

# Table of Contents

<b>Chapter I</b>	<b>Introduction</b>	<b>1</b>
<b>Chapter II</b>	<b>Program start</b>	<b>1</b>
<b>Chapter III</b>	<b>Tool bar</b>	<b>2</b>
<b>Chapter IV</b>	<b>Property editor</b>	<b>4</b>
<b>Chapter V</b>	<b>Template window</b>	<b>5</b>
<b>Chapter VI</b>	<b>Working with components</b>	<b>5</b>
<b>Chapter VII</b>	<b>MultiTemplate</b>	<b>6</b>
1	Working with MultiTemplates .....	6
<b>Chapter VIII</b>	<b>Description of the components</b>	<b>9</b>
1	ActiveX .....	9
2	Bitmap .....	11
3	BkBitmap .....	16
4	Button .....	17
5	Chart .....	19
6	CheckBox .....	23
7	ComboBox .....	25
8	Edit .....	28
9	FileListBox .....	30
10	Frame .....	33
11	Gauge .....	36
12	GroupBox .....	41
13	HistoricalAlarmsView .....	42
14	HistoricalView .....	45
15	HtmlHelp .....	48
16	Label .....	50
17	Led .....	53
18	Metafile .....	56
19	RadioButton .....	59
20	StatusBar .....	61
21	Switch .....	65
22	TabSheet .....	68

23	Template .....	69
24	ThermMap .....	73
25	UpDown .....	75
26	DevView .....	77
27	OperatorView .....	79
28	ReportView .....	81
29	AlarmsView .....	83
30	HSlider .....	86
31	VSlider .....	90
32	Dial .....	93
33	GearDial .....	95
34	HMeter .....	98
35	VMeter .....	101
36	120Meter .....	105
37	180Meter .....	109
38	270Meter .....	113
39	ThermoMeter .....	116
40	RockerSwitch .....	120
41	ToggleSwitch .....	122
42	WebBrowser .....	125

# 1 Introduction



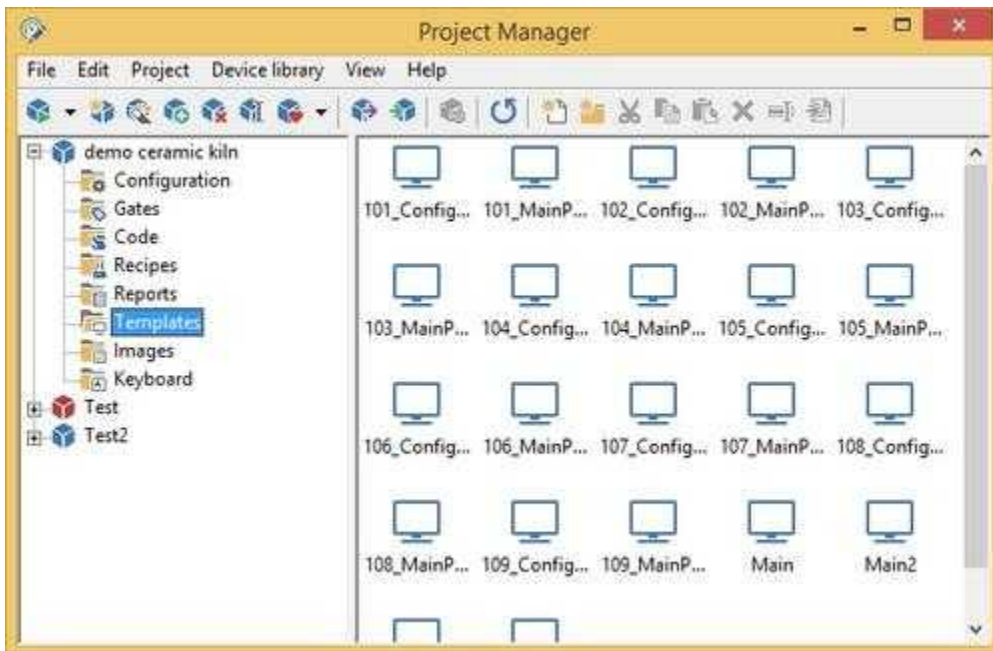
*Template Builder* is the section that allows to design supervision pages using a very intuitive interface, in order to easily carry out all the project schemes.

On the supervision pages you can use all the components available in the software (buttons, labels, status bars...). Each of them is configurable in different aspects, so you can adjust it to the application characteristics in the best possible way.

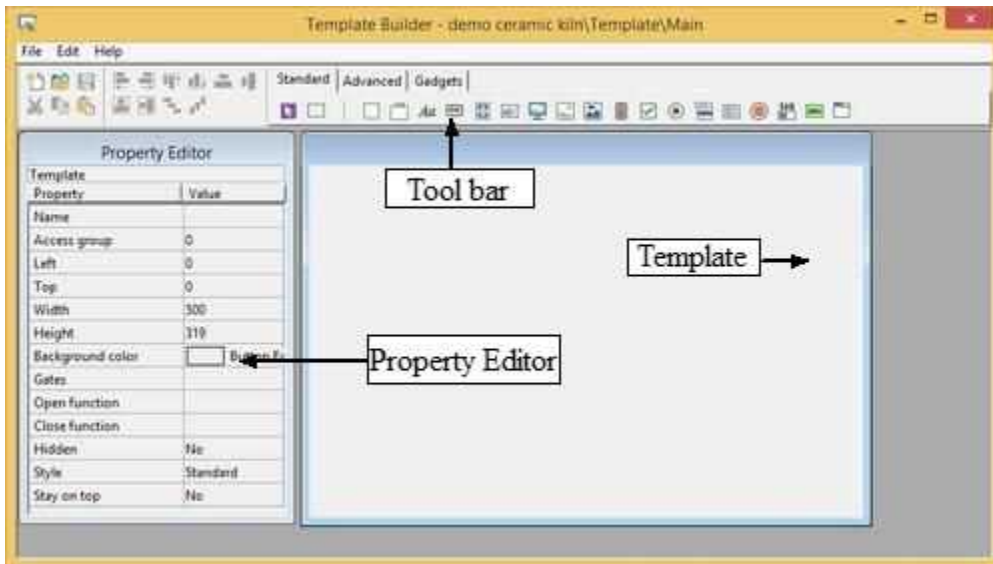
## 2 Program start

The creation of a new blank Template is performed in the *Project Manager*, by selecting the *Template* folder and then moving the mouse on the right window and pressing the right button: the "New / *Template*" menu entry is then displayed

*Template Builder* is started automatically by the *Project Manager* by double clicking on the icon to any Template previously created.



In figure below it is shown the *Template Builder* working environment. You may note three main sections: at the top the tool bar, on the left the property editor, and on the right the template window.

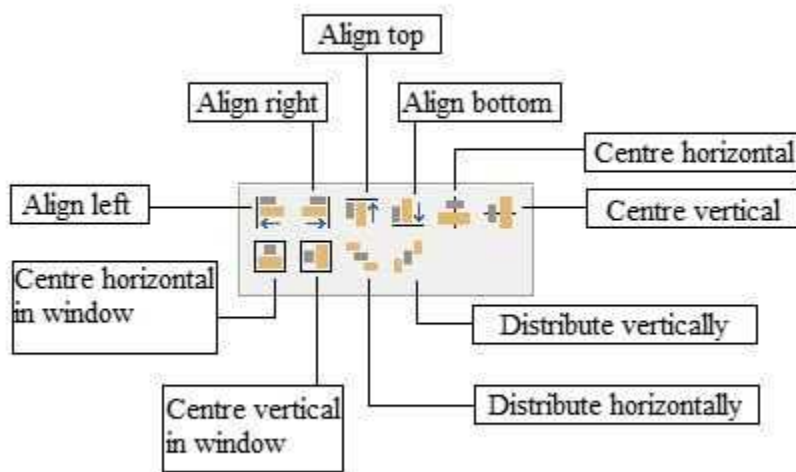
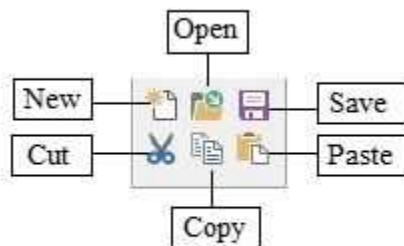


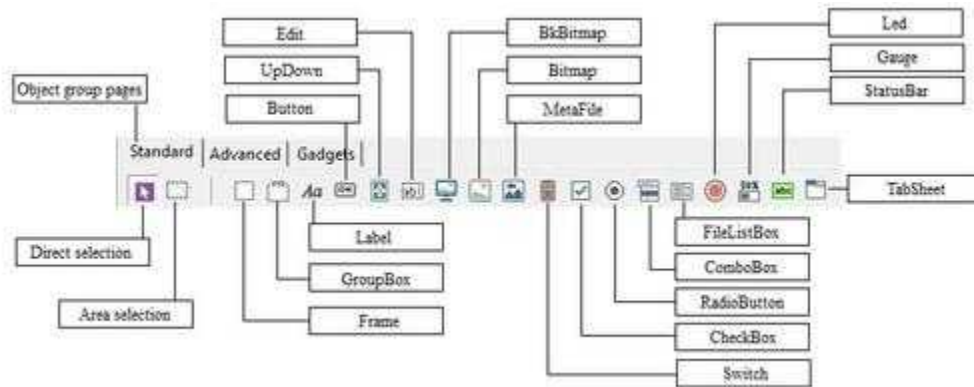
### 3 Tool bar

In figure there is a detailed description of the tool bar: it is divided into two parts, the one on the left contains the commands to create a new template (function that can be called up also from the menu *File | New*), to open a template (*File | Open...*), to save the current template (*File | Save*) and to use the clipboard (functions that can be called up also from the menu *Edit | Cut, Copy, Paste*); while the part on the right contains the buttons to select the objects to be placed in the template window. The

objects are divided into two categories: Standard and Advanced. The standard page, contains the objects frequently used; in the Advanced section there are complex objects such as trends and alarm lists. To place an object in the template, you just need to press the corresponding button on the tool bar and click on the template window in the desired position: the new component will be created and automatically selected, so as to be able to move and modify it in the desired manner (as described below).

Note that when a new component is created in the template window, the button that had been selected on the tool bar, is deselected, and the button on its left is selected (the arrow). When this button is pressed, it is possible to select and modify the components already present in the template by clicking on them, without creating new ones.





## 4 Property editor

Every component of a template has several properties. These properties can be modified at will, and *property editor* does this. Selecting a component in the template, in the *property editor* will appear its properties. If, on the contrary, you select the window template, in the *property editor* will appear the rows with the template properties. To modify the shown properties, first of all you have to position the selection on the desired row (with the mouse or the cursor keys). On the left part of the selected row (which looks lowered) there is a label indicating the name of the property, while on the right part there is the present value of the property. Basically there are three types of rows in the *property editor*: the edit rows (figure A), the multiple choice rows (figure B) and the rows with button (figure C).

Height	17	<b>(A)</b>
Description	Temperature	
Background color	<input type="text" value="Custom (240,240,240)"/>	
Font	"MS Sans Serif",8,0000	<b>(B)</b>
Cursor	White	
Label	CheckBox	
Background color	<input type="text" value="Custom (240,240,240)"/>	<b>(C)</b>
Font	"MS Sans Serif",8,0000	
Cursor	(default)	

With the first type of rows it is possible to modify the property straight from the *property editor*: you just need to write with the keyboard the value of the property in the edit square. The multiple choice rows allow to select the value of the property from a list that is shown by pressing with the mouse the small button to the right of the line. As far as the button rows are concerned, the matter is different: in the row it is shown the present value of the property, but to modify it you have to press the button on the right with the mouse. Then a specific window for the property will be opened. In it the present settings of the property are shown in detail. So it will be possible to modify the property by acting on the parameters of this window, and then confirm the choices by pressing the *Ok* button: the window will be closed and the new settings of the property will be shown in the row of the *property editor*.

Every time that a property is modified using the *property editor*, it is possible to undo the change simply pressing the *ESC* key before selecting a new line.

Note that every time that in the *property editor* you modify a property concerning in some way the display of the component (for example the Color of a component, its font, its size and so on), the component selected in the template is updated consistently with the new settings of the property. The same can be said also for the opposite. As you'll see in the next paragraph, it is possible to modify the properties of a component straight from the template (usually its position and size): in such cases the data displayed in the *property editor* will be updated accordingly.

## 5 Template window

The template window shows a draft of the supervision page (i.e. the template) once it has been displayed by the software.

In this window will be placed all the components forming the template, by simply selecting the desired component from the tool bar and clicking in the template window. The object just inserted in the template will be selected; so it will be soon and easily identified. When a component is selected, *Template Builder* will place eight small black squares along the object outline, and in the *property editor* all its properties will be shown. The selected object is the one in which to carry out all the changes specified in the *property editor*. To select an object, you just need to click on it with the mouse: the selection will be removed from the component previously selected, and placed on the new component, and the properties of the *property editor* will be updated consistently with the new selected object.

As said before, it is possible to change the position and the size of the component straight from the template window, without using the *property editor*.

To move an object, you just need to click on it and drag it in the new position, then release the mouse left button: while you are dragging it, you will see the object follow the mouse pointer, and when it is released you will note that the *property editor* rows corresponding to the position are updated. Note that, in order to keep the objects hierarchies (see below the description of objects that can include other objects) it is not possible to move a child object outside the boundaries of its parent (the object including it).

As easily, you can change the size of the selected object: you just need to click on and drag one of the eight small squares forming the selection, and release it in the desired position. You will notice that the object will change its size according to your wishes and that the *property editor* rows will be updated when the mouse is released.

To make easier the positioning of the objects in the template, its position and size cannot have a value whatsoever. Normally, in fact, position and size can only have values that are multiples of five: this way it is easier to line up the objects consistently in all the template. If you want to position or resize an object more precisely, it is possible to specify the desired position or size in the *property editor*, or, as an alternative, drag or resize the object by holding down the CTRL key: the object will be moved or resized pixel by pixel, and not any longer by five pixels at a time.

The moving and resizing operations can also be carried out using the keyboard: to move the selected object you just need to use the cursor keys, while to resize it you have to do the same while you hold down the SHIFT key. Also for the moving and resizing operations with the keyboard you can hold down CTRL key in order to allow movements of one pixel.

There is also the possibility to select multiple objects for move, copy and delete operations. To use multiselection option, hold down the SHIFT key and click the mouse on each object you want to include into multiselection.

## 6 Working with components

To work on the templates more easily and quickly you can use the editing functions provided by the Edit menu: it is possible to copy (Edit | Copy), cut (Edit | Cut) and paste (Edit | Paste) components or components hierarchies. For example if you have to create a template made up of similar parts

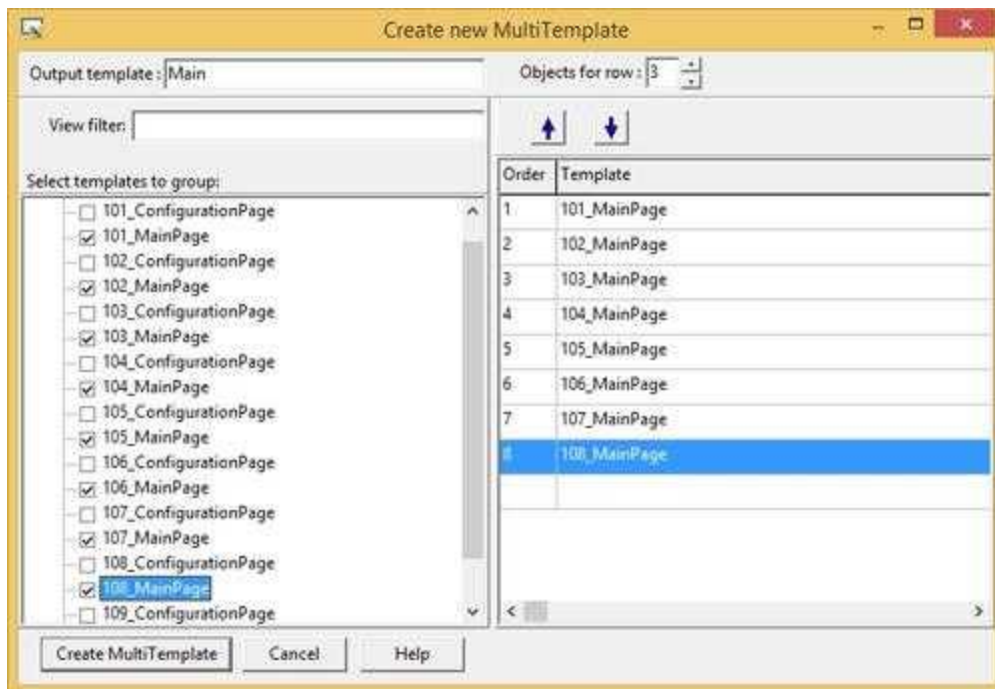
repeating themselves, you can create only one of these parts in a frame, than select the frame and copy it (Edit | Copy or CTRL+C). With the paste function (Edit | Paste or CTRL+V), a copy of the frame (with all its children components), that had been previously copied in the clipboard, will be placed in the component presently selected.

To make even easier the creation of a template from already existing templates, it is possible to consolidate more templates using the function *Edit | Paste from file*. Thanks to this function, in the current template will be copied all the components of the specified template, and along with them all its gates. This way, at the end of the operation, the set of gates of the current template will consist of gates of the initial template and gates of the template that has been copied.

With SHIFT key pressed is possible to select some objects as multiselection and so is possible to move, cut, copy and paste them through a single operation.

## 7 MultiTemplate

### 7.1 Working with MultiTemplates



**MultiTemplate** is a *Template Builder* tool that allow to create a template using other templates like source objects.

Let's see how to use it.

From *Template Builder* select *File->New->MultiTemplate* item : the window above will appear.

In the left side of the window will be displayed all available templates present in the project: if the checkbox near a template is checked, the relative template will be displayed in the right side of the main window, where there are all the templates that will build the final *Output Template*. To remove a template from this window, uncheck the relative template name in the left window.

Using the arrows button placed at the top of the right window, the MultiTemplate creation order list can be modified.

*Objects for row* specify the number of objects (Templates) to dispose on a single row.



The height of each row is equal to the higher height size found between templates that will compound the multitemplate.

*View filter:* only templates files that match the filter will be displayed in the left side of the main window.

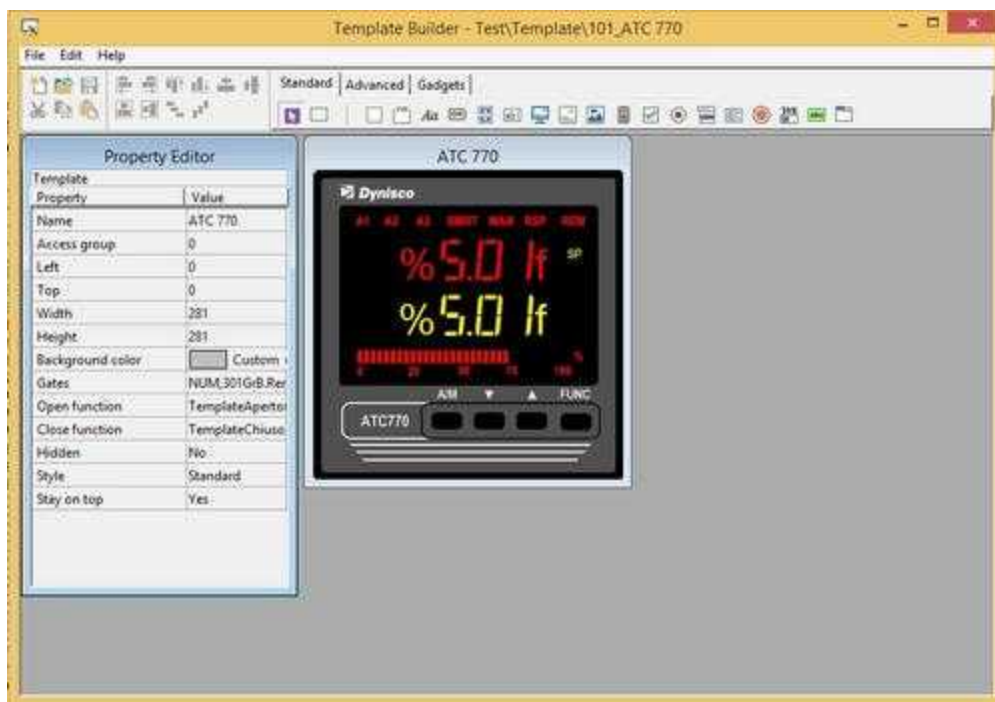
*Create MultiTemplate:* build the Output template file.

### Example:

Using *Application Builder* create an application with 6 devices Dynisco ATC 770 and name it "Test". In *Project Manager* select the application and open *Template Builder*.

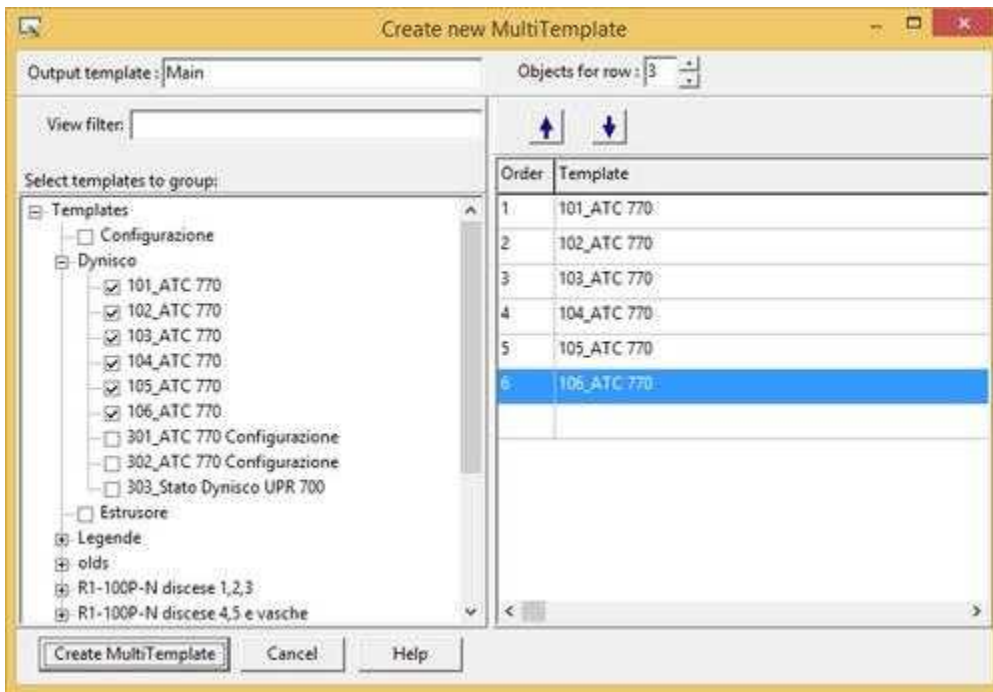
In *Template Builder* select File->New->MultiTemplate menu item and activate the checkbox of all the following template files: 101\_ATC 770, 102\_ATC 700,103\_ATC 770, 104\_ATC 770,105\_ATC 770,106\_ATC 770.

Note that all the above templates are like in the following figure:



On *Output Template* specify "Main".

On *Objects for row* specify "3".



After that, press "Create MultiTemplate" button.  
The following template called "Main" will be created.



## 8 Description of the components

### 8.1 ActiveX

This object allow to utilize, inside Template, some software objects produced by other developers in ActiveX technology. It is possible to interact with ActiveX object by linking gates to its dynamic properties.

#### Properties (TemplateBuilder)

**ID:** number than can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the ActiveX Object (in pixels).

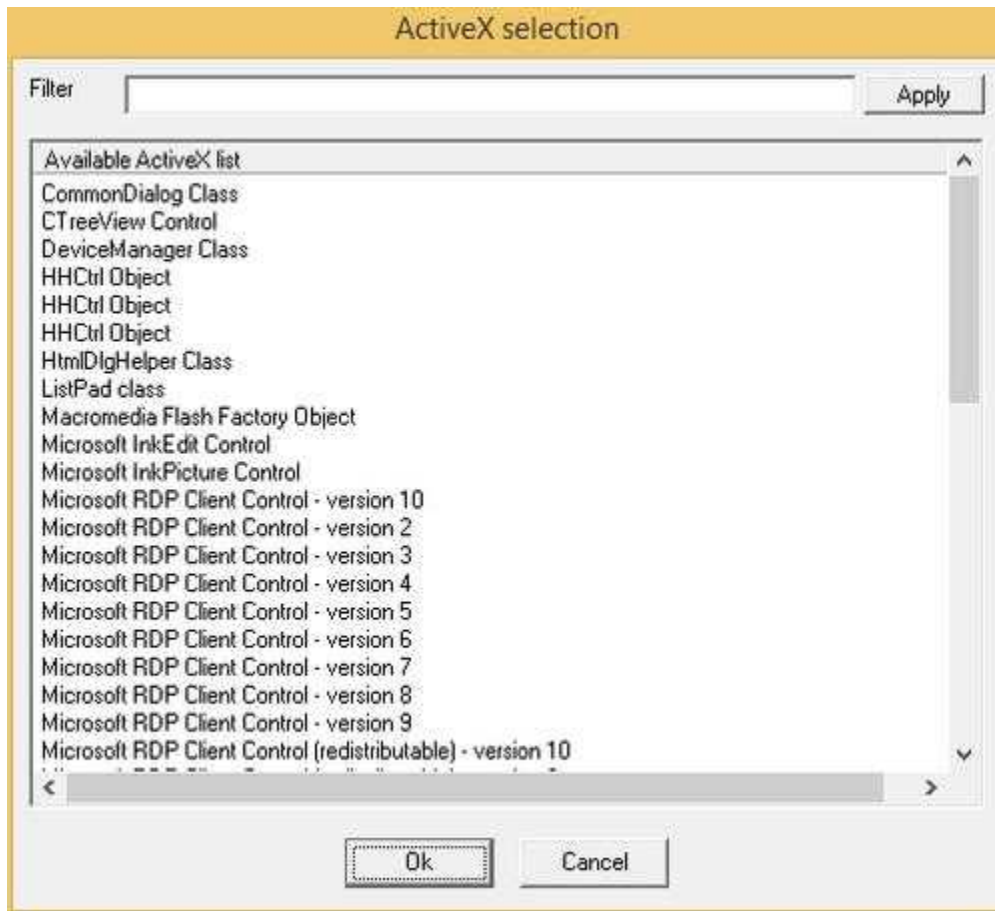
**Top:** vertical position of the top left corner of the ActiveX Object (in pixels).

**Width:** width of the ActiveX Object (in pixels).

**Height:** height of the ActiveX Object (in pixels).

**Description:** description of the Frame (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Name:** Selected ActiveX object name. Trough the following window it is possible to select one of the ActiveX objects installed on the computer.



**ActiveX properties:** let to configure the ActiveX object static properties by calling its static properties window (if the ActiveX object make this window available).

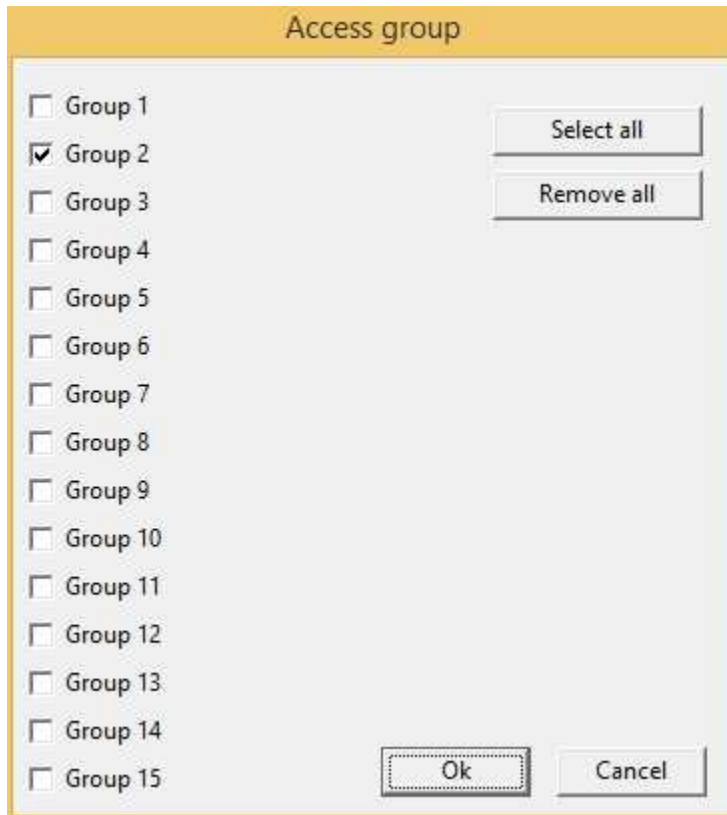
**Gates:** let to connect gates to the ActiveX dynamic properties (if the ActiveX object have some dynamic properties).(Details)

**Need apply:** it indicates when the gates connected to the ActiveX properties will be update. There are two options:

- No: the gates will be updated when the user presses the *Return* or *Tab* key.
- Yes: the gates will be updated when the user gives confirmation from the outside, that is to say when he uses Button with the "On Click" property set to "*Apply changes*".

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group:** users groups enabled to operate on the object in runtime mode. By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.

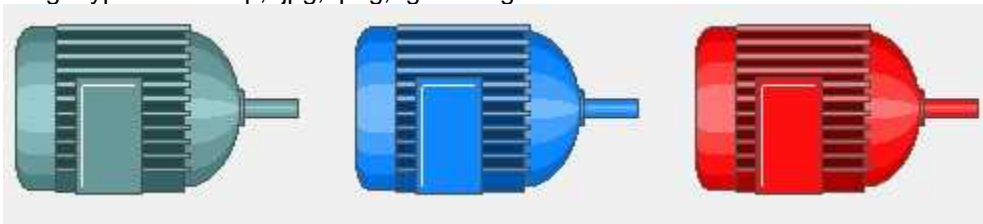


**Enable:** operation on the object in runtime mode will be enabled if is "true" at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

## 8.2 Bitmap

Unlike BkBitmap component, that represent a single static image, the Bitmap component allow to have in the template an image that changing during the supervision, so as to show graphically some conditions present in the plant. With every Bitmap can be associated a list of images, each with its conditions. This way when a condition takes place, in the template (during the supervision phase) the corresponding image or animation will be shown.

Using Bitmaps objects, it is very easy to produce very schematic and efficient templates. Supported image type are : .bmp, .jpg, .png, .gif and .gif animated.



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the Bitmap (in pixels).

**Top:** vertical position of the top left corner of the Bitmap (in pixels).

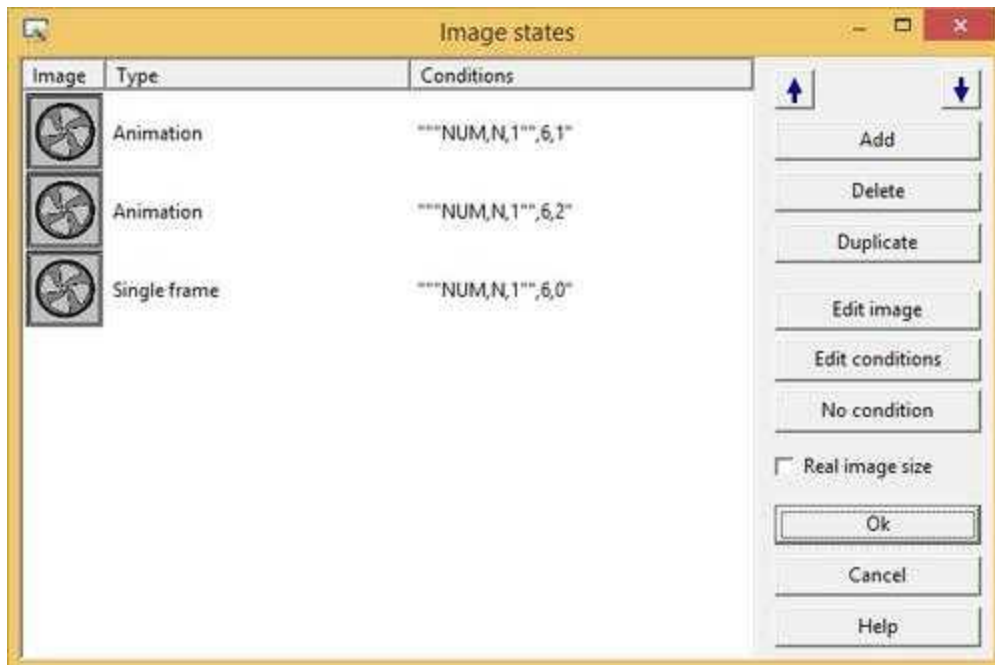
**Width:** width of the Bitmap (in pixels).

**Height:** height of the Bitmap (in pixels).

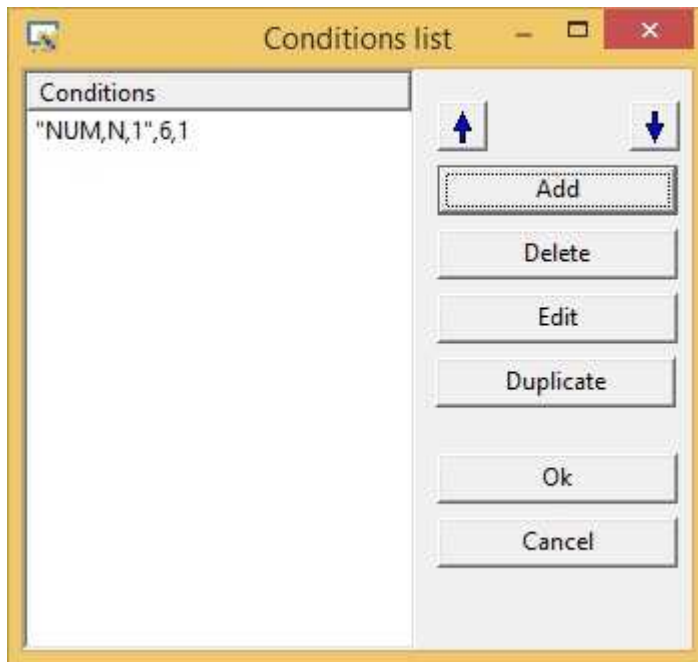
**Description:** a brief description of the Bitmap (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Cursor:** shape that the cursor of the mouse must have when it goes over the Bitmap during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

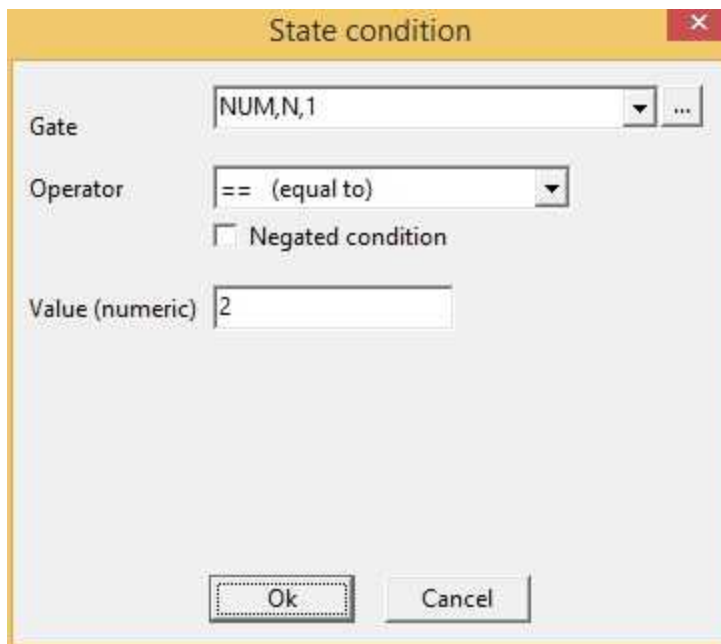
**Bitmap states:** images and conditions of the Bitmap. To select the states press the button on the properties row: the window in figure below will appear. It shows in every row of the left table the image or animation that will be displayed in the Bitmap when the status is active, and the list of conditions specifying when the status is to be considered active. To add a new status just press the button *Add*: a dialog box for selecting the image will appear. Once you have chosen the image, this will be inserted in the table with no conditions. It is possible to go back using the buttons *Edit image* (to select a new image) and *Remove* (to delete a status from the list). To sort the Bitmap states you can use the buttons with the arrows on the right top corner.



At this point it is possible to indicate the conditions making the image active with the button *Edit conditions*: press it to show the dialog box in figure below. It displays the list of the conditions for the status selected in the previous window.



Even in this case it is possible to add, remove, modify or move the existing conditions. Press the buttons for adding or modifying conditions to show the dialog box in figure.



First of all you have to select the template gate on which to verify the condition. Then you will have to select the type of condition to verify (in the list of the possible conditions) and the value with which to make a comparison. Note that the list of the possible conditions depends on the type of gate chosen, while the type of value with which to make a comparison depends both on the type of chosen gate and the condition to verify (for example if the gate is a string, the comparison value will also have to be a string). The possible conditions, for every type of gate, are indicated in the table below.

Condition	Description	Numeric	Digital	String	Compou	Event
-----------	-------------	---------	---------	--------	--------	-------

		gates	gates	gates	nd gates	gates
<b>No condition</b>	Condition never active	x	x	x	x	x
<b>Communication KO</b>	Condition active if communication is KO	x	x	x	x	
<b>&lt; (lower than)</b>	Condition active if gate value is lower than value	x		x	x	
<b>&gt; (higher than)</b>	Condition active if gate value is higher than value	x		x	x	
<b>&lt;= (lower or equal to)</b>	Condition active if gate value is lower or equal to value	x		x	x	
<b>&gt;= (higher or equal to)</b>	Condition active if gate value is higher or equal to value	x		x	x	
<b>== (equal to)</b>	Condition active if gate value is equal to value	x	x	x	x	x
<b>!= (not equal to)</b>	Condition active if gate value is not equal to value	x		x	x	
<b>At least one bit != value</b>	Condition active if at least one bit is not equal to value	x			x	
<b>At least one bit == value</b>	Condition active if at least one bit is equal to value	x			x	
<b>All bits == value</b>	Condition active if all bits are equal to value	x			x	
<b>Device disabled</b>	Condition active if the device associated to the gate is disabled	x	x	x	x	

**Color replacement:** allows to specify the color substitutions of the Bitmap (only in case of .bmp files). There are three choices:

- None: no color substitution will be performed
- Transparent color: permits to indicate the color that will be made transparent. The background behind the Bitmap will appear where the Bitmap has this color. This option is useful when is needed to put some bitmaps on the template to give information on the objects behind of them (alarms, anomalies, ...)
- Replace color ... with ... : With this option you can specify the color that will replace another color of the bitmap

For a correct management of transparent color and the replacing of colors, it is needed to use bitmap with 256 colors.

If you use the transparent color, the dimensions of the Bitmap will be fixed to the original ones.

**Tab num:** number that indicates the order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

**X Animation:** gate that must be used to read the X coordinate (in pixels) of the Bitmap. Using this property it is possible to make the Bitmap moving across the template using the position given by the specified gate. The value 0 corresponds to the left edge of the template.

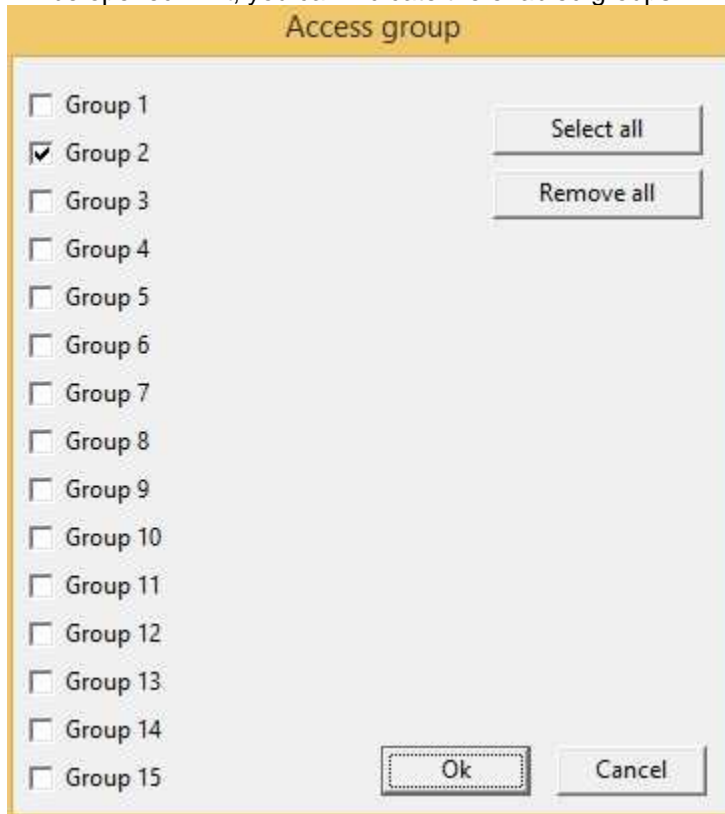
**Y Animation:** gate that must be used to read the Y coordinate (in pixels) of the Bitmap. Using this property it is possible to make the Bitmap moving across the template using the position given by the specified gate. The value 0 corresponds to the top edge of the template.

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always



visible if no conditions are specified.

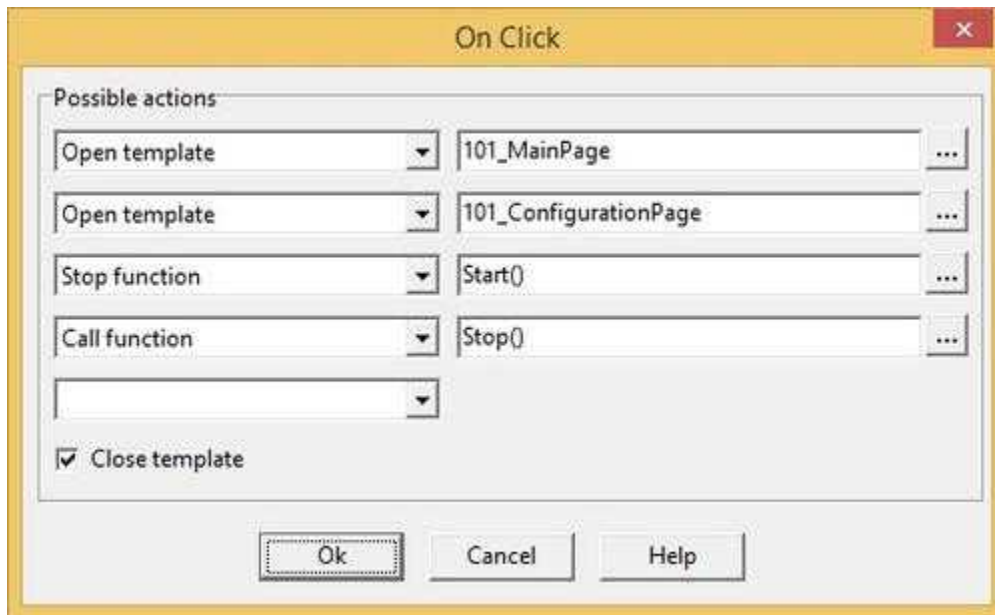
**Access group:** users groups enabled to operate on the object in runtime mode (enables "On Click" and "On Double Click" functions). By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled (enables "On Click" and "On Double Click" functions) if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

**On Click:** operation to carry out when the user click on the object. Next figure show the window through which to specify the list of the operations to carry out:

- Call function: call the indicated function.
- Stop function: stop the indicated function (if this is on).
- Open template: open the indicated template.
- Apply changes: apply the changes in all the object brothers or child components that have the "Need apply" property set to "Yes" (the new values will be written in the gates).
- Undo changes: undo the changes in all the object brothers or child components that have the "Need apply" property set to "Yes".
- Close template: close the current template.



**On Double Click:** operation to carry out when the user double click on the object. Refer to "On Click" property to specify the list of the operations to carry out.

## 8.3 BkBitmap

The BkBitmaps (background bitmaps) represent images that can be inserted in the template in order to show static graphic elements (that don't have to change during the supervision).



### Properties (TemplateBuilder)

**Left:** horizontal position of the top left corner of the BkBitmap (in pixels).

**Top:** vertical position of the top left corner of the BkBitmap (in pixels).

**Width:** width of the BkBitmap (in pixels).

**Height:** height of the BkBitmap (in pixels).

**Image:** image to display in the BkBitmap. To specify the image just press the button[...] on the properties row.

Supported image type are : .bmp, .jpg, .png, .gif (not animated).

## 8.4 Button

The fundamental role of the Button is to carry out some operations (during the supervision phase) when it is pressed by the user. For example you can open or close a template, call a function or apply changes.



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the Button (in pixels).

**Top:** vertical position of the top left corner of the Button (in pixels).

**Width:** width of the Button (in pixels).

**Height:** height of the Button (in pixels).

**Description:** description of the Button (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Cursor:** shape that the cursor of the mouse must have when it goes over the Button during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click. (for details see the cursor table)

**Label:** text displayed by the Button.

**Font:** font to use for the Label text. Press the button on the properties line to show the dialog for the choice of the font, size, style (normal, bold or italic) and effects (underlined and strikeout).

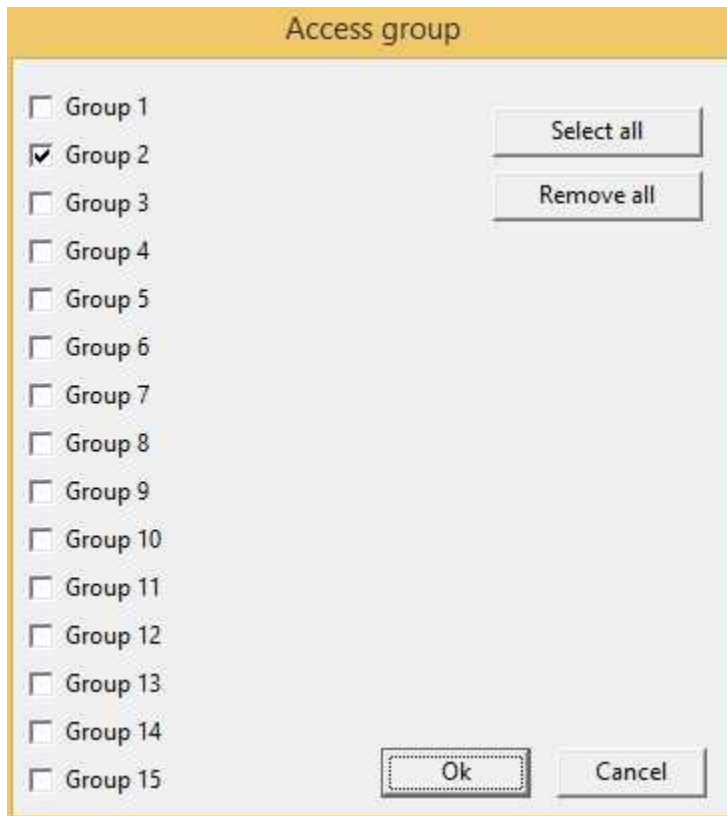
**Function Key:** Button binding with a keyboard function key (F1, F2, F3..F9); pressing the selected function key is equivalent to pressing the Button.

**Tab num:** number that indicates the order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

**Help file name :** if in this field is specified a valid ".CHM" file name and it is present an HtmlHelp object in one of the templates currently showed on the screen, when the Button is clicked or receive the focus, the .CHM file will be showed in the HtmlHelp window. The path can be specified either in absolute (eg. "c:\Data\Help.chm") or in relative way (eg. "..\Data\Help.chm"); using the relative way, the base directory will be the application's templates directory. See HtmlHelp object for more details.

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

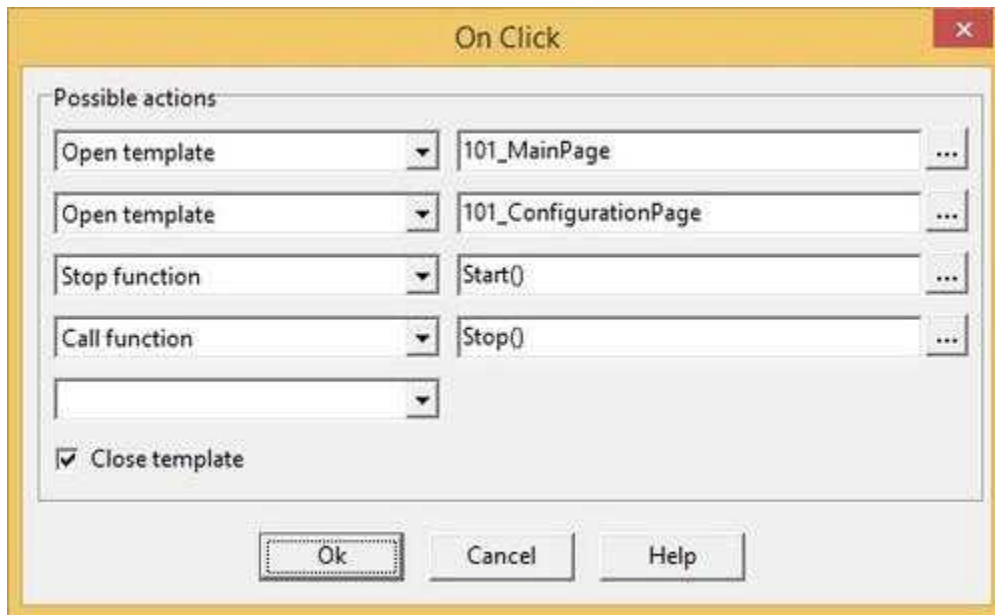
**Access group:** users groups enabled to operate on the object in runtime mode (enables "On Click" function). By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled (enables "On Click" function) if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

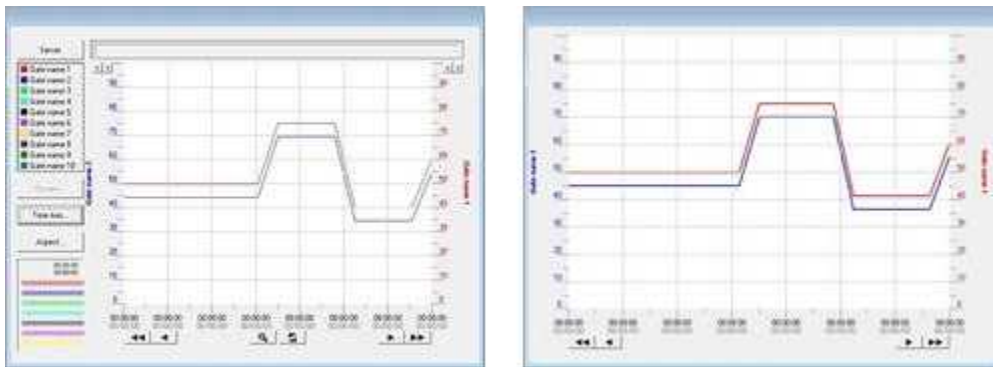
**On Click:** operation to carry out when the user click on the object. Next figure show the window through which to specify the list of the operations to carry out:

- Call function: call the indicated function.
- Stop function: stop the indicated function (if this is on).
- Open template: open the indicated template.
- Apply changes: apply the changes in all the object brothers or child components that have the "Need apply" property set to "Yes" (the new values will be written in the gates).
- Undo changes: undo the changes in all the object brothers or child components that have the "Need apply" property set to "Yes".
- Close template: close the current template.



## 8.5 Chart

The Chart component allows to show the chart of a group of gates. It is possible to specify the group of gates to display in the chart and indicate the time interval the chart is taking into consideration. For more details on this component refer to the paragraph speaking about charts in the *RunTime* chapter.



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder..

**Left:** horizontal position of the top left corner of the Chart (in pixels).

**Top:** vertical position of the top left corner of the Chart (in pixels).

**Width:** width of the Chart (in pixels).

**Height:** height of the Chart (in pixels).

**Chart group name:** name of the chart group that contains the set of gates to draw in the chart.(Charts groups)

**Enable groups:** indicates whether to allow the user to change during the runtime phase the group of gates to display.

**Save group update:** if this option is enabled, then the changes that are made on the chart group in runtime mode are considered permanent changes, otherwise are considered temporary changes.

**Autorange:**

- No: Y scale is the scale defined for that curve in the chart group.
- Individual: Y scale is calculated using the maximum and minimum Y value of the single curve, found in the width time showed in the chart window .
- Common: Y scale is calculated using the maximum and minimum Y value of all curves found in the width time showed in the chart window.

**Enable points:** indicates whether or not to draw the points that constitutes the chart

**Line width:** thickness of the line that draws the chart

**Interpolation:** interpolation to use to draw the graph

**Time range options:** time range of the chart. There are four choices for TimeRange:

- Normal: the software will show charts with the time interval that starts at the time of opening of the template and that lasts for the specified TimeRange.
- External: the length of the x axis is set externally by using the code (see the function ChartSetTimeRange)
- External start: the start of the x axis of the chart is set from the code ( see the function ChartSetTimeRange), and its length is given by the specified TimeRange
- External stop: the end of the x axis of the chart is set from the code ( see the function ChartSetTimeRange), and its length is given by the specified TimeRange.

**Enable OnLine:** indicates if the x axis of the chart must shift in order to follow the variations of the gates in realtime

**Enable graph. sel.:** indicates if the graphics scale selection must be enable.

**Enable mouse right:** indicates if the right button function menu must be enabled on the chart object.

**Enable grid:** indicates whether or not to draw the grid of the chart

**Y grid division:** grid division of Y axis.

**Enable Zoom:** indicates if the graphics Zoom function must be enabled.

**Show Reset Zoom button :** indicates if the Reset Zoom button must be shown .

**Show time buttons:** indicates if buttons for graphics time scrolling must be shown.

**Show OnLine button:** indicates if the button that enable Online mode must be shown.

**Show Y axis labels:** indicates if the labels of the selected scale must be shown.

**Show right Y :** indicates if the right vertical axis must be shown.

**Show only time:** indicates if must only the time must be shown on the horizontal axis or also the date.

**Show Server button:** indicates if the button that allow to read historical data from server PC must be shown.

**Default server:** indicates from which server historical data must be read.

- "Local": historical data must be read from the local computer.
- "Channel\_x": historical data must be read from the server specified in channel configuration.

**Show Groups button:** indicates if and where to show the button that allow chart groups selection.

**Show Time Axis button:** indicates if and where to show the button that allow time range selection.

**Show Aspect button:** indicates if and where to show the button that allow charts aspect selection (Line width, show points, etc.).

**Window Background color:** chart main window background color.

**Graphic Background color:** chart window background color.

**Grid color:** grid color.

**Scale color:** axis color.

**Time scale color:** color of the time labels.

**Date scale color:** color of the date labels.

**Show Legend window:** indicates if and where to show the Legend window.

**Legend description:** indicates if Gate Name Or Gate Description must be used in the Legend window.

**Legend Font:** font to use in the Legend window.

**Legend text color:** text color to use in the Legend window.

**Legend background color:** color to use in the Legend window background.

**Legend items:** maximum number of items to display in the Legend window: this parameter is used to calculate the maximum size of the window. If "Auto" is selected then "Legend items" will coincide with the number of elements in the group currently selected.

**Legend columns:** number of columns on which to display the elements on the legend: this parameter is used to calculate the maximum size of the window.

**Show Cursor window:** indicates if and where to show the Cursor window.

**Cursor type:** indicates how to display the cursor coordinates (Positional / Analytical).

- "Positional" shows the Y coordinate of the point where is the mouse pointer.
- "Analytic" shows the Y coordinate of each graph with respect to the X coordinate indicated by the mouse pointer.

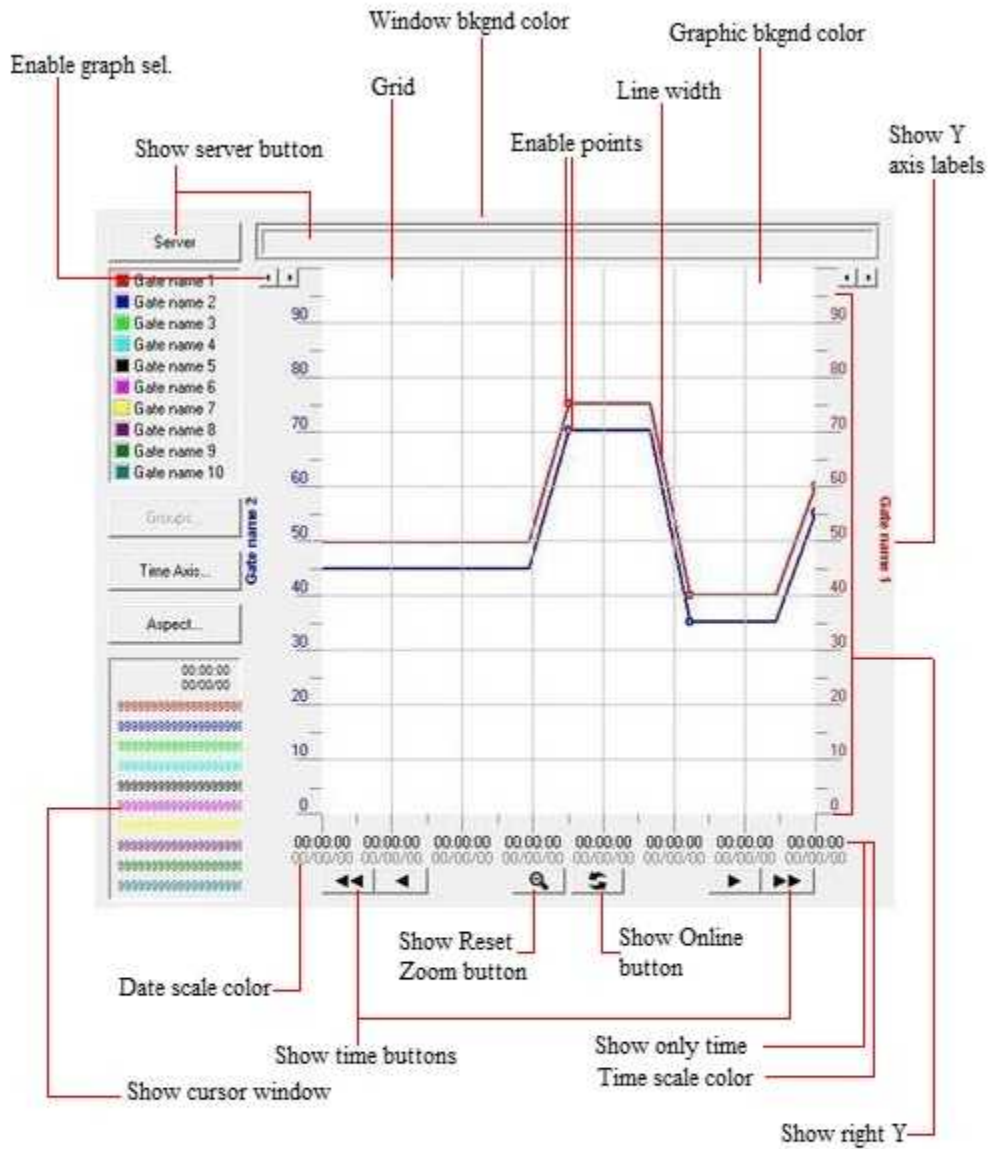
**Cursor Font:** font to use in the Cursor window.

**Cursor text color:** text color to use in the Cursor window.

**Cursor background color:** color to use in the Cursor window background.

**Cursor items:** maximum number of items to display in the Cursor window: this parameter is used to calculate the maximum size of the window. If "Auto" is selected then "Cursor items" will coincide with the number of elements in the group currently selected.

**Cursor columns:** number of columns on which to display the elements on the Cursor: this parameter is used to calculate the maximum size of the window.



### Commands (CodeBuilder)

There is the possibility to send some commands to Chart object by using "**TObjFunction(int ObjId, int Function)**" CodeBuilder instruction.

Parameters:

**int ObjId** : the number that identify the component (see ID property of Chart object).



**int Function** : function to perform.

Function	Description
1	Show server selection window
2	Show configure graphic set window
4	Show configure time window
5	Show configure chart aspect window
7	Forward time
8	Fast forward time
9	Rewind time
10	Fast rewind time
11	Reset zoom factor
12	Set current time (online mode)
13	Stop loading historical files

## 8.6 CheckBox

The CheckBox is a component very similar to the Switch. In fact, it is used for allowing the user to operate on the values of a gate according to two default states. So the CheckBox too can have two states: CheckBox selected or not. To allow the CheckBox to work properly, you have to specify on which gate to write, and what to write when the CheckBox is selected, and when it is not selected. As for the Switch, *RunTime* checks if during the supervision the value read from the gate associated with the CheckBox verifies the ON condition: if so, the CheckBox is selected, if not is deselected.



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the CheckBox (in pixels).

**Top:** vertical position of the top left corner of the CheckBox (in pixels).

**Width:** width of the CheckBox (in pixels).

**Height:** height of the CheckBox (in pixels).

**Description:** description of the CheckBox (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**BkColor:** background color of the CheckBox. To specify the desired color, use the button on the properties line, and choose the color.

**Cursor:** shape that the cursor of the mouse must have when it goes over the CheckBox during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

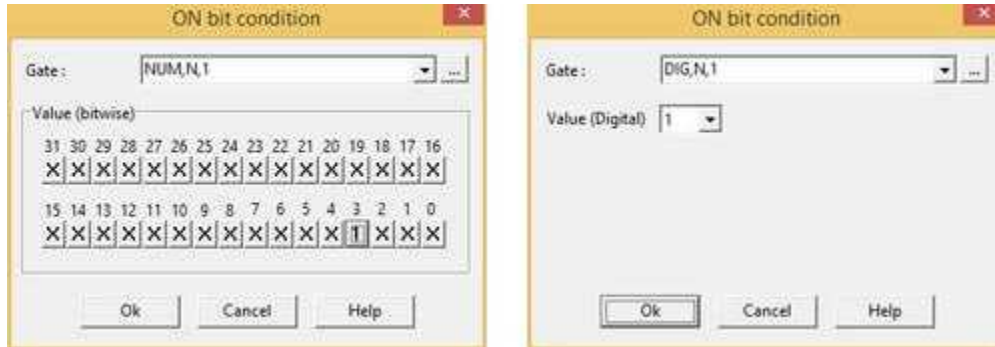
**Label:** label of the CheckBox.

**Off label:** label of the CheckBox when this is not selected. If the off label is missing, when CheckBox is not selected the normal label will be shown.

**X label:** label of the CheckBox when the value of the gate associated to the ON condition is not corresponding to the ON neither OFF condition.

**Need apply:** it indicates whether the status change of the CheckBox will have to be confirmed from outside with a Button (Yes), or will be applied immediately (No).

**ON condition:** it indicates the gate to be written on following the status change of the CheckBox, and the value to write. Push the button on the properties row to show one of the windows in figure below. If a numeric gate have been selected, then will be possible to set the ON value by selecting which bits must be set to 1, which will be set to 0 and which must not be considered. If a digital gate have been selected then must be specified the ON value of a single bit. In both cases, when the CheckBox is set to OFF (unselected), the negated ON value will be written.

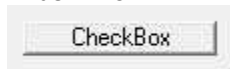


**Style:** style of the CheckBox. There are three available choices:

- Standard



- Push like



- Right button



**Tab num:** number that indicates the order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

**Help file name :** if in this field is specified a valid ".CHM" file name and it is present an HtmlHelp object in one of the templates currently showed on the screen, when the CheckBox is clicked or receive the focus, the .CHM file will be showed in the HtmlHelp window. The path can be specified either in absolute (eg. "c:\Data\Help.chm") or in relative way (eg. "..\Data\Help.chm"): using the relative way, the base directory will be the application's templates directory. See HtmlHelp object for more details.

**Show** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group:** users groups enabled to operate on the object in runtime mode. By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

## 8.7 ComboBox

With the ComboBox it is possible to offer the user of the supervision page a list of alternatives among which to choose. Each of them is associated with a condition to be imposed on the value of a gate. By selecting an item, the user write on a gate the corresponding value. The opposite can also be done: *RunTime* checks during the supervision if one of the conditions in the ComboBox items is verified; if so, the corresponding item is selected in the ComboBox .



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the ComboBox (in pixels).

**Top:** vertical position of the top left corner of the ComboBox (in pixels).

**Width:** width of the ComboBox (in pixels).

**Height:** height of the ComboBox (in pixels).

**Visible items:** number of items that will be visible in the ComboBox selection box.

**Description:** description of the ComboBox (maximum 150 characters). The description will be shown

when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Cursor:** shape that the cursor of the mouse must have when it goes over the ComboBox during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

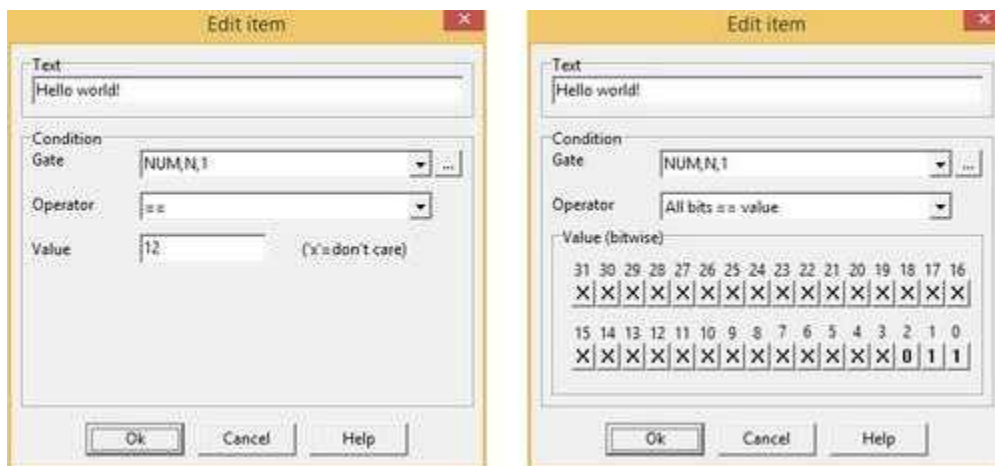
**Font:** font to use in the ComboBox text. Press the button on the properties row to show the dialog box for the choice of font, size, style (normal, bold or italic) and effects (underlined and strikethrough).

**Need apply:** it indicates whether the status change of the ComboBox will have to be confirmed from outside with a Button (Yes), or will be applied immediately (No).

**Items:** list of elements in the ComboBox; each is associated with its condition. To define the elements, click on the button on the properties row and you'll see the window in figure below.



Here is the list of the elements in the ComboBox: it is possible to add, remove, delete or sort the elements using the buttons on the right. When you choose to add or modify an element, the window in follow figure will be shown.



First of all you have to specify the text to display in the ComboBox. Then you have to select the condition type between "==" and "All bits == value" and the gate to which the element condition is referring. For "==" you have to indicate the value to write on the gate, if the element is selected during

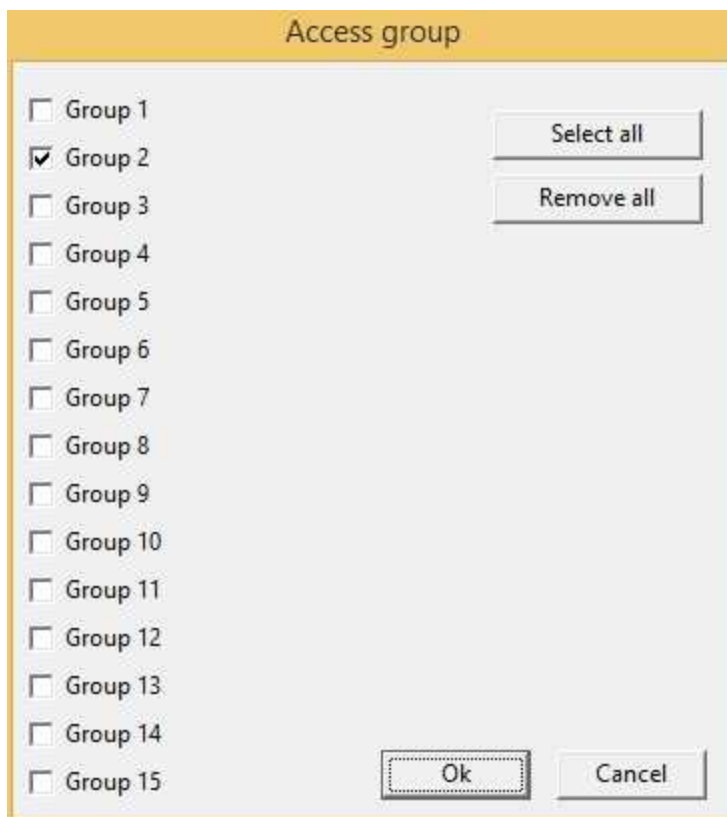
the supervision phase; (you are allowed to leave indefinite some figures using the 'x'; the value of these figures won't be changed). If you use the "All bits = value" condition, you have to indicate which bits must be set to 1, which to 0 and which must be not modified.

**Tab num:** number that indicates the order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

**Help file name :** if in this field is specified a valid ".CHM" file name and it is present an HtmlHelp object in one of the templates currently showed on the screen, when the ComboBox is clicked or receive the focus, the .CHM file will be showed in the HtmlHelp window. The path can be specified either in absolute (eg. "c:\Data\Help.chm") or in relative way (eg. ".\Data\Help.chm"): using the relative way, the base directory will be the application's templates directory. See HtmlHelp object for more details.

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group:** users groups enabled to operate on the object in runtime mode. By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

## 8.8 Edit

The Edit component allows the user of the supervision page to indicate a value to give to a gate. You can set the Edit so that it accepts only some strings (or numbers); this way the value will be written on the gate when the user leaves the Edit or when he confirms the entered data with a button.



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the Edit (in pixels).

**Top:** vertical position of the top left corner of the Edit (in pixels).

**Width:** width of the Edit (in pixels).

**Height:** height of the Edit (in pixels).

**Description:** description of the Edit (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**BkColor:** background color of the Edit. To specify the desired color, use the button on the properties row, and choose the color.

**Cursor:** shape that the cursor of the mouse must have when it goes over the Edit during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click. (for details see the cursor table)

**TextColor:** color of the Edit text. To specify the desired color, use the button on the properties row, and choose the color.

**Font:** font to use for the Edit text. Press the button on the properties row to show the dialog box for the choice of font, size, style (normal, bold or italic) and effects (underlined and strikeout).

**Gate:** gate on which to write the value that the template user writes in the Edit. Press the button on the properties row in the *property editor* to show the list of the template gates. From it you can select the desired gate.

**Need apply:** it indicates when the value changed by the user must be written on the gate. There are two options:

- No: the value will be written on the gate when the user presses the *Return* or *Tab* key.
- Yes: the value will be written on the gate when the user gives confirmation from the outside, that is to say when he uses Button with the "On Click" property set to "*Apply changes*".

**Password:**

- No: will be displayed the char effectively inserted by user.
- Yes: a '\*' char will be displayed in place of the char effectively inserted by user.

**Validation string:** string that allows to define a scheme the user will have to follow for entering the data in the Edit. When you specify the validation string, you will have to consider the conventions described in the table:

Character	Description
#	Accepts only a numeric digit
?	Accepts only a letter (upper or lower case)
&	Accepts only a letter and force it to upper case
@	Accepts all characters (letter, numeric digit,...)
!	Accepts all characters (letter, numeric digit,...) and force it to upper case
[]	Optional character
Other	All other characters are taken literally. If it is needed to take literally one of the above characters, it must be preceded by the ; character (e.g. ";@")

For example, if you want the user to be able to enter in the Edit only 3-figured numbers, signed or unsigned, with at most two decimals, you can use the validation string "[+][-]###.[#]#[#]".

**Horizontal align:** text alignment. There are three options: Left, Center and Right.

**Upper/Lower case:** there are three options: Normal,Uppercase,Lowercase.

**Tab num:** number that indicates the order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

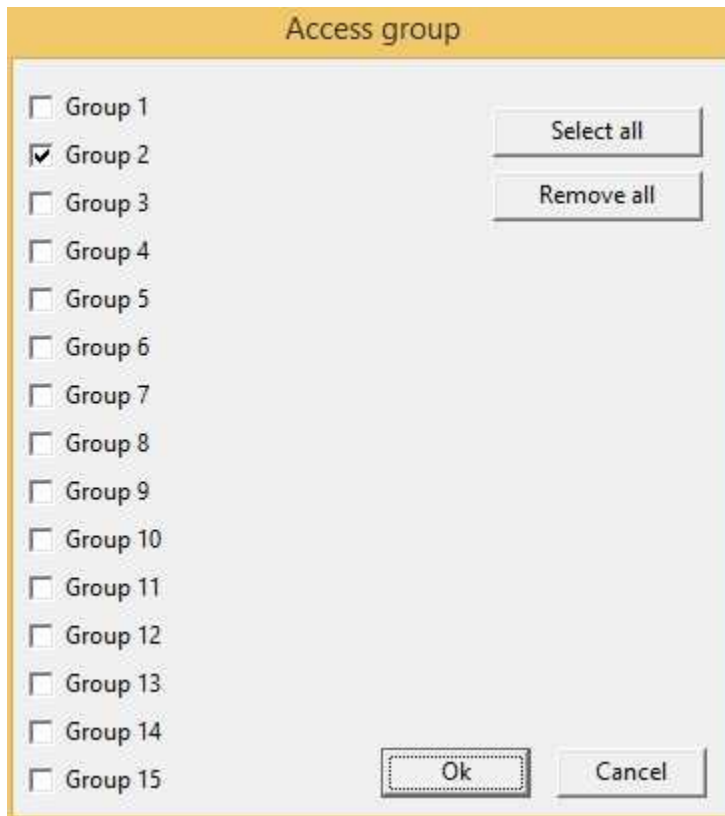
**Keyboard:** keyboard to view when the edit object get focus control.If the application supports several languages, then will be automatically loaded the keyboard named (name)+"\_" +(current language). (for example : "AlphanumericKeyb\_English") This function is utilized for applications that are executed on computer whit no hardware keyboard and with touch screen monitor. The keyboard must be defined using keyboard Builder tools. Following there are some examples of touch screen keyboard:



**Help file name :** if in this field is specified a valid ".CHM" file name and it is present an HtmlHelp object in one of the templates currently showed on the screen, when the Edit is clicked or receive the focus, the .CHM file will be showed in the HtmlHelp window. The path can be specified either in absolute (eg. "c:\Data\Help.chm") or in relative way (eg. "..\Data\Help.chm"): using the relative way, the base directory will be the application's templates directory. See HtmlHelp object for more details.

**Show** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group:** users groups enabled to operate on the object in runtime mode. By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

## 8.9 FileListBox

The FileListBox component allows the user to select a file within a directory. The files of the directory will be shown in a list where the user can select the desired file. The selected filename will be written into the string gate associated to the component.



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the FileListBox (in pixels).



**Top:** vertical position of the top left corner of the FileListBox (in pixels).

**Width:** width of the FileListBox (in pixels).

**Height:** height of the FileListBox (in pixels).

**Description:** description of the FileListBox (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Font:** font to use for the FileListBox text. Press the button on the properties row to show the dialog box for the choice of font, size, style (normal, bold or italic) and effects (underlined and strikethrough).

**Gate:** gate on which will be copied the name (with extension) of the file selected from the user. Press the button on the properties row in the *property editor* to show the list of the template gates. From it you can select the desired gate.

**Gate path:** gate on which will be copied the name of the file directory of the selected file. Press the button on the properties row in the *property editor* to show the list of the template gates. From it you can select the desired gate.

**Need apply:** it indicates when the value changed by the user must be written on the gate. There are two options:

- No: the value will be written on the gate when the user presses the *Return* or *Tab* key.
- Yes: the value will be written on the gate when the user gives confirmation from the outside, that is to say when he uses Button with the "On Click" property set to "Apply changes".

**Path:** path of the directory to show into the component. The path can be specified either in absolute (eg. "c:\Data\*.dat") or in relative way (eg. "..\Data\*.dat"): using the relative way, the base directory will be the application's templates directory. It is necessary to indicate the type of the file to show using the appropriate wildcards (eg. "\*.dat", "\*.tm?", "\*.\*", ...).

**Sorted:** indicates if the files must be shown in an alphabetically ordered way (Yes) or not (No).

**Show files:** indicates if the files must be shown (Yes) or not (No).

**Show directories:** indicates the how to show directories.

- No: don't show directories.
- Yes, with filter: shows only directory with the extension in according with the file type specified in the "Path" property.
- Yes, without filter: shows all directories.

**Show hidden:** indicates if the hidden files must be shown (Yes) or not (No).

**Show extension:** indicates if the file extension must be shown (Yes) or not (No).

**Allow change dir,:** indicates if the directory change is allowed (Yes) or not (No).

**Tab num:** number that indicates the order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

**Help file name :** if in this file is specified a valid ".CHM" file name and it is present an HtmlHelp object in one of the templates currently showed on the screen, when the FileListBox is clicked or receive the focus, the .CHM file will be showed in the HtmlHelp window. The path can be specified either in absolute (eg. "c:\Data\Help.chm") or in relative way (eg. "..\Data\Help.chm"): using the relative

way, the base directory will be the application's templates directory. See HtmlHelp object for more details.

**Show** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group:** users groups enabled to operate on the object in runtime mode. By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

#### Commands (CodeBuilder)

Following properties can be read also from language (see Code Builder help)

Property	Function
Selected path	TObjGetPropertyString(Id,"PathSelected")
Selected file	TObjGetPropertyString(Id,"FileSelected")

```
Function void GetSelectedFile()
    string PathName;
    string FileName;
    PathName=TObjGetPropertyString(100,"PathSelected");
    FileName=TObjGetPropertyString(100,"FileSelected");
    ...
```

*end*

Following properties can be set also from language (see Code Builder help)

Property	Function
Path	TObjSetPropertyString(Id,"Path",...)

Before setting all required properties, it is necessary to call **TObjBeginUpdate(Id)** function. After setting all required properties, it is necessary to call **TObjEndUpdate(Id)** function to update the object.

*Example:*

```
...
TObjBeginUpdate(100);
TObjSetPropertyString(100,"Path","C:\Documents\*.");
TObjEndUpdate(100);
...
```

There is the possibility to send some commands to FileListBox object by using "**TObjFunction(int ObjId, int Function)**" CodeBuilder instruction.

Parameters:

**int ObjId** : the number that identify the component (see ID property of Chart object).

**int Function** : function to perform.

Function	Description
1	Force a directory rescan

*Example:*

```
...
TObjFunction(100,1);
...
```

## 8.10 Frame

The Frame is an object that is normally used for containing other objects. For this reason it's a component that can have children components. To place a component whatsoever inside the Frame, you just have to position the latter in the template, then select the component you want to place inside the Frame from the tool bar and click inside the Frame. The new component will be created inside the Frame, and so will be called child of the Frame. This means that the child component cannot be moved outside its parent's boundaries, but above all that, moving the parent, all its children will be moved with it, and that deleting the parent, its children will be removed too. In figure it is shown an example of how a Frame can be used for containing children components and thus forming a components hierarchy.



### Properties (TemplateBuilder)

**ID:** number than can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the Frame (in pixels).

**Top:** vertical position of the top left corner of the Frame (in pixels).








**Width:** width of the Frame (in pixels).

**Height:** height of the Frame (in pixels).

**Description:** description of the Frame (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**BkColor:** background color of the Frame. To specify the desired color, use the button on the properties row, and choose the colour. "clNone"color means transparent frame.

**Style:** style of the Frame. There are seven available styles. They are described in the table.

Frame Style	Appearance
Plain	
Raised (thin)	
Recessed (thin)	
Embossed	
Grooved	
Raised (thick)	
Recessed (thick)	

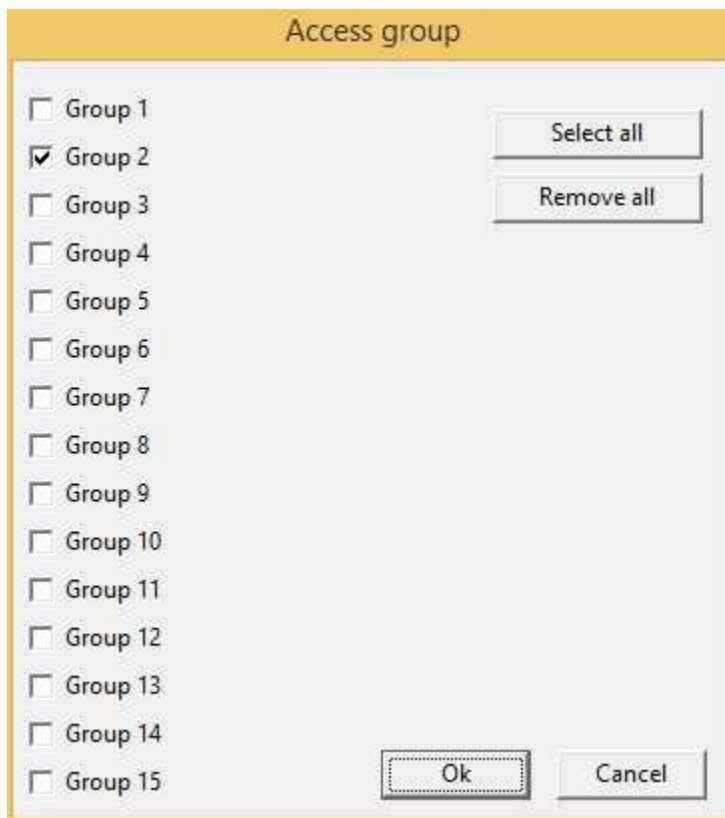
**Tab num:** number that indicates the order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

**X Animation:** gate that must be used to read the X coordinate (in pixels) of the Frame. Using this property it is possible to make the Frame moving across the template using the position given by the specified gate. The value 0 corresponds to the left edge of the template.

**Y Animation:** gate that must be used to read the Y coordinate (in pixels) of the Frame. Using this property it is possible to make the Frame moving across the template using the position given by the specified gate. The value 0 corresponds to the top edge of the template.

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

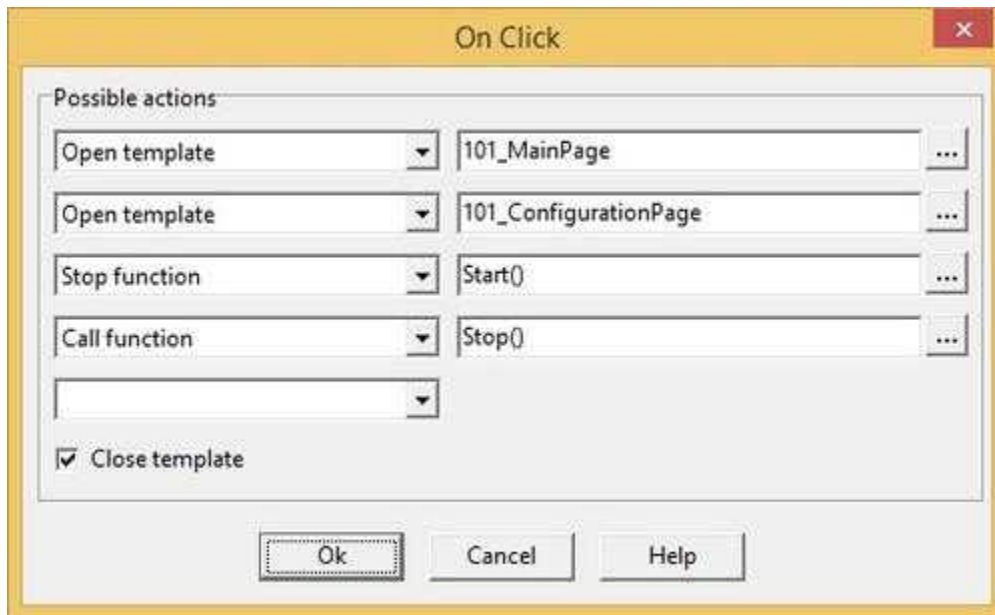
**Access group:** users groups enabled to operate on the object in runtime mode (enables "On Click" and "On Double Click" functions). By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled (enables "On Click" and "On Double Click" functions) if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

**On Click:** operation to carry out when the user click on the object. Next figure show the window through which to specify the list of the operations to carry out:

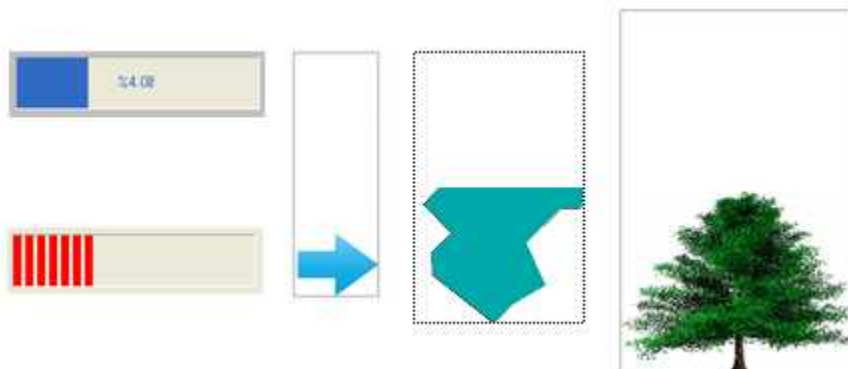
- Call function: call the indicated function.
- Stop function: stop the indicated function (if this is on).
- Open template: open the indicated template.
- Apply changes: apply the changes in all the object brothers or child components that have the "Need apply" property set to "Yes" (the new values will be written in the gates).
- Undo changes: undo the changes in all the object brothers or child components that have the "Need apply" property set to "Yes".
- Close template: close the current template.



**On Double Click:** operation to carry out when the user double click on the object. Refer to "On Click" property to specify the list of the operations to carry out.

## 8.11 Gauge

The Gauge is a component that allows to display graphically the value of a gate. In fact, with the Gauge are associated a gate, a minimum and a maximum value: during the supervision phase the Gauge will display a bar indicating where to position the value read from the gate within the range limited by the minimum and maximum values. Besides, to give more information, it is possible to insert a text in the Gauge; this can also contain the value read from the gate. Following some examples:



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the Gauge (in pixels).

**Top:** vertical position of the top left corner of the Gauge (in pixels).

**Width:** width of the Gauge (in pixels).

**Height:** height of the Gauge (in pixels).

**Description:** description of the Gauge (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**BkColor:** background color of the Gauge. To specify the desired color, use the button on the properties row, and choose the color. "clNone" color means transparent Gauge.

**Cursor:** shape that the cursor of the mouse must have when it goes over the Gauge during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for the details see the cursor table).

**Color:** color of the Gauge bar.

**Font:** font to use in the Gauge text. Push the button on the properties row to show the dialog box for the choice of font, size, style (normal, bold or italic) and effects (underlined and strikethrough).

**Gate:** gate from which to read the value that will be displayed by the Gauge during the supervision phase. Since the Gauge operates with the values (numerical) read from the gates, you can indicate only numerical or composed gates.

**Minimum value:** minimum value of the Gauge (the lower end of the scale). If the Gauge reads this value from the associated gate, a 0-sized bar will be shown.

**Maximum value:** maximum value of the Gauge (the upper end of the scale). If the Gauge reads this value from the associated gate, it will be shown a bar whose size is the same as the Gauge width (or height, depending on the Gauge direction). If the minimum and maximum values are both zero, *RunTime* will consider as limits of the Gauge the minimum and maximum values read from the gate properties.

**Label:** text displayed in the standard Gauge. If in the label text you want to include the value read from the gate linked to the Gauge "**Gate**" property, you have to resort to the following conventions:  
For Numeric, Digital and Compound gates:

- **"%x.ylf"** If you want to specify in detail how to display the number, where:
  - **x** is optional and indicates the total number of digits to display. If not present, will be shown all the digits of the value read from the gate, and if it is preceded by 0 then 0 will appear before the number to reach the number of digits specified.
  - **y** is optional and indicates the number of digits to display after the decimal point. If y is equal to "\*" then is used as the number of decimal places the number specified for the gate linked to the "**Gate**" field of the Gauge.  
Some examples will make things clearer:  
"%5.2lf" produce 123.45  
"%5.0lf" produce 123  
"%07.2lf" produce 00123.45  
"%7.\*lf" produce 123.456 if the number of decimal digits specified for the linked gate (in Gate Builder) is 3.  
For example, if you need to show a temperature read from a numeric gate, using a Gauge, it is sufficient to link the desired gate to the Gauge "**Gate**" property and set the Gauge "**Label**" property with the string "Temperature:% 5.1lf ° C". If during supervision the gate value will be 25.7, the label will display the string "Temperature: 25.7 ° C".
- **"%g"** Using this format, the number will be displayed in order to take up as little space as possible (possibly uses the exponential form).
- **"%d"** for integers.
- **"%xd"** for integers with a maximum number of characters: the meaning of the parameter x is identical to that seen above.

**Style:** it indicates whether the Gauge will have to look like a normal indication bar or like an image moving, clipping or stretching across the Gauge depending on the value of the gate associated to the "Gate" property.

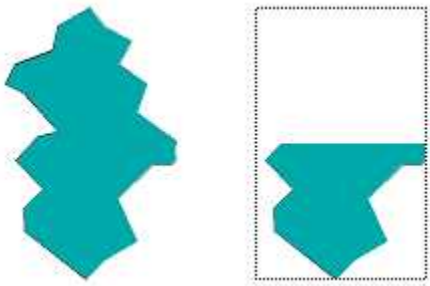
- Standard



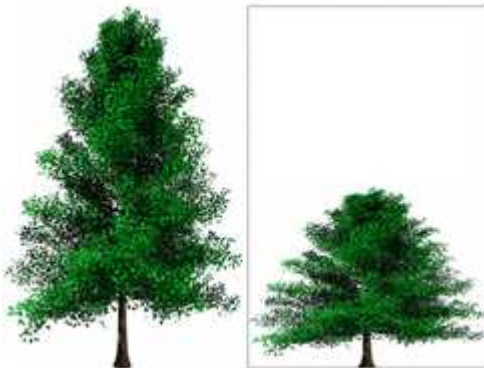
- Image moving



- Image clipping

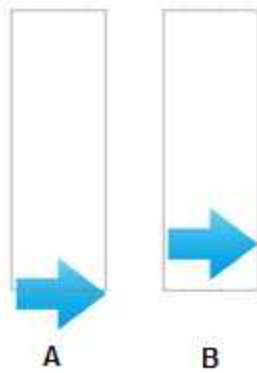


- Image stretching



In case of "Style" = Image moving, it is possible to specify a minimum and maximum offset to allow to move the image outside the Gauge delimiter. In the following examples, case "A" has a Minimum offset of 40 pixels, while case "B" has a minimum offset of -10 pixels





**Segments:** number of segments forming the Gauge bar in case of "Style" property = Standard.

**Space between segments:** number of pixels separating a segment from another in case of "Style" property = Standard.

**Direction:** Gauge directions. There are four options:

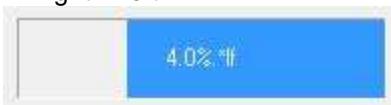
- Left - Right



- Bottom- Top



- Right - Left



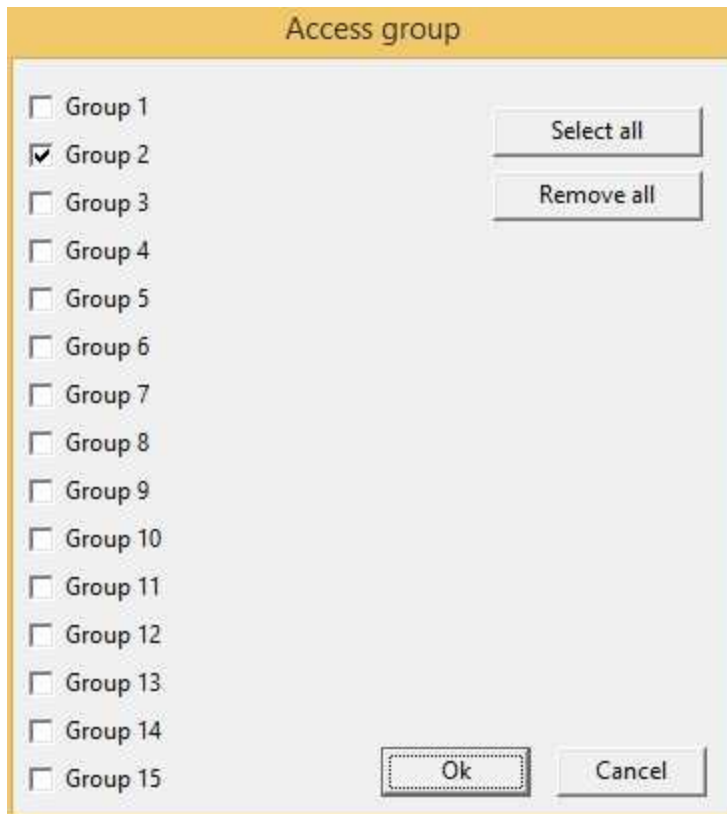
- Top - Bottom



**Frame:** type of frame enclosing the Gauge. There are the same options as for the Frame's (see table in the Frame section).

**Show** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

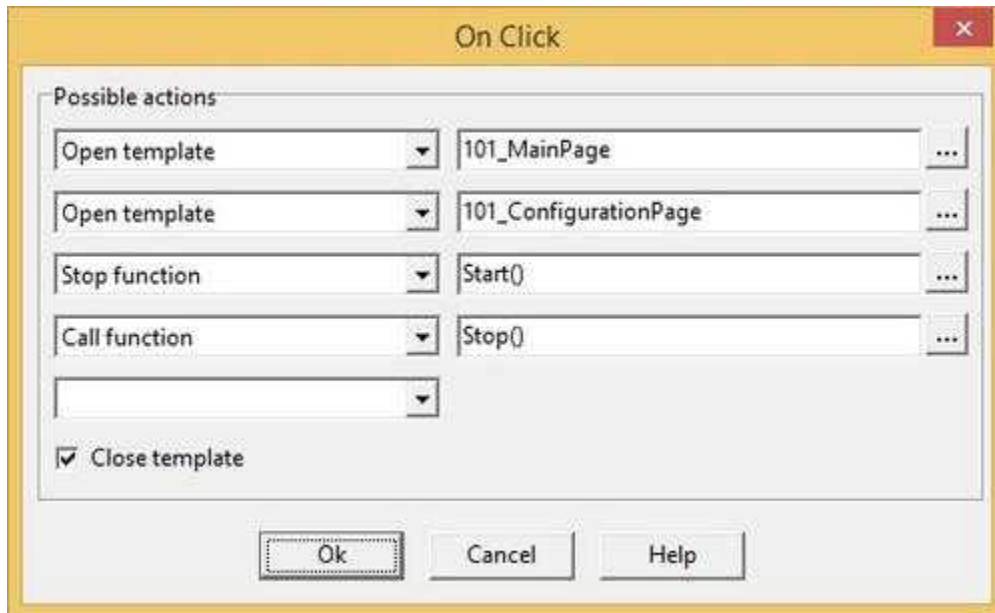
**Access group:** users groups enabled to operate on the object in runtime mode (enables "On Click" and "On Double Click" functions). By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled (enables "On Click" and "On Double Click" functions) if it is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

**On Click:** operation to carry out when the user click on the object. Next figure show the window through which to specify the list of the operations to carry out:

- Call function: call the indicated function.
- Stop function: stop the indicated function (if this is on).
- Open template: open the indicated template.
- Apply changes: apply the changes in all the object brothers or child components that have the "Need apply" property set to "Yes" (the new values will be written in the gates).
- Undo changes: undo the changes in all the object brothers or child components that have the "Need apply" property set to "Yes".
- Close template: close the current template.



**On Double Click:** operation to carry out when the user double click on the object. Refer to "On Click" property to specify the list of the operations to carry out.

## 8.12 GroupBox

The GroupBox is a particular type of Frame, for which you can use a label that will be applied on the upper part of the component. Like the Frame, the GroupBox can contain other objects.



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the GroupBox (in pixels).

**Top:** vertical position of the top left corner of the GroupBox (in pixels).

**Width:** width of the GroupBox (in pixels).

**Height:** height of the GroupBox (in pixels).

**Description:** description of the GroupBox (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**BkColor:** background color of the GroupBox. To specify the desired color, use the button on the properties row, and choose the color.

**TxtColor:** color of the GroupBox label. To specify the desired color, use the button on the properties row, and choose the color.

**Font:** font to use for the label. Press the button on the properties row, to show the dialog box for the choice of font, size, style (normal, bold or italic) and effects (underlined and strikeout).

**Label:** label that will be put on the GroupBox.

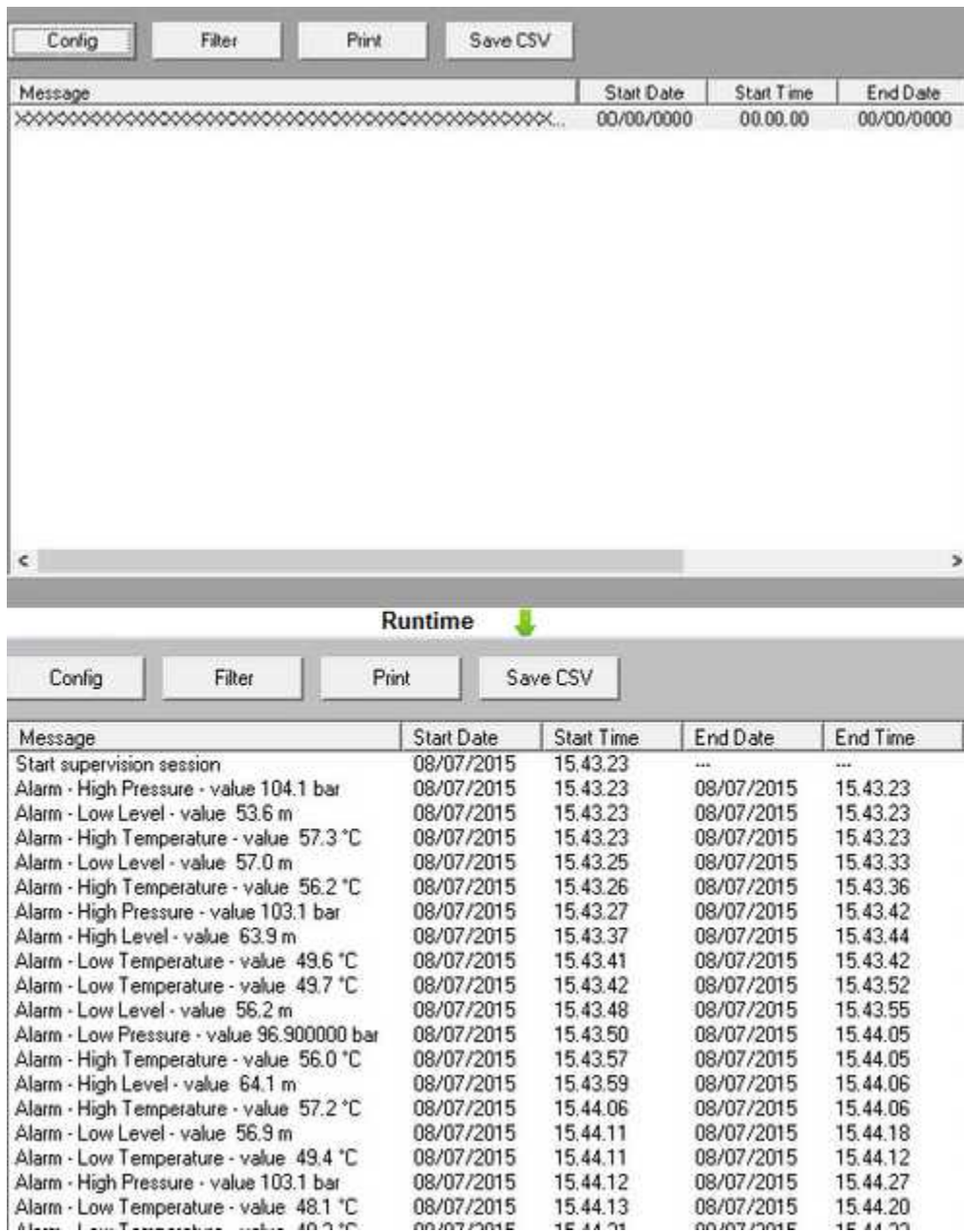
**Horizontal align:** it indicates where the label will be placed. There are three options: Left, Center and Right. Note that in *Template Builder* the label will be always on the left, while during the supervision phase the label will be put in the indicated position.

**Tab num:** number that indicates the order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

## 8.13 HistoricalAlarmsView

With the HistoricalAlarmsView component you can show the historical alarms page or the historical events page and also select one of the alarms displayed. For more details refer to the paragraph Alarms Historical DB in the *RunTime* chapter.



The screenshot displays two panels of a software interface. The top panel is empty, showing a header with columns: Message, Start Date, Start Time, and End Date. The bottom panel, labeled "Runtime" with a green arrow pointing down, contains a table with columns: Message, Start Date, Start Time, End Date, and End Time. The table lists various alarm events such as "Start supervision session", "Alarm - High Pressure", "Alarm - Low Level", "Alarm - High Temperature", "Alarm - Low Temperature", "Alarm - High Level", "Alarm - Low Temperature", "Alarm - Low Level", "Alarm - Low Pressure", "Alarm - High Temperature", "Alarm - High Level", "Alarm - High Temperature", "Alarm - Low Level", "Alarm - Low Temperature", "Alarm - High Pressure", and "Alarm - Low Temperature".

Message	Start Date	Start Time	End Date	End Time
Start supervision session	08/07/2015	15.43.23	---	---
Alarm - High Pressure - value 104.1 bar	08/07/2015	15.43.23	08/07/2015	15.43.23
Alarm - Low Level - value 53.6 m	08/07/2015	15.43.23	08/07/2015	15.43.23
Alarm - High Temperature - value 57.3 °C	08/07/2015	15.43.23	08/07/2015	15.43.23
Alarm - Low Level - value 57.0 m	08/07/2015	15.43.25	08/07/2015	15.43.33
Alarm - High Temperature - value 56.2 °C	08/07/2015	15.43.26	08/07/2015	15.43.36
Alarm - High Pressure - value 103.1 bar	08/07/2015	15.43.27	08/07/2015	15.43.42
Alarm - High Level - value 63.9 m	08/07/2015	15.43.37	08/07/2015	15.43.44
Alarm - Low Temperature - value 49.6 °C	08/07/2015	15.43.41	08/07/2015	15.43.42
Alarm - Low Temperature - value 49.7 °C	08/07/2015	15.43.42	08/07/2015	15.43.52
Alarm - Low Level - value 56.2 m	08/07/2015	15.43.48	08/07/2015	15.43.55
Alarm - Low Pressure - value 96.900000 bar	08/07/2015	15.43.50	08/07/2015	15.44.05
Alarm - High Temperature - value 56.0 °C	08/07/2015	15.43.57	08/07/2015	15.44.05
Alarm - High Level - value 64.1 m	08/07/2015	15.43.59	08/07/2015	15.44.06
Alarm - High Temperature - value 57.2 °C	08/07/2015	15.44.06	08/07/2015	15.44.06
Alarm - Low Level - value 56.9 m	08/07/2015	15.44.11	08/07/2015	15.44.18
Alarm - Low Temperature - value 49.4 °C	08/07/2015	15.44.11	08/07/2015	15.44.12
Alarm - High Pressure - value 103.1 bar	08/07/2015	15.44.12	08/07/2015	15.44.27
Alarm - Low Temperature - value 48.1 °C	08/07/2015	15.44.13	08/07/2015	15.44.20
Alarm - Low Temperature - value 49.2 °C	08/07/2015	15.44.21	08/07/2015	15.44.22

### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the HistoricalAlarmsView (in pixels).

**Top:** vertical position of the top left corner of the HistoricalAlarmsView (in pixels).

**Width:** width of the HistoricalAlarmsView (in pixels).

**Height:** height of the HistoricalAlarmsView (in pixels).

**Class 1:** will be shown only alarms or events wich have Class1 equal to the number specified in this

field.

**Class 2:** will be shown only alarms or events wich have Class2 equal to the string specified in this field.

**Class 3,Class 4,Class 5,Class 6,Class 7:** will be shown only alarms or events wich have ClassX equal to the value of the gates specified in this fields.

**Options:** allow to choose the object type and timerange.

For the Object type two choices are available:

- Alarms: will be shown Historical alarms.
- Events: will be shown Historical events.

For TimeRange there are four choices:

- Normal: will be shown alarms or events that have been generated in the interval time that starts at the time of opening of the template and that lasts for the specified TimeRange.
- External: the interval time is set externally by using the code (see the function HisViewSetTimeRange)
- External start: the start of the interval time for alarms or events visualization is set from the code ( see the function HisViewSetTimeRange), and its length is given by the specified TimeRange
- External stop: the end of the interval time for alarms or events visualization is set from the code ( see the function HisViewSetTimeRange), and its length is given by the specified TimeRange

**Show server button:** enable/disable "SERVER" button to select from which server computer alarms/events historical files must be loaded.

**Show Config button:** enable/disable "CONFIG" button to select the historical alarm view time range.

**Show Filter button:** enable/disable "FILTER" button to filter (through alarms class filter) which type of alarms/events must be showed.

**Show Print button:** enable/disable "PRINT" button to print the historical alarm view list on the default printer

**Show Save CSV button:** enable/disable "Save CSV" button to export the historical alarm view list on a formatted text file with TAB separator.

**Window Bkgrnd color:** window background color.

**Default server:** indicates from which server historical data must be read.

- "Local": historical data must be read from the local computer.
- "Channel\_x": historical data must be read from the server specified in channel configuration.

**Columns:** allow to select which columns to display in the object.

#### Commands (CodeBuilder)

There is the possibility to send some commands to HistoricalAlarms view object by using "

**TObjFunction(int ObjId, int Function)"** CodeBuilder instruction.

Parameters:

**int ObjId** : the number that identify the component (see ID property of the object).

**int Function** : function to perform.

Function	Description
1	Show server selection window
2	Show configure time window

3	Save CSV file
4	Print
5	Show alarm filter window
13	Stop loading historical files

Following properties can be set also from language (see Code Builder help)

Property	Function
Filter Class 1 ( *Example 2)	TObjSetPropertyInt(Id," <b>Class1</b> ",...)
Filter Class 2 ( *Example 2)	TObjSetPropertyString(Id," <b>Class2</b> ",...)

\*Example 2 : suppose to have a template with an HistoricalAlarmsView object with ID=100 and want to show only alarms having Class1=102 and Class2="Temp":

```
Function void SetAlarmsFilter()
    TObjBeginUpdate(100);
    TObjSetPropertyInt(100,"Class1",102);
    TObjSetPropertyString(100,"Class2","Temp");
    TObjEndUpdate(100);
end
```

## 8.14 HistoricalView

With the HistoricalView component you can show the historical values of each gate recorded on DataBase.

Date	Time	N-1	N-2	N-3	N-4
00/00/0000	00.00.00	%10.2%	%10.2%	%10.2%	%10.2%

**Runtime** ↓

Date	Time	N-1	N-2	N-3	N-4
26/09/2013	17.27.15	37.00	43.00	4.00	92.00
26/09/2013	17.27.16	9.00	99.00	3.00	82.00
26/09/2013	17.27.17	87.00	39.00	40.00	20.00
26/09/2013	17.27.18	68.00	26.00	21.00	17.00
26/09/2013	17.27.19	10.00	97.00	95.00	3.00
26/09/2013	17.27.20	19.00	39.00	32.00	99.00
26/09/2013	17.27.21	86.00	84.00	72.00	0.00
26/09/2013	17.27.22	94.00	29.00	41.00	28.00
26/09/2013	17.27.23	99.00	23.00	94.00	54.00
26/09/2013	17.27.24	53.00	41.00	43.00	85.00
26/09/2013	17.27.25	54.00	6.00	97.00	30.00
26/09/2013	17.27.26	65.00	42.00	19.00	40.00
26/09/2013	17.27.27	34.00	49.00	10.00	73.00
26/09/2013	17.27.28	10.00	22.00	70.00	12.00
26/09/2013	17.27.29	13.00	20.00	31.00	75.00
26/09/2013	17.27.30	85.00	19.00	44.00	82.00
26/09/2013	17.27.31	97.00	52.00	79.00	73.00
26/09/2013	17.27.32	23.00	8.00	20.00	43.00

### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the HistoricalView (in pixels).

**Top:** vertical position of the top left corner of the HistoricalView (in pixels).

**Width:** width of the HistoricalView (in pixels).

**Height:** height of the HistoricalView (in pixels).

**Date :** width (in pixels) and the text of the "Date" column.

**Time:** width (in pixels) and the text of the "Time" column.

**Time range options:** allow to choose the timerange.

- Normal: will be shown alarms or events that have been generated in the interval time that starts at the time of opening of the template and that lasts for the specified TimeRange.
- External: the interval time is set externally by using the code (see the function



HisViewSetTimeRange)

- External start: the start of the interval time for alarms or events visualization is set from the code ( see the function HisViewSetTimeRange), and its length is given by the specified TimeRange
- External stop: the end of the interval time for alarms or events visualization is set from the code ( see the function HisViewSetTimeRange), and its length is given by the specified TimeRange

**Row options:** allow to choose how each row record will be displayed.

**Gates:** list of gates that must be displayed in the objects:can be selected Numeric,Digital,String and Compound gates. For each gate the following options must be specified:

- Format: it indicates how to show to value in the column.  
For NUMERIC or COMPOUND gates can be used the following format specifier:

“%x.ylf”, to display the number in decimal format,where:

- **x** is a number (optional), and indicates the number of digits to show. If it is not present, all digits from the value read from the gate will be shown. If it is preceded by a 0, some 0 before the number will be shown in order to reach the specified number of digits.
- **y** is a number (optional), and indicates the number decimal digits to show.

Some examples:

“%5.2lf” will produce 123.45

“%5.0lf” will produce 123

“%07.2lf” will produce 00123.45

“%nX”, to display the number in hexadecimal format,where:

- **n** is a number (optional), and indicates the number of digits to show. If it is not present, all digits from the value read from the gate will be shown. If it is preceded by a 0, some 0 before the number will be shown in order to reach the specified number of digits.

Some examples:

“%5X” will produce 1AB

“%05X” will produce 001AB

“%n.yb”, to display the number in binary format,where:

- **n** is a number that indicates the number of digits to show. It must be always present.
- **y** is a number (optional), and indicates how many digits compose a group.(between two groups a blank space will be shown)

Some examples:

“%16.4b” will produce 0100 0111 0011 1111

“%16.8b” will produce 01000111 00111111

For DIGITAL gates can be used the following format specifier:

“%d”, to display the number in decimal format.

For STRING gates can be used the following format specifier:

“%s”

- Width: it indicates the width of the displayed column (in pixels).

**Show server button:** enable/disable "SERVER" button to select from wich server computer alarms/events historical files must be loaded.

**Show Config button:** enable/disable "CONFIG" button to select the historical view time range.

**Show Print button:** enable/disable "PRINT" button to print the historical view list on the default printer

**Show Save CSV button:** enable/disable "Save CSV" button to export the historical view list on a formatted text file with TAB separator.

**Window Bkgrnd color:** window background color.

**Default server:** indicates from which server historical data must be read.

- "Local": historical data must be read from the local computer.
- "Channel\_x": historical data must be read from the server specified in channel configuration.

#### Commands (CodeBuilder)

There is the possibility to send some commands to Historical view object by using "**TObjFunction(int ObjId, int Function)**" CodeBuilder instruction.

Parameters:

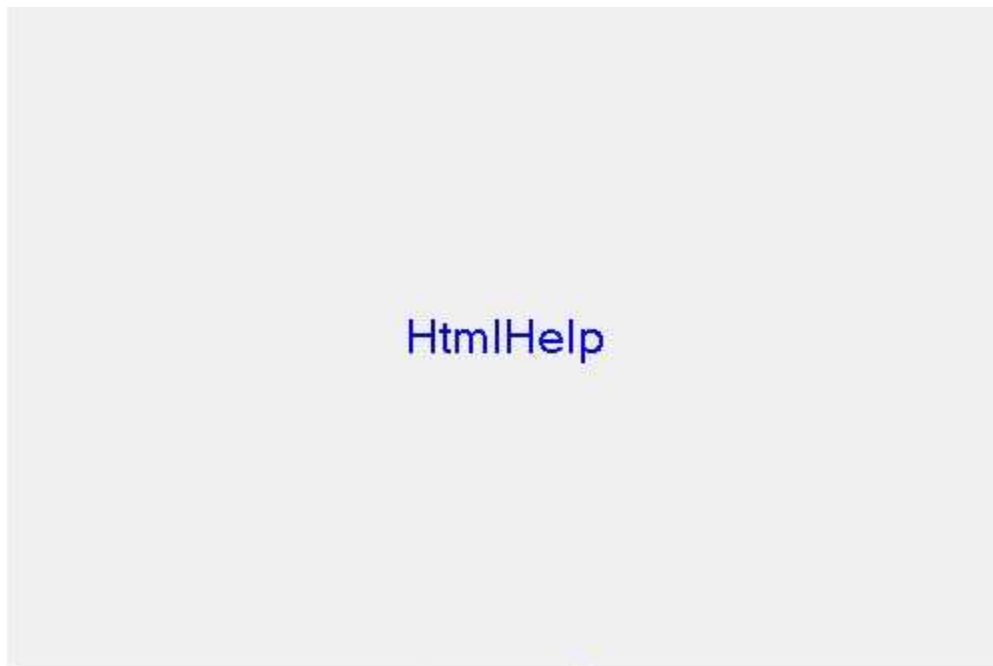
**int ObjId** : the number that identify the component (see ID property of Historical view object).

**int Function** : function to perform.

Function	Description
1	Show server selection window
2	Show configure time window
3	Save CSV file
4	Print
13	Stop loading historical files

## 8.15 HtmlHelp

With HtmlHelp component you can show a ".CHM" file in a template.



Runtime ↓



#### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the HtmlHelp window (in pixels).

**Top:** vertical position of the top left corner of the HtmlHelp window (in pixels).

**Width:** width of the HtmlWindow (in pixels).

**Height:** height of the HtmlWindow (in pixels).

**Path:** full path file name of the ".CHM" file to show into the component. The path can be specified either in absolute (eg. "c:\Data\Help.chm") or in relative way (eg. ".\Data\Help.chm"): using the relative way, the base directory will be the application's templates directory.

HtmlHelp object supports also multilanguage help: an example will clarify some points:

Suppose to have an application with two languages defined: English and Italiano.

Suppose to specify the Help file name in the Path item as this: "C:\Data\Help.CHM".

When the application starts, it starts with its default language (suppose "English"), so first of all will be searched an help file named "C:\Data\Help\_English.CHM"; if it is not found then the specified "C:\Data\Help.CHM" file will be loaded.

When user selects language "Italiano", first of all will be searched an help file named "C:\Data\Help\_Italiano.CHM"; also in this case if it is not found then the specified "C:\Data\Help.CHM" file will be loaded.

**Show back button:** enable/disable "BACK" button into the HtmlHelp window.

**Show forward button:** enable/disable "FORWARD" button into the HtmlHelp window.

**Show Print button:** enable/disable "PRINT" button to into the HtmlHelp window..

## 8.16 Label

The Labels are text labels that can be used to communicate useful information to the user. For example it is possible to use them as simple labels, or to write on them the value read from a gate.



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the Label (in pixels).

**Top:** vertical position of the top left corner of the Label (in pixels).

**Width:** width of the Label (in pixels).

**Height:** height of the Label (in pixels).

**Description:** description of the Label (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**BkColor:** background color of the Label. To specify the desired color, use the button on the properties row, and choose the color."clNone"color means label with transparent background.

**Cursor:** shape that the cursor of the mouse must have when it goes over the Label during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

**TxtColor:** color of the Label text. To specify the desired color, use the button on the properties row, and choose the color.

**Font:** font to use for the Label text. Press the button on the properties line to show the dialog for the choice of the font, size, style (normal, bold or italic) and effects (underlined and strikeout).

**Gate:** gate from which to read the value to write in the Label text. Press the button on the properties row of the *property editor* to show the list of the template gates. From it you can select the desired gate.

**Label:** text displayed by the Label. If in the label text you want to include the value read from the gate linked to the Label "**Gate**" property, you have to resort to the following conventions:

For Numeric, Digital and Compound gates:

- "**%x.ylf**" If you want to specify in detail how to display the number, where:
  - **x** is optional and indicates the total number of digits to display. If not present, will be shown all the digits of the value read from the gate, and if it is preceded by 0 then 0 will appear before the number to reach the number of digits specified.
  - **y** is optional and indicates the number of digits to display after the decimal point. If y is equal to "\*" then is used as the number of decimal places the number specified for the gate linked to the "**Gate**" field of the Label.

Some examples will make things clearer:

"%5.2lf" produce 123.45

"%5.0lf" produce 123

"%07.2lf" produce 00123.45

"%7.\*lf" produce 123.456 if the number of decimal digits specified for the linked gate (in Gate Builder) is 3.

For example, if you need to show a temperature read from a numeric gate, using a Label, it is sufficient to link the desired gate to the Label "**Gate**" property and set the Label "**Label**" property with the string "Temperature:% 5.1lf ° C". If during supervision the gate value will be 25.7, the label will display the string "Temperature: 25.7 ° C".

- "**%g**" Using this format, the number will be displayed in order to take up as little space as possible (possibly uses the exponential form).
- "**%d**" for integers.
- "**%xd**" for integers with a maximum number of characters: the meaning of the parameter x is identical to that seen above.

For String gates:

- "**%s**" strings.
- "**%xs**" for strings with maximum number of characters: the meaning of the parameter x is identical to that seen above.

**Horizontal align:** horizontal position where the label of the Label will be put. There are three options: Left, Center and Right.

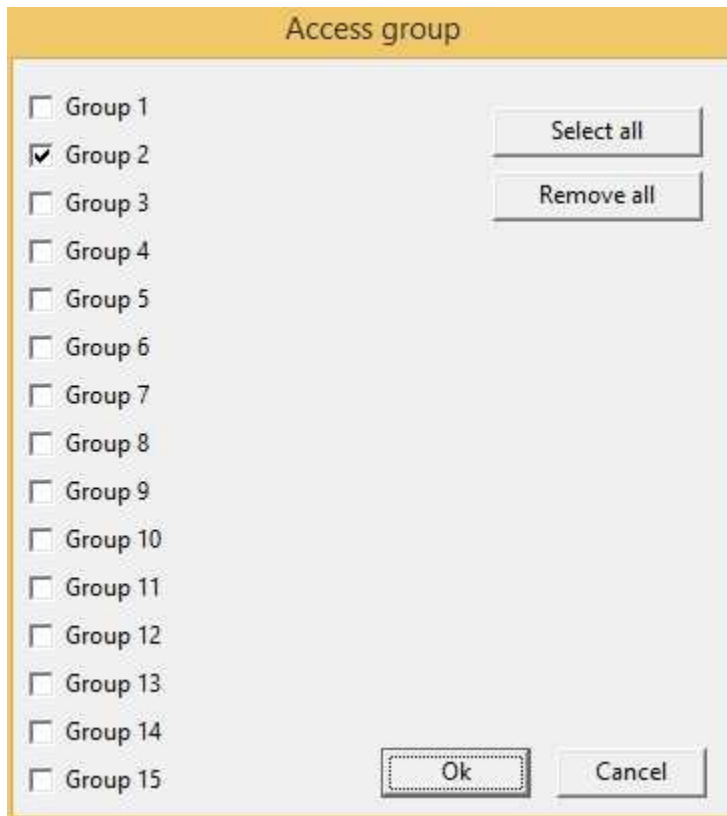
**Vertical align:** vertical position where the label of the Label will be put. There are three options: Top, Center and Bottom.

**Frame:** for each Label, it is possible to indicate the style of the frame enclosing it. There are the same options as for the Frame's (see table). *Template Builder* will always show the flat frame, while during the supervision phase the frame will be as specified in the property.

**Multiline :** specifies whether the text wraps when it is too long for the width of the object.

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

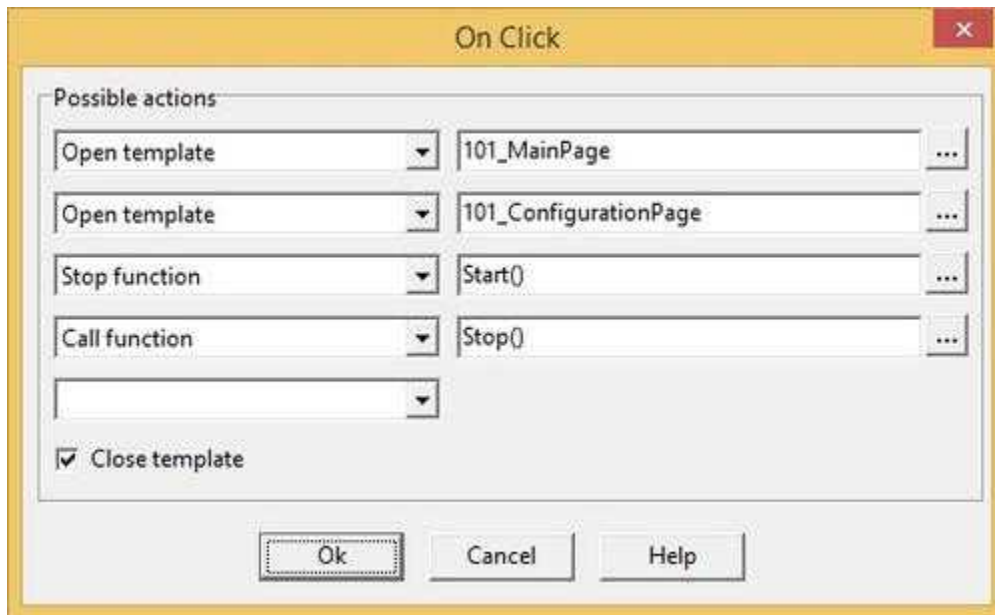
**Access group:** users groups enabled to operate on the object in runtime mode (enables "On Click" and "On Double Click" functions). By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled (enables "On Click" and "On Double Click" functions) if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

**On Click:** operation to carry out when the user click on the object. Next figure show the window through which to specify the list of the operations to carry out:

- Call function: call the indicated function.
- Stop function: stop the indicated function (if this is on).
- Open template: open the indicated template.
- Apply changes: apply the changes in all the object brothers or child components that have the "Need apply" property set to "Yes" (the new values will be written in the gates).
- Undo changes: undo the changes in all the object brothers or child components that have the "Need apply" property set to "Yes".
- Close template: close the current template.



**On Double Click:** operation to carry out when the user double click on the object. Refer to "On Click" property to specify the list of the operations to carry out.

## 8.17 Led

The Led is a component that allows to display graphically the status of some conditions on the template gates. In fact, with every Led is associated a list of conditions: if one of them takes place, the Led is in ON status, if it doesn't, the Led is OFF. In each status the Led will show an image: it is possible to indicate the file with the desired image, or select it from the default ones.



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the Led (in pixels).

**Top:** vertical position of the top left corner of the Led (in pixels).

**Width:** width of the Led (in pixels).

**Height:** height of the Led (in pixels).

**Description:** description of the Led (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Transparent Color:** color that will be made transparent. The background behind the Image will appear where the image has this color.

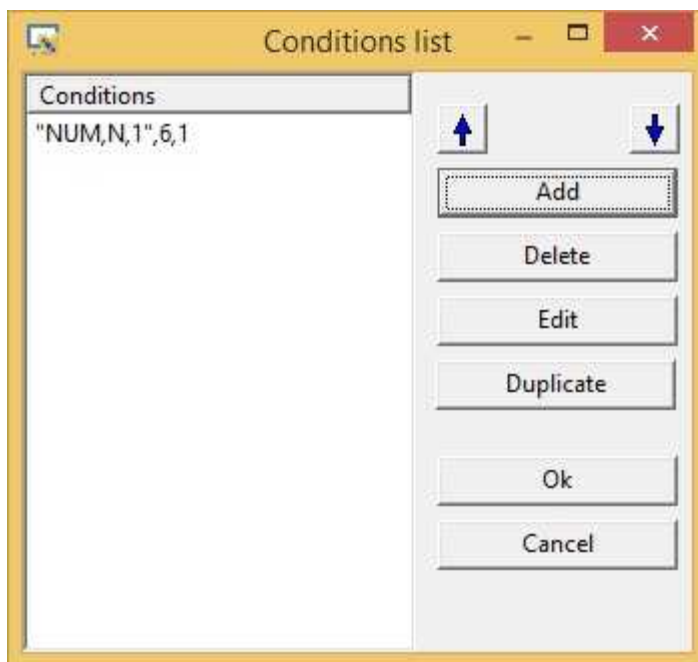
**Cursor:** shape that the cursor of the mouse must have when it goes over the Led during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table ).

**OFF appearance:** appearance of the Led when it is OFF. To select the desired image, use a dialog box like the one of the switch. In it you can choose from the default images (shown in the table below) or indicate a file from which to read the image.

Resource	Led	Resource	Led
01	 (red)	02	 (red)
11	 (green)	12	 (green)
21	 (blue)	22	 (blue)
31	 (gray)	32	 (gray)
41	 (yellow)	42	 (yellow)

**ON appearance:** appearance of the Led when it is ON. To select the desired image, use a dialog box like the one of the Switch. In it you can choose from the default images (shown in the table above) or indicate a file from which to read the image.

**Led ON conditions:** list of conditions prompting the Led to be ON. Press the button on the properties row to show the window in figure below. As usual, it is possible to add, remove or modify the conditions in the list: when you add or modify a condition a window like *StateCondition* in the Bitmap section. In it you can indicate the gate and the condition prompting the Led to be ON (for more details on how to specify the desired conditions, see what has been said about the Bitmap states conditions).

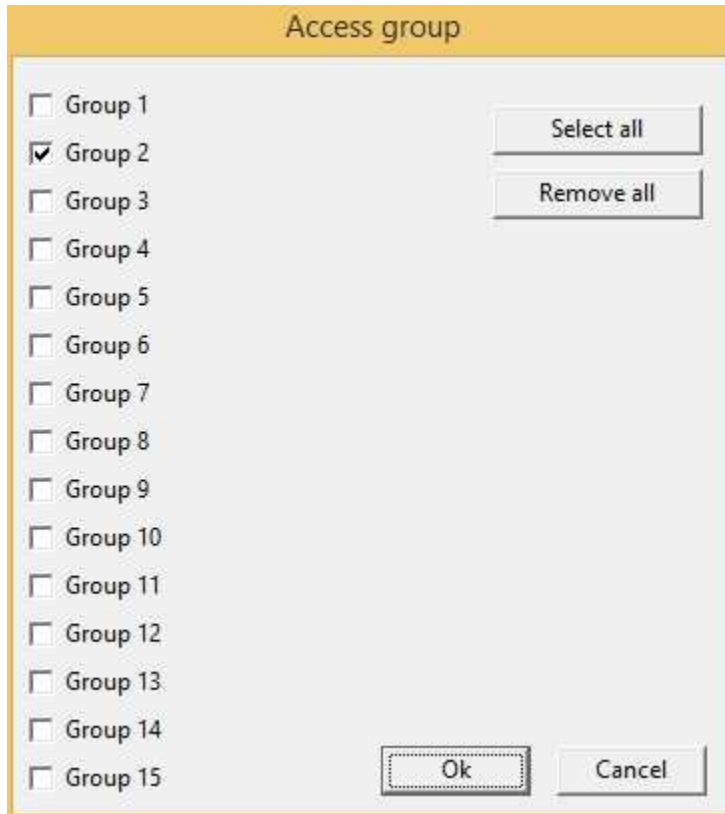


**Flashing:** if it is enabled, the led will flash when it is in the ON status.

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.



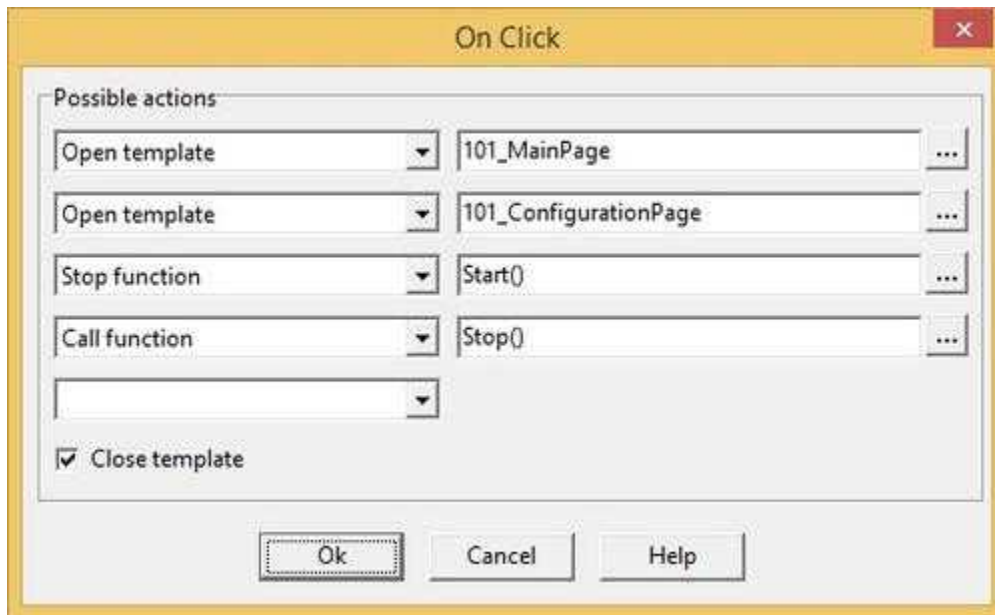
**Access group:** users groups enabled to operate on the object in runtime mode (enables "On Click" and "On Double Click" functions). By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled (enables "On Click" and "On Double Click" functions) if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

**On Click:** operation to carry out when the user click on the object. Next figure show the window through which to specify the list of the operations to carry out:

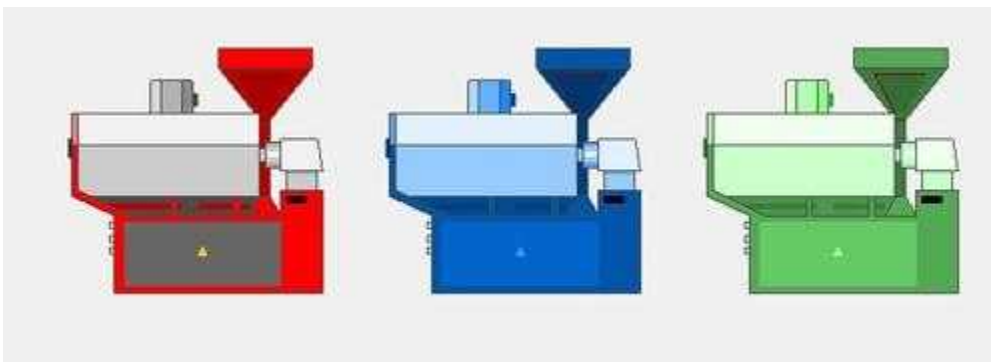
- Call function: call the indicated function.
- Stop function: stop the indicated function (if this is on).
- Open template: open the indicated template.
- Apply changes: apply the changes in all the object brothers or child components that have the "Need apply" property set to "Yes" (the new values will be written in the gates).
- Undo changes: undo the changes in all the object brothers or child components that have the "Need apply" property set to "Yes".
- Close template: close the current template.



**On Double Click:** operation to carry out when the user double click on the object. Refer to "On Click" property to specify the list of the operations to carry out.

## 8.18 Metafile

The MetaFile Object is a component very similar to the Bitmap Object. Like the latter, in fact, it can be used for displaying dynamic images, changing according to the conditions present in the plant. In this case, however, the images displayed are not simple bitmaps, but they are metafiles. As with the Bitmap, also MetaFile can have child objects.



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the MetaFile (in pixels).

**Top:** vertical position of the top left corner of the MetaFile (in pixels).

**Width:** width of the MetaFile (in pixels).

**Height:** height of the MetaFile (in pixels).

**Description:** description of the MetaFile (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Cursor:** it indicates what shape the cursor of the mouse must have when it goes over the MetaFile during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

**MetaFile states:** images and the conditions of the MetaFile. The management of the states of the MetaFile is like the Bitmap one; the only difference is that the images have to be selected from the available metafiles (the software operates with *Enhanced MetaFile* images).

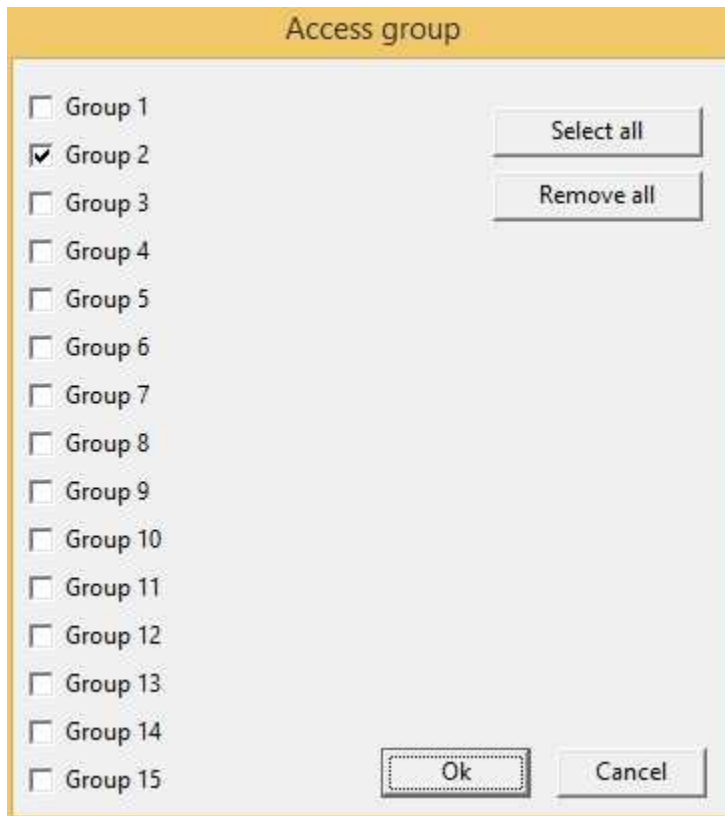
**Tab num:** number that indicates the order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

**X Animation:** gate that must be used to read the X coordinate (in pixels) of the MetaFile. Using this property it is possible to make the MetaFile moving across the template using the position given by the specified gate. The value 0 corresponds to the left edge of the template.

**Y Animation:** gate that must be used to read the Y coordinate (in pixels) of the MetaFile. Using this property it is possible to make the MetaFile moving across the template using the position given by the specified gate. The value 0 corresponds to the top edge of the template.

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

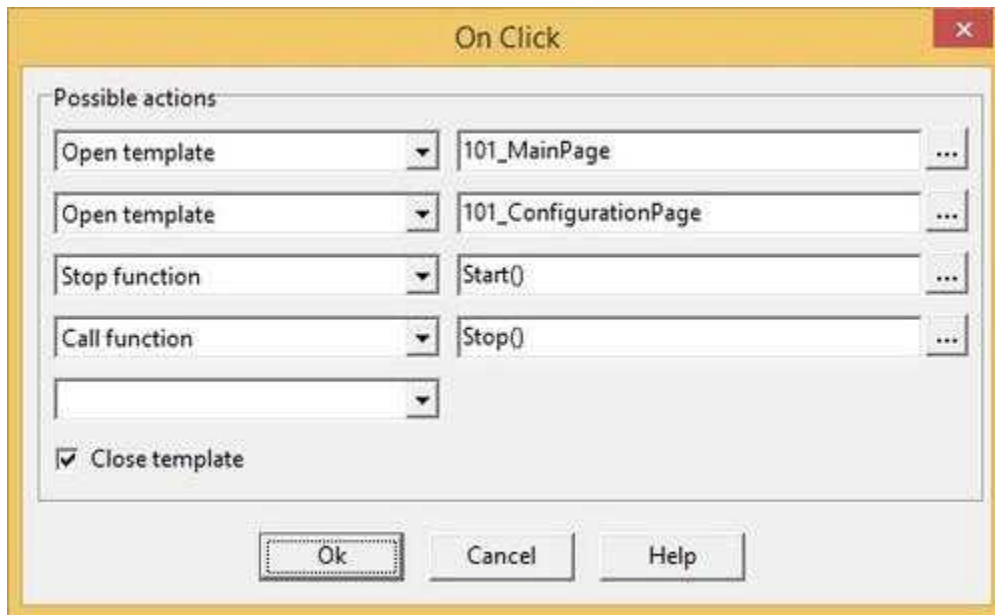
**Access group:** users groups enabled to operate on the object in runtime mode (enables "On Click" and "On Double Click" functions). By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled (enables "On Click" and "On Double Click" functions) if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

**On Click:** operation to carry out when the user click on the object. Next figure show the window through which to specify the list of the operations to carry out:

- Call function: call the indicated function.
- Stop function: stop the indicated function (if this is on).
- Open template: open the indicated template.
- Apply changes: apply the changes in all the object brothers or child components that have the "Need apply" property set to "Yes" (the new values will be written in the gates).
- Undo changes: undo the changes in all the object brothers or child components that have the "Need apply" property set to "Yes".
- Close template: close the current template.



**On Double Click:** operation to carry out when the user double click on the object. Refer to "On Click" property to specify the list of the operations to carry out.

## 8.19 RadioButton

The RadioButton is a component very similar to the CheckBox: the only difference is its behaviour during the supervision phase. Unlike for the CheckBox, in fact, during the supervision the user can select only one RadioButton among a set of brothers (that is to say children of the same component, for example GroupBox containing a set of RadioButtons). This way when you select a RadioButton, the one previously selected is unselected, so as to allow the user of the supervision page to make a choice between several alternatives (the various RadioButtons) that exclude one another. For the RadioButton too, *RunTime* checks during the supervision if the value read from the gate associated with the RadioButton verifies the ON condition: if so, the RadioButton is selected, if not it is unselected.



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left :** horizontal position of the top left corner of the RadioButton (in pixels).

**Top:** vertical position of the top left corner of the RadioButton (in pixels).

**Width:** width of the RadioButton (in pixels).

**Height:** height of the RadioButton (in pixels).

**Description:** description of the RadioButton (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**BkColor:** background color of the RadioButton. To specify the desired color, use the button on the

properties row, and choose the color.

**Font:** font to use for the text. Press the button on the properties line to show the dialog for the choice of the font, size, style (normal, bold or italic) and effects (underlined and strikethrough).

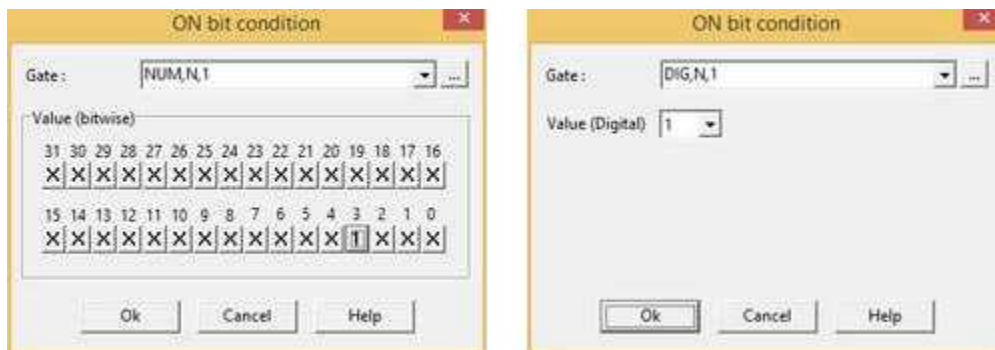
**Cursor:** shape that the cursor of the mouse must have when it goes over the RadioButton during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

**Label:** label of the RadioButton.

**Off label:** label of the RadioButton when this is not selected. If the off label is missing, when the RadioButton is not selected the normal label will be shown.

**Need apply:** it indicates whether the status change of the RadioButton will have to be confirmed from outside with a Button (Yes), or will be applied immediately (No).

**ON condition:** gate to be written on following the status change of the RadioButton, and the value to write. Press the button on the properties row to show one of the windows in figure below. If a numeric gate have been selected, then will be possible to set the ON value by selecting which bits must be set to 1, which will be set to 0 and which must not be considered. If a digital gate have been selected then must be specified the ON value of a single bit. In both cases, when the RadioButton is set to OFF, the denied ON value will be written.



**Style:** style of the RadioButton. There are three available styles:

•Standard:



•Push like:



•Right button:

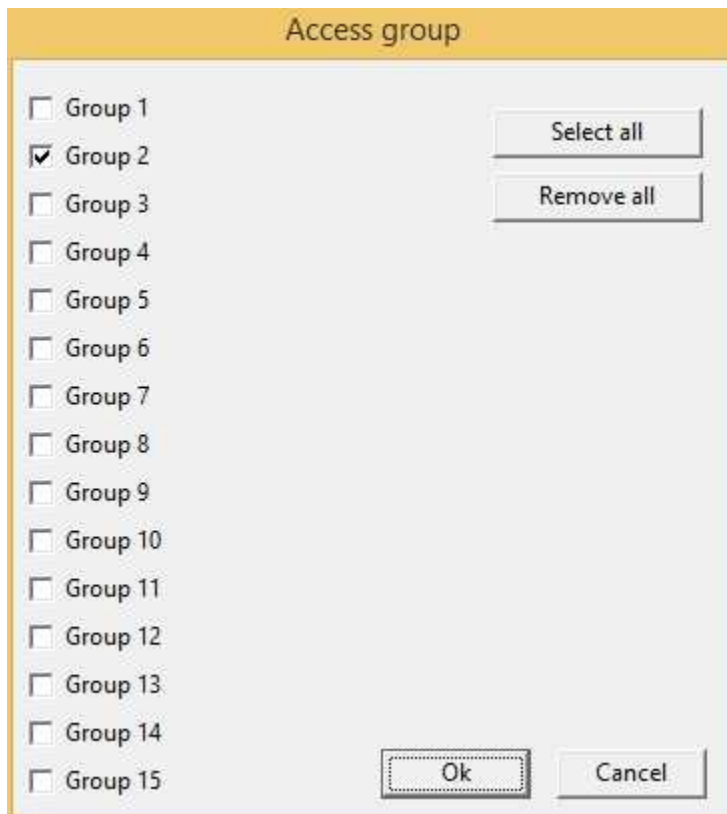


**Tab num:** number that indicates the order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

**Help file name** : if in this field is specified a valid ".CHM" file name and it is present an HtmlHelp object in one of the templates currently showed on the screen, when the RadioButton is clicked or receive the focus, the .CHM file will be showed in the HtmlHelp window. The path can be specified either in absolute (eg. "c:\Data\Help.chm") or in relative way (eg. "..\Data\Help.chm"): using the relative way, the base directory will be the application's templates directory. See HtmlHelp object for more details.

**Show**: object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group**: users groups enabled to operate on the object in runtime mode. By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable**: object is enabled if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

## 8.20 StatusBar

The StatusBar object is, in some ways, very similar to the Bitmap object: like this one, in fact, it has different states that are shown if, during the supervision, takes place the condition associated with the status. In the StatusBars every single status is represented by a text, that is shown in the component if the status becomes active. The StatusBar allows to inform the user of the supervision page (thanks to text messages) about the taking place of some conditions, for which the StatusBar has been designed.



**Properties (TemplateBuilder)**

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the StatusBar (in pixels).

**Top:** vertical position of the top left corner of the StatusBar (in pixels).

**Width:** width of the StatusBar (in pixels).

**Height:** height of the StatusBar (in pixels).

**Description:** description of the StatusBar (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Cursor:** shape that the cursor of the mouse must have when it goes over the StatusBar during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

**BkColor:** background color of the StatusBar when no status is active. To specify the desired color, use the button on the properties row, and choose the color.

**Font:** font to use in the StatusBar text. Press the button on the properties line to show the dialog box for the choice of font, size, style (normal, bold or italic) and effects (underlined and strikeout).

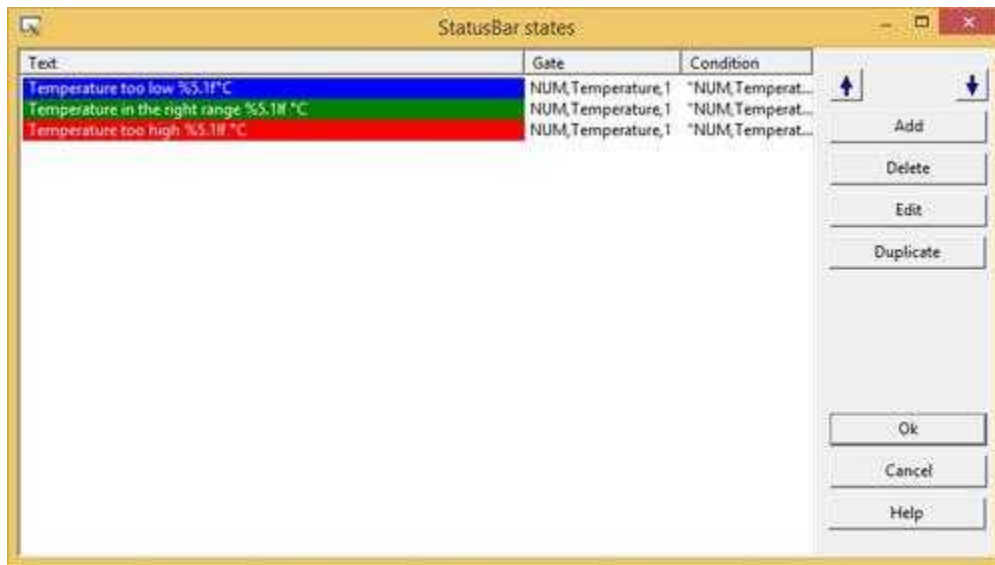
**Horizontal align:** horizontal position of the text inside the StatusBar. There are three options: Left, Center and Right.

**Vertical align:** vertical position of the text inside the StatusBar. There are three options: Top, Middle and Bottom.

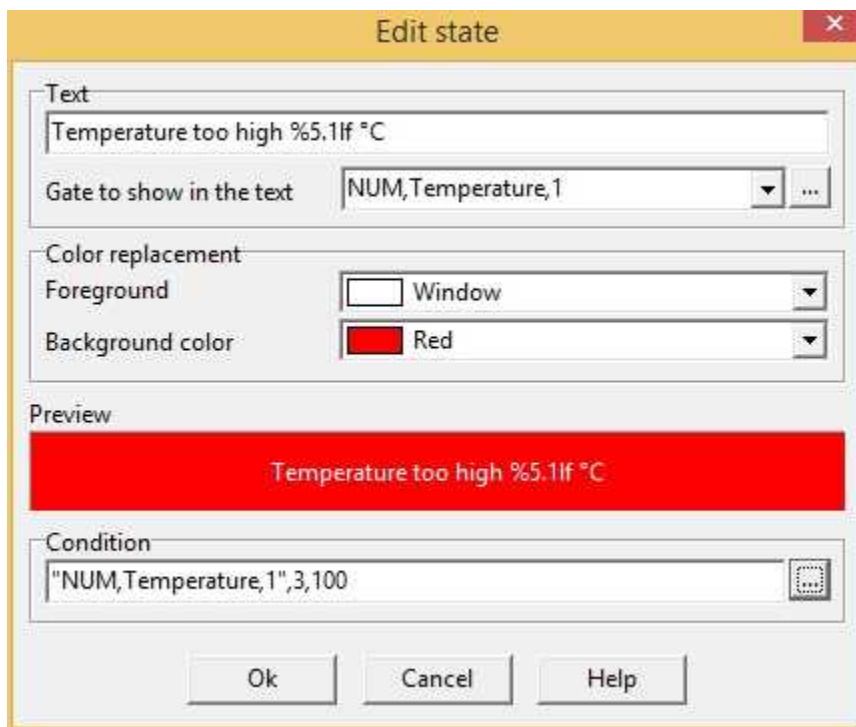
**Frame:** type of frame enclosing the StatusBar. There are the same options as for the Frame's (see table of Frame section): Flat, Relief (thin), Bas-relief (thin), Raised Frame, Recessed Frame, Relief (medium), Bas-relief (medium).

**States:** list of states of the StatusBar. To define a status use the window in figure below. In it there is a list of states present: with buttons on the right you can add, modify, remove and sort the states.





When you add or modify a state, the window in figure below will be shown.



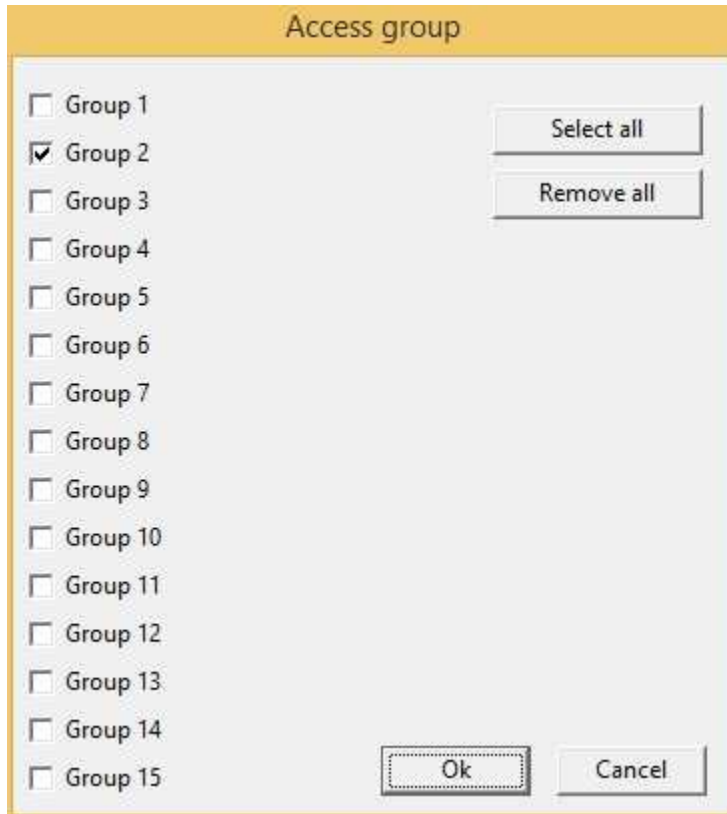
In this window it is possible to specify:

- *Text* : text to show when the state became active (in the text there can't be any commas).
- *Foreground color* : color of the text to show.
- *Background color* : background color of the text to show
- *Gate to show in the text* : if a gate is specified then the text row will be built combining the *Text* field with the values of the specified gate. Refer to *Label* field of the Label object for more information.
- *Condition* : condition that must be verified to show the *Text*.

**Multiline** :specifies whether the text wraps when it is too long for the width of the object.

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

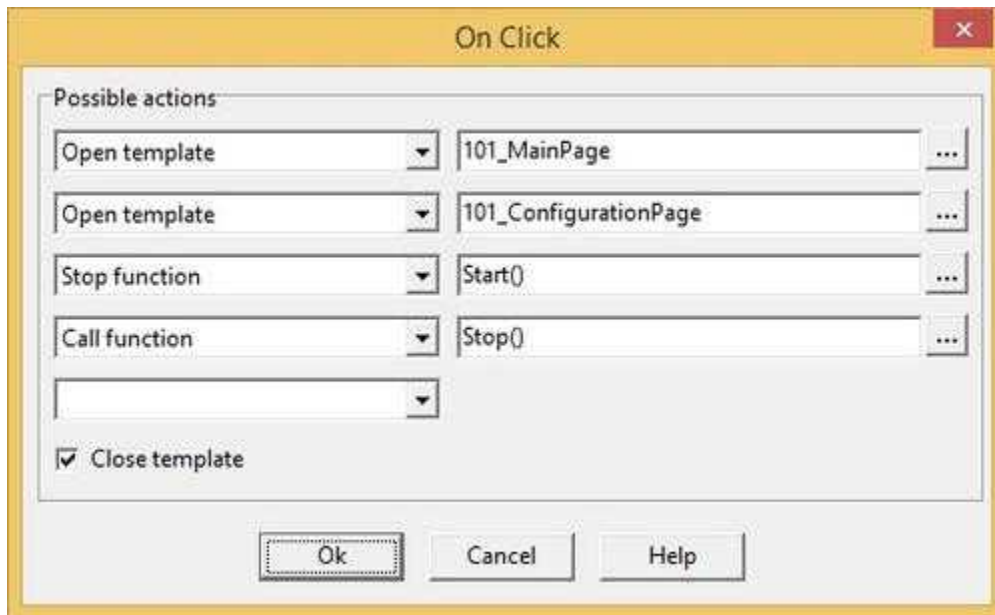
**Access group:** users groups enabled to operate on the object in runtime mode (enables "On Click" and "On Double Click" functions). By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled (enables "On Click" and "On Double Click" functions) if it is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

**On Click:** operation to carry out when the user click on the object. Next figure show the window through which to specify the list of the operations to carry out:

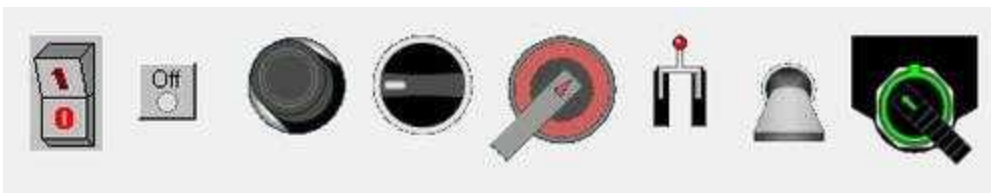
- Call function: call the indicated function.
- Stop function: stop the indicated function (if this is on).
- Open template: open the indicated template.
- Apply changes: apply the changes in all the object brothers or child components that have the "Need apply" property set to "Yes" (the new values will be written in the gates).
- Undo changes: undo the changes in all the object brothers or child components that have the "Need apply" property set to "Yes".
- Close template: close the current template.



**On Double Click:** operation to carry out when the user double click on the object. Refer to "On Click" property to specify the list of the operations to carry out.

## 8.21 Switch

The Switch object is a component that acts as a switch. Therefore its status can be either ON or OFF. During the supervision phase the user can commute the Switch status by clicking on it. The Switch can be programmed to react to the status change modifying the value of a gate in the desired way. *RunTime* checks if the value of the gate on which the Switch writes, changes during the supervision: if the new value verifies the ON condition the Switch commutes to ON, otherwise the Switch commutes to OFF. This way the user can see by himself the change in the situation of the gate the Switch acts on.



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the Switch (in pixels).

**Top:** vertical position of the top left corner of the Switch (in pixels).

**Width:** width of the Switch (in pixels).

**Height:** height of the Switch (in pixels).

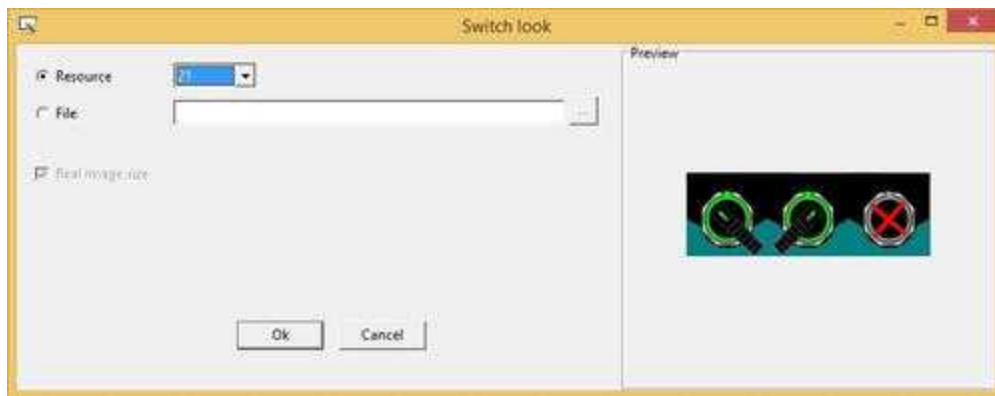
**Description:** description of the Switch (maximum 150 characters). The description will be shown when

you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Transparent Color:** color that will be made transparent. The background behind the Image will appear where the image has this color.

**Cursor:** shape that the cursor of the mouse must have when it goes over the Switch during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click. (for details see the cursor table)

**Image:** appearance of the Switch. Click on the button on the properties row to show the window in figure.



The image can be of type :

- *Resource*
- *File*

*Resource* image type can be selected from a list of predefined images (as shown in the following example):

Resource	Shape
01	
02	
03	

File image can be chosen by pressing the [...] button near the *File* field.

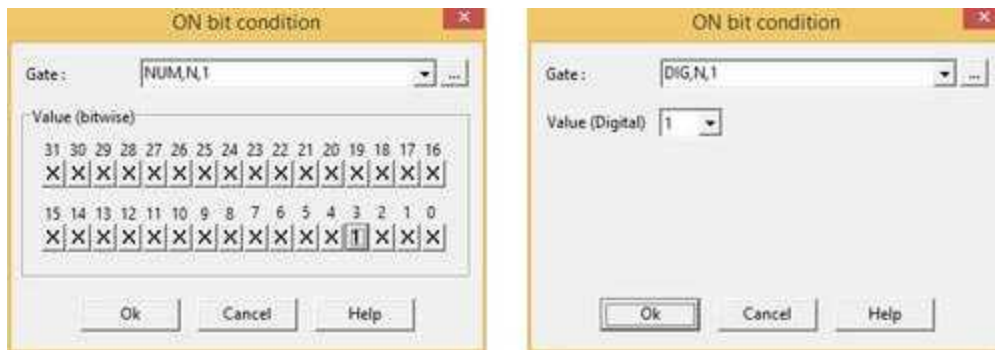
Images for the Switches must be compound of three states placed side by side horizontally: the OFF, the ON and the disabled status.

**Needs apply:** it indicates whether the status change of the Switch will have to be confirmed from outside with a Button (Yes), or will be applied immediately (No).

**Mode:** if switch mode is selected then the object changes its status on a mouse click. If Button mode is selected then the object changes its status to the ON status on left mouse button down and automatically restore the OFF status on left mouse button up.

**ON condition:** gate to be written on following the status change of the Switch, and the value to write. Press the button on the properties row to show one of the windows in figure below. If a numeric gate have been selected, then will be possible to set the ON value by selecting which bits must be set to 1, which will be set to 0 and which must not be considered. If a digital gate have been selected then must

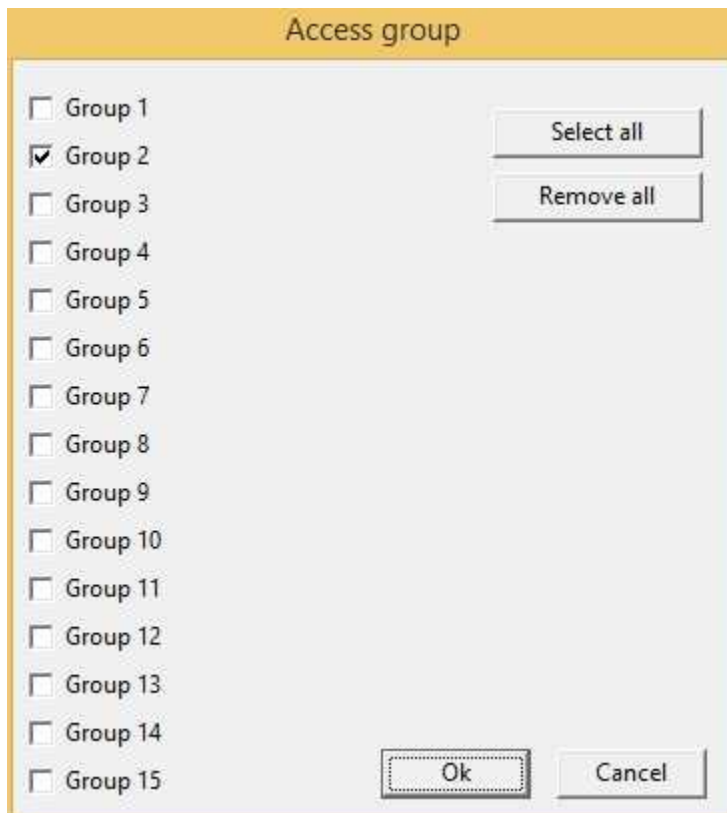
be specified the ON value of a single bit. In both cases, when the switch is set to OFF, the denied ON value will be written.



**Help file name** : if in this field is specified a valid ".CHM" file name and it is present an HtmlHelp object in one of the templates currently showed on the screen, when the Switch is clicked the .CHM file will be showed in the HtmlHelp window. The path can be specified either in absolute (eg. "c:\Data\Help.chm") or in relative way (eg. "..\Data\Help.chm"): using the relative way, the base directory will be the application's templates directory. See HtmlHelp object for more details.

**Show**: object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group**: users groups enabled to operate on the object in runtime mode. By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

## 8.22 TabSheet

With the TabSheet it is possible to organise the template more clearly (or part of it) spreading it on more pages. This way you can so as to group in each page the controls referring to a common function. To move from one page to the other (during both the supervision and design phase of the template) you just need to click with the mouse on the title of the page displayed in the upper part of the TabSheet (as shown in figure).



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

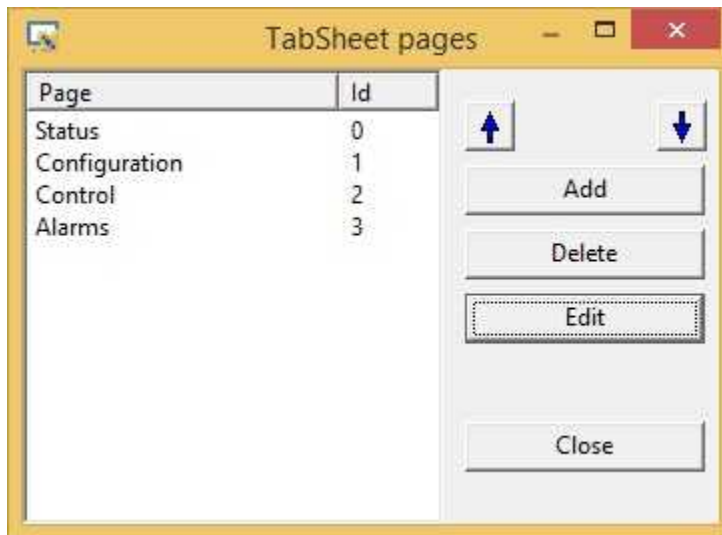
**Left:** horizontal position of the top left corner of the TabSheet (in pixels).

**Top:** vertical position of the top left corner of the TabSheet (in pixels).

**Width:** width of the TabSheet (in pixels).

**Height:** height of the TabSheet (in pixels).

**Pages:** set of pages that compound the TabSheet . The window in figure below shows the list of these pages: you can add, modify, remove and sort the pages using the buttons on the right. Note that when you sort the pages, along with them you move also the components on them. This way the components referring to a certain function will remain on the correct page.a.



**Tab num:** number that indicates the order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

#### Commands (CodeBuilder)

Following properties can be set also from language (see Code Builder help)

Property	Function
Current page	TObjSetPropertyInt(Id,"CurrentPage",...)

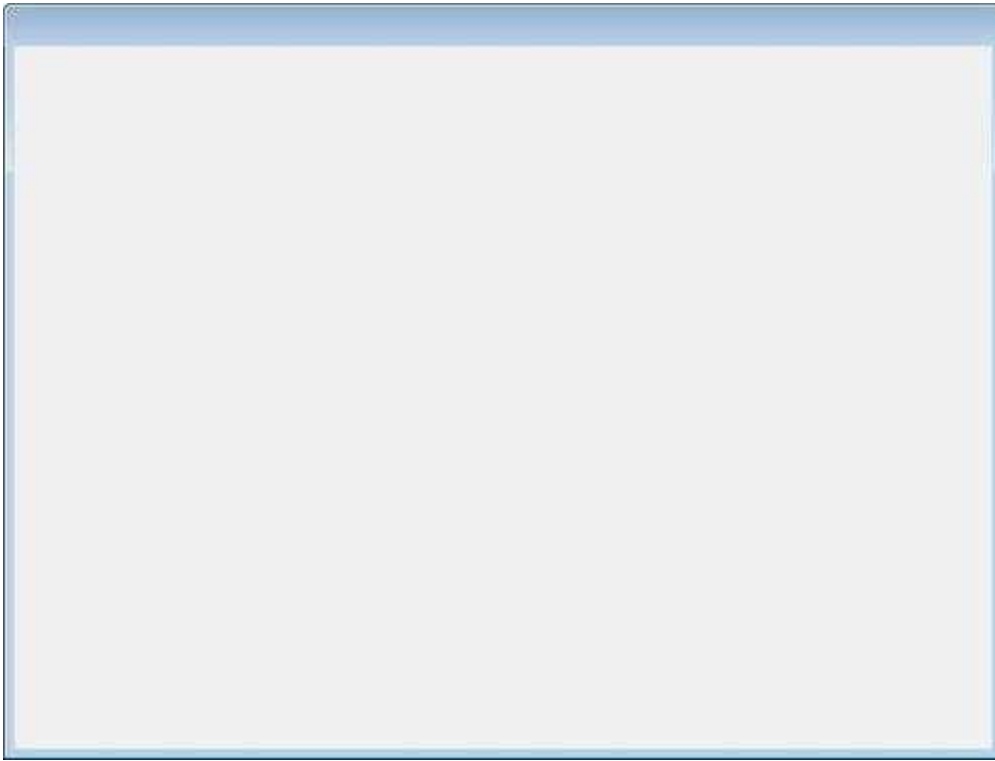
Before setting all required properties, it is necessary to call **TObjBeginUpdate(Id)** function. After setting all required properties, it is necessary to call **TObjEndUpdate(Id)** function to update the object.

*Example:*

```
...
TObjBeginUpdate(100);
TObjSetPropertyInt(100,"CurrentPage",3);
TObjEndUpdate(100);
```

## 8.23 Template

The Template is a component that is managed differently from the others: since it represents the supervision page, it is always present while working with *Template Builder*, and for this reason it is not on the tool bar. To show the Template properties in the *property editor*, you just need to click on the template window.



#### Properties (TemplateBuilder)

**Name:** name of the Template. It is the name standing also as a title of the supervision page. If it's missing, the title will be the name of the template file.

**Access group:** users groups allowed to open the Template. By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.





**Left:** horizontal position of the top left corner of the Template inside the *RunTime* window (in pixels).

**Top:** vertical position of the top left corner of the Template inside the *RunTime* window (in pixels).

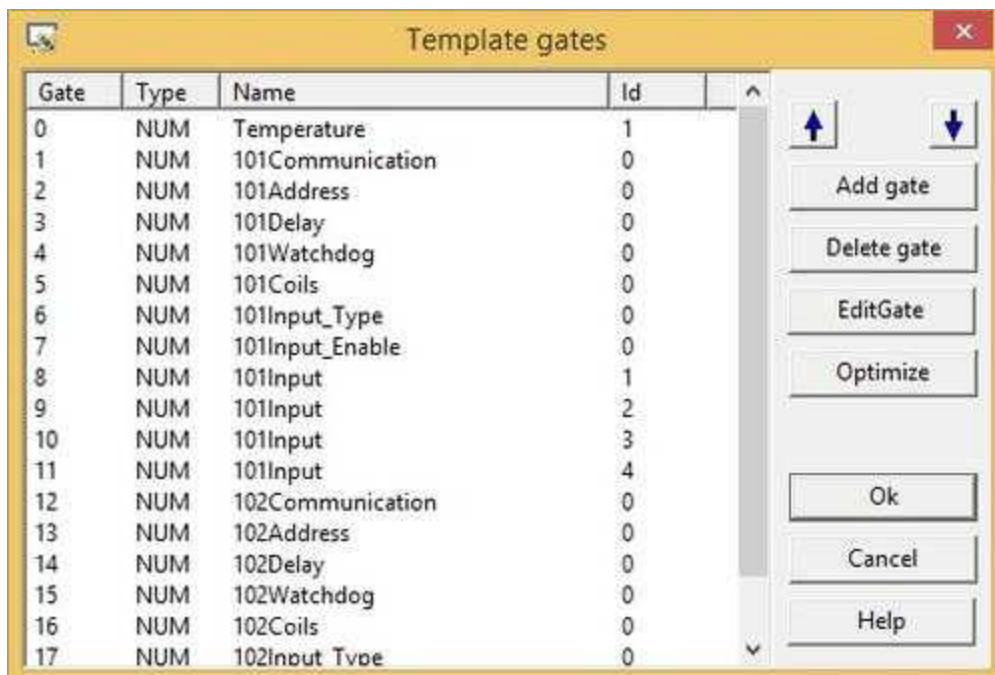
**Width:** width (in pixels) of the Template window.

**Height:** height (in pixels) of the Template window.

**BkColor:** background Color of the Template window. To specify the Color you have to click on the button in the corresponding row of the *property editor*: it will be opened the dialog box for the choice of the Color.



**Gates:** list of the gates used in the Template. Press the button on the row, to show the window in figure below. Through it you can modify the list. Press the button *Add gate* to show dialog box in which you have to indicate the type of gate (Numerical, Digital, String, Composed or Event), the name and the Id of the gate to add; pressing then *Ok* the new gate will be added in the list. The button *Remove gate* allows to remove, from the list of the gates used in the template, the one presently selected. With the button *Modify gate* you can modify the selected gate. To sort the gates list, there are two buttons with an up and a down arrow. With these buttons it is possible to move up or down the presently selected gate, so as to obtain the desired sorting of the list. *Optimize* button eliminates from template all the gates that have no objects associated with them and optimizes the gates definition order to speed up template loading in runtime mode.



**Open function:** possible function called up when the template is displayed from *RunTime*.

**Close function:** possible function called up when the template is closed from *RunTime*.

**Hidden:** indicates whether the template will appear in the *RunTime* menu. There are two possible choices:

- Yes: the template won't appear in the *RunTime Supervision | Template* menu.
- No: the template will appear in the *RunTime Supervision | Template* menu.

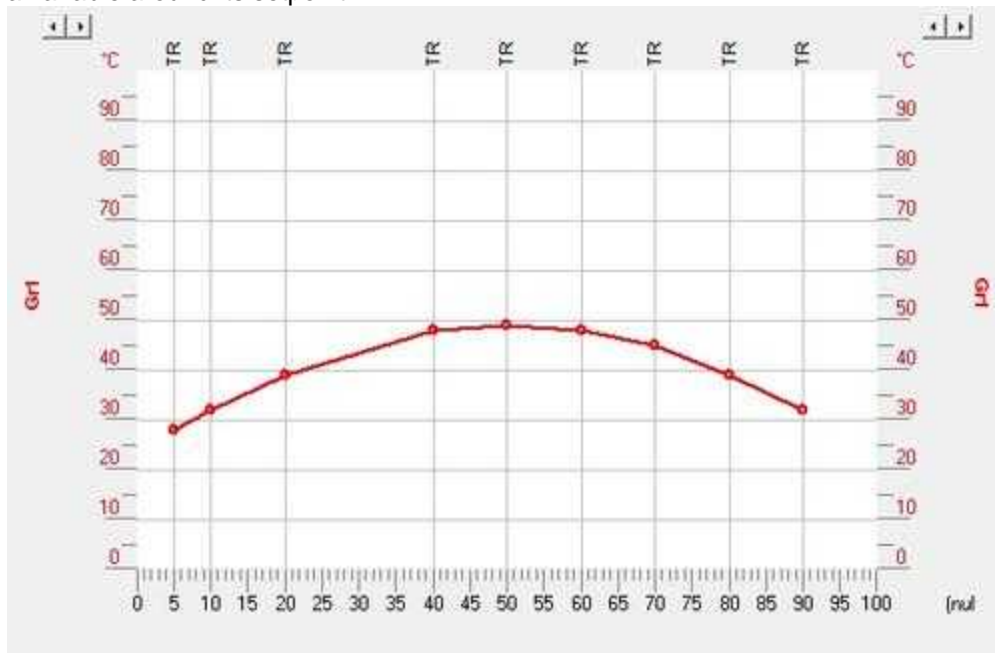
**Style:** indicates the style in which will be displayed the Template in *RunTime*. There are three available styles:

- Standard: the Template will be displayed as a normal window
- Fixed: the Template will be displayed as a window without borders, and so it won't be possible to move it inside the *RunTime* window.
- Maximized: it is similar to the fixed style, but in this case the Template will be placed on the top left corner of the *RunTime* window.

## 8.24 ThermMap

The ThermMap component can be used to show in *RunTime* phase a graph that reassumes the trend of a set of gates at the same moment. This component is the right choice to monitor the trend of a variable (e.g. a temperature) in a plant.

It is possible to show up to 9 trends at the same moment. Every trend consists of a broken line, which endpoint corresponds to the values read from the gates. For each trend it is possible to specify whether the line should appear as a thin line or as a thick line: in this case the thickness is given by a gate (the thickness gate). This case is particularly useful when is needed to draw the allowed region of a variable around its setpoint.



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the HistView (in pixels).

**Top:** vertical position of the top left corner of the HistView (in pixels).

**Width:** width of the HistView (in pixels).

**Height:** height of the HistView (in pixels).

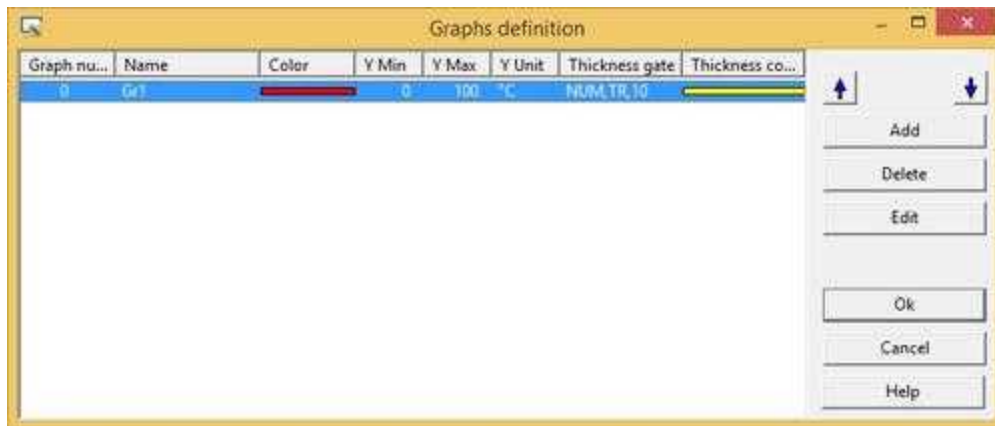
**Description:** description of the StatusBar (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**X Min:** minimum value to show on the X axis of the graph.

**X Max:** maximum value to show on the X axis of the graph.

**X Unit:** measurement unit of the X axis.

**Graphs:** set of charts that will be shown into the *ThermMap*. By pressing the button, it is possible to specify the list of charts using the dialog window that will be shown.



The window below must be used to select the graph's parameters: the name of the graph, its color, its range (Y min and Y Max), the measurement unit of the Y axis and the set of points that composes the graph. For each point you must indicate its position (in the measurement unit of X axis), and the gate where read the value.

It is also possible to enable drawing of a thickness around the graph: in this case must be specified a numeric gate that gives the thickness width (in pixel).

Graph name: Gr1

Graph color: [Red]

Y axis:

Y Min: 0 Y Max: 100

Y Unit: °C

Use thickness

Thickness gate: NUM,TR,10

Thickness color: [Yellow]

X Position	Gate
5	NUM,TR,1
10	NUM,TR,2
20	NUM,TR,3
40	NUM,TR,4
50	NUM,TR,5
60	NUM,TR,6
70	NUM,TR,7
80	NUM,TR,8
90	NUM,TR,9

Buttons: Add, Delete, Edit, Ok, Cancel, Help

**Enable grid:** indicates whether or not to draw the grid of the chart

**Enable points:** indicates whether or not to draw the points that constitutes the chart

**Line width:** thickness of the line that draws the chart

**Interpolation:** interpolation to use to draw the graph

**Points name:** indicates if Gate Name Or Gate Description must be used as Points Name Label on the ThermMap object.

**Window Bkgrnd color:** ThermMap main window background color.

**Graphic Bkgrnd color:** chart window background color.

**Grid color:** grid color.

**Scale color:** axis color.

## 8.25 UpDown

The UpDown component is used to increase or decrease the value of a gate of an amount previously decided. The UpDown also checks the gate value: it is in fact possible to specify the minimum and maximum values the gate can have.



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the UpDown (in pixels).

**Top:** vertical position of the top left corner of the UpDown (in pixels).

**Width:** width of the UpDown (in pixels).

**Height:** height of the UpDown (in pixels).

**Description:** description of the UpDown (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Cursor:** shape that the cursor of the mouse must have when it goes over the UpDown during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

**Gate:** gate to be increased or decreased. Press the button on the properties row of the *property editor* to show the list of the template gates. From it you can select the desired gate.

**Minimum value:** minimum value the gate can have.

**Maximum value:** maximum value the gate can have.

**Step:** value that will be added or subtracted at every increase or decrease.

**Direction:** appearance of the component. There are two options, shown in the table below.

Direction	Appearance
Horizontal	
Vertical	

**Tab num:** number that indicates the order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

**Help file name :** if in this field is specified a valid ".CHM" file name and it is present an HtmlHelp object in one of the templates currently showed on the screen, when the UpDown is clicked or receive the focus, the .CHM file will be showed in the HtmlHelp window. The path can be specified either in absolute (eg. "c:\Data\Help.chm") or in relative way (eg. "..\Data\Help.chm"): using the relative way, the base directory will be the application's templates directory. See HtmlHelp object for more details.

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group:** users groups enabled to operate on the object in runtime mode. By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled

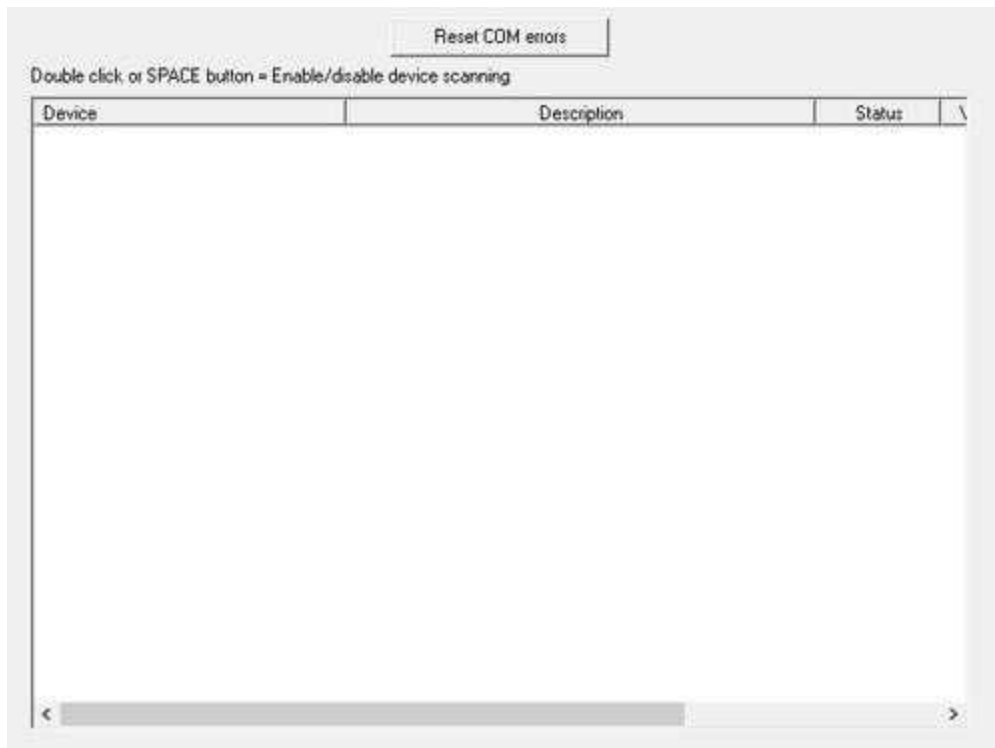
groups.



**Enable:** object is enabled if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

## 8.26 DevView

With the DevView component you can show the status of the devices. For more details refer to the paragraph Devices Status in the *RunTime* chapter.



#### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the DevView (in pixels).

**Top:** vertical position of the top left corner of the DevView (in pixels).

**Width:** width of the DevView (in pixels).

**Height:** height of the DevView (in pixels).

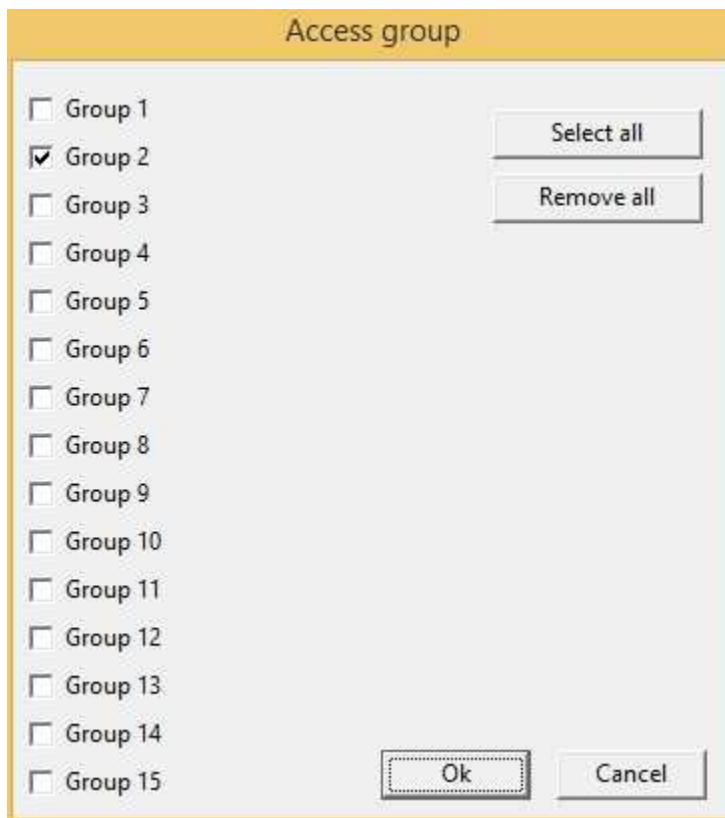
**Show Reset button:** if it is set to "Yes", will be shown the button that allow to reset the communications errors with devices.

**Enable/Disable Devices:** if it is set to "Yes", will be allowed to the user to enable/disable the communication with each single device by double clicking the mouse button or pressing SPACE key.

**Window Bkgrnd color:** window background color.

**Access group:** users groups enabled to operate on the object in runtime mode (Enable/Disable devices communication and reset communication errors). By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.





**Enable:** object is enabled if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

## 8.27 OperatorView

With the OperatorView component you can show the historical file of the operator changes. For more details refer to the paragraph Operator Interventions in the *RunTime* chapter.

The screenshot illustrates the transition from a configuration state to a runtime state. The top part shows a table with placeholder data, and the bottom part shows a table with actual runtime logs. A green arrow labeled "Runtime" points from the top to the bottom.

**Configuration State:**

Code	User	Date	Time	Message
XXXXXXXXXX	XXXXXXXXXX	00/00/0000	00.00.00	XXXXXXXXXX

**Runtime State:**

Code	User	Date	Time	Message
Start		26/09/2013	18.32.52	Start supervision session
Password	SuperUser	26/09/2013	18.33.02	User access
Recipe	SuperUser	26/09/2013	18.33.11	New: New recipe
Recipe	SuperUser	26/09/2013	18.33.24	Renamed: SetPoints
Recipe	SuperUser	26/09/2013	18.33.30	Imported: 26/09/2013
Recipe	SuperUser	26/09/2013	18.33.34	Modified: SetPoints
Num Gate	SuperUser	26/09/2013	18.33.40	N 4: 0 => 44
Num Gate	SuperUser	26/09/2013	18.33.42	N 7: 0 => 54
Num Gate	SuperUser	26/09/2013	18.33.44	N 2: 0 => 33
Num Gate	SuperUser	26/09/2013	18.33.46	N 10: 0 => 22
Recipe	SuperUser	26/09/2013	18.33.53	Imported: 26/09/2013
Recipe	SuperUser	26/09/2013	18.33.55	Modified: SetPoints
Num Gate	SuperUser	26/09/2013	18.34.09	N 4: 44 => 0
Num Gate	SuperUser	26/09/2013	18.34.11	N 2: 33 => 0
Num Gate	SuperUser	26/09/2013	18.34.17	N 2: 0 => 33

### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the OperatorView object (in pixels).

**Top:** vertical position of the top left corner of the OperatorView object (in pixels).

**Width:** width of the OperatorView object (in pixels).

**Height:** height of the OperatorView object (in pixels).

**Show server button:** enable/disable "SERVER" button to select from which server computer the operator historical files must be loaded.

**Show Config button:** enable/disable "CONFIG" button to select the operator historical view time range.

**Show Print button:** enable/disable "PRINT" button to print the operator historical view list on the default printer

**Show Save CSV button:** enable/disable "Save CSV" button to export the operator historical view list on a formatted text file with TAB separator.

**Window Bkgrnd color:** window background color.

**Default server:** from which server historical data must be read.

- "Local": historical data must be read from the local computer.
- "Channel\_x": historical data must be read from the server specified in channel configuration.

#### Commands (CodeBuilder)

There is the possibility to send some commands to Operator view object by using "**TObjFunction(int ObjId, int Function)**" CodeBuilder instruction.

Parameters:

**int ObjId** : the number that identify the component (see ID property of the object).

**int Function** : function to perform.

Function	Description
1	Show server selection window
2	Show configure time window
3	Save CSV file
4	Print
13	Stop loading historical files

## 8.28 ReportView

With the ReportView component you can show the reports management page. For more details refer to the paragraph "How to show historical reports" in the *RunTime* chapter.



#### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the ReportView object (in pixels).

**Top:** vertical position of the top left corner of the ReportView object (in pixels).

**Width:** width of the ReportView object (in pixels).

**Height:** height of the ReportView object (in pixels).

**Show Rename button:** indicates if RENAME BUTTON must be shown in the ReportView object.

**Show Delete button:** indicates if DELETE BUTTON must be shown in the ReportView object.

**Show Report button:** indicates if REPORT BUTTON must be shown in the ReportView object.

**Show Template button :** indicates if TEMPLATE BUTTON must be shown in the ReportView object.

**Show report type:** indicates if must be shown only one window with all the reports generated or if must be shown also another window with a tree of reports type to facilitate the search of a specific file.

**Window Bkgrnd color:** window background color.

## 8.29 AlarmsView

With the AlarmsView component you can show the online alarms or events. For more details refer to the paragraph Alarms Status or Events Status in the *RunTime* chapter.



**Height:** height of the AlarmsView (in pixels).

**Class 1:** will be shown only alarms or events which have Class1 equal to the number specified in this field.

**Class 2:** will be shown only alarms or events which have Class2 equal to the string specified in this field.

**Class 3, Class 4, Class 5, Class 6, Class 7:** will be shown only alarms or events which have ClassX equal to the value of the gates specified in this fields.

**Show:** allow to select alarms or events visualization.

**Show Sort button:** enable/disable visualization of SORT button.

**Show Filter button:** enable/disable visualization of FILTER button.

**Show Confirm button:** enable/disable visualization of CONFIRM button.

**Show Confirm all button:** enable/disable visualization of CONFIRM ALL button.

**Show Exclude chkBtn:** enable/disable visualization of EXCLUDE check button.

**Window Bkgrnd color:** window background color.

**Items Bkgrnd color:** background color of the area that contains the alarms list.

**On Click:** operation to carry out after a click on a row of an active alarm. Following the possible operations:

- Call function: call the indicated function.
- Stop function: stop the indicated function (if this is on).
- Open template: open the indicated template.
- Close template: close the current template.

**Columns:** allow to select which columns to display in the object.

#### Commands (CodeBuilder)

There is the possibility to send some commands to Alarms view object by using "TObjFunction(int ObjId, int Function)" CodeBuilder instruction.

Parameters:

**int ObjId** : the number that identify the component (see ID property of the object).

**int Function** : function to perform.

Function	Description
1	Show alarm sort type window
2	Acknowledge selected alarm
3	Acknowledge all alarms
4	Set alarm excluded
5	Show alarm filter window
6	Print

Following properties can be read also from language (see Code Builder help)

Property	Function
Selected item (*Example 1)	TObjGetPropertyInt(Id,"ItemSelected")

\*Example 1 : this function return the index of the "Event/Alarm" gate currently selected in the object. Index refer to the Event/Alarms gates list.

Suppose to want to show "Hello World!" message if the gate corresponding to the row currently selected in an AlarmsView object with ID=100 is "Alarm,3"; the following code provide the solution:

```
Function void GetSelectedAlarm()
    string Gateld;
    int NId;
    int Index;
    Index=TObjGetPropertyInt(100,ItemSelected");
    if (Index!=-1) then
        Gateld=GetEvnGateGateID(Index);
        NId=GetEvnGateNID(Index);
        if (Gateld=="Alarm" && NId==3) then
            MessageBox("Hello World!","Found");
        end
    end
end
```

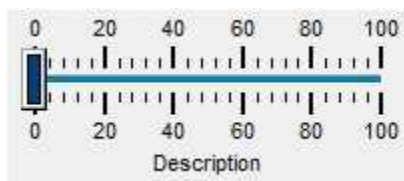
Following properties can be set also from language (see Code Builder help)

Property	Function
Filter Class 1 (*Example 2)	TObjSetPropertyInt(Id,"Class1",...)
Filter Class 2 (*Example 2)	TObjSetPropertyString(Id,"Class2",...)

\*Example 2 : suppose to have a template with an AlarmsView object with ID=100 and want to show only alarms having Class1=102 and Class2="Temp":

```
Function void SetAlarmsFilter()
    TObjBeginUpdate(100);
    TObjSetPropertyInt(100,"Class1","102");
    TObjSetPropertyString(100,"Class2","Temp");
    TObjEndUpdate(100);
end
```

## 8.30 HSlider



Horizontal Slider



**Properties (TemplateBuilder)**

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the object (in pixels).

**Top:** vertical position of the top left corner of the object (in pixels).

**Width:** width of the object (in pixels).

**Height:** height of the object (in pixels).

**Description:** description of the object (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Gate:** gate from which to read the value that will be displayed by the object during the supervision phase.

**Cursor:** shape that the cursor of the mouse must have when it goes over the object during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

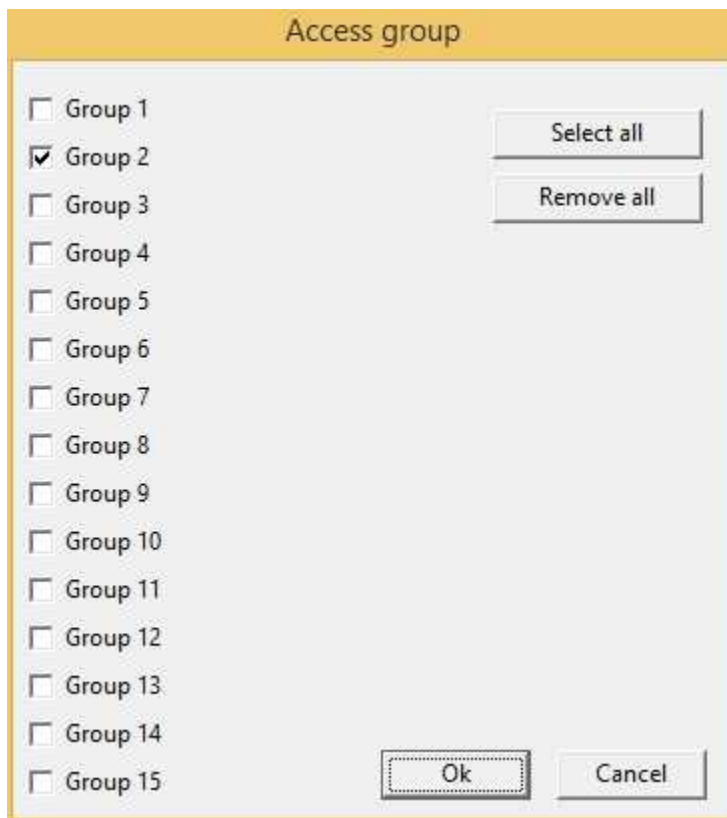
**Need apply:** it indicates whether the status change of the object will have to be confirmed from outside with a Button (Yes), or will be applied immediately (No).

**Tab num:** order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

**Help file name :** if in this field is specified a valid ".CHM" file name and it is present an HtmlHelp object in one of the templates currently showed on the screen, when the HSlider is clicked, the .CHM file will be showed in the HtmlHelp window. The path can be specified either in absolute (eg. "c:\Data\Help.chm") or in relative way (eg. "..\Data\Help.chm"): using the relative way, the base directory will be the application's templates directory. See HtmlHelp object for more details.

**Show** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

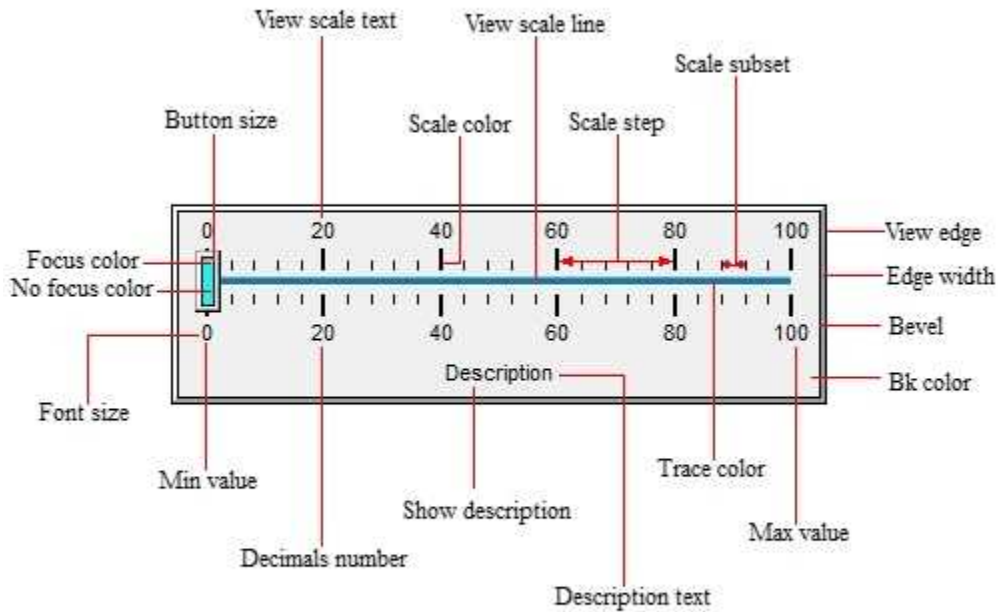
**Access group:** users groups enabled to operate on the object in runtime mode. By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



The screenshot shows a dialog box titled "Access group". It contains a list of 15 groups, each with a checkbox. "Group 2" is checked. To the right of the list are two buttons: "Select all" and "Remove all". At the bottom of the dialog are "Ok" and "Cancel" buttons.

Group	Selected
Group 1	<input type="checkbox"/>
Group 2	<input checked="" type="checkbox"/>
Group 3	<input type="checkbox"/>
Group 4	<input type="checkbox"/>
Group 5	<input type="checkbox"/>
Group 6	<input type="checkbox"/>
Group 7	<input type="checkbox"/>
Group 8	<input type="checkbox"/>
Group 9	<input type="checkbox"/>
Group 10	<input type="checkbox"/>
Group 11	<input type="checkbox"/>
Group 12	<input type="checkbox"/>
Group 13	<input type="checkbox"/>
Group 14	<input type="checkbox"/>
Group 15	<input type="checkbox"/>

**Enable:** object is enabled if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.



#### Commands (CodeBuilder)

Following properties can be set also from language (see Code Builder help)

Property	Function
Min value	TObjSetPropertyReal(Id,"ScaleMin",...)
Max value	TObjSetPropertyReal(Id,"ScaleMax",...)
Decimals number	TObjSetPropertyInt(Id,"DecimalNumber",...)
Scale step	TObjSetPropertyInt(Id,"ScaleStep",...)
Scale substep	TObjSetPropertyInt(Id,"ScaleSubStep",...)
Description text	TObjSetPropertyString(Id,"DescriptionText",...)

Before setting all required properties, it is necessary to call **TObjBeginUpdate(Id)** function. After setting all required properties, it is necessary to call **TObjEndUpdate(Id)** function to update the object.

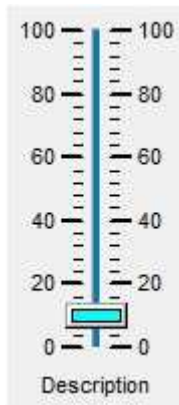
*Example:*

```

...
TObjBeginUpdate(100);
TObjSetPropertyReal(100,"ScaleMin",20);
TObjSetPropertyReal(100,"ScaleMax",80);
TObjEndUpdate(100);
...

```

## 8.31 VSlider



**Vertical Slider**

### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the object (in pixels).

**Top:** vertical position of the top left corner of the object (in pixels).

**Width:** width of the object (in pixels).

**Height:** height of the object (in pixels).

**Description:** description of the object (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Gate:** gate from which to read the value that will be displayed by the object during the supervision phase.

**Cursor:** shape that the cursor of the mouse must have when it goes over the object during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

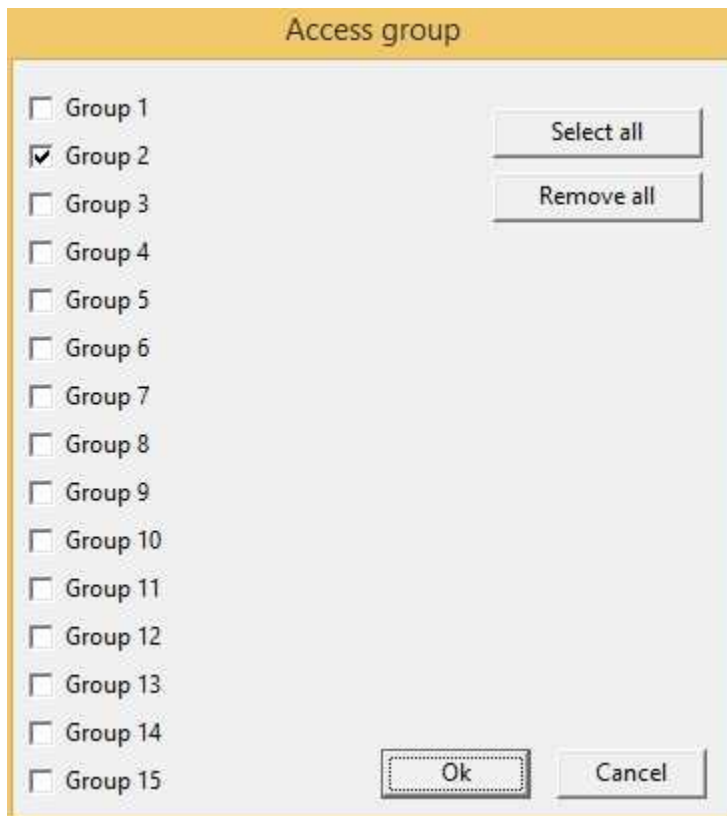
**Need apply:** it indicates whether the status change of the object will have to be confirmed from outside with a Button (Yes), or will be applied immediately (No).

**Tab num:** order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

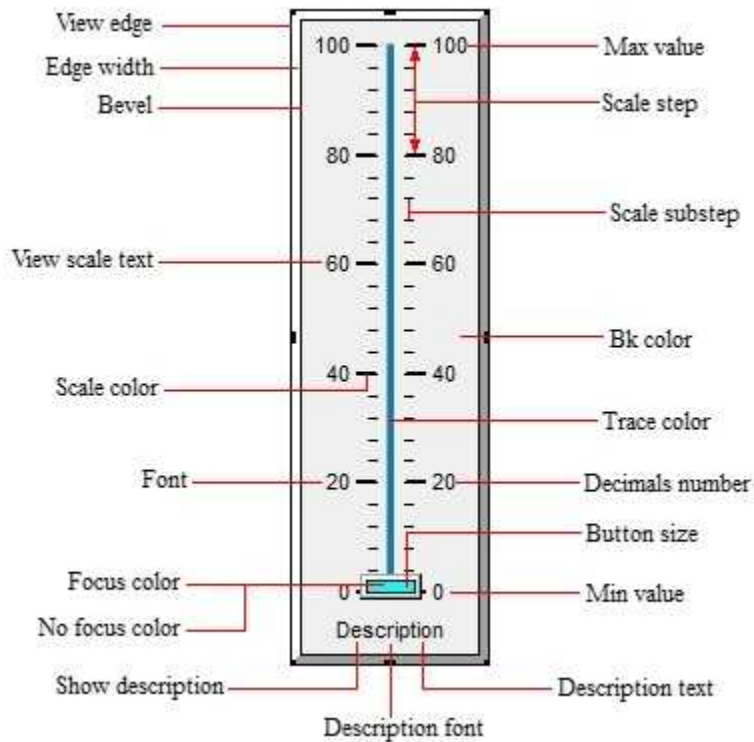
**Help file name :** if in this field is specified a valid ".CHM" file name and it is present an HtmlHelp object in one of the templates currently showed on the screen, when the VSlider is clicked, the .CHM file will be showed in the HtmlHelp window. The path can be specified either in absolute (eg. "c:\Data\Help.chm") or in relative way (eg. "..\Data\Help.chm"): using the relative way, the base directory will be the application's templates directory. See HtmlHelp object for more details.

**Show** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group:** users groups enabled to operate on the object in runtime mode. By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.



### Commands (CodeBuilder)

Following properties can be set also from language (see Code Builder help)

Property	Function
Min value	TObjSetPropertyReal(Id,"ScaleMin",...)
Max value	TObjSetPropertyReal(Id,"ScaleMax",...)
Decimals number	TObjSetPropertyInt(Id,"DecimalNumber",...)
Scale step	TObjSetPropertyInt(Id,"ScaleStep",...)
Scale substep	TObjSetPropertyInt(Id,"ScaleSubStep",...)
Description text	TObjSetPropertyString(Id,"DescriptionText",...)

Before setting all required properties, it is necessary to call **TObjBeginUpdate(Id)** function. After setting all required properties, it is necessary to call **TObjEndUpdate(Id)** function to update the object.

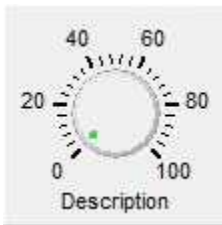
*Example:*

```

...
TObjBeginUpdate(100);
TObjSetPropertyReal(100,"ScaleMin",20);
TObjSetPropertyReal(100,"ScaleMax",80);
TObjEndUpdate(100);
...

```

## 8.32 Dial



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the object (in pixels).

**Top:** vertical position of the top left corner of the object (in pixels).

**Width:** width of the object (in pixels).

**Height:** height of the object (in pixels).

**Description:** description of the object (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Gate:** gate from which to read the value that will be displayed by the object during the supervision phase.

**Cursor:** shape that the cursor of the mouse must have when it goes over the object during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

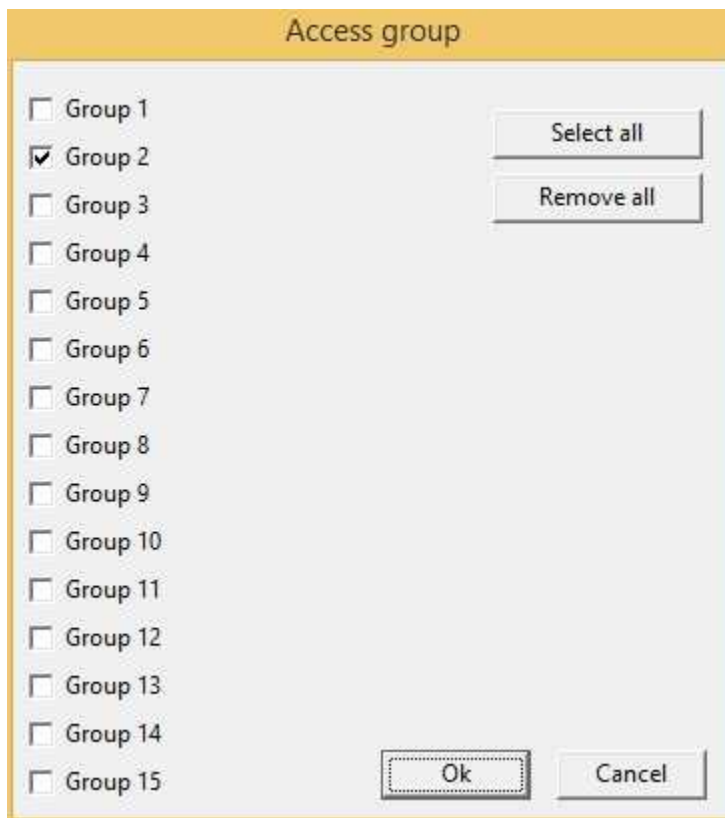
**Need apply:** it indicates whether the status change of the object will have to be confirmed from outside with a Button (Yes), or will be applied immediately (No).

**Tab num:** order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

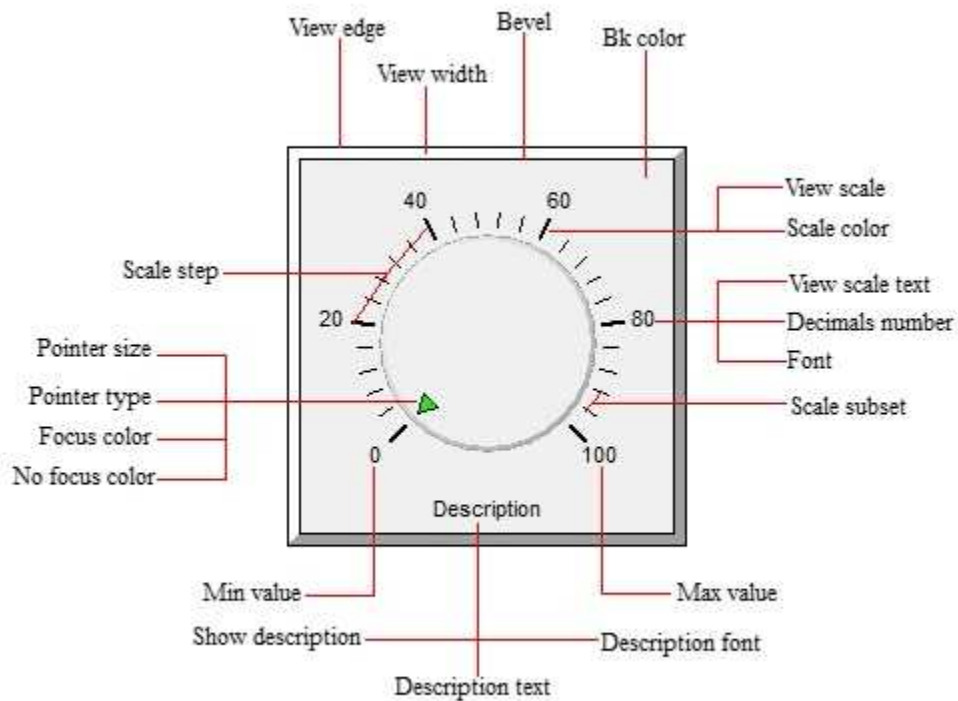
**Help file name :** if in this field is specified a valid ".CHM" file name and it is present an HtmlHelp object in one of the templates currently showed on the screen, when the Dial is clicked, the .CHM file will be showed in the HtmlHelp window. The path can be specified either in absolute (eg. "c:\Data\Help.chm") or in relative way (eg. "..\Data\Help.chm"): using the relative way, the base directory will be the application's templates directory. See HtmlHelp object for more details.

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group:** users groups enabled to operate on the object in runtime mode. By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.





**Commands (CodeBuilder)**

Following properties can be set also from language (see Code Builder help)

Property	Function
Min value	TObjSetPropertyReal(Id,"ScaleMin",...)
Max value	TObjSetPropertyReal(Id,"ScaleMax",...)
Decimals number	TObjSetPropertyInt(Id,"DecimalNumber",...)
Scale step	TObjSetPropertyInt(Id,"ScaleStep",...)
Scale substep	TObjSetPropertyInt(Id,"ScaleSubStep",...)
Description text	TObjSetPropertyString(Id,"DescriptionText",...)

Before setting all required properties, it is necessary to call **TObjBeginUpdate(Id)** function.  
After setting all required properties, it is necessary to call **TObjEndUpdate(Id)** function to update the object.

*Example:*

```
...
TObjBeginUpdate(100);
TObjSetPropertyReal(100,"ScaleMin",20);
TObjSetPropertyReal(100,"ScaleMax",80);
TObjEndUpdate(100);
...
```

**8.33 GearDial****Properties (TemplateBuilder)**

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the object (in pixels).

**Top:** vertical position of the top left corner of the object (in pixels).

**Width:** width of the object (in pixels).

**Height:** height of the object (in pixels).

**Description:** description of the object (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Gate:** gate from which to read the value that will be displayed by the object during the supervision phase.

**Cursor:** shape that the cursor of the mouse must have when it goes over the object during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

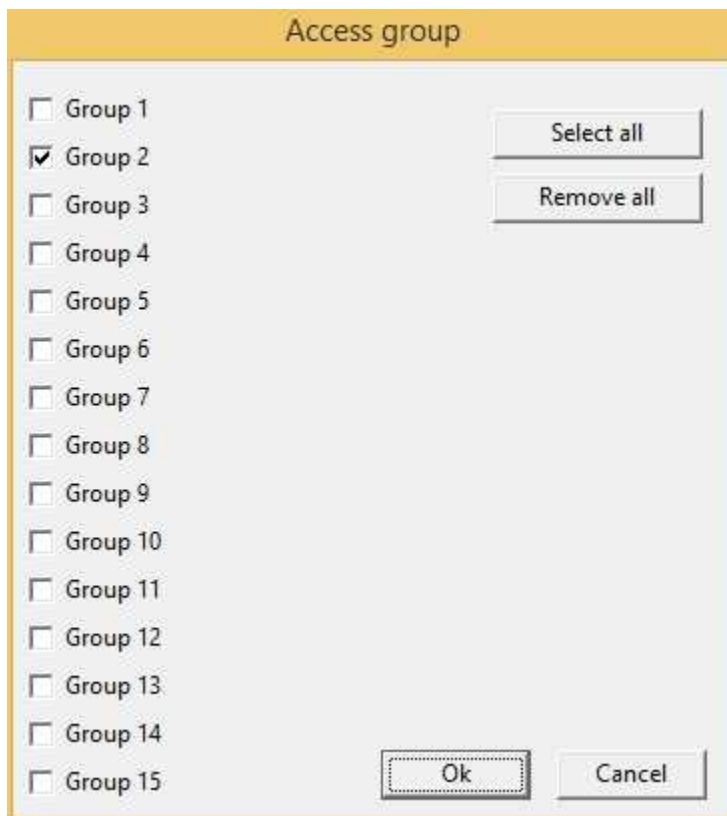
**Need apply:** it indicates whether the status change of the object will have to be confirmed from outside with a Button (Yes), or will be applied immediately (No).

**Tab num:** order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

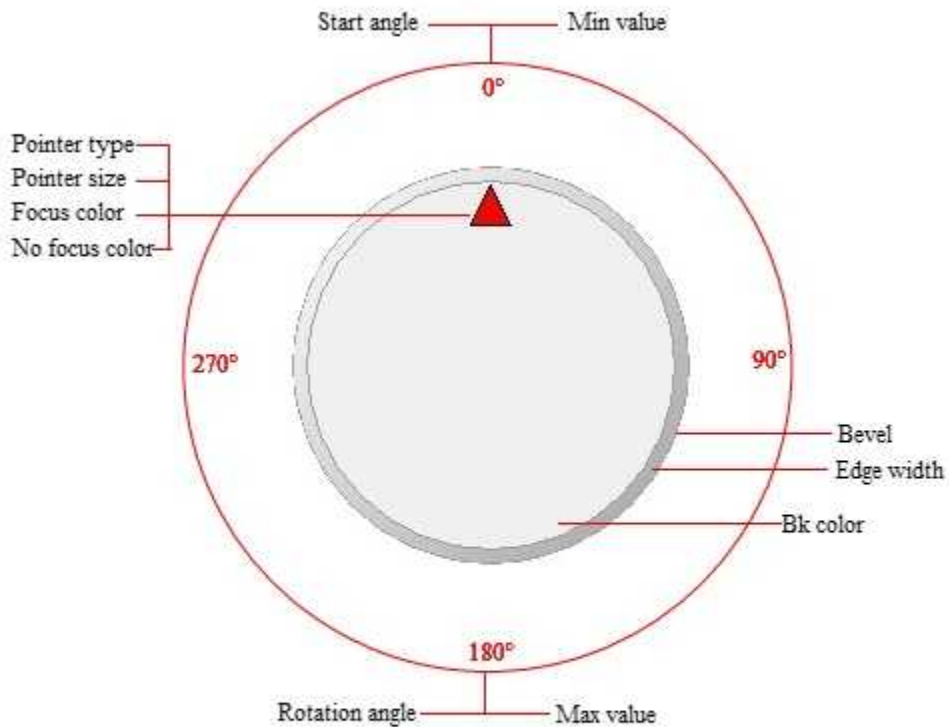
**Help file name :** if in this field is specified a valid ".CHM" file name and it is present an HtmlHelp object in one of the templates currently showed on the screen, when the GearDial is clicked, the .CHM file will be showed in the HtmlHelp window. The path can be specified either in absolute (eg. "c:\Data\Help.chm") or in relative way (eg. ".\Data\Help.chm"): using the relative way, the base directory will be the application's templates directory. See HtmlHelp object for more details.

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group:** users groups enabled to operate on the object in runtime mode. By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.



#### Commands (CodeBuilder)

Following properties can be set also from language (see Code Builder help)

Property	Function
Min value	TObjSetPropertyReal(Id,"ScaleMin",...)
Max value	TObjSetPropertyReal(Id,"ScaleMax",...)

Before setting all required properties, it is necessary to call **TObjBeginUpdate(Id)** function.  
 After setting all required properties, it is necessary to call **TObjEndUpdate(Id)** function to update the object.

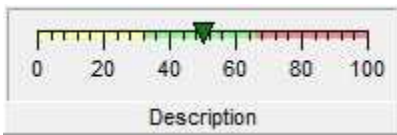
*Example:*

```

...
  TObjBeginUpdate(100);
  TObjSetPropertyReal(100,"ScaleMin",20);
  TObjSetPropertyReal(100,"ScaleMax",80);
  TObjEndUpdate(100);
...

```

## 8.34 HMeter



**Horizontal Meter**

### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the object (in pixels).

**Top:** vertical position of the top left corner of the object (in pixels).

**Width:** width of the object (in pixels).

**Height:** height of the object (in pixels).

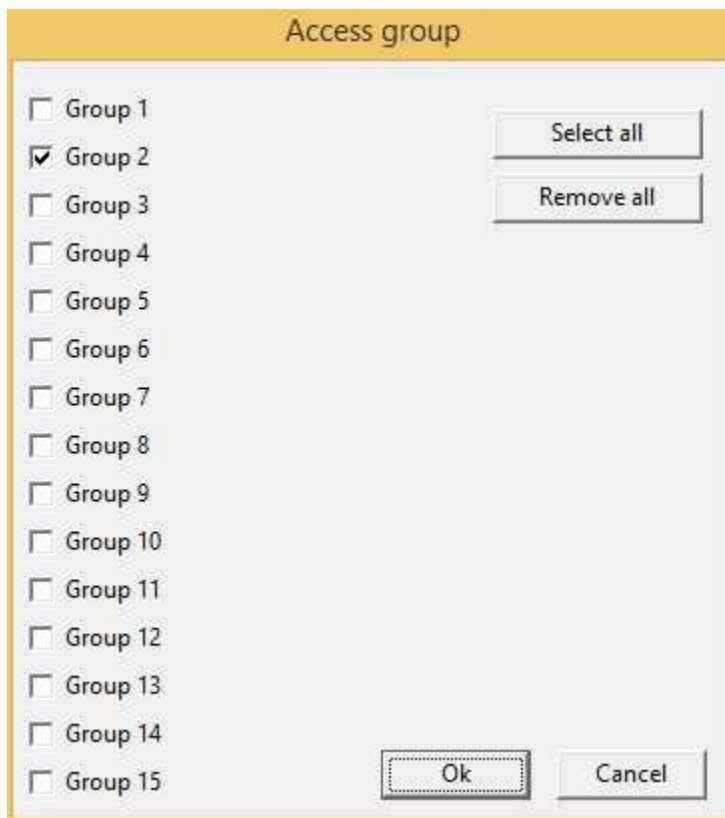
**Description:** description of the object (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Gate:** gate from which to read the value that will be displayed by the object during the supervision phase.

**Cursor:** shape that the cursor of the mouse must have when it goes over the object during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

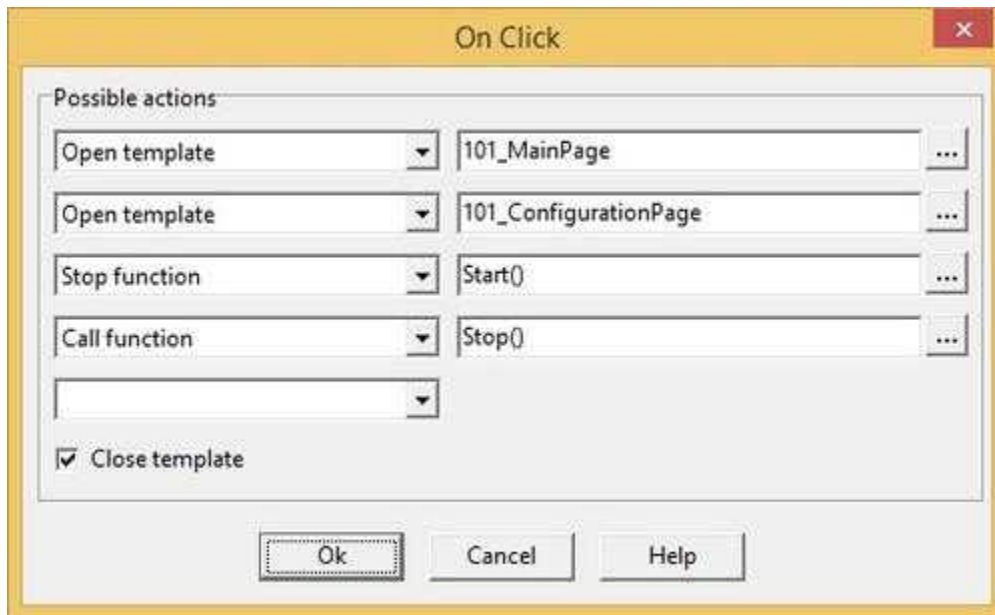
**Access group:** users groups enabled to operate on the object in runtime mode (enables "On Click" function). By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



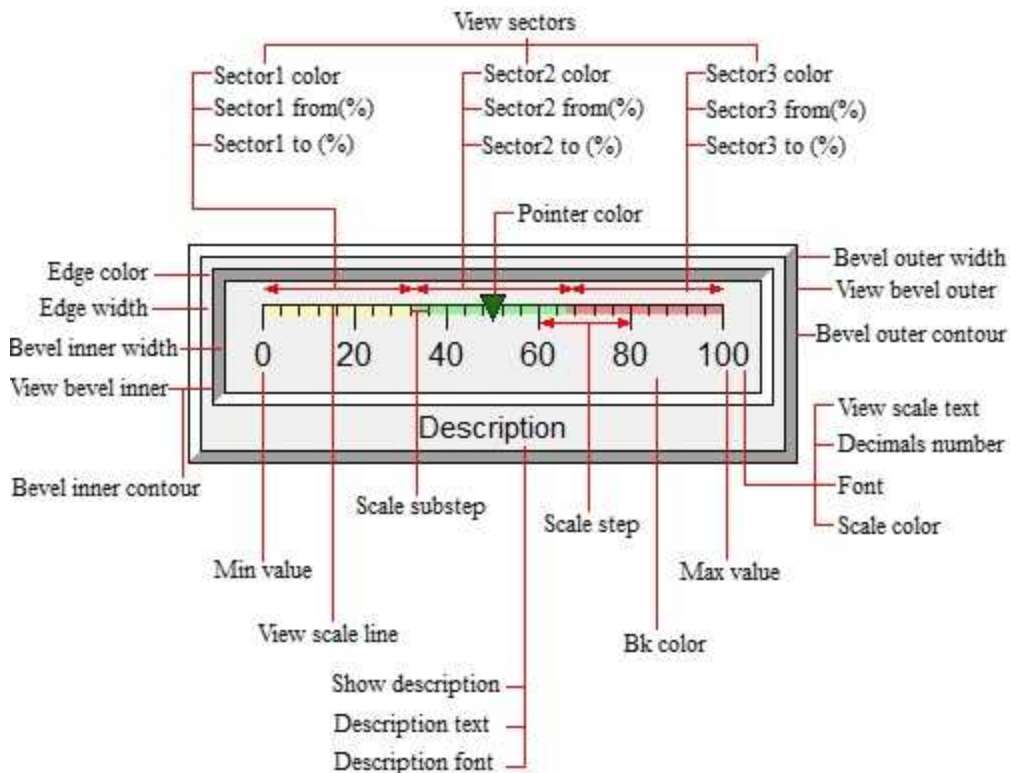
**Enable:** object is enabled (enables "On Click" and "On Double Click" functions) if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

**On Click:** operation to carry out when the user click on the object. Next figure show the window through which to specify the list of the operations to carry out:

- Call function: call the indicated function.
- Stop function: stop the indicated function (if this is on).
- Open template: open the indicated template.
- Apply changes: apply the changes in all the object brothers or child components that have the "Need apply" property set to "Yes" (the new values will be written in the gates).
- Undo changes: undo the changes in all the object brothers or child components that have the "Need apply" property set to "Yes".
- Close template: close the current template.



**On Double Click:** operation to carry out when the user double click on the object. Refer to "On Click" property to specify the list of the operations to carry out.



#### Commands (CodeBuilder)

Following properties can be set also from language (see Code Builder help)

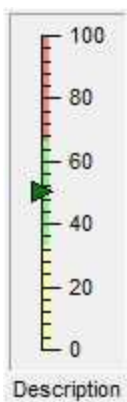
Property	Function
Min value	TObjSetPropertyReal(Id,"ScaleMin",...)
Max value	TObjSetPropertyReal(Id,"ScaleMax",...)
Decimals number	TObjSetPropertyInt(Id,"DecimalNumber",...)
Scale step	TObjSetPropertyInt(Id,"ScaleStep",...)
Scale substep	TObjSetPropertyInt(Id,"ScaleSubStep",...)
Sector 1 from (%)	TObjSetPropertyInt(Id,"Sector1From",...)
Sector 1 to (%)	TObjSetPropertyInt(Id,"Sector1To",...)
Sector 2 from (%)	TObjSetPropertyInt(Id,"Sector2From",...)
Sector 2 to (%)	TObjSetPropertyInt(Id,"Sector2To",...)
Sector 3 from (%)	TObjSetPropertyInt(Id,"Sector3From",...)
Sector 3 to (%)	TObjSetPropertyInt(Id,"Sector3To",...)
Description text	TObjSetPropertyString(Id,"DescriptionText",...)

Before setting all required properties, it is necessary to call **TObjBeginUpdate(Id)** function. After setting all required properties, it is necessary to call **TObjEndUpdate(Id)** function to update the object.

*Example:*

```
...
TObjBeginUpdate(100);
TObjSetPropertyReal(100,"ScaleMin",20);
TObjSetPropertyReal(100,"ScaleMax",80);
TObjEndUpdate(100);
...
```

### 8.35 VMeter



**Vertical Meter**

#### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the object (in pixels).

**Top:** vertical position of the top left corner of the object (in pixels).

**Width:** width of the object (in pixels).

**Height:** height of the object (in pixels).

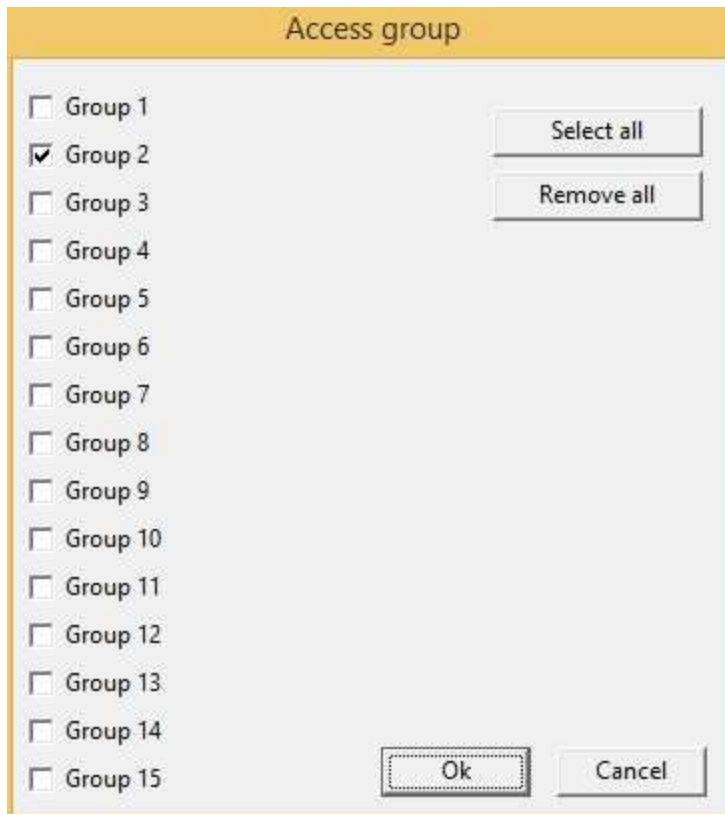
**Description:** description of the object (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Gate:** gate from which to read the value that will be displayed by the object during the supervision phase.

**Cursor:** shape that the cursor of the mouse must have when it goes over the object during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group:** users groups enabled to operate on the object in runtime mode (enables "On Click" function). By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.

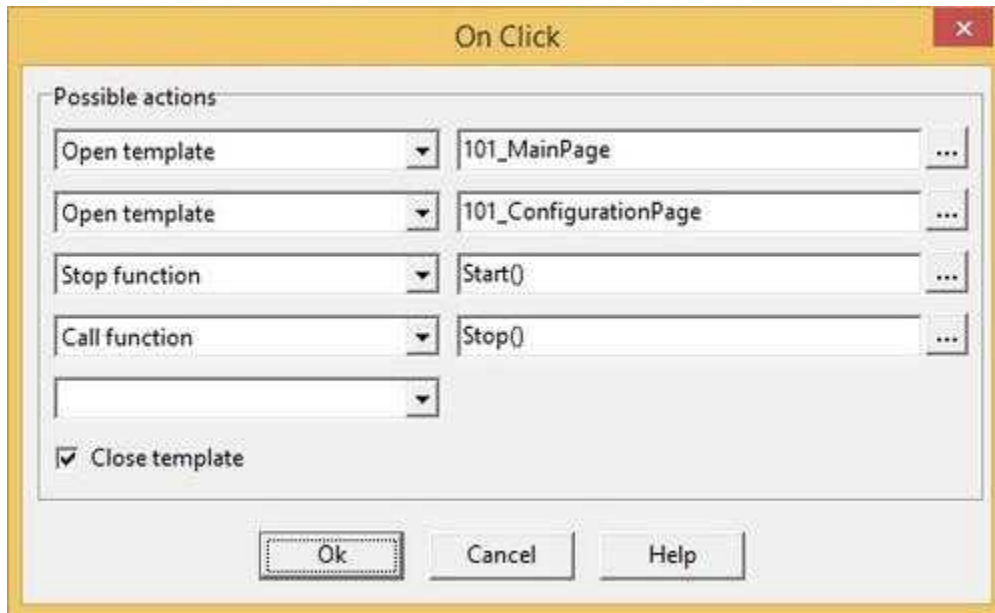


**Enable:** object is enabled (enables "On Click" and "On Double Click" functions) if it is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

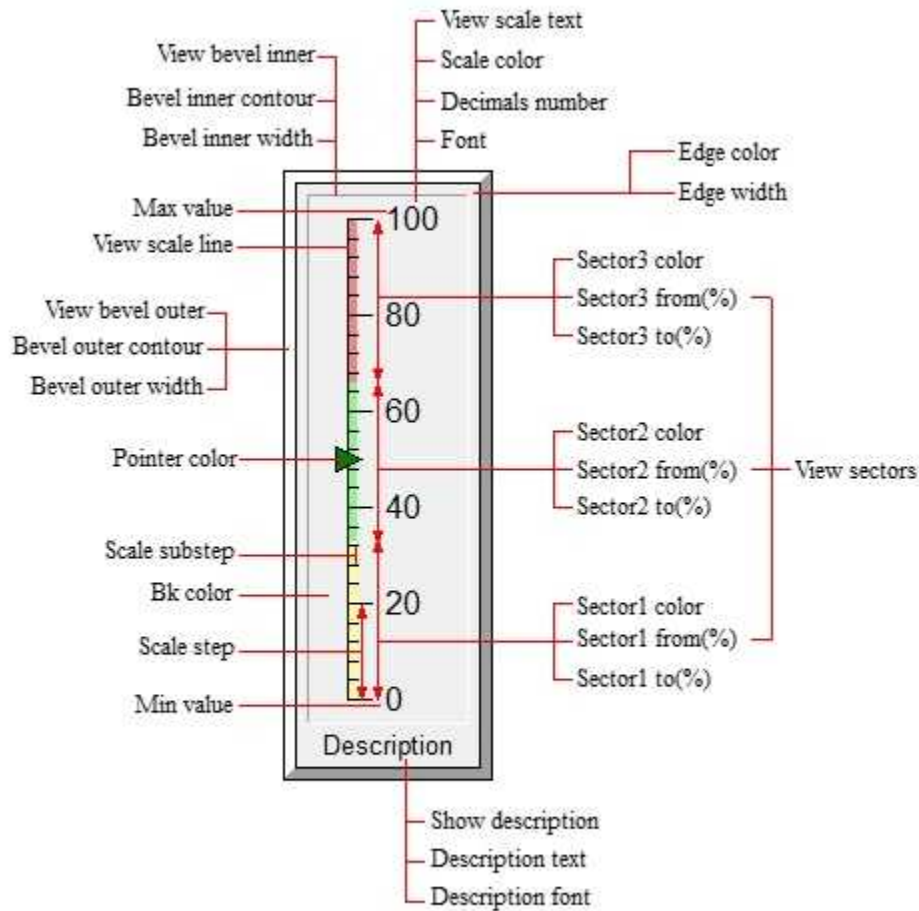


**On Click:** operation to carry out when the user click on the object. Next figure show the window through which to specify the list of the operations to carry out:

- Call function: call the indicated function.
- Stop function: stop the indicated function (if this is on).
- Open template: open the indicated template.
- Apply changes: apply the changes in all the object brothers or child components that have the "Need apply" property set to "Yes" (the new values will be written in the gates).
- Undo changes: undo the changes in all the object brothers or child components that have the "Need apply" property set to "Yes".
- Close template: close the current template.



**On Double Click:** operation to carry out when the user double click on the object. Refer to "On Click" property to specify the list of the operations to carry out.



### Commands (CodeBuilder)

Following properties can be set also from language (see Code Builder help)

Property	Function
Min value	<code>TObjSetPropertyReal(Id,"ScaleMin",...)</code>
Max value	<code>TObjSetPropertyReal(Id,"ScaleMax",...)</code>
Decimals number	<code>TObjSetPropertyInt(Id,"DecimalNumber",...)</code>
Scale step	<code>TObjSetPropertyInt(Id,"ScaleStep",...)</code>
Scale substep	<code>TObjSetPropertyInt(Id,"ScaleSubStep",...)</code>
Sector 1 from (%)	<code>TObjSetPropertyInt(Id,"Sector1From",...)</code>
Sector 1 to (%)	<code>TObjSetPropertyInt(Id,"Sector1To",...)</code>
Sector 2 from (%)	<code>TObjSetPropertyInt(Id,"Sector2From",...)</code>
Sector 2 to (%)	<code>TObjSetPropertyInt(Id,"Sector2To",...)</code>
Sector 3 from (%)	<code>TObjSetPropertyInt(Id,"Sector3From",...)</code>

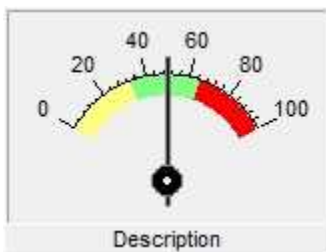
Sector 3 to (%)	TObjSetPropertyInt(Id,"Sector3To",...)
Description text	TObjSetPropertyString(Id,"DescriptionText",...)

Before setting all required properties, it is necessary to call **TObjBeginUpdate(Id)** function.  
After setting all required properties, it is necessary to call **TObjEndUpdate(Id)** function to update the object.

*Example:*

```
...
  TObjBeginUpdate(100);
  TObjSetPropertyReal(100,"ScaleMin",20);
  TObjSetPropertyReal(100,"ScaleMax",80);
  TObjEndUpdate(100);
...
```

## 8.36 120Meter



**120° Meter**

### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the object (in pixels).

**Top:** vertical position of the top left corner of the object (in pixels).

**Width:** width of the object (in pixels).

**Height:** height of the object (in pixels).

**Description:** description of the object (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Gate:** gate from which to read the value that will be displayed by the object during the supervision phase.

**Cursor:** shape that the cursor of the mouse must have when it goes over the object during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group:** users groups enabled to operate on the object in runtime mode (enables "On Click")

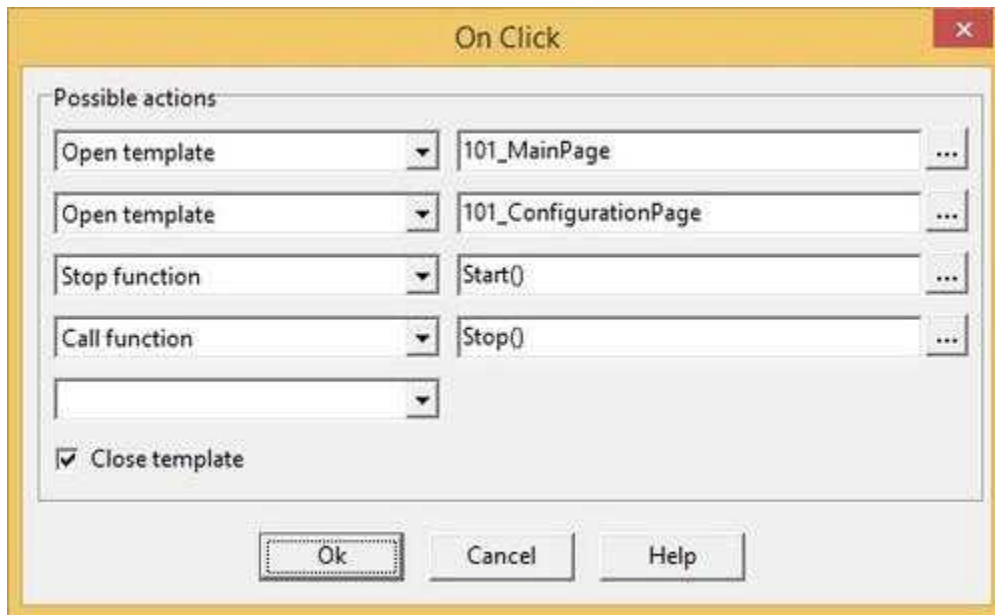
function). By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



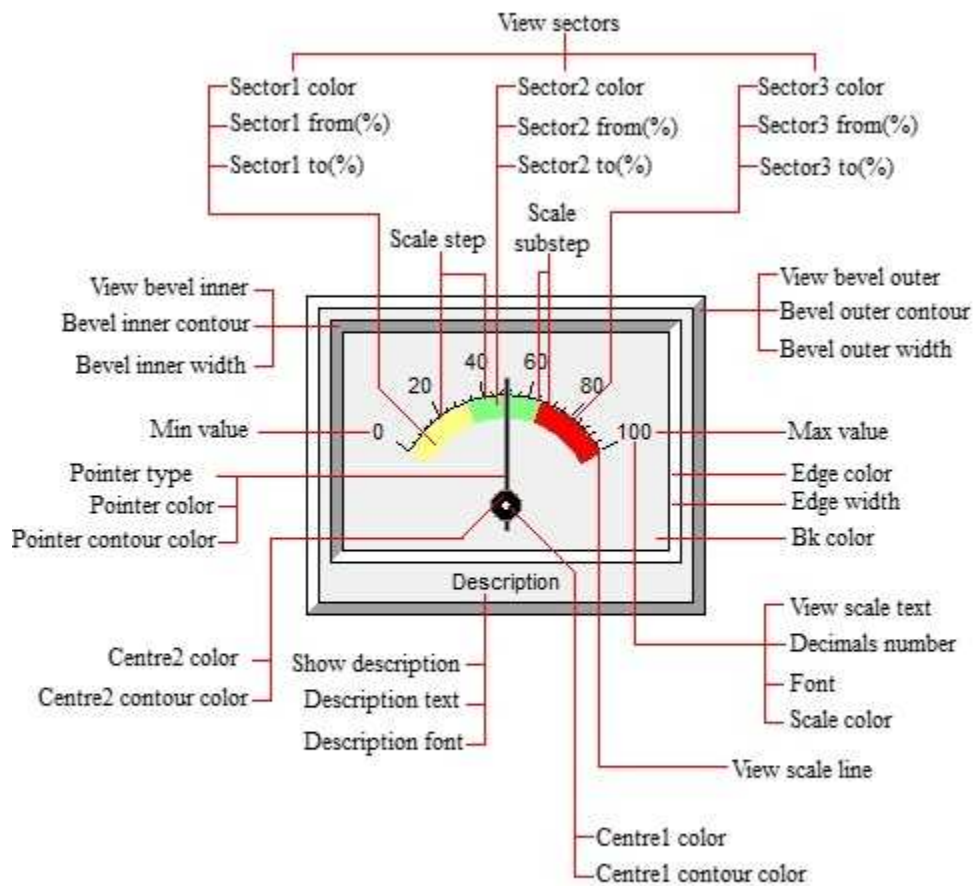
**Enable:** object is enabled (enables "On Click" and "On Double Click" functions) if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

**On Click:** operation to carry out when the user click on the object. Next figure show the window through which to specify the list of the operations to carry out:

- Call function: call the indicated function.
- Stop function: stop the indicated function (if this is on).
- Open template: open the indicated template.
- Apply changes: apply the changes in all the object brothers or child components that have the "Need apply" property set to "Yes" (the new values will be written in the gates).
- Undo changes: undo the changes in all the object brothers or child components that have the "Need apply" property set to "Yes".
- Close template: close the current template.



**On Double Click:** operation to carry out when the user double click on the object. Refer to "On Click" property to specify the list of the operations to carry out.



### Commands (CodeBuilder)

Following properties can be set also from language (see Code Builder help)

Property	Function
Min value	TObjSetPropertyReal(Id,"ScaleMin",...)
Max value	TObjSetPropertyReal(Id,"ScaleMax",...)
Decimals number	TObjSetPropertyInt(Id,"DecimalNumber",...)
Scale step	TObjSetPropertyInt(Id,"ScaleStep",...)
Scale substep	TObjSetPropertyInt(Id,"ScaleSubStep",...)
Sector 1 from (%)	TObjSetPropertyInt(Id,"Sector1From",...)
Sector 1 to (%)	TObjSetPropertyInt(Id,"Sector1To",...)
Sector 2 from (%)	TObjSetPropertyInt(Id,"Sector2From",...)
Sector 2 to (%)	TObjSetPropertyInt(Id,"Sector2To",...)

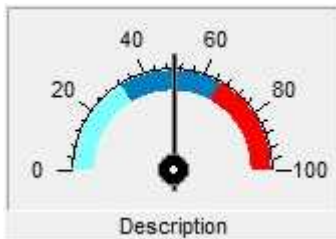
Sector 3 from (%)	TObjSetPropertyInt(Id,"Sector3From",...)
Sector 3 to (%)	TObjSetPropertyInt(Id,"Sector3To",...)
Description text	TObjSetPropertyString(Id,"DescriptionText",...)

Before setting all required properties, it is necessary to call **TObjBeginUpdate(Id)** function.  
After setting all required properties, it is necessary to call **TObjEndUpdate(Id)** function to update the object.

*Example:*

```
...
  TObjBeginUpdate(100);
  TObjSetPropertyReal(100,"ScaleMin",20);
  TObjSetPropertyReal(100,"ScaleMax",80);
  TObjEndUpdate(100);
...
```

### 8.37 180Meter



**180° Meter**

#### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the object (in pixels).

**Top:** vertical position of the top left corner of the object (in pixels).

**Width:** width of the object (in pixels).

**Height:** height of the object (in pixels).

**Description:** description of the object (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Gate:** gate from which to read the value that will be displayed by the object during the supervision phase.

**Cursor:** shape that the cursor of the mouse must have when it goes over the object during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group:** users groups enabled to operate on the object in runtime mode (enables "On Click")

function). By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.

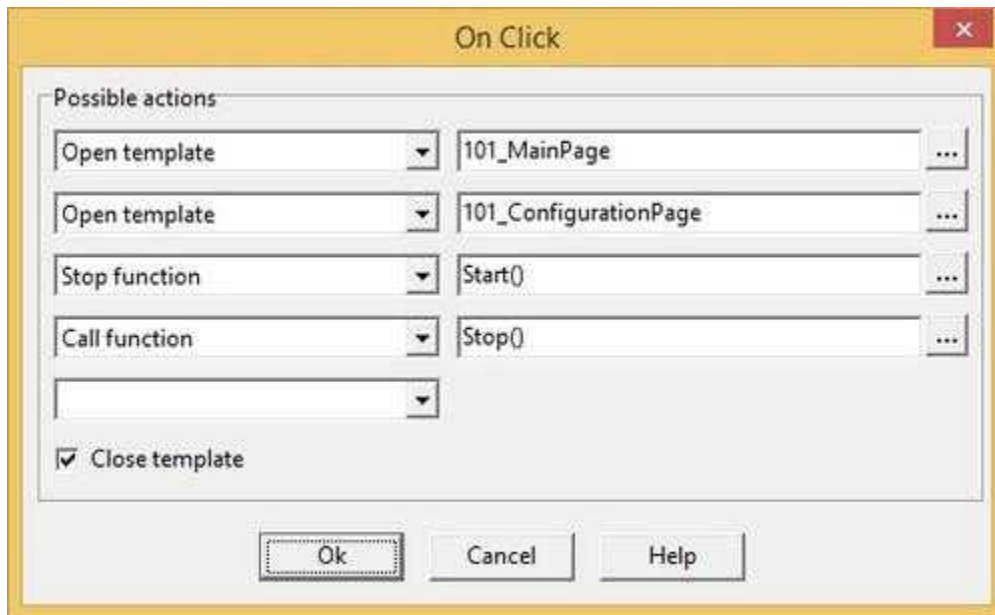


**Enable:** object is enabled (enables "On Click" and "On Double Click" functions) if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

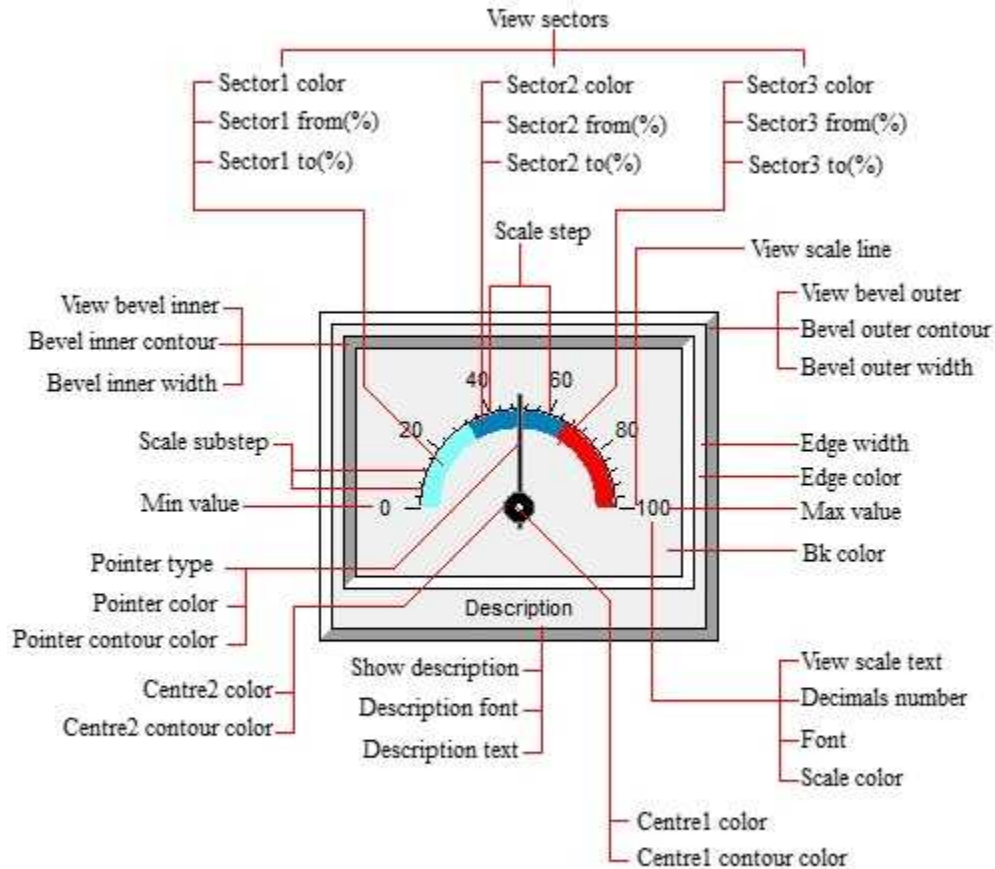
**On Click:** operation to carry out when the user click on the object. Next figure show the window through which to specify the list of the operations to carry out:

- Call function: call the indicated function.
- Stop function: stop the indicated function (if this is on).
- Open template: open the indicated template.
- Apply changes: apply the changes in all the object brothers or child components that have the "Need apply" property set to "Yes" (the new values will be written in the gates).
- Undo changes: undo the changes in all the object brothers or child components that have the "Need apply" property set to "Yes".
- Close template: close the current template.





**On Double Click:** operation to carry out when the user double click on the object. Refer to "On Click" property to specify the list of the operations to carry out.



### Commands (CodeBuilder)

Following properties can be set also from language (see Code Builder help)

Property	Function
Min value	TObjSetPropertyReal(Id,"ScaleMin",...)
Max value	TObjSetPropertyReal(Id,"ScaleMax",...)
Decimals number	TObjSetPropertyInt(Id,"DecimalNumber",...)
Scale step	TObjSetPropertyInt(Id,"ScaleStep",...)
Scale substep	TObjSetPropertyInt(Id,"ScaleSubStep",...)
Sector 1 from (%)	TObjSetPropertyInt(Id,"Sector1From",...)
Sector 1 to (%)	TObjSetPropertyInt(Id,"Sector1To",...)
Sector 2 from (%)	TObjSetPropertyInt(Id,"Sector2From",...)
Sector 2 to (%)	TObjSetPropertyInt(Id,"Sector2To",...)
Sector 3 from (%)	TObjSetPropertyInt(Id,"Sector3From",...)

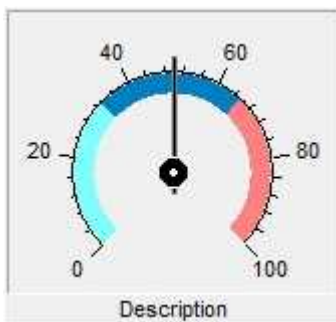
Sector 3 to (%)	TObjSetPropertyInt(Id,"Sector3To",...)
Description text	TObjSetPropertyString(Id,"DescriptionText",...)

Before setting all required properties, it is necessary to call **TObjBeginUpdate(Id)** function.  
After setting all required properties, it is necessary to call **TObjEndUpdate(Id)** function to update the object.

*Example:*

```
...
  TObjBeginUpdate(100);
  TObjSetPropertyReal(100,"ScaleMin",20);
  TObjSetPropertyReal(100,"ScaleMax",80);
  TObjEndUpdate(100);
...
```

### 8.38 270Meter



**270° Meter**

#### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the object (in pixels).

**Top:** vertical position of the top left corner of the object (in pixels).

**Width:** width of the object (in pixels).

**Height:** height of the object (in pixels).

**Description:** description of the object (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Gate:** gate from which to read the value that will be displayed by the object during the supervision phase.

**Cursor:** shape that the cursor of the mouse must have when it goes over the object during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

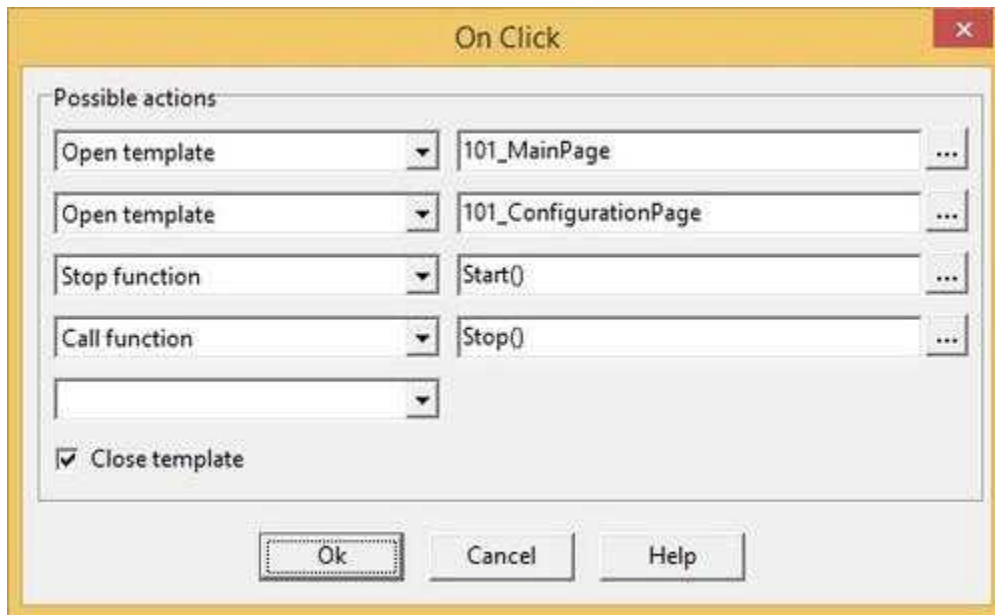
**Access group:** users groups enabled to operate on the object in runtime mode (enables "On Click" function). By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



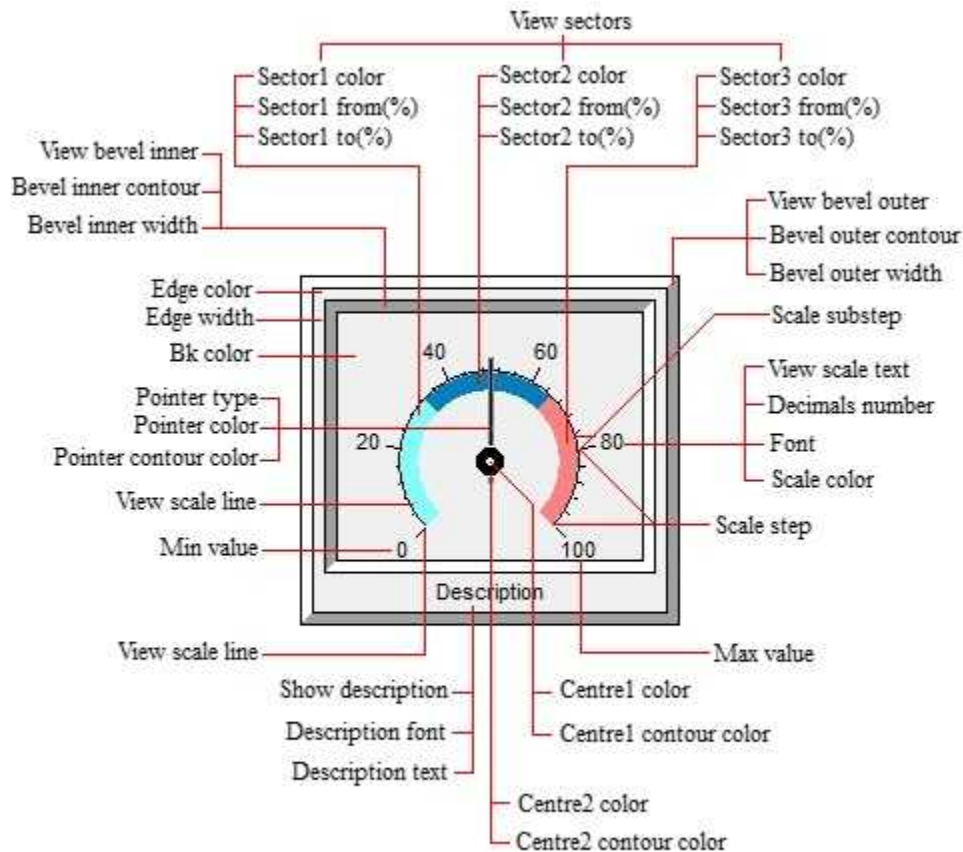
**Enable:** object is enabled (enables "On Click" and "On Double Click" functions) if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

**On Click:** operation to carry out when the user click on the object. Next figure show the window through which to specify the list of the operations to carry out:

- Call function: call the indicated function.
- Stop function: stop the indicated function (if this is on).
- Open template: open the indicated template.
- Apply changes: apply the changes in all the object brothers or child components that have the "Need apply" property set to "Yes" (the new values will be written in the gates).
- Undo changes: undo the changes in all the object brothers or child components that have the "Need apply" property set to "Yes".
- Close template: close the current template.



**On Double Click:** operation to carry out when the user double click on the object. Refer to "On Click" property to specify the list of the operations to carry out.



### Commands (CodeBuilder)

Following properties can be set also from language (see Code Builder help)

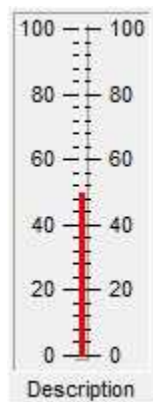
Property	Function
Min value	TObjSetPropertyReal(Id,"ScaleMin",...)
Max value	TObjSetPropertyReal(Id,"ScaleMax",...)
Decimals number	TObjSetPropertyInt(Id,"DecimalNumber",...)
Scale step	TObjSetPropertyInt(Id,"ScaleStep",...)
Scale substep	TObjSetPropertyInt(Id,"ScaleSubStep",...)
Sector 1 from (%)	TObjSetPropertyInt(Id,"Sector1From",...)
Sector 1 to (%)	TObjSetPropertyInt(Id,"Sector1To",...)
Sector 2 from (%)	TObjSetPropertyInt(Id,"Sector2From",...)
Sector 2 to (%)	TObjSetPropertyInt(Id,"Sector2To",...)
Sector 3 from (%)	TObjSetPropertyInt(Id,"Sector3From",...)
Sector 3 to (%)	TObjSetPropertyInt(Id,"Sector3To",...)
Description text	TObjSetPropertyString(Id,"DescriptionText",...)

Before setting all required properties, it is necessary to call **TObjBeginUpdate(Id)** function.  
After setting all required properties, it is necessary to call **TObjEndUpdate(Id)** function to update the object.

*Example:*

```
...
    TObjBeginUpdate(100);
    TObjSetPropertyReal(100,"ScaleMin",20);
    TObjSetPropertyReal(100,"ScaleMax",80);
    TObjEndUpdate(100);
...
```

## 8.39 ThermoMeter



**Properties (TemplateBuilder)**

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the object (in pixels).

**Top:** vertical position of the top left corner of the object (in pixels).

**Width:** width of the object (in pixels).

**Height:** height of the object (in pixels).

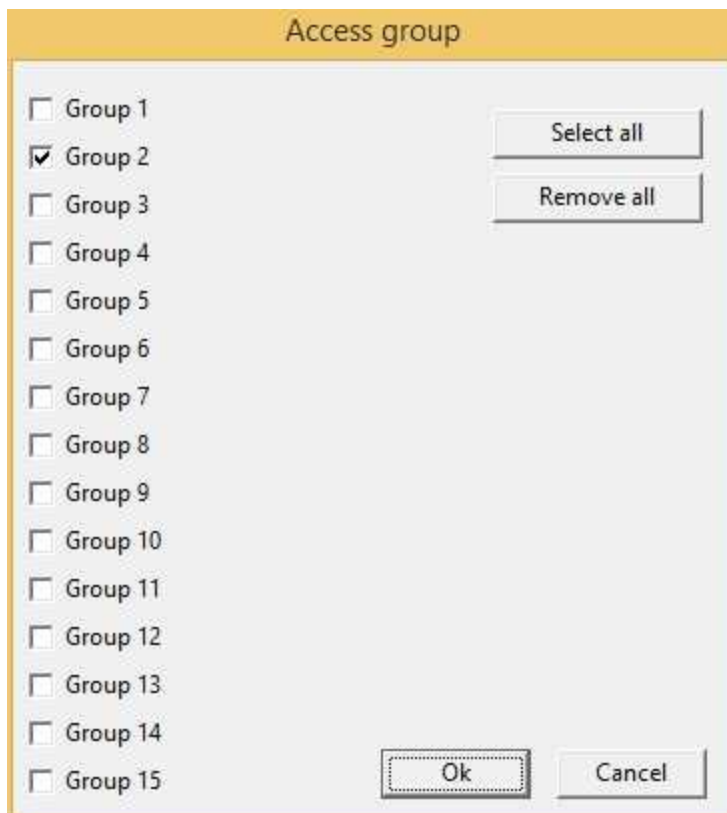
**Description:** description of the object (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Gate:** gate from which to read the value that will be displayed by the object during the supervision phase.

**Cursor:** shape that the cursor of the mouse must have when it goes over the object during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

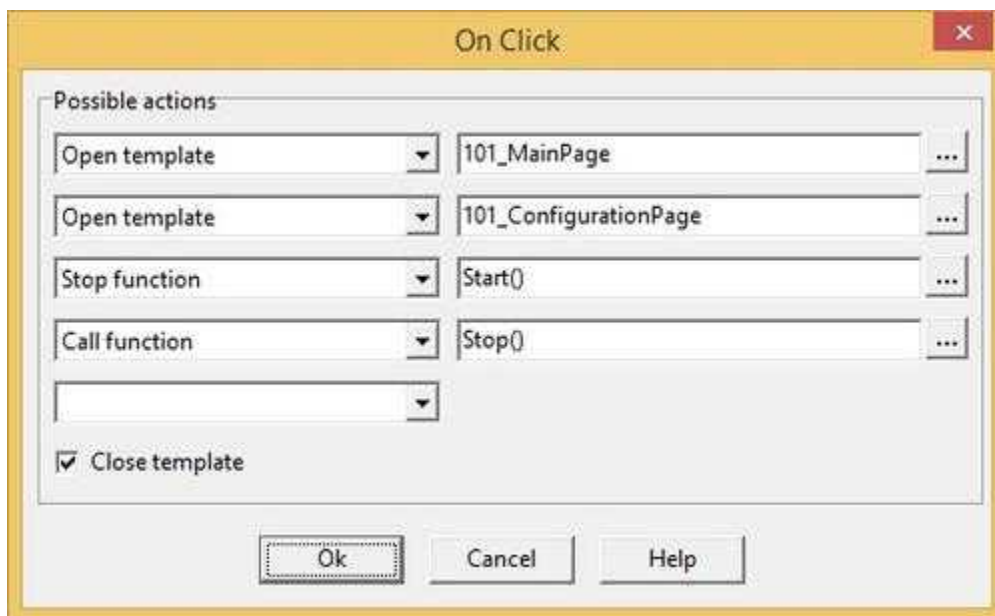
**Access group:** users groups enabled to operate on the object in runtime mode (enables "On Click" function). By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled (enables "On Click" and "On Double Click" functions) if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

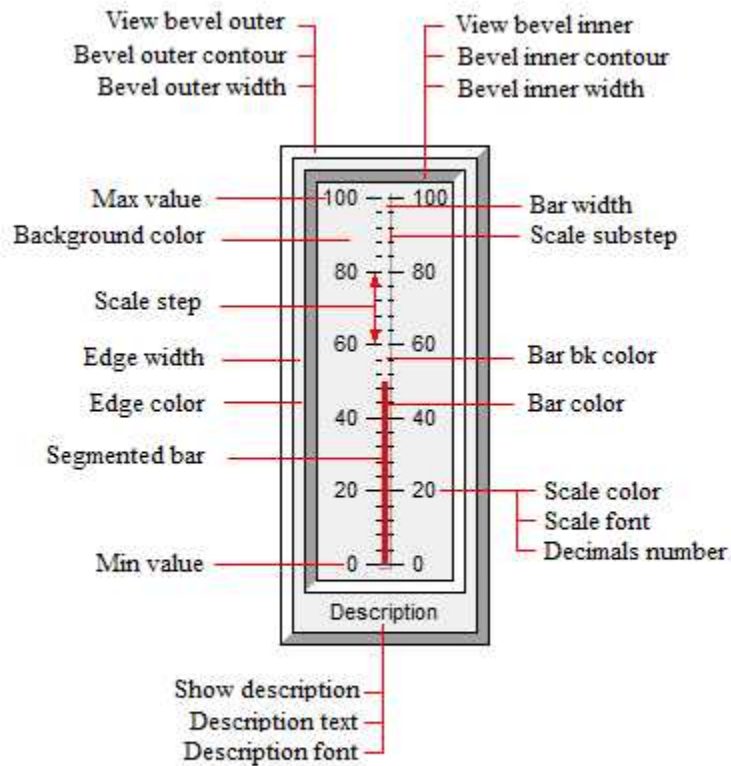
**On Click:** operation to carry out when the user click on the object. Next figure show the window through which to specify the list of the operations to carry out:

- Call function: call the indicated function.
- Stop function: stop the indicated function (if this is on).
- Open template: open the indicated template.
- Apply changes: apply the changes in all the object brothers or child components that have the "Need apply" property set to "Yes" (the new values will be written in the gates).
- Undo changes: undo the changes in all the object brothers or child components that have the "Need apply" property set to "Yes".
- Close template: close the current template.



**On Double Click:** operation to carry out when the user double click on the object. Refer to "On Click" property to specify the list of the operations to carry out.





### Commands (CodeBuilder)

Following properties can be set also from language (see Code Builder help)

Property	Function
Min value	TObjSetPropertyReal(Id,"ScaleMin",...)
Max value	TObjSetPropertyReal(Id,"ScaleMax",...)
Decimals number	TObjSetPropertyInt(Id,"DecimalNumber",...)
Scale step	TObjSetPropertyInt(Id,"ScaleStep",...)
Scale substep	TObjSetPropertyInt(Id,"ScaleSubStep",...)
Description text	TObjSetPropertyString(Id,"DescriptionText",...)

Before setting all required properties, it is necessary to call **TObjBeginUpdate(Id)** function.  
After setting all required properties, it is necessary to call **TObjEndUpdate(Id)** function to update the object.

*Example:*

```
...
TObjBeginUpdate(100);
TObjSetPropertyReal(100,"ScaleMin",20);
TObjSetPropertyReal(100,"ScaleMax",80);
```

```
TObjEndUpdate(100);
```

```
...
```

## 8.40 RockerSwitch



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the object (in pixels).

**Top:** vertical position of the top left corner of the object (in pixels).

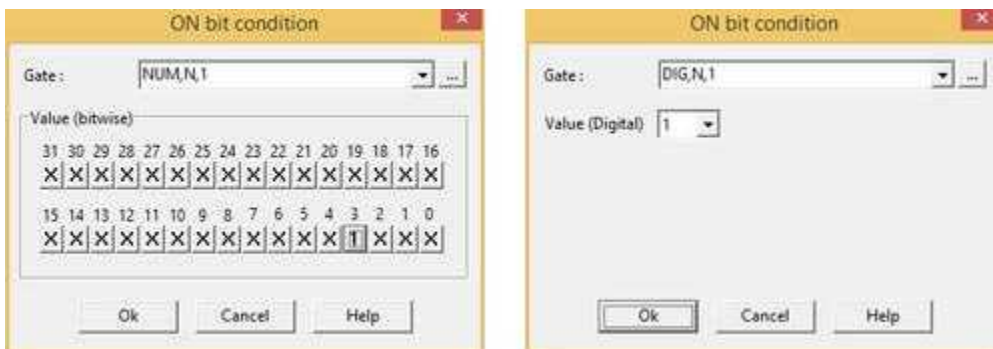
**Width:** width of the object (in pixels).

**Height:** height of the object (in pixels).

**Description:** description of the object (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Need apply:** it indicates whether the status change of the object will have to be confirmed from outside with a Button (Yes), or will be applied immediately (No).

**ON condition:** gate to be written on following the status change of the Switch, and the value to write. Press the button on the properties row to show one of the windows in figure below. If a numeric gate have been selected, then will be possible to set the ON value by selecting which bits must be set to 1, which will be set to 0 and which must not be considered. If a digital gate have been selected then must be specified the ON value of a single bit. In both cases, when the switch is set to OFF, the denied ON value will be written.



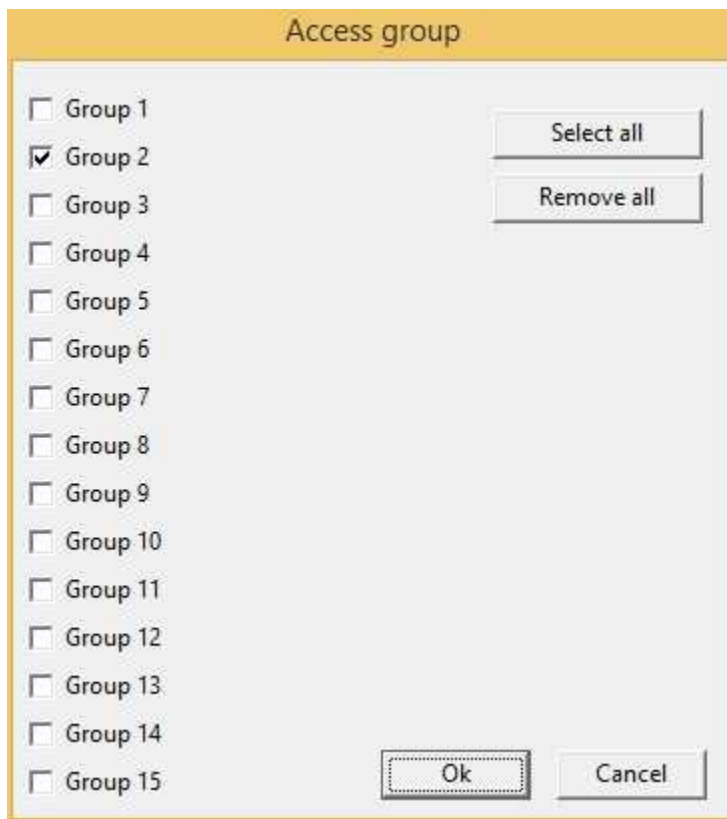
**Cursor:** shape that the cursor of the mouse must have when it goes over the object during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

**Tab num:** order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

**Help file name** : if in this field is specified a valid ".CHM" file name and it is present an HtmlHelp object in one of the templates currently showed on the screen, when the RockerSwitch is cliccked, the .CHM file will be showed in the HtmlHelp window. The path can be specified either in absolute (eg. "c:\Data\Help.chm") or in relative way (eg. "..\Data\Help.chm"): using the relative way, the base directory will be the application's templates directory. See HtmlHelp object for more details.

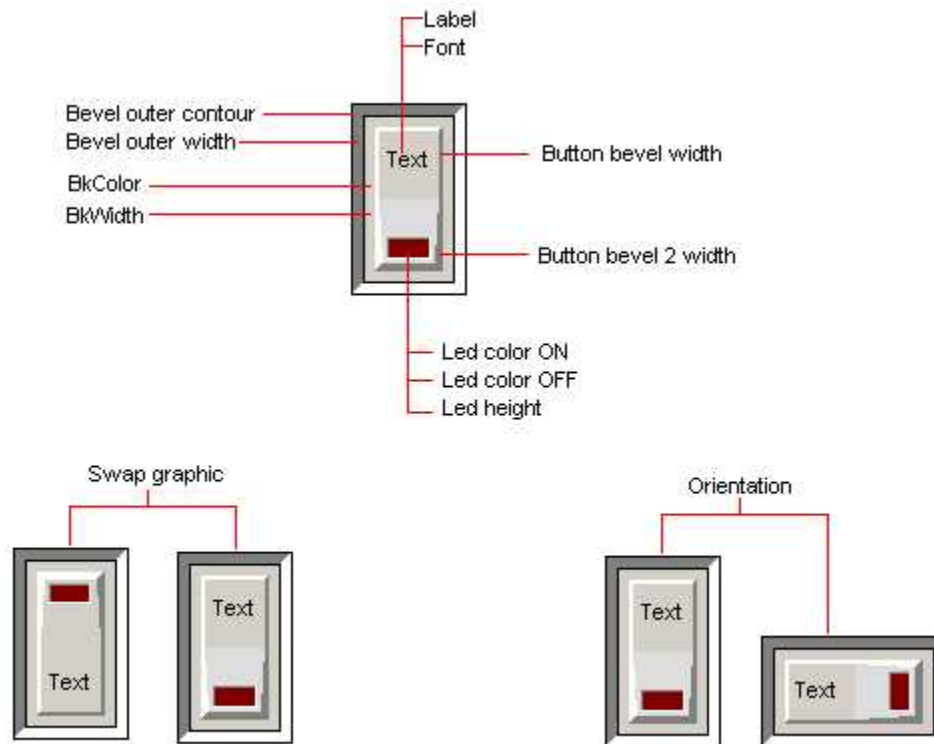
**Show**: object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group**: users groups enabled to operate on the object in runtime mode. By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable**: object is enabled if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

**Mode**: if switch mode is selected then the object changes its status on a mouse click. If Button mode is selected then the object changes its status to the ON status on left mouse button down and automatically restore the OFF status on left mouse button up.



## 8.41 ToggleSwitch



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the object (in pixels).

**Top:** vertical position of the top left corner of the object (in pixels).

**Width:** width of the object (in pixels).

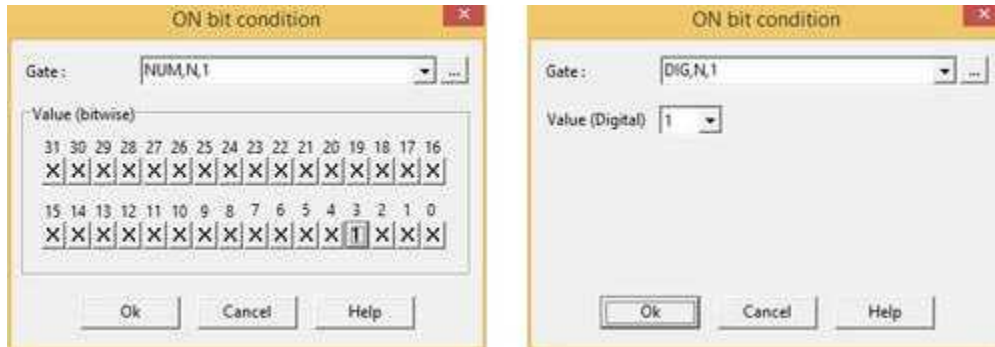
**Height:** height of the object (in pixels).

**Description:** description of the object (maximum 150 characters). The description will be shown when you press the right button of the component during the supervision phase. If in the description you want to begin a new paragraph, use the character '~'.

**Need apply:** it indicates whether the status change of the object will have to be confirmed from outside with a Button (Yes), or will be applied immediately (No).

**ON condition:** gate to be written on following the status change of the Switch, and the value to write. Press the button on the properties row to show one of the windows in figure below. If a numeric gate have been selected, then will be possible to set the ON value by selecting which bits must be set to 1,

which will be set to 0 and which must not be considered. If a digital gate has been selected then must be specified the ON value of a single bit. In both cases, when the switch is set to OFF, the denied ON value will be written.



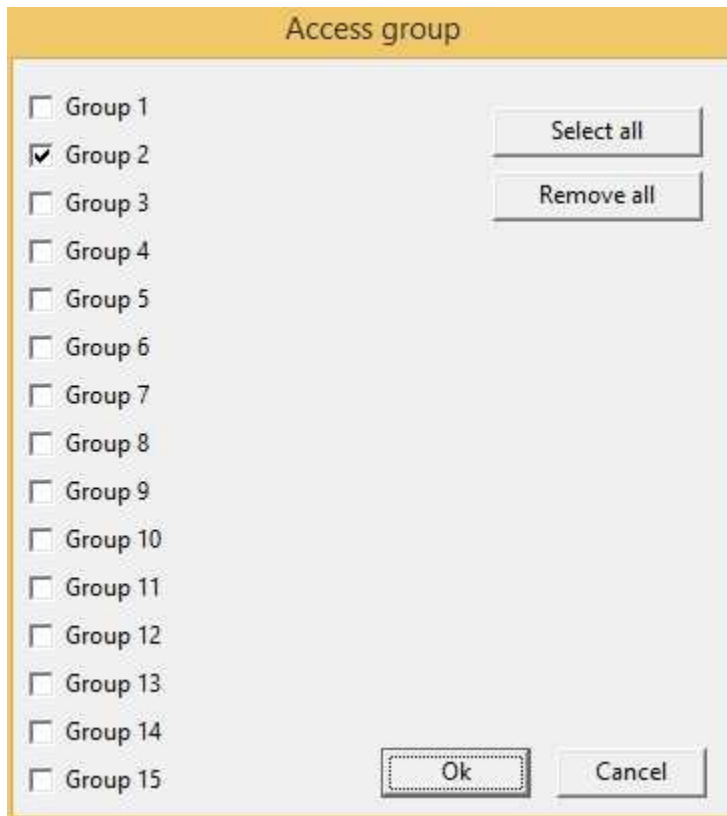
**Cursor:** shape that the cursor of the mouse must have when it goes over the object during the supervision phase. There are six options: Default, White, Red, Yellow, Click, Double Click (for details see the cursor table).

**Tab num:** order according to which the selection passes from component to component in RunTime: pressing the TAB key the active control will become the next in the order given by Tab num.

**Help file name :** if in this field is specified a valid ".CHM" file name and it is present an HtmlHelp object in one of the templates currently showed on the screen, when the ToggleSwitch is clicked, the .CHM file will be showed in the HtmlHelp window. The path can be specified either in absolute (eg. "c:\Data\Help.chm") or in relative way (eg. "..\Data\Help.chm"): using the relative way, the base directory will be the application's templates directory. See HtmlHelp object for more details.

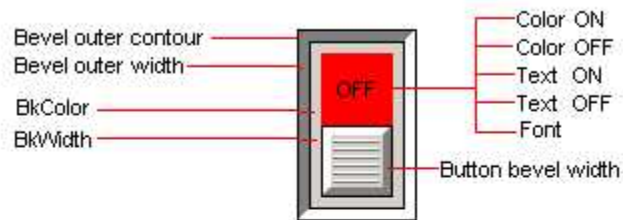
**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Access group:** users groups enabled to operate on the object in runtime mode. By pressing the button on the properties row, the dialog box in figure will be opened. In it, you can indicate the enabled groups.



**Enable:** object is enabled if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.

**Mode:** if switch mode is selected then the object changes its status on a mouse click. If Button mode is selected then the object changes its status to the ON status on left mouse button down and automatically restore the OFF status on left mouse button up.

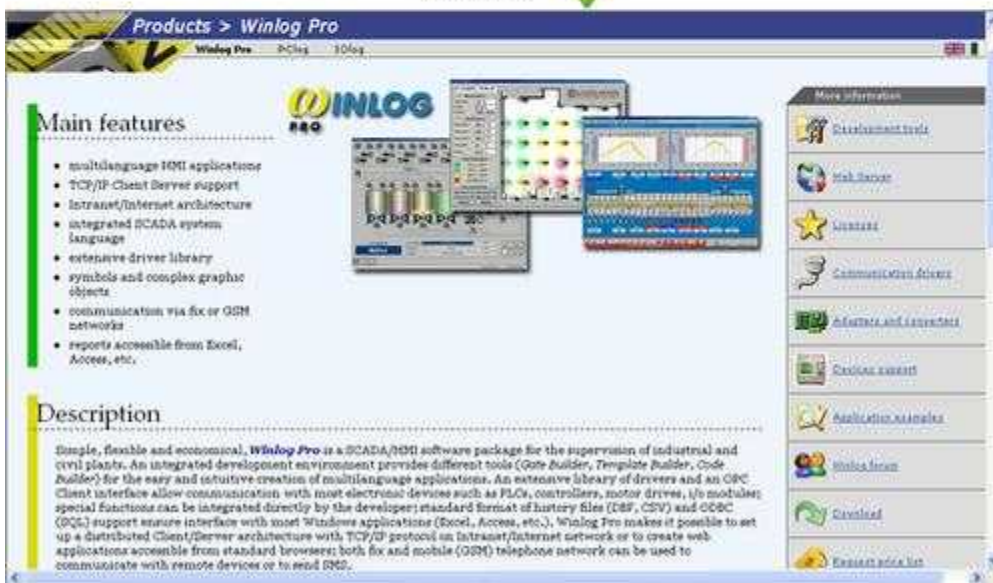


## 8.42 WebBrowser

WebBrowser allow you to view internet pages in a object of a template.  
This object requires Internet Explorer 4 and later.



Runtime



### Properties (TemplateBuilder)

**ID:** number that can be used for identifying the component from Code Builder.

**Left:** horizontal position of the top left corner of the WebBrowser window (in pixels).

**Top:** vertical position of the top left corner of the WebBrowser window (in pixels).

**Width:** width of the WebBrowser (in pixels).

**Height:** height of the WebBrowser (in pixels).

**Enable:** object is enabled if is true at least one of the conditions specified in this field. Object is always enabled if no conditions are specified.



**Show:** object is visible if it is true at least one of the conditions specified in this field. Object is always visible if no conditions are specified.

**Address :** URL (Uniform Resource Locator)

#### Commands (CodeBuilder)

Following properties can be set also from language (see Code Builder help)

Property	Function
Address	TObjSetPropertyString(Id,"Address",...)

Before setting all required properties,it is necessary to call **TObjBeginUpdate(Id)** function.  
After setting all required properties,it is necessary to call **TObjEndUpdate(Id)** function to update the object.

*Example:*

```
...
    TObjBeginUpdate(100);
    TObjSetPropertyString(100,"Address","C:\Documents\File.txt");
    TObjEndUpdate(100);
...
```

There is the possibility to send some commands to WebBrowser object by using "**TObjFunction(int ObjId, int Function)**" CodeBuilder instruction.

Parameters:

**int ObjId** : the number that identify the component (see ID property of Historical view object).

**int Function** : function to perform.

Function	Description
1	Page update