .....</b>	<b>144</b>
<b>Exporting data to .csv files .....</b>	<b>146</b>
<b>Data transfer limitations and suggestions .....</b>	<b>146</b>

# Data transfer editor

Path: **ProjectView** > **Config** > double-click **Data transfer**

Use the Data transfer editor to map transfer rules.

Each line in the Data transfer editor defines a mapping rule between two tags. Define more mapping rules if you need different direction, update method or trigger.



	TAG A	TAG B	Direction	Update method	Trigger	Low limit	High limit	on Startup
1	COIL_1	2_COIL_1	A->B	On update		0	0	<input type="checkbox"/>
2	COIL_2	2_COIL_2	A->B	On update		0	0	<input type="checkbox"/>
3	ANALOG_1	2_ANALOG_1	A<->B	On update		0	0	<input type="checkbox"/>
4	ANALOG_2	2_ANALOG_2	A->B	On trigger	Enable_Transfer1	0	0	<input type="checkbox"/>
5	ANALOG_3	2_ANALOG_3	A->B	On trigger	Enable_Transfer1	0	0	<input type="checkbox"/>
6	ANALOG_4	2_ANALOG_4	A->B	On trigger	Enable_Transfer2	-2	20	<input type="checkbox"/>







To add a new rule, click **+**: a new tag line is added.

## Data transfer toolbar

Parameter	Description
<b>Import/ Export</b>	Imports or exports data transfer settings from or to a .csv file.
<b>Search</b>	Displays only rows containing the search keyword.
<b>Filter by</b>	Display only rows matching filter and search field.

## Data transfer parameters

Parameter	Description
<b>TAG A/ TAG B</b>	Pair of tags to be mapped for exchanging through the HMI device.
<b>Direction</b>	Transfer direction.  <b>A-&gt;B</b> and <b>B-&gt;A</b> : Unidirectional transfers, values are always copied from one tag and sent to the other tag in the specified direction.  <b>A&lt;-&gt;B</b> : Bidirectional transfer, values are transferred to and from both tags.
<b>Update Method</b>	<b>On trigger</b> : Data transfer occurs when the value of the tag set as trigger changes above or below the values set as boundaries. Limits are recalculated on the previous tag value, the same that triggered the update.

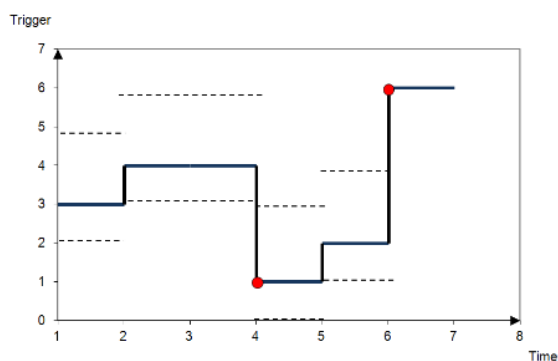
Parameter	Description
	<p> Note: This method applies only to unidirectional transfers (A-&gt;B or B-&gt;A).</p> <p><b>On Update:</b> Data transfer occurs whenever the value of the source tag changes.</p> <p> Note: This method applies both to unidirectional and to bidirectional transfers (A-&gt;B, B-&gt;A and A&lt;-&gt;B).</p> <p> Note: The Runtime cyclically monitors source tags changes (trigger tag when using On Trigger or tags to transfer when using On Update) based on Tag editor <b>Rate</b> parameter. If <b>Rate</b> setting for source Tag is 500 ms (default), the system checks for updates every 500 ms.</p> <p> Note: Changes on source tags faster than <b>Rate</b> may be not detected .</p>
<b>Trigger, High limit, Low limit</b>	<p>Tag that triggers the data transfer process. When this tag changes its value outside the boundaries set as <b>High limit</b> and <b>Low limit</b>, data transfer is started. The range of tolerance is recalculated according to the specified limits on the tag value which triggered the previous update. No action is taken if the change falls within the limits.</p> <p>This mechanism allows triggering data transfers only when significant variations of the reference values occur.</p> <p><b>Low limit</b> is less or equal to zero.</p> <p> Note: If both <b>Low limit</b> and <b>High limit</b> are set to "0", data transfer occurs whenever the value of the trigger tag changes.</p>
<b>on Startup</b>	<p>When selected, data transfer is executed on startup if the quality of the source tag is good.</p> <p>See "<a href="#">Objects</a>" on page 257 for details on quality.</p> <p> <b>Important: Data transfers executed on startup may have major impact on the HMI device boot time. Enable this option only when necessary.</b></p>

### Example of limit setting

**High limit** = 1,9

**Low limit** = - 0,9

• = points where the data transfer is triggered




## Exporting data to .csv files

Configuration information for data transfers can be exported to a .csv file.

### Example of data transfer settings in .csv file

A	B	C	D	E	F	G	H	I	J
COIL_1	2_COIL_1	A->B	On update		0	0	data1	true	1
COIL_2	2_COIL_2	A->B	On update		0	0	data2	true	1
ANALOG_1	2_ANALOG_1	A<->B	On update		0	0	data3	true	1
ANALOG_2	2_ANALOG_2	A->B	On trigger	Enable_Transfer1	0	0	data4	true	1
ANALOG_3	2_ANALOG_3	B->A	On trigger	Enable_Transfer1	0	0	data5	true	1
ANALOG_4	2_ANALOG_4	A->B	On trigger	Enable_Transfer2	-10	20	data6	true	1

Column	Description
<b>A to G</b>	Same data as in the Data transfer editor
<b>H</b>	Unique identifier automatically associated to each line.   <b>Important: When you edit the .csv file and you add any extra line, make sure you enter a unique identifier in this column.</b>
<b>I and J</b>	Reserved for future use.

## Data transfer limitations and suggestions

Correct definition of data transfer rules is critical for the good performance of the HMI devices. To guarantee reliability of operation and performance, keep in mind the following rules.

### On trigger method

The **On trigger** method allows only unidirectional transfers, (A->B or B->A)

Data transfer based on the **On Trigger** mode should be preferred since it allows you to force the transfer and monitors only the trigger tags and not all the tags involved in the transfer.

## On update method

The **On update** method allows changing the values in accordance with the direction settings only when the source value changes.

Using the **On Update** method you force the system to continuously read all the defined source tags to check if there are changes that need to be transferred. The default value of the update rate of each tag is 500 ms and can be modified with Tag editor.

## Performance observations

Data transfer performance depends on:

- number of data transfers defined,
- number of data transfers eventually occurring at the same time,
- frequency of the changes of the PLC variables that are monitored,



**Important: Always test performance of operation during project development.**



**Important: If inappropriately set, data transfer tasks can lead to conditions where the tags involved create loops. Identify and avoid such conditions.**



Tip: Use the scheduler to calibrate the update rate based on the performance of your entire project.



Tip: Use array type tags to optimize data transfer and reduce workload.



Tip: Reduce the number of data transfers to reduce page change time and boot time.






# 16 Offline node management

---

When one of the controllers communicating with the HMI device goes offline, communication performance of the system may eventually decrease.

The offline node management feature recognizes offline controllers and removes them from communication until they come back online.

Additionally, if you know that any of the controllers included in the installation is going to be offline for a certain time, you can manually disable it to maximize system performance.



Note: This feature is not supported by all communication protocols. Check protocol documentation to know if it is supported or not.

---

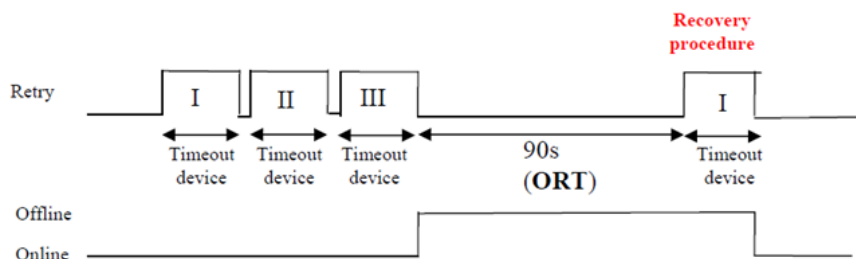
<b>Offline node management process .....</b>	<b>150</b>
<b>Manual offline node management process .....</b>	<b>150</b>
<b>Manual offline configuration .....</b>	<b>150</b>
<b>Automatic offline node detection .....</b>	<b>151</b>

# Offline node management process

Steps of the process are:

- The system communicates normally with a certain device. When the device is not responding to a communication request, the system will repeat the request twice before declaring the device offline.
- When a device is offline, the system sends communication requests to the device with a longer interval, called Off-line Retry Timeout. If the device answers to one of these requests, the system declares it online and restarts normal communication.

The diagram shows the three communication attempts and the recovery procedure that starts when the Offline Retry Timeout is elapsed.



## Manual offline node management process

Offline node management can be done manually. When a specific device is online and it is communicating normally you can:

- use an action to declare the device offline: the system stops communication with the device.
- use an action to declare the device online: the system restarts normal communication with the device.

## Manual offline configuration

When you know that some devices in communication with the HMI device are going to remain offline for a certain period of time, you can exclude them from communication using the **EnableNode** action.



**WARNING:** All disabled device nodes will remain disabled if the same project is downloaded on the device, on the other hand, if a different project is downloaded, all disabled devices will be re-enabled. The same happens with a package update.



**Tip:** To make this feature more dynamic, you may decide not to indicate a specific **NodeID** but attach it to the value of a tag or to an internal variable created to identify different devices that might be installed in your network.

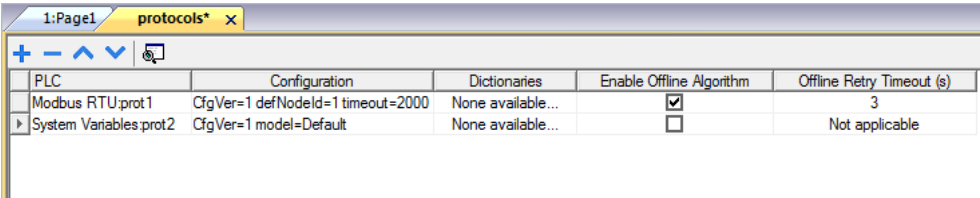


**Note:** When using the action **EnableNode** to force a device node back online, communication will start immediately.


# Automatic offline node detection

When a device is not answering to communication requests, it is de-activated. The HMI device stops sending requests to this device. After three seconds, the HMI device sends a single command to check if device is available, if so the communication is restarted, otherwise it is disabled for another timeout interval.

Default settings can be modified in Protocol editor.



PLC	Configuration	Dictionaries	Enable Offline Algorithm	Offline Retry Timeout (s)
Modbus RTU:prot1	CfgVer=1 defNodeId=1 timeout=2000	None available...	<input checked="" type="checkbox"/>	3
System Variables:prot2	CfgVer=1 model=Default	None available...	<input type="checkbox"/>	Not applicable

 Note: Not all protocols support this feature.

Parameter	Description
Enable Offline Algorithm	Enables offline management for the protocol
Offline Retry Timeout	Interval in seconds for the retry cycle after a device has been deactivated. Range: 1–86.400 seconds (24h).



# 17 Multi-language

Multi-language feature has been designed for creating HMI applications that include texts in more than one language at the same time

Multi-language feature uses code pages support to handle the different languages. A code page (or a script file) is a collection of letter shapes used inside each language.

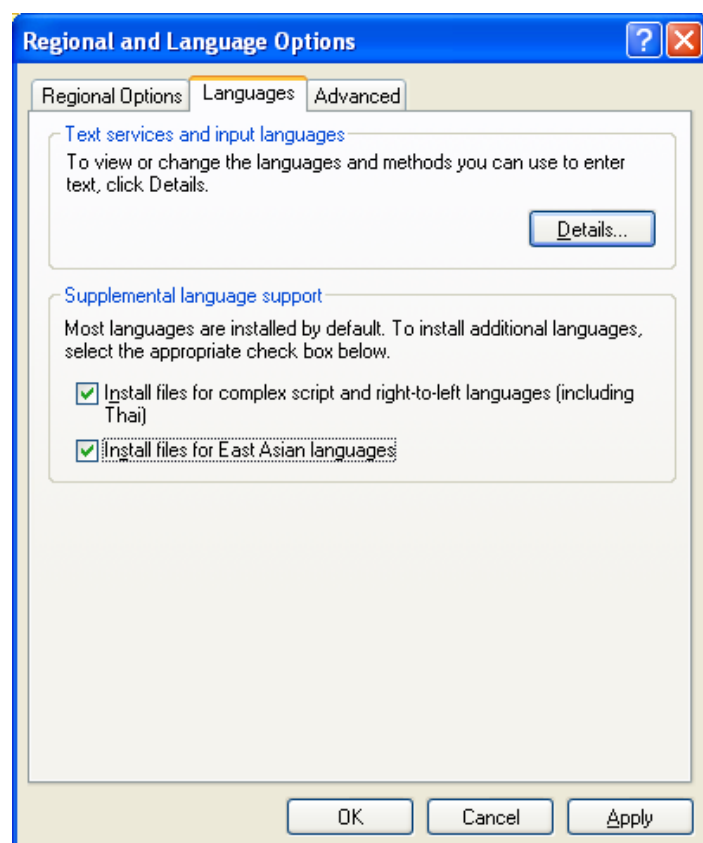
Multi-language feature can be used to define languages and character sets in a project. PB610-B Panel Builder 600 also extends the TrueType Fonts provided by Windows systems to provide different font faces associated with different character sets.

PB610-B Panel Builder 600 also allows you to provide strings for each of the languages supported.

PB610-B Panel Builder 600 also allows you to change the display language so that you can see the page look and feel during the design phase.



**Important: In Windows XP operating systems you have to install the support for complex script and East Asian languages.**



## Supported fonts for Simplified Chinese

For Simplified Chinese, the following fonts are supported:

---

Font name	Font file
Fangsong	simfang.ttf
Arial Unicode MS	ARIALUNI.TTF
Kaiti	simkai.ttf
Microsoft Yahei	msyh.ttf
NSImSun	simsun.ttc
SimHei	simhei.ttf
Simsun	simsun.ttc

## Supported fonts for Traditional Chinese

For Traditional Chinese, the following fonts are supported:

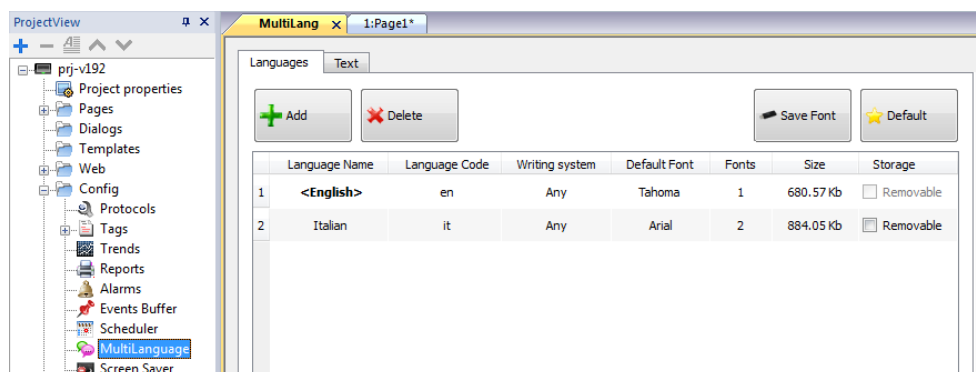
Font name	Font file
DFKai-SB	kaiu.ttf
Microsoft Sheng Hai	msjh.ttf
Arial Unicode MS	ARIALUNI.TTF
MingLiU	mingliu.ttc
PMingLiU	mingliu.ttc
MingLiU_HKSCS	mingliu.ttc

---



The Multi-language editor .....	155
Changing language .....	156
Multi-language widgets .....	156
Exporting/importing multi-language strings .....	158
Changing language at run time .....	160
Limitations in Unicode support .....	160

# The Multi-language editor

Path: **ProjectView**> **Config** > double-click **MultiLanguage**



## Language settings

Parameter	Description
<b>Language Name</b>	Name identifying the language in the project.
<b>Language Code</b>	ISO 639 language code identifier, used to match language items when importing resources from external xml files.
<b>Writing system</b>	Select the set of fonts to be used with the language
<b>Default Font</b>	Default font for project's widgets.  Note: When you choose a new font you are prompted to replace the font used in the widgets you already created.
<b>Fonts</b>	Number of fonts associated with the selected language.
<b>Size</b>	Memory used to store font files.
<b>Storage</b>	Location of file fonts is a removable external memory.  Tip: Store large font files on removable memory to free memory requirements in the HMI device.

## Adding a language

1. In the **Languages** tab, click **+**: a line is added to the table.
2. Enter all language settings.
3. Click **Default** to set the selected language as the default language when the Runtime starts.
4. Click **Save Font** to copy the fonts you marked as **Removable** on an external memory.

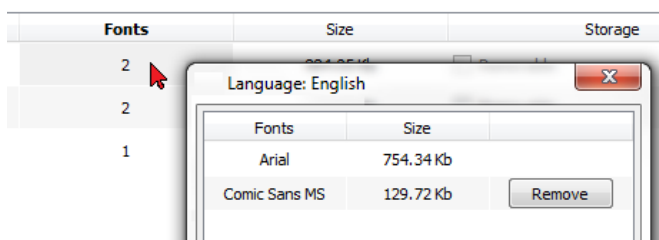


**Important:** Font files configured to be stored on removable memory must be provided to the final user to complete font installation on the HMI device.

## Removing fonts

To remove fonts no longer needed:

1. Click on the font number in the Multi-language editor: a dialog with the list of the used fonts is displayed.



2. Select the fonts to be removed and click **Remove**: removed fonts are replaced with the default font.

## Changing language

### Changing language during page design

A combo box is available for changing language during page design. If no texts appears, please check **Text** tab in the Multi-language editor and insert missing string.



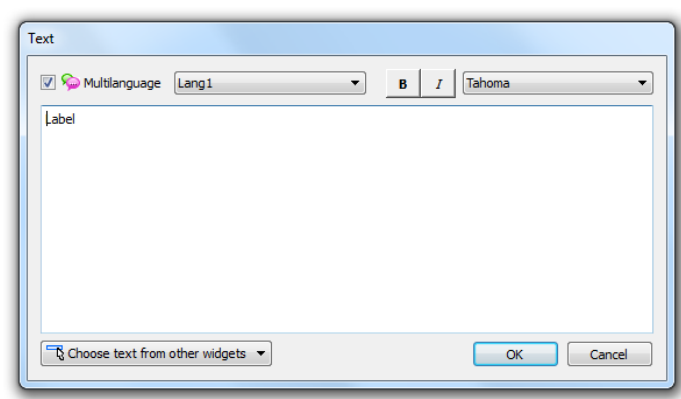
## Multi-language widgets

Multi-language support is available for objects such as buttons, static text, messages, alarm descriptions and pop-up messages.


### Multi-language for label widgets

Double-click on a text widget in a page to open the **Text** dialog.





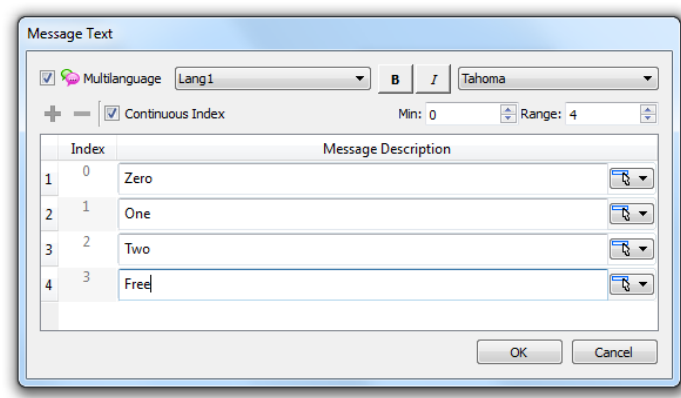
Enable/disable multi-language function, edit the text for the selected language and choose the font.

 Note: Bold, italic and color properties set here for the widget are applied to all languages .

Parameter	Description
Multilanguage	Enable/disable multi-language function for the widget.
Choose text from other widget	Click on button to browse existing message strings in project to pick text for the widget.

Multi-language for message widgets

Double-click on a message widget in a page to open the **Message Text** dialog.

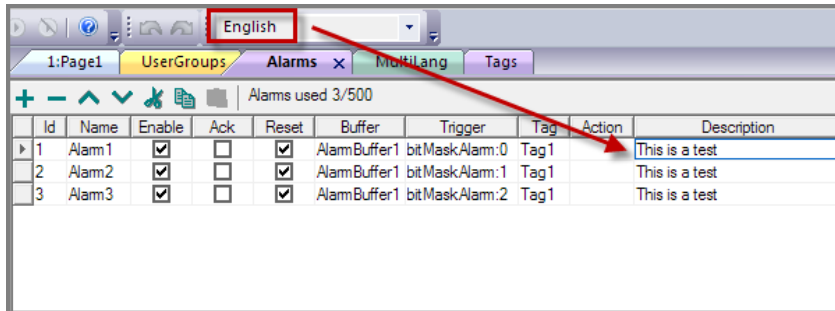


Parameter	Description
Multilanguage	Enable/disable multi-language function for the widget.
Continuous Index	Index for the widget is set of contiguous numbers (example 3, 4,5,6)
Min	Starting number for index
Range	Number of messages
Choose text from other widget	Click on button to browse existing message strings in project to pick text for the widget.

## Multi-language for alarm messages

To add a multi-language strings for alarm messages:

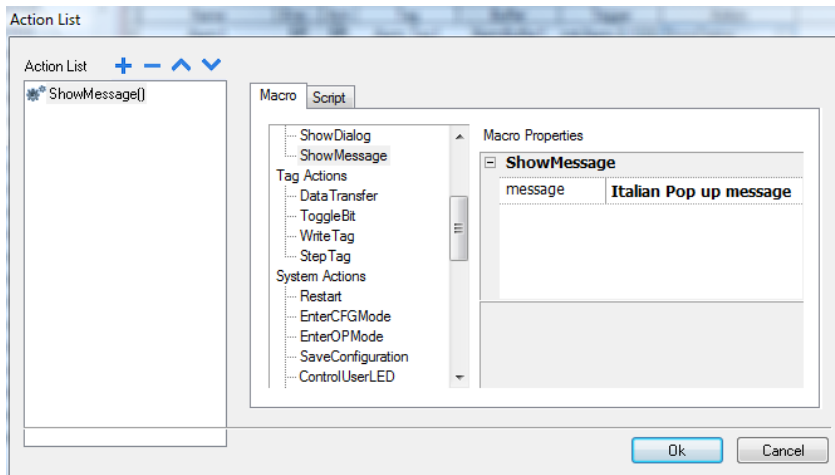
1. Open the Alarm editor.
2. Select a language using the language combo box.
3. Enter the text for the alarm in the **Description** column.



## Multi-Language for pop-up messages

To add a multi-language pop-up message:

1. Select a language from the language combo box.
2. Add the Page action **ShowMessage** and enter the text in the selected language.



## Exporting/importing multi-language strings

The easiest way to translate a project into multiple languages is to export all texts to a .csv file, translate the resulting document and then import the translated text back into the project.



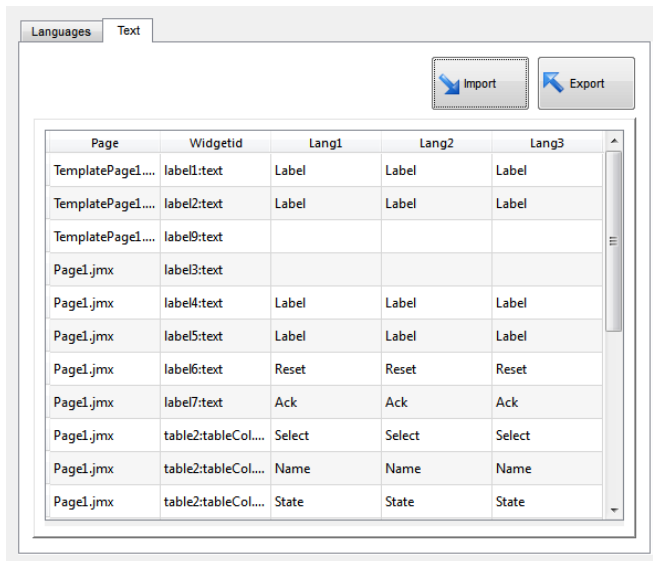
**Important:** The .csv file exported by PB610-B Panel Builder 600 is coded in Unicode, to edit it you need a specific tool supporting Unicode encoded .csv files.

## Exporting and reimporting strings

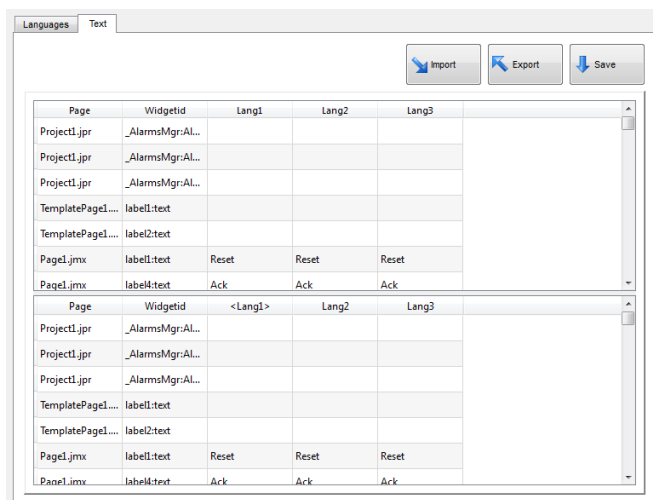
Path: **ProjectView > Config > double-click MultiLanguage**

To export and re-import multi-language strings:

1. In the **Text** tab, click **Export**: all multi-language strings are exported to a .csv file.



**Important:** Set all languages that will be used in the project before exporting the file. This will guarantee that the exported file will contain all columns and language definitions.



2. Once the strings have been translated, click **Import** to re-import them into the project: strings are imported matching the widget ID and the page number of each widget.
3. Click **Save** to save the new widget data.



**Note:** To change the separator used in the exported file, change the regional settings of your computer. When importing, the separator information is retrieved from the file; if not found, the default character "," is used.

## Import constraints

The following formats are supported for import:

- Comma Separated Values (.csv)
- Unicode Text (.txt)



Note: Use the Unicode Text file format when you import a file modified using Microsoft® Excel®.

## Changing language at run time

### Changing language with an action

After the project download, the HMI Runtime will start using the language set as default. You can change the language using the **SetLanguage** action. See "[MultiLanguage actions](#)" on page 77.



Note: Once the language has been changed, it will be used also in future sessions.

### Missing fonts

When you change language, if the required fonts are not available in the device memory, a pop-up message prompts you to insert the memory card containing the missing fonts. At the end of the operation you can remove the memory card.



## Limitations in Unicode support

PB610-B Panel Builder 600 has been designed for working with Unicode text. However, for compatibility issues with some platforms, Unicode is supported only in a subset of properties.

Area	Property	Charset Accepted	Reserved Chars/Strings
Protocol editor	Alias	ASCII [32..126]	(space) , ; : . < * > ' "
Tag editor	Name	ASCII [32..126]	. \ / * ? : > <   " & # % ; =
	Group	ASCII [32..126]	<New> \ / * ? : > <   " & # % ;
	Comment	Unicode	
Trends	Name	ASCII [32..126]	\ / * ? : > <   " & # % ;
Printing Reports	Name	ASCII [32..126]	\ / * ? : > <   " & # % ;

Area	Property	Charset Accepted	Reserved Chars/Strings
<b>Alarms</b>	Name	ASCII [36..126]	\ / * ? : > <   " & # % ;
	Description	Unicode	[] - for live tags, \ escape seq for [ and \
<b>Events</b>	Buffer Name	ASCII [32..126]	\ / * ? : > <   " & # % ;
<b>Scheduler</b>	Name	ASCII [32..126]	\ / * ? : > <   " & # % ;
<b>Languages</b>	Language Name	ASCII [32..126]	\ / * ? : > <   " & # % ;
	Texts in widgets	Unicode	-
	Texts from import files	Unicode	-
<b>User Group</b>	Group Name	a-z A-Z _	admin,guest,unauthorized
	Comments	Unicode	-
<b>User</b>	Name	ASCII [32..126]	\ / * ? : > <   " & # % ;
	Password	Unicode	-
	Comment	Unicode	-
<b>Recipes</b>	Name	ASCII [32..126]	\ / * ? : > <   " & # % ; ! \$ ' ( ) + , = @ [ ] { } ~ `
	Set Name	ASCII [32..126]	\ / * ? : > <   " & # % ; ! \$ ' ( ) + , = @ [ ] { } ~ `
	Element name	ASCII [32..126]	\ / * ? : > <   " & # % ; ! \$ ' ( ) + , = @ [ ] { } ~ `
<b>General</b>	Project Name	A-Z,a-z,0-9,-,_,	"PUBLIC", "readme", "index.html"
	Page Name	A-Z,a-z,0-9,-,_,	-
	Dialog Page Name	A-Z,a-z,0-9,-,_,	-
	Template Page Name	A-Z,a-z,0-9,-,_,	-
	Keypad Name	A-Z,a-z,0-9,-,_,	-
	Files (Images/Video/etc..)	A-Z,a-z,0-9,-,_,	-
	Widgets ID	A-Z,a-z,0-9,-,_,	-
<b>Runtime</b>	PLC Communication	UTF-8, Latin1, UCS-2BE, UCS-2LE, UTF-16BE, UTF-16LE	-



# 18 Scheduler

---

PB610-B Panel Builder 600 provides a scheduler engine that can execute specific actions at set intervals, or on a time basis.

Creating a schedule is typically a two-step process:

- 1. You create a schedule with a list of actions to be executed when the scheduled event occurs. You do this in the Scheduler editor
- 2. You create a run-time user interface that allows the end-user to change settings for each schedule. You do this adding a **Scheduler** widget to a page of your project and configuring it to fit user scheduling needs.

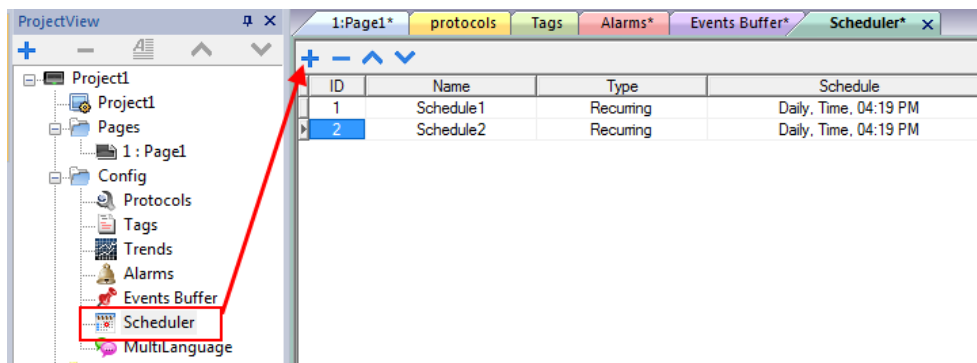
---

Creating a schedule .....	164
HighResolution schedule .....	164
Recurring schedule .....	164
Configuring location for schedules .....	166
Configuring the Scheduler widget .....	167
Scheduling events at run time .....	168

# Creating a schedule

Path: **ProjectView**> **Config**> double-click **Scheduler**

- Click **+** to add a schedule.



## Schedule parameters

Parameter	Description
<b>ID</b>	Unique code assigned automatically to the schedule
<b>Name</b>	Name of schedule
<b>Type</b>	Type of schedule: <ul style="list-style-type: none"> <li><b>Recurring</b>, see "<a href="#">Recurring schedule</a>" below for details.</li> <li><b>HighResolution</b>, see "<a href="#">HighResolution schedule</a>" below for details</li> </ul>
<b>Schedule</b>	Scheduler settings and options. See " <a href="#">Recurring schedule</a> " below for details.
<b>Action</b>	Actions to be executed at the scheduled time
<b>Priority</b>	Priority level for the event. If two schedules occur at the same time, the event with the higher priority will be executed first.

## HighResolution schedule

The **HighResolution** schedule is used to perform actions that need to be repeated at specified intervals. The interval between executions is set in milliseconds in the **Schedule** column.





Note: You cannot change at run time the settings of this type of schedule. If you need to change the action time settings at run time, choose **Recurring** schedule and set **Type** to **Every**. See "[Recurring schedule](#)" below for details.

## Recurring schedule

The Recurring schedule is used to perform actions at specified points in time. Settings can be modified at run time.



## Recurring scheduler parameters

Parameter	Description
<b>Type</b>	Frequency of the scheduled actions
<b>Mode</b>	Specific settings required by each scheduler type
<b>Condition</b>	<p>Boolean tag (true/false) to activate the specified actions at the moment the timer is triggered. Actions will be executed if tag = true. By default, actions are executed when the timer is triggered.</p> <p> Note: Only tags attached to the Boolean data type are shown.</p>
<b>Actions</b>	<p>Actions to be executed by the schedule.</p> <p> <b>Important: Actions and schedule parameters cannot be modified at run time</b></p>
<b>Date</b>	Date when the scheduled actions will be executed
<b>Time/Offset</b>	<p>This field display one of the following:</p> <p><b>Time</b> = when the scheduled actions will be executed</p> <p><b>Offset</b>= delay or advance with respect to the selected mode.</p>
<b>Location</b>	Reference location to calculate sunset/sunrise time.
<b>weekdays</b>	Days of the week in which the scheduled actions will be executed.
<b>On startup</b>	Executes schedule at start up
<b>Enable schedule</b>	Enables/disables the schedule
<b>Execute only at start-up</b>	Executes the schedule only once at start up

## Schedule type options

Option	Description
<b>By Date</b>	Actions are executed on the specified date and time.
<b>Daily</b>	Actions are executed daily at the specified time.
<b>Every</b>	Actions are executed with the specified interval (Range: 1 s–1 day)
<b>Hourly</b>	Actions are executed every hour at the specified minute.
<b>Monthly</b>	Actions are executed every month at the specified date and time.

Option	Description
<b>Weekly</b>	Actions are executed every week on the specified weekday(s) and time.
<b>Yearly</b>	Actions are executed every year at the specified date and time.

## Schedule mode options

Option	Description
<b>Time</b>	Depends on the schedule type. Allows you to specify date/time/week data.
<b>Random10</b>	Actions are executed in the time interval of 10 minutes before or after the set time. For example, if set time is 10:30, actions are executed any time between 10:20 and 10:40.
<b>Random20</b>	Actions are executed in the time interval of 20 minutes before or after the set time. For example, if set time is 10:30, actions are executed any time between 10:10 and 10:50.
<b>Sunrise+</b>	Actions are executed with a specified delay after sunrise. The delay is set in minutes/hours and sunrise time is location specific.
<b>Sunrise-</b>	Actions are executed with a specified advance before sunrise. The advance is set in minutes/hours and sunrise time is location specific.
<b>Sunset+</b>	Actions are executed with a specified delay after sunset. The delay is set in minutes/hours and sunset time is location specific.
<b>Sunset-</b>	Actions are executed with a specified advance before sunset. The advance is set in minutes/hours and sunset time is location specific.

See "[Configuring location for schedules](#)" below for details on sunset and sunrise settings.



Note: **Mode** options are not available for all schedule types.

## Configuring location for schedules

Scheduled actions can be configured to be executed at a specific time with respect to sunrise and/or sunset. To do this you need to define the correct location, based on UTC information. The system will automatically calculate the sunrise and sunset time.

Only a few locations are available by default. If your location is not listed, you can add it by entering latitude, longitude and UTC information in the Target\_Location.xml file.



**Important:** Each platform has its own Target\_Location.xml file.

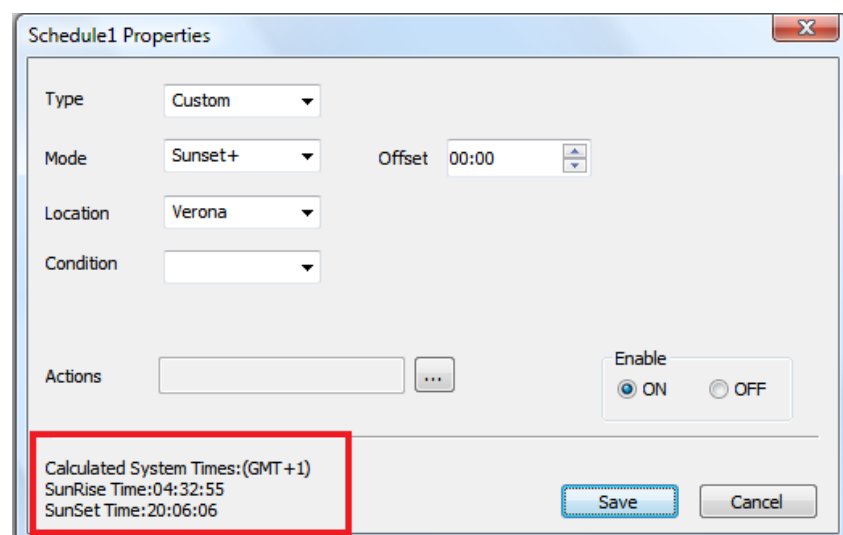
## Location files position

Application	Location file path
<b>PB610-B Panel Builder 600</b>	<i>PB610-B Panel Builder 600\languages\shared\studio\config\Target_Location.xml</i>
<b>Devices</b>	<i>PB610-B Panel Builder 600\runtime\UN20_WCE6 (MIPSIV_FP)\config\Target_Location.xml</i>
	<i>PB610-B Panel Builder 600\runtime\UN30_SDK (ARMV4I)\config\Target_Location.xml</i>
	<i>PB610-B Panel Builder 600\runtime\UN31_SDK (ARMV4I)\config\Target_Location.xml</i>
<b>Simulator</b>	<i>PB610-B Panel Builder 600\simulator\config\Target_Location.xml</i>

For example, the information for the city of Verona (IT) is shown below:

```
<file city="Verona" latitude="45.44" longitude="10.99" utc="1"/>
```

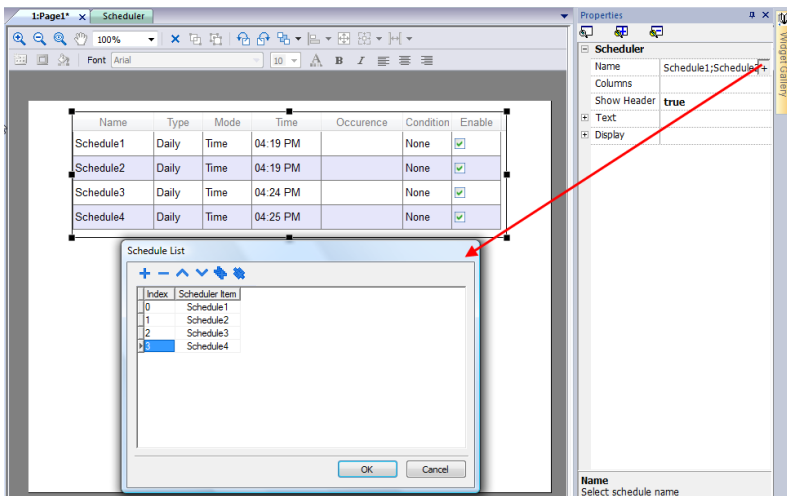
Location information is also displayed in the dialog together with sunset and sunrise times.



## Configuring the Scheduler widget

To display scheduler data on a page:

1. Drag and drop a **Scheduler** widget from the widget gallery into the page.
2. In the **Properties** pane, click + for the **Name** parameter: the **Schedule List** dialog is displayed.
3. Add all the schedules you want to display in the page.



4. In the **Properties** pane, customize all settings.

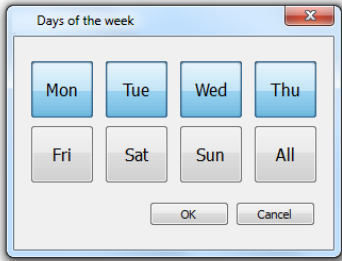
## Scheduler settings

Parameter	Description
<b>Name</b>	Schedule to be displayed
<b>Columns</b>	Columns to be displayed and their characteristics
<b>Show Header</b>	Shows/hides column headers
<b>Time Spec</b>	Time to be displayed at run time
<b>Text</b>	Font used for text
<b>Display</b>	Table styles

## Scheduling events at run time

At run time you can modify the following scheduling parameters.

Name	Type	Mode	Time	Occurence	Condition	Enable
Schedule1	By Date	Time	11:01	JUN 20,2013	None	<input checked="" type="checkbox"/>
Schedule3	Monthly	Sunrise+	11:01	Day : 3	None	<input checked="" type="checkbox"/>
Schedule4	Weekly	Rando...	16:19	M T W T F S S	None	<input checked="" type="checkbox"/>
Schedule5	Yearly	Time	01:00			
Schedule6	Custom	Time	01:16			



Parameter	Description
Occurrence	Information on the type of schedule and time of execution
Condition	Condition applied to action execution
Enable	Enabels/disables the execution of the scheduled actions without deleting the schedule.

See ["Recurring schedule" on page 164](#) for details on schedule parameters.



# 19 User management and passwords

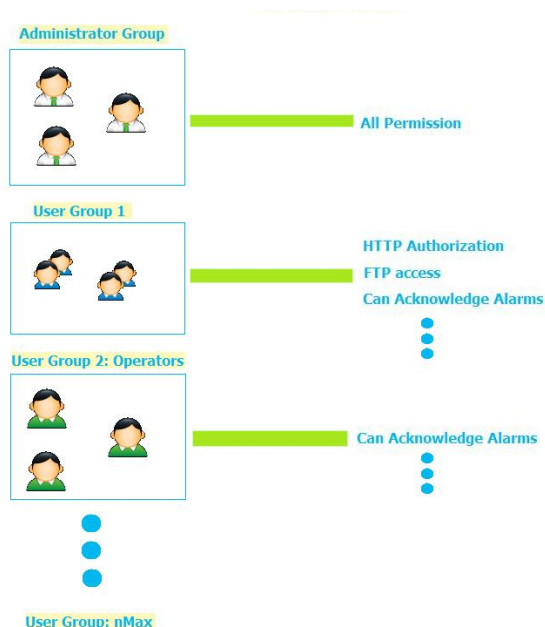
You can restrict access to various widgets and operations by configuring users, users groups and assigning specific authorizations to each group.

Each user must be member of one and only one group. Each group has specific authorizations and permissions.

Authorizations and permissions are divided in two categories:

- Widget permissions: hide, read only, full access
- Action permissions: allowed or not allowed.

By organizing permissions and groups you can define the security options of a project.

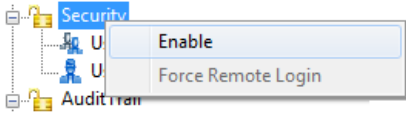


<b>Enable/disable security management .....</b>	<b>172</b>
<b>Configuring groups and authorizations .....</b>	<b>172</b>
<b>Modifying access permissions .....</b>	<b>173</b>
<b>Assigning widget permissions from page view .....</b>	<b>177</b>
<b>Configuring users .....</b>	<b>178</b>
<b>Default user .....</b>	<b>179</b>
<b>Managing users at run time .....</b>	<b>179</b>
<b>Force remote login .....</b>	<b>180</b>

# Enable/disable security management

Path: **ProjectView**> right-click **Security**> **Enable**

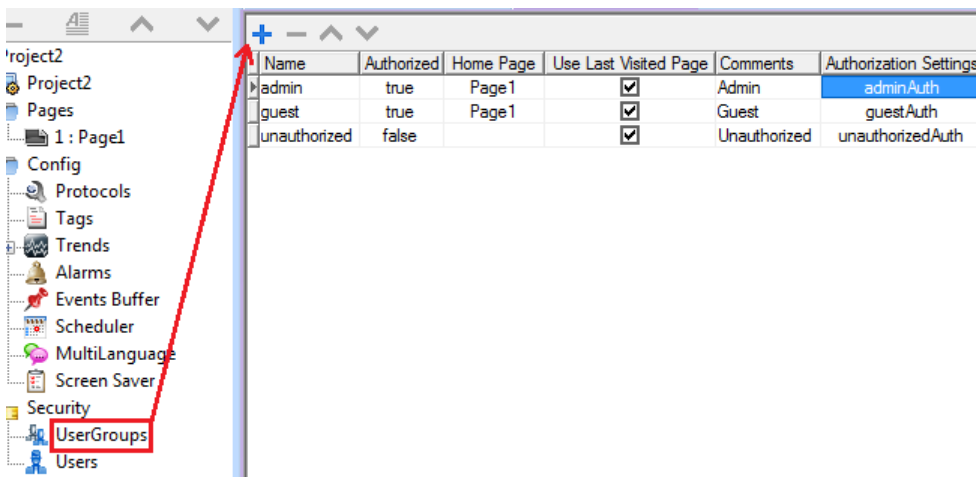
The padlock symbol indicates whether the function is enabled or disabled.



**Important:** Security settings are effective only if the security function is enabled.

## Configuring groups and authorizations

Path: **ProjectView**> **Security**> double-click **UserGroups**



Three predefined groups are available by default (**admin**, **guest** and **unauthorized**): they cannot be deleted nor renamed. You can, however, modify authorizations and other settings.

### Adding a user group

Click **+** to add user group.

Parameter	Description
<b>Name</b>	Name of users group
<b>Authorized</b>	Authorization granted
<b>Home Page</b>	Page displayed when users belonging to this group log in
<b>Use Last Visited Page</b>	When selected, the last page displayed by the previous user will be displayed when users belonging to this group log in

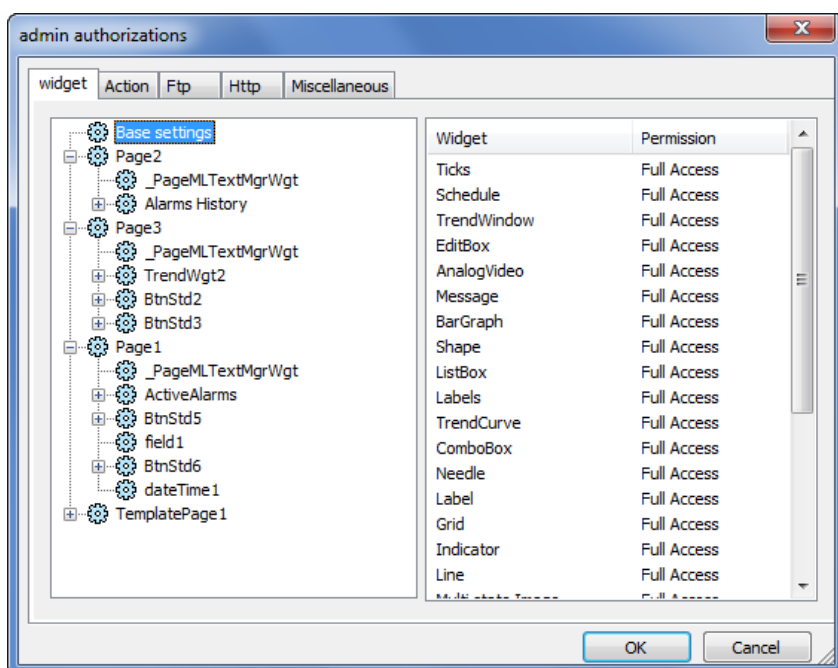


Parameter	Description
<b>Comments</b>	Any comment or description for the group
<b>Authorization Settings</b>	Opens the Admin Authorization dialog to set access permissions. See " <a href="#">Modifying access permissions</a> " below for details.

## Modifying access permissions

*Path: **ProjectView** > **Security** > double-click **UserGroups** > **Authorization Settings** column*

Click the button: a dialog appears with a list of widgets and actions. You can modify access permissions for each one in the list.



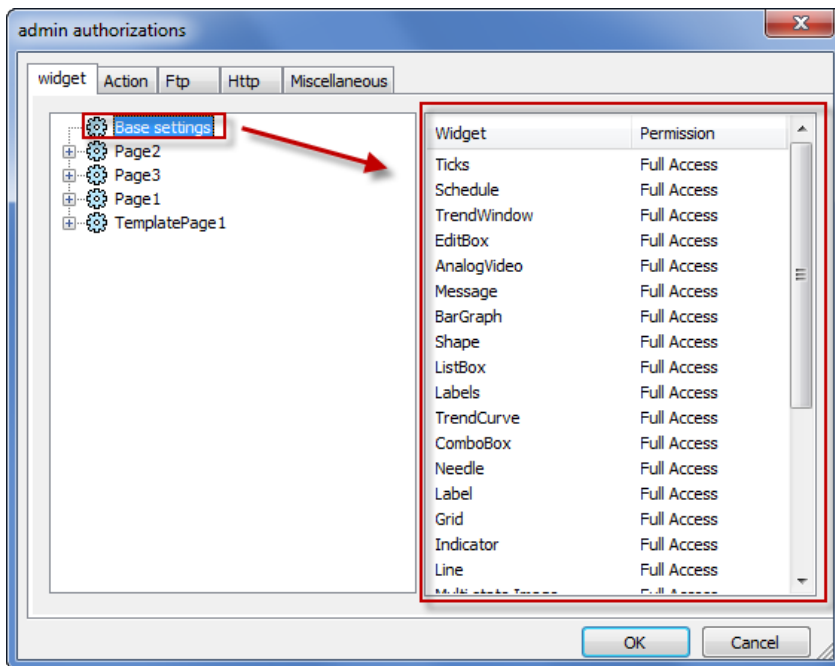
## Widget permissions

In the **Widget** tab you can define widget access options at project level, at page level or at widget level for all the widgets used in the project. Lower levels permission (for example, widget level) overrides higher levels (that is, page and project levels).

Use **Base settings** to set default permissions at project level.

Possible settings are:

- **Full Access** to enable read/write access to the widget
- **Read Only** to enable readonly access to the widget
- **Hide** to hide widget for selected group



## Changing a widget permission

To change access permission for an individual widget in a page of the project, navigate to that widget within its page on the right pane and customize its access options. Otherwise, all widgets take the permissions set at project or page level.

For example, if page permission for a widget is set at project level to **Read Only**, then all the same widgets will have permission **Read Only**. When you select a widget inside a page from the tree structure, permission is actually set to **Use Base Settings**. You can change this setting and modify access permissions only for this widget in this page.

## Access priority

Widget permissions are considered with the following priority:

Permission level	Priority
Project level - Basic settings	Low
Page level	Medium
Widget level	High

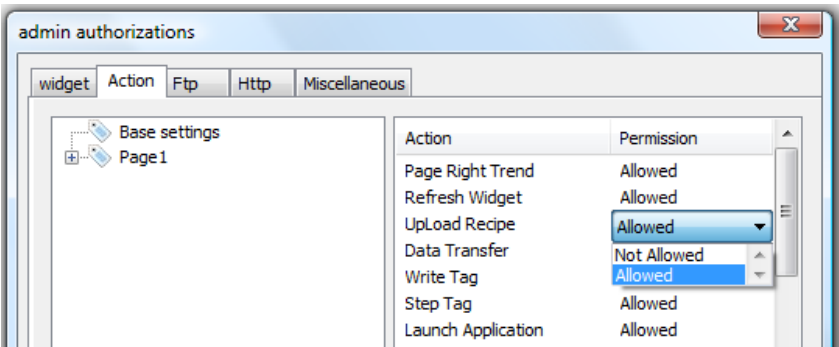
This allows you to specify exceptions for an action or a widget directly from the page view.

For example, if you set permissions for a widget at project level to Read Only and to Full Access at page level then the page level settings will prevail.

Access permissions can be modified directly from the project page. See ["Assigning widget permissions from page view" on page 177](#) for details.

## Action permissions

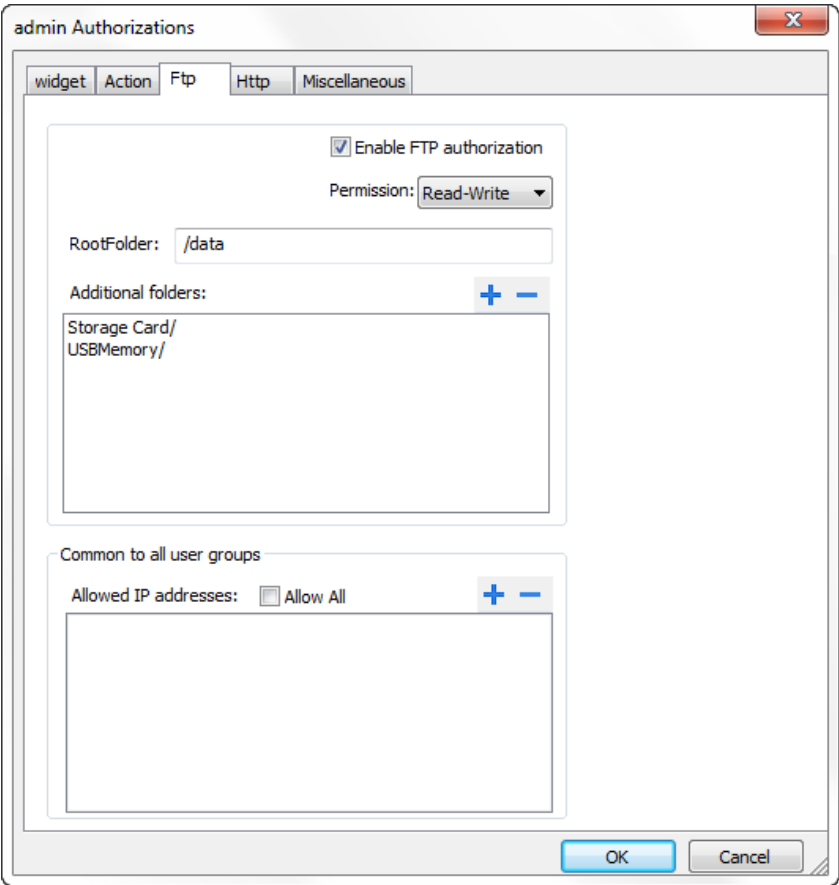
In the **Action** tab you can define action authorizations at project level, at page level or at widget level. Actions can be either **Allowed** or **Not Allowed**.




Action permissions can be modified directly from the project page. See ["Assigning widget permissions from page view"](#) on [page 177](#) for details.

### FTP authorizations

In the **Ftp** tab you can set specific authorizations for the FTP server.



Element	Description
Enable FTP authorization	Enables the FTP function for the specific group
Permission	Type of permission: <ul style="list-style-type: none"><li>• Read-Only</li></ul>

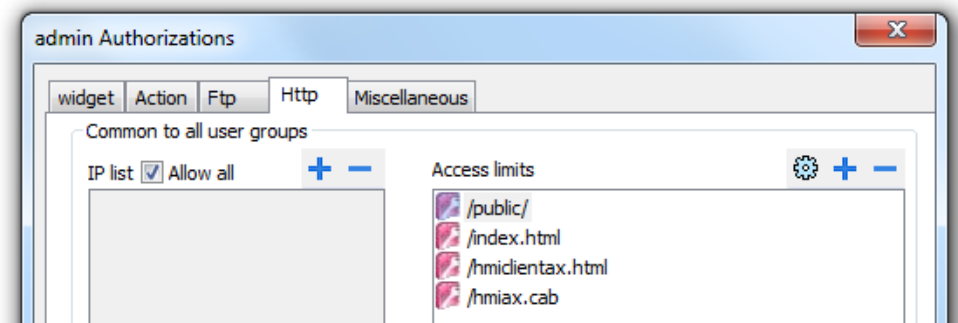
Element	Description
	<ul style="list-style-type: none"> <li>• <b>Read-Write</b></li> </ul>
<b>Root Folder</b>	Folder to be used as root for FTP access. This is a relative path.
<b>Additional folder</b>	Extra folders to be used as root for FTP access (for example, on USB drive or SD card)
<b>Allowed IP Addresses</b>	List of IP addresses from which FTP connection can be accepted.  <b>Important: This setting is common to all users groups.</b>

## HTTP authorizations

In the **HTTP** tab you set restrictions to HTTP access to the web server integrated in HMI Runtime.



**Important: This setting is common to all users groups.**



Element	Description
<b>IP list</b>	IP addresses authorized to access the HTTP server. By default all.
<b>Access limits</b>	List of resources for which access is limited

Effect of these settings depends on whether the option **Force Remote Login** has been selected. See ["Force remote login" on page 180](#) for details.

Force Remote Login	Default Access to work-space	Access limits
-	Full	-
<b>Disable</b>	Full	Can be used to block access to some files/folders or to require authorization
<b>Enable</b>	No Access	Can be used to open access to files/folders

## Adding an HTTP configuration

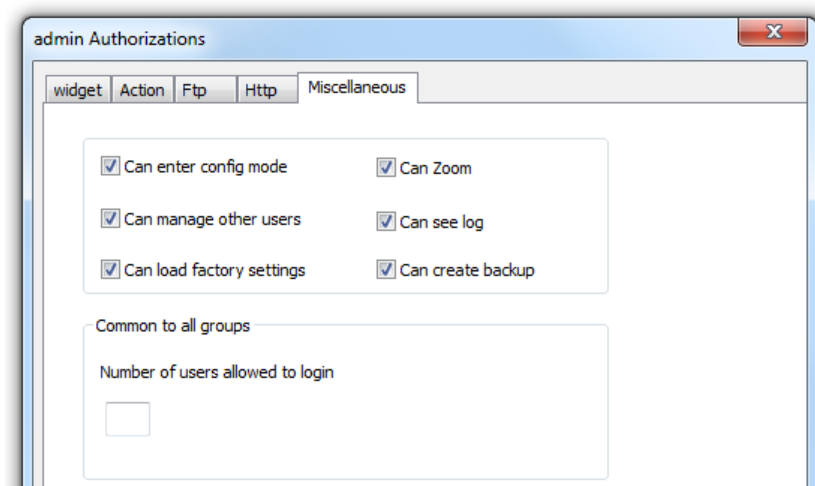
To add and configure a new access click **+**: the **Access limits** dialog is displayed.

To restore the default configuration click the **Set default access limits** icon. Default configuration allows access to the following:

- PUBLIC folder and Index.html, that contain web console and public resources

## Miscellaneous settings

In the **Miscellaneous** tab you can define various authorization settings.



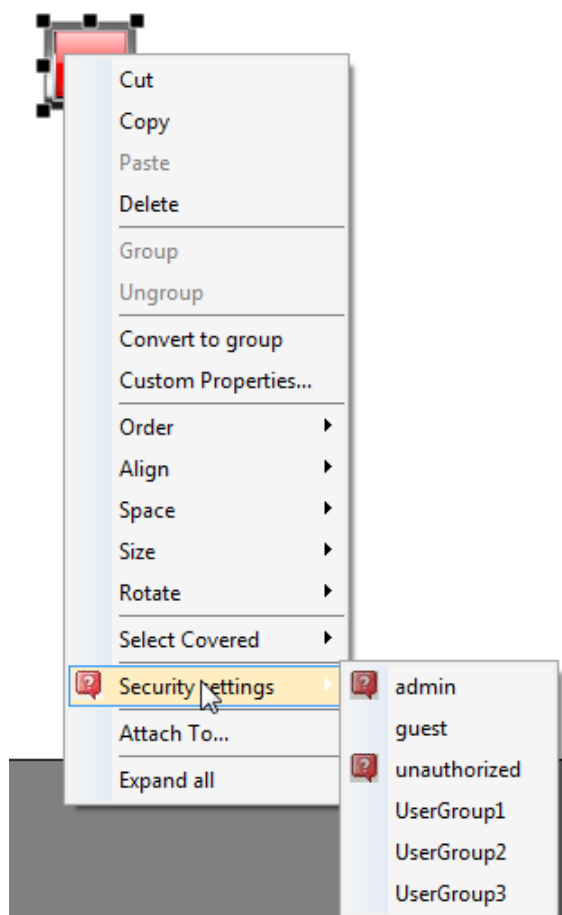
Note: Some of these settings are group specific, while other are common to all groups.

Option	Description
<b>Can enter config mode</b>	Enables switching from runtime to configuration mode. Normally used for maintenance.
<b>Can manage other users</b>	Gives superuser privileges at run time. Allows adding, deleting and modifying users' permissions.
<b>Can load factory settings</b>	Restores factory settings.
<b>Can zoom</b>	Enables zoom in/out in context menu at run time
<b>Can see log</b>	Allows user to see logs at run time
<b>Can create backup</b>	Allows user to backup project.
<b>Number of users allowed to login</b>	Maximum number of users that can be connected to the HMI Runtime at the same time. Default is 3.

## Assigning widget permissions from page view

You can assign different levels of security, to different user groups, on a single widget, directly from the project pages.

1. Right-click on the widget and select **Security settings**.
2. Choose the group: the authorization dialog for the group is displayed.
3. Set the security properties to access the widget.

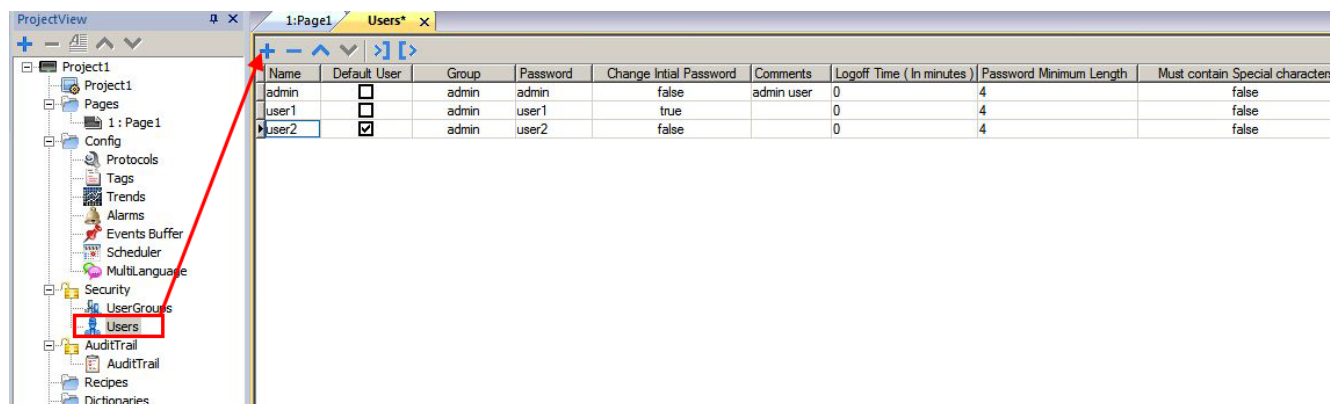


See ["Modifying access permissions" on page 173](#) for details.

## Configuring users

**Path:** *ProjectView* > *Security* > double-click *Users*

In the Users editor, click + to add a user: one row is added to the table.



Parameter	Description
<b>Name</b>	User name
<b>Default User</b>	This user is automatically logged in when the system is started or after another user has logged off. Only one Default user can be set.
<b>Group</b>	User group
<b>Password</b>	User password
<b>Change Initial Password</b>	This user is forced to change his password at first log in.
<b>Comments</b>	Further user description
<b>Logoff time</b>	Minutes of inactivity after which the user is logged off. Set to 0 to disable.
<b>Password Minimum Length</b>	Minimum length of password
<b>Must Contain Special Characters</b>	Password must contain at least one special character.
<b>Must Contain Numbers</b>	Password must contain at least one numeric digit.

## Default user

You can define only one default user in a project. This is the user automatically logged in at system start up and when the currently logged user logs out or is logged out after time-out.

To log into HMI Runtime with a different user, use one of the actions:

- **SwitchUser**
- **LogOut**

See ["User management actions" on page 98](#) for details.

## Managing users at run time

The default user, if any, is automatically logged in when the HMI Runtime is started. If no default user is configured, the system requires a user name and password. See ["User management actions" on page 98](#) for details on the actions that can be executed on users.

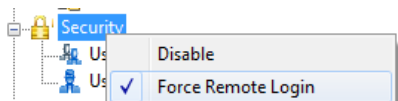
### Removing user data

All the user information modified at run time is stored in dedicated files. To remove these dynamic files and all the changes applied to user configuration at run time you can:

- on HMI Runtime: execute the action **DeleteUMDynamicFile**
- with PB610-B Panel Builder 600: select the **Delete Dynamic Files** in the download dialog.

# Force remote login

Path: **ProjectView**> right-click **Security**> **Force Remote Login**



Select this option to force user to log in when using remote access (via HMI Client). If not selected, remote access will use the same level of protection of local access.



**Important: This function only works when user management is enabled.**



**Tip:** Use this option when you have a default user but at the same time you want to protect remote access.

See "[Enable/disable security management](#)" on [page 172](#) for details.

The only files/folders still accessible when this flag is enabled are:

- PUBLIC folder and Index.html, that contain web console and public resources

See "[Modifying access permissions](#)" on [page 173](#) for details on HTTP access limits.



## 20 Audit trails

---

The Audit trail is a chronological sequence of audit records. Each record contains information on the actions executed and the user that performed them.

This function provides process tracking and user identification with time stamp for events.

If User Management is enabled, the actions are traced together with the name of the user. Only administrator user can modify this setting.

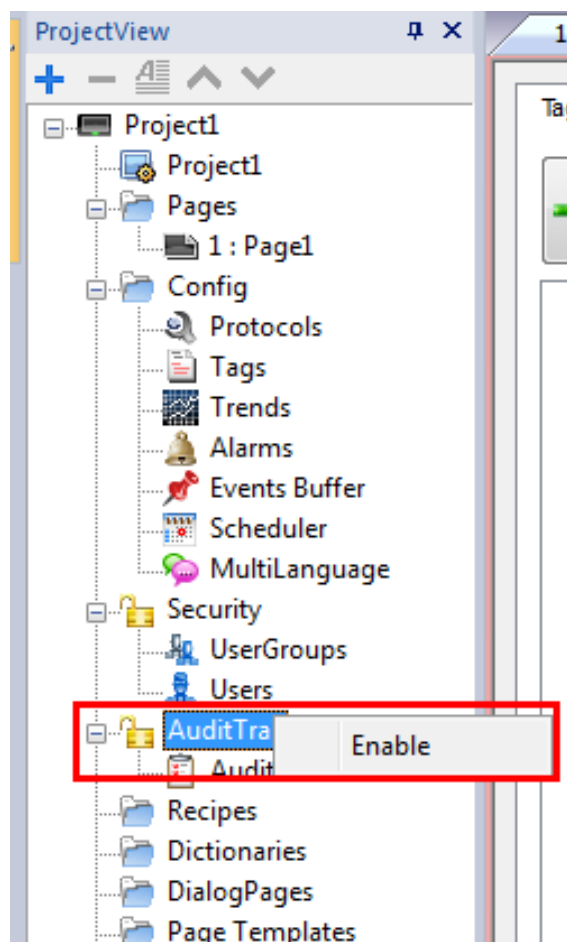
---

<b>Enable/disable audit trail .....</b>	<b>182</b>
<b>Configure audit events .....</b>	<b>182</b>
<b>Configure tags for audit trail .....</b>	<b>183</b>
<b>Configure alarms for audit trail .....</b>	<b>184</b>
<b>Configure recipes for audit trail .....</b>	<b>184</b>
<b>Configure login/logout details .....</b>	<b>185</b>
<b>Exporting audit trail as .csv files .....</b>	<b>185</b>
<b>Viewing audit trails .....</b>	<b>186</b>

## Enable/disable audit trail

Path: **ProjectView**> right-click **AuditTrail**> **Enable**

The padlock symbol indicates status of the function.

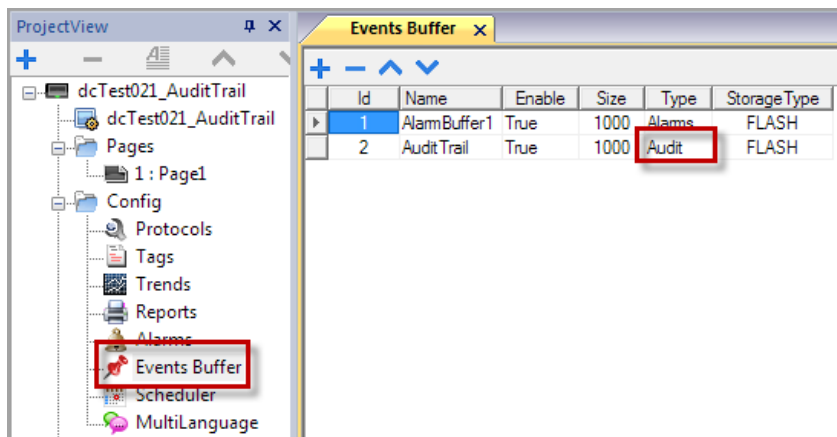


## Configure audit events

You can have more than one set of audit records. You need to configure a dedicated event buffer.

### Creating an event buffer

Path: **ProjectView**> **Config**> double-click **Event Buffer**



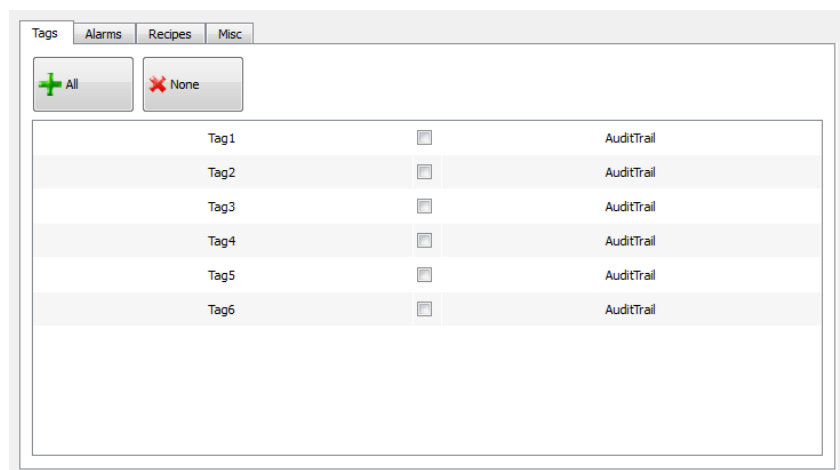
1. In the **Event Buffer** editor, click **+**: a row is added to the table.
2. Select **Audit** for **Type**.
3. Configure buffer parameters.

Parameter	Description
<b>Id</b>	Buffer identification number
<b>Name</b>	Buffer name
<b>Enable</b>	Enable/disable logging
<b>Size</b>	Size of log file. Data is automatically saved to disk every 5 minutes.
<b>Type</b>	Type of events logged: <ul style="list-style-type: none"> <li>• <b>Alarms</b></li> <li>• <b>Audit</b></li> <li>• <b>Generic</b></li> </ul>
<b>Storage Device</b>	Device where audit data will be stored

## Configure tags for audit trail

**Path:** *ProjectView* > *AuditTrail* > click *AuditTrail*

Track only the tags related to actions that you want to keep under control. For tracked tags, all write operations will be logged together with the time stamp and user that performed the operation.

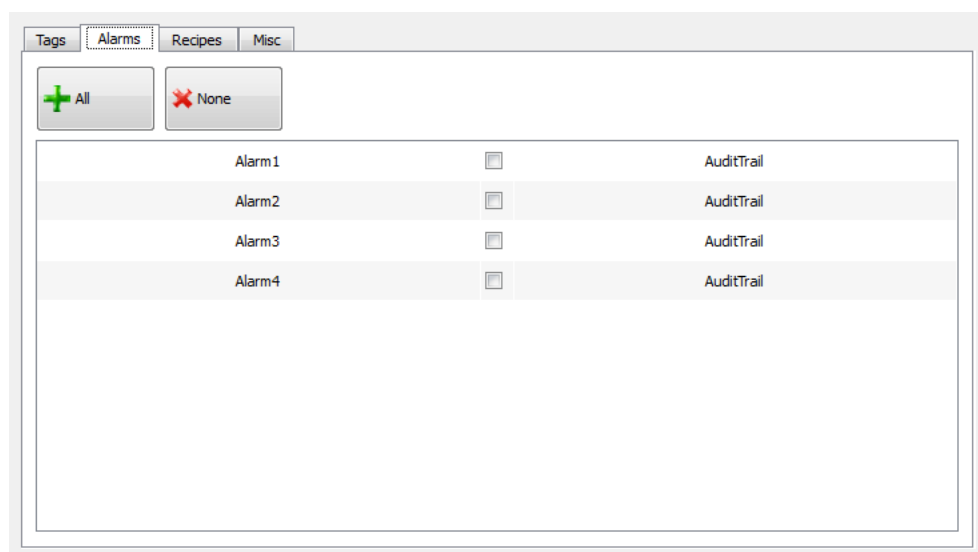


## Configure alarms for audit trail

Path: **ProjectView> AuditTrail> click AuditTrail**

You can specify the alarms to be tracked by the audit trail.

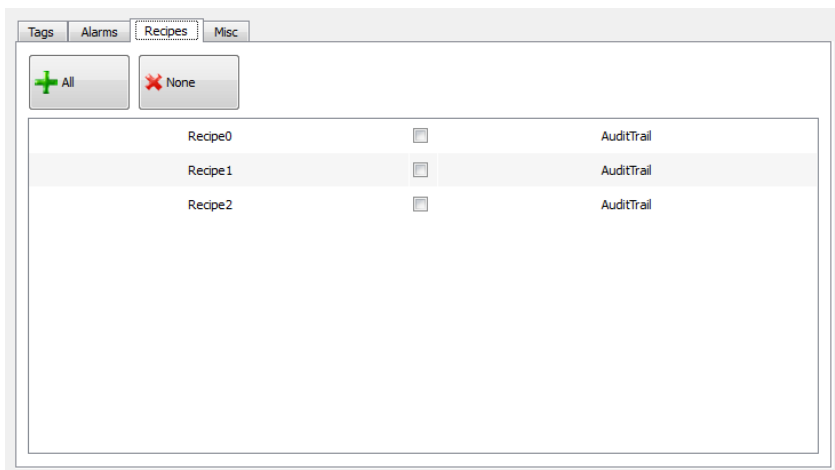
1. In Audit Trail editor, select the **Alarms** tab.
2. Select all the alarms to log in the audit trail: all operations performed on the specified alarms will be logged.



## Configure recipes for audit trail

Path: **ProjectView> AuditTrail> click AuditTrail**

Track only the recipes related to actions that you want to keep under control. For tracked recipes, all transfer operations will be logged together with the time stamp and user that performed the operation.

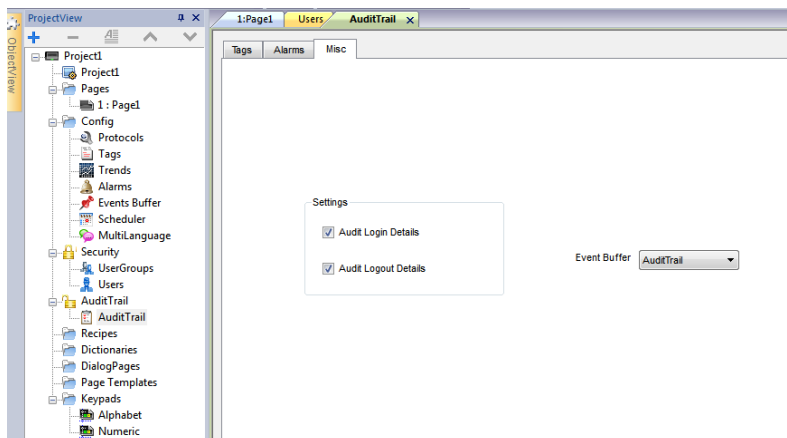


## Configure login/logout details

Path: **ProjectView**> **AuditTrail**> click **AuditTrail**

Audit trail can trace information about user login and user logout events.

1. In Audit Trail editor, select the **Misc** tab.



2. Select the information you want to log.
3. If you created additional event buffers of type **Audit**, then you can choose them from the **Event Buffer** combo box or you can leave the value **AuditTrail** that will use the default buffer.

## Exporting audit trail as .csv files

To view audit trail data you have to export it to a csv file using the **DumpEventArchive** action. See ["System actions" on page 88](#) for details.

## File structure

A	B	C	D	E	F
EventType	SubType	TimeStamp	Interface	Action	Information
18	1	2015-05-26T08:42:32.135+05:30	LOCAL	LOGIN	Status:1(S_OK); User:user2; Data:-1;
18	1	2015-05-26T08:42:35.607+05:30	LOCAL	WRITE_TAG	Status:1(S_OK); User:user2; Data:Tag4;111;
18	1	2015-05-26T09:01:30.635+05:30	CGI	LOGIN	Status:1(S_OK); User:admin; Data:c2367249b48189cde33fc43cc4352c56;
18	1	2015-05-26T09:01:30.647+05:30	CGI	LOGOUT	Status:1(S_OK); User:admin; Data:c2367249b48189cde33fc43cc4352c56;
18	1	2015-05-26T09:01:30.662+05:30	CGI	LOGIN	Status:1(S_OK); User:admin; Data:9e84a4f45b7afd310b768af62b59f57e;
18	1	2015-05-26T09:01:31.195+05:30	CGI	LOGOUT	Status:1(S_OK); User:admin; Data:9e84a4f45b7afd310b768af62b59f57e;
18	1	2015-05-26T09:01:31.196+05:30	CGI	LOGIN	Status:1(S_OK); User:admin; Data:5ee6d7fe1ef88c00f12da86d47a1f1f4;
18	1	2015-05-26T09:01:31.202+05:30	CGI	LOGOUT	Status:1(S_OK); User:admin; Data:5ee6d7fe1ef88c00f12da86d47a1f1f4;
18	1	2015-05-26T09:01:31.349+05:30	CGI	LOGIN	Status:1(S_OK); User:admin; Data:98f8942d1c587a232c4478b94f9e722e;
18	1	2015-05-26T09:01:35.446+05:30	CGI	WRITE_TAG	Status:1(S_OK); User:admin; Data:Tag5;222;
18	1	2015-05-26T09:01:38.696+05:30	CGI	WRITE_TAG	Status:1(S_OK); User:admin; Data:Tag1;1;
18	1	2015-05-26T09:01:41.163+05:30	CGI	WRITE_TAG	Status:1(S_OK); User:admin; Data:Tag1;0;
18	1	2015-05-26T09:01:44.109+05:30	CGI	ACK_ALARM	Status:1(S_OK); User:admin; Data:Alarm1;
18	1	2015-05-26T09:01:44.109+05:30	CGI	ACK_ALARM	Status:-1(E_FAIL); User:admin; Data:Alarm2;
18	1	2015-05-26T09:01:44.109+05:30	CGI	ACK_ALARM	Status:-1(E_FAIL); User:admin; Data:Alarm3;
18	1	2015-05-26T09:01:45.219+05:30	CGI	RESET_ALARM	Status:1(S_OK); User:admin; Data:Alarm1;
18	1	2015-05-26T09:01:45.219+05:30	CGI	RESET_ALARM	Status:-1(E_FAIL); User:admin; Data:Alarm2;
18	1	2015-05-26T09:01:45.219+05:30	CGI	RESET_ALARM	Status:-1(E_FAIL); User:admin; Data:Alarm3;

Exported data file has the following content:

<b>EventType</b>	For internal use
<b>SubType</b>	
<b>TimeStamp</b>	Event time stamp. Time can be configured as local or global from the dump action.
<b>Interface</b>	LOCAL, when the action is performed in HMI Runtime. CGI, when the action is performed by a remote client.
<b>Action</b>	Action executed.
<b>Information</b>	Action status and operation executed. For example, write Tag - Tag1:50

## Viewing audit trails

Audit trail data must be exported as a data file for viewing.

See ["Exporting audit trail as .csv files" on the previous page](#) for details.

# 21 Reports

---

A report is a collection of information that will be printed when triggered by an event. When the programmed event is triggered, the printing starts in background.

You can configure reports, their contents, trigger conditions and output printer in the Reports editor.

Not all widgets can be used in reports. When configuring reports, PB610-B Panel Builder 600 provides access to a dedicated widget gallery featuring only widgets available for reports.

Reports format can be customized using predefined templates for page layout.



Note: Report printing is not supported by HMI Client.

---



<b>Adding a report .....</b>	<b>188</b>
<b>Configuring text reports .....</b>	<b>188</b>
<b>Configuring graphic reports .....</b>	<b>189</b>
<b>Print triggering events .....</b>	<b>190</b>
<b>Default printer .....</b>	<b>191</b>

# Adding a report

Path: **ProjectView** > **Config** > double-click **Reports**

In **Reports** editor, click **Graphic Report** or **Text Report**: one new row is added to the table.

## Report types

Report type	Description
<b>Text Reports</b>	<p>Use for line-by-line printing of alarms.</p> <p>Only used for line printers.</p> <p>Text is sent to the printer without using any special driver.</p> <p> <b>Important: This printing mode requires using a physical port and only works on Windows CE platforms.</b></p>
<b>Graphic Reports</b>	<p>Contain graphical elements and may include complex widgets such as screenshots or alarms.</p> <p> <b>Important: Each printer requires a specific printer driver. See "<a href="#">Configuring graphic reports</a>" on the facing page for a list of supported printer drivers.</b></p>

## Configuring text reports

Use the **Reports** editor . **Paper Size** in number of characters.

### Setting printer options

Use printer options to control flush of pages on printer.

Printing starts either immediately or after a timeout. In printer options you can force flush as soon as a specific condition occurs, after a specified number of events, lines or seconds.



Note: Text reports do not support PDF format.

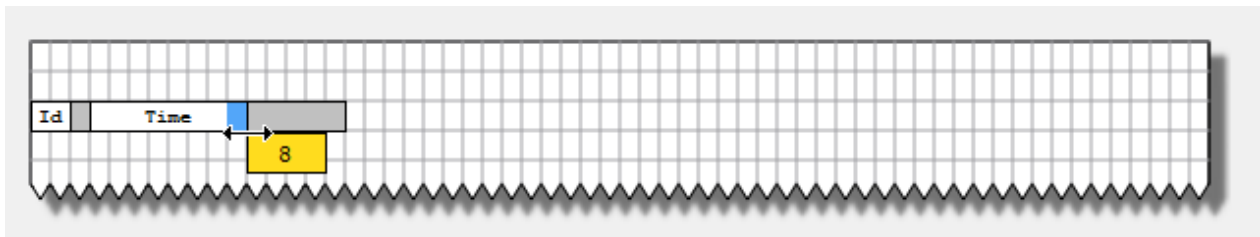
### Setting alarms layout

**Paper Size** is the width of paper in number of characters.

### Adding fields to the report

To add an item to the report, drag and drop it on the template page from the **Available fields** list.





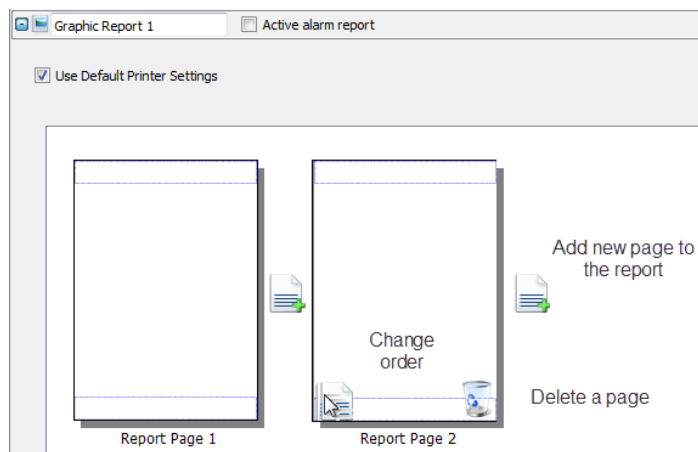
Re-size the field using the mouse, a tool tip shows the dimension in number of characters.



Note: If the text does not fit in the dedicated space, the auto wrap is applied.

## Configuring graphic reports

Use the **Report** editor to configure graphic reports.



### Adding a report page

Click **+** to add a new page to the report layout.

When the mouse goes over a page, two icons are displayed and allow you to reorder or delete the pages.

### Modifying report page content

1. Double click on a page to edit its content: the **Graphic Report** editor appears.


Each page is divided in: header, footer and page body.

2. Double click on the area you want to edit: the edit area is shown in white, others are grayed out.

The Widget Gallery is context-sensitive and displays only the widgets available for the area you are editing.

### Widgets available for reports

Widgets that can be used for a graphic report:

Widget	Function
Page Number	Automatic page numbering
Screenshot	Screen capture of the page currently displayed by the HMI device. The report page is automatically resized to fit the HMI device page.   Note: The full screen is printed, including all open dialogs.
Alarm	Entire contents of the event buffer (default buffer is Alarm Buffer1).
Text	Widgets such as labels and numeric fields

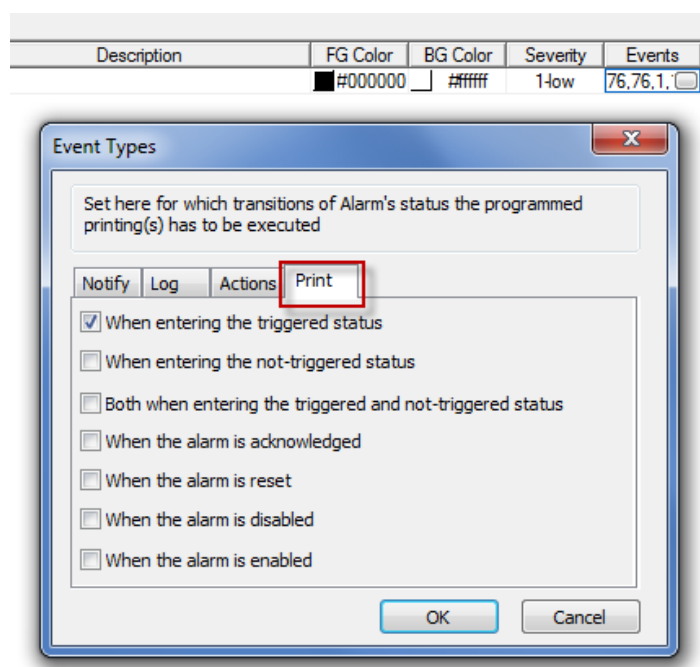
## Print triggering events

Report printing can be triggered by events.

### Configuring alarm printing

Path: **ProjectView** > **Config** > double-click **Alarms**

1. In the Alarms editor, open the **Event Types** dialog from the **Events** column.
2. In **Print** tab select all the conditions for which you want to trigger printing.

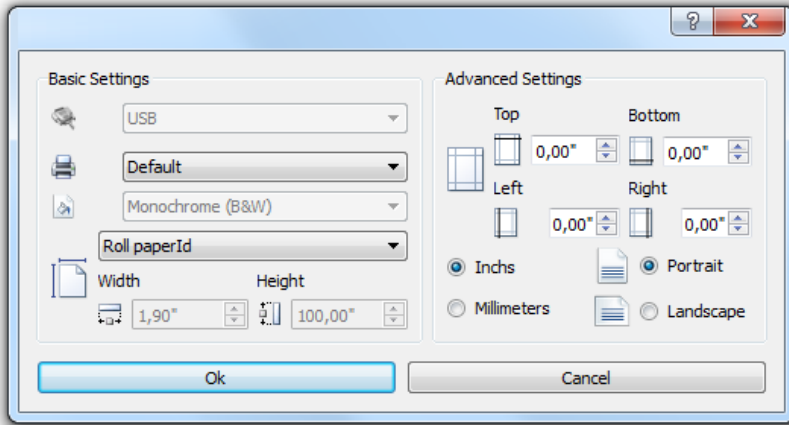


**Important:** Only one report can be set as Active alarm report in a project and it can be either a text report or a graphic report.

## Adjusting printer settings at run time

A graphic report printing can be started also using the action **PrintGraphicReport**.

Set the action property **silent** to **false** to have a pop-up dialog.



## Default printer

### Printer setting

You can set a default printer for all graphic reports. Each report can then be configured to use the default printer or any other printer available. Click **Printer Setting** button to set printer parameters.

For PDF printers you also define the folder where files are saved by using **Printed Files Location**.

### Supported printers

List of printers and printer languages supported by the Windows CE driver printCE.dll. Printers not available in the list but compatible with these languages are supported.

Printer	Languages
HP PCL 3, HP PCL 5e, HP PCL3GUI	HP PCL3/PCL5e/PCL3GUI, including DeskJet, LaserJet, DesignJet
Epson ESC/P2	ESC/P2, LQ
Epson Stylus Color	Epson Stylus Color
Epson LX (9-pin)	9-pin printers, Epson LX, FX, PocketJet
Cannon iP100, iP90, BubbleJet	BubbleJet, iP90, iP100
PocketJet II, 200, 3	Pocket Jet
MTE Mobile Pro Spectrum	MTE Mobile Pro Spectrum
Adobe PDF File	Adobe PDF file

Printer	Languages
<b>SPT-8</b>	SPT-8
<b>M1POS</b>	M1POS
<b>MP300</b>	MP300
<b>Zebra</b>	Zebra CPCL language
<b>Intermec PB42, PB50, PB51, PB2, PB3</b>	Intermec PB42/50/51/2/3 with ESC/P language
<b>Datamax Apex</b>	Datamax Apex

## Supported ports

The following ports are supported:


- LPT1 (USB printers)
- File (PDF)




Note: On Win32 platform, only PDF and default printers are supported. Default printer is the default OS printer and it can be connected with any kind of port (not only USB).

## Tested printers

The following printers have been tested with printCE drivers in Windows CE HMI devices.

Driver	Printer Model	Graphic	Line
<b>Custom</b>	Plus 4 Kube II	Yes	Yes
<b>Epson ESC/P 2</b>	Epson AcuLaser M2310	Yes	Simulate
<b>Epson LX (9-pin)</b>	Epson LX-300+II	No	Yes
<b>HP PCL 3</b>	HP LaserJet P2015dm	Yes	Simulate
	HP LaserJet 4700dtn	Yes	Yes
<b>HP PCL 3 GUI</b>	HP Deskjet 1010	Yes	No
	HP Deskjet D5560	Yes	No
	HP LaserJet 4700dtn	No	Yes
<b>HP PCL 5e</b>	HP LaserJet P2015dm	Yes	Simulate
	HP LaserJet 4700dtn		
<b>INTERMEC</b>	Intermec PB50 with ESC/P language with 4 inch roll paper.  Note: The HMI device crashes when trying to print on Inter-	Yes	Yes

Driver	Printer Model	Graphic	Line
	 mec PB50 printers in standby mode after a first successful print job.		
PDF	-	Yes	No



## 22 Screen saver

Screen saver can be used to display a slide show when the HMI device is not in use. The slide show starts after a timeout if none of the following events occur:

- touch of display
- mouse movement
- external keyboard key pressed

### Enabling the screen saver function

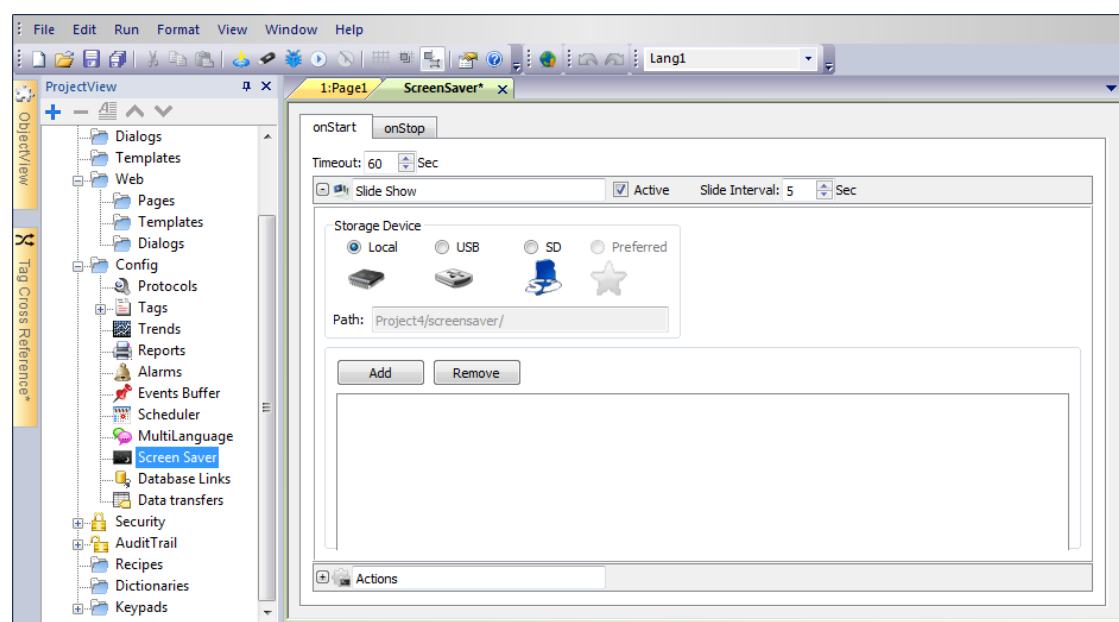
Path: **ProjectView** > **Config** > right-click **Screen Saver** > **Enable**



**Important:** You must enable the screen saver before you can configure it.


### Configuring a screen saver

Path: **ProjectView** > **Config** > double-click **Screen Saver**



### Slide show parameters


Parameter	Description
Timeout	Time after which the slide show starts
Slide Interval	Interval between slides

Parameter	Description
<b>Storage Device</b>	<p>Location of the images used in the slide show.</p> <p>Images stored locally are saved in <i>workspace\projectname\screensaver</i> and can be downloaded to the HMI device when the project is downloaded.</p> <p>Images stored on USB or SD devices are saved in a screensaver folder on the device itself.</p> <p> <b>Important: Only JPEG and PNG images are supported.</b></p>

## Associating actions to the screen saver

Actions can be triggered by the screen saver start and/or stop.

- Click **+** next to **Actions** in the **onStart** tab to configure actions to be executed when the screen saver starts.
- Click **+** next to **Actions** in the **onStop** tab to configure actions to be executed when the screen saver stops.

 Note: The screen saver function is supported by Windows CE & Win32 devices and can also be used in HMI Client client.



## 23 Backup/restore of Runtime and project

---

You can backup all the content of the HMI device, including HMI Runtime and project, to an external memory. This backup copy can be used to restore the content of the HMI device at a later time or copy it to a new HMI device.

The backup function is available only if enabled for the logged user. See ["Modifying access permissions" on page 173](#) for details.



Note: Backup is available only on Windows CE platform. It is not supported in Win32 / HMI Client.

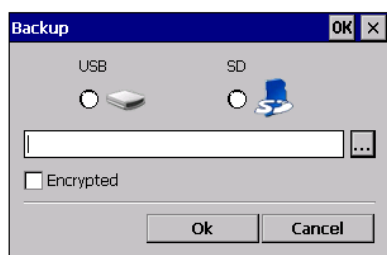
### Backup function

The backup function automatically performs the following procedure:

1. Unloads the current project to unlock files in use.
2. Archives the content of the \QTHMI folder (containing HMI Runtime, projects, dynamic files such as recipes, alarms, trends and so on) to a .zip file (standard or encrypted).
3. Reloads the project.

To start the backup procedure:

1. In HMI Runtime right click to open the context menu.
2. Select **Backup**: the **Backup** dialog is displayed.



3. Select the path for storing the backup file.



Note: The backup process does not include files stored in USB and SD cards. Dynamic data such as recipes, trends, events stored in these devices will not be included in the backup.

### Restore function

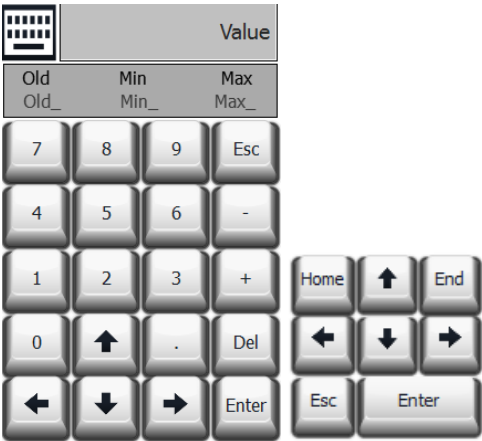
Restore the backup package using the **Transfer from disk** option in the Loader menu.

Select the backup file: the system will automatically check for compatibility with the current platform and install it.



# 24 Keypads

Several keypads are provided by default in the PB610-B Panel Builder 600 so that they can be used for data entry. Here are a few examples:



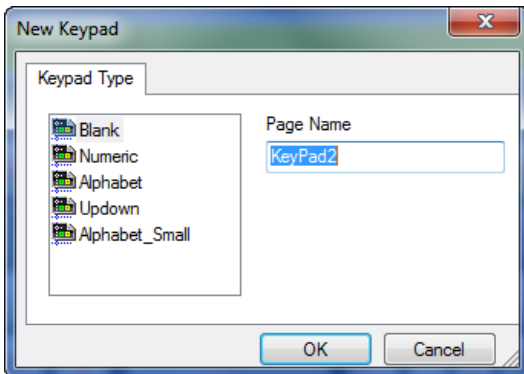
Creating and using custom keypads .....	200
Deleting or renaming custom keypads .....	202
Keypad type .....	202
Keypad position .....	203

# Creating and using custom keypads

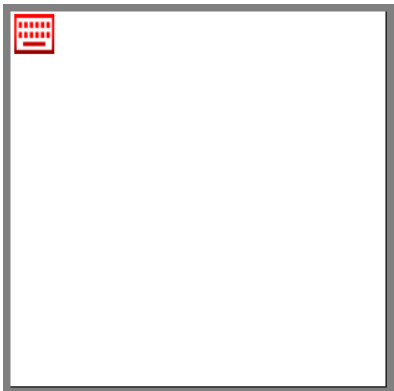
You can either create a new keypad or customize an existing one.

## Creating a keypad

1. In **ProjectView**, right-click **Keypads** and select **Insert Keypad**: the **New Keypad** dialog is displayed.



2. Select one of the available keypads, or **Blank** to create a keypad from scratch. In this case a blank keypad is displayed.



3. Use the **Keypad Widgets** and **Keypad Buttons** from the Widget Gallery to create your custom keypad.

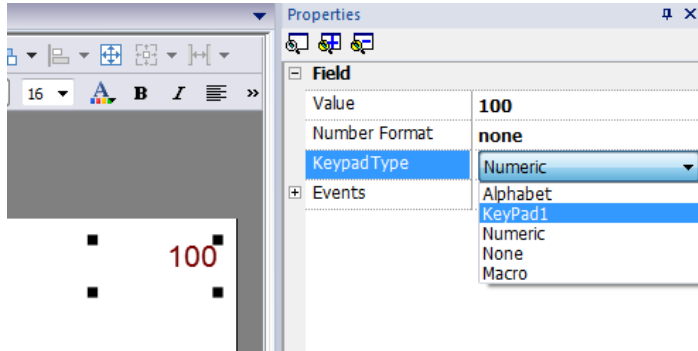


The keypad you create, as in this example, will be saved in the project folder.



## Attaching custom keypads to fields

Custom keypads can then be reused for any field where the **Keypad** property points to it as in this example.



## Tips and tricks with custom keypads

By default, any numeric widget (read/write numeric field) are assigned the numeric keypad.

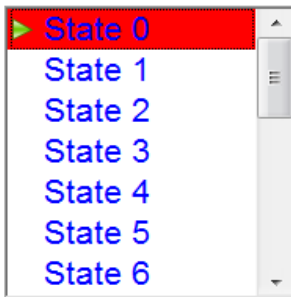
If you want to apply a customized version of the numeric keypad to all the numeric widgets you add to your project proceed as follows:

1. Create a new keypad and select **Numeric** as **Keypad** type. This will be a backup of the original settings for the numeric keypad.
2. Customize the default numeric keypad and save it. This customized version of the numeric keypad will now be assigned as default in the project.

See ["Deleting or renaming custom keypads" on the next page](#) for details on how to rename a custom keypad.

## Up-down arrows keypad

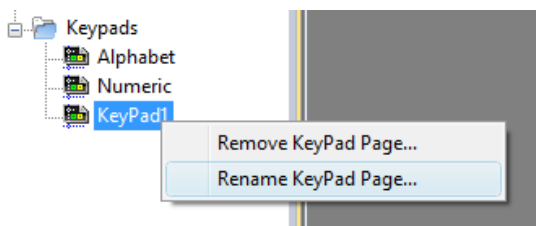
This type of keypad is particularly useful to move the cursor up and down within widget requiring this functionality. Here an example using a **Control List** widget. See ["Control list widgets" on page 233](#) for details.



## Deleting or renaming custom keypads

In **ProjectView**, right-click on a custom keypad and select one of the options:

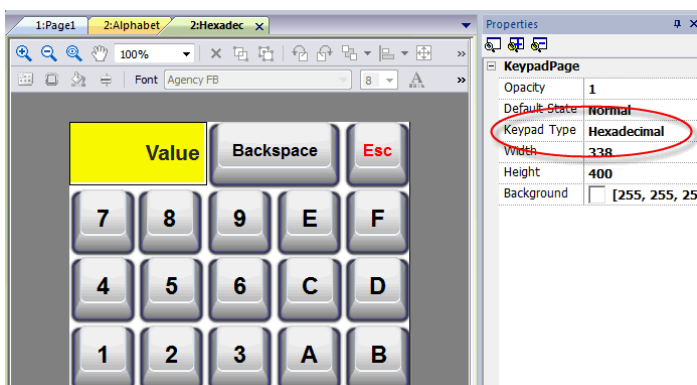
- **Remove KeyPad Page** to remove the keypad from the project
- **Rename Keypad Page** to rename the keypad.



## Keypad type

**Path:** *ProjectView* > *Keypads* > double-click a keypad > *Properties*

Set **Keypad Type** parameter for a keypad to define the type of data entry.



Keypad Type	Description
<b>Auto</b>	Default setting
<b>Decimal</b>	Only numeric keys are accepted. Entering 10, the keypad returns 10 that will be displayed as "10" if the attached field is numeric or ASCII, as 'A' if the attached field is hexadecimal.
<b>Hexadecimal</b>	Only hexadecimal keys are accepted. Entering 10, the keypad returns 16 that will be displayed as "16" if the attached field is numeric or ASCII, as "10" if the attached field is hexadecimal.
<b>Ascii</b>	All keys are enabled. Entering 1A, the keypad returns 1A that will be displayed as '1' if the attached field is numeric, as "1A" if the attached field is ASCII or if the attached field is hexadecimal.

## Keypad position

**Runtime Positioning** property of keypads can be used to define where keypads will appear in the screen.

Option	Description
<b>Automatic</b>	The best position is selected according to here data entry is required.
<b>Absolute</b>	X,Y coordinates are entered to identify the exact position
<b>Left-top</b>	Predefined screen positions
<b>Left-center</b>	
<b>Left-bottom</b>	
<b>Center-top</b>	
<b>Center-center</b>	
<b>Center-bottom</b>	
<b>Right-top</b>	
<b>Right-cente</b>	
<b>Right-bottom</b>	

Select the **Lock Keypad position** option if you don't want the keypad to be moved by dragging.





# 25 External keyboards

HMI Runtime has been designed to work with external keyboards connected via USB.

Keyboards can be used for:

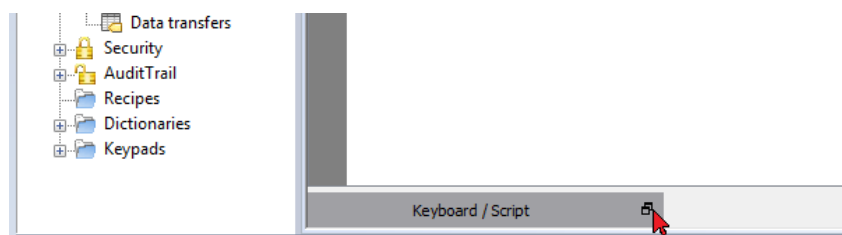
- data entry (default)
- execute actions mapped on specific keys

For example, the right arrow key **OnClick** event can be mapped to the **LoadPage** action.

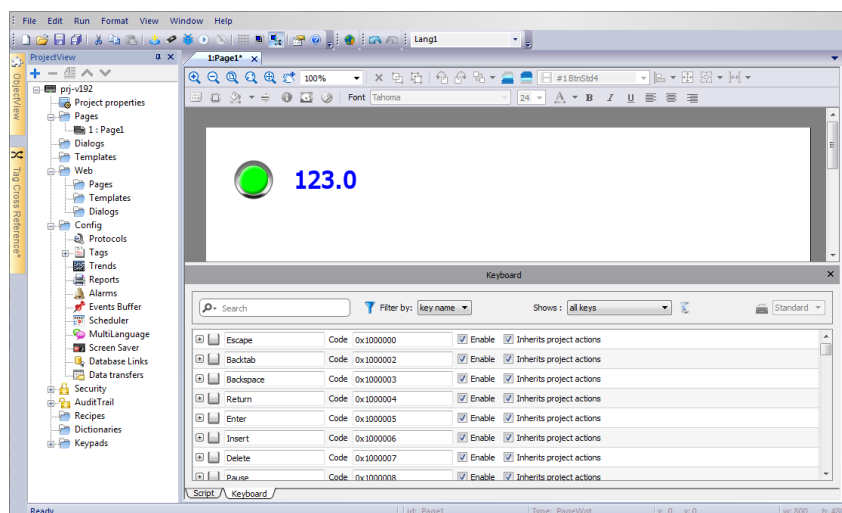
Keyboard can be programmed at project level so that settings will be inherited by all the pages. In each page you can then choose which key setting will be inherited from the project and which one you will customize for the specific page.

## Opening external keyboards

1. In the Page Editor, click on the icon on the right of **Keyboard/Script** at the bottom of the workspace: the Keyboard/Script Editor is displayed.
2. Select the **Keyboard** tab.



Each row in the Keyboard Editor corresponds to a key.



For each key, the following information is displayed:

Element	Description
Label	Key name
Code	Key code
Enable	Key enable status
Inherits project actions	Defines whether the key is inheriting the action programmed at the project level

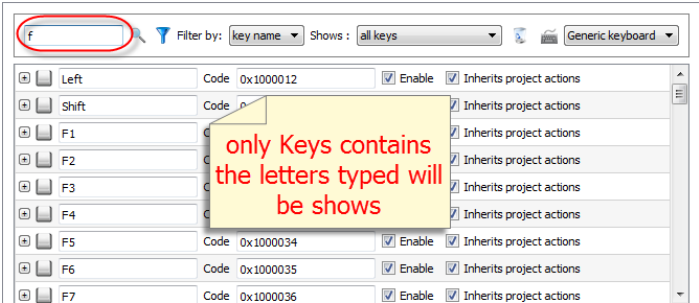
Here the possible configurations:

Enable	Inherits project actions	Editor appearance	HMI Runtime behavior
Checked	Unchecked	Action lists show the page actions (or nothing if the list is empty)	Only the page actions (if any) will be executed.
Checked	Checked	Action lists show the project actions only and cannot be edited	Only the configured project actions (if any) will be executed.
Unchecked	Checked	Inherits project actions check box and all action lists are disabled. Action lists show the project actions only.	No page or project action will be executed.
Unchecked	Unchecked	Inherits project actions check box and all action lists are disabled. Action lists show the project actions only.	No page or project action will be executed.

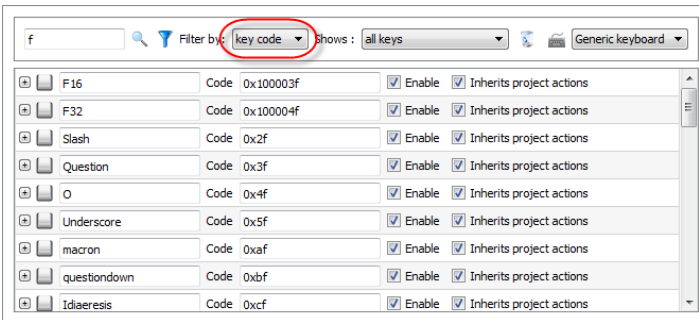
<b>Search and filter .....</b>	<b>207</b>
<b>Displayed keys .....</b>	<b>207</b>
<b>Removing action associations .....</b>	<b>207</b>
<b>Keyboard layout .....</b>	<b>208</b>
<b>Enable/disable keyboard .....</b>	<b>208</b>
<b>Associating actions to keys .....</b>	<b>208</b>

# Search and filter

To display a filtered set of keys, in **Filter by** select **key name** and type a letter in the search field: only the keys containing that letter in their name will be displayed in the Keyboard editor.



Alternatively, in **Filter by** select **key code** and type a letter in the search field: only the key containing that letter in their code will be displayed in the Keyboard editor.



# Displayed keys

You can easily select what keys will be listed in the Keyboard editor window. To display a limited set of keys, select an option in **Shows**.

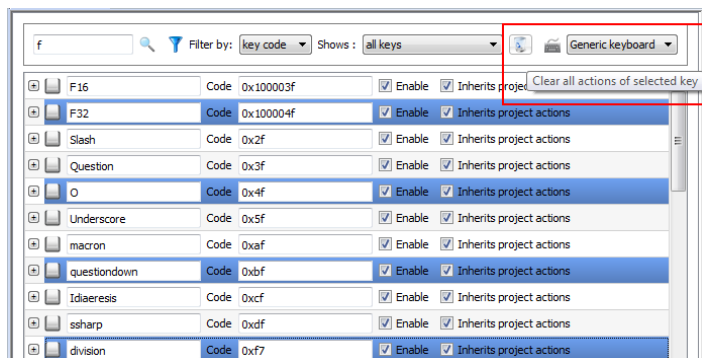
Option	Description
all keys	All keys available in the keyboard layout are listed
modified keys	Only the keys associated with actions at the page level are listed
modified keys in project	Only the keys associated with actions at project level are listed

# Removing action associations

To remove all the associations you created between keys and actions:

1. Select the keys for which you want to remove the association.
2. Click the **Clear all actions of selected keys** button.

If you are working at page level, page actions will be removed, if you are working a project level, project actions will be removed.

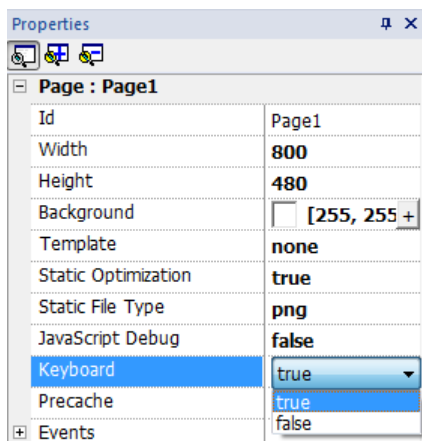


## Keyboard layout

Select the layout of the keyboard from the **Keyboard Layout** combo box. **Generic Keyboard** refers to a generic international keyboard layout.

## Enable/disable keyboard

You can enable/disable keyboard actions both at project and at page level. To enable keyboard actions, in the **Properties** pane set **Keyboard macro** to **true**.

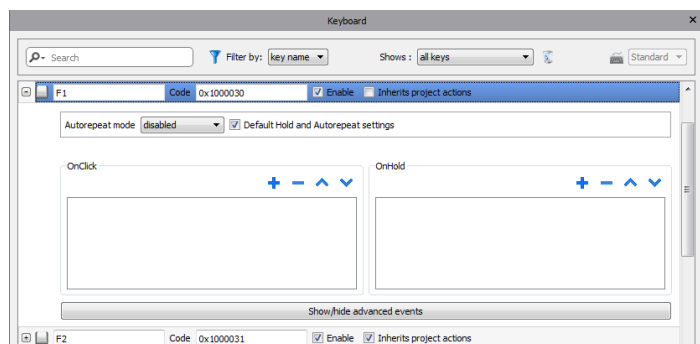


You can enable/disable keyboard actions also at run time using the KeyboardMacros action. See ["Keyboard actions" on page 77](#) for details.

## Associating actions to keys

You associate actions to a keys from the Keyboard editor.

1. Click + next to the key you want to program: the fields for key configuration are displayed.



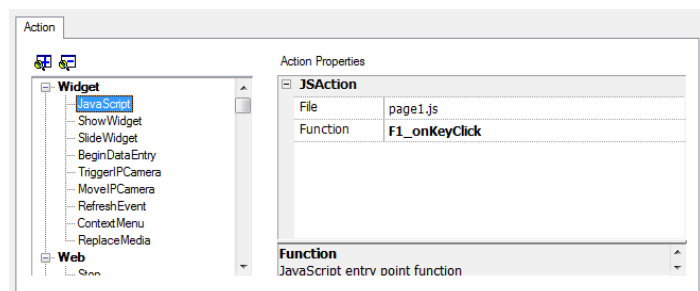
1. Click **+** to add actions.

You can associate actions both to the **OnClick** event and to the **OnHold** event.

See ["Events" on page 32](#) for details.



Note: Also JavaScript code can be associated to a key event.





## 26 Tag cross reference

---

The **Tag Cross Reference** pane displays a list of tag names used in current project organized according to their location and use.

From this pane you can:

- verify where each tag is used (alarms, pages, recipes, schedulers, trends, and so on)
- identify invalid tag references (references to tags not defined in the tag editor)
- identify tags not used in the project



Note: The Tag Cross Reference pane does not list tags used in JavaScript code.

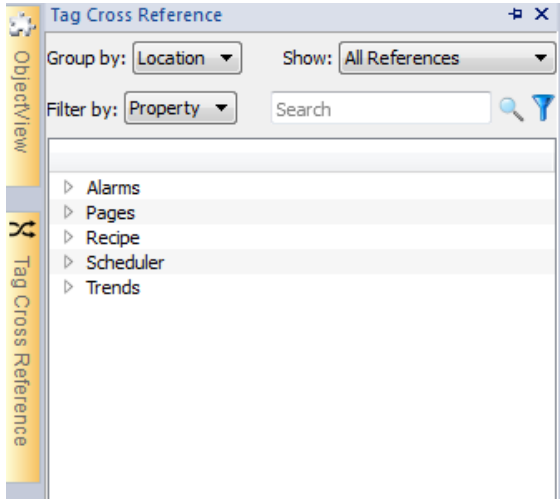
---

Updating data in the Tag Cross Reference pane .....	212
---	-----

## Opening the Tag Cross Reference pane

**Path:** *View > Toolbars and docking windows > Tag Cross Reference*

Click the **Tag Cross Reference** tab to open the Tag Cross Reference pane.



## Working in the Tag Cross Reference pane

The Tag Cross Reference pane provides a set of standard functions.

Element	Function
<b>Group by</b>	Groups tags by <b>Location</b> (alarms, pages, trends and so on) or <b>Tag</b> name
<b>Show</b>	Filters tags and displays: <ul style="list-style-type: none"> <li>• <b>All Reference:</b> all tags</li> <li>• <b>Invalid Tag Reference:</b> tags not listed in the Tag Editor.</li> <li>• <b>Unused Tags:</b> tags listed in the Tag Editor but not used in project.</li> </ul>
<b>Search field</b>	Applies a filter to display a limited number of tags
<b>Filter by</b>	Filters tags by <b>Location</b> , <b>Tag</b> or <b>Property</b> .

Navigate the listed tags to find where they are used inside the project.

Double-click on a tag to open the editor or page where it is used.

## Updating data in the Tag Cross Reference pane

### Manual update

By default, the information displayed in the Tag Cross Reference pane must be updated manually. To do this, click the

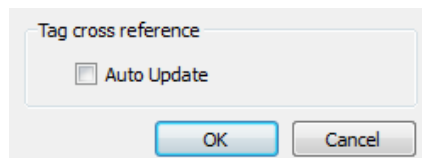
refresh button . A warning sign is displayed when a refresh is needed.



## Automatic update

*Path:* **View > Properties**

You enable the automatic update of the Tag Cross Reference pane from the PB610-B Panel Builder 600 **Properties** page.



Select the **Auto Update** option.

## Exporting data

Data displayed in the Tag Cross Reference pane can be exported in .csv file.

Data is organized in the exported file according to how it was grouped in the pane.

Grouped by	File format
Location	RESOURCE, RESOURCE DESC, WIDGET-ID, ATTRIBUTE, TAG
Tag	TAG, RESOURCE, RESOURCE DESC, WIDGET-ID, ATTRIBUTE



Note: The separators used in export operation depends on regional settings of your computer.



# 27 Indexed addressing

---

Indexed addressing allows you to select a set of tags depending on the value of another tag. This is very useful, for example, to use the same graphics to visualize a set of data coming from different sources, all the user has to do is pick the source to monitor from a list.

---

Creating an indexed addressing set .....	216
Using indexed tag set in pages .....	219

# Creating an indexed addressing set

## Scenario

In this scenario, environment data is collected from with four rooms, each equipped with temperature, pressure, and humidity sensors. Data is available as follows:

Room Number	Temperature	Pressure	Humidity
1	Room1-Temperature	Room1-Pressure	Room1-Humidity
2	Room2-Temperature	Room2-Pressure	Room2-Humidity
3	Room3-Temperature	Room3-Pressure	Room3-Humidity
4	Room4-Temperature	Room4-Pressure	Room4-Humidity

Using the indexed addressing feature, you can use a single table format to arrange all data in the HMI device.

Data from the three different sensors can be displayed in a single page where the room number is used as a selector (combo box) to pick the correct set of tags.

Room 1

Temperature (°C) 21  
Pressure 1  
Umidity (%) 75

## How to create an indexed tag set

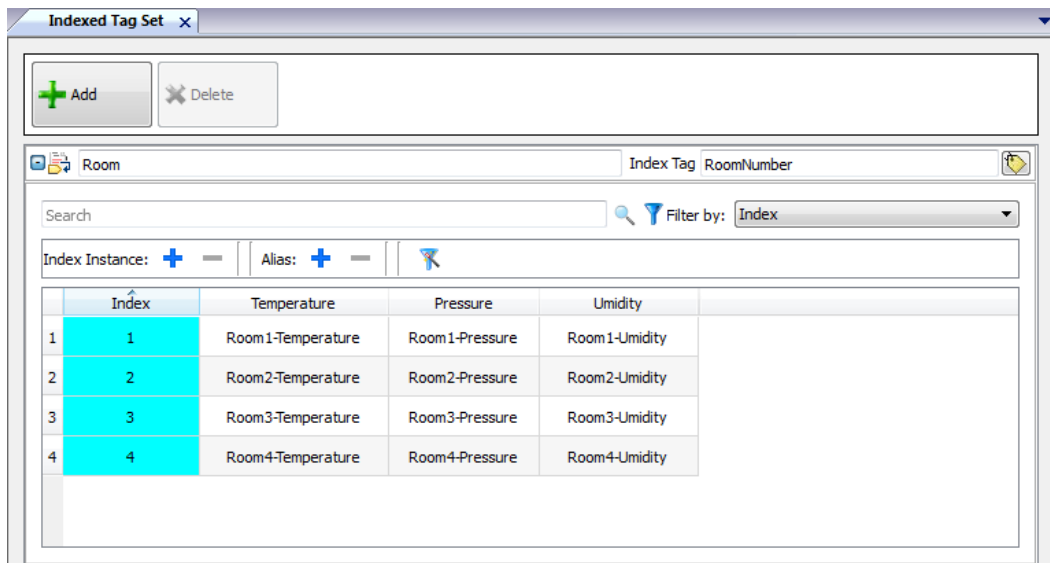
Path: **ProjectView> Tags**

To do this you need to create an indexed tag set.

1. In the Tag Editor, define protocols and tag. Define a tag for each data to be indexed, in this example you must create a tag for each sensor in each room.

Name	Group	Driver	Address
Room1-Temperature		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400001 unsignedShort
Room1-Pressure		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400002 unsignedShort
Room1-Umidity		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400003 unsignedShort
Room2-Temperature		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400004 unsignedShort
Room2-Pressure		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400005 unsignedShort
Room2-Umidity		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400006 unsignedShort
Room3-Temperature		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400007 unsignedShort
Room3-Pressure		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400008 unsignedShort
Room3-Umidity		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400009 unsignedShort
Room4-Temperature		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400010 unsignedShort
Room4-Pressure		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400011 unsignedShort
Room4-Umidity		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400012 unsignedShort

2. Create a tag to be used as index tag. In this example you create a "RoomNumber" tag that could be of type UnsignedInt using Variable protocol.
3. From **ProjectView**, select **Config> Tags**, double-click **Indexed Tag Set**: the Indexed Tag Set editor is displayed.
4. Click + to add an Indexed Tag Set. In this example you will call it "Room".
5. Select the tag "RoomNumber" to use as a selector for the room number.
6. Create an **Index Instance** for each set of data. In this example, one for each room.
7. Create an **Alias** for each type of data and rename the table columns appropriately. In this example "Temperature", "Pressure" and "Humidity".
8. Double-click on each cell to associate the correct tag.



Note: The Index Tag datatype can be a number, a string or any type of simple data types.

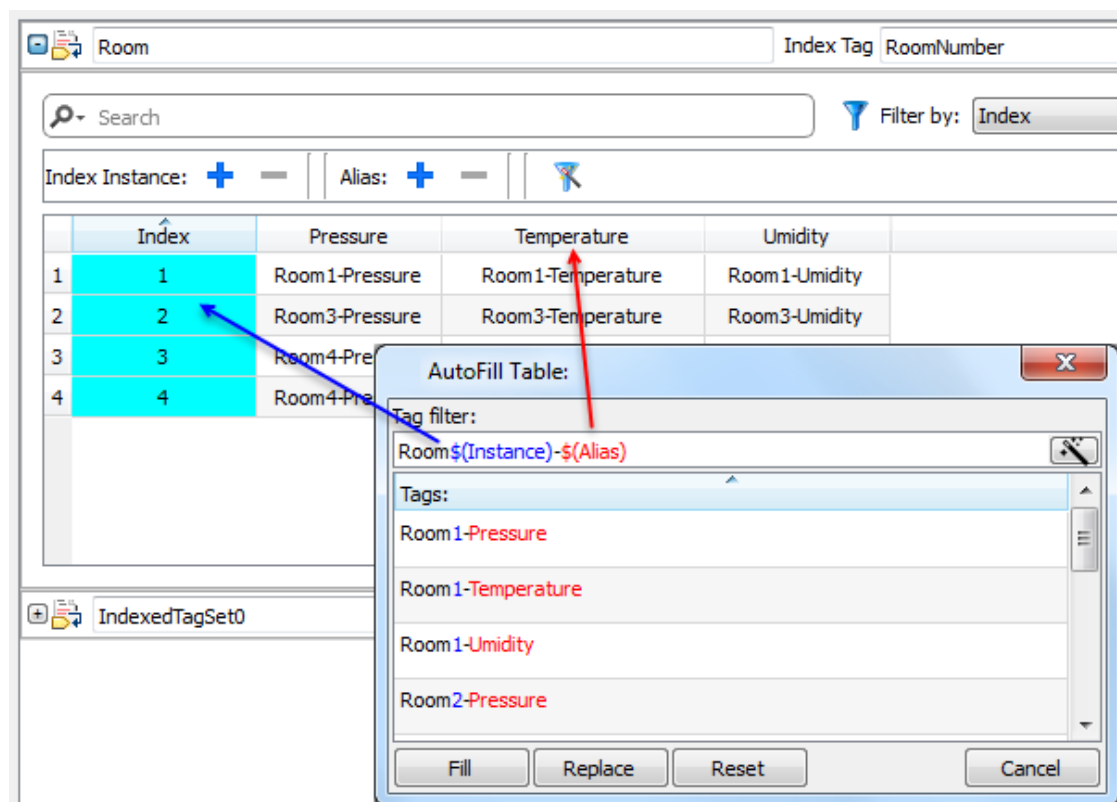


Note: To reference an array data type use the array index = -1

## Autofill function

An Indexed Tag Set table may become very complex and filling it may be an error prone procedure. Enable the Autofill feature to make sure aliases are entered correctly.

Click to enable the Autofill feature: the **Autofill Table** is displayed.



This function uses regular expression for populating the table with tags trying to match the filter where the keyword `$(Instance)` will be replaced with the defined Index values and the keyword `$(Alias)` with the defined alias labels.

## Autofill example


“Room\$(Instance)-\$(Alias)” will match all tag names:

Room1-Temperature,  
Room1-Pressure,  
Room1-Humidity,  
Room2-Temperature,  
...

“Room0\*\$(Instance)-\$(Alias)” will match all tag names:

Room1-Temperature,  
Room01-Pressure,  
Room001-Humidity,  
Room2-Temperature,  
Room02-Pressure,  
Room002-Humidity,  
...

## Autofill table elements

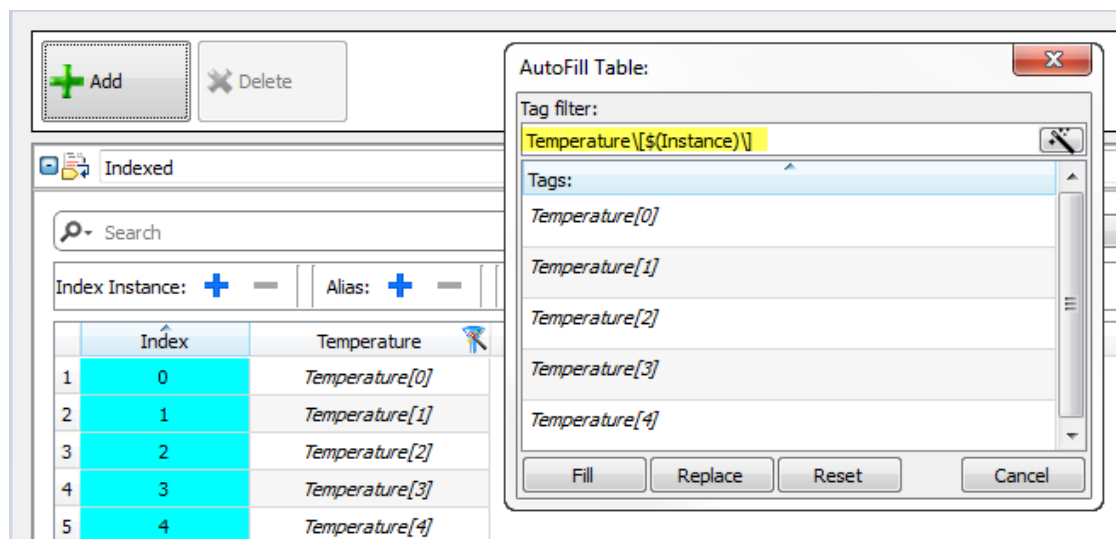
Element	Description
<b>Fill</b>	Fills in missing entries in the tag table using the set filter (if any). For example, when new instances or new aliases are added you can use this option to fill in the new entries.
<b>Replace</b>	Replace all table entries with those provided by the Autofill table.
<b>Reset</b>	Resets the tag filter to empty, no automatic fill is done.
	Suggests a valid filter expression for your project.



Note: Filters are saved as project preferences and can be set for the entire table or for a column. Once a filter is set for a column, the table filter is ignored. You can therefore selectively change the filter for handling a particular alias only.



Note: To reference the elements of an array use the \ character to disable the regular expression interpretation of the square brackets (array tags are differentiated by *Italic*).



## Using indexed tag set in pages

Once an indexed tag set has been created, you can use it to create a page for the HMI device as in this example.

Room 1

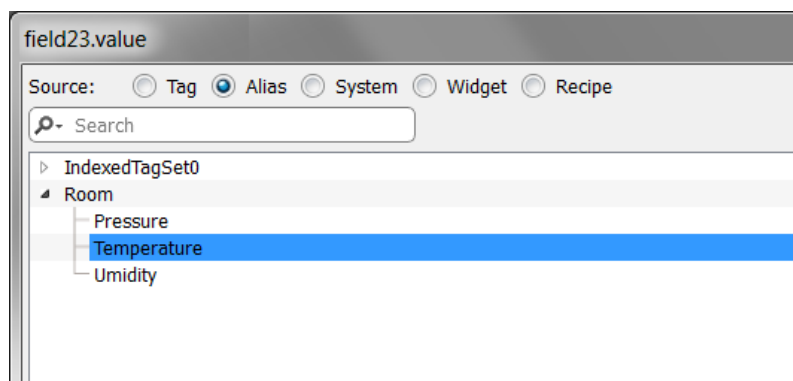
Temperature (°C)	21
Pressure	1
Umidity (%)	75

To create this page:

1. Create a page and add a combo box, three labels and three numeric fields.
2. Use the index tag created for the room number for the combo box, "RoomNumber" in this example. This will be the selector for the room number.
3. Create a list for the combo box. In this example use the following list.

Index	String List
0	Room Number
1	Room 1
2	Room 2
3	Room 3
4	Room 4

4. Attach to each numeric field value the corresponding Alias variable (**Room > Temperature**, **Room > Humidity**, **Room > Pressure**).





## 28 Special widgets

---

Widgets designed for special purposes are called special widgets and include control lists, date and time widgets, variable widgets and so on.

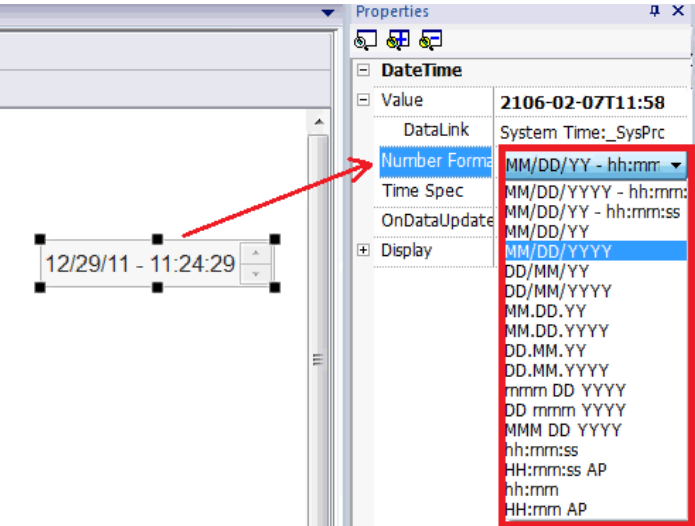
---

<b>DateTime widget</b> .....	<b>222</b>
<b>Multistate Image widget</b> .....	<b>222</b>
<b>Multistate Image Multilayer widget</b> .....	<b>223</b>
<b>Combo Box widget</b> .....	<b>225</b>
<b>Consumption Meter widget</b> .....	<b>226</b>
<b>RSS Feed widget</b> .....	<b>228</b>
<b>Scrolling RSS Feed widget</b> .....	<b>229</b>
<b>IPCamera widgets</b> .....	<b>229</b>
<b>Browser widget</b> .....	<b>232</b>
<b>Control list widgets</b> .....	<b>233</b>
<b>Variables widget</b> .....	<b>235</b>

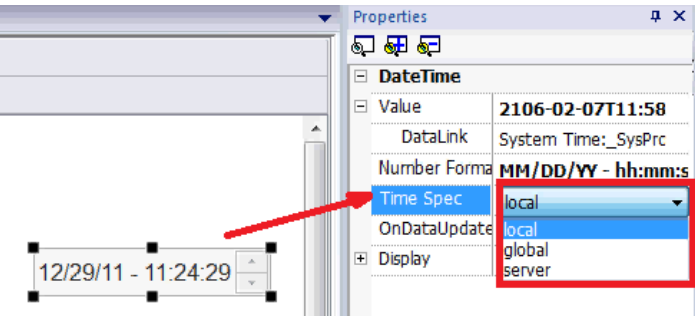
# DateTime widget

Path: *Widget Gallery> Basic> Controls*

Use this widget to display and edit current date and time .  
In the **Properties** pane different formats are available for representing date and time.



For the **Time Spec** property select which time the widget will show at run time.



## Time options

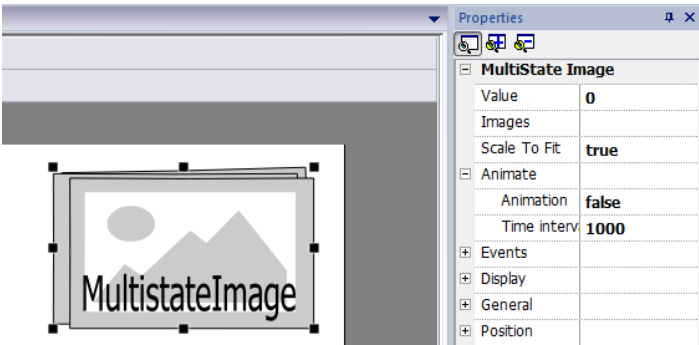
Option	Description
local	shows local time, the time of the HMI device where the project is running
global	shows Global Time (GMT)
server	shows time information as handled by the server side of the HMI device

See "[Runtime modes](#)" on [page 6](#) for details on system architecture.

# Multistate Image widget

Path: *Widget Gallery> Basic> Images*

Use this widget to display an image from a collection based on the value of a tag used as Index. You can use this widget also for simple animations.

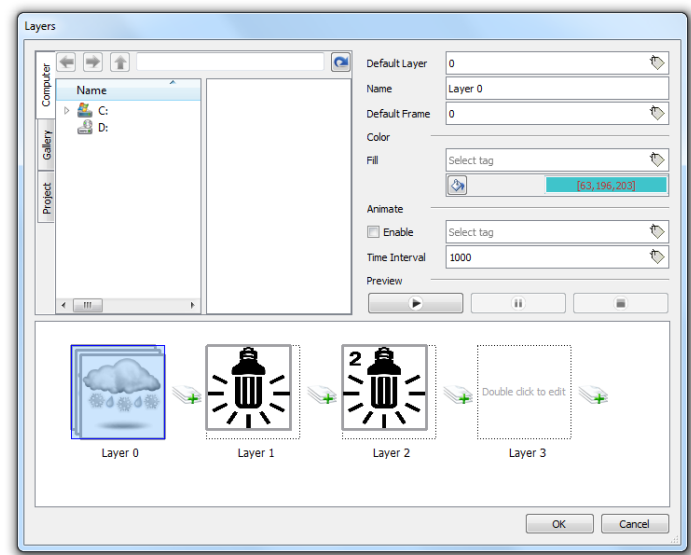


Parameter	Description
Value	Index of image to display. For example, set Value=0, to display the image with index 0 in the image collection.
Images	Images collection with associated index.
Animate	Set to true, to enable a slide show.
Time interval	Interval between images in the slide show.

## Multistate Image Multilayer widget

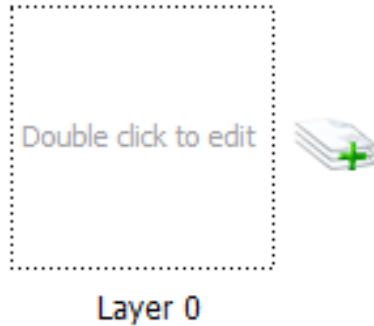
**Path:** *Widget Gallery> Basic> Images*

Use this widget to create different animations and select the most suitable at run time.

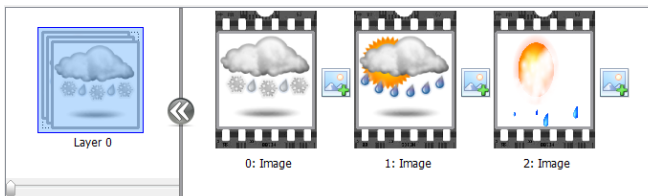


## Setting up widget layers

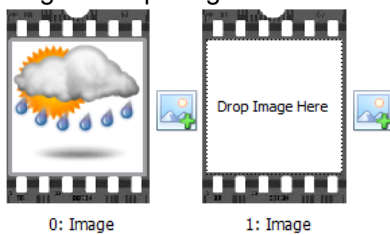
1. Open the **Layers** dialog from the **Properties** pane.
2. Click **+** to add as many layers as you need.



3. Double click on each layer to add as many images as you want to include in the layer.



4. Drag and drop images into the frame to add it to current layer.



5. Define widget properties.

Parameter	Description
<b>Default Layer</b>	Layer shown at run time.
<b>Name</b>	Name of selected layer.
<b>Default Frame</b>	Frame shown when current layer is displayed.
<b>Color / Fill</b>	Fill color for images of current layer.
<b>Animate</b>	Enables slide show for active layer. Animations can be started/stopped at run time attaching it to a tag.
<b>Time Interval</b>	Time interval of slide show, if enabled.
<b>Preview</b>	Slide show simulation.

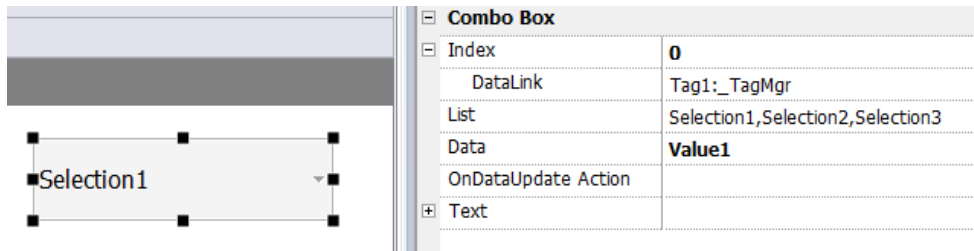




Note: **Default Layer**, **Default Frame**, **Color** and **Fill** can be changed at run time, attaching the to a tag.

## Combo Box widget

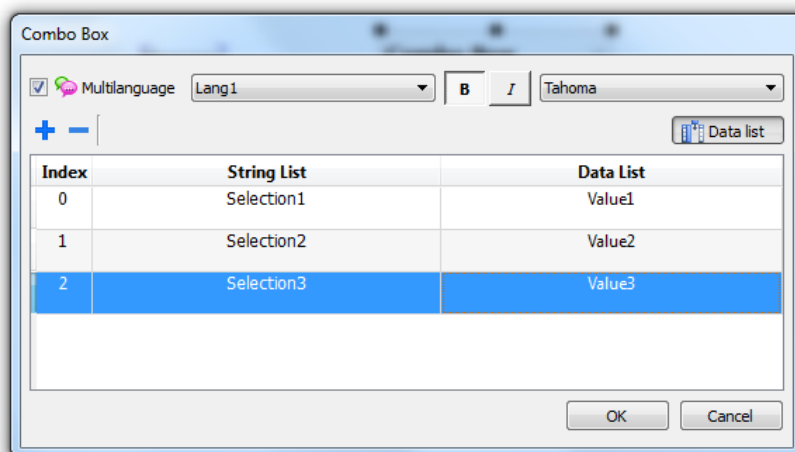
Path: **Widget Gallery**> **Basic**> **Controls**

Use this widget as a selector widget or to filter rows in a table to display only the values selected in the combo box.



Parameter	Description
Index	Index of the selected item.
List / String List	Item strings in the combo box.  Note: This field is multi-language.
Data / Data List	Returns the value in the Data List column (as string) in the Data field of the widget.  Tip: Use this parameter to return a custom value based on an item selected in the combo box.
Text	Format of displayed text.

## Attaching data vs. attaching indexes



In many projects you may need to attach fields such as **Index** or **Data** to tags to know the values of the selected item in the combo box. Use:

- **Index**: to display the index (integer) of the selected item (0...n).
- **Data**: to display the data value (string) specified in the Data List column.

## Consumption Meter widget

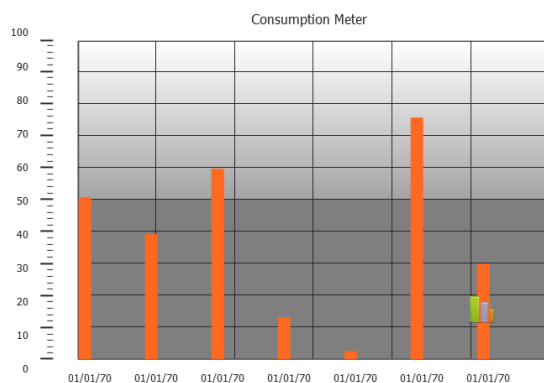
**Path:** *Widget Gallery > Basic > Trends/Graphs*

Use this widget to monitor a resource which is continuously increasing. The system reads the value of the resource and calculates the increment in a set range of time, the increment is then displayed in a bar-graph in a trend-like window.

Different colors can be used to used in the graph based on the time frame.



Tip: Use this widget to calculate the power consumption of a system.

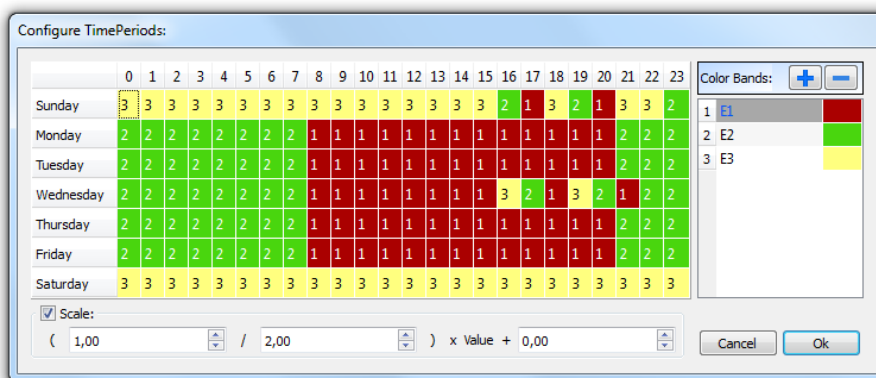


Parameter	Description
<b>Value</b>	Resource monitored
<b>Graph Duration / Graph Duration Units</b>	Time period displayed in the window
<b>Bar Duration/Bar Duration Units</b>	Time period represented by each bar in the graph
<b>Time Periods</b>	Assigns a specific color to highlight the increment of the monitored resource in a specified time period (minimum resolution = 1 hour).
<b>Consumption Meter</b>	Number of labels to be displayed on graph.

### Example: how to monitor energy consumption

In the following example a widget is design tho monitor energy consumption with a weekly scale and a daily unit.

1. Attach a tag to the physical variable to monitor. In this example, to the total energy consumed (Tag KWh). This tag contains an incremental number that indicates how many KW/h have been consumed from when energy consumption started.
2. Add a Trend and link it to the tag to be monitored, Tag KWh.
3. Add a **Consumption Meter** widget to a page.
4. Attach the **Value** property of the Consumption Meter to the Trend you created in step 2.
5. Set **Graph Duration/Units** to 1 week: this will give you a weekly graph of consumed energy.
6. Set **Bar Duration/Units** to 1 day, this is the time range when energy consumption is calculated.
7. In **Consumption Meter** set the number of labels to show in the bar graph, in this case 7 to display a weekly graph.
8. From the **Time Periods** property open the **Configure Time Periods** dialog: set the different colors for different values of Tag KWh in each bar.




Tip: To assign the color to the cells of the table, select the cells and click on the desired color, or enter the index value of the band (1, 2, 3) into the cell.

9. Add as many color bands as you need, in this example 3 color bands.
10. Assign a band to each hour in the weekly table, in this example a red band (E1) is used to indicate the range of time in the day/week where the cost of energy is the highest.



Note: You can apply a scale factor to each color band, if needed.

Consumption Meter	
Value	
DataLink	Trend3:IdalHistoDataWgt1
Graph Duration	1
Graph Duration Units	week
Bar Duration	1
Bar Duration Units	day
Time periods	Periods (3)
Color	 [255, 104, 32]
Bar Width	15
Show Background Image	true
Consumption Meter	
MinY	0
MaxY	100
X Labels	7
Y Labels	11

The result is a bar graph consumption meter showing daily consumption of energy in KW/h, with colors indicating the different energy costs. The height of each bar represents the amount of energy in the time range considered, 1 day in this example.

Use the action ConsumptionMeterPageScroll to scroll the bar graph back and forth and the action RefreshTrend to refresh the bar graph since data is not refreshed automatically.



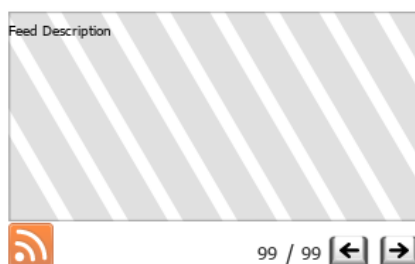
**Important: No other Trend action is currently supported by the Consumption Meter widget.**


## RSS Feed widget

**Path:** *Widget Gallery > Media > RSSFeed Source*

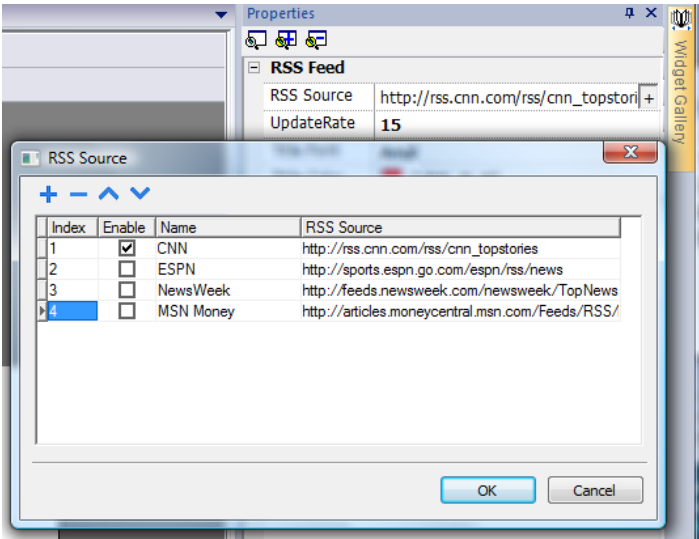
Use this widget to display on the HMI device your favorite RSS feeds directly from the Internet.

RSSFeed



Parameter	Description
RSS Source	Feed URL  Note: Feed sources cannot be modified at run time.
UpdateRate	Refresh time





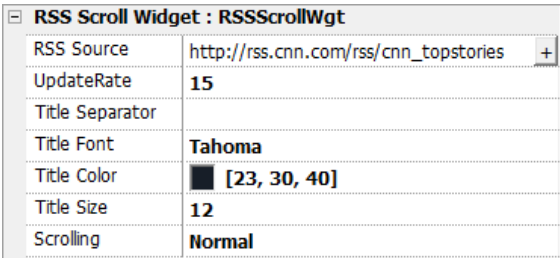
The RSS Feed widget has been specifically designed to work with Pocket Internet Explorer.

# Scrolling RSS Feed widget

Path: **Widget Gallery**> **Media**> **RSSFeed Scroll**

Use this version of the main RSS Feed widget to display highlights inside a text line using a smoothing scrolling text.

[RSSFeed Scroll](#) 



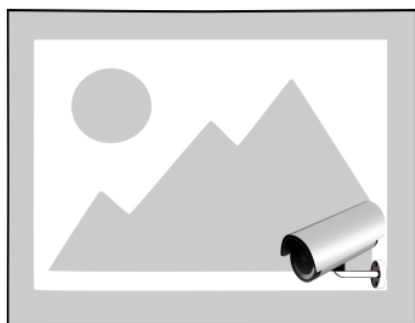
This widget has additional properties.

Parameter	Description
Scrolling	Scrolling speed
Title Separator	Separator character between highlights

# IPCamera widgets

Path: **Widget Gallery**> **Media**> **IP Camera**

Use these these widgets to show images captured from an IPCamera or a video stream.



Parameter	Description
<b>Camera URL</b>	URL of the IPCamera when used in JPEG format.
<b>Refresh Rate</b>	Number of JPEG images for second allowed. Max rate = 1 fps.
<b>User Name</b>	Name of user allowed to access the camera. Set this parameter when access to the camera is password protected.
<b>Password</b>	Password to access the camera.
<b>MJPEG Camera URL</b>	URL of MJPEG streaming (for example, http://192.168.0.1/video.cgi)

When this widget is used to stream HTTP MJPEG, **Camera URL** and **Refresh Rate** are ignored.

Performance of streaming is not fixed and depends on many factors such as: frame size, frame compression level, CPU of HMI device, quality of IPCamera. Based on these factors the widget can reach up to 25 fps.

You can add multiple IPCamera widgets, but this will reduce the frame rate fore each widget.

## Supported IPCameras

The following IPCamers have been tested so far:

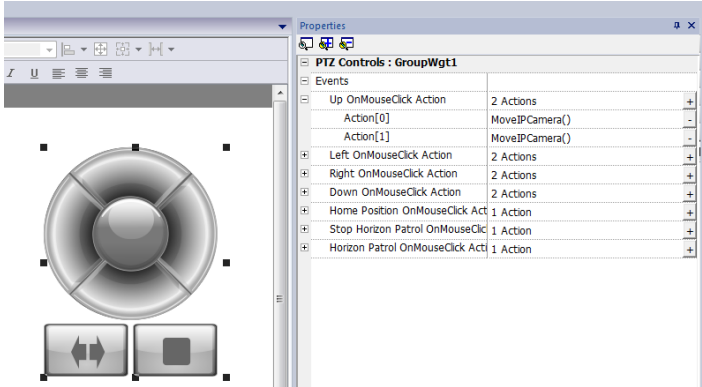
IPCamera	Protocol	URL
<b>Apexis APM-J901-Z-WS PTZ IP Camera</b>	MJPEG	http://{ip_address}/videostream.cgi
	HTTP	http://{ip_address}/snapshot.cgi
<b>AXIS M3027-PVE Network Camera</b>	MJPEG	http://{ip_address}/axis-cgi/mjpg/video.cgi
	HTTP	http://{ip_address}/axis-cgi/jpg/image.cgi
<b>DAHUA DH-IPC-HD2100P-080B 1.3mp Outdoor Vandalproof</b>	HTTP	http://{ip_address}:9988/onvif/media_service/snapshot
<b>D-Link DCS-5605 PTZ</b>	MJPEG	http://{ip_address}/video/mjpg.cgi
<b>D-Link DCS-900W IP Camera</b>	MJPEG	http://{ip_address}/video.cgi
<b>D-Link DCS-932L</b>	MJPEG	http://{ip_address}/video.cgi

IPCamera	Protocol	URL
<b>Edimax IC-7100P PTZ</b>	MJPEG	http://{ip_address}/mjpg/video.mjpg
	HTTP	http://{ip_address}/picture.jpg
<b>Foscam FI8916W</b>	MJPEG	http://{ip_address}/videostream.cgi
	HTTP	http://{ip_address}/snapshot.cgi
<b>Foscam FI9803 EP</b>	MJPEG	http://{ip_address}:88/cgi-bin/CGIStream.cgi?cmd=GetMJStream&usr={user}&pwd={pass}
		NOTE: <ul style="list-style-type: none"> <li>• port 88 may be different as per IP Camera settings</li> <li>• {user} = username defined into IP Camera settings</li> <li>• {pass} = password defined into IP Camera settings</li> </ul>
<b>Hamlet HNIPCAM IP Camera</b>	MJPEG	http://{ip_address}/video.cgi
	HTTP	http://{ip_address}/image.jpg
<b>MOXA VPort 254</b> (Rugged 4-channel MJPEG/MPEG4 industrial video encoder)	MJPEG	http://{ip_address}/moxa-cgi/mjpeg.cgi
	HTTP	http://{ip_address}/moxa-cgi/-getSnapShot.cgi?chindex=1
<b>NVS30 network video server</b>	MJPEG	http://{ip_address}:8070/video.mjpeg
	HTTP	http://{ip_address}/jpg/image.jpg
<b>Panasonic WV-Series Network Camera</b>	MJPEG	http://{ip_address}/cgi-bin/mjpeg
<b>Ubiquiti UniFi Video Camera</b>	HTTP	http://{ip_address}:7080/images/snapshot/camera/{camera_guid}?force=true
		NOTE: <ul style="list-style-type: none"> <li>• {camera_guid} can be found into IP Camera Webpage</li> <li>• port 7080 may be different as per IP Camera settings</li> </ul>
<b>Zavio F3210 2MP Day &amp; Night Compact IP Came</b>	MJPEG	http://{ip_address}/stream?uri=video.pro3
	HTTP	http://{ip_address}/cgi-bin/view/image?pro_0
		NOTE: <ul style="list-style-type: none"> <li>• MJPEG video streaming can be configured selecting "video profile 3" with 640x480 resolution into IP Camera settings.</li> </ul>

# PTZ Controls widget

PTZ (pan-tilt-zoom) cameras are cameras capable of remote directional and zoom control.

The PTZ Controls widget uses the MoveIPCamera action to send HTTP/cgi commands to the PTZ IPCamera.



Parameter	Description
Camera URL	URL of IPCamera
User Name	Name of user allowed to access the camera. Set this parameter when access to the camera is password protected.
Password	Password to access the camera.
Command	Command to send to the PTZ controller (for example, decoder_control.cgi?command=0)

# Authentication methods

The authentication method is automatically set by the camera web server to which the widget connects. Authentication methods supported are:

- Basic
- NTLM version 1
- Digest-MD5

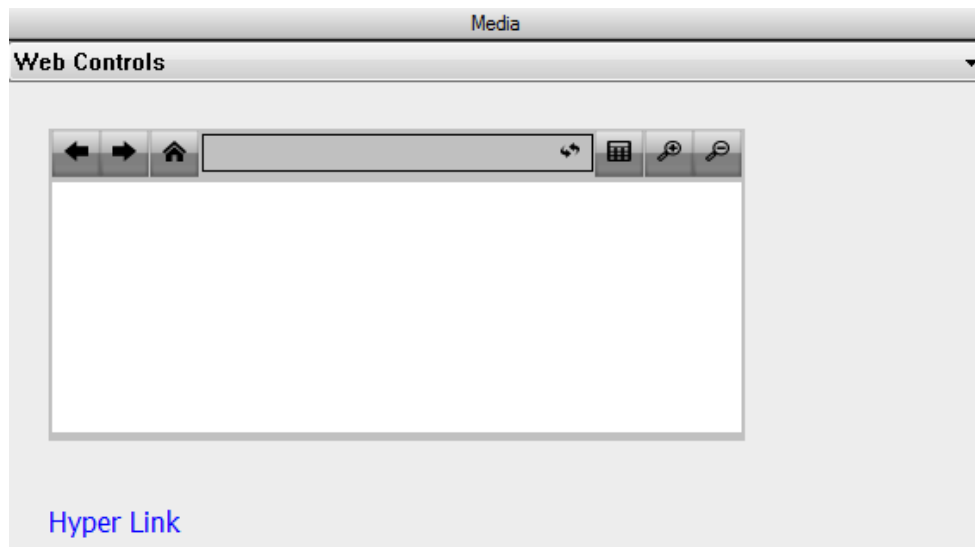
# Browser widget

Path: **Widget Gallery> Media> Web Controls**

Use this widget to embed web pages into your HMI device pages. This is an HTML5 compatible browser widget based on the WebKit engine.



**Important: This widget is not supported by MIPS based devices.**



Parameter	Description
<b>Home Page</b>	Default URL to open when widget is shown on the page.
<b>Zoom to Fit</b>	Automatically scales content to the size of view area.
<b>Time out</b>	Page load timeout in seconds.
<b>Clear History</b>	Automatic history clear on load
<b>Scroll</b>	Shows/hides scrollbars
<b>Show Progress cursor</b>	Shows/hides loading cursor

This allows you to save around 3 MB of space if the widget is not required in your project.

An **Hyper Link** widget is available to create pages hyperlinks. Once clicked these links notify to the browser widget that a particular web page is to be loaded.



**Important: HTTPs protocol is not supported.**

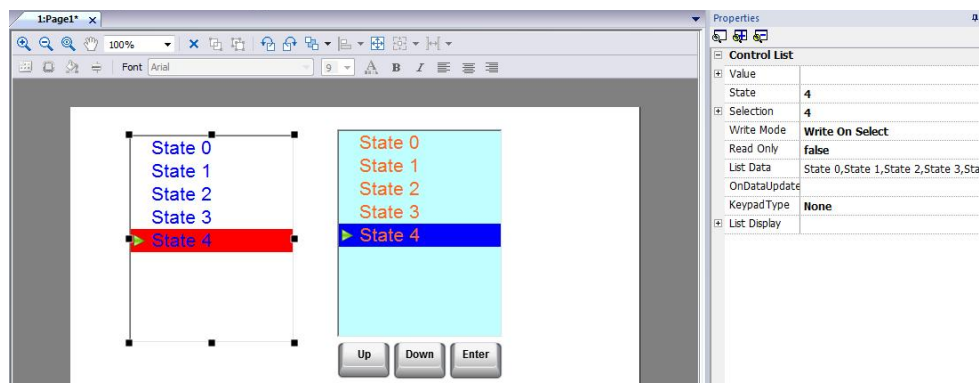
## Control list widgets

*Path: **Widget Gallery**> **Advanced**> **Control List***

Use these widgets to represent the status associated with a particular process and to control that process from the same widget.

Two types of control lists are available:

- a group control list, with a limited set of navigation button already included, and
- a basic control list with no pre-configured button to be navigated using the touch screen feature.

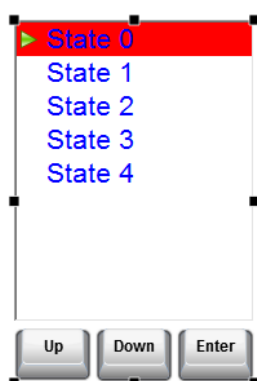
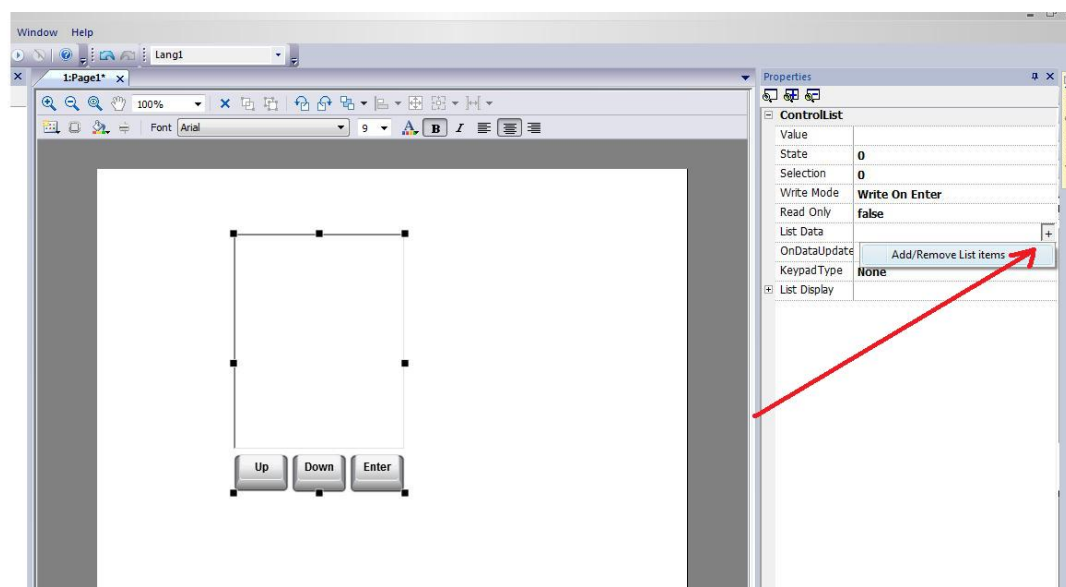


Parameter	Description
<b>Value</b>	<p>If <b>Write mode</b> is <b>Write On Select</b>: value of the item selected.</p> <p>If <b>Write mode</b> is <b>Write On Enter</b>: value of item selected and confirmed pressing enter button.</p> <p>This field can be attached to a tag to control selected and confirmed item.</p>
<b>State</b>	Default state when widget is loaded.
<b>Selection</b>	Currently selected item, displayed as a highlight cursor moving up and down. This property can be attached to a tag.
<b>Write Mode</b>	<p><b>Write On Select</b>: the value is automatically written to the tag when one of the items is selected.</p> <p><b>Write On Enter</b>: the value is written to the tag only when one of the items is selected and the enter key is pressed.</p>
<b>Read Only</b>	Defines whether the list is only an indicator.
<b>List Data</b>	Adds/removes list items.

## Defining states

Add/remove states, that is items in the list, from the **List Data** property.

Any value can be assigned to a state. When you activate the state, by selecting the related item if in **WriteOnSelect** mode or selecting it and confirming with enter if **Write On Enter**, this will write the value assigned to state to the tag linked to the Control List widget **Value**.



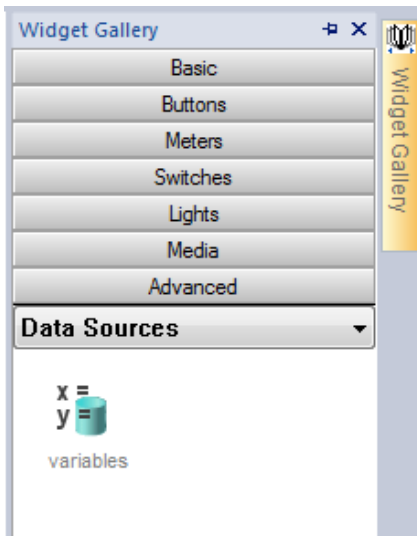
## Variables widget

**Path:** *Widget Gallery > Advanced > Data Sources*

Use this widget to add internal variables for operations such as data transfer or to be used in JavaScript programs.



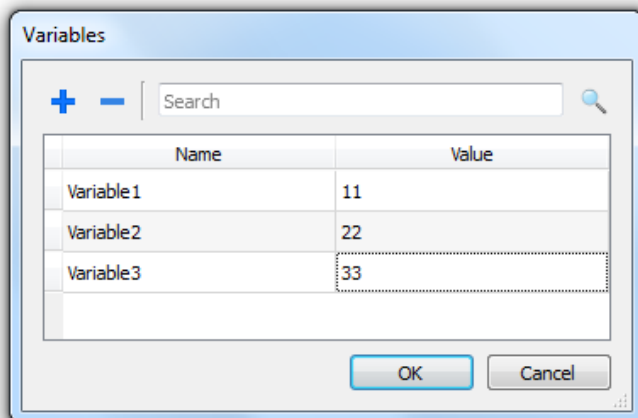
Note: The variables are local to the page where the widget has been inserted.



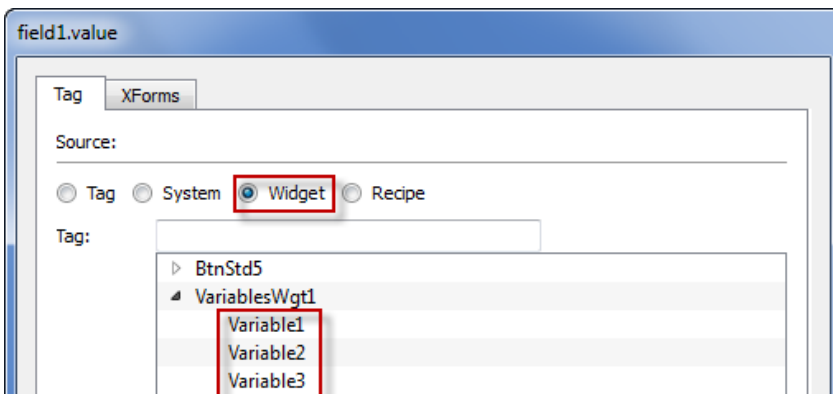
When you drag and drop this widget into your page, a placeholder will be displayed to indicate the widget location, but it will not be visible at run time.

## Setting the widget

To create variables and assign values to them, open the **Variables** dialog from the **Variables** property in the **Properties** pane.

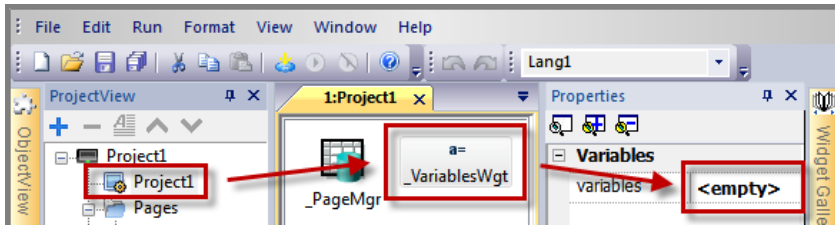


These variables can then be referenced from the **Attach tag** dialog, from the Page Editor.





If you need global variables, configure them at project level, adding the desired variables to the global variable widget.



## Using variables in JavaScript

Variables can be also referenced in JavaScript programs with the following syntax:

For local variables:

```
var varWgt = page.getWidget("_VariablesWgt");
var compVar = varWgt.getProperty("VariableName");
```

For global variables:

```
var varWgt = project.getWidget("_VariablesWgt");
var compVar = varWgt.getProperty("VariableName");
```



# 29 Custom widgets

---

PB610-B Panel Builder 600 has a large widget library which includes predefined dynamic widgets (buttons, lights, gauges, switches, trends, recipes, and dialog items), as well as static images (shapes, pipes, tanks, motors).

You can drag and drop an object from the gallery to the page, and then size, move, rotate or transform it. All widgets in the gallery are vector based, so they do not loose definition when resized.

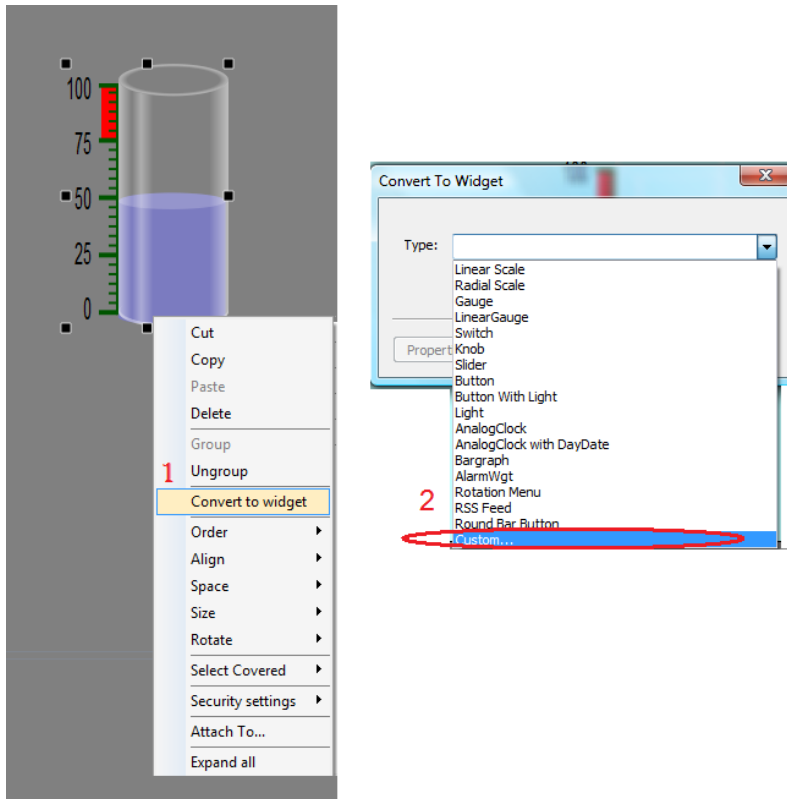
You can, however, modify any of the pre-defined widgets to create your own custom widget. Custom widgets can be made up of several elements only including the properties needed to their purpose.


---

<b>Creating a custom widget .....</b>	<b>240</b>
<b>Adding properties to a custom widget .....</b>	<b>240</b>
<b>Editing custom widgets properties .....</b>	<b>242</b>

# Creating a custom widget

1. Drag and drop on a page all the widget you want to use to compose your custom widget.
2. Select and group them.
3. Right-click on the grouped object and select **Convert To Widget**: the **Convert to Widget** dialog is displayed.



 Note: This dialog shows widget types defined in the gallery, not the types that are specifically created for a project.

4. Select an existing category or **Custom** to create your own.
5. If you create your own, assign a name to it.

## Using widgets components

Widgets are usually made up of many parts, for example a button is a complex widget including two image widgets, a button widget and label.

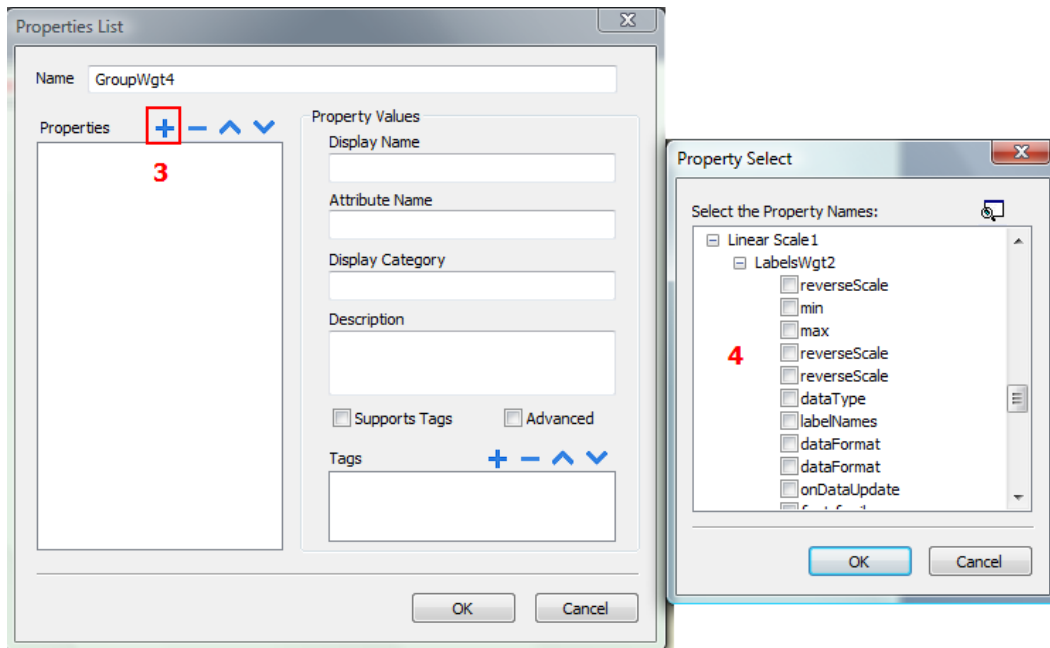
To display a list of all the elements that are part of a widget, select the widget and open the **ObjectView** pane: all the element making up a complex widget are listed in hierarchical order.

To select a single widget without ungrouping the complex widget, select it directly from the **ObjectView** pane.

## Adding properties to a custom widget

When you create a custom widget, you need to define the properties that will be displayed for it in the **Properties** pane.

1. Right-click on the grouped object and select **Custom Properties**: the **Properties List** dialog is displayed.
2. Click **+** to open the **Property Select** dialog: this lists all the properties of all the grouped widgets.
3. Select the properties you want to define for your custom widget.



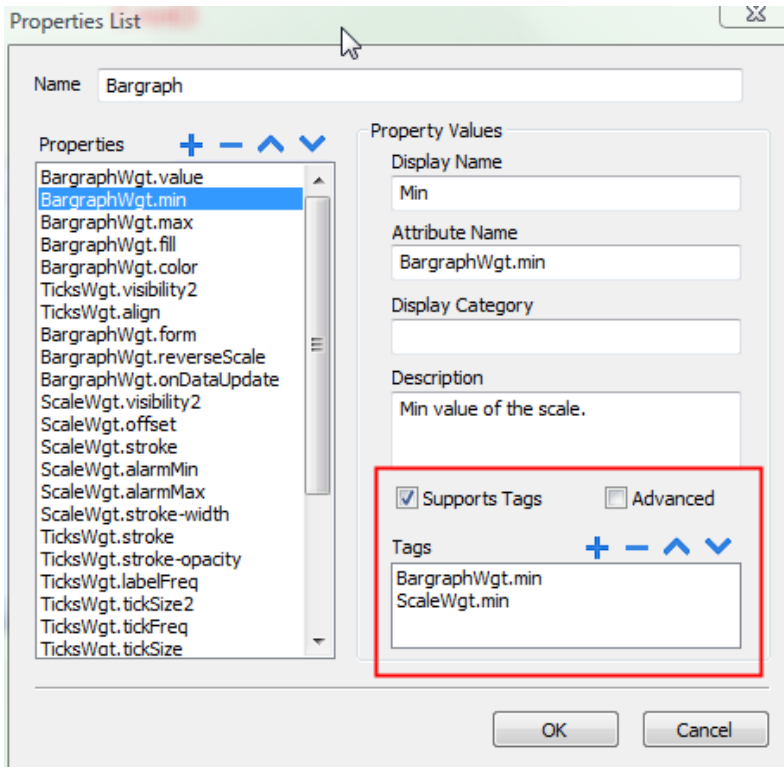
4. Define each property's details.

Parameter	Description
<b>Display Name</b>	Name shown in the <b>Properties</b> pane.
<b>Attribute Name</b>	<p>The name exposed by PB610-B Panel Builder 600, to JavaScript functions and Attach Tag dialog. The default property name format is <b>WidgetType.name</b>, where <b>WidgetType</b> is the type of widget; and <b>name</b> is the attribute name.</p> <p>If you have more than one widget of the same type, the widget type name will be WidgetType01, WidgetType02, an so on.</p>
<b>Display Category</b>	<p>The category or group of the property in the <b>Properties</b> pane.</p> <p>All properties in the same category are shown together, this allows you to organize the properties in the pane (for example, you can declare position properties, such as X coordinate, height, width properties in a single display category called Position).</p>
<b>Description</b>	Any comment on the property to be displayed in the <b>Properties</b> pane.
<b>Advanced</b>	Specifies whether each property should appear in the advanced, or in the simple view mode of the <b>Properties</b> pane.
<b>Support Tags</b>	Specifies if the property supports the "Attach to" attribute.
<b>Tags</b>	Internal tag name for the widget. Typically this is the same as the attribute name; however, you can assign a different attribute name for your custom widget. The tag list is also used to combine tags.

## Combining properties

To combine two or more properties:

1. Select the primary property in the **Properties List** dialog.
2. Click **+**: the **Property Select** dialog is displayed.
3. Select the properties you want to combine.



Note: The dialog only shows the properties that can be combined.

4. Click **OK**: the combined attributes will be shown in the **Tags** list box.

Use the up or down buttons to rearrange the order of the properties and click - to delete it

Select a property to display its details in the dialog box.

## Editing custom widgets properties

To change the properties of a custom widget :

1. Right-click on the widget and select **Custom Properties**: the **Properties List** dialog is displayed.
2. Modify all the properties you need.
3. Click **OK** to confirm.

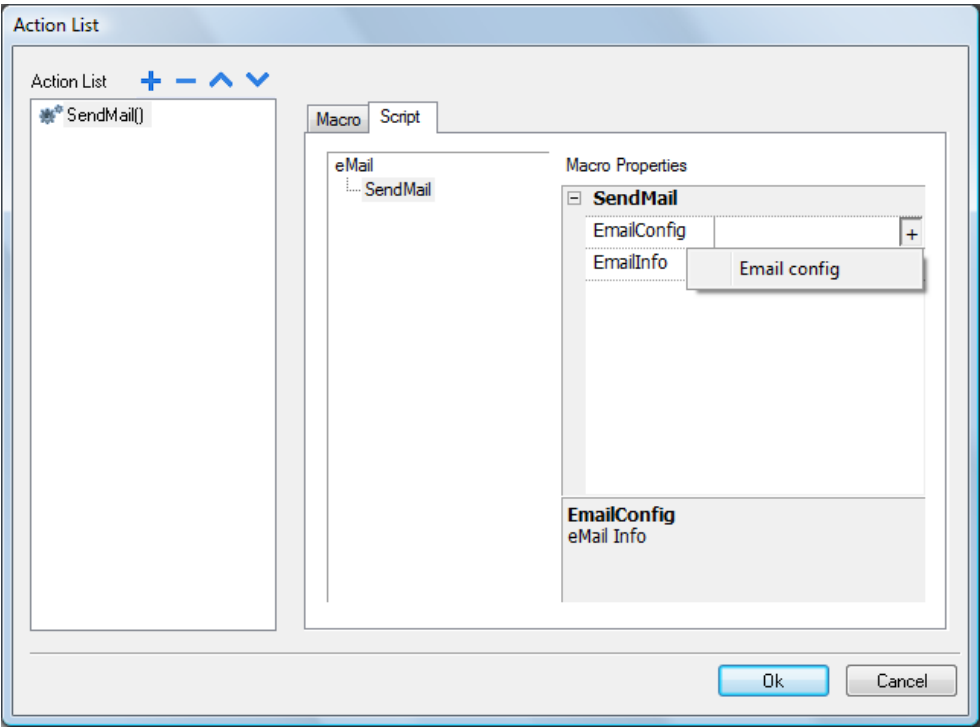
See ["Adding properties to a custom widget" on page 240](#) for details.

# 30    Sending an email message

---

Send emails using the SendMail action, including tags in the email body and attachments.

The SendMail action has been created for working with alarms and schedulers but can be triggered and executed by many other events.



---

<b>Configuring the email server .....</b>	<b>244</b>
<b>Configure emails .....</b>	<b>244</b>

## Configuring the email server


To configure the email server, enter the following information for the **EmailConfig** setting:

Parameter	Description
<b>SMTP Address</b>	SMTP server address.
<b>Server Port</b>	Port for SMTP server connection (default = 25).
<b>Require Auth</b>	Select if the SMTP server requires authentication.
<b>User Name</b>	Username for sending mail using SMTP server.
<b>Password</b>	Password for sending mails using SMTP server.
<b>Encryption</b>	Encryption type (none or SSL).

Click **+** to add more email servers.

## Configure emails

Enter the following information for the **EmailInfo** setting:

Parameter	Description
<b>Name</b>	Optional, this information is only for the log.
<b>Description</b>	Optional, this information is only for the log.
<b>From</b>	Optional, sender email address (for example, John@domain.com).
<b>To</b>	Recipient e-mail addresses. To enter multiple addresses, separate them with a semi-colon.
<b>Subject</b>	Subject of email.
<b>Attachment</b>	<p>Path of the file to be sent as attachment. Only one attachment at a time can be sent.</p> <p> Note: The maximum size of the attachments is usually set by the SMTP server.</p>
<b>Body</b>	<p>Main content of the email. Here you can insert live tags if you include them in square brackets.</p> <p>For example, a message body as "Tag1 value is [Tag1]", will be sent as "Tag1 value is 45", if the current value of Tag1 is 45.</p>

Attach a string tag to the **From**, **To** and **Subject** fields so that their value can be changed in the HMI Runtime.



**WARNING:** The maximum size for the message body is 4096 bytes, the exceeding text will be truncated.



## Adding email templates

Click **+** to add more templates.

Emails

Drafts

+ - ^ v

eMail1

Name

Name

Description

Description

From

Edit value

To

Edit value

Subject

Edit value

Attachment

Message

OK

Cancel



# 31 JavaScript

---

The purpose of this section is to describe how JavaScript is used in the PB610-B Panel Builder 600 applications, not to explain the JavaScript language.

PB610-B Panel Builder 600 JavaScript is based on the ECMAScript programming language <http://www.ecmascript.org>, as defined in standard ECMA-262.

If you are familiar with JavaScript, you can use the same type of commands in PB610-B Panel Builder 600 as you do in a web browser. If you are not familiar with the ECMAScript language, refer to:

<https://developer.mozilla.org/en/JavaScript>

---

<b>JavaScript editor</b> .....	<b>249</b>
<b>Execution of JavaScript functions</b> .....	<b>249</b>
<b>Events</b> .....	<b>251</b>
<b>Widget events</b> .....	<b>252</b>
<b>Page events</b> .....	<b>254</b>
<b>System events</b> .....	<b>255</b>
<b>Objects</b> .....	<b>257</b>
<b>Widget class objects</b> .....	<b>257</b>
<b>Widget properties</b> .....	<b>258</b>
<b>Widget methods</b> .....	<b>260</b>
<b>Page object</b> .....	<b>262</b>
<b>Page object properties</b> .....	<b>262</b>
<b>Page object methods</b> .....	<b>263</b>
<b>Group object</b> .....	<b>265</b>
<b>Group object methods</b> .....	<b>265</b>
<b>Project object</b> .....	<b>266</b>
<b>Project object properties</b> .....	<b>266</b>
<b>Project object methods</b> .....	<b>266</b>
<b>State object</b> .....	<b>275</b>
<b>State object methods</b> .....	<b>276</b>
<b>Keywords</b> .....	<b>277</b>
<b>Global functions</b> .....	<b>277</b>
<b>Handling read/write files</b> .....	<b>278</b>

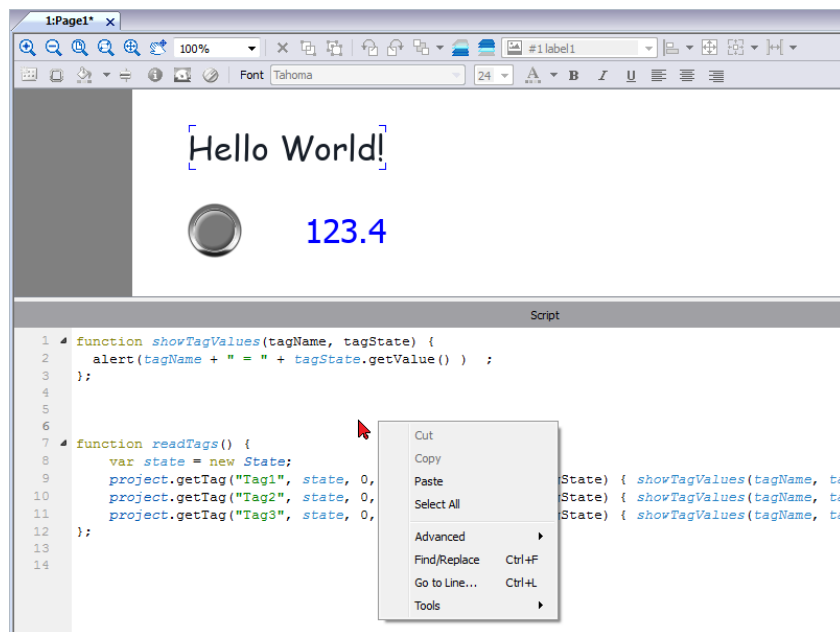
---

<b>Limitations in working with widgets in JavaScript .....</b>	<b>280</b>
<b>Debugging of JavaScript .....</b>	<b>281</b>

## JavaScript editor

PB610-B Panel Builder 600 includes a powerful JavaScript editor.

Right-click in the editor to display available commands.



## Execution of JavaScript functions

JavaScript functions are executed when events occur. For example, a user can define a script for the OnMouseClicked event and the JavaScript script will be executed when the button is pressed on the HMI device.

JavaScript functions are executed only when the programmed event occurs and not cyclically. This approach minimizes the overhead required to execute logic in the HMI device.

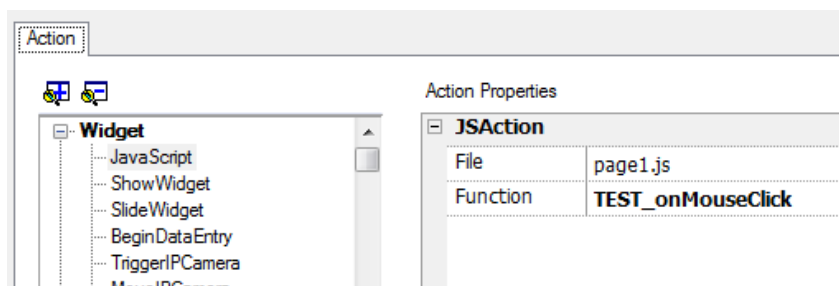
PB610-B Panel Builder 600 provides a JavaScript engine running on the client side. Each project page can contain scripts having a scope local to the page where they are added; global scripts can be created to be executed by scheduler events or alarm events.

In both cases scripts are executed on the client. This means that if more than one client is connected to the HMI device (for external computer running the HMI Client), each client will run the same script, providing different output results depending on the input, since inputs provided to different clients may be different.

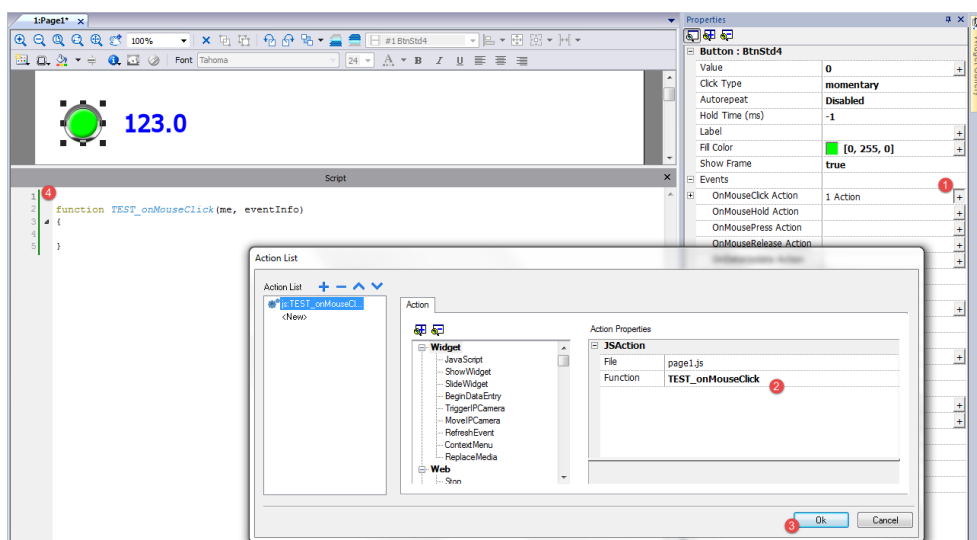
For example, if a script acts according to the position of a slider and this position is different on the different clients, the result of the script will be different on each client.

### JavaScript functions for page events

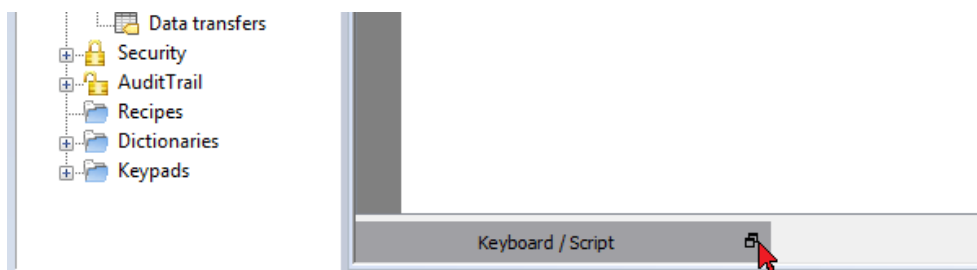
JavaScript editor will open when you add a JavaScript action inside an action list.



1. Select the event that will execute the action.
2. Add a **JavaScript** action from the **Widget** category.
3. Either leave the default function name, or type a new one.
4. Click **OK** to confirm: the JavaScript editor displays your function structure.



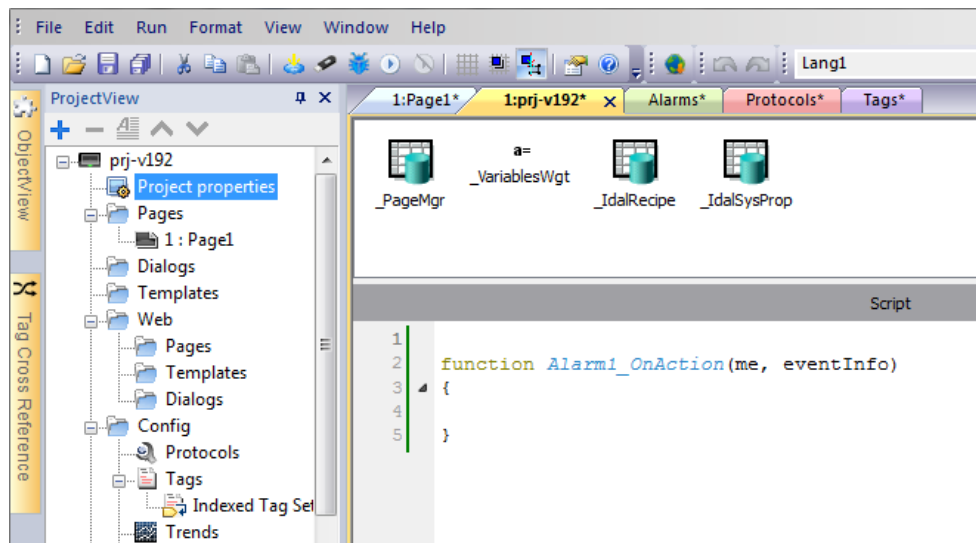
You can also open the JavaScript editor from the **Script** tab at the bottom of the workspace.



## JavaScript functions for alarms and scheduled events

JavaScript code associated with alarms and scheduled events and not associated with a specific page, can be edited from the main **Project properties** page.

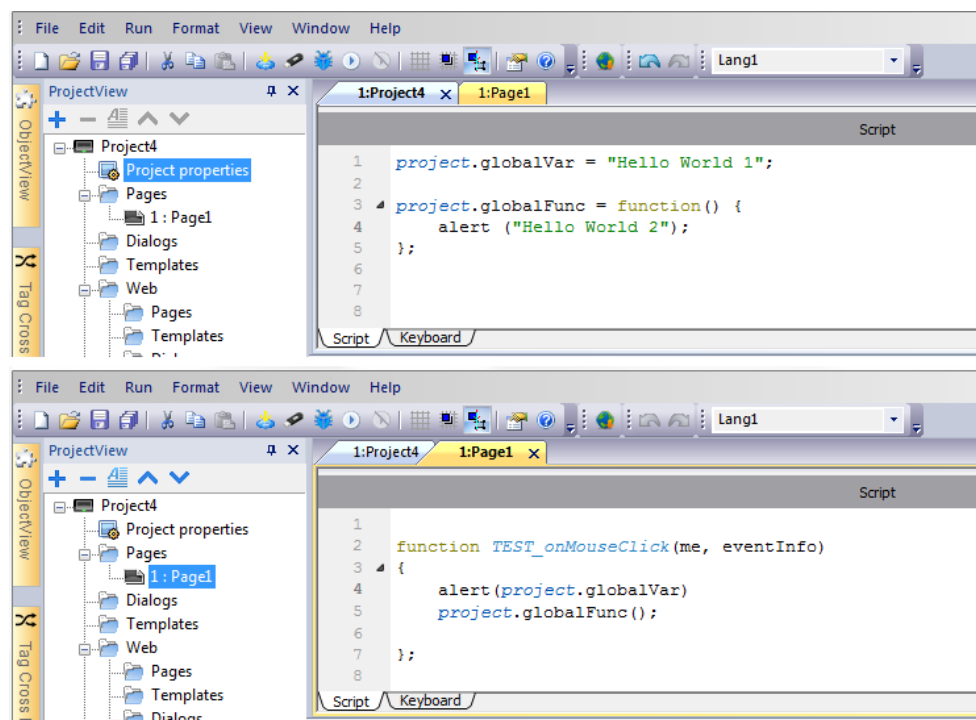
**Path:** *ProjectView* > double-click *Project properties*



Note: JavaScript actions are client actions so they are executed only when a client is logged in.

## Shared JavaScript code

The **project** global variable can be used to share JavaScript code between the pages. Variables are created/initialized from the main JavaScript code from the main **Project properties** page and can then be used from the project pages.



## Events

You can add JavaScript to the following categories of events:

- Widget events
- Page events
- System events

For events of type:

- OnMousePress
- OnMouseRelease
- OnMouseClicked
- OnWheel

JavaScript **eventInfo** parameter contains the following additional properties:

Parameter	Description
<b>eventInfo.posX</b>	Local mouse/touch X coordinate with respect to widget coordinates
<b>eventInfo.posY</b>	Local mouse/touch Y coordinate with respect to widget coordinates
<b>eventInfo.pagePosX</b>	Page X mouse/touch coordinate
<b>eventInfo.pagePosY</b>	Page Y mouse/touch coordinate
<b>eventInfo.wheelDelta</b>	Mouse wheel delta. Integer value with sign representing the rotation direction.  The actual value is the rotation amount in eighths of a degree. The smallest value depends on the mouse resolution. Typically this is 120, corresponding to 15 degrees.

## Widget events

### onMouseClicked

```
void onMouseClick (me, eventInfo)
```

This event is available only for buttons and it occurs when the button is pressed and released quickly.

Parameter	Description
<b>me</b>	Object triggering the event
<b>eventInfo</b>	Details of triggered event

```
function buttonStd1_onMouseClicked(me, eventInfo) {
    //do something...
}
```

### onMouseHold

```
void onMouseHold (me, eventInfo)
```



This event is available only for buttons and it occurs when the button is pressed and released after the number of seconds set as **Hold Time** in the widget properties.

Parameter	Description
<b>me</b>	Object triggering the event
<b>eventInfo</b>	Details of triggered event

```
function buttonStd1_onMouseHold(me, eventInfo) {
    //do something...
}
```

## onMousePress

```
void onMousePress(me, eventInfo)
```

This event is available only for buttons and it occurs when the button is pressed.

Parameter	Description
<b>me</b>	Object triggering the event
<b>eventInfo</b>	Details of triggered event

```
function buttonStd1_onMousePress(me, eventInfo) {
    //do something...
}
```

## onMouseRelease

```
void onMouseRelease (me, eventInfo)
```

This event is available only for buttons and it occurs when the button is released.

Parameter	Description
<b>me</b>	Object triggering the event
<b>eventInfo</b>	Details of triggered event

```
function buttonStd1_onMouseRelease(me, eventInfo) {
    //do something...
}
```

## onDataUpdate

```
boolean onDataUpdate (me, eventInfo)
```

This event occurs when data attached to the widget changes.

Parameter	Description
<b>me</b>	Object triggering the event
<b>eventInfo</b>	<p>An object with the fields listed below (you can refer fields using “.” - dot notation)</p> <p><b>oldValue</b> = Widget value before the change</p> <p><b>newValue</b> = Value which will be updated to the widget</p> <p><b>attrName</b> = Attribute on which the event is generated</p> <p><b>index</b> = Integer attribute index if any, default = 0</p> <p><b>mode</b> = W when the user is writing to the widget. R in all others status.</p>

The event is triggered before the value is passed to the widget, this means the JavaScript code can modify the value before it is actually passed to the widget.

The code can terminate with a return true or return false. After terminating the code with return false, control is returned to the calling widget that may launch other actions.

After terminating the code with true, the control is not returned to the widget and this makes sure that no additional actions are executed following the calling event.

```
function buttonStd1_onDataUpdate(me, eventInfo) {
  if ( eventInfo.oldValue < 0) {
    //do something...
  }
  return false;
}
```

## Page events

### onActivate

```
void onActivate( me, eventInfo )
```

This event occurs each time the page is displayed.

Parameter	Description
<b>me</b>	Object triggering the event
<b>eventInfo</b>	Reserved for future use

JavaScript will be executed when the page is active, that is when the page is loaded.

```
function Page1_onActivate(me, eventInfo) {
  //do something...
}
```

## onDeactivate

```
void onDeactivate( me, eventInfo )
```

This event occurs when leaving the page.

Parameter	Description
<b>me</b>	Object triggering the event
<b>eventInfo</b>	Reserved for future use

```
function Page1_onDeactivate(me, eventInfo) {
    //do something...
}
```

## onWheel

```
void onMouseWheelClock( me, eventInfo )
```

This event occurs when a wheel device is moving (for example, a mouse wheel).

Parameter	Description
<b>me</b>	Object triggering the event
<b>eventInfo</b>	Details of triggered event

```
function Page1_onMouseWheelClock(me, eventInfo) {
    //do something...
}
```

# System events

System events can be related to:

- scheduler
- alarms
- a wheel device

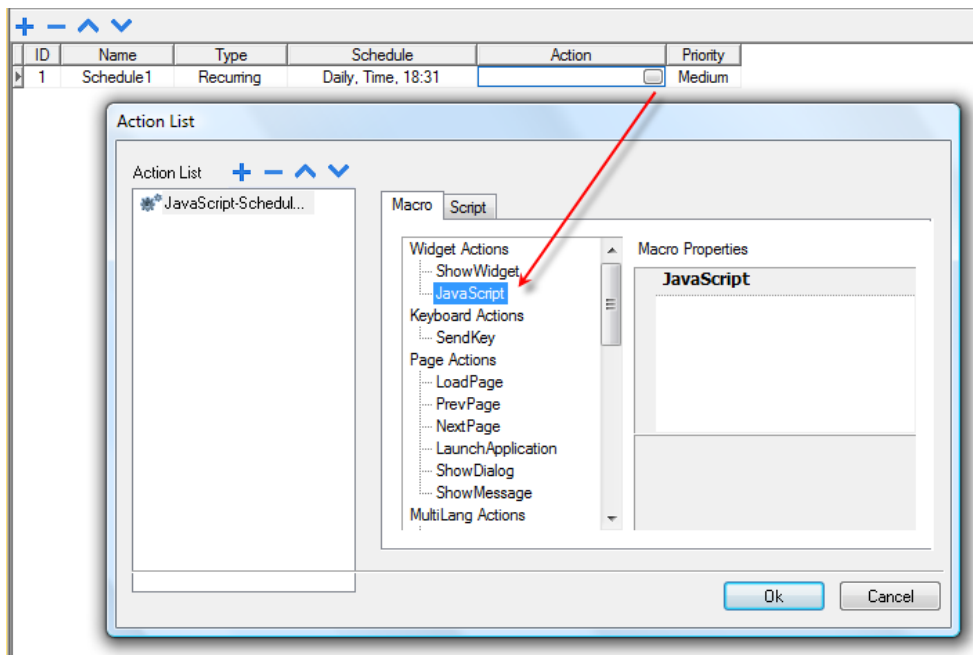


**Important: Make sure you do not duplicate JavaScript function names at page and project level. When a conflict happens, that is two functions with the same name in current page and at project level, the system execute the JavaScript callback at page level.**

When a JavaScript callback is not found in the current page, the system automatically searches for it at project level.

## Scheduler events

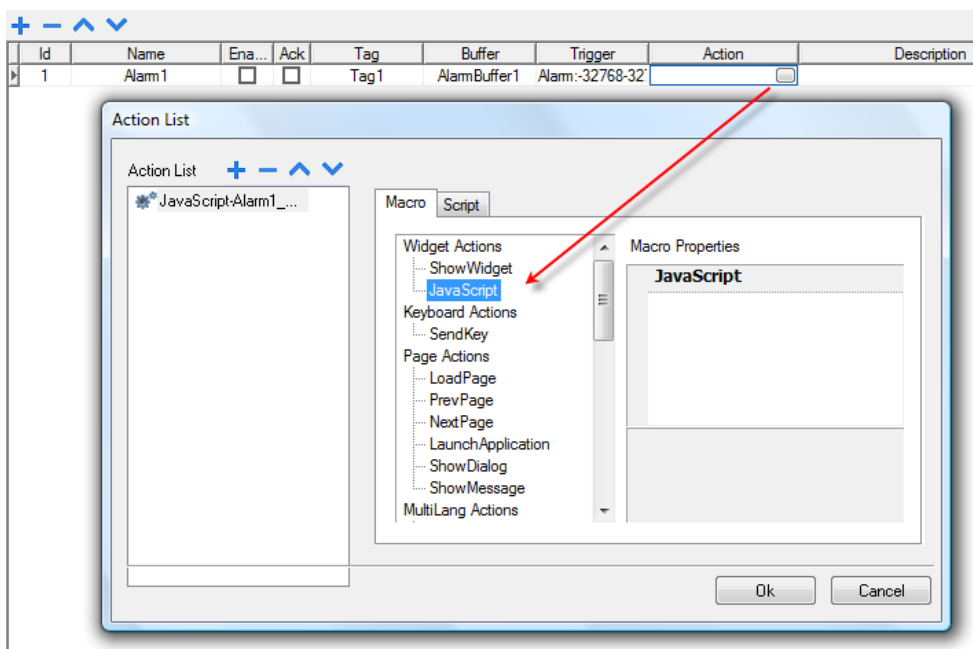
These events occur when triggered by the associated action in the scheduler.



You can edit the JavaScript from the **Project Properties** tab.

## Alarm events

These events occur when triggered by the associated alarm condition.



You can edit the JavaScript from the **Project Properties** tab.

## onWheel

```
void onMouseWheelClock( me, eventInfo )
```

This event occurs when a wheel device is moving (for example, a mouse wheel).

Parameter	Description
<b>me</b>	Object triggering the event
<b>eventInfo</b>	Details of triggered event

```
function Project1_onMouseWheelClock(me, eventInfo) {
    //do something...
}
```

## Objects

PB610-B Panel Builder 600 uses JavaScript objects to access the elements of the page. Each object is composed of properties and methods that are used to define the operation and appearance of the page element. The following objects are used to interact with elements of the HMI device page:

Object	Description
Widget	This is the base class for all elements on the page including the page element
Page	This object references the current HMI device page. The page is the top-level object of the screen.
Group	This object associates a set of tags to allow uniform operation on a set of logically connected tags
Project	This object defines the project widget. The project widget is used to retrieve data about the project such as tags, alarms, recipes, schedules, tags and so on. There is only one widget for the project and it can be referenced through the project variable.
State	This object is the class holding the state of a variable acquired from the controlled environment. Beside the value itself, it contains the timestamp indicating when the value was collected and flags marking the quality of the value.

## Widget class objects

The Widget class is the base class for all the elements on a page including the page element.

Widget, in this case, is not used to indicate a specific screen object but a JavaScript class.

### Changing widget properties with JavaScript

If you want to change the properties of widgets with JavaScript set the widget property **Static Optimization** to **Dynamic**.



**Important:** If the widget property **Static Optimization** is not set to **Dynamic**, changes to properties will be ignored.

Whenever a call to `getWidget` fails, the remote debugger reports the following error:

*"Trying to access static optimized widget "label1". Disable widget static optimization to access widget from script."*

This error is visible also using following code fragment:

```
var wgt;
try {
  wgt = page.getWidget('label1');
} catch(err) {
  alert("" + err);
}
```

## Widget properties

Some properties are common to all widgets.

### objectName

string objectName

Gets the name of the widget, a unique id.

```
function btnStd04_onMouseRelease(me) {
  var wgt = page.getWidget("rect1");
  var name = wgt.objectName;
}
```

### x

number x

Gets or sets the widget 'x' position in pixels.

```
function btnStd1_onMouseRelease(me) {
  var wgt = page.getWidget("rect1");
  wgt.x = 10;
}
```

### y

number y

Gets or sets the widget 'y' position in pixels.

```
function btnStd1_onMouseRelease(me) {  
    var wgt = page.getWidget("rect1");  
    wgt.y = 10;  
}
```

## width

number width

Gets or sets the widget width in pixels.

```
function btnStd1_onMouseRelease(me) {  
    var wgt = page.getWidget("rect1");  
    wgt.width = 10;  
}
```

## height

number height

Gets or sets the widget height in pixels.

```
function btnStd1_onMouseRelease(me) {  
    var wgt = page.getWidget("rect1");  
    wgt.height = 10;  
}
```

## visible

boolean visible

Gets or sets the widget visible state.

```
function btnStd4_onMouseRelease(me) {  
    var wgt = page.getWidget("rect1");  
    wgt.visible = false;  
}  
  
function btnStd5_onMouseRelease(me) {  
    var wgt = page.getWidget("rect1");  
    wgt.visible = true;  
}
```

## value

number value

Gets or sets the widget value.

```
function btnStd6_onMouseRelease(me) {  
    var wgt = page.getWidget("field1");  
    wgt.value = 100;  
}
```

## opacity

number opacity (range from 0 to 1)

Gets or sets the widget opacity. Values are decimals from 0 to 1, where 1 is 100% opaque.

```
function btnStd8_onMouseRelease(me) {  
    var wgt = page.getWidget("rect1");  
    wgt.opacity = 0.5;  
}
```

## rotation

number rotation (in degrees)

Gets or sets the rotation angle for the widget. The rotation is done clockwise and by degrees, starting at the East position.

```
function btnStd9_onMouseRelease(me) {  
    var wgt = page.getWidget("rect1");  
    wgt.rotation = 45;  
}
```

## userValue

string userValue

Gets or sets a user-defined value for the widget. This field can be used by JavaScript functions to store additional data with the widget.

```
function btnStd9_onMouseRelease(me) {  
    var wgt = page.getWidget("rect1");  
    wgt.userValue = "Here I can store custom data";  
}
```

Every widget has some specific properties that you can access using dot notation. For an up-to-date and detailed list of properties you can use the JavaScript Debugger inspecting the widget methods and properties.

# Widget methods

Some methods are common to all widgets.



## getProperty

object getProperty( propertyName, [index] )

Returns a property.

Parameter	Description
<b>propertyName</b>	String containing the name of property to get
<b>index</b>	Index of the element to get from the array (default = 0)

Almost all properties that are shown in the PB610-B Panel Builder 600 **Properties** pane can be retrieved using the `getProperty` method. The index value is optional and only used for widgets that support arrays.

```
function buttonStd1_onMouseRelease(me, eventInfo) {
    var shape = page.getWidget("rect2");
    var y_position = shape.getProperty("y");
}
```

```
function buttonStd2_onMouseRelease(me, eventInfo) {
    var image = page.getWidget("multistate1");
    var image3 = image.getProperty("imageList", 2);
    //...
}
```

## setProperty

boolean setProperty( propertyName, value, [index] )

Sets a property for the widget.

### Parameters

Parameter	Description
<b>propertyName</b>	String containing the name of property to set
<b>value</b>	String containing the value to set the property.
<b>index</b>	Index of the element to set in the array (default = 0)

Almost all properties that are shown in the PB610-B Panel Builder 600 **Properties** pane can be set by this method. The index value is optional and only used for Widgets that support arrays (for example, a MultiState Image widget). The `setProperty` method returns a boolean value (true or false) to indicate if the property was set or not.

```
function buttonStd1_onMouseRelease(me, eventInfo) {
    var setting_result = shape.setProperty("y", 128);
    if (setting_result)
        alert("Shape returned to start position");
}
```

```

}

function buttonStd2_onMouseRelease(me, eventInfo) {
    var image = page.getWidget("multistate1");
    var result = image.setProperty("imageList", "Fract004.png", 2);
    //...
}

```

## Page object

This object references the current HMI device page. The page is the top-level object of the screen.

## Page object properties

Properties available at page level.

### backgroundColor

string backgroundColor (in format rgb(xxx, xxx, xxx) where xxx range from 0 to 255)

Page background color.

```

function btnStd11_onMouseRelease(me) {
    page.backgroundColor = "rgb(128,0,0)";
}

```

### width

number width

Page width in pixels.

```

function btnStd05_onMouseRelease(me) {
    var middle_x = page.width / 2;
}

```

### height

number height

Page height in pixels.

```

function btnStd05_onMouseRelease(me) {
    var middle_y = page.height / 2;
}

```

## userValue

string userValue

Gets or sets a user-defined value for the widget. This field can be used by JavaScript functions to store additional data with the page.

```
function btnStd9_onMouseRelease(me) {  
    page.userValue = "Here I can store custom data";  
}
```

## Page object methods

Methods that can be used at page level.

### getWidget

object getWidget( wgtName )

Returns the widget with the given name.

Parameter	Description
<b>wgtName</b>	String containing the widget name

### Return value

An object representing the widget. If the widget does not exist, null is returned.

```
function btnStd1_onMouseRelease(me) {  
    var my_button = page.getWidget("btnStd1");  
}
```

### setTimeout

number setTimeout( functionName, delay )

Starts a timer to call a given function after a given delay.

Parameter	Description
<b>functionName</b>	String containing the name of function to call
<b>delay</b>	Delay in milliseconds

### Return value

A number corresponding to the timerID.

```
var duration = 3000;
```

```
var myTimer = page.setTimeout("innerChangeWidth()", duration);
```

## clearTimeout

```
void clearTimeout( timerID )
```

Stops and clears the timeout timer with the given timer.

Parameter	Description
timerID	Timer to be cleared and stopped

```
var duration = 3000;
var myTimer = page.setTimeout("innerChangeWidth()", duration);
// do something
page.clearTimeout(myTimer);
```

## setInterval

```
number setInterval( functionName, interval )
```

Starts a timer that executes the given function with the given interval.

Parameter	Description
functionName	String containing the name of function to call
interval	Interval in milliseconds

### Return value

A number corresponding to the timerID.

```
var interval = 3000;
var myTimer = page.setInterval("innerChangeWidth()", interval);
```

## clearInterval

```
void clearInterval( timerID )
```

Stops and clears the interval timer with the given timer.

Parameter	Description
timerID	Timer to be cleared and stopped

```
var interval = 3000;
var myTimer = page.setInterval("innerChangeWidth()", interval);
// do something
```

```
page.clearInterval(myTimer);
```

## clearAllTimeouts

```
void clearAllTimeouts()
```

Clears all the timers started.

```
page.clearAllTimeouts();
```

# Group object

A group is a basic logical element that associates a set of logical tags.

## Group object methods

Methods that can be used with group objects.

### getTag

```
object getTag( TagName )
```

Gets the tag specified by TagName from the group object.

Parameter	Description
TagName	String representing the tag name

### Return value

An object that is the value of the tag or, if tag value is an array, the complete array. If you need to retrieve an element of the array, check the method `getTag` available in the project object. Undefined is returned if tag is invalid.

```
var group = new Group();
project.getGroup("GroupName", group);
var value = group.getTag("Tag1");
```

### getCount

```
number getCount()
```

Returns total number of tags in this group.

```
var group = new Group();
project.getGroup("GroupName", group);
var value = group.getCount();
```

## getTags

object getTags()

Returns the list of all tags in group.

```
function {  
  var group = new Group();  
  project.getGroup("enginesettings", group);  
  var tagList = group.getTags();  
  for(var i = 0; i < tagList.length; i++){  
    var tagName = tagList[i];  
    //do something...  
  };  
};
```

# Project object

This object defines the project widget. The project widget is used to retrieve data about the project such as tags, alarms, recipes, schedules, tags and so on. There is only one widget for the project and it can be referenced through the project variable.

## Project object properties

Properties to be set at project level.

### startPage

string startPage

Page shown when the project is started.

```
var startPage = project.startPage;  
project.startPage = "Page2.jmx";
```

## Project object methods

Methods to be used at project level.

### nextPage

void nextPage()

The script executes the Next page action.

```
project.nextPage();
```

## prevPage

```
void prevPage()
```

The script executes the previous page action.

```
project.prevPage();
```

## homepage

```
void homePage()
```

The script executes the Home page action.

```
project.homePage();
```

## loadPage

```
void loadPage(pageName)
```

The script executes to load the set page defined in the script.

```
project.loadPage("Page5.jmx");
```

## showDialog

```
void showDialog(pageName)
```

The script executes to show the dialog page.

```
project.showDialog("Dialog.jmx");
```

## closeDialog

```
void closeDialog()
```

The script executes to close the currently-opened dialog page.

```
project.closeDialog();
```

## showMessage

```
void showMessage( message )
```

The script executes to display the message popup.

```
project.showMessage("Hi This is test message");
```

## getGroup

```
number getGroup( groupName, groupInstance, [callback] )
```

Fast read method; this gets the values of all tags in a group.

Parameter	Description
<b>groupName</b>	String containing the name of the group
<b>groupInstance</b>	Group element to be filled
<b>callback</b>	String containing the name of the function to be called when the group is ready

### Return value

A number value that is the status: 1 for success, 0 for fail.

```
var group = new Group();
var status = project.getGroup ("enginesettings", group);
if (status == 1) {
    var value = group.getTag("Tag1");
    if (value!=undefined) {
        // do something with the value
    }
}
```

```
var g = new Group();
var status = project.getGroup ("enginesettings", g,
    function (groupName, group) { fnGroupReady(groupName, group);} );

function fnGroupReady(groupName, group) {
    var val = group.getTag("Tag1");
    if (val!=undefined) {
        // do something with the value
    }
}
```

## getTag

object getTag( tagName, state, index, forceRefresh)

```
void getTag( tagName, state, index, callback, forceRefresh)
```

It returns the tag value or the complete array if index value is -1 of the given tagName.

Parameter	Description
<b>tagName</b>	String of tag name
<b>state</b>	State element to be filled



Parameter	Description
<b>index</b>	Index if the tag is of array type. -1 returns the complete array. Default = 0.
<b>callback</b>	Function name if an asynchronous read is required. Default = "".
<b>forceRefresh</b>	(Optional parameter) True = the Runtime will read an updated value of the tag directly from the device. Default is false.

### Return value

Tags value is returned. If tag is array type and index = -1 then the complete array is returned. For non-array tags provide index as 0.

```
var state = new State();
var value = project.getTag("Tag1", state, 0);
//
//for non array type
//tags index is not considered, so can be left as 0
//
if (value!=undefined) {
//...do something with s
}
```

```
var state = new State();
project.getTag("Tag1", state, -1,
    function(tagName, tagState) { fnTagReady(tagName, tagState); });
function fnTagReady(tagName, tagState) {
    if (tagName=="Tag1") {
        var myValue = tagState.getValue();
    }
}
```

### setTag

number setTag( tagName, tagValue, [index], [forceWrite] )

Sets the given tag in the project. Name and value are in strings.

Parameter	Description
<b>tagName</b>	String of tag name
<b>tagValue</b>	Object containing the value to write
<b>index</b>	Index if the tag is of array type. -1 pass the complete array. Default = 0.
<b>forceWrite</b>	Boolean value for enabling force write of tags, the function will wait for the value to be written before it returns back. Default = false.

## Return value

Integer value for denoting success and failure of action when forceWrite is true. 0 means success and -1 means failure. If forceWrite is false, returned value will be undefined.

```
var val = [1,2,3,4,5];
var status = project.setTag("Tag1", val, -1, true);
if (status == 0) {
    // Success
} else {
    // Failure
}
```

```
var val = "value";
project.setTag("Tag1", val);
```

## updateSystemVariables

```
project.updateSystemVariables()
```

Force system variables to refresh.

```
project.updateSystemVariables()
```

## getRecipeItem

```
object getRecipeItem (recipeName, recipeSet, recipeElement)
```

Gets the value of the given recipe set element.

Parameter	Description
<b>recipeName</b>	String representing the recipe name
<b>recipeSet</b>	String representing the recipe set, can be either the recipe set name or 0 based set index.
<b>recipeElement</b>	String representing the recipe Element, can be either the element name or 0 based element index.

## Return value

An object with the value of the recipe. undefined is returned if invalid. If of type array, an array object type is returned.

```
var value = project.getRecipeItem("recipeName", "Set", "Element");
```

## setRecipeItem

```
number setRecipeItem (recipeName, recipeSet, recipeElement, value )
```

Gets the value of the given recipe set element.

Parameter	Description
<b>recipeName</b>	String representing the recipe name
<b>recipeSet</b>	String representing the recipe set, can be either the recipe set name or 0 based set index.
<b>recipeElement</b>	String representing the recipe Element, can be either the element name or 0 based element index.
<b>value</b>	An object containing the value to store in the recipe. It can be an array type.

### Return value

Integer value for denoting success and failure of action. A '0' means success and '-1' means failure.

```
var val = [2,3,4];
project.setRecipeItem("recipeName", "Set", "Element", val);
if (status == 0) {
    // Success
} else {
    // Failure
}
```

### downloadRecipe

void downloadRecipe (recipeName, recipeSet )

Downloads the recipe set to the corresponding tag.

Parameter	Description
<b>recipeName</b>	String representing the recipe name
<b>recipeSet</b>	String representing the recipe set, can be either the recipe set name or 0 based set index.

```
project.downloadRecipe("recipeName", "Set");
```

### uploadRecipe

void uploadRecipe (recipeName, recipeSet )

Uploads the value of tags into the provided recipe set.

Parameter	Description
<b>recipeName</b>	String representing the recipe name
<b>recipeSet</b>	String representing the recipe set, can be either the recipe set name or 0 based set index.

```
project.uploadRecipe("recipeName", "Set");
```

## launchApp

```
void launchApp( appName, appPath, arguments, singleInstance)
```

Executes an external application.

Parameter	Description
<b>appName</b>	String containing the application name
<b>appPath</b>	String containing the application absolute path
<b>Arguments</b>	String containing the arguments to be sent to application
<b>singleInstance</b>	true = only single instance allowed, false = multiple instances allowed

```
project.launchApp("PDF.exe","\\Flash\\QTHMI\\PDF","\\USBMemory\\file.pdf","true");
```

## printGfxReport

```
void printGfxReport( reportName, silentMode)
```

Prints the graphic report specified by reportName.

Parameter	Description
<b>reportName</b>	String containing the report name
<b>silentMode</b>	True = silent mode enabled. No printer settings dialog is displayed.

```
project.printGfxReport("Report Graphics 1", true);
```

## printText

```
void printText( text, silentMode)
```

Prints a fixed text.

Parameter	Description
<b>text</b>	String to print
<b>silentMode</b>	True = silent mode enabled. No printer settings dialog is displayed.

```
project.printText("Hello I Am Text Printing",true);
```

## printBytes

```
void printBytes( text, silentMode)
```

Prints a hexadecimal string representing data to print. For example, "1b30" to print < ESC 0 >

Parameter	Description
<b>text</b>	Hexadecimal string to print
<b>silentMode</b>	True = silent mode enabled. No printer settings dialog is displayed.

```
project.printText("Hello I Am Text Printing",true);
```

## emptyPrintQueue

```
void emptyPrintQueue()
```

Empties the print queue. Current job will not be aborted.

```
project.emptyPrintQueue();
```

## pausePrinting

```
void pausePrinting();
```

Suspends printing operations. Will not suspend the print of a page already sent to the printer.

```
project.pausePrinting();
```

## resumePrinting

```
void resumePrinting();
```

Resumes previously suspended printing.

```
project.resumePrinting();
```

## abortPrinting

```
void abortPrinting();
```

Aborts current print operation and proceed with the next one in queue. This command will not abort the print of a page already sent to the printer.

```
project.abortPrinting();
```

## printStatus

```
project.printStatus;
```

Returns a string representing current printing status.

Status string	Description
error	An error occurred during printing
printing	Ongoing printing
idle	System is ready to accept new jobs
paused	Printing has be suspended

```
var status = project.printStatus;
project.setTag("PrintStatus",status);
```

## printGfxJobQueueSize

```
project.printGfxJobQueueSize;
```

Returns the number of graphic reports in queue for printing.

```
var gfxqueuesize = project.printGfxJobQueueSize;
project.setTag("printGfxJobQueueSize",gfxqueuesize);
```

## printTextJobQueueSize

```
project.printTextJobQueueSize;
```

Returns the number of text reports in queue for printing.

```
var textjobqueuesize = project.printTextJobQueueSize;
project.setTag("printTextJobQueueSize",textjobqueuesize);
```

## printCurrentJob

```
project.printCurrentJob;
```

Returns a string representing current job being printed

```
var currentjob = project.printCurrentJob;
project.setTag("printCurrentJob",currentjob);
```

## printActualRAMUsage

```
project.printActualRAMUsage;
```

Returns an estimate of RAM usage for printing queues

```
var myVar = project.printActualRAMUsage;
alert(" actual ram usage is    "+ myVar);
```

## printRAMQuota

```
project.printRAMQuota;
```

Returns the maximum allowed RAM usage for printing queues

```
var ramquota = project.printRAMQuota;  
project.setTag("printRAMQuota", ramquota);
```

## printActualDiskUsage

```
project.printActualDiskUsage;
```

Returns the spool folder disk usage (for PDF printouts)

```
var myVar1 = project.printActualDiskUsage;  
alert(" actual disk usage is " + myVar1);
```

## printDiskQuota

```
project.printDiskQuota;
```

Returns the maximum allowed size of spool folder (for PDF printouts).

```
var ramquota = project.printRAMQuota;  
var diskquota = project.printDiskQuota;
```

## printSpoolFolder

```
project.printSpoolFolder;
```

Returns current spool folder path (for PDF printouts).

```
var spoolfolder = project.printSpoolFolder;  
project.setTag("printSpoolFolder", spoolfolder);
```

## printPercentage

```
project.printPercentage;
```

Returns current job completion percentage (meaningful only for multipage graphic reports)

```
var percentage = project.printPercentage;  
project.setTag("printPercentage", percentage);
```

# State object

This is the class holding the state of a tag acquired from the controlled environment.

# State object methods

Methods to be used with state objects.

## getQualityBits

number getQualityBits()

Returns an integer - a combination of bits indicating tag value quality.

```
var state = new State();
var value = project.getTag("Tag1", state, 0);
var qbits = state.getQualityBits();
```

## getTimestamp

number getTimestamp()

Returns time the value was sampled.

### Return value

A number containing the timestamp (for example 1315570524492).



Note: Date is a native JavaScript data type.

```
var state = new State();
var value = project.getTag("Tag1", state, 0);
var ts = state.getTimestamp();
```

## isQualityGood

boolean isQualityGood()

Returns whether the value contained in this state object is reliable.

### Return value

A Boolean true if quality is good, false otherwise.

```
var state = new State();
var value = project.getTag("Tag1", state, 0);
if (state.isQualityGood()) {
    // do something...
}
```



## Keywords

Global objects are predefined and can be referenced by the following names.

### page

object page

References the page object for the current page.

```
function btnStd04_onMouseRelease(me) {  
    var wgt = page.getWidget("rect1");  
    var name = wgt.objectName;  
}
```

### project

object project

References the project widget.

```
var group = new Group();  
project.getGroup("GroupName", group);  
var value = group.getCount("Tag1");
```

## Global functions

### print

void print( message )

Prints a message to the HMI Logger window.

Parameter	Description
message	Message string

```
print("Test message");
```

### alert

void alert( message )

Displays a pop-up dialog with the given message. The user must press the **OK** button in the dialog to continue with the execution of the script.

Parameter	Description
message	Message string



Note: The alert function may be used for debugging JavaScript functions.

```
alert("Test message");
```

## Handling read/write files

### Create folder

```
boolean fs.mkdir(strPath);
```

Creates a folder, if not already existing, in the specified path. Returns true on success and false if it fails.

Parameter	Description
<b>strPath</b>	Path string

### Remove folder

```
boolean fs.rmdir(dirPath);
```

Remove directory at strPath if exists and empty. Returns true on success and false if it fails.

Parameter	Description
<b>dirPath</b>	Folder string

### Read folder content

```
object fs.readdir(dirPath);
```

Reads the contents of a folder. Returns an array of the names of the files in the folder excluding '.' and '..'. Returns empty list if it fails.

Parameter	Description
<b>dirPath</b>	Folder string

### Read file

```
object fs.readFile(strfile [,strFlag]);
```

Opens the strFile file in read mode, reads its contents and returns it.

Parameter	Description
<b>strFile</b>	File name string
<b>strFlag</b>	Read file mode: "b" reads and returns as binary file (otherwise returns a text file)

## Write file

```
fs.writeFile(strFile, fileData, [strFlag]);
```

Creates the strFile file if not present. Opens the strFile file in write mode and writes the data fileData to the file.

Parameter	Description
<b>strFile</b>	File name string
<b>fileData</b>	Data to be write on the file in byte array
<b>strFlag</b>	Write file mode: <ul style="list-style-type: none"> <li>• “a”: appends fileData to the end of the text file</li> <li>• “r”: replaces the contents of the file with fileData</li> <li>• “ab”: appends fileData to the end of the binary file</li> <li>• “rb”: replaces the contents of the binary file with fileData</li> </ul>

Default flag is for writing text file in append and write mode. File path will be created if not present.

Returns -1 if write error occurs.

## File exists

```
boolean fs.exists(strPath)
```

Returns true if the file or folder exists at strPath.

Parameter	Description
<b>strPath</b>	Path string

## Remove file

```
boolean fs.unlink(strPath)
```

Removes the given file at strPath from filesystem if exists. Returns true on success and false if it fails.

Parameter	Description
<b>strPath</b>	Path string

## File status

```
object fs.stat(strPath)
```

Retrieves information on the file/folder present at the specified path.

Parameter	Description
<b>strPath</b>	File/folder path string

```
var fileStats = fs.stat(strPath)
```

<code>fileStats.isFile</code>	True if path is a file
<code>fileStats.isDir</code>	True if path is a folder
<code>fileStats.size</code>	Size in bytes of that file
<code>fileStats.atime</code>	Date object representing the last read access time
<code>fileStats.mtime</code>	Date object representing the last write access time
<code>fileStats.ctime</code>	Date object representing the creation time
<code>fileStats.perm</code>	File permissions

If path is invalid both `isFile` and `isDir` fields return false.

### File permission table

0x4000	File is readable by the owner of the file
0x2000	File is writable by the owner of the file
0x1000	File is executable by the owner of the file
0x0400	File is readable by the user
0x0200	File is writable by the user
0x0100	File is executable by the user
0x0040	File is readable by the group
0x0020	File is writable by the group
0x0010	File is executable by the group
0x0004	File is readable by anyone
0x0002	File is writable by anyone

## Important notes on file handling

Path for files and folders are expected to be UNIX style. This means the backslash character (\) is not recognized. Use slash character (/) instead.

File system object is a client side object. So operations are performed on local file system, not on server file system.

Current JavaScript API to get access at the device file system has been designed to manipulate small files. When a file is read, the entire file contents is temporarily stored inside the RAM available for JavaScript environment (16MB) and an exception is raised when there is not enough available memory. Good programming practice is to include the `fs.readFile()` call inside a try/catch block.

## Limitations in working with widgets in JavaScript

Widgets cannot be instantiated by JavaScript, they can only be accessed and changed. If you need additional widgets on the page, you can add hidden widgets on the page, and then display or position them using JavaScript.

# Debugging of JavaScript

PB610-B Panel Builder 600 and HMI Runtime include a JavaScript debugger.

Two types of debuggers are available:

- Runtime debugger: a debugger running directly on the HMI device
- Remote debugger: a debugger running on a remote computer connected to the HMI device via Ethernet (usually computer running PB610-B Panel Builder 600)

## Enabling debugging

In the **Properties** pane of a page, set **JavaScript Debug** to **true**.

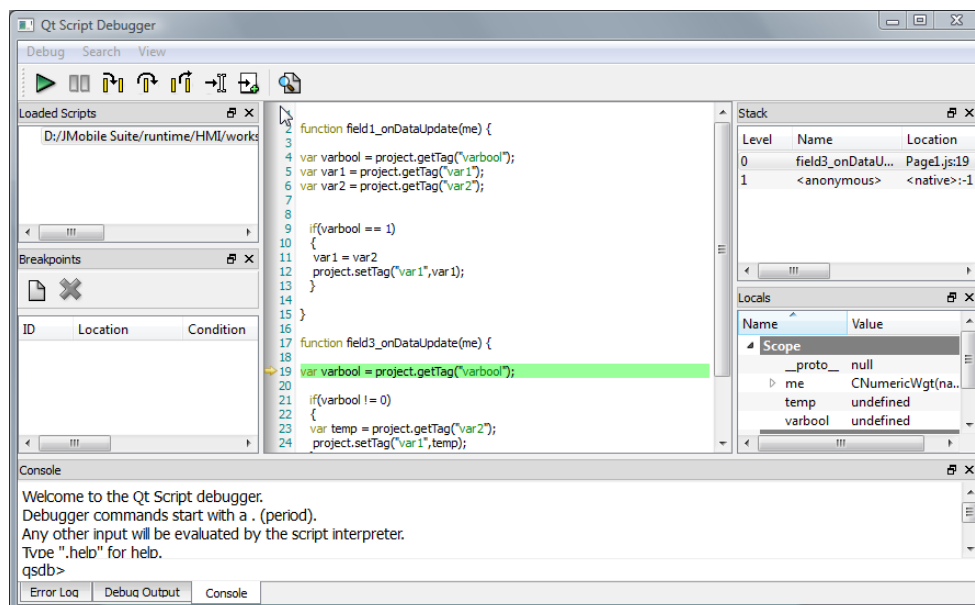
Project Widget	
Id	Project
Full Path	
Version	
Context Menu	on delay
Developer Tools	false
Keyboard	true
JavaScript Debug	true
Allow JavaScript Remote	true

Page	
Id	Page1
Width	1024
Height	768
Background	<input type="checkbox"/> [255, 255, :
Template	none
Static File Type	png
JavaScript Debug	true

For schedulers and alarms debugging, enable JavaScript Debug in Project properties.

In the HMI Runtime, when the events are called, the debugger will show the debug information. In the **Locals** pane you can inspect all variables and elements.



For a complete reference guide about JavaScript Debugger refer to :

<http://qt-project.org/doc/qt-4.8/qtscriptdebugger-manual.html>

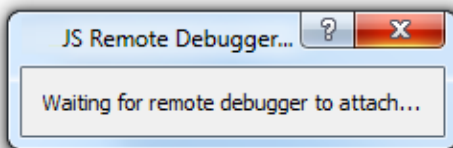


Note: For UN20 HMI devices (Windows CE MIPS HMI devices), the local debugger has been disabled. However, remote debugger is available for JavaScript debugging from a computer connected to HMI device via Ethernet.

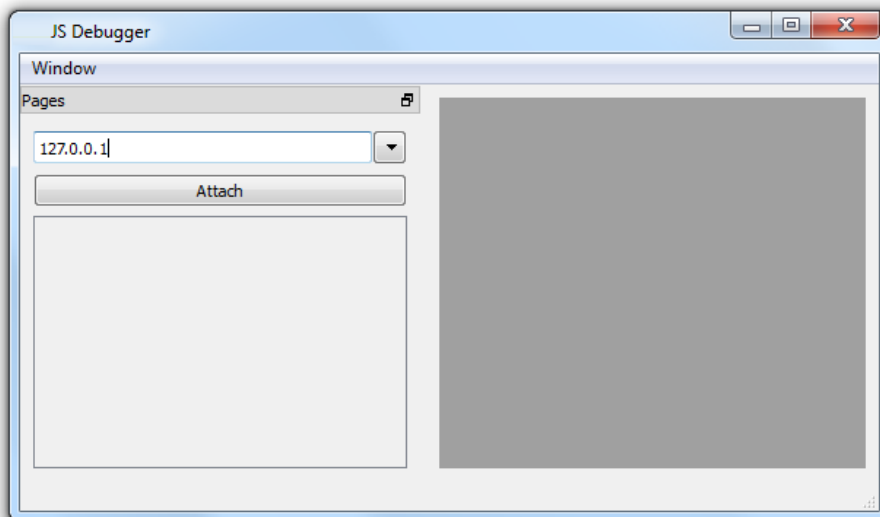
## Remote JavaScript Debugger

Path: **Run> Start JS Remote Debugger**

1. Set the **Allow JavaScript Remote** and the **JavaScript Debug** parameters in the project Properties to true in all the pages where debugging is required.
2. Download the project: the following message is displayed on the runtime.



3. In the **JS Debugger** window, select the IP of the HMI device and click **Attach** to connect the debugger to the HMI device.



Remote JavaScript debugger connects to HMI Runtime using port 5100/TCP.



Note: The Remote JavaScript debugger tool is not supported in HMI Client.

## JavaScript Memory Usage

When the memory exceeds the maximum, an out of memory exception is thrown with a custom message. Please note that we don't have a fine control over the actual memory usage so it is mainly a soft limit. Moreover we can't forbid the allocation (this will break the engine implementation), so exception is thrown only when the memory is already over the limit. Before raising the exception, a garbage collection is forced to see if some memory can be freed.

JavaScript memory limit can be accessed from the global object **\$EngineMemory**. The default is 16MB, which should be enough for the typical JavaScript usage (mainly control, without many allocations).

- `$EngineMemory.setLimit()`  
set maximum memory allowed for JavaScript (the default limit is 0x00FFFFFF)
- `$EngineMemory.getLimit()`  
get maximum memory allowed for JavaScript
- `$EngineMemory.getSize()`  
get currently used memory from JS (fastMallocStat)

### Test memory exception

To generate and test memory exception you can use the following snippet. Please note that we need to reset the memory limit to 0xffffffff to be able to run the alert, otherwise the memory allocations required to pop up the alert would fail.

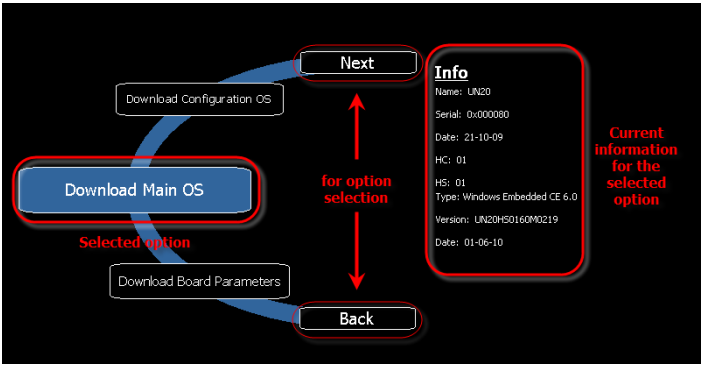
```
try
{
    // Generate out-of-memory error
    var a = [];
    while(1)
    {
        a.push("a");
    };
} catch(e)
{
    // Ensure there is enough memory to pop up error message
    $EngineMemory.setLimit(0xffffffff);
    alert("Exception: " + e);
};
```





# 32 System Settings tool

The System Settings tool includes a rotating menu, and navigation buttons to scroll between the available options.



For each function and component on the let, the **Info** pane on the right displays all available information. In the example the version of the Main OS component is shown.

The System Settings tool can be used in two operating modes:

- User mode
- System mode.


For each mode different options are available.

<b>User Mode</b> .....	<b>285</b>
<b>System Mode</b> .....	<b>286</b>

## User Mode

In User Mode a simplified interface gives users access to the basic settings of the HMI device.

When you access the tool at runtime selecting **Show system settings** from the context menu, the tool is started by default in User Mode.



Note: Press and hold on a screen area without buttons or other touch sensitive elements to display the context menu.

### Elements available in User Mode

Element	Description
Calibrate Touch	Calibrates the touch screen
Display set-tings	Controls backlight and brightness

Element	Description
<b>Time</b>	Sets internal RTC
<b>BSP Settings</b>	Displays operating system version. Sets unit operating timers (buzzer control, battery LED control, etc.)
<b>Network</b>	Sets IP address and other network settings
<b>Plug-in List</b>	Displays a list of the plug-in modules installed and recognized by the system. This option may not be supported by all platforms and versions.

## System Mode

In System Mode a full interface gives users access to all the tool's options.




A special procedure is required to start the tool in System Mode, or when the standard access procedure cannot be used for some reason. Once activated by this special procedure, the System Settings tool always starts in System Mode.

To access System Mode execute a tap sequence on the touch screen during the power-up phase. A tap sequence is a high frequency sequence of touch activations executed immediately after the device has been powered.

### Elements available in System Mode

In addition to those available in User Mode, the following features are also available:

Element	Description
<b>Format Flash</b>	Formats the internal device flash disk. All projects and the HMI Runtime will be erased, returning the device to its factory settings.
<b>Restore Factory Settings</b>	<p>Restores factory settings as an alternative to Format Flash, in a more flexible way. The following options are available:</p> <p><b>Uninstall HMI:</b> removes the HMI Runtime (entire qthmi folder) at the next start the device will behave as a brand new unit. This command does not reset settings such as IP address, brightness or RTC.</p> <p><b>Clear System Settings:</b> resets system parameters (registry settings) and deletes the following files:</p> <p><i>\\Flash\\Documents and Settings\\system.hv</i></p> <p><i>\\Flash\\Documents and Settings\\default\\user.hv</i></p> <p><i>\\Flash\\Documents and Settings\\default.mky</i></p> <p><i>\\Flash\\Documents and Settings\\default.vol</i></p> <p>System Mode password is also reset.</p> <p><b>Clear Controller Application:</b> clears current folders used by CODESYS V2.3 and CODESYS V3 internal controllers for applications:</p> <ul style="list-style-type: none"> <li><i>\\Flash\\QtHmi\\RTS\\APP\\*. *</i></li> </ul>

Element	Description
	<ul style="list-style-type: none"> <li>• \Flash\QtHmi\RTS\VISU\*. *</li> <li>• \Flash\QtHmi\codesys\*</li> <li>• \Flash\SSysData\$\codesys\*</li> </ul> <p><b>Clear sysdata settings:</b> clears \Flash\SSysData\$ folder</p> <p> <i>Service call: To be used only by technical support to fix display problems.</i></p> <p> Note: Not all these options are available for all HMI devices and BSPs.</p>
<b>Resize Image Area</b>	Resizes the flash memory reserved to store the splash screen image displayed at power up. Default settings are normally suitable for all units.
<b>Download Configuration OS</b>	Checks and upgrades the current version of the operating system used in System Mode(see " <a href="#">List of upgradable components</a> " on page 290 for details)
<b>Download Main OS</b>	Checks and upgrades the current version of the main operating system (see " <a href="#">List of upgradable components</a> " on page 290 for details)
<b>Download Splash Image</b>	<p>Loads a new file for the splash screen image displayed by the unit at power up.</p> <p> Tip: Update the splash screen image directly from the PB610-B Panel Builder 600 programming software.</p> <p>See "<a href="#">Update of system components from the application</a>" on page 291 for details.</p>
<b>Download Bootloader</b>	Checks and upgrades the current version of the system boot loader.
<b>Download Main FPGA</b>	Checks and upgrades the current version of the main FPGA file. This function may not be available for all platforms and versions.
<b>Download Safe FPGA</b>	Checks and upgrades the current version of the backup copy of the FPGA file. This function may not be available for all platforms and versions.
<b>Download System Supervisor</b>	Checks and upgrades the current version of the system supervisor firmware (used for the RTC and power supply handling).

## System Setting tool password protection

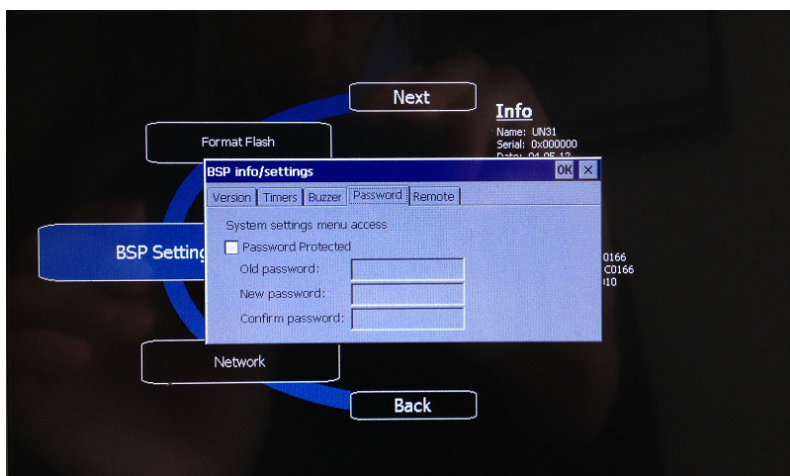


**CAUTION:** Working with the System Settings tool is a critical operation and, when not performed correctly, may cause product damages requiring service of the product. Contact technical support for assistance.

Restrict access to the System Settings tool with a password, so that critical functions can only be accessed by authorized personnel.

To activate password protection:

1. Choose **BSP Settings**: the **BSP Info/Settings** dialog is displayed.
2. In the **Password** tab select the **Password Protected** option and enter password.



The password must be at least 5 characters long.

From this dialog you can also change current password.



**Important:** Store password in a safe place since you cannot be reset this password. You will have to return the device to factory for password reset.

# 33    Updating system components in HMI devices

---

Most of the system software components can be easily upgraded ensuring a high degree of flexibility in providing updates and fixes to existing and running systems.

New software modules can be uploaded using USB flash drives and following an upload procedure (see ["Update system components via USB"](#) on page 293 for details).

Each HMI device is labeled with a product code including all factory settings (hardware, software and firmware components). Refer to this label for information on your HMI device. The HMI device update tool also provides detail on the components actually running on the device.



**CAUTION:** *Make sure you use the correct upgrade files, since loading upgrade files unsuitable for your device will cause serious system malfunction. Always check your device product code.*



Note: Upgrade files are distributed upon request as a part of technical support activity.



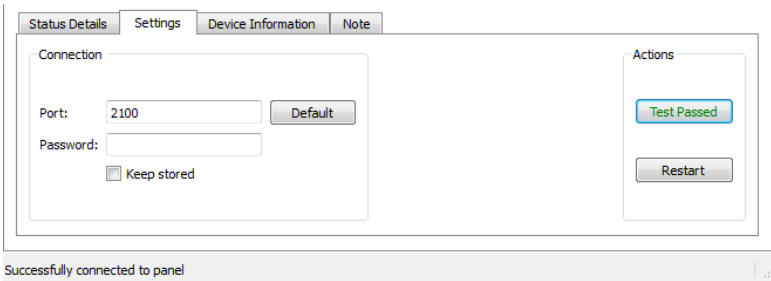
*Service call: Downgrade operations are complex tasks which might cause serious damage to your equipment if not performed correctly. These operations are reserved to technical support.*

---

Display information on connected devices .....	290
List of upgradable components .....	290
Update of system components from the application .....	291
Update system components via USB .....	293



## Display information on connected devices

The lower part of the **Manage Target** dialog displays information on the connected HMI devices.

Tab	Content
Status Details	Status of executed commands
Settings	<p>Password and communication port settings. <b>Test</b>: verifies HMI device connection parameters</p> <p><b>Restart</b>: resets the HMI device.</p> 
Device Information	Shows HMI device internal information
Note	Shows information on the selected component

## List of upgradable components

The HMI devices support the upgrade of the following components:

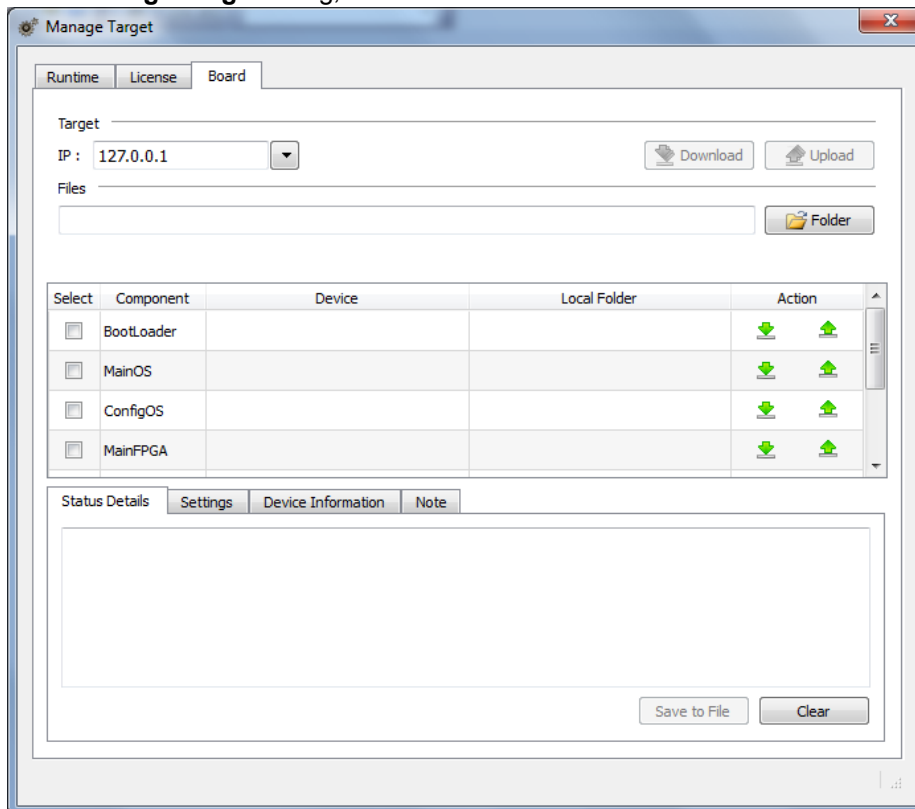
Component	Description
System Supervisor	<p>Firmware of the system supervisor controller (for example: packaged_GekkoZigBee_v4.13.bin).</p> <p><i>The System Supervisor component can be upgraded from V4.13 or above.</i></p> <p> <b>Important: Do not try to update versions V4.08, V4.09, V4.10 and V4.11 since they do not support automatic update from System Settings.</b></p>
Main FPGA	FPGA firmware (for example: h146xaf02r06.bin)
Safe FPGA	<p>Backup copy of the Main FPGA that ensures unit booting in case of main FPGA corruption (for example: h146xaf02r06.bin)</p> <p> <b>Important: Use the same file for updating Main and Safe FPGA components.</b></p>
Bootloader	Loader to handle device startup (for example: redboot_UN20HS010025.bin)
Main OS	Main Operating System (for example: mainos_UN20HS0160M0237.bin)
Configuration OS	Backup operating system that ensures units recovery in case of main operating system corruption (for example: configos_UN20HS0160C0237.bin)

## Update of system components from the application

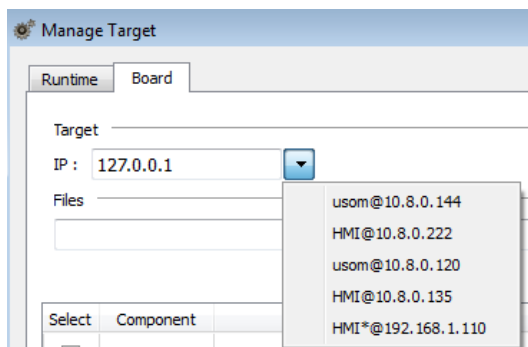
You can download system components to the HMI device using the Ethernet communication interface.

Path: **Run> Manage Target**

1. In the **Manage Target** dialog, click the **Board** tab.



2. Select the IP of the HMI device from the **Target** list. If the desired IP is not listed, type it directly into the box.




Note: Discovery service is a broadcast service. When a remote connection is done via VPN or from external networks it will not work and you will have to enter the address manually.

When the device has been recognized the HMI device details are displayed as in the example.

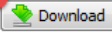

Select	Component	Device	Local Folder	Action
<input type="checkbox"/>	BootLoader	UN30HSxx012		
<input type="checkbox"/>	MainOS	UN30HSXX60M0183		
<input type="checkbox"/>	ConfigOS	UN30HSXX60C0183		
<input type="checkbox"/>	MainFPGA	h148xbc02r26		


3. Click **Folder** and select the local folder containing the system files that can be used for the update.

Files  

Select	Component	Device	Local Folder	Action
<input checked="" type="checkbox"/>	BootLoader	UN31HSxx012	uboot_UN31HSxx012.bin	
<input checked="" type="checkbox"/>	MainOS	UN31HSXX60M0195	mainos_UN31HSXX60M0195.bin	
<input checked="" type="checkbox"/>	ConfigOS	UN31HSXX60C0195	configos_UN31HSXX60C0195.bin	
<input type="checkbox"/>	MainFPGA			

4. Click the download icon next to each component to download it. Click **Download** to download several selected components at once: download progress is displayed in the **Status Display** tab.

Target   

Files  

Select	Component	Device	Local Folder	Action
<input checked="" type="checkbox"/>	BootLoader	UN30HSxx012		
<input checked="" type="checkbox"/>	MainOS	UN30_usbhid_2		
<input checked="" type="checkbox"/>	ConfigOS	UN30HSXX60C0176		
<input checked="" type="checkbox"/>	MainFPGA	h148xbc_emc_3		



Note: You need to restart the HMI device to finalize the update.

## Uploading a splash screen picture

You can replace the default splash screen image shown by the devices during the power up phase.

The image used as splash screen must comply with the following requirements:

Format	Bitmap, RGB 565 format
Size	< 500 KB
Bitmap width	Even number (for example 430x239)

To upload the splash screen image:



1. Rename the new image splash.bmp and copy it in the source folder.
2. Click **Download**.



Note: To ensure the best visual results, splash screen images must have a black background.

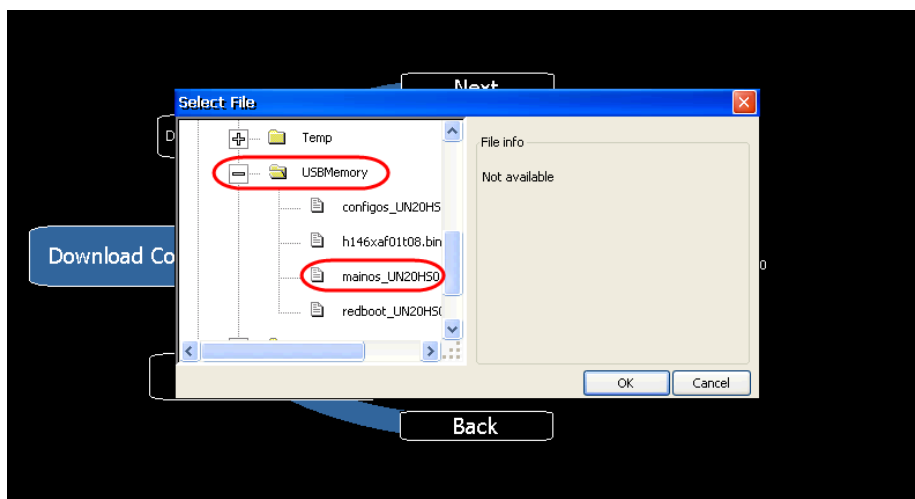
## Update system components via USB

System components can be updated using a USB flash drives. For each component a specific update file is provided.



**Important: A checksum .md5 file must be saved in the same location as the system file to be upgraded.**

1. Copy all the upgrade files you need to a USB drive and plug it into the USB port of the HMI device.
2. Start the System Settings tool in System Mode (see "[System Mode](#)" on page 286 for details).
3. Click on the desired download function.
4. Browse the content of the USB drive to the files to download. The example shows Main OS components.



5. Click **Download** to transfer files to the HMI device.



Note: From this dialog click **Upload** to transfer files to the USB device.

6. Follow the instructions displayed to complete the update: the progress of the operation is displayed in a progress bar.

This operation may require a few minutes.



**Important: Do not turn off the device while a system component is being upgraded.**



Note: Upgrading procedures depend on hardware and operating system versions. Contact technical support for assistance.



# 34    Protecting access to HMI devices

---

The following operations are password protected on the HMI device:

- HMI Runtime management: install HMI Runtime and update HMI Runtime
- Board management: replace main BSP components such as Main OS, Configuration OS, Bootloader, and so on
- Download and upload of project files

A default password is used HMI Runtime and PB610-B Panel Builder 600 to access the HMI device.

If you change your password on the HMI device you will need to change it also PB610-B Panel Builder 600 side.

---

<b>Changing password .....</b>	<b>296</b>
<b>Changing password on HMI device .....</b>	<b>296</b>
<b>Ports and firewalls .....</b>	<b>297</b>

## Changing password

To change the password in PB610-B Panel Builder 600:

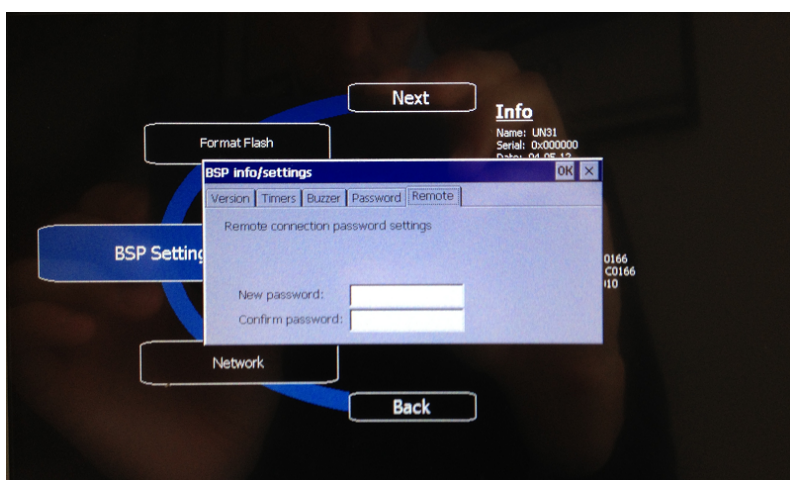
1. On the **Run** menu, select **Manage Target> Board> Connection** settings.
2. Enter password to allow PB610-B Panel Builder 600 to access HMI Runtime. Default port = 2100.

Leave **Old password** empty if no password is set on the HMI device side.

## Changing password on HMI device

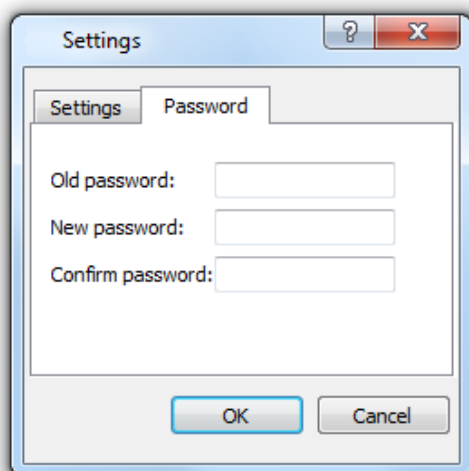
To change the password on the HMI device, use one of the following methods:

- In the System Settings Tool (System mode): **BSP Settings> Remote** tab (see ["System Mode" on page 286](#) for details).



*This feature is available from BSP versions V1.64 ARM UN30/31 and V2.73 MIPS UN20 based on WCE OS.*

- From the HMI Runtime context menu: **Settings> Password** tab.



Here you enter the same password set in the HMI device: the new password is stored into the device registry.

- Use the **Set Target Password** function in update package: the password is updated by HMI Runtime just after the update process is completed.



Note: A format of HMI device reset password device side.

For Win32 HMI Runtime, password is saved into *Users\[username]\AppData\Roaming\ABB\buildNumber\server\config\RemoteUpdateConfig.xml*.

## Ports and firewalls

Here a list of all the ports used by PB610-B Panel Builder 600 components.

Port	Usage	Remote Access	Board Management	Runtime/Project Management	CODESYS iPLC
80/tcp	HTTP port	Yes	-	Yes	-
21/tcp	FTP cmd port	-	-	Yes	-
2100/tcp	Board port	-	Yes	-	-
16384-17407/tcp	FTP data port (passive mode)	-	Yes	Yes	-
990/udp	UDP broadcast (Device discovery)	-	Optional	Optional	-
991/udp	UDP broadcast (Device discovery)	-	Optional	Optional	-
998/udp	UDP broadcast (Device discovery)	-	Optional	Optional	-
999/udp	UDP broadcast (Device discovery)	-	Optional	Optional	-
5100/tcp	JS Remote Debugger	-	-	Optional	-
1200/tcp	CODESYS 2.3 iPLC	-	-	-	Yes

### Remote access

Remote access is required to connect to HMI Runtime using:

- HMI Client
- Web access

### Runtime and project management ports

You use these ports to connect to HMI Runtime for operations such as update, installation and project download.

## Board management ports

You use these ports to connect to the HMI device for Board operations such as BSP update, splash image download and so on.



Note: When broadcast service is not available, for example in VPN networks, type in the exact IP address to connect to the HMI device from PB610-B Panel Builder 600.

# 35 Factory restore

---

If you're having problems with the HMI device, try and restore factory default settings from System Mode.

1. Enter **System Mode**.
2. Use one of the following operations available in rotating menu:
  - **Format Flash**, to clean the flash drive and registry configuration.
  - **Restore Factory Settings**, to clean only the select components.



Note: Both operations do not involve firmware factory restore (MainOS, ConfigOS, Bootloader, FPGA images, etc).

See "[System Mode](#)" on [page 286](#) for details.





# 36 Tips and tricks to improve performance

---

PB610-B Panel Builder 600 allows great flexibility for a project designers.

Follow these guidelines to create projects that perform better in terms of boot time, page change and animations.

---

<b>Static Optimization .....</b>	<b>302</b>
<b>FAQ on Static Optimization .....</b>	<b>305</b>
<b>Page caching .....</b>	<b>306</b>
<b>Image DB .....</b>	<b>306</b>
<b>Precaching .....</b>	<b>306</b>
<b>FAQ on precaching .....</b>	<b>306</b>

# Static Optimization

Static optimization is a technique used in PB610-B Panel Builder 600 to improve run-time performance.

Using a lot of images and pictures in a project might degrade performances, static optimization merges several images into a single background image thus reducing rendering and loading times. Using this method only one raster image needs to be loaded and rendered instead of many single raster and/or vector images.

When you create a project in PB610-B Panel Builder 600, the pages might contain widgets such as texts, images, background images, background colors and so on which can be classified as:

- **Static:** values or properties do not change at run time.
- **Dynamic:** values or properties change at run time.



Note: Based on security settings, static parts of widgets could be not merged to background. This happens when a widget is configured as "hide" in security settings.



**Important:** When you change the properties of widgets with JavaScript set the widget Static Optimization to Dynamic, otherwise changes to properties will be ignored.

When downloading or validating a project, PB610-B Panel Builder 600 identifies static components and renders them as background images to .png files. These background images are saved as a part of the project under the folder /opt.

Background images can be created as follows:

- full page background images, containing all widgets merged to page background
- group background images, containing a group of static widgets merged together to form a group background. For example, the Gauge group is normally composed by a background, a scale, a label and a needle, where background scale and label can all be merged to a single background image.

The **Static Optimization** page attribute enables and disables static optimization of the whole page. If it is set to **false** the optimization is totally disabled.

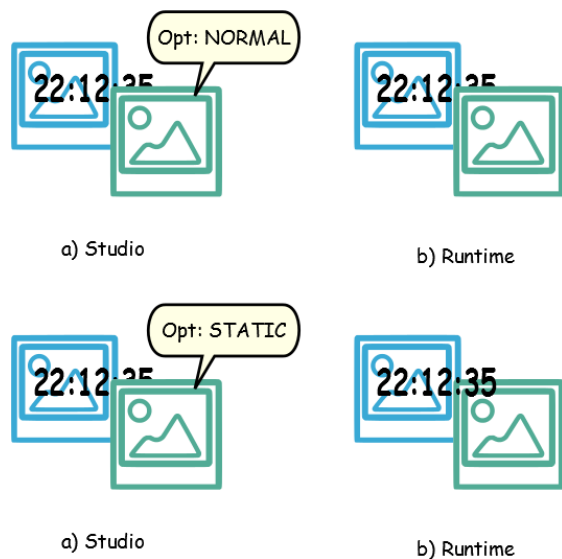
Finer control can be achieved setting the **Static Optimization** attribute of each single widget as follows:

- **Normal:** PB610-B Panel Builder 600 automatically detects if the widget can be merged with the background. This can be used if the widget is not a dynamic widget and does not overlap, that is it is not stacked above, a dynamic widget.
- **Static:** The image is forced to be merged with the background. This can be used when the static widget overlaps a dynamic transparent widget.



Note: In this case the automatic optimization will fail because it does not make any assumption on invisible areas which might be rendered at run time.

- **Dynamic:** The widget is not optimized at all. Use this flag when a static widget needs to be changed by Javascript.



## Tips for best performance

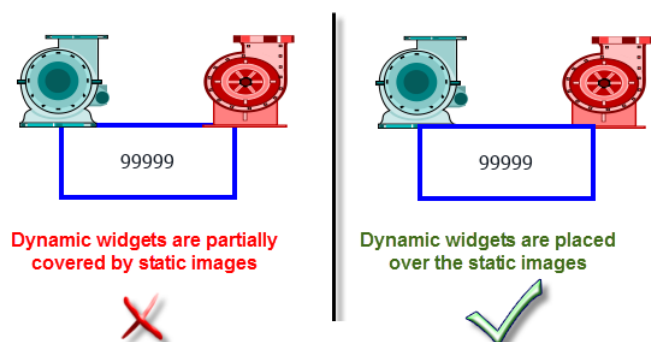
1. First of all: avoid placing static widgets over a dynamic widget. The overlapping area is computed considering the bounding rectangles of the widgets, that is the rectangles delimited by editing handles.
2. Don't use static optimization if your pages contain almost only dynamic objects. Static optimization would save many almost identical full size images for each page using up a lot of memory space that could be more effectively used to improve project performance with other techniques (such as, for example, page caching).
3. Bounding rectangles can include transparent areas, minimize transparent areas (for example splitting the image in multiple images) since they can be a waste of resources even when optimized.
4. Optimize image size. The image will be rendered at the size of the image widget containing the image. For best performances the widget needs to be the same size of the image.
5. Avoid using **Scale to fit** for image widgets, since this forces a rescaling at run time for dynamic images and "hides" the actual image size during editing.
6. Use **Size to fit** to make the widget to the real size of his contents.
7. If overlapping cannot be avoided make sure to place the static widgets in the back, that is behind the dynamic widget.
8. Choose the image file format based on the HMI device you are connecting to.

9. Avoid using too many widgets in a single page. Often widgets are placed outside the visible area or their transparency is controlled by a tag. Since widgets are loaded even if they are not visible, having too many widgets in a page can significantly slow down the page change time.
10. Split a page with many widgets into multiple pages with less widgets.
11. For popping up new graphic elements in a page, prefer dialog pages with controlled positioning to transparent widgets.
12. Check the *opt* folder to see if static optimization is working as expected, the widgets z-order might need to be adjusted.
13. Numeric fields are often used to run JavaScript code on OnDataUpdate event even if the widget doesn't need to be visible on the page. In this case place the widget outside the page visible area instead of making it invisible, altering font color or visibility property. In the latter case you might end up with many left over wedges.
14. Use a HotSpot button if you need a touch area to react to user inputs.
15. If you reuse a widget from the gallery or you create your own, remember to set the correct optimization properties. For example button widgets are dynamic widgets, if you use a button widget just for its frame it won't be optimized since the button widget is dynamic. If you just need the frame you should use the Up image.
16. With many pages having many dynamic widgets and using a common template:
  1. set template static optimization to **true**,
  2. set page static optimization to **false**, since the background is already provided by the template.

In this scenario the background image can be reused by many different pages thus saving memory space.

17. Do not use dynamic widgets, such as buttons, only for graphic purposes, when the button function is not needed, use image widgets instead to obtain the same graphical effect.

Here is an example of a correct and an incorrect use of static optimization.



## Supported image formats

PB610-B Panel Builder 600 supports several raster formats like BMP, PNG, JPEG, TIFF and the vector format SVG. Here a list of pros and cons:

Image format	Pros	Cons
RASTER	<ul style="list-style-type: none"> <li>• Fast rendering</li> <li>• Well standardized</li> </ul>	<ul style="list-style-type: none"> <li>• Big file size</li> <li>• Fixed resolution</li> </ul>
VECTOR (SVG)	<ul style="list-style-type: none"> <li>• Small file size</li> <li>• Rescale without quality loss</li> <li>• Can handle dynamic properties</li> </ul>	<ul style="list-style-type: none"> <li>• Complex SVG images with many graphic items and layers can be slow to render.</li> <li>• Creating an optimized SVG is not simple.</li> <li>• Only Tiny 1.2 (<a href="http://www.w3.org/TR/SVGTiny12/">http://www.w3.org/TR/SVGTiny12/</a>) supported.</li> </ul>



Note: Scour software is free tool that can be used to remove foreign code from file (<http://www.codedread.com/scour/>).

## Static optimization of templates

Template pages can have large amounts of static content. However, static optimization cannot be applied to a template page, since where the template is used is based on the page design.

If a huge background image should be repeated in every page that uses the same template, this would increase the footprint of the device as the same static image would be created for each of the pages using the template page.

## FAQ on Static Optimization

**Q: In a page where there are a few identical widgets, in the *opt* folder I see a PNG for each one of them. If they are really identical, why should the software duplicate them instead of having just one PNG?**

A: The software does not know if static images are actually the same since each widget could have different settings/properties altering the actual rendering at run time.

**Q: Why are the static images stored in a separate folder called *opt* instead of storing them directly in the project folder?**

A: This avoids name collisions and allows skipping the upload of optimization images

**Q: Why are the static images stored as a PNG files instead of common JPEG files?**

A: PNG format uses a lossless compression for images and supports transparencies. JPEG files would render fuzzier compared to the PNG files with a different result in PB610-B Panel Builder 600(not using optimization) and HMI Runtime.

**Q: What will happen when no optimization is done in the software?**

A: Every single widget is rendered at run time. In particular SVG images may require a lot of time to render in an embedded platform.

## Page caching

Once accessed all pages are kept in a RAM cache up to the maximum allowed cache size depending on the actual platform's available RAM. This allows a much faster access since cached pages, once reloaded, only need to re-paint their content without reloading all page resources.

## Image DB

Image DB is a technique used to track the usage of image files and reduce the cost of image loading by caching most frequently used images (example, Push Button images, Gauge needles, Slider thumbs and so on). The same image used in many different places is therefore loaded just once.

The image DB function will preload the top most used images at startup until memory limit is reached. This would further improve the individual page loading times.

The file `imagecachelist.xml` is created in `project/opt` folder, containing relevant information:

- Fill color (in case of SVG images)
- Size of SVG image
- Number of times an image is used in the project
- Number of different sizes for the same image

### Tips for using the Image DB function

1. Use uniform size of buttons, gauges and other widgets wherever possible.
2. Use same color themes among widgets of the same kind.

## Precaching

The Precache attribute of pages can be used to notify HMI Runtime to preload some pages in RAM at boot time for quicker access. Precaching is useful for complex pages having many dynamic widgets.

When this function is enabled on a page, access to the page is faster, however it also slows down boot-time since the system is not ready until all pages to be precached are not saved into the RAM.

### Tips to precaching

1. Enable the precache function just for few pages having many dynamic widgets or for pages frequently used by users.
2. Do not enable the precache function for all the pages in the project since you would hit out of memory and have no benefit at all.
3. Disable static optimization for pages where the precache function is enabled to reduce memory used.

## FAQ on precaching

### Page limit for precaching

Based on the size and complexity of a page, the space required for precaching can be from 1,5Mb to 3Mb.

When a project is loaded, HMI Runtime proceeds as follows:

1. Page images are preloaded until 76 MB of memory space is still available (imageDBLowMem)
2. Pages where precache is set to **true** are preloaded until 64 MB of memory space is still available (pageCacheLowMemMax). The images of these pages are loaded in the RAM (into the Image DB).

When the project is ready:

1. Any new page visited is saved in the cache (RAM) with all related images until 40 MB of memory space is still available (pageCacheLowMemMin)
2. When a page change happens and space in RAM is critical (<40MB), the HMI Runtime starts emptying the cache (RAM) removing pages and related images until 64 MB of memory space is made available. HMI Runtime removes data stored in the cache in the following order:
  1. last visited pages and bigger and unused images (>320x240),
  2. if more memory is needed also the pages in precache and all images loaded in Image DB can be removed.





## 37 FAQ

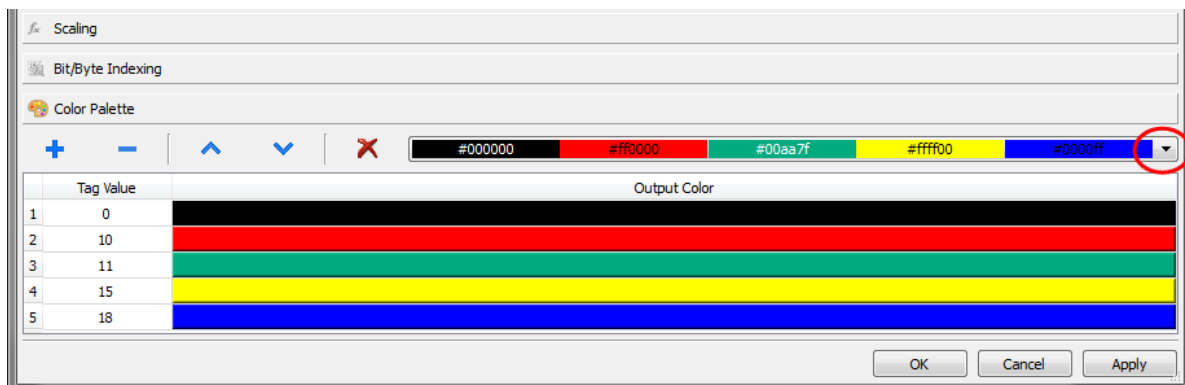
### Changing fill color property according to tag values

PB610-B Panel Builder 600 allows to change the color property of a widget dynamically, based on tag values in two ways:

- Using ColorPalette
- Connecting the Color property to a String type tag

#### Changing color property using ColorPalette

1. Create the tag (internal or PLC) that you want to refer to for color management. The tag can be of any data type. On the basis of the value of this tag, the color will change.
2. Attach this tag to the **Fill Color** property of an object (for example, a button).
3. In the same dialog select the **ColorPalette** tab and add the colors that will be used for the object according to the tag value.



Note: The last used colors' tables are saved and can be reused selecting them from the colors list box on the toolbar.

#### Changing color property connecting Color property to a String type tag

1. Create the tag (internal or PLC) that you want to refer to for color management. On the basis of the value of this tag, the color will change. The tag must be of String type and the **Arraysizes** property of the tag must be big enough to contain the string formatted as explained here.
2. Attach this tag to the **Fill Color** property of an object (for example, a button).
3. Write in the **String** tag the RGB color code of the required color. Use one of these formats:
  - **#XXYYZZ**, Where XX, YY and ZZ are the RGB components of the needed color expressed in Hexadecimal format (range 00–FF).
  - **rgb(XXX,YYY,ZZZ)**, where XXX, YYY and ZZZ are the RGB components of the needed colors expressed in Decimal format (range 0–255).



Note: This feature can be applied to all the objects available in the Widget gallery that have a color property. The run-time change of the color is possible only thanks to the properties of the SVGs that are composing the object. This feature can not be applied to other image formats such as JPEG or BMP files.

---

# 38 CP600-eCo products

---

This section describes CP600-eCo HMI products and includes a description of the main features of this product family.  
CP600-eCo HMI products must be programmed using PB610-B Panel Builder 600 V2.00 or higher.

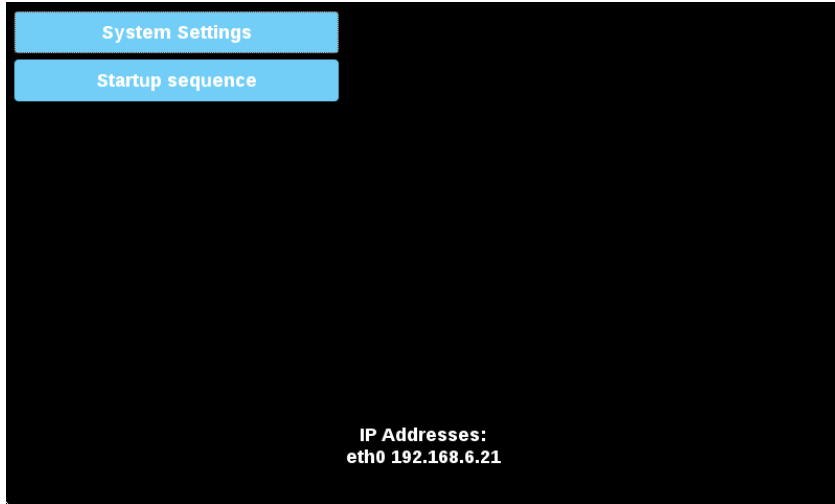
---

<b>The Runtime loader .....</b>	<b>312</b>
<b>Limitations .....</b>	<b>314</b>
<b>Converting projects between different HMI types .....</b>	<b>317</b>

# The Runtime loader

CP600-eCo HMI devices are delivered from factory without HMI Runtime.

When you power up the device for the first time, the Runtime Loader window is displayed.



## System Settings

The user interface of System Settings is based on HTML pages and can be accessed both locally on the HMI device screen and remotely using a Web browser.

1. Click **System Settings** to configure system options.
2. To access System Settings using a Web browser, enter the IP address of the device, in the following format:

*https://IP/machine\_config*

Administrator username with full access right is "admin" with default password "admin". Generic username is "user" with default password "user"

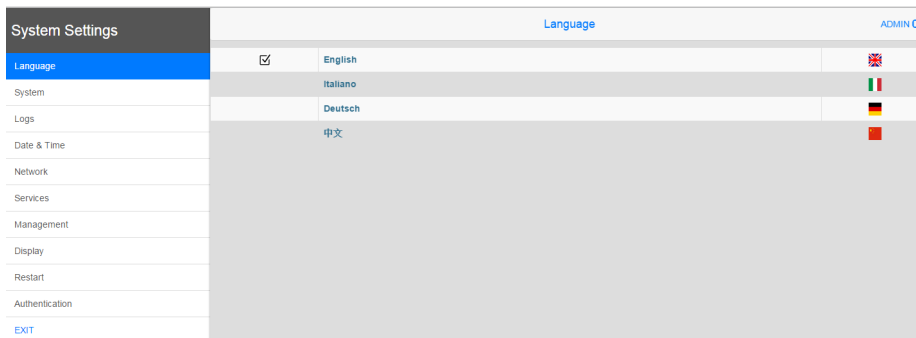


**WARNING: For security reasons, change the default passwords for both usernames (passwords can be modified from the "System Settings -> Authentication" command)**



Note: Remote access requires port 443.

3. Browse through the options available in the menu on the left: the active item is highlighted and related information is displayed on the right.



Default security protocols proposed by the HTTPS server in the CP600-eCo HMI device are:

- SSLv3 256 bits ECDHE-RSA-AES256-SHA
- TLSv1 256 bits ECDHE-RSA-AES256-SHA



**WARNING: We discourage usage of CBC cyber suites in the context of SSL3 or TLSv1.0 connections since potentially affected by some vulnerabilities.**

See the CP600-eCo Series Operating Instructions manual for detailed information on how to set up system options.

## Installing applications

1. Click **Startup sequence** to install/uninstall applications (for example HMI Runtime).
2. Click **Install** to install the application using an update package: the procedure is run automatically.

See "[Transferring the project to HMI device](#)" on page 53 for instructions on how to prepare the update package.

If you are using USB drives, the default path is /mnt/usbmemory.



Note: File systems supported are FAT16/32 and Linux Ext2, Ext3 and Ext4.

## Uninstalling applications

Select one of the applications in the list and click **Uninstall** to remove it from HMI device.



Note: When uninstalling the runtime, you remove from the system not only the application but also all associated data such as projects, dynamic data and settings.

To change the order of execution of installed applications, go to **Boot Sequence** and drag and drop applications to change their order.

# Limitations

Feature	Available in PB610-B Panel Builder 600 V2.00	Notes
Backup & Restore	No	-
LaunchApplication	Yes	<p>Only generic Launch Application is supported. The other specific Launch Applications are not supported:</p> <ul style="list-style-type: none"> <li>• LaunchBrowser</li> <li>• LaunchVNC</li> <li>• LaunchPDFViewer</li> <li>• LaunchUpdater</li> <li>• LaunchHMICloudEnabler</li> </ul>
Buzzer	Yes	<p>Buzzer on touch is configured in the project file. See <a href="#">"Project properties pane" on page 40</a> for details.</p> <p>System variable "Buzzer Setup = 1 (on touch)" is not supported . See <a href="#">"Buzzer variables" on page 65</a> for details.</p>
Reports	Only PDF printing is supported	-
Access to a network path	No	-

## Not supported System Variables

Following System Variables are not supported:

- Battery LED
- Battery Timeout
- External Timeout (seconds are converted to minute rounding up to next int value, for example, 60, 120, 180...)
- SD Card Name
- SD Card Size
- SD Card Freespace
- SD Card Status

## Not supported actions

Following actions are not supported:

Action group	Action
Page	LaunchVNC LaunchPDFViewer LaunchBrowser LaunchHMICloudEnabler
System	ControlUserLED CopyCodesysProject
Print	PrintText PrintBytes

## Security and access protection to HMI devices

The following operations can be protected with a password on the device:

Context	Operation
Runtime management	Install runtime Update runtime
Board management	Replace main BSP components such as: <ul style="list-style-type: none"> <li>• MainOS</li> <li>• ConfigOS</li> <li>• Bootloader</li> </ul>
Project file	Download Upload

HMI Runtime and PB610-B Panel Builder 600 use a default password to access the HMI device.

## Changing HMI device password

You can change the device password in HMI Runtime using one of these methods:

- **System Settings > Authentication**
- Runtime context menu > **Settings > Password** tab
- Set **Target Password** in update package: the password is updated by HMI Runtime after the update process has been completed. If the update fails (for example, the old password does not match the HMI password) a popup warning is displayed.

You can also change the password from PB610-B Panel Builder 600: **Manage Target > Board > Connection Setting > Connection**

Port used for this service is 443.

## Supported protocols

Protocols based on Ethernet and serial protocols (RS-232, RS-422 and RS-485) are supported. Protocols requiring plug-in modules are not supported.

## Table of functions and limits

Function	Max limit
Number of pages	100
Number of basic widgets	1000 x page
Number of tags	1000
Number of dialog pages	50 (max 5 can be opened at the same time)
Number of Recipes	32
Number of parameter sets for a recipe	1000
Number of elements per Recipe	1000
Number of user groups	16
Number of users	16
Number of concurrent remote clients	1
Number of schedulers	10
Number of alarms	500
Number of templates pages	50
Number of actions programmable per button state	16
Number of Trend Buffers	15
Number of curves per trend widget	5
Number of samples per trend buffer	200000
Number of tags per trend buffer	200
Number of trend buffer samples for a project	600000
Number of messages in a message field	1024
Number of languages	12
Number of events per buffer	1024
Number of event buffers	4
JavaScript file size per page	8KB
Size of project on disk	60MB



Function	Max limit
Number of indexed instances	100
Number of indexed alias	100
Number of indexed tag sets	30
Number of physical protocols	2
Number of reports	32
Number of reports pages	32
Max number of variables in variables widget	255
User folder size (UpdatePackage.zip)	5MB
FTP additional folders	5

## Converting projects between different HMI types

CP600-eCo HMI devices have some functional limitations compared to standard HMI devices. Project conversion is supported, however, some manual operations may be required if the project uses features not supported in one of the two platforms.

### Converting projects from CP600-eCo to CP600

CP600-eCo HMI devices support a subset of features compared to CP600 HMI devices, therefore, converting from CP600-eCo to CP600 does not require any adjustment. However, CP600-eCo HMI devices are based on Linux OS while CP600 HMI devices are based on Windows CE OS. Projects that use OS-specific external applications or paths (as an example the action **LaunchApplication**) may require manual adjustment.

### Converting projects from CP600 to CP600-eCo

Converting a project from CP600 to CP600-eCo may require manual changes to compensate for unsupported features.



**Warning: Adjust your projects removing not supported features before you convert them from CP600 to CP600-eCo.**

Before you convert your project:

- Verify limitations and features not supported by the CP600-eCo HMI device (see "[Limitations](#)" on page 314 for details).
- Remove unsupported widgets, actions, system variables, protocols, project properties.
- If the project uses external storage, such as SD card not supported in CP600-eCo HMI devices, remove it or change paths to point to USB or internal flash memory.
- Adjust OS-specific external applications or paths.

- Reduce project size according to the HMI device type limitations (see "[Limitations](#)" on page 314 for details).
- Since CP600 and CP600-eCo devices are based on different hardware platforms with different CPU speed, RAM memory size, cache size, make sure you check project boot time and page loading time for each page in the project
- If the project includes features such as Reports or Buzzer, make sure your applications are compatible with the available features
- Verify JavaScript code for OS-specific operations.

## OS-specific features

Linux is case sensitive, Windows CE is not. As a consequence, projects on CP600-eCo HMI devices, which are based on Linux OS, might have two different files named 'dump1.csv' and 'Dump1.csv'; this would not be possible on a CP600 HMI device which is Windows CE based.

# 39 Communication protocols

---

This section describes the available protocols.



Note: Changes in controller hardware or protocols may have occurred since this documentation was created. Always test and verify the functionality of the application. To accommodate developments in the controller hardware and protocols, drivers are continuously updated. Accordingly, always ensure that the latest driver is used in the application.

Different physical media, gateways, routers and hubs can be used in the communication network. Also, other devices can independently make simultaneous use of the network. However, it is important to ensure that the traffic generated by these devices does not degrade the communication speed (round-trip time) to an unacceptable level.

---

<b>ABB CODESYS Ethernet .....</b>	<b>320</b>
<b>ABB Mint Controller HCP .....</b>	<b>328</b>
<b>ABB Modbus RTU .....</b>	<b>332</b>
<b>ABB Modbus TCP .....</b>	<b>339</b>
<b>ABB Pluto .....</b>	<b>345</b>
<b>BACnet .....</b>	<b>351</b>
<b>ABB CODESYS Serial .....</b>	<b>360</b>
<b>CODESYS V2 Ethernet .....</b>	<b>367</b>
<b>Modbus RTU .....</b>	<b>377</b>
<b>Modbus RTU Server .....</b>	<b>387</b>
<b>Modbus TCP .....</b>	<b>393</b>
<b>Modbus TCP Server .....</b>	<b>404</b>

# ABB CODESYS Ethernet

The ABB CODESYS Ethernet communication driver for Ethernet has been specifically designed to support communication with ABB controllers series AC500 designed for standardized IEC 61131-3 programming, based on the CODESYS V2.3 system.



Note: To accommodate developments in the controller protocol and hardware, drivers are continuously updated. Make sure the latest driver is used in the application.



Note: Changes in the controller protocol or hardware may have occurred since this documentation was created. This may interfere with the functionality of this driver. Therefore, always test and verify the functionality of the application.

## Adding a protocol

To configure the protocol:

1. In the **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The driver configuration dialog is displayed.

## Protocol Editor settings

Add a driver in the Protocol editor and select **ABB CODESYS ETH** from the list of available protocols.

The following protocols type are supported:

- Tcp/Ip Level 2 Route
- ABB Tcp/Ip Level 2 Route AC
- Tcp/Ip

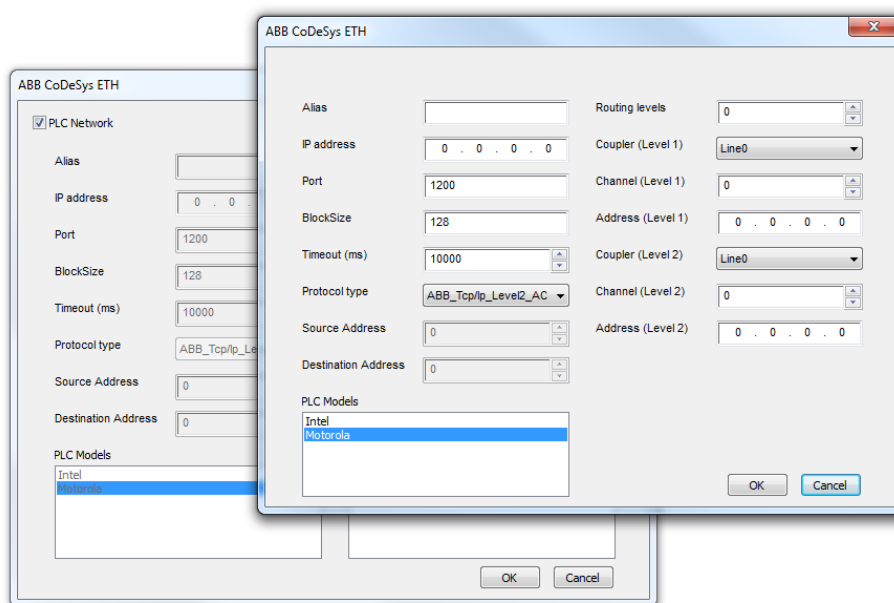
Select protocol type from the **Protocol type** combo box in the **ABB CODESYS ETH** dialog.

Some of the parameters of the dialog are common to the different protocol types, others are specific.

The parameters common to the different protocol types are:

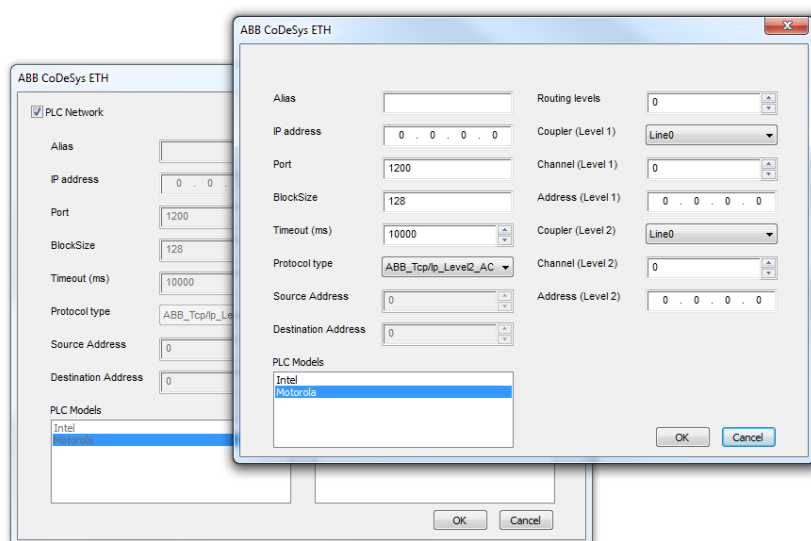
Element	Description
<b>Alias</b>	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
<b>IP address</b>	Ethernet IP address of the controller
<b>Port</b>	Port number used for the communication. Default value is 1200 for ABB drivers. For AC500 and 3S drivers select port 1201.
<b>BlockSize</b>	The max block size supported by your controller
<b>Timeout</b>	Time delay in milliseconds between two retries in case of missing response from the server device.
<b>PLC Model</b>	Byte order that will be used by the communication driver when sending communication frames to the PLC; Intel is also commonly referred as “little-endian”, Motorola as “big-endian” Select “Motorola” for AC500.

Element	Description
<b>Protocol type</b>	<p>Three different protocol types are available:</p> <ul style="list-style-type: none"> <li>• <b>Tcp/Ip</b></li> <li>• <b>Tcp/IP Level2 Route</b></li> <li>• <b>ABB Tcp/Ip Level2 AC</b></li> </ul>
<b>PLC Network</b>	<p>The protocol allows the connection to multiple controllers. To set-up multiple connections, check “PLC network” checkbox and enter the IP Address per each slave you need to access.</p>



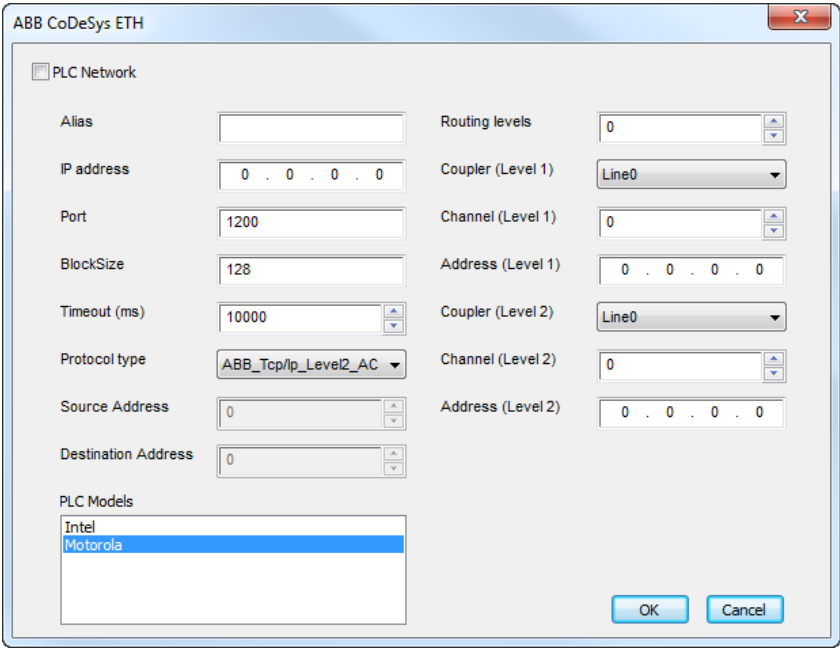
## Protocol types

The **Tcp/Ip** protocol type corresponds to the 3S Level 4 driver and does not require additional setup besides the common parameters.



The **Tcp/IP Level2 Route** protocol type corresponds to the standard 3S Level 2 Route driver and requires two additional parameters:

Parameter	Description
<b>Source Address (SrcAdr), Destination Address</b>	Destination is the node of the PLC and allows the protocol to read variables in a sub-network. The address is used to read variables when multiple PLCs are connected in a sub-network (serial network) but only one of it has the Ethernet interface. <div><i>This is currently not applicable for AC500 PLCs.</i></div>



The **ABB Tcp/Ip Level2 AC** protocol type implements a specific variation of the standard Level 2 protocol with the additional use of a routing driver. This protocol type is normally used to connect to PLCs via other PLCs acting as gateways.

This protocol type requires the following additional parameters:

- Routing Levels
- Coupler (Level 1)
- Channel (Level 1)
- Address (Level 1)
- Coupler (Level 2)
- Channel (Level 2)
- Address (Level 2)

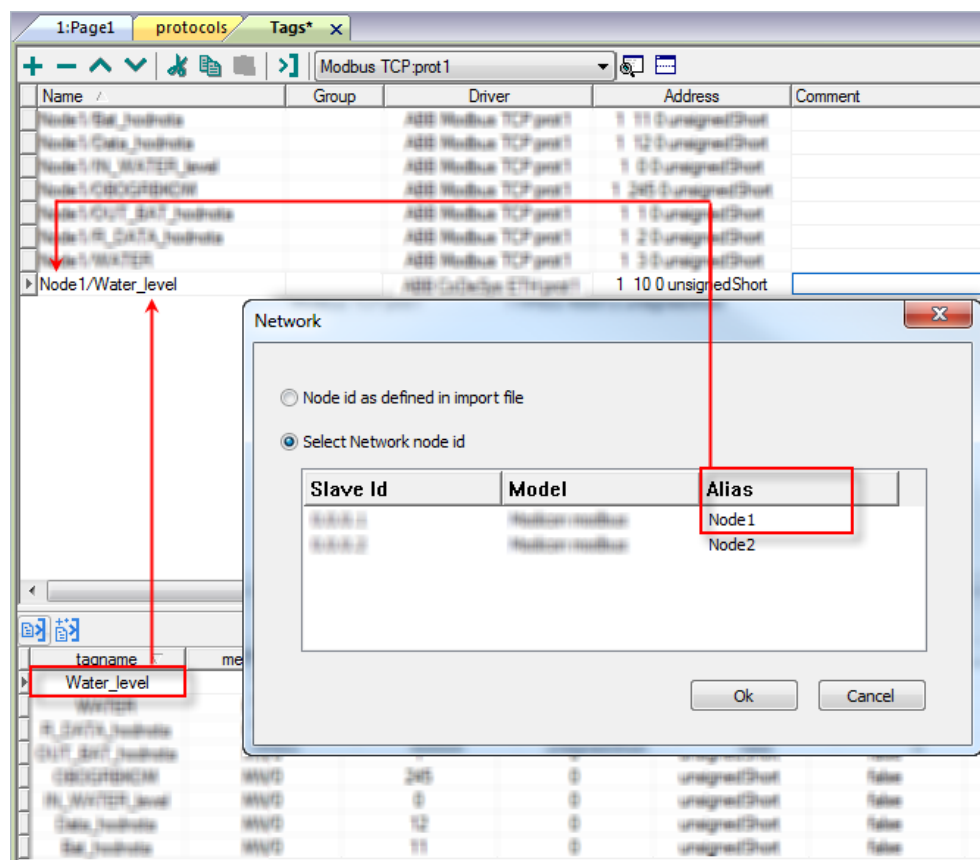
For detailed information see *AC500 and Control Builder* documentation, chapter *Programming interfaces to the AC500 used by the Control Builder*.

### Adding an alias name to a protocol

Tag names must be unique at project level, however, the same tag names might need to be used for different controller nodes (for example when the HMI device is connected to two devices running the same application).

When creating a protocol you can add an alias name that will be added to tag names imported for this protocol.

In the example, the connection to a certain controller is assigned the name **Node1**. When tags are imported for this node, all tag names will have the prefix **Node1** making each of them unique at the network/project level.



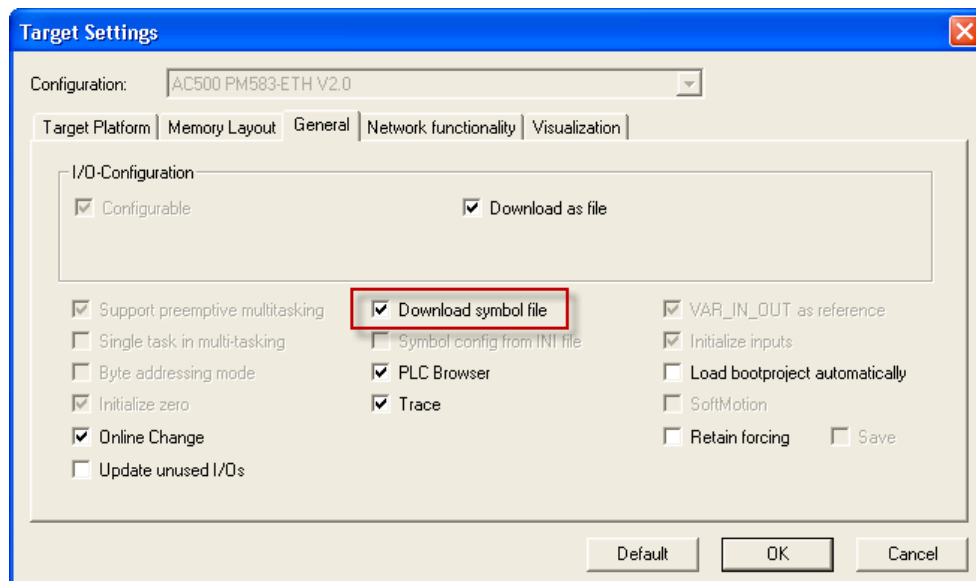
Note: Aliasing tag names is only available for imported tags. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name.

The Alias string is attached on the import. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are re-imported, all tags will be re-imported with the new prefix string.

## CODSYS software settings

When you create the project in CODESYS V2, select **Download symbol file** (*Target Settings > General*).



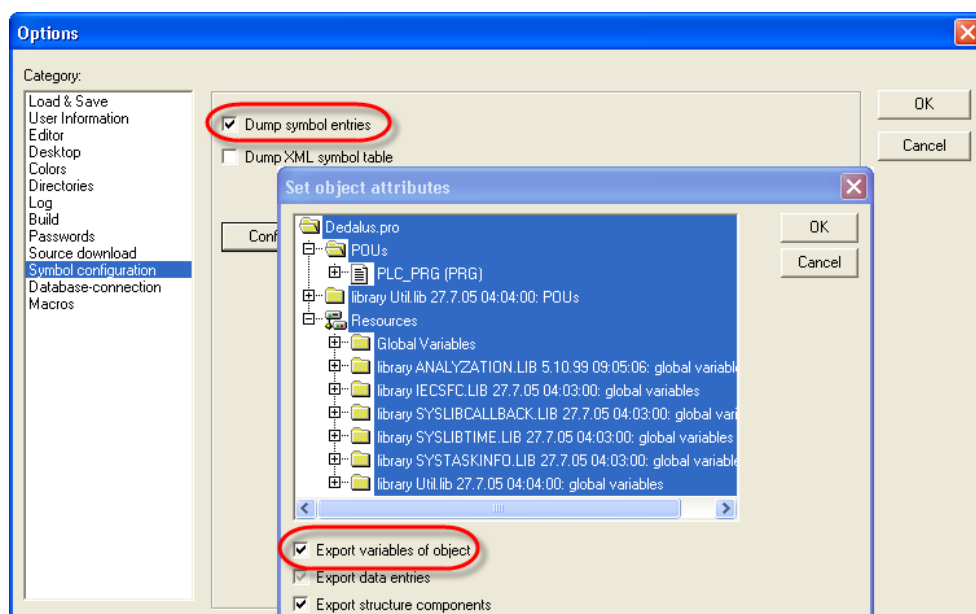


Note: ABB CODESYS Ethernet driver supports the automatic symbol file (SDB) upload from the controller; any change in the tag offset due to new compilation on PLC software side does not require a symbol file re-import. The Tag file has to be re-imported only in case of tag renaming or addition of new tags.

## Exporting tags from the controller

When configuring PLC using the manufacturer's configuration software, enable Symbol file (file with .sym extension) creation under the CODESYS programming software:

1. In the **Project** menu, click **Options**.
2. Select **Symbol configuration**.
3. Select **Dump symbol entries**.
4. Click **Configure symbol file**: the **Set object attributes** dialog is displayed.
5. Select **Export variables of object**.
6. Click **OK**.



## Importing tags

You may import tags from a .sym file exported from a controller. See ["Importing tags" on page 21](#).

## Standard data types

The import module supports variables of standard data types and user defined data types.

The following are considered as standard data types:

### Supported data types

- BOOL
- WORD
- DWORD
- INT
- UINT
- UDINT
- DINT
- STRING\*
- REAL
- TIME
- DATE & TIME

and 1-dimensional ARRAY of the types above.



Note \*: String length for a STRING variable in PLC should be max 80 characters. Declare a STRING variable either with a specified size (str: STRING(35) or default size (str: STRING) which is 80 characters.

## Node Override IP

The protocol provides the special data type Node Override IP which allows you to change the IP address of the target controller at runtime.

This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

Node Override IP	Modbus operation
0.0.0.0	Communication with the controller is stopped, no request frames are generated any-more.
Different from 0.0.0.0	It is interpreted as node IP override and the target IP address is replaced runtime with the new value.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override IP variable.



Note: Node Override IP values assigned at runtime are retained through power cycles.

## Limitations

CODESYS Level 4 is not supported. Maximum block size is 1024.

## Communication status

The current communication status can be displayed using system variables. See ["Communication variables" on page 66](#).

Codes supported for this communication driver:

Error	Cause and action
<b>Symbols file not present</b>	Check Symbol file and download again the PLC program
<b>“tag” not present in Symbols files</b>	Check if the Tag is present into the PLC project
<b>Time out on Acknowledge</b>	Controller didn't send acknowledge
<b>Time out on last Acknowledge</b>	Controller didn't send last acknowledge
<b>Time out on data receiving</b>	Controller didn't reply with data
<b>Connection timeout</b>	Device not connected

# ABB Mint Controller HCP

This communication protocol allows the HMI devices to connect to the ABB motion and servo drive devices using the HCP and HCP2 communication protocols.

## Adding a protocol

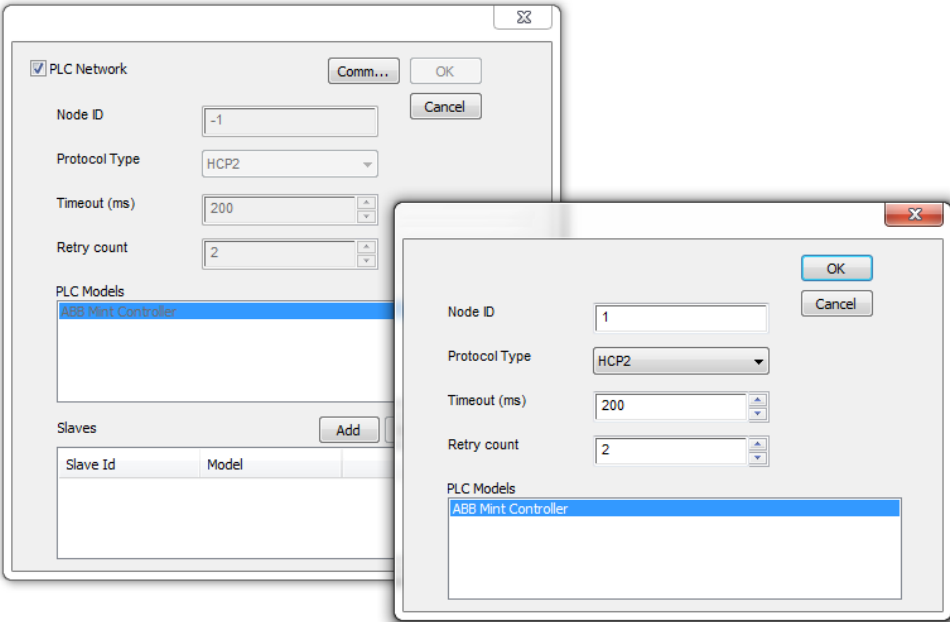
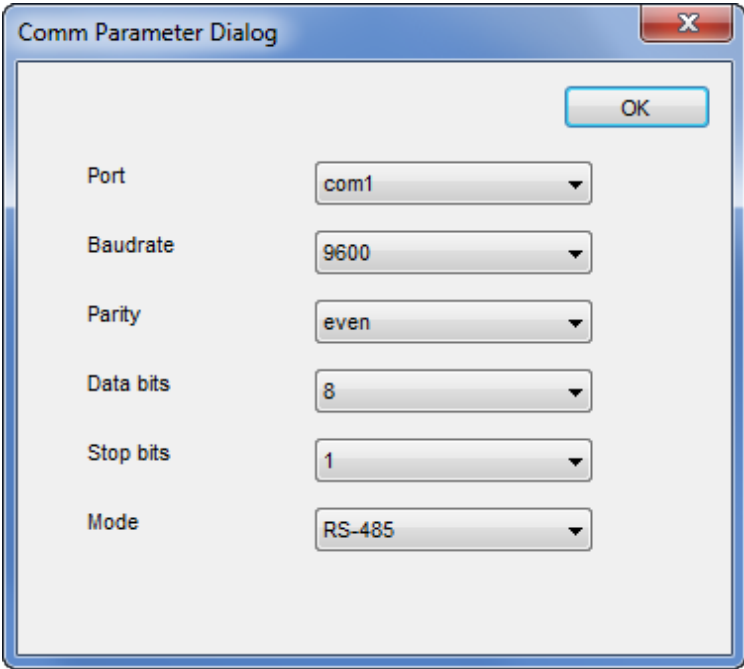
To configure the protocol:

1. In the **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The driver configuration dialog is displayed.

## Protocol Editor settings

Element	Description
<b>Node ID</b>	Node ID assigned to the controller device.
<b>Protocol Type</b>	Two protocols are available: <ul style="list-style-type: none"> <li>• <b>HCP</b></li> <li>• <b>HCP2</b></li> </ul>
<b>Timeout (ms)</b>	Time delay in milliseconds between two retries in case of missing response from the server device.
<b>Retry count</b>	Number of times a certain message will be sent to the controller before reporting the communication error status.
<b>PLC Models</b>	PLC model you are going to connect to.

Element	Description
PLC Network	<p>The protocol allows the connection of multiple controllers to one HMI device. To set-up multiple connections, check “PLC network” checkbox and enter the node ID per each slave you need to access.</p> 
Comm...	<p>If clicked displays the communication parameters setup dialog.</p> 

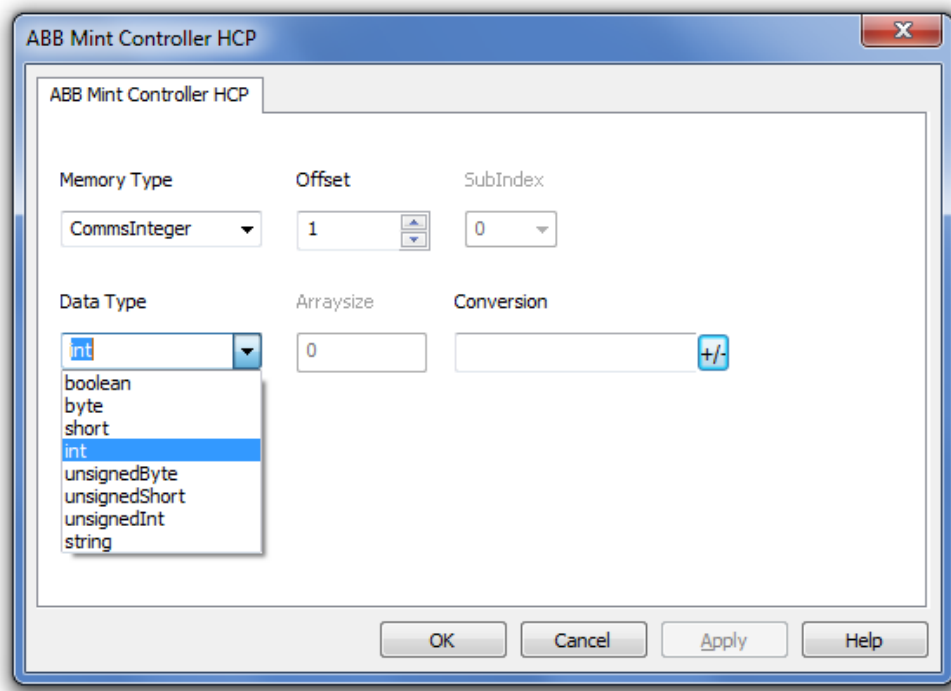
Element	Description	
	Element	Description
	Port	Serial port selection. <ul style="list-style-type: none"> <li>• <b>COM1</b>= device PLC port.</li> <li>• <b>COM2</b>= computer/printer port.</li> </ul>
	Baudrate, Parity, Data Bits, Stop bits	Serial line parameters.
	Mode	Serial port mode. Available modes: <ul style="list-style-type: none"> <li>• <b>RS-232</b>.</li> <li>• <b>RS-485</b> (2 wires).</li> <li>• <b>RS-422</b> (4 wires).</li> </ul>

## Data types

The ABB Mint Controller HCP driver provides the support for two Memory Types which are referring to the same physical memory area in the Mint controller:

- **Comms**: should only be used with floating point values. The Mint program on the ABB controller should use COMMS to access this data.
- **CommsInteger**: allows a variety of integer-based data types to be selected.

If the Mint controller program uses...	then...
COMMS keyword for a tag setup to use the Commsinteger memory type	only the bottom 23 bits will be accurate (due to floating point precision of the COMMS keyword).
COMMSINTEGER keyword for a tag setup to use the Commsinteger memory type	the value is precise for the full 32 bits.



## Communication status

Current communication status can be displayed using system variables. See ["Communication variables" on page 66](#).

Codes supported by this communication driver:

Error	Cause	Action
<b>NAK</b>	The controller replies with a not acknowledge.	-
<b>Timeout</b>	A request is not replied within the specified timeout period.	Check if the controller is connected and properly configured to get network access.
<b>Line Error</b>	An error on the communication parameter setup is detected (parity, baud rate, data bits, stop bits).	Check if the communication parameter settings of the controller is compatible with the device communication setup.
<b>Invalid response</b>	The device did received a response with invalid format or contents from the controller.	Ensure the data programmed in the project are consistent with the controller resources.
<b>General Error</b>	Unidentifiable error. Should never be reported.	Contact technical support.

# ABB Modbus RTU

The HMI devices can be connected to a Modbus network as the network master using this driver.

This specific implementation of the Modbus RTU driver provides easy handling of the connections to ABB controllers providing specific support for PLC models and tag import facilities.

## Adding a protocol

To configure the protocol:


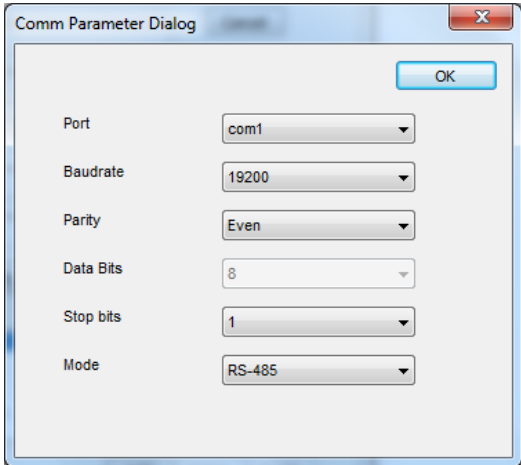
1. In the **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

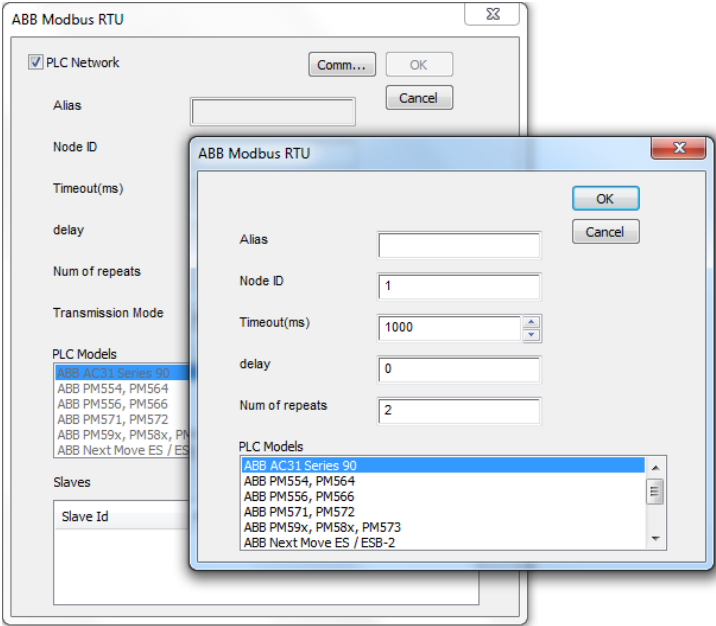

The driver configuration dialog is displayed.

## Protocol Editor settings

Element	Description
<b>Alias</b>	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
<b>Node ID</b>	Modbus node of the slave device.
<b>Timeout (ms)</b>	Time delay in milliseconds between two retries in case of missing response from the server



Element	Description								
	device.								
<b>delay</b>	Time delay in milliseconds between the end of the last received frame and the starting of a new request. If set to 0, the new request will be issued as soon as the internal system is able to reschedule it.								
<b>Num of repeats</b>	Number of times a certain message will be sent to the controller before reporting the communication error status.								
<b>Transmission Mode</b>	<ul style="list-style-type: none"> <li>• <b>RTU</b>: use RTU mode</li> <li>• <b>ASCII</b>: use ASCII mode</li> </ul>  Note: When PLC network is active, all nodes will be configured with the same Transmission Mode.								
<b>PLC Models</b>	PLC model you are going to connect to. The selection influences the data range offset per each data type according to the specific PLC memory resources.								
<b>Comm...</b>	<p>If clicked displays the communication parameters setup dialog.</p>  <table border="1"> <thead> <tr> <th>Element</th><th>Parameter</th></tr> </thead> <tbody> <tr> <td><b>Port</b></td><td>Serial port selection. <ul style="list-style-type: none"> <li>• <b>COM1</b>: device PLC port.</li> <li>• <b>COM2</b>: computer/printer port.</li> </ul> </td></tr> <tr> <td><b>Baudrate, Parity, Data Bits, Stop bits</b></td><td>Serial line parameters.</td></tr> <tr> <td><b>Mode</b></td><td>Serial port mode. Available modes: <ul style="list-style-type: none"> <li>• <b>RS-232</b>.</li> <li>• <b>RS-485</b> (2 wires).</li> <li>• <b>RS-422</b> (4 wires).</li> </ul> </td></tr> </tbody> </table> <p>When using the controllers:</p>	Element	Parameter	<b>Port</b>	Serial port selection. <ul style="list-style-type: none"> <li>• <b>COM1</b>: device PLC port.</li> <li>• <b>COM2</b>: computer/printer port.</li> </ul>	<b>Baudrate, Parity, Data Bits, Stop bits</b>	Serial line parameters.	<b>Mode</b>	Serial port mode. Available modes: <ul style="list-style-type: none"> <li>• <b>RS-232</b>.</li> <li>• <b>RS-485</b> (2 wires).</li> <li>• <b>RS-422</b> (4 wires).</li> </ul>
Element	Parameter								
<b>Port</b>	Serial port selection. <ul style="list-style-type: none"> <li>• <b>COM1</b>: device PLC port.</li> <li>• <b>COM2</b>: computer/printer port.</li> </ul>								
<b>Baudrate, Parity, Data Bits, Stop bits</b>	Serial line parameters.								
<b>Mode</b>	Serial port mode. Available modes: <ul style="list-style-type: none"> <li>• <b>RS-232</b>.</li> <li>• <b>RS-485</b> (2 wires).</li> <li>• <b>RS-422</b> (4 wires).</li> </ul>								

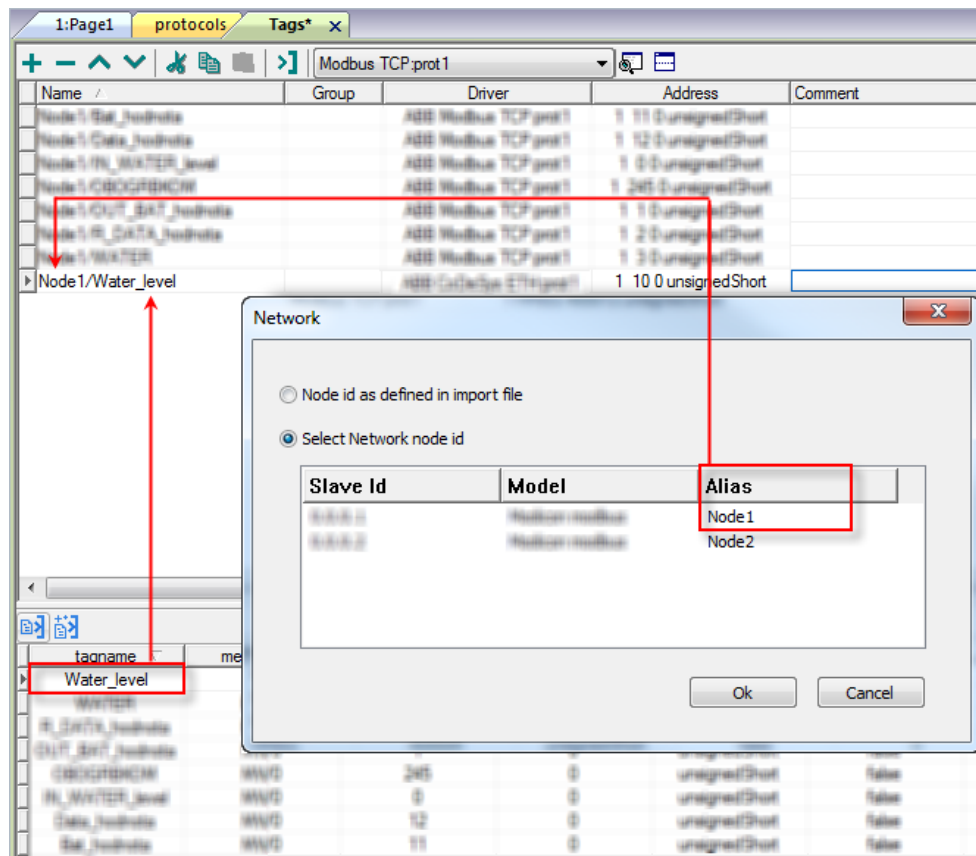
Element	Description
	<ul style="list-style-type: none"> <li>• ABB NextMove ES / ESB-2</li> <li>• ABB e100 Motion Product</li> <li>• ABB e150 Motion Product</li> </ul> <p>make sure that Parity has been set to "None"</p>
<b>PLC Network</b>	<p>The protocol allows the connection of multiple controllers to one operator panel. <b>PLC Network</b> must be selected to enable multiple connections.</p>  <p> Note: PLC model <b>Pluto Safety PLC</b> is available for compatibility reasons. If you need to connect to this PLC model, please select <b>ABB Pluto</b> protocol.</p>

## Adding an alias name to a protocol

Tag names must be unique at project level, however, the same tag names might need to be used for different controller nodes (for example when the HMI device is connected to two devices running the same application).

When creating a protocol you can add an alias name that will be added to tag names imported for this protocol.

In the example, the connection to a certain controller is assigned the name **Node1**. When tags are imported for this node, all tag names will have the prefix **Node1** making each of them unique at the network/project level.



Note: Aliasing tag names is only available for imported tags. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name.

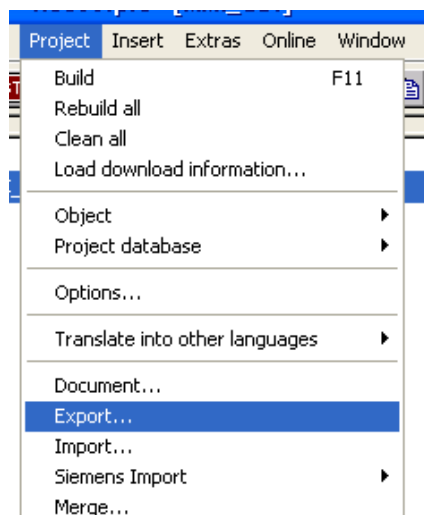
The Alias string is attached on the import. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are re-imported, all tags will be re-imported with the new prefix string.

## Exporting tags from a controller

The ABB controllers programming supports tag export in .exp format.

To export tags:

Select **Project> Export...**: an .exp file will be created.



## Importing tags

You may import tags from an .exp file exported from a controller. See ["Importing tags" on page 21](#).

## Node Override ID

The protocol provides the special data type Node Override ID which allows you to change at runtime the Modbus address defined for the HMI device. This memory type is an unsigned byte.

The node Override ID is initialized with the value of the node ID specified in the project at programming time.

Node Over-ride ID	Modbus operation
0	Communication with the controller is stopped. In case of write operation, the request will be transmitted without waiting for a reply.
1 to 254	It is interpreted as the value of the new node ID and is replaced for runtime operation.
255	Communication with the controller is stopped; no request messages are generated.



Note: Node Override ID value assigned at runtime is retained through power cycles.

## Communication status

Current communication status can be displayed using System Variables. See ["Communication variables" on page 66](#).

Codes supported for this communication driver:

Error	Cause	Action
<b>NAK</b>	The controller replies with a not acknowledge.	-
<b>Timeout</b>	A request is not replied within the specified timeout period.	Check if the controller is connected and properly configured to get network access.
<b>Line Error</b>	An error on the communication parameter setup is detected (parity, baud rate, data bits, stop bits).	Check if the communication parameter settings of the controller is compatible with the device communication setup.
<b>Invalid response</b>	The device did received a response with invalid format or contents from the controller .	Ensure the data programmed in the project are consistent with the controller resources.
<b>General Error</b>	Unidentifiable error. Should never be reported.	Contact technical support.

## Implementation details

The ABB Modbus RTU implementation supports only a subset of the Modbus standard RTU function codes.

Code	Function	Description
<b>01</b>	Read Coil Status	Reads multiple bits in the device Coil area
<b>02</b>	Read Input Status	Read the ON/OFF status of the discrete inputs (1x reference) in the slave
<b>03</b>	Read Holding Registers	Read multiple Registers

Code	Function	Description
04	Read Input Registers	Reads the binary contents of input registers (3x reference) in the slave
05	Force Single Coil	Forces a single Coil to either ON or OFF
06	Preset Single Register	Presets a value in a Register
16	Preset Multiple Registers	Presets value in multiple Registers



Note: Communication speed with controllers is supported up to 115200 baud.



Note: Floating point data format is IEEE standard compliant.

# ABB Modbus TCP

ABB Modbus TCP driver provides easy handling of the connection to the ABB controllers providing specific supports for PLC models and tag import facilities.

Various Modbus TCP-capable devices can be connected to the HMI device. To set-up your Modbus TCP device, please refer to the documentation you have received with the device.

The implementation of the protocol operates as a Modbus TCP client.

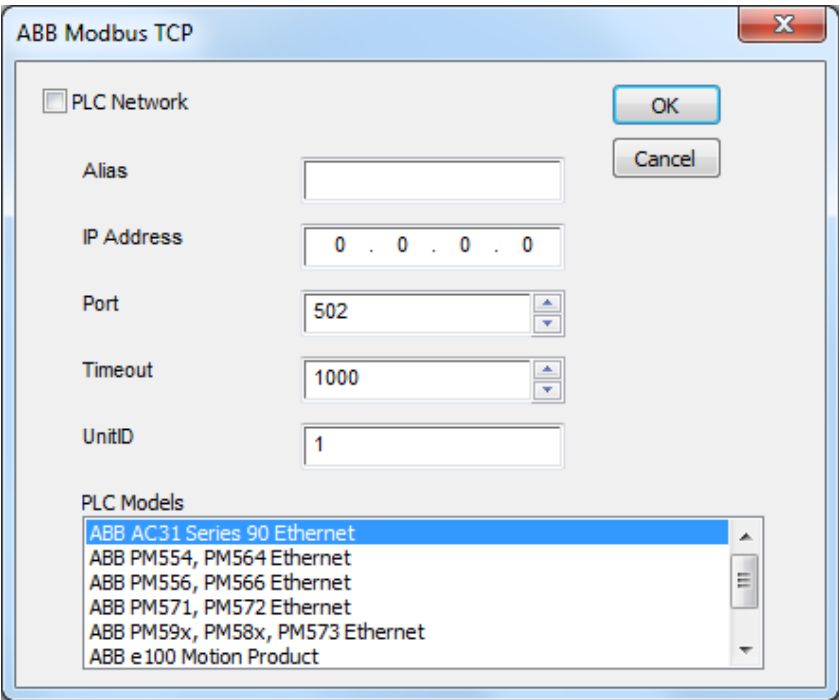
## Adding a protocol

To configure the protocol:

- 1. In the **Config** node double-click **Protocols**.
- 2. To add a driver, click **+**: a new line is added.
- 3. Select the protocol from the **PLC** list.

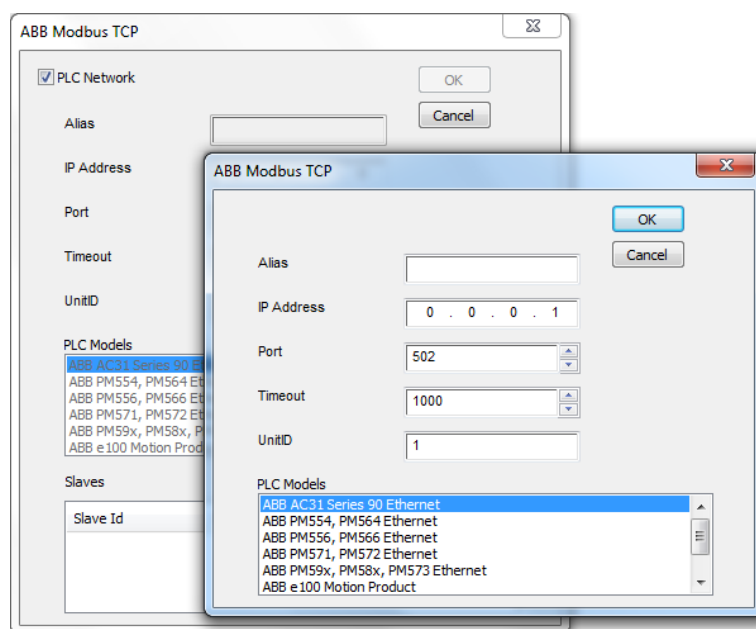
The driver configuration dialog is displayed.

## Protocol Editor settings



Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP Address	Ethernet IP address of the controller.

Element	Description
<b>Port</b>	Port number used by the Modbus TCP driver. The default value can be changed when the communication goes through routers or Internet gateways where the default port number is already in use.
<b>Timeout</b>	Time delay in milliseconds between two retries in case of missing response from the server device.
<b>UnitID</b>	Usually used when communicating over Ethernet-to-serial gateways and then interpreted as the Slave ID. This value is simply copied into the Unit Identifier field of the Modbus TCP communication frame. This is rarely used and in most cases can be left zero.
<b>PLC Models</b>	PLC model you are going to connect to. The selection influences the data range offset per each data type according to the specific PLC memory resources.
<b>PLC Network</b>	IP address for all controllers in multiple connections. <b>PLC Network</b> check box must be selected to enable multiple connections.



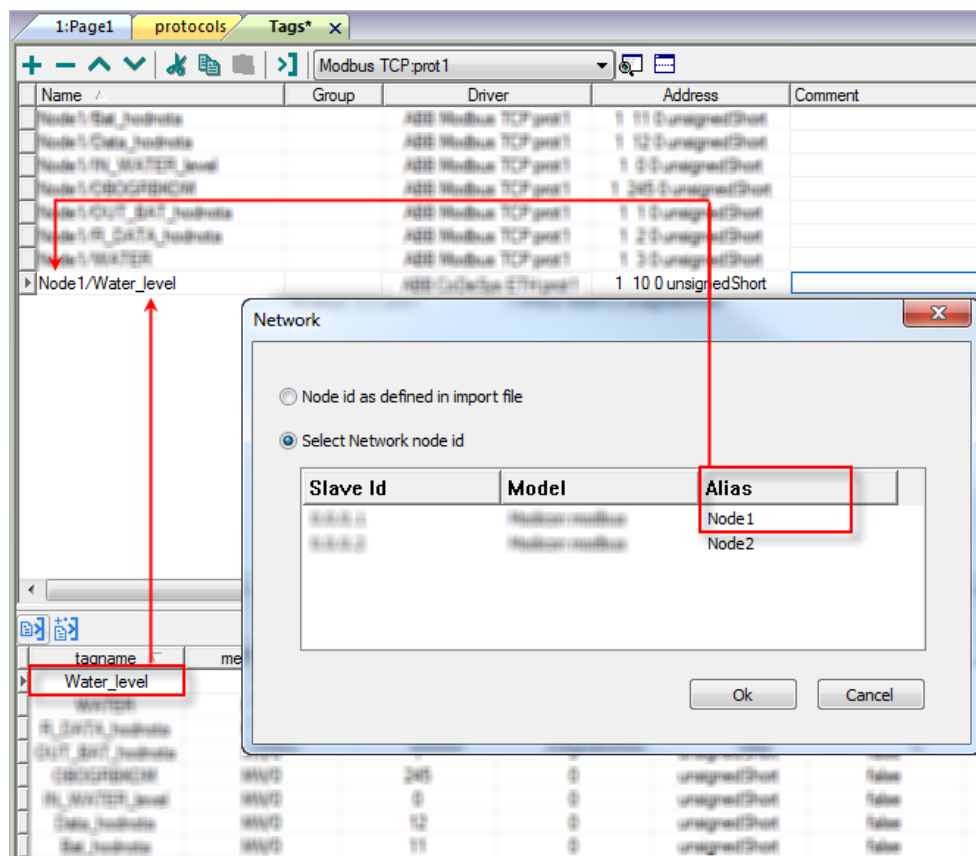
## Adding an alias name to a protocol

Tag names must be unique at project level, however, the same tag names might need to be used for different controller nodes (for example when the HMI device is connected to two devices running the same application).

When creating a protocol you can add an alias name that will be added to tag names imported for this protocol.

In the example, the connection to a certain controller is assigned the name **Node1**. When tags are imported for this node, all tag names will have the prefix **Node1** making each of them unique at the network/project level.





Note: Aliasing tag names is only available for imported tags. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name.

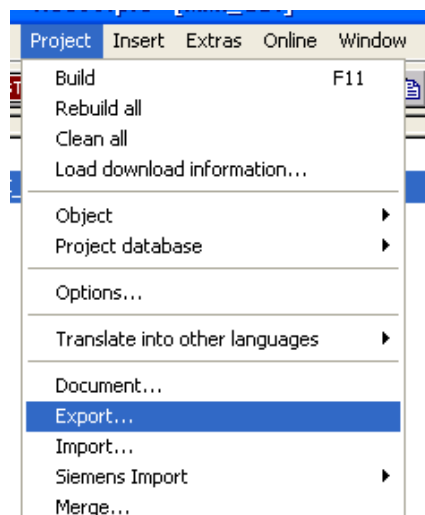
The Alias string is attached on the import. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are re-imported, all tags will be re-imported with the new prefix string.

## Exporting tags from the controller

The ABB controllers programming supports tag export in .exp format.

To export tags:

Select **Project> Export...**: an .exp file will be created.



## Importing tags

You may import tags from an .exp file exported from a controller. See ["Importing tags" on page 21](#).

## Node Override IP

The protocol provides the special data type Node Override IP which allows you to change the IP address of the target controller at runtime.

This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

Node Override IP	Modbus operation
<b>0.0.0.0</b>	Communication with the controller is stopped, no request frames are generated any-more.
<b>Different from 0.0.0.0</b>	It is interpreted as node IP override and the target IP address is replaced runtime with the new value.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override IP variable.



Note: Node Override IP values assigned at runtime are retained through power cycles.

## Communication status

Current communication status can be displayed using system variables. See "[Communication variables](#)" on page 66.

Codes supported for this communication driver:

Error	Cause	Action
<b>NAK</b>	The controller replies with a not acknowledge.	-
<b>Timeout</b>	A request is not replied within the specified timeout period.	Check if the controller is connected and properly configured to get network access.
<b>Invalid response</b>	The device did received a response with invalid format or contents from the controller.	Check if the data programmed in the project are consistent with the controller resources.
<b>General Error</b>	Unidentifiable error.	Contact technical support.

## Implementation details

The ABB Modbus TCP supports only a subset of the standard Modbus TCP function codes.

Code	Function	Description
<b>01</b>	Read Coil Status	Reads multiple bits in the device Coil area
<b>02</b>	Read Input Status	Read the ON/OFF status of the discrete inputs (1x reference) in the slave
<b>03</b>	Read Holding Registers	Read multiple Registers
<b>04</b>	Read Input Registers	Reads the binary contents of input registers (3x reference) in the slave
<b>05</b>	Force Single Coil	Forces a single Coil to either ON or OFF

---

Code	Function	Description
06	Preset Single Register*	Presets a value in a Register
16	Preset Multiple Registers*	Presets value in multiple Registers

## ABB Pluto

The HMI devices can be connected to a Modbus network as the network master using this generic driver.

This specific implementation of the Modbus RTU driver provides easy handling of the connections to the ABB controllers providing specific support for ABB Pluto Safety PLC and tag import facilities.

### Adding a protocol

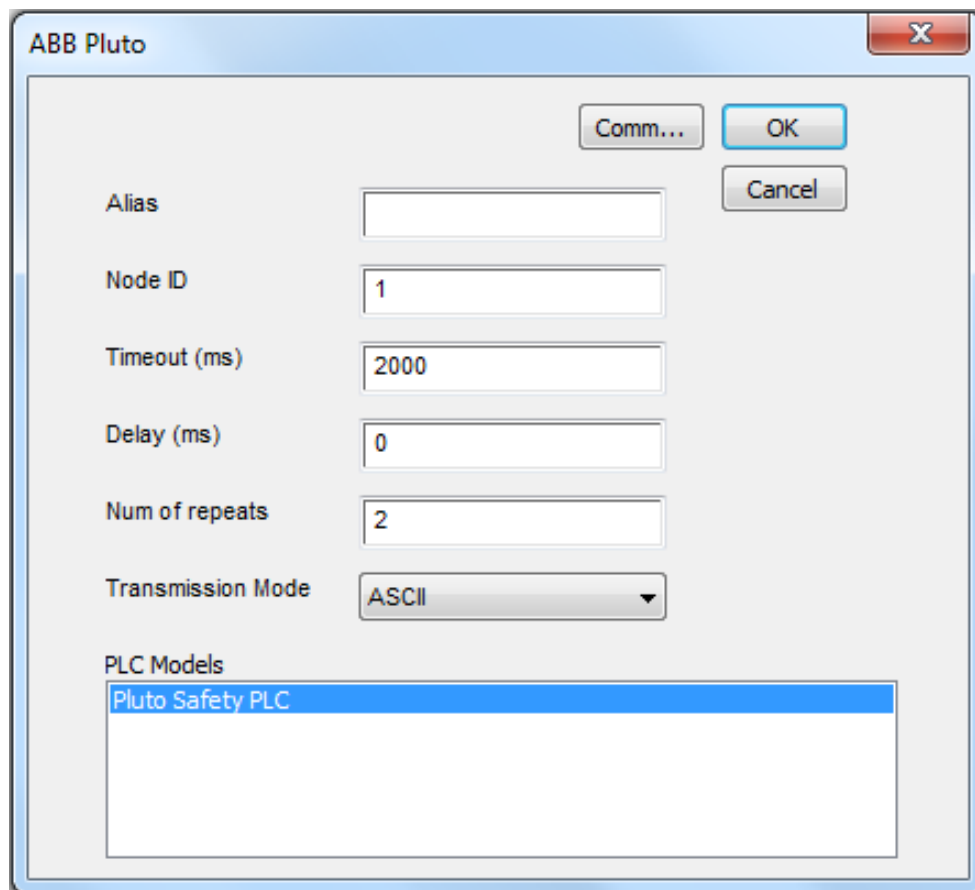
To configure the protocol:

1. In the **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The driver configuration dialog is displayed.

### Protocol Editor settings

The driver configuration dialog is shown in the following figure.



The screenshot shows the 'ABB Pluto' configuration dialog box. It has a title bar with a close button (X). Inside the dialog, there are three buttons at the top right: 'Comm...', 'OK', and 'Cancel'. Below these are several input fields and a dropdown menu:

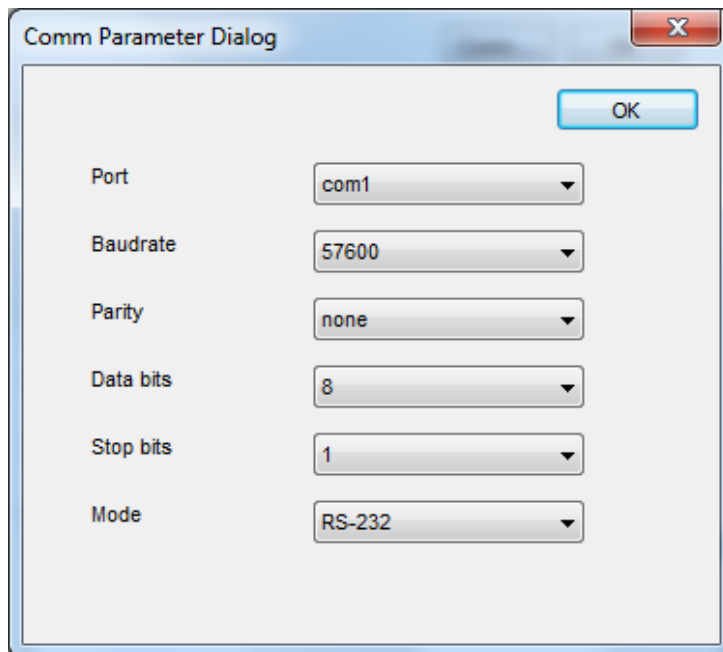
- Alias**: An empty text input field.
- Node ID**: A text input field containing the value '1'.
- Timeout (ms)**: A text input field containing the value '2000'.
- Delay (ms)**: A text input field containing the value '0'.
- Num of repeats**: A text input field containing the value '2'.
- Transmission Mode**: A dropdown menu currently set to 'ASCII'.

Below these fields is a section titled 'PLC Models' which contains a list box. The list box has one item, 'Pluto Safety PLC', which is currently selected and highlighted in blue.

Element	Description
<b>Alias</b>	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
<b>Node ID</b>	Modbus node of the slave device.
<b>Timeout (ms)</b>	Time delay in milliseconds between two retries in case of missing response from the slave device.
<b>Delay (ms)</b>	Time delay in milliseconds between the end of the last received frame and the starting of a new request. If set to 0, the new request will be issued as soon as the internal system is able to reschedule it.
<b>Num of repeats</b>	<p>Number of times a certain message will be sent to the controller before reporting the communication error status.</p> <p>When set to 1 the panel will report the communication error if the response to the first request packet is not correct.</p>
<b>Transmission Mode</b>	<ul style="list-style-type: none"><li>• <b>RTU</b>: use RTU mode</li><li>• <b>ASCII</b>: use ASCII mode</li></ul>

Element	Description
<b>PLC Models</b>	PLC model you are going to connect to. The selection influences the data range offset per each data type according to the specific PLC memory resources.

**Comm...** If clicked displays the communication parameters setup dialog.




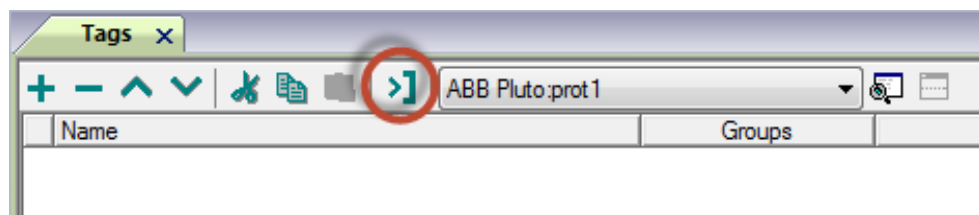
Element	Description
<b>Port</b>	Serial port selection. <ul style="list-style-type: none"> <li>• <b>COM1</b>: device PLC port.</li> <li>• <b>COM2</b>: computer/printer port (if available)</li> </ul>
<b>Baudrate, Parity, Data bits, Stop bits</b>	Serial line parameters.
<b>Mode</b>	Serial port mode. Available modes: <ul style="list-style-type: none"> <li>• <b>RS-232</b>.</li> <li>• <b>RS-485</b> (2 wires).</li> <li>• <b>RS-422</b> (4 wires).</li> </ul>

## Tag import

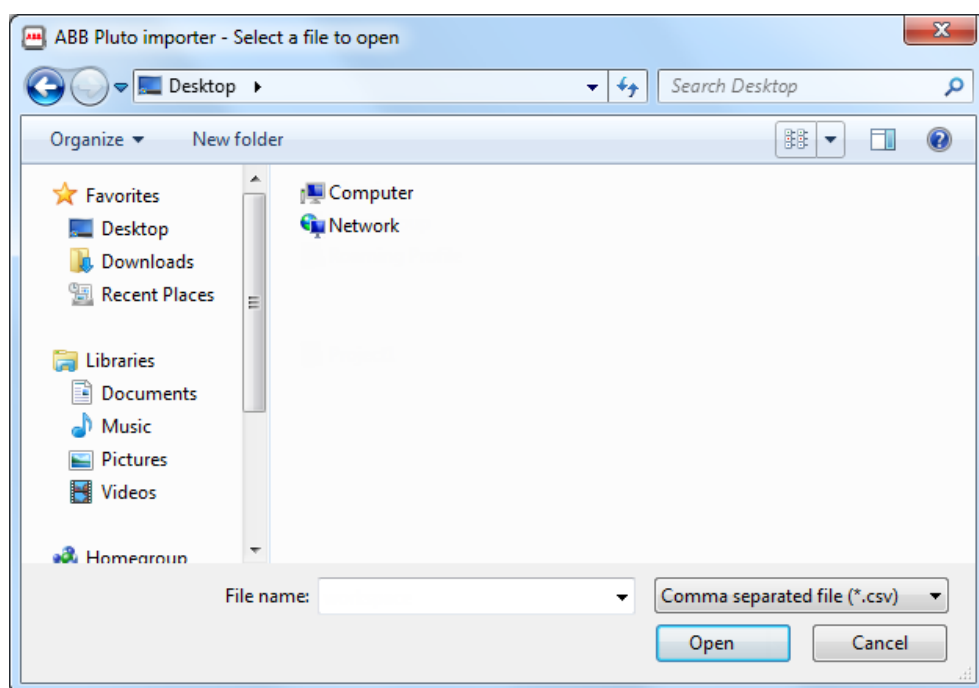
ABB Pluto driver supports tag import.


The ABB Pluto Safety PLC programming suite allows to export tags in .csv format.

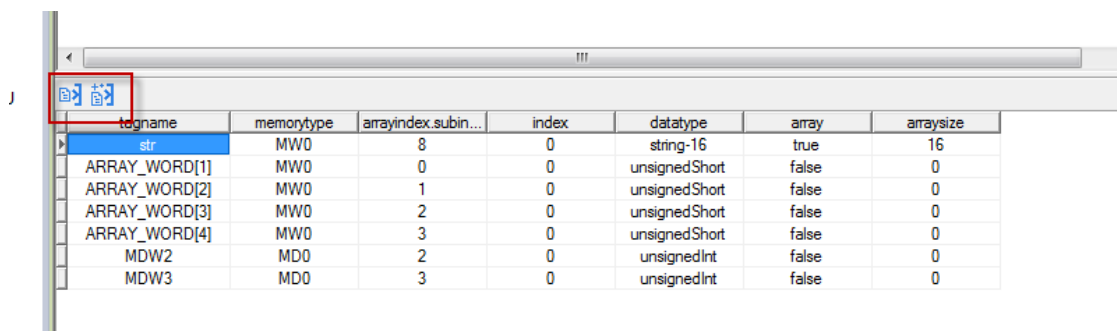
1. In the Tag Editor select the driver.
2. Click the **Import Tags**  button to start the import.



3. Locate the .csv file and confirm.



4. To import tags, select one or more tags in the .csv file and click the  **Import tag** button: tags are copied to the project.



tagname	memorytype	arrayindex.subin...	index	datatype	array	arraysize
str	MW0	8	0	string-16	true	16
ARRAY_WORD[1]	MW0	0	0	unsignedShort	false	0
ARRAY_WORD[2]	MW0	1	0	unsignedShort	false	0
ARRAY_WORD[3]	MW0	2	0	unsignedShort	false	0
ARRAY_WORD[4]	MW0	3	0	unsignedShort	false	0
MDW2	MD0	2	0	unsignedInt	false	0
MDW3	MD0	3	0	unsignedInt	false	0

## Node Override ID

The protocol provides the special data type Node Override ID which allows you to change at runtime the Modbus address defined for the HMI device. This memory type is an unsigned byte.

The node Override ID is initialized with the value of the node ID specified in the project at programming time.



Node Over-ride ID	Modbus operation
0	Communication with the controller is stopped. In case of write operation, the request will be transmitted without waiting for a reply.
1 to 254	It is interpreted as the value of the new node ID and is replaced for runtime operation.
255	Communication with the controller is stopped; no request messages are generated.



Note: Node Override ID value assigned at runtime is retained through power cycles.

## Communication status

Current communication status can be displayed using System Variables. See "[Communication variables](#)" on page 66.

Codes supported for this communication driver:

Error	Cause	Action
<b>NAK</b>	The controller replies with a not acknowledge.	-
<b>Timeout</b>	A request is not replied within the specified timeout period.	Check if the controller is connected and properly configured to get network access.
<b>Line Error</b>	An error on the communication parameter setup is detected (parity, baud rate, data bits, stop bits).	Check if the communication parameter settings of the controller is compatible with the device communication setup.
<b>Invalid</b>	The device did received a response with invalid	Ensure the data programmed in the project are consistent

Error	Cause	Action
<b>response</b>	format or contents from the controller .	with the controller resources.
<b>General Error</b>	Unidentifiable error. Should never be reported.	Contact technical support.

## Implementation details

This Modbus RTU implementation supports only a subset of the standard Modbus function codes.

Code	Function	Description
<b>01</b>	Read Coil Status	Reads multiple bits in the device Coil area
<b>02</b>	Read Input Status	Read the ON/OFF status of the discrete inputs (1x reference) in the slave
<b>03</b>	Read Holding Registers	Read multiple Registers
<b>04</b>	Read Input Registers	Reads the binary contents of input registers (3x reference) in the slave
<b>05</b>	Force Single Coil	Forces a single Coil to either ON or OFF
<b>06</b>	Preset Single Register	Presets a value in a Register
<b>16</b>	Preset Multiple Registers	Presets value in multiple Registers



Note: Communication speed with controllers is supported up to 115200 baud.



Note: Floating point data format is IEEE standard compliant.

## BACnet

The BACnet communication driver has been designed to connect HMI devices to BACnet networks and supports IP and MS/TP communication.

The HMI device operates as a BACnet device.

### Adding a protocol

To configure the protocol:

1. In the **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The driver configuration dialog is displayed.

## Protocol Editor settings

**BACnet**

Comm... OK Cancel

Panel Device ID: 262000

Object Name: DEV262000

Description: HMI

Media: IP

Timeout (ms): 5000

Panel Node: 1

COV Lifetime (s): 60

Max Master: 127

Max Info Frames: 1

max MS/TP APDU: 480

max IP APDU: 1476

IP UDP Port: 47808

LocalIP: 192.168.40.12

PLC Models

- default

Element	Description
<b>Panel Device ID</b>	Identifies the HMI device in the network.
<b>Object</b>	BACnet Object Name for the HMI device.

Element	Description
<b>Name</b>	
<b>Description</b>	HMI device description, for documentation purposes.
<b>Media</b>	Type of communication of the protocol. <ul style="list-style-type: none"> <li>• <b>MS/TP</b>: Master-Slave/Token-Passing communication (RS-485).</li> <li>• <b>IP</b>: based on standard UDP/IP communication.</li> </ul>
<b>Timeout (ms)</b>	Time delay in milliseconds between two retries in case of missing response from the BACnet device.
<b>Panel Node *</b>	MS/TP address. Physical device address on the link; it is not passed through routers.
<b>COV Life-time (s)</b>	Desired lifetime of the subscription in seconds before the it shall be automatically cancelled.. A value of zero indicates an indefinite lifetime, without automatic cancellation.
<b>Max Maste *</b>	Highest allowable address for master nodes. Must be less than or equal to 127.
<b>Max Info Frames *</b>	Maximum number of information frames the node may send before it must pass the token. Max Info Frames may have different values on different nodes and may be used to allocate more or less of the available link bandwidth to particular nodes.
<b>Max MS/TP APDU *</b>	Max MS/TP APDU length protocol
<b>Max IP APDU **</b>	Max IP APDU length protocol
<b>IP UDP Port **</b>	Port number for IP communication.
<b>Local IP **</b>	IP Address of the network adapter to use for protocol. Not required if the device has only one Ethernet adapter

Element	Description
<b>PLC Models</b>	Reserved for future use.

**Comm...** \* If clicked displays the communication parameters setup dialog.

Element	Description
<b>Port</b>	Communication port.
<b>Baudrate, Parity, Data bits, Stop bits</b>	Communication parameters.
<b>Mode</b>	Communication mode. Available modes: <ul style="list-style-type: none"> <li>• <b>RS-232</b></li> <li>• <b>RS-485</b></li> <li>• <b>RS-422</b></li> </ul>



Note \*: Available only if media is set to **MS/TP**.



Note \*\*: Available only if media is set to **IP**.

## Tag definition

Path: **ProjectView** > **Config** > double-click **Tags**

1. To add a tag, click **+**: a new line is added.
2. Select **BACnet** from the **Driver** list: the tag definition dialog is displayed.

BACnet

BACnet

Object Type

Device ID

Data Type

Arraysize

Conversion

Object Instance

Object Property

Array Index

Write Priority

☐ COV

Analog Value

508

float

0

3000015

85

-1

0

OK

Cancel

Apply

Help

Element	Description
Object Type	Type of BACnet object to be referenced. Available object types: <ul style="list-style-type: none"><li>• Device</li><li>• Analog Input</li><li>• Analog Output</li><li>• Analog Value</li><li>• Binary Input</li><li>• Binary Output</li><li>• Binary Value</li><li>• Multi-state Input</li><li>• Multi-state Output</li><li>• Multi-state Value</li><li>• Integer Value</li><li>• Positive Integer Value</li><li>• Large Analog Value</li></ul>
Device ID	ID of the device containing the object.
Data Type	Data type for display presentation. Available data types: <ul style="list-style-type: none"><li>• boolean</li><li>• int</li><li>• unsignedInt</li><li>• float</li></ul>

Element	Description																																	
	<ul style="list-style-type: none"><li>• <b>double</b></li><li>• <b>string</b></li><li>• <b>binary</b></li><li>• <b>boolean[]</b></li></ul> <p>These data types are data types as defined in the software.</p> <p>The equivalence with BACnet data types is shown in the table:</p> <table><tr><th>BACnet data type</th><th>Software data type</th><th>Notes</th></tr><tr><td><b>BOOLEAN</b></td><td>Boolean</td><td>-</td></tr><tr><td><b>INTEGER</b></td><td>Int</td><td>-</td></tr><tr><td><b>UNSIGNED_INTEGER</b></td><td>unsignedInt</td><td>-</td></tr><tr><td><b>REAL</b></td><td>Float</td><td>-</td></tr><tr><td><b>BIT_STRING</b></td><td>boolean-x</td><td><b>x = size</b></td></tr><tr><td><b>CHARACTER_STRING</b></td><td>string-x</td><td><b>x = size</b></td></tr><tr><td><b>OCTET_STRING</b></td><td>binary-x</td><td><b>x = size</b></td></tr><tr><td><b>DATE</b></td><td>int or unsignedInt</td><td>-</td></tr><tr><td><b>TIME</b></td><td>int or unsignedInt</td><td>-</td></tr><tr><td><b>BACnetObjectIdentifier</b></td><td>int or unsignedInt</td><td><b>Use conversions instance and objType for proper display</b></td></tr></table>	BACnet data type	Software data type	Notes	<b>BOOLEAN</b>	Boolean	-	<b>INTEGER</b>	Int	-	<b>UNSIGNED_INTEGER</b>	unsignedInt	-	<b>REAL</b>	Float	-	<b>BIT_STRING</b>	boolean-x	<b>x = size</b>	<b>CHARACTER_STRING</b>	string-x	<b>x = size</b>	<b>OCTET_STRING</b>	binary-x	<b>x = size</b>	<b>DATE</b>	int or unsignedInt	-	<b>TIME</b>	int or unsignedInt	-	<b>BACnetObjectIdentifier</b>	int or unsignedInt	<b>Use conversions instance and objType for proper display</b>
BACnet data type	Software data type	Notes																																
<b>BOOLEAN</b>	Boolean	-																																
<b>INTEGER</b>	Int	-																																
<b>UNSIGNED_INTEGER</b>	unsignedInt	-																																
<b>REAL</b>	Float	-																																
<b>BIT_STRING</b>	boolean-x	<b>x = size</b>																																
<b>CHARACTER_STRING</b>	string-x	<b>x = size</b>																																
<b>OCTET_STRING</b>	binary-x	<b>x = size</b>																																
<b>DATE</b>	int or unsignedInt	-																																
<b>TIME</b>	int or unsignedInt	-																																
<b>BACnetObjectIdentifier</b>	int or unsignedInt	<b>Use conversions instance and objType for proper display</b>																																
<b>Arraysize</b>	Array size for the variable.																																	
<b>Conversion</b>	Value conversion for presentation.																																	
<b>Object Instance</b>	BACnet ID of the object to be referenced.																																	
<b>Object Property</b>	Numeric value of the property to be referenced (example: the value 85 means <i>present-value</i> for most standard objects).																																	
<b>Array Index</b>	Index for subscribing elements in BACnet arrays. <ul style="list-style-type: none"><li>• -1 means read all elements</li><li>• 0 to n means read the specified element</li></ul>																																	
<b>Write Priority</b>	Write requests priority level. The value is in the range 1-16. 0 is interpreted as 16.																																	
<b>COV</b>	Enable the Change Of Value notification.																																	



## Clear/Set Priority

The system offers actions for a more flexible handling of Write Priority.

Action	Description
<b>BACnetClearPriority</b>	<p>Clears the priority array at the position associated to the BACnet tag passed as parameter.</p> <p>This action has immediate effect on the BACnet device.</p>
<b>BACnetClearAllPriorities</b>	<p>Clears all positions in the priority array.</p> <p>This action has immediate effect on the BACnet device.</p>
<b>BACnetSetPriority</b>	<p>Overrides the Write Priority value configured in the BACnet tag definition.</p> <p>This action has two parameters:</p> <ul style="list-style-type: none"> <li>• <b>TagName</b>: name of the BACnet tag.</li> <li>• <b>TagPriority</b>: new value of Write Priority for the BACnet tag passed as parameter.</li> </ul> <p>This action only overrides the value of Write Priority in the BACnet tag definition and does not perform any communication with the BACnet device. Any write command that will be performed to the Present Value property of the BACnet device identified by the tag, will be performed using the new Write Priority value.</p> <p>The priority value will be valid until:</p> <ul style="list-style-type: none"> <li>• A new call to the BACnetSetPriority action changes it.</li> <li>• The HMI device is restarted. The value of WritePriority defined in the project is valid in this case.</li> </ul>

## Importing tags

You may import BACnet object information from BACnet EDE (Engineering Data Exchange) files. The EDE file must have the .csv extension.

The importer uses the characters “,” and “;” as delimiters. They are considered as reserved characters and you cannot use them in file name.

Tags will be created using the string specified in the column **object-name** of the EDE file. The importer will add the device ID as a prefix to avoid duplication of tag names.

To import tags from the EDE file see ["Importing tags" on page 21](#).



Note: The importer will ask to locate the StateTexts files. Click **Cancel** to ignore.

## Communication status

Current communication status can be displayed using system variables. See ["Communication variables" on page 66](#).

Codes supported by this communication driver:

Error	Cause
Cannot bind to the device_id	Cannot establish communication with the Device ID provided for this tag.
Cannot read the property data type	The type of the property to write cannot be determined.
write conversion error	A conversion associated to this tag has failed.
Cannot write ICOM type .... BACnet type ....	A datatype selected for this tag is not compatible with the BACnet property to set.
Timeout on COV subscription	A request for COV subscription for this tag has timed out.
Timeout on waiting COV update	A COV notification has not been received for this tag within timeout.
Can't get COV for this property	The selected property for COV notification is unsupported.
datagramItem conversion error	A conversion associated to a tag that is part of a datagram has failed.
Timeout waiting on response	No response for a request of read or write property within timeout.
datagram element ....., no data available	No data available for a tag that is part of datagram.
datagram element ....., Unsupported BACnet data type	Read datagram element is of unsupported BACnet type.
datagram element ....., can't convert BACnet type to ....	A Data Type selected for a tag which is part of a datagram is not compatible with the BACnet property to read.
No data in response	No data available for a tag.
Datagram element 'element_URI' error: 'error_class': error_code	The reading of indicated datagram element 'element_URI' was reported as error. The error descriptions <b>error_class</b> and <b>error_code</b> are included in the message.
datagram object does not match	The object of the received datagram item does not match the asked object.
datagram property does not match	The property of the received datagram item does not match the asked property.
BACnet abort: reason_of_abort	BACnet abort message was received. The reason of abort is given.
BACnet reject: reason_of_rejection	BACnet reject message was received. The reason of rejection is given.

Error	Cause
<b>BACnet error: error_class: error_code</b>	BACnet error message was received. The error description is given as combination of <b>error_class</b> and <b>error_code</b> .
<b>parameter 'parameter_name' out of range</b>	The protocol parameter <b>parameter_name</b> value is out of range.

## DEVICE object properties

A BACnet network scanner can detect properties when exploring the network and obtaining data from HMI device.

This are the supported DEVICE object properties:

Property	Description
<b>Object_Identifier</b>	BACnetObjectIdentifier
<b>Object_Name</b>	CharacterString
<b>Object_Type</b>	BACnetObjectType
<b>System_Status</b>	BACnetDeviceStatus
<b>Vendor_Name</b>	CharacterString
<b>Vendor_Identifier</b>	Unsigned16
<b>Model_Name</b>	CharacterString
<b>Firmware_Revision</b>	CharacterString
<b>Application_Software_Version</b>	CharacterString
<b>Protocol_Version</b>	Unsigned
<b>Protocol_Revision</b>	Unsigned
<b>Protocol_Services_Supported</b>	BACnetServicesSupported
<b>Protocol_Object_Types_Supported</b>	BACnetObjectTypesSupported
<b>Object_List</b>	BACnetARRAY[N]of BACnetObjectIdentifier
<b>Max_APDU_Length_Accepted</b>	Unsigned
<b>Segmentation_Supported</b>	BACnetSegmentation
<b>APDU_Timeout</b>	Unsigned
<b>Number_Of_APDU_Retries</b>	Unsigned
<b>Device_Address_Binding</b>	List of BACnetAddressBinding
<b>Database_Revision</b>	Unsigned

# ABB CODESYS Serial

The ABB CODESYS Serial communication driver has been specifically designed to support communication with Series 500 ABB controllers designed for IEC 61131-3 programming, based on the CODESYS V2.3 system.

## Adding a protocol


To configure the protocol:

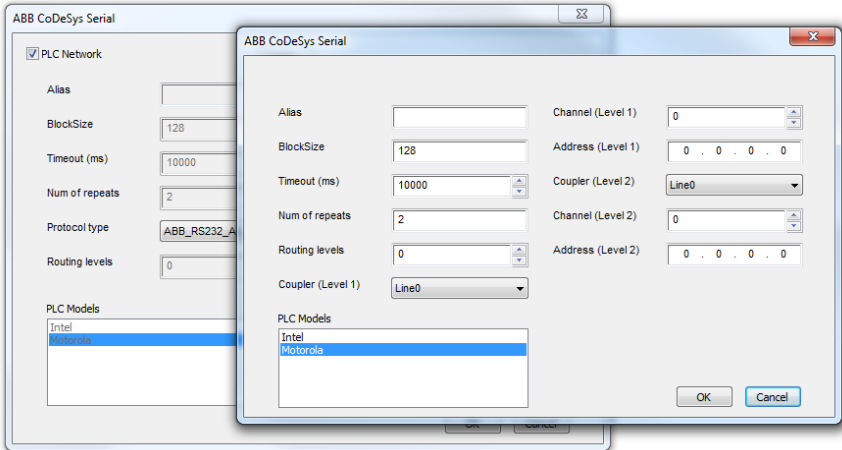
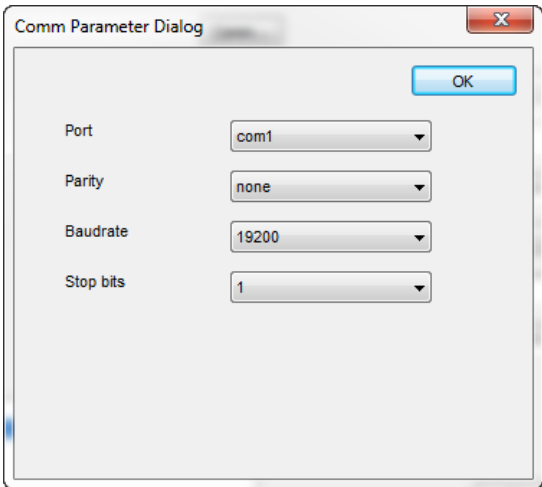



1. In the **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The driver configuration dialog is displayed.

## Protocol Editor settings

Element	Description
<b>Alias</b>	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
<b>BlockSize</b>	The max block size supported by your controller (limit is 1024 kB).
<b>Timeout (ms)</b>	Time delay in milliseconds between two retries in case of missing response from the server device.
<b>Num of repeats</b>	Number of times a certain message will be sent to the controller before reporting the communication error status.
<b>Protocol</b>	Two different protocol types available:

Element	Description
<b>type</b>	<ul style="list-style-type: none"> <li>• <b>Serial_RS232</b>: corresponds to the standard 3S driver.</li> <li>• <b>ABB_RS232_AC</b>: implements a specific variation of the standard Level 2 protocol with the additional use of a routing driver. Normally used to connect to PLCs via other PLCs acting as gateways.</li> </ul> <p> The ABB_RS232_AC protocol type requires the proper settings of the following additional parameters:</p> <ul style="list-style-type: none"> <li>• Routing Levels</li> <li>• Coupler (Level 1)</li> <li>• Channel (Level 1)</li> <li>• Address (Level 1)</li> <li>• Coupler (Level 2)</li> <li>• Channel (Level 2)</li> <li>• Address (Level 2)</li> </ul> <p>For detailed information see <i>AC500 and Control Builder</i> documentation, chapter <i>Programming interfaces to the AC500 used by the Control Builder</i>.</p>
<b>PLC Models</b>	The list allows selecting the PLC model you are going to connect to. The selection will influence the data range offset per each data type according to the specific PLC memory resources.

Element	Description								
<b>PLC Network</b>	<p>The protocol allows the connection to multiple controllers. To set-up multiple connections, check "PLC network" checkbox and enter the node ID per each slave you need to access.</p> 								
<b>Comm...</b>	<p>If clicked displays the communication parameters setup dialog.</p>  <table border="1"> <thead> <tr> <th>Element</th><th>Description</th></tr> </thead> <tbody> <tr> <td><b>Port</b></td><td> <p>Serial port selection.</p> <ul style="list-style-type: none"> <li>• <b>COM1</b>: device PLC port.</li> <li>• <b>COM2</b>: computer/printer port.</li> </ul> </td></tr> <tr> <td><b>Parity, Baudrate, Stop bits</b></td><td> <p>Serial line parameters.</p> <p> <b>Parity must be set to none for AC500.</b></p> </td></tr> <tr> <td><b>Mode</b></td><td> <p>Serial port mode. Available modes:</p> </td></tr> </tbody> </table>	Element	Description	<b>Port</b>	<p>Serial port selection.</p> <ul style="list-style-type: none"> <li>• <b>COM1</b>: device PLC port.</li> <li>• <b>COM2</b>: computer/printer port.</li> </ul>	<b>Parity, Baudrate, Stop bits</b>	<p>Serial line parameters.</p> <p> <b>Parity must be set to none for AC500.</b></p>	<b>Mode</b>	<p>Serial port mode. Available modes:</p>
Element	Description								
<b>Port</b>	<p>Serial port selection.</p> <ul style="list-style-type: none"> <li>• <b>COM1</b>: device PLC port.</li> <li>• <b>COM2</b>: computer/printer port.</li> </ul>								
<b>Parity, Baudrate, Stop bits</b>	<p>Serial line parameters.</p> <p> <b>Parity must be set to none for AC500.</b></p>								
<b>Mode</b>	<p>Serial port mode. Available modes:</p>								

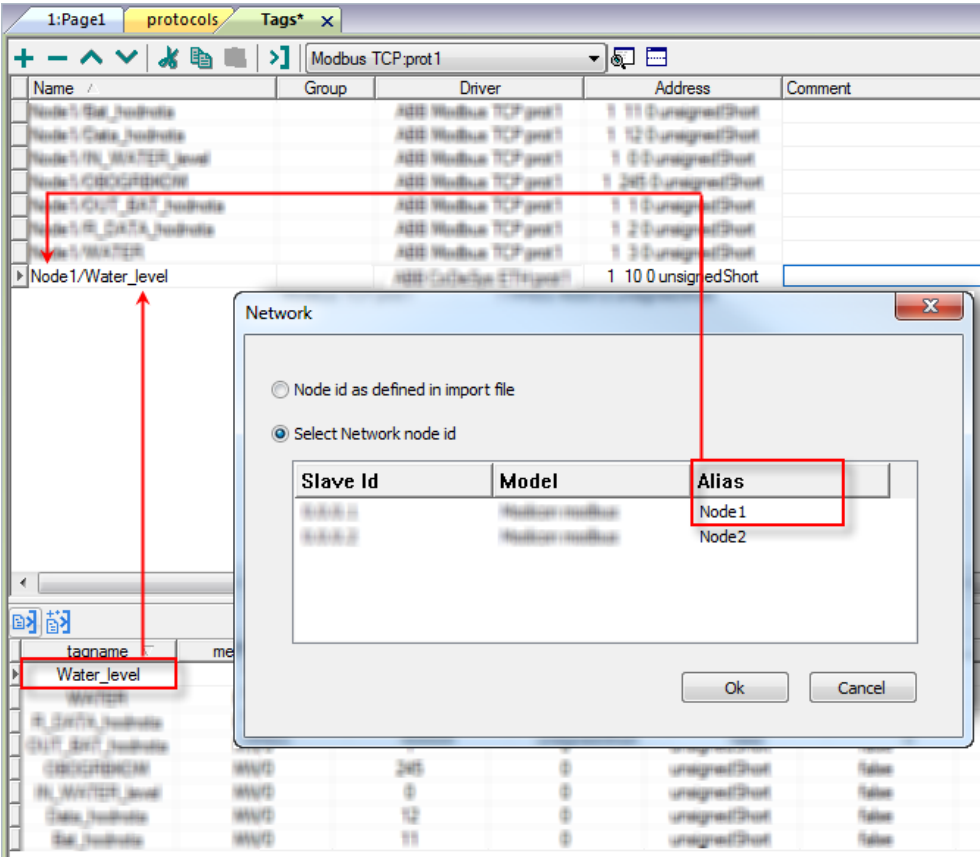
Element	Description	
	Element	Description
		<ul style="list-style-type: none"><li>• RS-232</li><li>• RS-485 (2 wires)</li><li>• RS-422 (4 wires)</li></ul>

Adding an alias name to a protocol

Tag names must be unique at project level, however, the same tag names might need to be used for different controller nodes (for example when the HMI device is connected to two devices running the same application).

When creating a protocol you can add an alias name that will be added to tag names imported for this protocol.

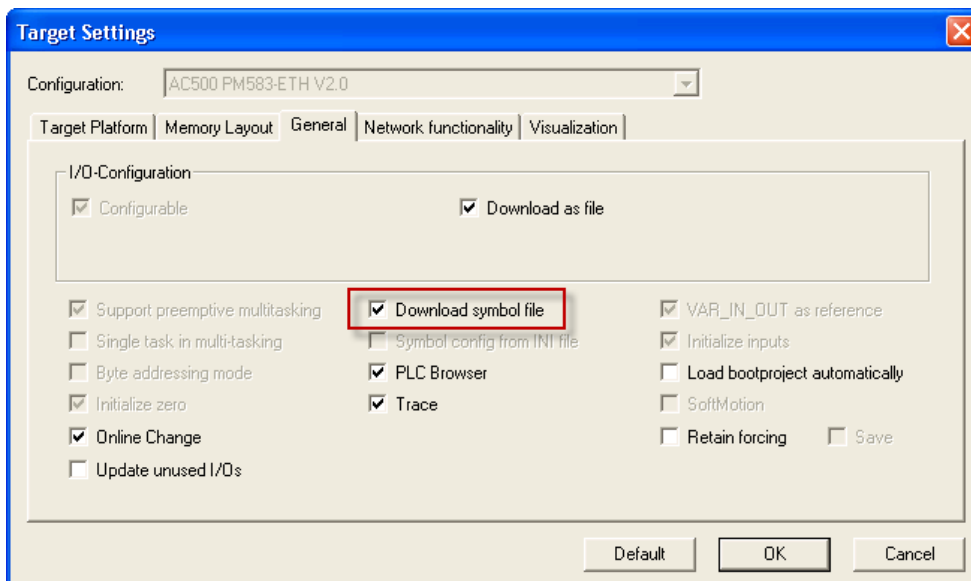
In the example, the connection to a certain controller is assigned the name **Node1**. When tags are imported for this node, all tag names will have the prefix **Node1** making each of them unique at the network/project level.




**Note:** Aliasing tag names is only available for imported tags. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name. The Alias string is attached on the import. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are re-imported, all tags will be re-imported with the new prefix string.

CODESYS software settings

When you create the project in CODESYS V2, select **Download symbol file** (*Target Settings > General*).

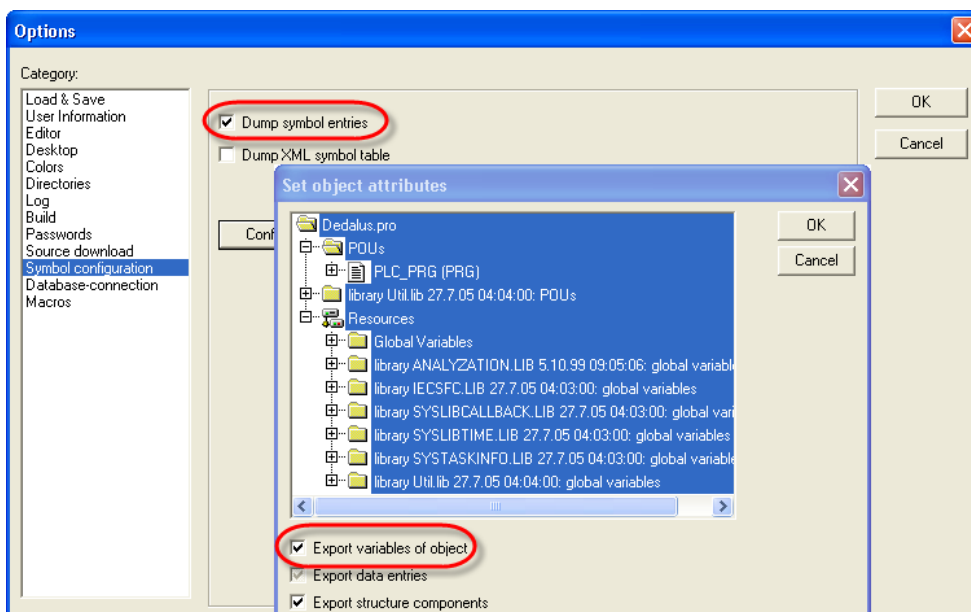


 Note: ABB CODESYS Serial driver supports the automatic symbol file (SDB) upload from the controller; any change in the tag offset due to new compilation on PLC software side does not require a symbol file re-import. The Tag file has to be re-imported only in case of tag renaming or addition of new tags.

## Exporting tags from the controller

When configuring PLC using the manufacturer's configuration software, enable Symbol file (file with .sym extension) creation under the CODESYS programming software:

1. In the **Project** menu, click **Options**.
2. Select **Symbol configuration**.
3. Select **Dump symbol entries**.
4. Click **Configure symbol file**: the **Set object attributes** dialog is displayed.
5. Select **Export variables of object**.
6. Click **OK**.






## Importing tags

You may import tags from a .sym file exported from a controller. See ["Importing tags" on page 21](#).

## Data types

The import module supports variables of standard data types and user defined data types.

Supported data types	<ul style="list-style-type: none"> <li>• BOOL</li> <li>• WORD</li> <li>• DWORD</li> <li>• INT</li> <li>• UINT</li> <li>• UDINT</li> <li>• DINT</li> <li>• STRING*</li> <li>• REAL</li> <li>• TIME</li> <li>• DATE &amp; TIME</li> </ul> <p>and 1-dimensional ARRAY of the types above.</p> <div data-bbox="323 1048 392 1122">  </div> <p>Note *: String length for a STRING variable in PLC should be max 80 characters. Declare a STRING variable either with a specified size (str: STRING(35) or default size (str: STRING) which is 80 characters.</p>
Unsupported data types	<ul style="list-style-type: none"> <li>• LWORD</li> <li>• LINT</li> <li>• LREAL</li> </ul>

## Limitations

This protocol does not support AC500 firmware version earlier than V2.0.

## Communication status

Current communication status can be displayed using system variables. See ["Communication variables" on page 66](#).

Codes supported for this communication driver:

Error	Cause and action
Symbols file not present	Check Symbol file and download again the PLC program
"tag" not present in Symbols files	Check if the Tag is present into the PLC project
Time out on Acknowledge	Controller didn't send acknowledge
Time out on last Acknowledge	Controller didn't send last acknowledge

---

Error	Cause and action
Time out on data receiving	Controller didn't reply with data
Connection timeout	Device not connected

# CODESYS V2 Ethernet

CODESYS V2 communication driver for Ethernet supports communication with controllers based on the CODESYS V2.3 version.

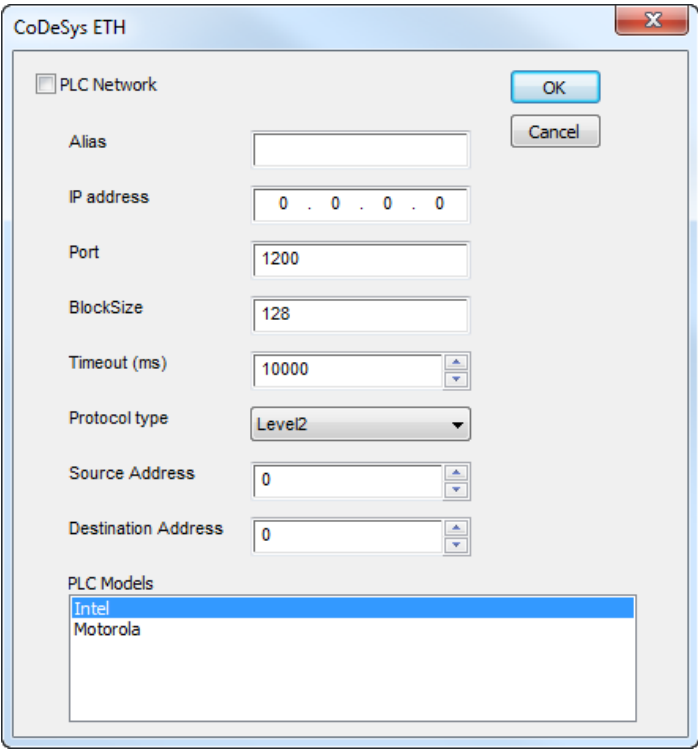
## Adding a protocol

To configure the protocol:

- 1. In the **Config** node double-click **Protocols**.
- 2. To add a driver, click **+**: a new line is added.
- 3. Select the protocol from the **PLC** list.

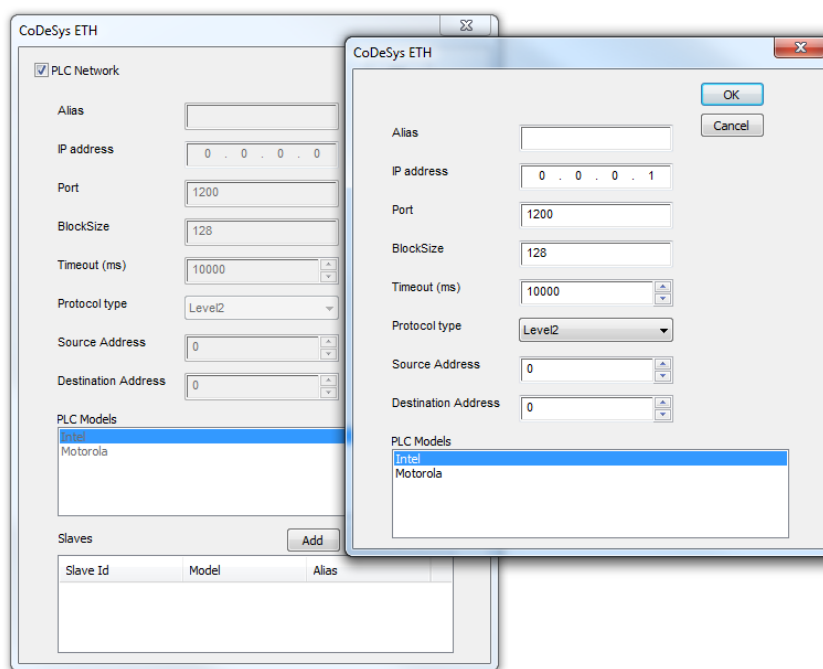
The driver configuration dialog is displayed.

## Protocol Editor settings



Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP address	Ethernet IP address of the controller.
Port	Port number used by the CODESYS V2 ETH driver. The default value is set to <b>1200</b> , which is also the default setting of CODESYS-based controllers.

Element	Description
<b>Block Size</b>	Maximum block size supported by your controller (limit is 1024 KB ).
<b>Timeout (ms)</b>	Time delay in milliseconds between two retries of the same message when communication fails.
<b>Protocol type</b>	Protocol variant to be used. Please make sure you check which protocol variant is supported by the CODESYS run-time you want to connect.
<b>Source Address, Destination Address</b>	Available only when <b>TCP/IP Level 2 Route</b> is selected in <b>Protocol Type</b> . The Destination is the node of the PLC and allows the protocol to read variables in a sub-network. The address is used to read variables when multiple PLCs are connected in a sub-network (serial network) but only one have the Ethernet interface.
<b>PLC Models</b>	Two PLC models are available. <ul style="list-style-type: none"> <li>• <b>Intel</b></li> <li>• <b>Motorola</b></li> </ul>
<b>PLC Network</b>	IP address for all controllers in multiple connections. <b>PLC network</b> check box must be selected to enable multiple connections.



*CODESYS V2 ETH driver supports connection to multiple controllers starting from version V1.60.*



Note: CODESYS V2 ETH driver is recommended when creating projects for the internal controller iPLC CODESYS. To use the CODESYS V2 ETH driver with iPLC, configure the IP address of the PLC as localhost (127.0.0.1).

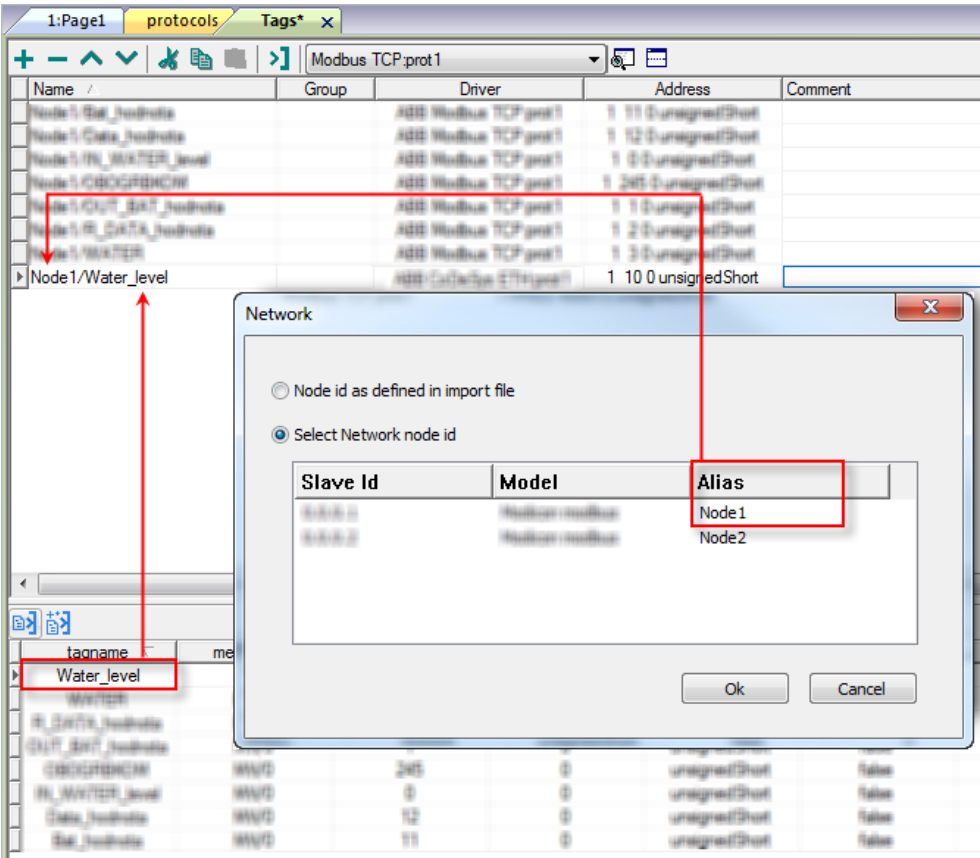
*iPLC CODESYS supports communication with CODESYS V2 ETH driver with symbol based support starting from V1.55 and above.*


## Adding an alias name to a protocol

Tag names must be unique at project level, however, the same tag names might need to be used for different controller nodes (for example when the HMI device is connected to two devices running the same application).

When creating a protocol you can add an alias name that will be added to tag names imported for this protocol.

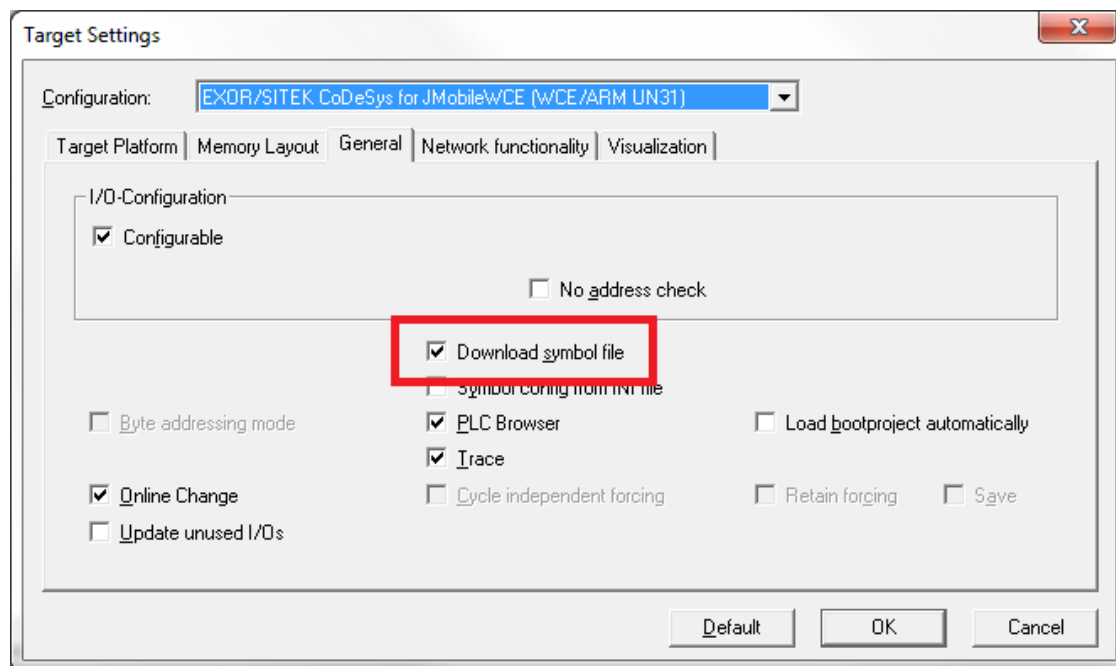
In the example, the connection to a certain controller is assigned the name **Node1**. When tags are imported for this node, all tag names will have the prefix **Node1** making each of them unique at the network/project level.



 Note: Aliasing tag names is only available for imported tags. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name.  
The Alias string is attached on the import. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are re-imported, all tags will be re-imported with the new prefix string.

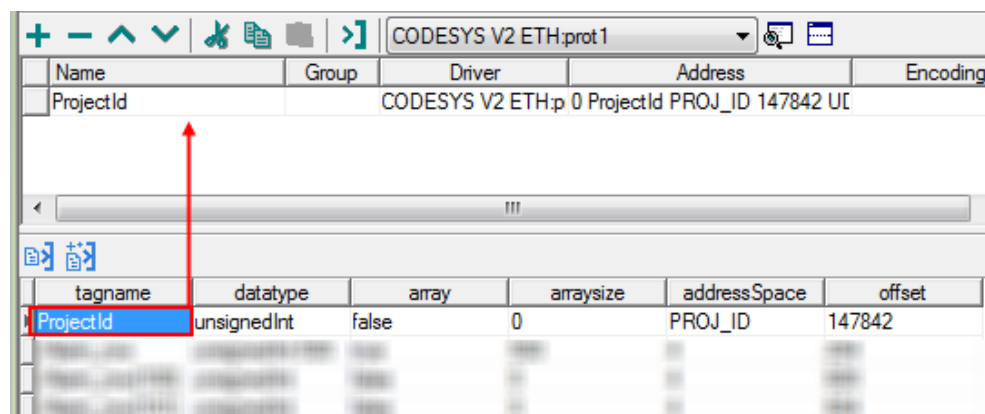
## CODESYS software settings

When creating the project in CODESYS, select **Download symbol file**.



**Note:** CODESYS V2 ETH communication driver supports the automatic symbol file (SDB) upload from the PLC; any change in the tag offset due to new compilation of the PLC program does not require a symbol file re-import. Tag file has to be re-imported only in case of tag rename or definition of new tags.

When the option **Download symbol file** is not available or cleared, the protocol can work only if the **ProjectId** tag is imported. If the tag offset changes because of a new compilation of the PLC program, the symbol file must be re-imported.

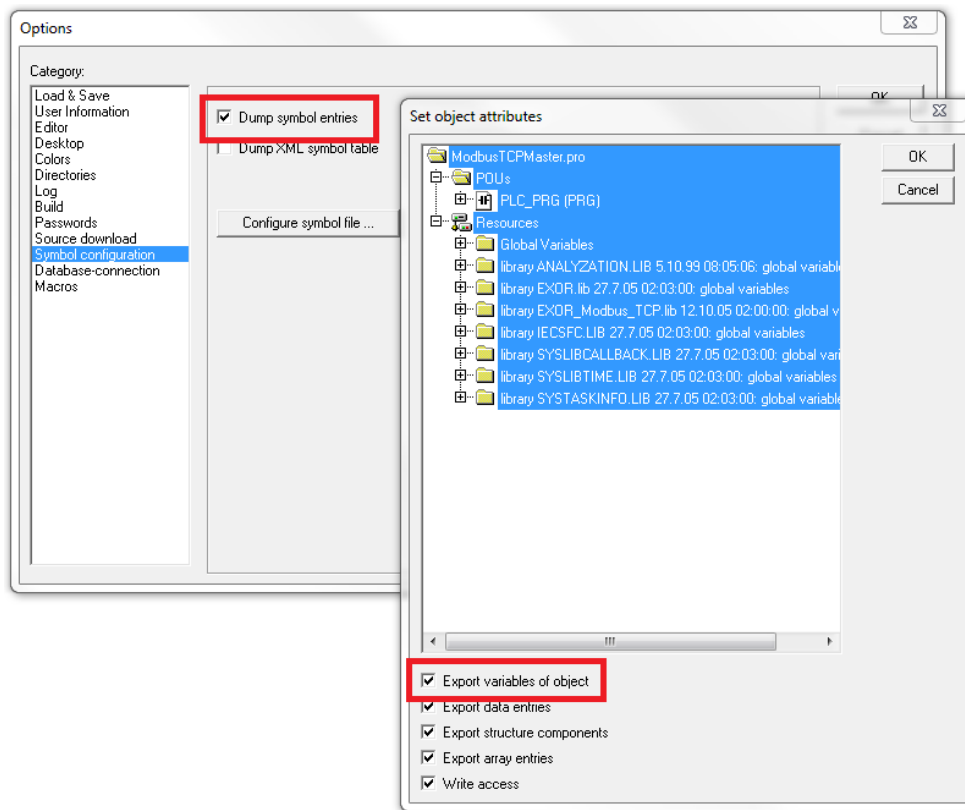


## Exporting tags from PLC

When configuring PLC using the manufacturer's configuration software, enable Symbol file (.sym extension) creation under the CODESYS programming software:

1. In the **Project** menu, click **Options**.
2. Click **Symbol configuration**.
3. Select **Dump symbol entries**.
4. Click **OK**.

**Note:** Click then **Configure symbol file...** and select **Export variables of object**. We recommend to clear the check box and re-select to be sure about the proper settings.

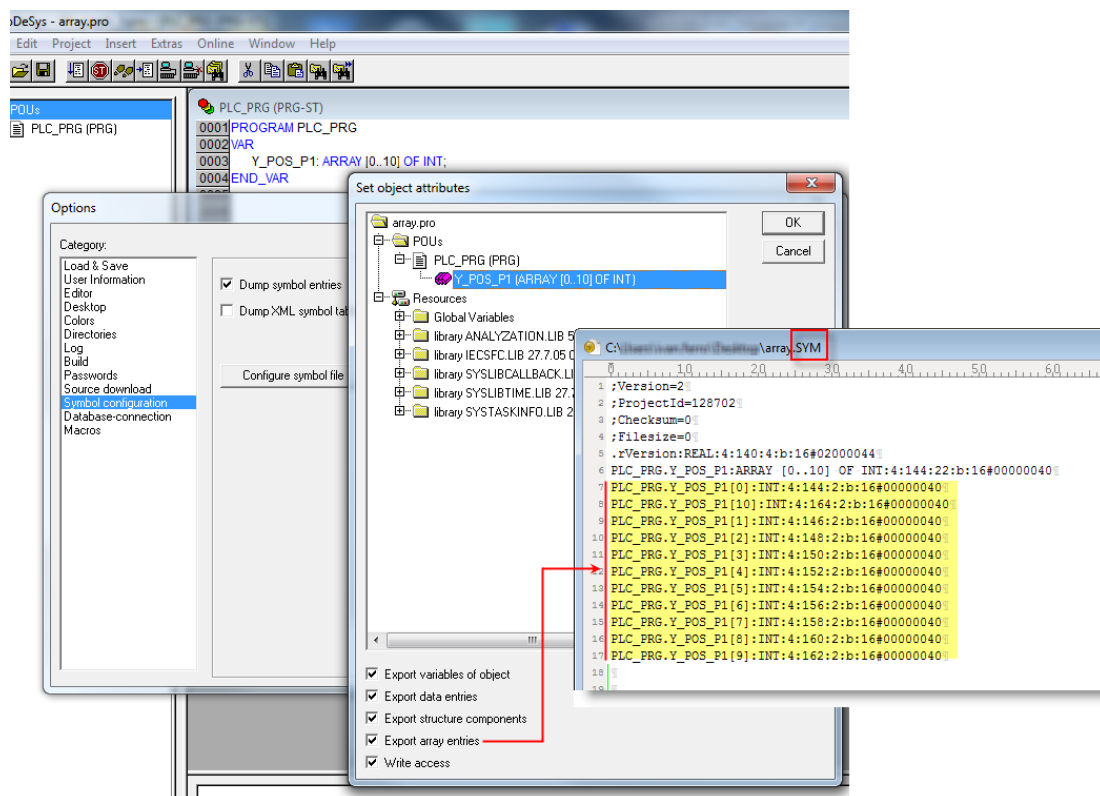


## Importing tags

To import tags from the symbol file see ["Importing tags" on page 21](#).

## Exporting tag arrays

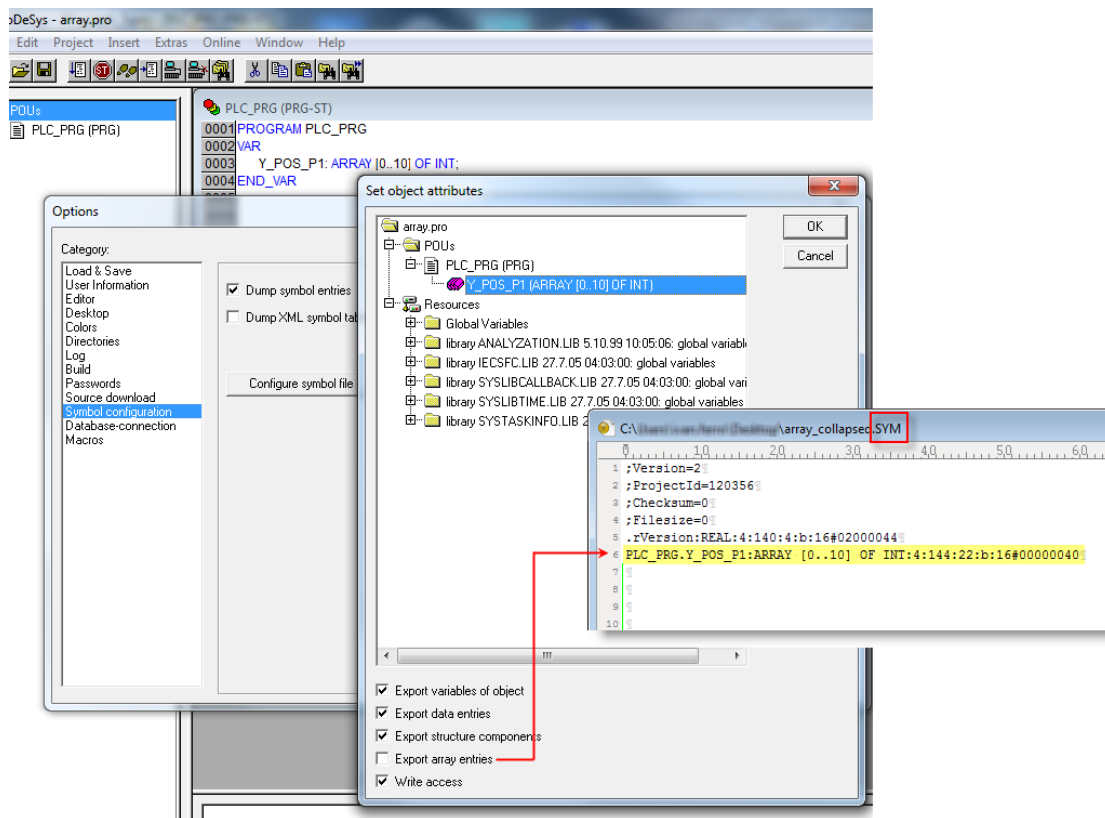
In CODESYS V2 program tag arrays are split into individual elements and one tag for each element is created. In the following example one array with 10 elements.



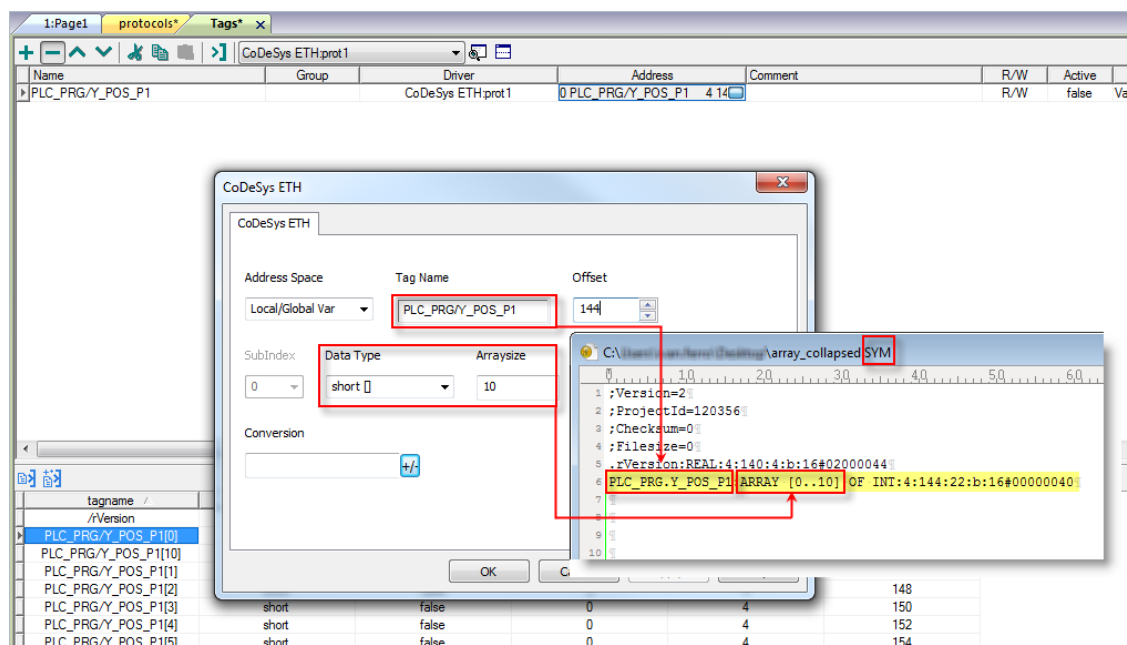
Note: If **Export array entries** is selected, a tag for each element will be created and exported into the .sym file. The entire tag list will be automatically imported into the Tag editor.

By clearing **Export array entries** only one tag for each one array can be created.

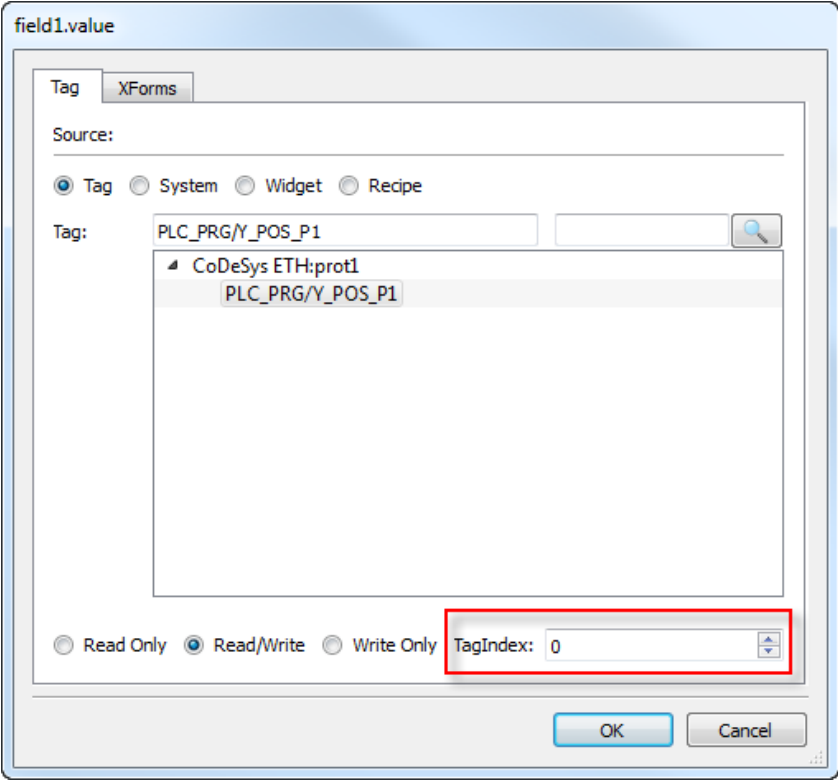




Note: When **Export array entries** has been cleared, only one tag is created and exported into the .sym file. The array is not automatically imported in the Tag editor and tags need to be manually configured in Tag editor.



All tag elements can be referenced in the editor using **TagIndex** in the **Attach to Tag** dialog.




Data types

The import module supports variables of standard data types and user defined data types.

<b>Supported data types</b>	<ul style="list-style-type: none"><li>• BOOL</li><li>• WORD</li><li>• DWORD</li><li>• INT</li><li>• UINT</li><li>• UDINT</li><li>• DINT</li><li>• STRING *</li><li>• REAL</li><li>• TIME</li><li>• DATE &amp; TIME</li></ul>
<b>Unsupported data types</b>	<ul style="list-style-type: none"><li>• LWORD</li><li>• LINT</li><li>• LREAL</li></ul>

and 1-dimensional ARRAY of the types above.

 Note \*: String length for a STRING variable in PLC should be max 80 characters. Declare a STRING variable either with a specified size (str: STRING(35) or default size (str: STRING) which is 80 characters.

## Node Override IP

The protocol provides the special data type Node Override IP which allows you to change the IP address of the target controller at runtime.

This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

Node Override IP	Modbus operation
<b>0.0.0.0</b>	Communication with the controller is stopped, no request frames are generated anymore.
<b>Different from 0.0.0.0</b>	It is interpreted as node IP override and the target IP address is replaced runtime with the new value.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override IP variable.



Note: Node Override IP values assigned at runtime are retained through power cycles.

## Communication status

Current communication status can be displayed using system variables. See ["Communication variables" on page 66](#).

Codes supported by this communication driver:

Error	Cause and action
<b>Symbols file not present</b>	Check Symbol file and download again the PLC program.
<b>“tag” not present in Symbols files</b>	Check if the Tag is present into the PLC project.

Error	Cause and action
Time out on Acknowledge	Controller didn't send acknowledge.
Time out on last Acknowledge	Controller didn't sent last ack.
Time out on data reciving	Controller does not reply with data.
Connection timeout	Device not connected.

# Modbus RTU

The operator panels can be connected to a Modbus network as the network master using this communication driver.

## Adding a protocol

To configure the protocol:

- 1. In the **Config** node double-click **Protocols**.
- 2. To add a driver, click **+**: a new line is added.
- 3. Select the protocol from the **PLC** list.

The driver configuration dialog is displayed.

## Protocol Editor settings

Modbus RTU

☐ PLC Network

Comm...OKCancel

Alias

Node ID

1

Timeout (ms)

2000

Delay (ms)

0

Num of repeats

2

Max read block

250

Max read bit block

2000

Write Holding Register

16

Write Coils

15

Transmission Mode


RTU

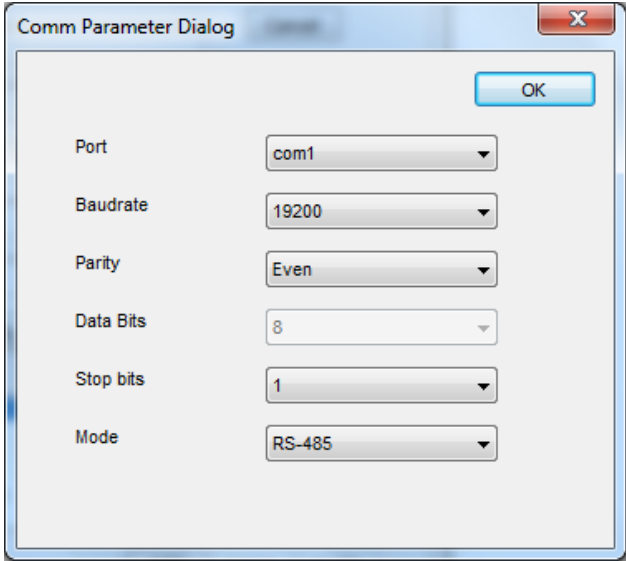
PLC Models

Modicon Modbus(1-based)

Generic Modbus(0-based)

Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
Node ID	Modbus node of the slave device.
Timeout (ms)	Time delay in milliseconds between two retries in case of missing response from the server

Element	Description
	device.
<b>Delay (ms)</b>	Time delay in milliseconds between the end of the last received frame and the starting of a new request. If set to 0, the new request will be issued as soon as the internal system is able to reschedule it.
<b>Num of repeats</b>	<p>Number of times a certain message will be sent to the controller before reporting the communication error status.</p> <p>When set to 1 the panel will report the communication error if the response to the first request packet is not correct.</p>
<b>Max read block</b>	Maximum length in bytes of a data block request. It applies only to read access of Holding Registers.
<b>Max read bit block</b>	Maximum length in bits of a block request. It applies only to read access of Input Bits and Output Coils.
<b>Write Holding Register</b>	<p>Modbus function for write operations to Holding Registers. Select between the function <b>06</b> (preset single register) and function <b>16</b> (preset multiple registers).</p> <p>If function <b>06</b> is selected, the protocol will always use function <b>06</b> for writing to the controller, even when writing to multiple consecutive registers.</p> <p>If function <b>16</b> is selected, the protocol will always use function <b>16</b> to write to the controller, even for a single register write request and the <b>Max read block size</b> parameter of the query is set to 2. The use of function <b>16</b> may result in higher communication performance.</p>
<b>Write Coils</b>	<p>Modbus function for write operations to Output Coils. Select between the function <b>05</b> (write single coil) and function <b>15</b> (write multiple coils).</p> <p>If Modbus function <b>05</b> is selected, the protocol will always use function <b>05</b> for writing to the controller, even when writing to multiple consecutive coils.</p> <p>If Modbus function <b>15</b> is selected, the protocol will always use function <b>15</b> to write to the controller, even for a single coil write request. The use of function <b>15</b> may result in higher communication performance.</p>
<b>Transmission Mode</b>	<ul style="list-style-type: none"> <li>• <b>RTU</b>: use RTU mode</li> <li>• <b>ASCII</b>: use ASCII mode</li> </ul> <p> Note: When PLC network is active, all nodes will be configured with the same Transmission Mode.</p>
<b>PLC Models</b>	<p>Two PLC models are available:</p> <ul style="list-style-type: none"> <li>• <b>Modicon Modbus</b> - addressing space starts from offsets 1 for all the memory types.</li> <li>• <b>Generic Modbus</b> - addressing space starts from offset 0 for all the memory types.</li> </ul>
<b>Comm...</b>	If clicked displays the communication parameters setup dialog.

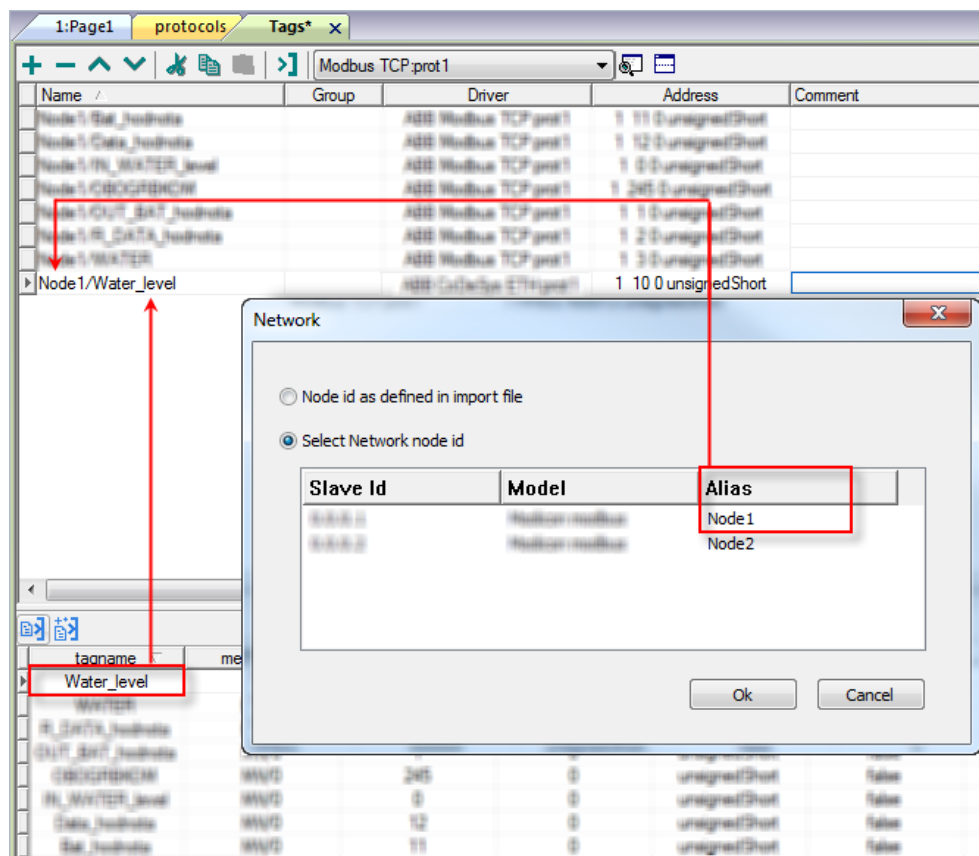
Element	Description								
	 <table border="1"> <thead> <tr> <th>Element</th><th>Parameter</th></tr> </thead> <tbody> <tr> <td><b>Port</b></td><td>Serial port selection. <ul style="list-style-type: none"> <li>• <b>COM1</b>: device PLC port.</li> <li>• <b>COM2</b>: computer/printer port on panels with 2 serial ports or optional Plug-In module plugged on Slot 1/2 for panels with 1 serial port on-board.</li> <li>• <b>COM3</b>: optional Plug-In module plugged on Slot 3/4 for panels with 1 serial port on-board.</li> </ul> </td></tr> <tr> <td><b>Baudrate, Parity, Data Bits, Stop bits</b></td><td>Serial line parameters.</td></tr> <tr> <td><b>Mode</b></td><td>Serial port mode. Available modes: <ul style="list-style-type: none"> <li>• <b>RS-232</b>.</li> <li>• <b>RS-485</b> (2 wires).</li> <li>• <b>RS-422</b> (4 wires).</li> </ul> </td></tr> </tbody> </table>	Element	Parameter	<b>Port</b>	Serial port selection. <ul style="list-style-type: none"> <li>• <b>COM1</b>: device PLC port.</li> <li>• <b>COM2</b>: computer/printer port on panels with 2 serial ports or optional Plug-In module plugged on Slot 1/2 for panels with 1 serial port on-board.</li> <li>• <b>COM3</b>: optional Plug-In module plugged on Slot 3/4 for panels with 1 serial port on-board.</li> </ul>	<b>Baudrate, Parity, Data Bits, Stop bits</b>	Serial line parameters.	<b>Mode</b>	Serial port mode. Available modes: <ul style="list-style-type: none"> <li>• <b>RS-232</b>.</li> <li>• <b>RS-485</b> (2 wires).</li> <li>• <b>RS-422</b> (4 wires).</li> </ul>
Element	Parameter								
<b>Port</b>	Serial port selection. <ul style="list-style-type: none"> <li>• <b>COM1</b>: device PLC port.</li> <li>• <b>COM2</b>: computer/printer port on panels with 2 serial ports or optional Plug-In module plugged on Slot 1/2 for panels with 1 serial port on-board.</li> <li>• <b>COM3</b>: optional Plug-In module plugged on Slot 3/4 for panels with 1 serial port on-board.</li> </ul>								
<b>Baudrate, Parity, Data Bits, Stop bits</b>	Serial line parameters.								
<b>Mode</b>	Serial port mode. Available modes: <ul style="list-style-type: none"> <li>• <b>RS-232</b>.</li> <li>• <b>RS-485</b> (2 wires).</li> <li>• <b>RS-422</b> (4 wires).</li> </ul>								
<b>PLC Network</b>	Multiple controllers can be connected to one HMI device. To set-up multiple connections, select <b>PLC network</b> and click <b>Add</b> to configure each slave								

## Adding an alias name to a protocol

Tag names must be unique at project level, however, the same tag names might need to be used for different controller nodes (for example when the HMI device is connected to two devices running the same application).

When creating a protocol you can add an alias name that will be added to tag names imported for this protocol.

In the example, the connection to a certain controller is assigned the name **Node1**. When tags are imported for this node, all tag names will have the prefix **Node1** making each of them unique at the network/project level.



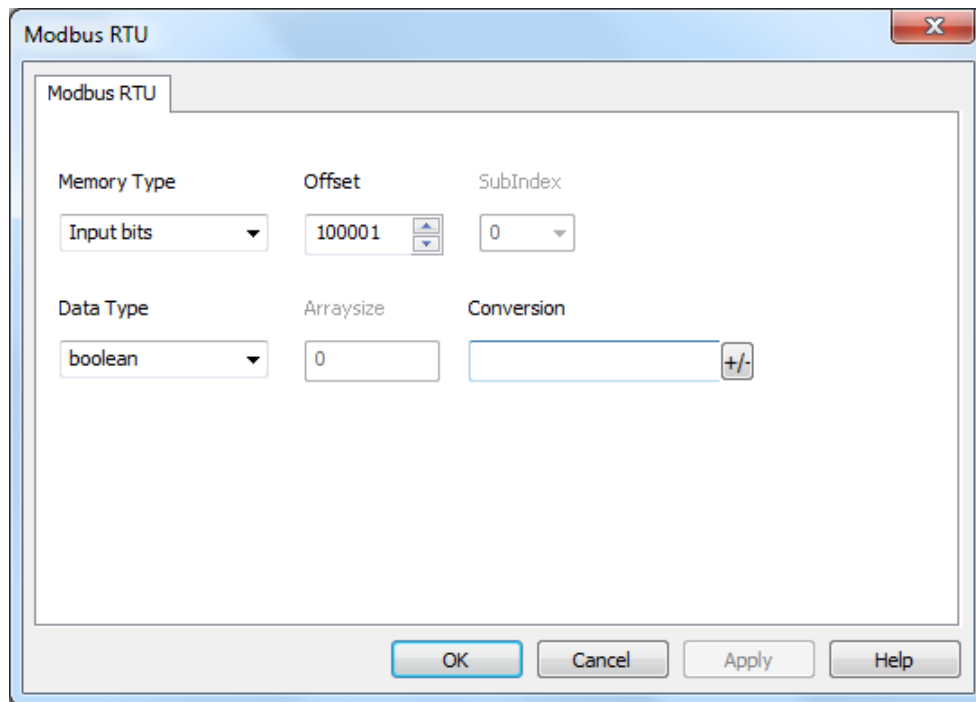
**Note:** Aliasing tag names is only available for imported tags. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name. The Alias string is attached on the import. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are re-imported, all tags will be re-imported with the new prefix string.

## Tag definition


Path: **ProjectView > Config > double-click Tags**


1. To add a tag, click **+**: a new line is added.
2. Select **Modbus RTU** from the protocol list: tag definition dialog is displayed.





The image shows a 'Modbus RTU' configuration dialog box. It has a title bar with a close button. Inside, there's a tab labeled 'Modbus RTU'. The dialog is divided into two sections. The top section has three fields: 'Memory Type' with a dropdown menu showing 'Input bits', 'Offset' with a numeric input field showing '100001' and up/down arrows, and 'SubIndex' with a dropdown menu showing '0'. The bottom section has three fields: 'Data Type' with a dropdown menu showing 'boolean', 'Arraysize' with a numeric input field showing '0', and 'Conversion' with an empty text field and a '+/-' button. At the bottom of the dialog are four buttons: 'OK', 'Cancel', 'Apply', and 'Help'.

Element	Description		
Memory Type	Modbus resource where tag is located:		
	Memory type	Modbus resource	
	Output Coils	Coils	
	Input Bits	Discrete Inputs	
	Input Registers	Input Registers	
	Holding Registers	Holding Registers	
Offset	Offset address where tag is located.		
	Memory Type	Offset	Resource Address
	Output coils	0 – 65535	0 – 65535
	Input bits	100000 – 165535	0 – 65535
	Input registers	300000 – 365535	0 – 65535
	Holding registers	400000 – 465535	0 – 65535
	 Note: Data in the table refer to PLC model “Generic Modbus (0-based)”.		
SubIndex	This allows resource offset selection within the register.		
Data Type	Available data types:		

Element	Description														
	<ul style="list-style-type: none"> <li>• <b>boolean</b></li> <li>• <b>byte</b></li> <li>• <b>short</b></li> <li>• <b>int</b></li> <li>• <b>unsignedByte</b></li> <li>• <b>unsignedShort</b></li> <li>• <b>unsignedInt</b></li> <li>• <b>float</b></li> <li>• <b>double</b></li> <li>• <b>string</b></li> <li>• <b>binary</b></li> </ul> <p>See <a href="#">"Data types" on page 28</a> for details.</p> <p> Note: To define arrays, select one of Data Type format followed by square brackets (byte[], short[]...).</p>														
<b>Arraysize</b>	Array size for the variable.														
<b>Conversion</b>	<p>Conversion to be applied.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td><b>Inv bits</b></td><td>Invert a bit (boolean tag).</td></tr> <tr> <td><b>Negate</b></td><td>Invert all bits.</td></tr> <tr> <td><b>AB -&gt; BA</b></td><td>Swap nibbles of a byte.</td></tr> <tr> <td><b>ABCD -&gt; CDAB</b></td><td>Swap bytes of a word.</td></tr> <tr> <td><b>ABCDEFGH -&gt; GHEFCBAB</b></td><td>Swap bytes of a double word.</td></tr> <tr> <td><b>BCD</b></td><td>BCD format conversion.</td></tr> </table>	Value	Description	<b>Inv bits</b>	Invert a bit (boolean tag).	<b>Negate</b>	Invert all bits.	<b>AB -&gt; BA</b>	Swap nibbles of a byte.	<b>ABCD -&gt; CDAB</b>	Swap bytes of a word.	<b>ABCDEFGH -&gt; GHEFCBAB</b>	Swap bytes of a double word.	<b>BCD</b>	BCD format conversion.
Value	Description														
<b>Inv bits</b>	Invert a bit (boolean tag).														
<b>Negate</b>	Invert all bits.														
<b>AB -&gt; BA</b>	Swap nibbles of a byte.														
<b>ABCD -&gt; CDAB</b>	Swap bytes of a word.														
<b>ABCDEFGH -&gt; GHEFCBAB</b>	Swap bytes of a double word.														
<b>BCD</b>	BCD format conversion.														

## Tag import file structure

This protocol supports the import of tag information when provided in .csv format according to the following format:

`NodeID, TagName, MemoryType, Address, DataFormat, ..., [Comment]`



Note: Fields in brackets are optional as well as fields between Data Format and Comment.

Field	Description
<b>NodeID</b>	Node the tag belongs to
<b>TagName</b>	Tag description
<b>MemoryType</b>	<ul style="list-style-type: none"> <li>• OUTP</li> <li>• INP</li> <li>• IREG</li> <li>• HREG</li> </ul>
<b>Address</b>	Offset compatible with Modbus notation
<b>DataFormat</b>	Data type in internal notation. See <a href="#">"Data types" on page 28</a>
<b>Comment</b>	Optional additional description.

## Tag file example

Example of .csv line:

```
2,Holding Register 1, HREG, 400001, unsignedShort,
```



Note: This line has no comment. When the Comment is missing, the comma as a terminator character is mandatory.

## Importing tags

See ["Importing tags" on page 21](#).

## Node Override ID

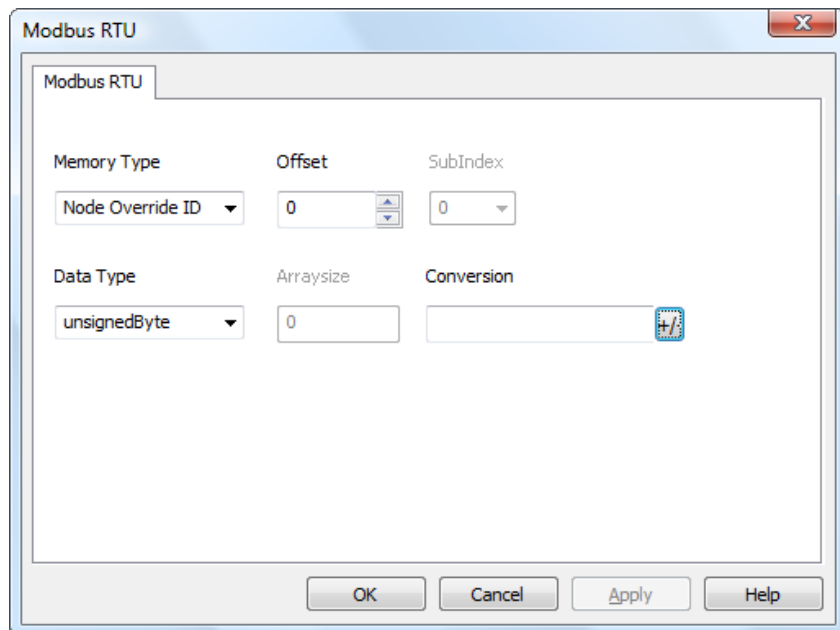
The protocol provides the special data type Node Override ID which allows you to change at runtime the Modbus address defined for the HMI device. This memory type is an unsigned byte.

The node Override ID is initialized with the value of the node ID specified in the project at programming time.

Node Over-ride ID	Modbus operation
<b>0</b>	Communication with the controller is stopped. In case of write operation, the request will be transmitted without waiting for a reply.
<b>1 to 254</b>	It is interpreted as the value of the new node ID and is replaced for runtime operation.
<b>255</b>	Communication with the controller is stopped; no request messages are generated.



Note: Node Override ID value assigned at runtime is retained through power cycles.



The image shows a 'Modbus RTU' configuration dialog box. It has a title bar with a close button. Inside, there's a tab labeled 'Modbus RTU'. The dialog is divided into two rows of settings. The first row contains 'Memory Type' (a dropdown menu showing 'Node Override ID'), 'Offset' (a numeric input field with '0'), and 'SubIndex' (a dropdown menu showing '0'). The second row contains 'Data Type' (a dropdown menu showing 'unsignedByte'), 'Arraysize' (a numeric input field with '0'), and 'Conversion' (a button with a small icon). At the bottom, there are four buttons: 'OK', 'Cancel', 'Apply', and 'Help'.

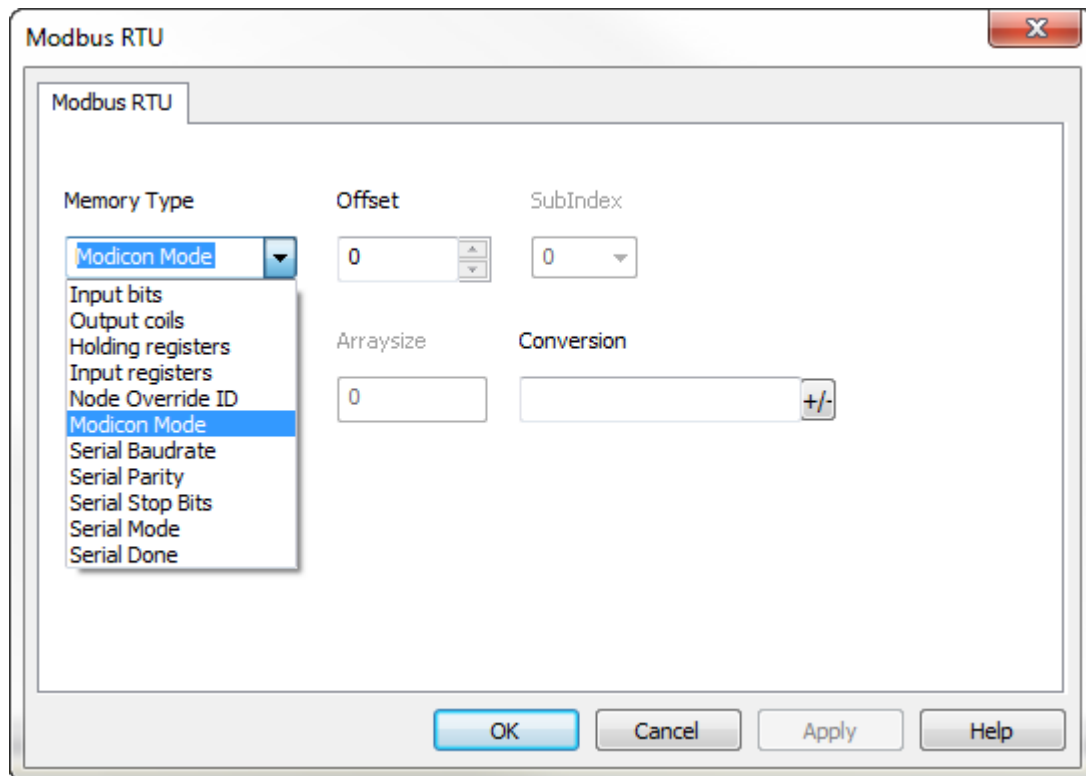
## Line parameters override

The protocol provide special data types to override the communication line parameters.

Line Parameters	Line Parameters Values
<b>Modicon Mode</b>	0=0-based, 1=1-based
<b>Serial Baudrate</b>	150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
<b>Serial Parity</b>	0=none, 1=parity even, 2=parity odd
<b>Serial Stop Bit</b>	1=1 stop bit, 2=2 stop bit
<b>Serial Mode</b>	0=RS-232, 1= RS-485, 2= RS-422
<b>Serial Done</b>	Set to 1 to overwrite the communication line parameters. The parameters are processed all together only when this variable is set to value 1



Note: Line parameter values assigned at runtime are retained through power cycles.



## Communication status

Current communication status can be displayed using System Variables. See "[Communication variables](#)" on page 66.

Codes supported for this communication driver:

Error	Cause	Action
<b>No response</b>	No reply within the specified timeout.	Check if the controller is connected and properly configured to get network access.
<b>Incorrect node address in response</b>	The device received a response with an invalid node address from the controller .	-
<b>The received message too short</b>	The device received a response with an invalid format from the controller .	-
<b>Incorrect writing data acknowledge</b>	The controller did not accept a write request.	Check if project data is consistent with the controller resources.

## Implementation details

The Modbus RTU implementation supports only a subset of the Modbus standard RTU function codes.

Code	Function	Description
01	Read Coil Status	Reads multiple bits in the device Coil area
02	Read Input Status	Read the ON/OFF status of the discrete inputs (1x reference) in the slave
03	Read Holding Registers	Read multiple Registers
04	Read Input Registers	Reads the binary contents of input registers (3x reference) in the slave
05	Force Single Coil	Forces a single Coil to either ON or OFF
06	Preset Single Register	Presets a value in a Register
16	Preset Multiple Registers	Presets value in multiple Registers



Note: Communication speed with controllers is supported up to 115200 baud.



Note: Floating point data format is IEEE standard compliant.

# Modbus RTU Server

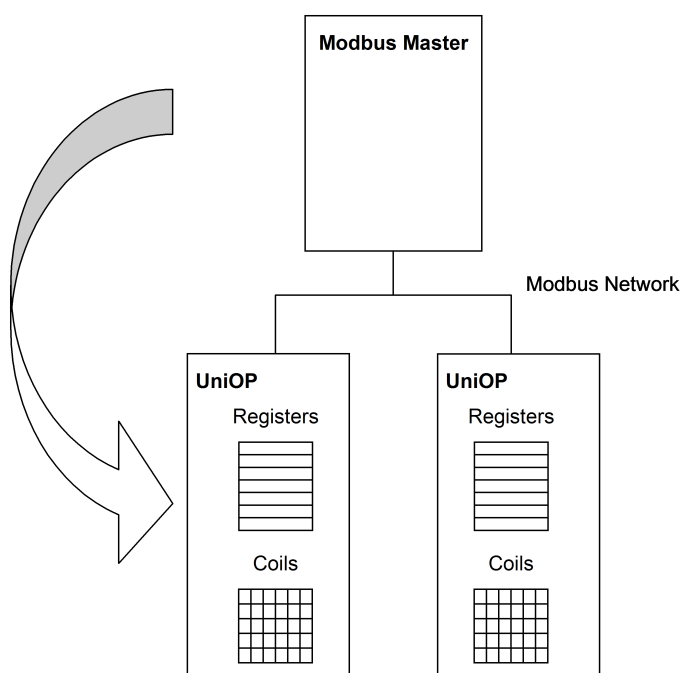
Modbus RTU Server communication driver allows connecting the HMI device as a slave in a Modbus RTU network. Standard Modbus messages are used for information exchange.

This approach allows connecting HMI devices to SCADA systems through the universally supported Modbus RTU communication protocol.

## Principle of operation

This communication driver implements a Modbus RTU slave unit in the HMI device. A subset of the complete range of Modbus function codes is supported. The available function codes allow data transfer between the master and the slave.

The following diagram shows the system architecture.



The HMI device is actually simulating the communication interface of a PLC: Coils and Registers are respectively boolean and 16 bit integers.

The device always access data in its internal memory. Data can be transferred to and from the Modbus Master only on initiative of the Master itself.

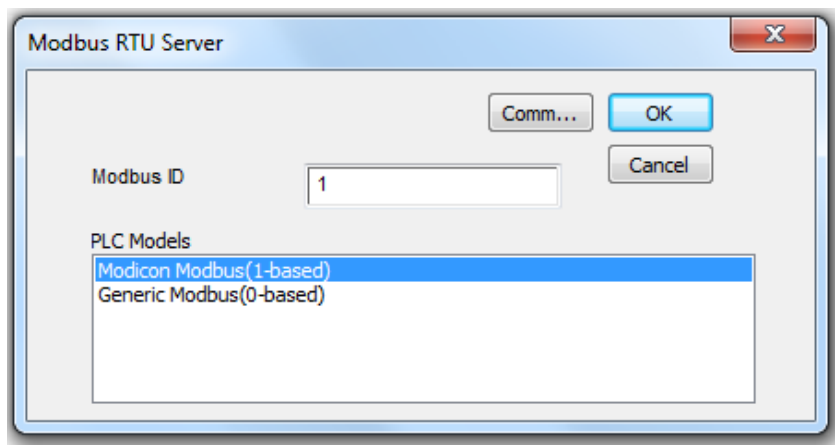
## Adding a protocol


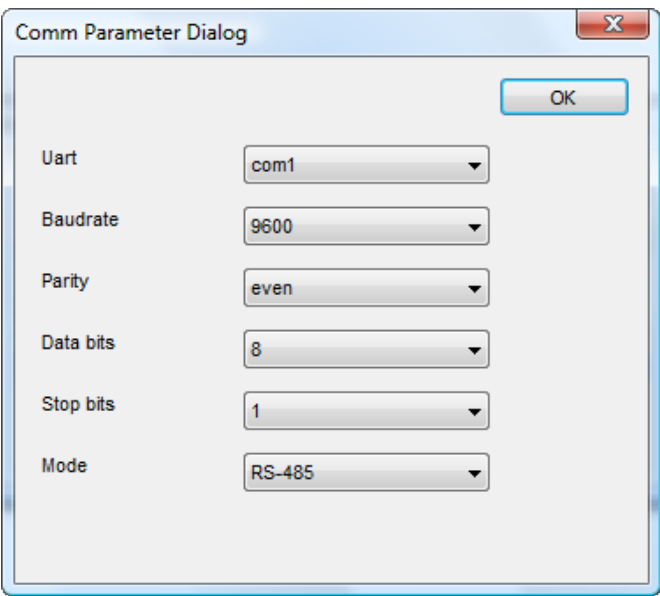
To configure the protocol:

1. In the **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The driver configuration dialog is displayed.

## Protocol Editor settings



Element	Description
<b>Modbus ID</b>	Modbus node ID. Every Modbus server device in the network must have its own Modbus ID.
<b>PLC Models</b>	<p>Allows you to select between two models of Modbus RTU Server.</p> <ul style="list-style-type: none"> <li><b>Modicon Modbus (1-based)</b>: implements a Holding Register range between 400001 and 465536 and an Output Coils range between 1 and 65536.</li> <li><b>Generic Modbus (0-based)</b>: implements a Holding Register range between 400000 and 465535 and an Output Coils range between 0 and 65535.</li> </ul> <p> Note: The address range used in the Modbus frames is always between 0 and 65535 for the Holding Registers and between 0 and 65535 for Coils.</p>
<b>Comm...</b>	<p>If clicked displays the communication parameters setup dialog.</p> <p>You have to set parameters according to the values programmed in Modbus Master.</p> 



Element	Description	
	Element	Description
	<b>Uart</b>	<p>Serial port selection.</p> <p>On UN20 devices:</p> <ul style="list-style-type: none"> <li>• <b>com1</b> = device port labeled <b>PLC</b></li> <li>• <b>com2</b> = device port labeled <b>PC/Printer</b></li> </ul> <p>On UN30 and UN31 devices:</p> <ul style="list-style-type: none"> <li>• <b>com1</b> = integrated serial port</li> <li>• <b>com2</b> = add-on module plugged in <b>Slot#1</b> or into <b>Slot#2</b></li> <li>• <b>com3</b> = add-on module plugged in <b>Slot#3</b> or into <b>Slot#4</b></li> </ul>
	<b>Baudrate, Parity, Data bits, Stop bits</b>	Serial line parameters.
	<b>Mode</b>	<p>Serial port mode. Available options:</p> <ul style="list-style-type: none"> <li>• <b>RS-232</b></li> <li>• <b>RS-485</b> (2 wires)</li> <li>• <b>RS-422</b> (4 wires)</li> </ul>

## Node Override ID

The protocol provides the special data type Node Override ID which allows you to change at runtime the Modbus address defined for the HMI device. This memory type is an unsigned byte.

The node Override ID is initialized with the value of the node ID specified in the project at programming time.

Node Over-ride ID	Modbus operation
<b>0</b>	Communication with the controller is stopped. In case of write operation, the request will be transmitted without waiting for a reply.
<b>1 to 254</b>	It is interpreted as the value of the new node ID and is replaced for runtime operation.
<b>255</b>	Communication with the controller is stopped; no request messages are generated.



Note: Node Override ID value assigned at runtime is retained through power cycles.

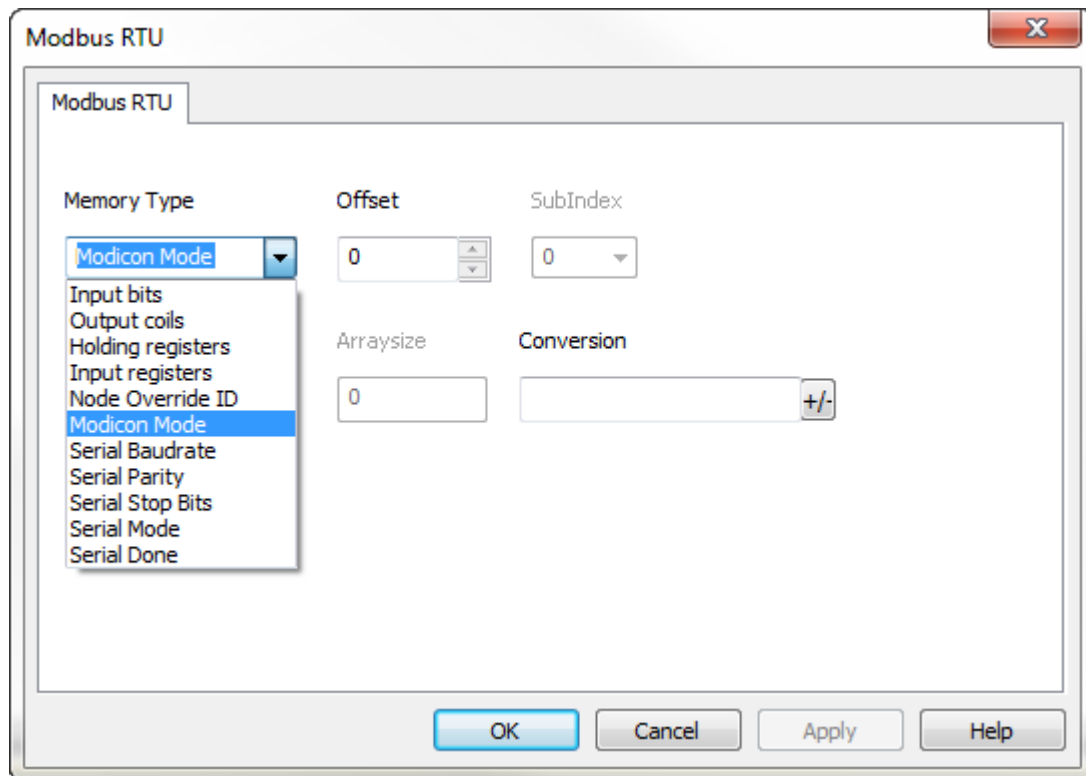
## Line parameters override

The protocol provide special data types to override the communication line parameters.

Line Parameters	Line Parameters Values
<b>Modicon Mode</b>	0=0-based, 1=1-based
<b>Serial Baudrate</b>	150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
<b>Serial Parity</b>	0=none, 1=parity even, 2=parity odd
<b>Serial Stop Bit</b>	1=1 stop bit, 2=2 stop bit
<b>Serial Mode</b>	0=RS-232, 1= RS-485, 2= RS-422
<b>Serial Done</b>	Set to 1 to overwrite the communication line parameters. The parameters are processed all together only when this variable is set to value 1



Note: Line parameter values assigned at runtime are retained through power cycles.



## Communication status

Current communication status can be displayed using system variables. See ["Communication variables" on page 66](#)

This communication protocol acts as server and doesn't return any specific Protocol Error Message.

## Implementation details

This Modbus RTU slave implementation supports only a subset of the standard Modbus function codes.

Code	Function	Description
01	Read Coil Status	Reads multiple bits in the device Coil area.
03	Read Holding Registers	Read multiple device Registers.
05	Force Single Coil	Forces a single device Coil to either ON or OFF.
06	Preset Single Register	Presets a value in a device Register.
08	Loopback Diagnostic Test	Only sub function 00 (Return Query Data) is supported. The HMI protocol will return the Exception Code 03 (Illegal Data Value) if a sub function other than 00 is specified here.
15	Force Multiple Coils	Forces multiple device Coils to either ON or OFF.

Code	Function	Description
16	Preset Multiple Registers	Presets value in multiple device Registers.
17	Report Slave ID	Returns diagnostic information of the controller present at the slave address.
23	Read Write Multiple Registers	Read & presets values in multiple device Registers

The HMI protocol will return the Exception Code 01 (Illegal Function) if the function code received in the query is not supported.

## Memory limits

The amount of memory available in the HMI device for this protocol is:

Data Type	Type	Range
Coils	Bit	0 – 65535
Registers	Word	0 – 65535

The HMI protocol will return the Exception Code 02 (Illegal Data Address) if the Data Address received in the query exceeds the predefined data ranges.

# Modbus TCP

Various Modbus TCP-capable devices can be connected to HMI devices. To set-up your Modbus TCP device, please refer to the documentation you have received with the device.

The implementation of the protocol operates as a Modbus TCP client only.

## Adding a protocol

To configure the protocol:

- 1. In the **Config** node double-click **Protocols**.
- 2. To add a driver, click **+**: a new line is added.
- 3. Select the protocol from the **PLC** list.

The driver configuration dialog is displayed.

## Protocol Editor settings

Modbus TCP

☐ PLC Network

OK

Cancel

Alias

IP address

0 . 0 . 0 . 0

Port

502

Timeout (ms)

2000

Modbus ID

1

Max read block

250

Max read bit block

2000

Write Holding Register

16

Write Coils

15

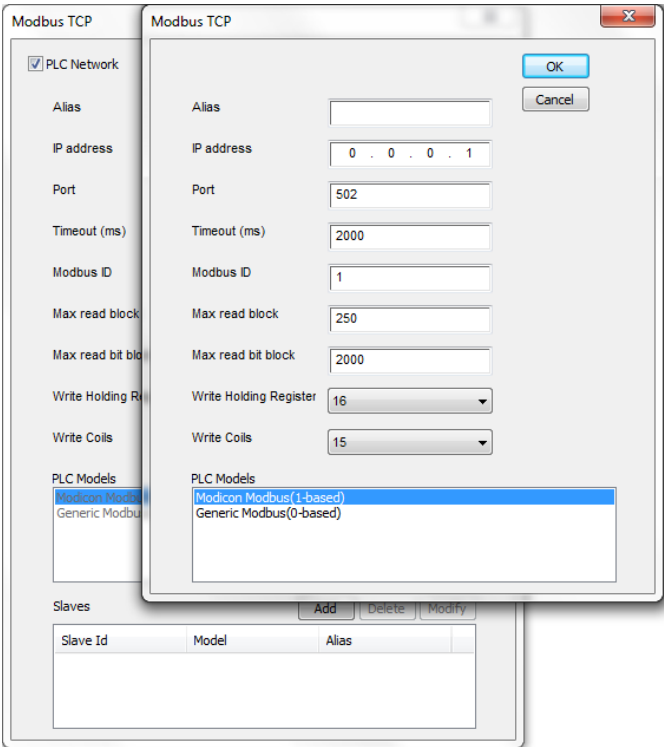
PLC Models

Modicon Modbus(1-based)

Generic Modbus(0-based)

Element	Description
Alias	Name identifying nodes in network configurations. The name will be added as a prefix to each tag name imported for each network node.
IP address	Address of the controller.

Element	Description
<b>Port</b>	Port number used by the Modbus TCP driver. The default value is <b>502</b> and can be changed when the communication goes through routers or Internet gateways where the default port number is already in use.
<b>Timeout (ms)</b>	Time delay in milliseconds between two retries in case of missing response from the server device.
<b>Modbus ID</b>	Usually used when communicating over Ethernet-to-serial gateways and then interpreted as the Slave ID. This value is simply copied into the Unit Identifier field of the Modbus TCP communication frame. This is rarely used and in most cases can be left zero.
<b>Max read block</b>	Maximum length in bytes of a data block request. It applies only to read access of Holding Registers.
<b>Max read bit block</b>	Maximum length in bits of a block request. It applies only to read access of Input Bits and Output Coils.
<b>Write Holding Register</b>	<p>Modbus function for write operations to Holding Registers. Select between the function <b>06</b> (preset single register) and function <b>16</b> (preset multiple registers).</p> <p>If function <b>06</b> is selected, the protocol will always use function <b>06</b> for writing to the controller, even when writing to multiple consecutive registers.</p> <p>If function <b>16</b> is selected, the protocol will always use function <b>16</b> to write to the controller, even for a single register write request and the <b>Max read block size</b> parameter of the query is set to <b>2</b>. The use of function <b>16</b> may result in higher communication performance.</p>
<b>Write Coils</b>	<p>Modbus function for write operations to Output Coils. Select between the function <b>05</b> (write single coil) and function <b>15</b> (write multiple coils).</p> <p>If Modbus function <b>05</b> is selected, the protocol will always use function <b>05</b> for writing to the controller, even when writing to multiple consecutive coils.</p> <p>If Modbus function <b>15</b> is selected, the protocol will always use function <b>15</b> to write to the controller, even for a single coil write request. The use of function <b>15</b> may result in higher communication performance.</p>

Element	Description
PLC Models	<p>Two PLC models are available.</p> <ul style="list-style-type: none"><li>• <b>Modicon Modbus (1-based)</b>: has an addressing space that starts from offsets 1 for all memory types.</li><li>• <b>Generic Modbus (0-based)</b>: has an addressing space that starts from offset 0 for all memory types.</li></ul>
PLC Network	<p>IP address for all controllers in multiple connections. <b>PLC Network</b> must be selected to enable multiple connections.</p> 


## Tag definition

Path: **ProjectView**> **Config** > double-click **Tags**

1. To add a tag, click **+**: a new line is added.
2. Select **Modbus TCP** from the **Driver** list: tag definition dialog is displayed.

Element	Description		
Memory Type	Modbus resource where tag is located.		
	Memory Type	Modbus Resource	
	Output coils	Coils	
	Input bits	Discrete Inputs	
	Input registers	Input Registers	
	Holding registers	Holding Registers	
Offset	Offset address where tag is located.		
	Memory Type	Offset	Resource Address
	Output coils	0 – 65535	Coils
	Input bits	100000 – 165535	Discrete Inputs
	Input registers	300000 – 365535	Input Registers
	Holding registers	400000 – 465535	Holding Registers
SubIndex	This allows resource offset selection within the register.		
Data Type	Available data types: <ul style="list-style-type: none"><li>• boolean</li><li>• byteshort</li><li>• unsignedByte</li><li>• unsignedShort</li></ul>		



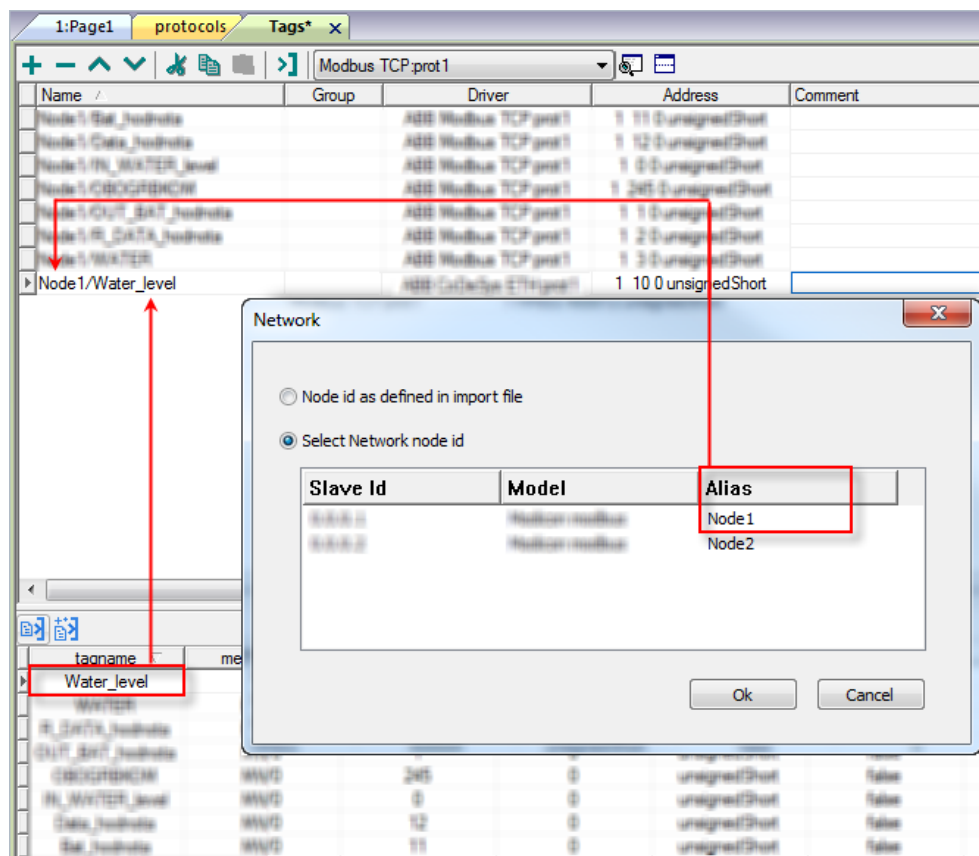
Element	Description														
	<ul style="list-style-type: none"> <li>• unsignedInt</li> <li>• float</li> <li>• double</li> <li>• string</li> <li>• binary</li> </ul> <p>See "Data types" on page 28 for details.</p> <p> Note: To define arrays, select one of Data Type format followed by square brackets.</p>														
<b>Arraysize</b>	The amount of array elements or characters of the string.														
<b>Conversion</b>	<p>Conversion to be applied.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>Inv bits</td><td>Invert a bit (boolean tag).</td></tr> <tr> <td>Negate</td><td>Invert all bits.</td></tr> <tr> <td>AB -&gt; BA</td><td>Swap nibbles of a byte.</td></tr> <tr> <td>ABCD -&gt; CDAB</td><td>Swap bytes of a word.</td></tr> <tr> <td>ABCDEFGH -&gt; GHEFCBAB</td><td>Swap bytes of a double word.</td></tr> <tr> <td>BCD</td><td>BCD format conversion.</td></tr> </table>	Value	Description	Inv bits	Invert a bit (boolean tag).	Negate	Invert all bits.	AB -> BA	Swap nibbles of a byte.	ABCD -> CDAB	Swap bytes of a word.	ABCDEFGH -> GHEFCBAB	Swap bytes of a double word.	BCD	BCD format conversion.
Value	Description														
Inv bits	Invert a bit (boolean tag).														
Negate	Invert all bits.														
AB -> BA	Swap nibbles of a byte.														
ABCD -> CDAB	Swap bytes of a word.														
ABCDEFGH -> GHEFCBAB	Swap bytes of a double word.														
BCD	BCD format conversion.														

## Adding an alias name to a protocol

Tag names must be unique at project level, however, the same tag names might need to be used for different controller nodes (for example when the HMI device is connected to two devices running the same application).

When creating a protocol you can add an alias name that will be added to tag names imported for this protocol.

In the example, the connection to a certain controller is assigned the name **Node1**. When tags are imported for this node, all tag names will have the prefix **Node1** making each of them unique at the network/project level.



**Note:** Aliasing tag names is only available for imported tags. Tags which are added manually in the Tag Editor do not need to have the Alias prefix in the tag name.

The Alias string is attached on the import. If you modify the Alias string after the tag import has been completed, there will be no effect on the names already present in the dictionary. When the Alias string is changed and tags are re-imported, all tags will be re-imported with the new prefix string.

## Tag import file structure

This protocol supports the import of tag information when provided in .csv format according to the following format:

`NodeID, TagName, MemoryType, Address, DataFormat, ..., [Comment]`

**Note:** Fields in brackets are optional as well as fields between Data Format and Comment.

Field	Description
<b>NodeID</b>	Node the tag belongs to
<b>TagName</b>	Tag description
<b>MemoryType</b>	<ul style="list-style-type: none"> <li>• OUTP</li> <li>• INP</li> <li>• IREG</li> <li>• HREG</li> </ul>

Field	Description
Address	Offset compatible with Modbus notation
DataFormat	Data type in internal notation. See <a href="#">"Data types" on page 28</a>
Comment	Optional additional description.

## Tag file example

Example of .csv line:

```
2,Holding Register 1, HREG, 400001, unsignedShort,
```



Note: This line has no comment. When the Comment is missing, the comma as a terminator character is mandatory.

## Importing tags

See ["Importing tags" on page 21](#).

## Node Override IP

The protocol provides the special data type Node Override IP which allows you to change the IP address of the target controller at runtime.

This memory type is an array of 4 unsigned bytes, one per each byte of the IP address.

The Node Override IP is initialized with the value of the controller IP specified in the project at programming time.

Node Override IP	Modbus operation
0.0.0.0	Communication with the controller is stopped, no request frames are generated any-more.
Different from 0.0.0.0	It is interpreted as node IP override and the target IP address is replaced runtime with the new value.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override IP variable.



Note: Node Override IP values assigned at runtime are retained through power cycles.

## Node Override ID

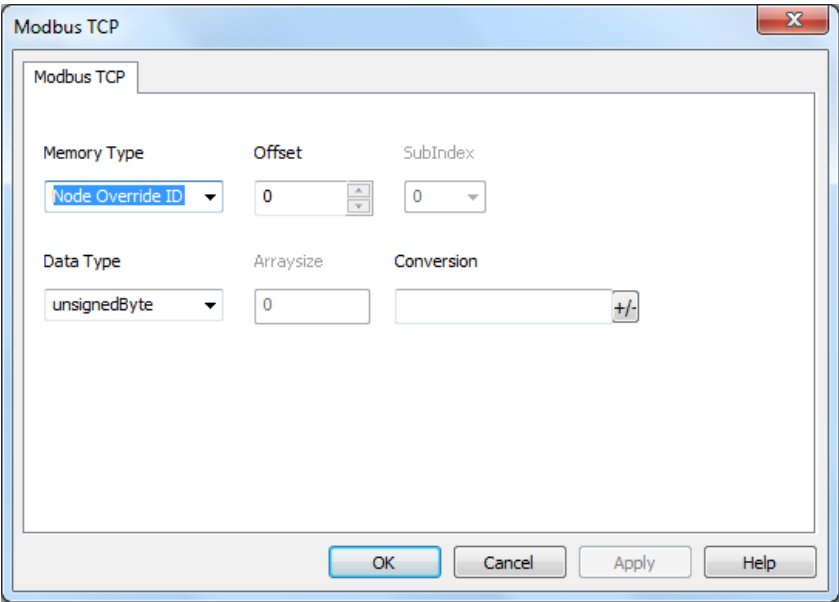
The protocol provides the special data type Node Override ID which allows you to change at runtime the Modbus address defined for the HMI device. This memory type is an unsigned byte.

The node Override ID is initialized with the value of the node ID specified in the project at programming time.

Node Over-ride ID	Modbus operation
0	Communication with the controller is stopped. In case of write operation, the request will be transmitted without waiting for a reply.
1 to 254	It is interpreted as the value of the new node ID and is replaced for runtime operation.
255	Communication with the controller is stopped; no request messages are generated.



Note: Node Override ID value assigned at runtime is retained through power cycles.



### Node Override Port


The protocol provides the special data type Node Override Port which allows you to change the network Port of the target controller at runtime.

This memory type is an unsigned short.

The Node Override Port is initialized with the value of the controller Port specified in the project at programming time.

Node Override Port	Modbus operation
0	Communication with the controller is stopped, no request frames are generated anymore.
Different from 0	It is interpreted as the value of the new port and is replaced for runtime operation.

If the HMI device is connected to a network with more than one controller node, each node has its own Node Override Port variable.



Note: Node Override Port values assigned at runtime are retained through power cycles.

Modbus TCP

Modbus TCP

Memory Type

Node Override Port

Offset

0

SubIndex

0

Data Type

unsignedShort

Arraysize

0

Conversion

1

OK

Cancel

Apply

Help

Modbus Mode Override

The protocol provide a special data types that can be used to override the Modicon Mode parameter at runtime

Line Parameters	Line Parameters Values
Modicon Mode	0=0-based, 1=1-based

Modbus TCP

Modbus TCP

Memory Type

Modicon Mode

Offset

0

SubIndex

0

Data Type

boolean

Arraysize

0

Conversion

OK

Cancel

Apply

Help



Note: Modicon Mode parameter value assigned at runtime is retained through power cycles.

## Communication status

Current communication status can be displayed using system variables. See ["Communication variables" on page 66](#).

Codes supported for this communication driver:

Error	Cause	Action
<b>No response</b>	No reply within the specified timeout.	Check if the controller is connected and properly configured to get network access.
<b>Incorrect node address in response</b>	The device received a response with an invalid node address from the controller .	-
<b>The received message too short</b>	The device received a response with an invalid format from the controller .	-
<b>Incorrect writing data acknowledge</b>	The controller did not accept a write request.	Check if project data is consistent with the controller resources.

## Implementation details

This Modbus TCP implementation supports only a subset of the Modbus TCP standard function codes.

Code	Function	Description
<b>01</b>	Read Coil Status	Reads multiple bits in the HMI device Coil area.
<b>02</b>	Read Input Status	Reads the ON/OFF status of the discrete inputs (1x reference) in the slave.
<b>03</b>	Read Holding Registers	Reads multiple registers.
<b>04</b>	Read Input Registers	Reads the binary contents of input registers (3x reference) in the slave.
<b>05</b>	Force Single Coil	Forces a single coil to either ON or OFF.
<b>06</b>	Preset Single Register	Writes a value to one register.
<b>15</b>	Write Multiple Coils	Writes each coil in a sequence of coils to either ON or OFF.
<b>16</b>	Preset Multiple Registers	Writes values to a block of registers in sequence.

# Modbus TCP Server

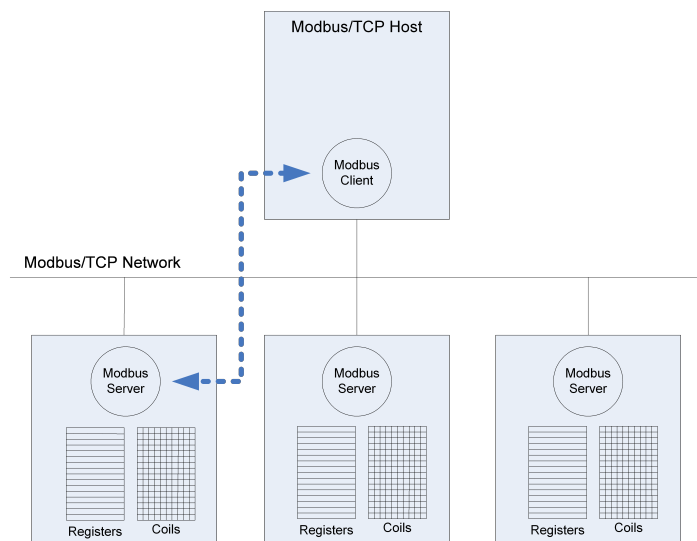
Modbus TCP Server communication driver allows connecting the HMI device as a server in a Modbus TCP network. Standard Modbus TCP messages are used for information exchange.

This approach allows connecting HMI devices to SCADA systems through the universally supported Modbus TCP communication protocol.

## Principle of operation

This communication driver implements a Modbus TCP Server unit in HMI device. A subset of the complete range of Modbus function codes is supported. The available function codes allow data transfer between clients on the TCP network and the server.

The following diagram shows the system architecture.



The device simulates the communication interface of a PLC: Coils and Registers data types are respectively boolean and 16 bit integers.

The device always access data in its internal memory. Data can be transferred to and from the Modbus Client only on the initiative of the client itself.

## Adding a protocol

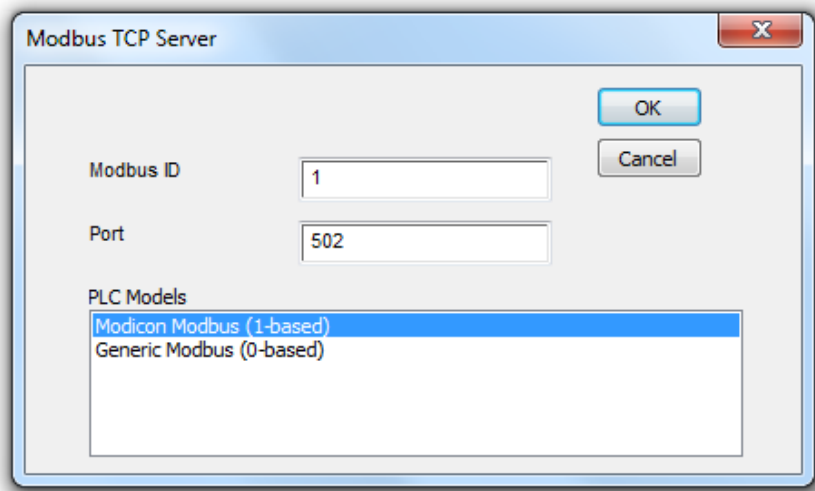
To configure the protocol:


1. In the **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The driver configuration dialog is displayed.



## Protocol Editor settings

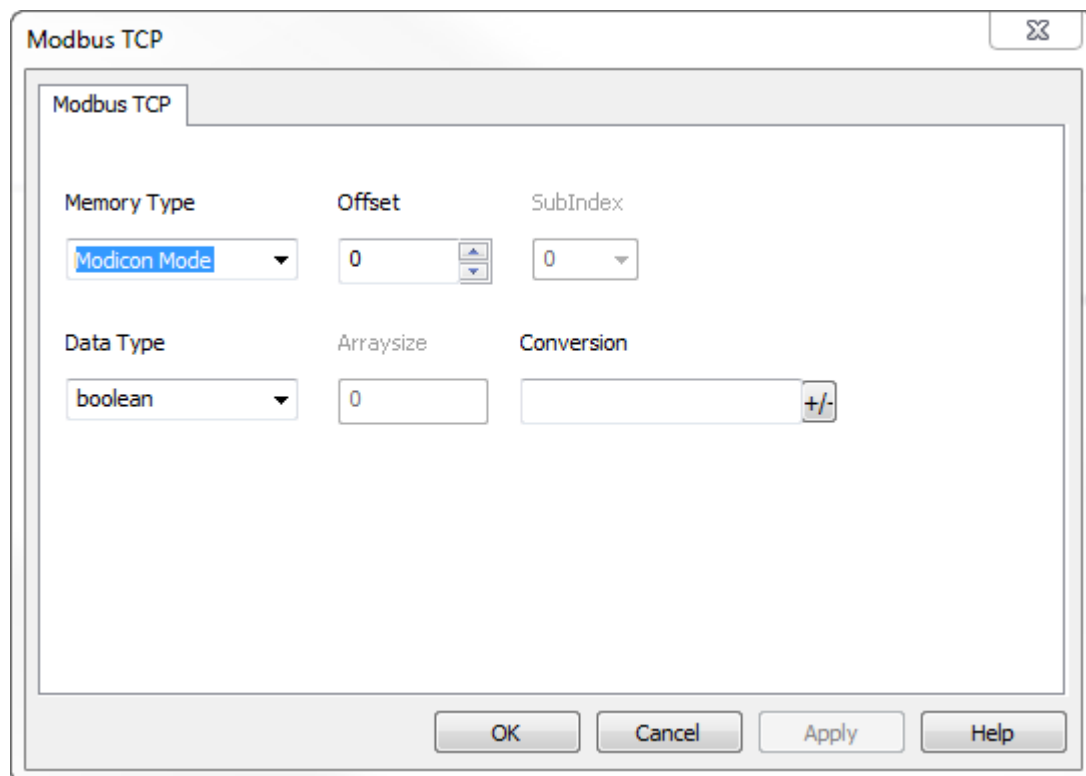


Element	Description
<b>Modbus ID</b>	Modbus node ID of the HMI device. Every Modbus server device in the network must have its own Modbus ID.
<b>Port</b>	Port number used by the Modbus TCP protocol. Default value is <b>502</b> . Set the value accordingly to the port number used by your Modbus TCP Network.
<b>PLC Models</b>	<p>Allows you to select between two models of Modbus TCP Server.</p> <ul style="list-style-type: none"> <li><b>Modicon Modbus (1-based)</b>: implements a Holding Register range between 400001 and 465536 and an Output Coils range between 1 and 65536.</li> <li><b>Generic Modbus (0-based)</b>: implements a Holding Register range between 400000 and 465535 and an Output Coils range between 0 and 65535.</li> </ul> <p> Note: The address range used in the Modbus frames is always between 0 and 65535 for the Holding Registers and between 0 and 65535 for Coils.</p>

## Special Data Type

The protocol provide a special data types that can be used to override the Modicon Mode parameter at runtime

Line Parameters	Line Parameters Values
<b>Modicon Mode</b>	0=0-based, 1=1-based



The image shows a 'Modbus TCP' configuration window. It has a title bar with a close button. Inside, there's a tab labeled 'Modbus TCP'. The window contains several settings:

- Memory Type:** A dropdown menu currently showing 'Modicon Mode'.
- Offset:** A numeric input field with '0' and up/down arrow buttons.
- SubIndex:** A dropdown menu currently showing '0'.
- Data Type:** A dropdown menu currently showing 'boolean'.
- Arraysize:** A numeric input field with '0'.
- Conversion:** An empty text field followed by a '+/-' button.

At the bottom of the window are four buttons: 'OK', 'Cancel', 'Apply', and 'Help'.



Note: Modicon Mode parameter value assigned at runtime is retained through power cycles.

## Communication status

The HMI device is a server station in the Modbus TCP network. The current implementation of the protocol doesn't report any communication error code apart from standard communication error codes related to the proper driver loading.

## Implementation details

This Modbus TCP Server implementation supports only a subset of the Modbus standard function codes.

Code	Function	Description
01	Read Coil Status	Reads multiple bits in the device Coil area.
02	Read Input Status	Reads multiple bits in the device Coil area.
03	Read Holding Registers	Read multiple device Registers.
04	Read Input Registers	Read multiple device Registers.
05	Force Single Coil	Forces a single device Coil to either ON or OFF.
06	Preset Single Register	Presets a value in a device Register.
15	Force Multiple Coils	Forces multiple device Coils to either ON or OFF.
16	Preset Multiple Registers	Presets value in multiple device Registers.
23	Read Write Multiple Registers	Read & presets values in multiple device Registers



# Contact us

## **ABB Automation Products GmbH**

Wallstadter Str. 59

68526 Ladenburg, Germany

Phone: +49 62 21 701 1444

Fax: +49 62 21 701 1382

E-Mail: [plc.sales@de.abb.com](mailto:plc.sales@de.abb.com)

**[www.abb.com/automationbuilder](http://www.abb.com/automationbuilder)**

**[www.abb.com/plc](http://www.abb.com/plc)**

### **Note:**

We reserve the right to make technical changes or modify the contents of this document without prior notice. With regard to purchase orders, the agreed particulars shall prevail. ABB AG does not accept any responsibility whatsoever for potential errors or possible lack of information in this document.

We reserve all rights in this document and in the subject matter and illustrations contained therein. Any reproduction, disclosure to third parties or utilization of its contents – in whole or in parts – is forbidden without prior written consent of ABB AG.

Copyright© 2015 ABB  
All rights reserved

3ADR059056M0201