free Studio

UserInterface User Manual

Revision 1.3 - May 2011

UserInterface User Manual
Revision 1.3 - 2011-05-20
Published by Eliwell Controls S.r.l.
Via dell'Industria, 15 Z.I. Paludi
32010 Pieve d'Alpago (BL)

Contents

1.	**Overview**	1
1.1	Main elements	1
1.2	Run-time functionalities	4
1.3	Communicating with the target	4
2.	**Creating a simple UserInterface project**	5
2.1	Purpose of this chapter	5
2.2	Creating a new project	5
2.3	Inserting the first age in the project	6
2.3.1	Creating a new page	6
2.3.2	Editing the colors of the page	7
2.4	Inserting a secondary page	8
2.4.1	Creating a secondary page	8
2.4.2	Dimensioning and setting the secondary page	8
2.4.3	Viewing the title bar and the system button	9
2.4.4	Assigning a style to the window	10
2.4.5	Choosing the start window	11
2.5	Inserting static controls	11
2.5.1	Inserting a line	12
2.5.2	Inserting a rectangle in the page	12
2.6	Inserting static images	14
2.6.1	Importing a bitmap in the project	14
2.6.2	Associating an imported bitmap with an image control	15
2.7	Text strings	16
2.7.1	Inserting a text string	16
2.8	Data management in UserInterface	17
2.8.1	Declaring a local variable	17
2.8.2	Declaring a global variable	18
2.8.3	Importing the PLC variables in the UserInterface project	19
2.8.4	Inserting field parameters	20
2.9	Inserting edit box	21
2.9.1	Inserting an edit box in the page	21
2.9.2	Edit box and UserInterface local variable association	23
2.9.3	Edit box and UserInterface global variable association	25
2.9.4	Linking an edit box with a target (or system) variable	25
2.9.5	Linking an edit box with a PLC Application variable	26
2.9.6	Linking an edit box to a parameter	26
2.9.7	Linking an edit box to a variable by dragging and dropping	27

2.10 Inserting buttons 28
2.10.1 Inserting a led-button 28
2.10.2 Inserting a boolean variable command button 29
2.10.3 Inserting a button to open a child page 30
2.10.4 Inserting a button aimed at launching a procedure of the user 31
2.11 Visibility and updating of controls 33
2.11.1 The visibility property 33
2.11.2 The refresh property 35
2.12 Compiling and downloading the project on the target 35
2.12.1 Connecting to the target 36
2.12.2 Compiling pages for the target 36
2.12.3 Downloading and executing the compiled pages on the target 37
2.12.4 Simulation 37
3. UserInterface layout 39
3.1 Project window 39
3.2 Embedded editors 39
3.3 Properties window 40
3.4 Toolbars 40
3.5 The output window 40
3.6 Target variables and parameters 40
3.7 Table of keys-actions associations 41
4. HMI project in UserInterface 43
4.1 Project properties 43
4.1.1 General 43
4.1.2 System options 44
4.1.3 Language selection 44
4.1.4 Global periodic procedure 46
4.2 Frame set 46
4.3 Pages 47
4.3.1 Navigating between pages 47
4.3.2 Child Pages 47
4.3.3 Pop-up pages 48
4.3.4 Asynchronous messages 48
4.4 Controls 49
4.4.1 Static 49
4.4.2 Graphic element 49
4.4.3 Edit box 49
4.4.4 Text box 50

4.4.5	Image	50
4.4.6	Animation	50
4.4.7	Button	50
4.4.8	Chart	51
4.4.9	Trend	51
4.4.10	Progress bar	51
4.4.11	Custom control	51
4.5	Variables	52
4.5.1	Local variables	52
4.5.2	Global variables	52
4.5.3	Variables imported from PLC	53
4.5.4	System variables	53
4.6	Multiple pages management	53
4.6.1	Association of elements of a set	53
4.6.2	Navigation of the elements of a set	54
4.6.3	Pages numbering	54
4.7	Advanced operations on pages	55
4.7.1	Export/import of pages to/from files	55
4.7.2	Export/import procedures and variables	56
4.7.3	Copy/paste of pages in the project	56
4.7.4	Rename pages	56
4.7.5	Templates of page management	57
4.8	Events	60
4.8.1	Page or control events	60
4.8.2	Key pressure events	61
4.8.3	Events raised by software	61
4.8.4	Procedures that can be associated to events	62
4.8.5	Actions that can be associated to key pressure	62
4.9	Resources	63
4.9.1	Fonts	63
4.9.2	Bitmaps	64
4.9.3	Strings table	65
4.9.4	Enumeratives	65
4.9.5	Images lists	66
4.9.6	Sets	66
4.10	Automatic documentation	67
4.11	Managing projects	68
4.11.1	Selecting the target device	68
5.	**Appendix I: page properties and object properties**	71
5.1	Frame set	71
5.1.1	Properties	71

5.2 Child page 72
5.2.1 Properties 72
5.2.2 Events 72
5.3 Pop-up page 73
5.3.1 Properties 73
5.3.2 Events 74
5.4 Static 74
5.4.1 Properties 74
5.4.2 Events 75
5.5 Line 75
5.5.1 Properties 75
5.6 Rectangle 76
5.6.1 Properties 76
5.7 Edit box 76
5.7.1 Properties 76
5.7.2 Format specification - printf 78
5.7.3 Events 79
5.8 Text box 79
5.8.1 Properties 79
5.8.2 Events 81
5.9 Image 81
5.9.1 Properties 81
5.10 Animation 82
5.10.1 Properties 82
5.10.2 Events 82
5.11 Button 83
5.11.1 Properties 83
5.11.2 Events 84
5.12 Progress bar 85
5.12.1 Properties 85
5.12.2 Events 86
5.13 Custom control 86
5.13.1 Properties 86
5.13.2 Events 87
5.14 Chart 87
5.14.1 Properties 87
5.14.2 Events 89
5.15 Trend 89
5.15.1 Properties 89

5.15.2 Events 92

6. APPENDIX II: FILE FOR TARGET DESCRIPTION 93

6.1 Target properties 93

6.1.1 Description 93

6.2 Object version 94

6.3 System enumeratives 94

6.3.1 Descriptions 94

6.3.2 Example 96

7. Appendix III: Description of parameter file 99

8. Appendix IV: elements of HMI runtime 101

8.1 Functions 101

8.1.1 System functions: hardware and operating system 101

8.1.2 Function for managing project resources and common properties 102

8.1.3 Functions for operating with pages 104

8.1.4 Function for objects 107

8.1.5 Drawing functions 108

8.1.6 Functions for text 111

8.1.7 Functions for parameter access 112

8.1.8 Functions for events 113

8.2 Function Blocks 114

1. OVERVIEW

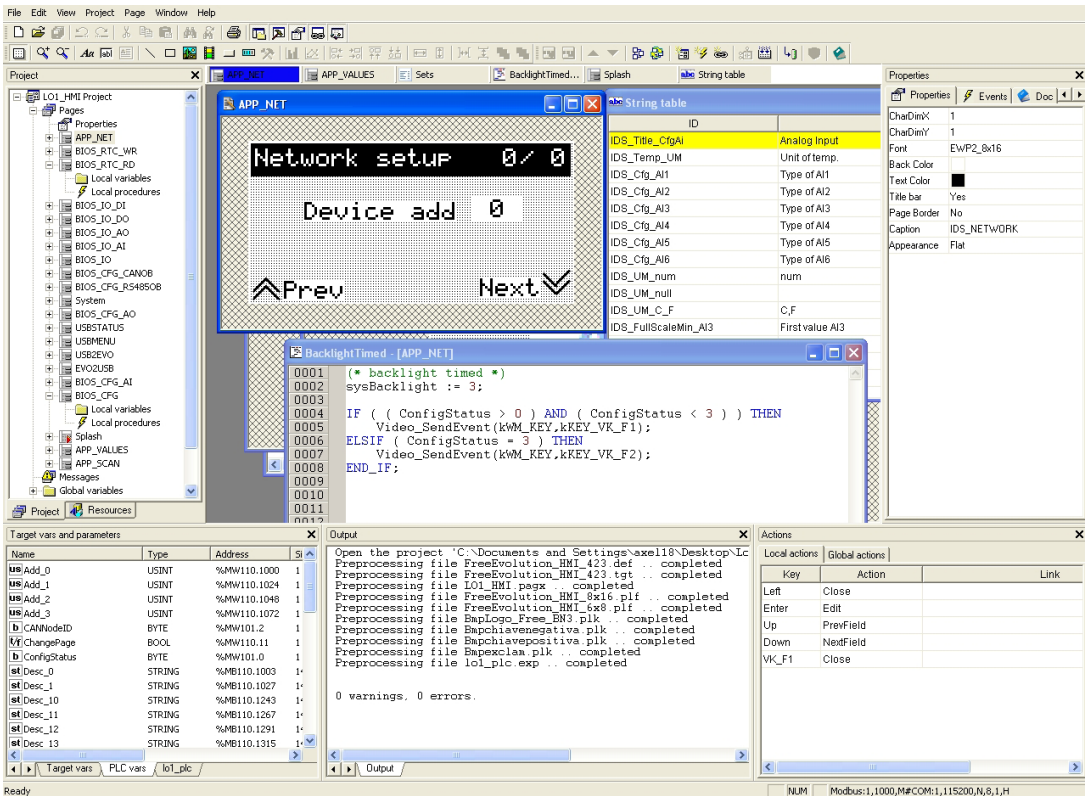
UserInterface is a software application that allows the developer to create user interfaces for embedded systems based on HMI runtime.

UserInterface is an easy to learn and use software, which allows the user to implement graphical interfaces in a visual way. The realized pages are viewed in UserInterface as they will appear on the final target.

Thanks to its multi-pages structure, UserInterface can support HMI (Human Machine Interface) applications with an arbitrary number of pages.

It is equipped with a considerable number of tools to realize even complex applications and it interfaces directly to the PLC IEC1131 Application compiler for managing the variables which are defined in the target PLC application.

The following paragraphs show you the main features of this product.



1.1 MAIN ELEMENTS

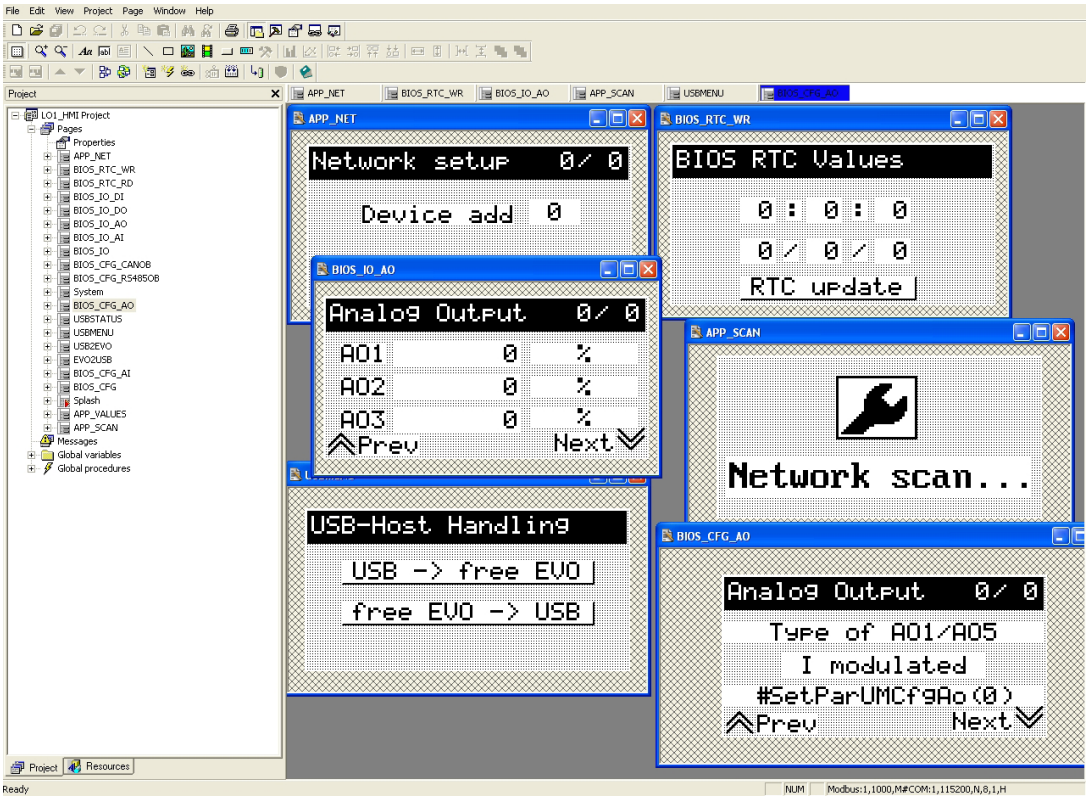
Set of controls

Each page may contain an arbitrary number of defined graphic controls. There are two classes of graphic controls:

- Static controls: drawing tools such as lines, rectangles, and figures.
- Dynamic controls: multilayered objects, which enable data and images display and user interaction (strings, editboxes, textboxes, buttons, progress, charts and trends, custom controls).

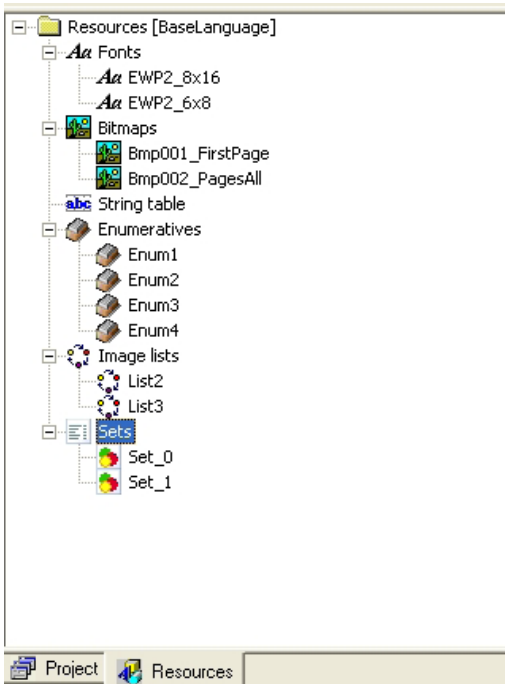
UserInterface is an open system, allowing the implementation of custom controls which may be included in the target system.

Multi-pages structure



UserInterface supports the definition of an arbitrary number of pages (full-screen or pop-up). Each page may contain links to other pages, so that the whole project takes a tree structure.

Resources management

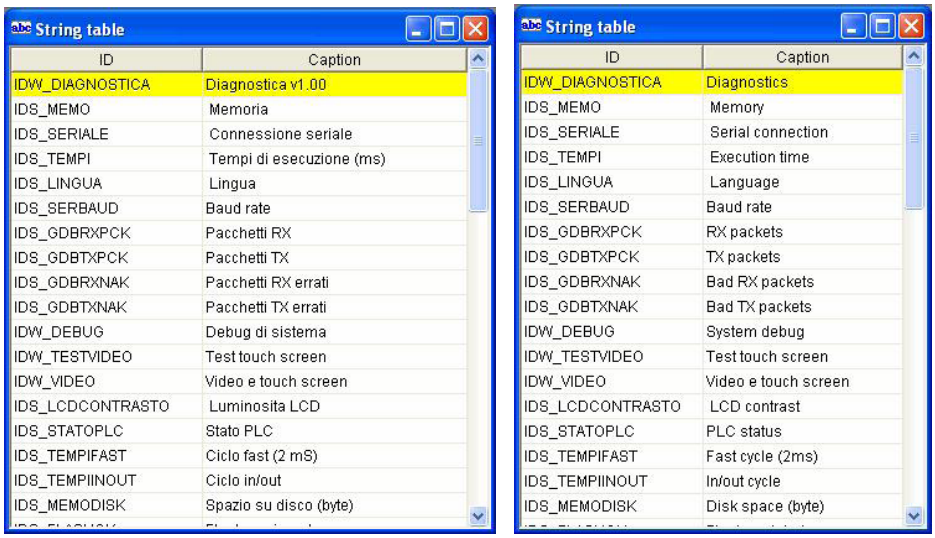


The controls' properties in the page are not statically defined in the project code, but they can be managed separately as resources.

Resources include fonts for characters display, images, string table, enumerated data types, and elements sets.

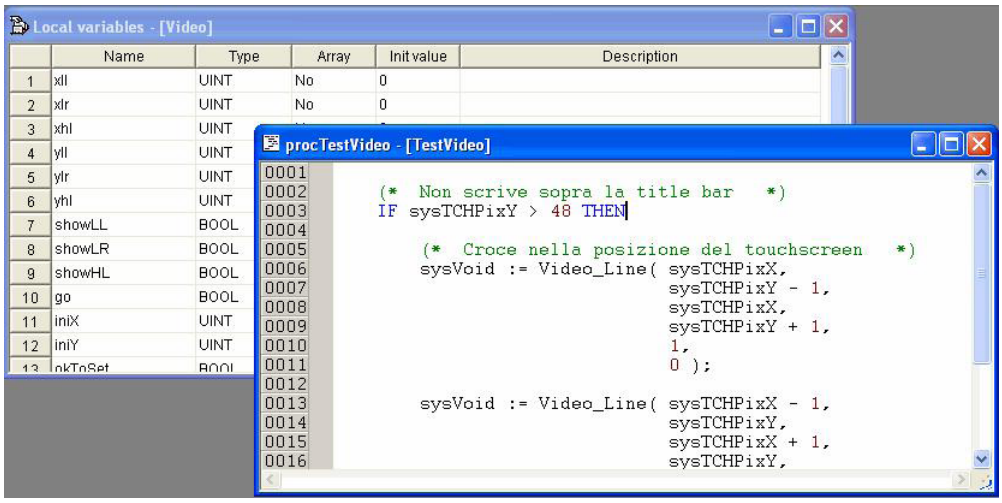
Specifically regarding the images, UserInterface allows to import bitmap files directly from the Windows-formatted file (*.bmp*, *.gif*, *.emf*, *.jpg*, *.ico* etc.).

Languages management



Strings and enumerated data types are structured as to ease the multilingual device; moreover UserInterface provides a function to export/import the above mentioned elements to/from a text file, in order to simplify the translation from a language to another.

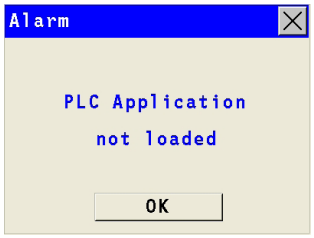
Variables and procedures



UserInterface enables the implementation of procedures which may be as complex as you want in the ST language. Through these procedures, the user can interact with the UserInterface application, the PLC application or the target system variables to customize the interface's behaviour or the whole CNC.

1.2 RUN-TIME FUNCTIONALITIES

Asynchronous messages management(1)

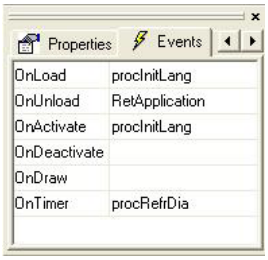


UserInterface supports the issue of asynchronous messages whatever their complexity. You can entirely customize the issue messages management by typing a simple ST procedure.

Multilingual support

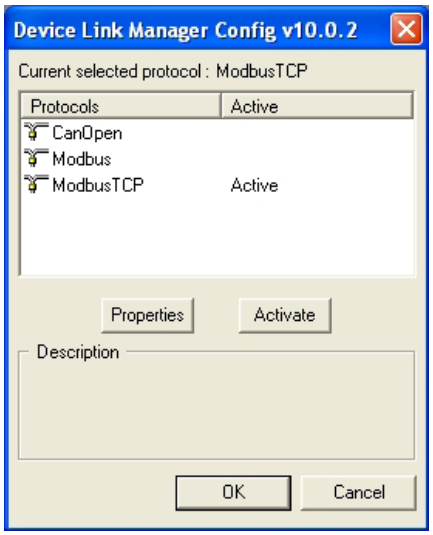
UserInterface allows you to change strings, resources, and enumerations language without recompiling nor reloading the application.

Events management



UserInterface applications are structured in events; the user may seize the available events and manage them through ST-coded procedures.

1.3 COMMUNICATING WITH THE TARGET



You can establish the communication with the target device through the PC communication drivers, thus using one of the available custom protocols (which can be easily implemented thanks to the modular structure of the communication system).

2. CREATING A SIMPLE USERINTERFACE PROJECT

2.1 PURPOSE OF THIS CHAPTER

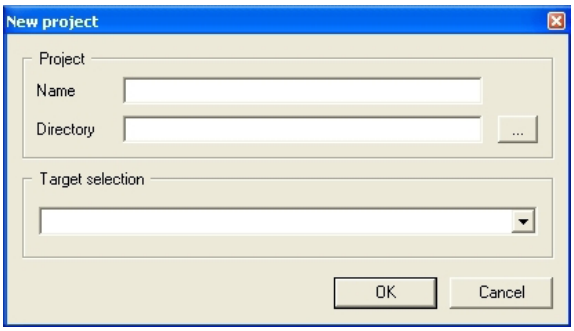
This chapter aims to lead the user to realize a simple HMI project with UserInterface, through a sequence of easy steps.

Here below you can find the list of this chapter's topics.

- Creating a new project: starting at zero the realization of a HMI project.
- Inserting the first page in the project.
- Inserting a secondary page.
- Inserting static controls: how to insert simple objects (lines, rectangles, etc.) in a page.
- Inserting static images: how to insert an image in a page, starting at a *.bmp* file.
- Inserting strings: how to insert a text label.
- Inserting edit boxes: how to access the data of the system and the control PLC, how to declare new variables, how to insert text frames to view/edit these data.
- Inserting buttons: learning to use an essential control for the interaction between the user and the system.
- Compiling and downloading the project.

2.2 CREATING A NEW PROJECT

Launch UserInterface, then select the *New Project* command from the *File* menu. The following dialog box appears.



Type the name you want to assign to the project in the *Name* field, and in the *Directory* field specify the directory where you want to create the project folder.

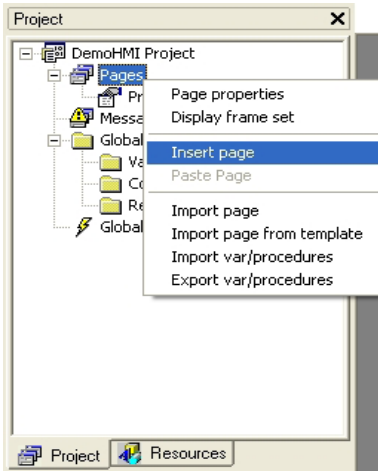
Select the target which will execute the HMI from the *Target selection* menu. The contents of this menu can be customized: if the desired target does not appear in the list, refer to your hardware provider.

Confirm your choice by pressing *OK*. UserInterface automatically creates the folder *I:\Demo manuale\Demo HMI* as specified in *Directory*.

2.3 INSERTING THE FIRST AGE IN THE PROJECT

2.3.1 CREATING A NEW PAGE

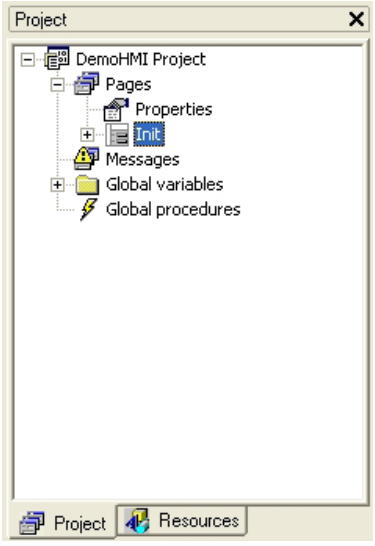
To insert a new page in the project, right-click on the *Pages* item of the project tree.



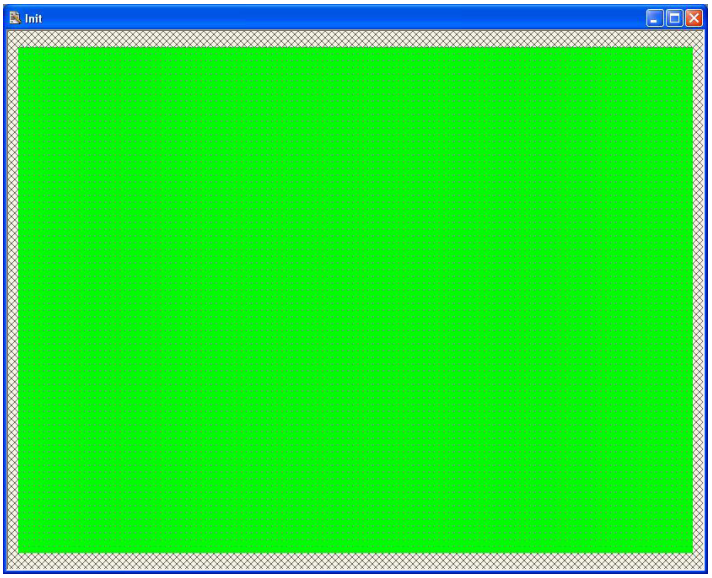
Select the *Insert page* option from the menu which has just shown up. This causes a dialog box to appear where you have to specify the page name and whether the page is a pop-up one or not.

If you do not select the *Pop-up* property when creating it, the page is called *Child Page*. Its main feature is that it fits the whole video area. Consequently the user cannot define position and size of a child page because they are automatically set depending on the video area and on an eventual frame set (see 4.2).

Choose to create a child page and call it *Init*: type the name *Init* in the apposite field and press *OK* to confirm your choice. A new node appears in the pages folder of the project tree.



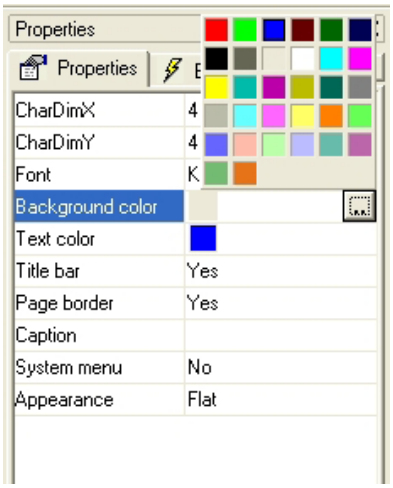
Double-click on the *Init* item to open the document with this page preview(1), which is blank at the moment.



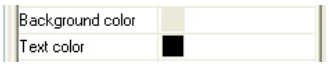
2.3.2 EDITING THE COLORS OF THE PAGE

You can edit the background color of the page and the foreground default text color through the page properties: double-click in the *Background Color* field. A little button appears.

Pressing it, the colors palette appears(1). Then you can select the desired color.



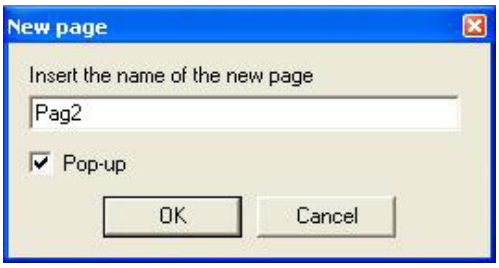
Choose grey as background color and black as default text color(1).



2.4 INSERTING A SECONDARY PAGE

2.4.1 CREATING A SECONDARY PAGE

Let us assume that you want to create a secondary page: right-click on the *Pages* item of the project tree and choose the *Insert page* option from the contextual menu. Type the name *Pag2* in the dialog box which appears and select the pop-up property.



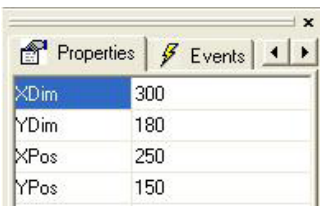
Consequently a new item appears in the *Pages* folder of the project tree.

2.4.2 DIMENSIONING AND SETTING THE SECONDARY PAGE

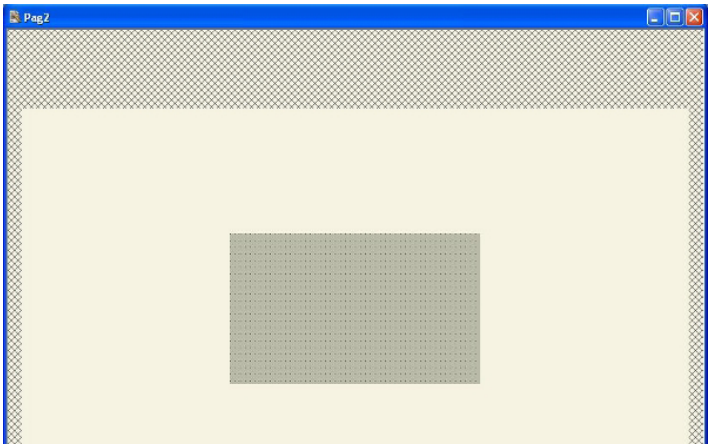
Note that the icon of the *Init* page different from the new *Pag2* one. In fact, the last one has been created as pop-up page, whereas the first one has been created as child page.

Pop-up pages are not subjected to any restriction from the frame set (see 4.2): their dimensions and positions can be chosen by the user.

Assign to the secondary window the dimensions 300x180 pixel and set it (x, y) = (250, 150) because these are the top left-hand corner's coordinates of the window. Double-click on the *Pag2* item of the project tree. In this way you open the corresponding document. Assign dimensions and position.



After editing the colors, too, the new window will look like the picture below(1).



The grey area in the centre is the active area of the *Pag2* page, whereas the clearer area which surrounds it represents the video area of the target system. In this way you obtain a clear vision of the new page placement.

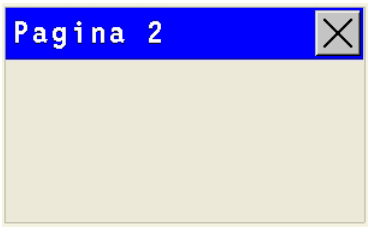
2.4.3 VIEWING THE TITLE BAR AND THE SYSTEM BUTTON

UserInterface enables the automatic creation of a title bar (*Title bar* properties = *Yes*) and of a button to close the page (*System menu* properties = *Yes*), besides the print of a text string as title (*Caption* properties).

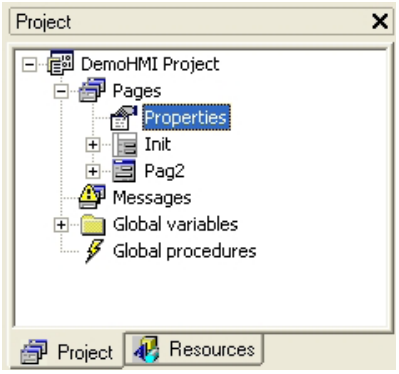
Let us assume that you want to activate the title bar and the close button, and to print the *Pagina 2* string as title.

Title bar	Yes
Page border	Yes
Caption	Pagina 2
System menu	No
Appearance	Flat

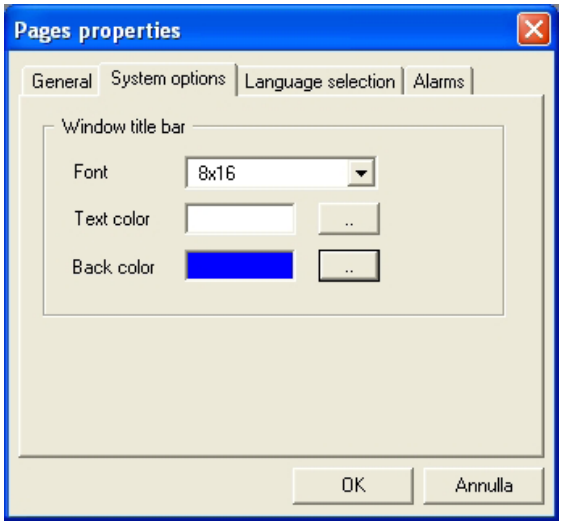
Then the secondary page looks like the following picture(1).



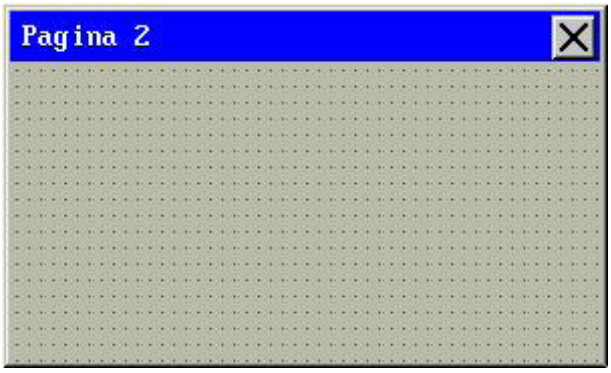
The text and the background color and the used font are the same for all the pages of the project, so you will not find them in this specific page properties. In order to customize these features, double-click on the *Properties* item of the project tree.



A multi-tabs window opens. In *System options* assign the font (in this case 8x16), the text color and the background color (in this case respectively white and blue).



Then the secondary page looks like the following figure(1).

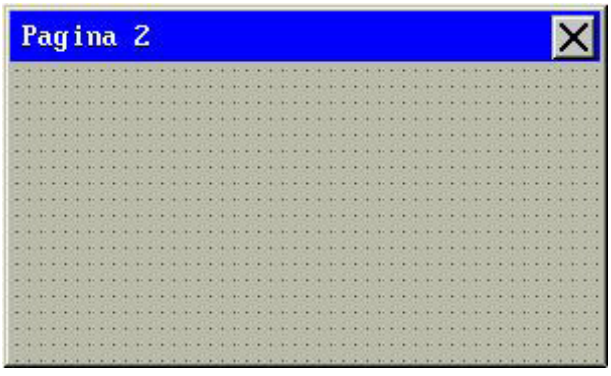


2.4.4 ASSIGNING A STYLE TO THE WINDOW

UserInterface supports three styles for the windows, which you can select through the *Appearance* property: *Flat* (the default style when you create a window), *Sunken* and *Raised*.

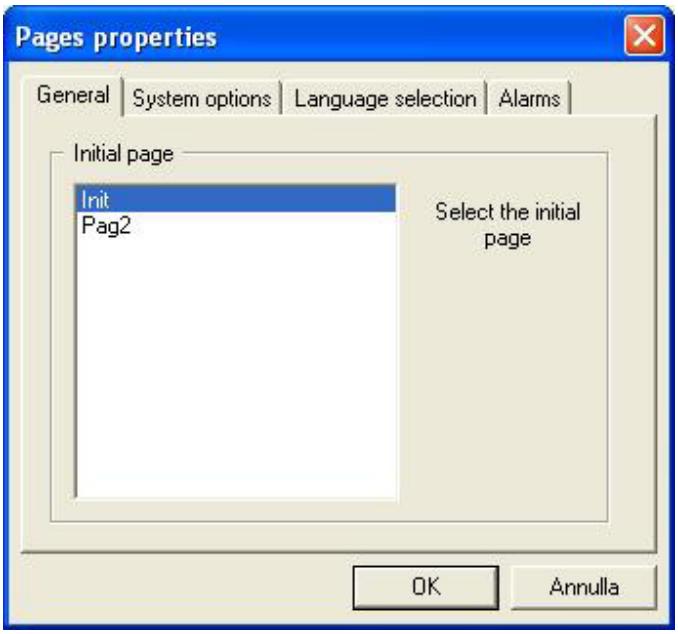
Choose the last one.

The window looks like the picture below(1).



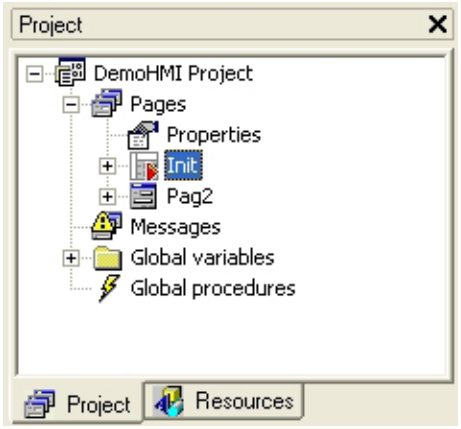
2.4.5 CHOOSING THE START WINDOW

The user has to indicate the start window of the whole HMI project. The start window will open at the HMI application start. If the project consists in one single page, the system will take this one as start page. You can indicate the start page in the project properties window, which you can open by double-clicking on the *Properties* item of the project tree. The *General* window is used for this purpose.



In order to indicate the start page, select the desired one from the list. Then confirm your choice by clicking *OK*.

The start page is marked in the project tree by a red triangle.



2.5 INSERTING STATIC CONTROLS

The two pages which you have just created are blank yet. Go back to the first page (*Init*) and start inserting some controls.

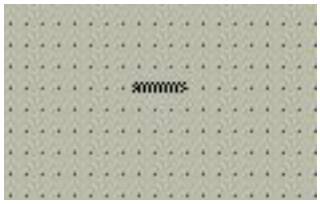
Static controls are objects which are drawn once, when opening the page, and they do not change until the page is active.

2.5.1 INSERTING A LINE

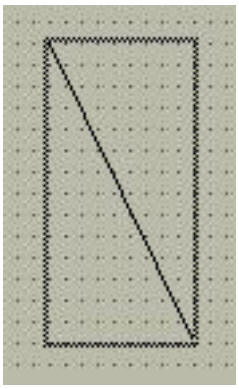
Insert a line by clicking the corresponding button in the *Page toolbar*.

Move the mouse to the active area of the page. A cross + appears. The object will be inserted in the grid near to the mouse cursor.

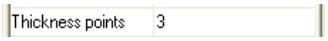
Confirm the insertion point by left-clicking. A new *Line* control appears(1). It has a default size and horizontal alignment.



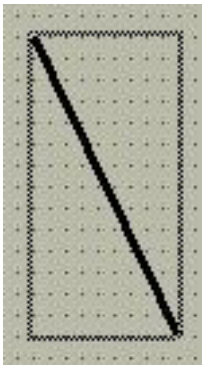
You can resize it by dragging one of the two ends of the line(1).



You can edit the line thickness through the *Thickness points* property of the control. For example, assign a 3 pixel thickness.



In the page preview you can see how the line looks like(1).

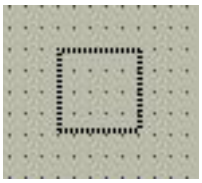


2.5.2 INSERTING A RECTANGLE IN THE PAGE

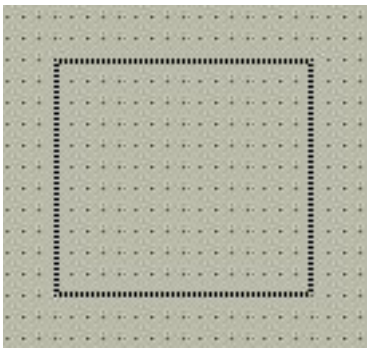
Press the corresponding button in the *Page toolbar*.

Move the mouse to the active area of the page. A cross + appears. The object will be inserted in the grid near to the mouse cursor.

Confirm the insertion point by left-clicking. A new *Rectangle* control appears(1). It has a default size.



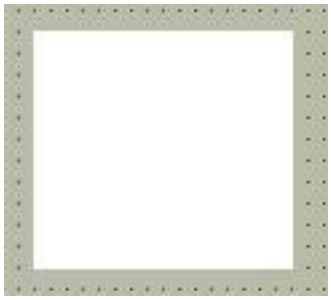
You can edit both the dimensions dragging one of the rectangle vertexes, or one dimension at a time dragging one of the rectangle's sides(1).



You can customize the border and the background color and the transparency through the control properties. For example, make the rectangle white and opaque with white border and thickness set to 1.

Border points	1
Border color	
Background color	
Transparent	TRUE

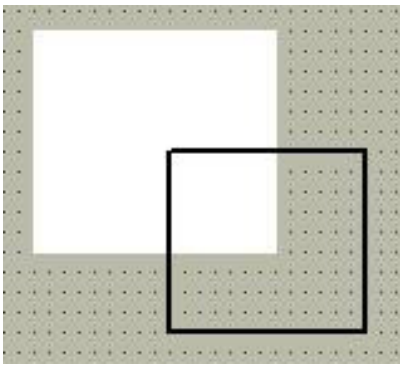
In the page preview you can see how the rectangle looks like(1).



Now superimpose another rectangle to the first one. Let us assume that you want the new rectangle to be transparent with black borders, and thickness set to 2.

Border points	2
Border color	
Background color	
Transparent	TRUE

In the page preview you will see the following image(1).



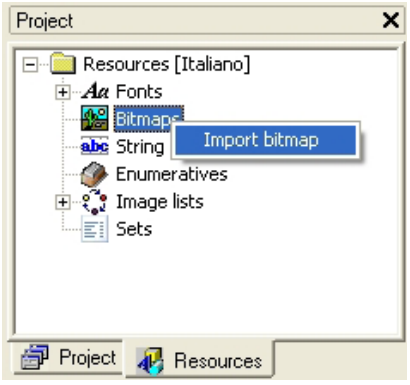
2.6 INSERTING STATIC IMAGES

The following paragraph shows you how to insert static images in the page. Static images are different from animations (images which may change dynamically, even though they have fixed position and dimensions) and from floating images (images which move in the page).

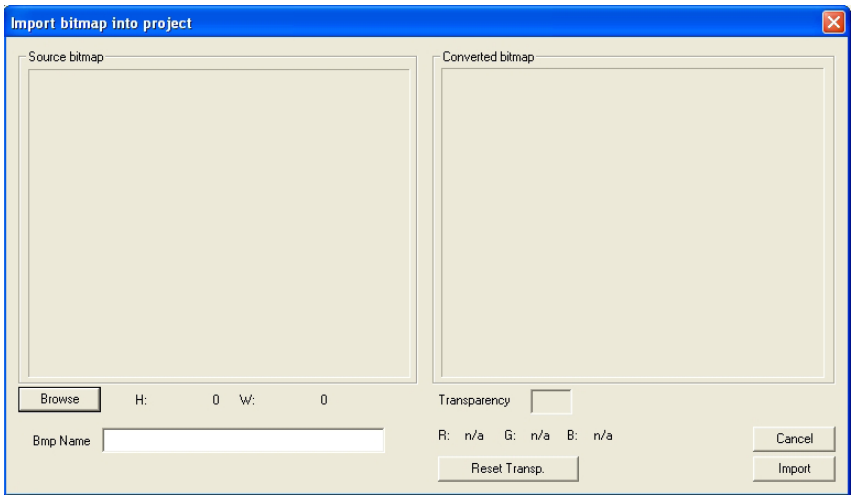
2.6.1 IMPORTING A BITMAP IN THE PROJECT

Image that has to be visualized must be available on PC as a basic Windows image file (*.bmp*, *.dib*, *.emf*, *.gif*, *.ico*, *.jpg*, *.wmf* ...). If this pre-condition holds, you can start the importing procedure.

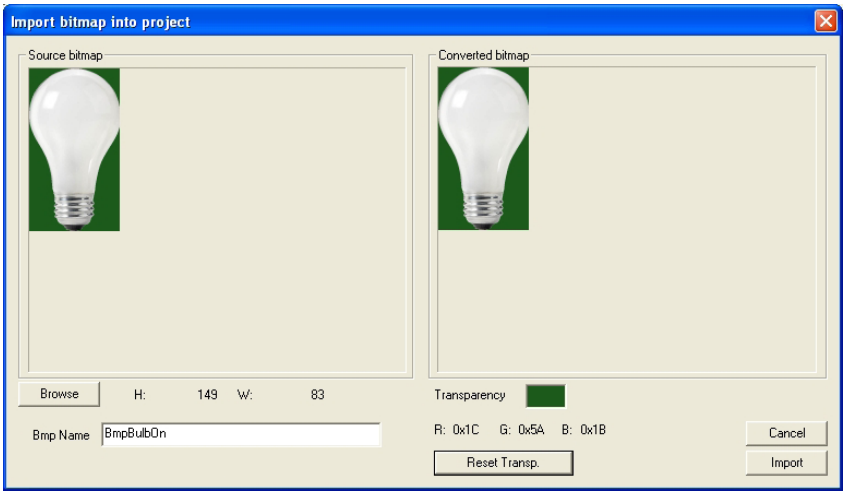
Right-click the *Bitmaps* item in the resources tree and select the *Import bitmap* command in the contextual menu which appears.



A dialog window opens.



Pressing the *Browse* button, you can navigate in the computer resources and select the source file. In this case, the source file is *BulbOn.jpg*, which represents a lighted bulb(1).



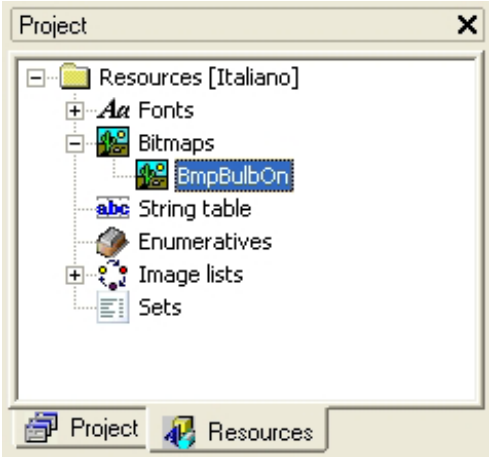
In the *Bitmap Name* field, you can assign the bitmap name which will appear in the resources tree; the default name is the file name without extension and preceded by the *Bmp* prefix.

The *Transparency color* field lets you specify the transparency color, that is a color which will not be really drawn but will let the elements appear through the bitmap background.

You can customize the transparency color by taking the desired one with the mouse from the *Converted bitmap* window.

RGB indicate the transparency color components. If the values are *n/a* it means that no transparency color has been selected. The *Reset Transp.* button lets to cancel the last selected transparency color.

At last you can confirm the operation by clicking the *Import* button. The imported bitmap appears as a new item in the resources tree.

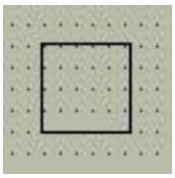


2.6.2 ASSOCIATING AN IMPORTED BITMAP WITH AN IMAGE CONTROL

The control which is aimed to display the static images is called *Image*: press the corresponding button in the *Page toolbar*.

Move the mouse to the active area of the page. A cross + appears. The object will be inserted in the grid near to the mouse cursor.

Confirm the insertion point by left-clicking. A new blank frame appears(1).



Trough the *Bitmap* property specify the image which this *Image* control must display.

Choose the desired bitmap from the list; in this case, you can see and select the only bitmap which you have imported: *BmpBulbOn*.

The control changes its size to be compatible with the assigned bitmap measures. The image in the page preview looks like the following picture(1).

2.7 TEXT STRINGS

Text strings are not part of static controls because they have some properties which let them change in a page through time. Visibility, selection, and refresh may be assigned to variables, which may change their value at any time.

2.7.1 INSERTING A TEXT STRING

Click the corresponding button in the *Page toolbar*.

Move the mouse to the active area of the page. A cross + appears. The object will be inserted in the grid near to the mouse cursor.

Confirm the insertion point by left clicking. A new *Static* (that is string) control with the default text *str* appears(1).

You can edit the contents of the string through the *Text* property of the control. For example, *Text string*.

The page preview looks like the image below.

This is the basic use of the string. Alternatively you can assign strings by taking them from the resources (see 4.9.3).

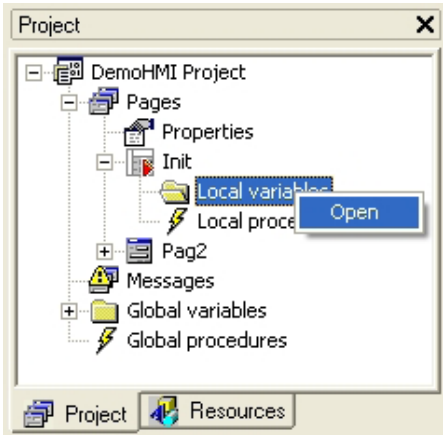
2.8 DATA MANAGEMENT IN USERINTERFACE

This paragraph shows you the variables management in UserInterface. It is possible to distinguish the data in local variables (visible in the page scope only) and global variables (visible from every page). For some controls it is possible to use parameters and sets.

2.8.1 DECLARING A LOCAL VARIABLE

First of all declare a local variable, which you can use just in the specific page where the declaration takes place.

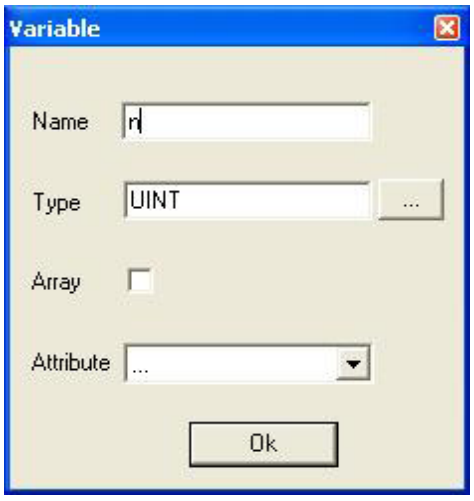
In the pages tree, under the *Init* page item, right-click on the *Local variables* item and select *Open* in the contextual menu which appears.



The local variables editor window opens. It is blank at present.

Click the *New record* button in the *Project toolbar.*

A dialog window opens requesting to specify the new variable's basic features. We can declare *n* as a new 16 bit unsigned integer variable.



Confirm the operation by clicking *Ok*. The new corresponding record is added to the variables editor.

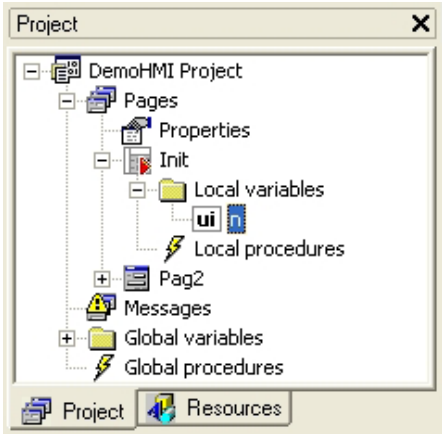
	Name	Type	Array	Init value	Description
1	n	UINT	No	0	

You can change this new variable's features editing the fields of the record which you have just created. For example, you may assign an initial value different from null and a comment.

	Name	Type	Array	Init value	Description
1	n	UINT	No	100	Variabile contatore

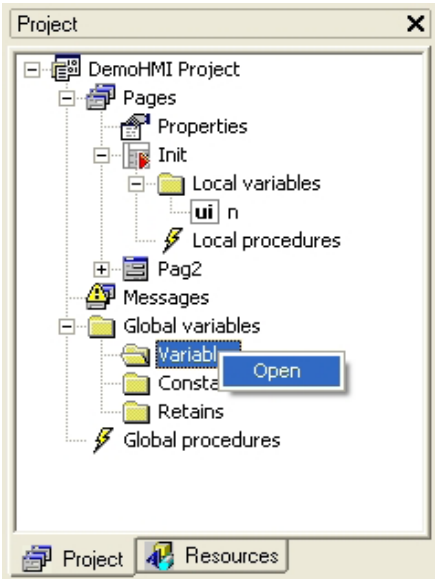
When you save the project by clicking the apposite button

or when you close the variables editor, UserInterface adds a new item in the pages tree. It corresponds to the local variable which you have just declared.



2.8.2 DECLARING A GLOBAL VARIABLE

Let us assume that you want to declare a floating point global variable *t*: right-click on the *Variables* item under the *Global variables* node of the resources tree and select the *Open* command in the contextual menu which appears.



Follow the steps as shown in paragraph 2.8.1, until the new global variable appears as a new item in the pages tree.

2.8.3 IMPORTING THE PLC VARIABLES IN THE USERINTERFACE PROJECT

Usually an HMI project is not a stand-alone one, but is an interface for a PLC. More precisely, if the PLC project has been carried out with Application, you can easily publish some variables to UserInterface.

A variable of the Application project can be exported to UserInterface if it has been allocated on a datablock (it is not an automatic variable). If this pre-condition holds, when compiling the PLC, the program automatically creates an *.exp* file, which contains a list of the exported variables with their location in the datablocks, which the UserInterface program can work out.

In order to import in UserInterface the variables which have been exported from the PLC Application project, you have to select the *Link PLC variables file...* from the *Project* menu.

A window opens and lets you select the file which contains the exported variables.

If you confirm to include the *.exp* file in the UserInterface project, a new table called *PLC vars* appears in the libraries window. It contains the list of the exported variables.

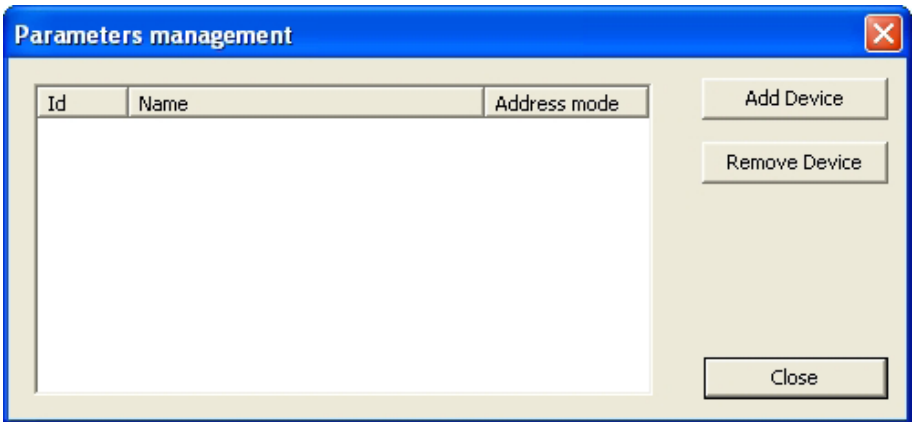
When you need to update the list of the exported variables, if the *.exp* file has not been moved to another directory, it is not necessary to repeat the above mentioned procedure. It is enough to launch the *Refresh PLC variables* command from the *Project* menu.

2.8.4 INSERTING FIELD PARAMETERS

Target system usually has internal variables and is connected on a fieldbus, so it needs to show some variables of the different devices which are connected on the net.

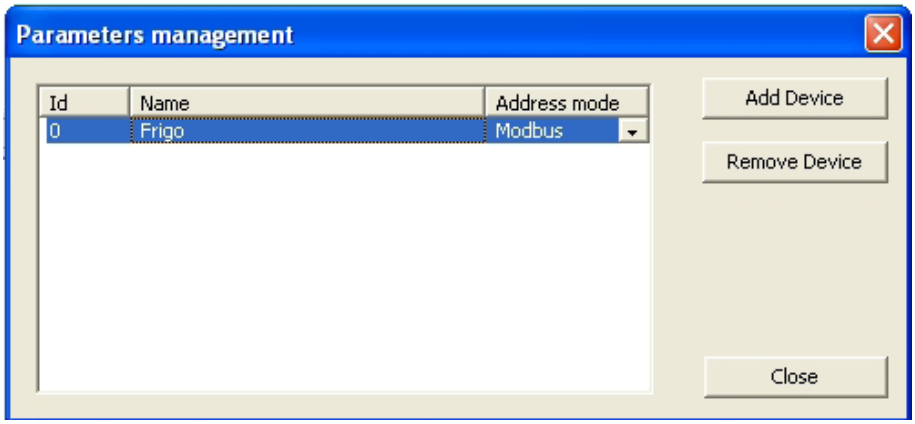
For this reason, UserInterface lets you link a specific file which contains the variables definition on the bus. Click the apposite button in the toolbar.

The parameters management window appears.



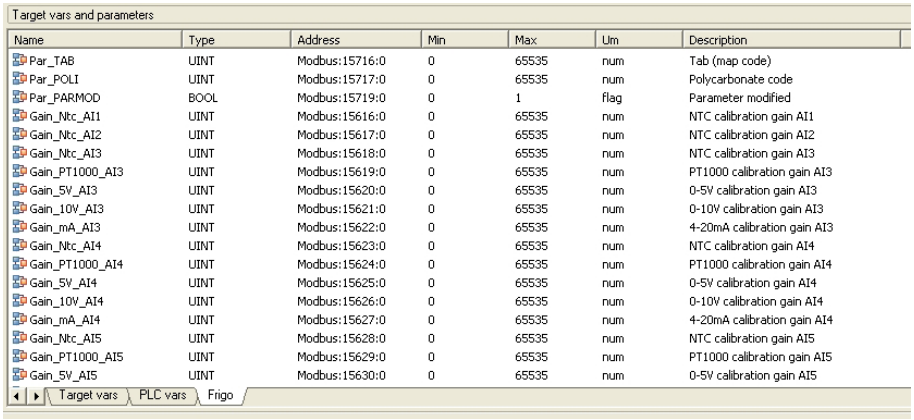
Through the *Add Device* button you can add a new object linked to the target on the fieldbus.

The selection window appears. Then you have to take from your PC a *.parx* file (see chapter 7). After inserting this file, the parameters management window will look like the image below.



A device called *Frigo* has been inserted. In order to see the relevant parameters, click the *Close* button.

In the *Window target vars and parameters* you will see the device and its parameters.


Target vars and parameters

Name	Type	Address	Min	Max	Um	Description
Par_TAB	UINT	Modbus:15716:0	0	65535	num	Tab (map code)
Par_POLI	UINT	Modbus:15717:0	0	65535	num	Polycarbonate code
Par_PARMOD	BOOL	Modbus:15719:0	0	1	flag	Parameter modified
Gain_Ntc_AI1	UINT	Modbus:15616:0	0	65535	num	NTC calibration gain AI1
Gain_Ntc_AI2	UINT	Modbus:15617:0	0	65535	num	NTC calibration gain AI2
Gain_Ntc_AI3	UINT	Modbus:15618:0	0	65535	num	NTC calibration gain AI3
Gain_PT1000_AI3	UINT	Modbus:15619:0	0	65535	num	PT1000 calibration gain AI3
Gain_5V_AI3	UINT	Modbus:15620:0	0	65535	num	0-5V calibration gain AI3
Gain_10V_AI3	UINT	Modbus:15621:0	0	65535	num	0-10V calibration gain AI3
Gain_mA_AI3	UINT	Modbus:15622:0	0	65535	num	4-20mA calibration gain AI3
Gain_Ntc_AI4	UINT	Modbus:15623:0	0	65535	num	NTC calibration gain AI4
Gain_PT1000_AI4	UINT	Modbus:15624:0	0	65535	num	PT1000 calibration gain AI4
Gain_5V_AI4	UINT	Modbus:15625:0	0	65535	num	0-5V calibration gain AI4
Gain_10V_AI4	UINT	Modbus:15626:0	0	65535	num	0-10V calibration gain AI4
Gain_mA_AI4	UINT	Modbus:15627:0	0	65535	num	4-20mA calibration gain AI4
Gain_Ntc_AI5	UINT	Modbus:15628:0	0	65535	num	NTC calibration gain AI5
Gain_PT1000_AI5	UINT	Modbus:15629:0	0	65535	num	PT1000 calibration gain AI5
Gain_5V_AI5	UINT	Modbus:15630:0	0	65535	num	0-5V calibration gain AI5

Target vars | PLC vars | Frigo

When you need to update the list of parameters, if the *.parx* file has not been moved to another directory, it is not necessary to repeat the above mentioned procedure, but it is enough to press the button

2.9 INSERTING EDIT BOX

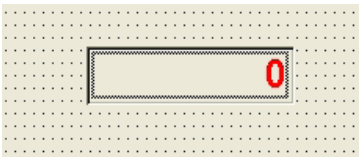
An edit box is a text frame which lets you display and eventually edit an associated variable or parameter.

2.9.1 INSERTING AN EDIT BOX IN THE PAGE

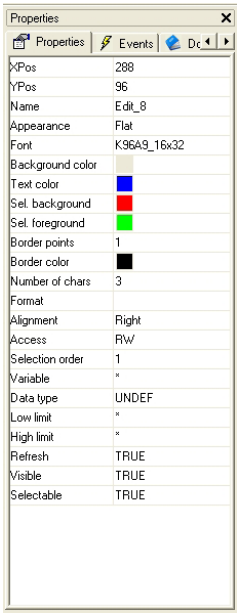
Insert an *Edit box* control in the page by pressing the corresponding button in the *Page toolbar*.

Move the mouse to the active area of the page. A cross + appears. The object will be inserted in the grid near to the mouse cursor.

Confirm the insertion point by left-clicking. A new text frame appears(1). It consists by default in a certain number of characters and its font is specified in the *Font* property of the page.

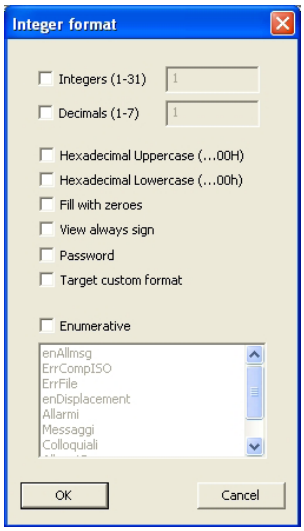


Edit this control's properties as you can see below(1).

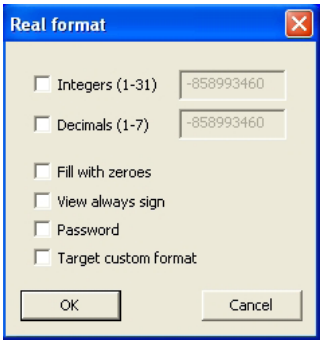


In the following list you can find all the changes which may be carried out:

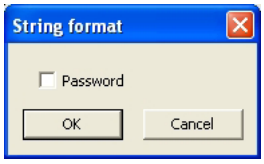
- *Appearance*: you can make the edit box appearance "sunken" by assigning the *Sunken* property.
- *Font*: you can customize font by choosing, for example, a 16x32 font instead of the default 8x16 font.
- *Select background* and *Select Foreground*: respectively text and background colors when the edit box is selected.
- *Number of Chars*: maximum number of characters which can be displayed.
- *Access*: in order to set the read-only mode, replace *RW* (read-write) with *RO* (read-only).
- *Refresh*: in order to constantly update the contents of the edit box, select the *TRUE* option. Otherwise, the contents are refreshed just when drawing the page for the first time.
- *Format*: it represents the display format of the associated variable's value. The format value can be inserted only if a variable is just available. It opens a dialog window with these settings according to the type of variable (integer, real, string).



- *Integers*: number of digit before comma
- *Decimals*: number of digit after comma
- *Hexadecimal Uppercase*: the number is shown as 0...0H representation with uppercase H letter
- *Hexadecimal Lowercase*: the number is shown as 0...0h representation with lowercase h letter
- *Fill with zeros*: fill the entire editbox controls with 0 where there are not numbers
- *View always sign*: show the +/- symbol in editbox
- *Password*: show only * symbols
- *Target custom format*: the target can define custom format to show the data in a particular way. In that case there is a variable on the target with the value of the corresponding user mode.
- *Enumerative*: this representation allows to select a string value corresponding to numeric value defined in *Resources*, under *Enumeratives*.



- *Integers*: number of digit before comma
- *Decimals*: number of digit after comma
- *Fill with zeros*: fill the entire editbox controls with 0 whrere there are not numbers
- *View always sign*: show the +/- symbol in editbox
- *Password*: show only * symbols
- *Target custom format*: the target can define custom format to show the data in a particular way. In that case there is a variable on the target with the value of the corresponding user mode.



- *Password*: show only * symbols.

The *Target custom format* is a special feature which enables a particular custom format implemented on the target.

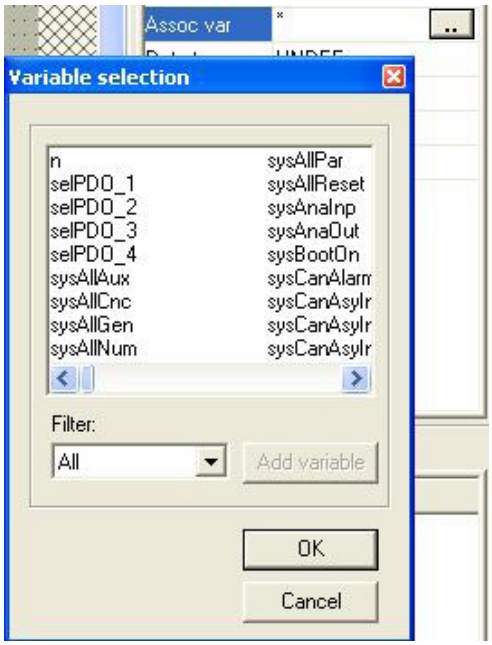
The format is specified according with language *printf* syntax (see 5.7.2).

2.9.2 EDIT BOX AND USERINTERFACE LOCAL VARIABLE ASSOCIATION

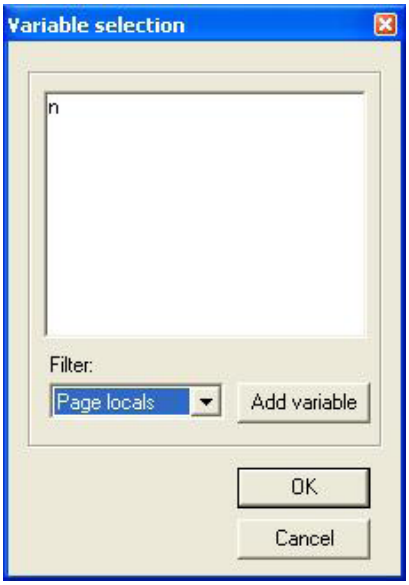
The edit box which you have just inserted lacks an essential element: the associated variable to take the values to display from. Let us assume that you want to link the edit box to a local variable (in order to get information on how to declare a local variable, see 2.8.1).

Select the edit box by clicking it once and select the *Variable* property.

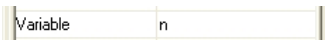
You can either type the name of the variable or click on the field and open the dialog window by clicking on the apposite button.



You can restrict the research just to the local variables of the *Init* page (consequently only the *n* variable) by using the *Filter* tool.



Select the local variable. The *Variable* field in the table properties refreshes accordantly.



Then the *Edit box* control shows the *n* local variable's value constantly refreshed.

2.9.3 EDIT BOX AND USERINTERFACE GLOBAL VARIABLE ASSOCIATION

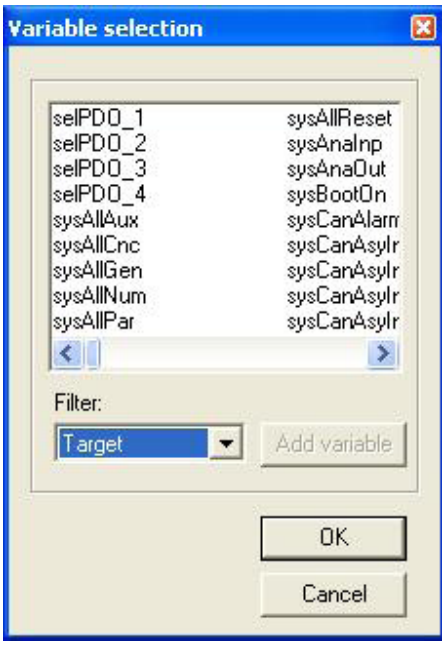
The principle to associate the *Edit box* control with a global variable is similar to the one to associate the *Edit box* control with a local variable. The difference consists in the variable declaration (in order to get information on how to declare a global variable, see § 2.8.2).

You can associate the Edit box with the global variable through the dialog window which was introduced in the preceding paragraph, but in this case it is necessary to use a different filter in the *Filter* field.

2.9.4 LINKING AN EDIT BOX WITH A TARGET (OR SYSTEM) VARIABLE

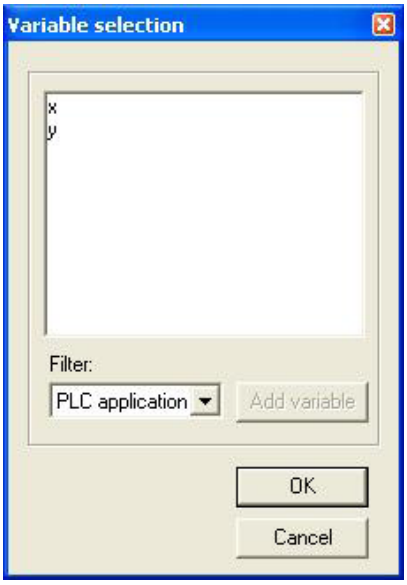
The target system executing PLC and HMI often publishes some variables which allow the interaction between user interface and system. In UserInterface, such variables are called target variables. You can view them in the *Target vars* table of the *Target Vars and Parameters* window.

You can associate an Edit box with a target variable through the dialog window which opens from the *Variable* field, but in this case it is necessary to use a different filter in the *Filter* field.



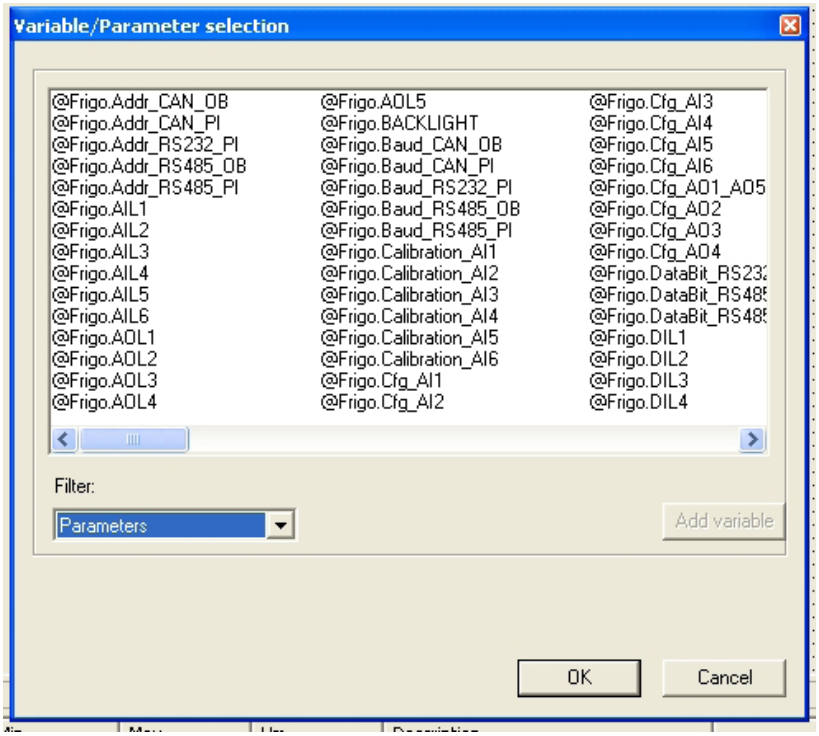
2.9.5 LINKING AN EDIT BOX WITH A PLC APPLICATION VARIABLE

You can associate an Edit box with a PLC Application variable through the dialog window which opens from the *Assoc var* field (see 2.9.2), but in this case it is necessary to use a different filter in the *Filter* field.



2.9.6 LINKING AN EDIT BOX TO A PARAMETER

You can associate an Edit box with a parameter through the dialog window which opens from the *Variable* field (see 2.9.2), but in this case it is necessary to use a different filter in the *Filter* field.



The name of parameters is composed of *@device.variable name*, differently from variables which show just their name.

The parameter may be inserted in the apposite controls property in the following forms:

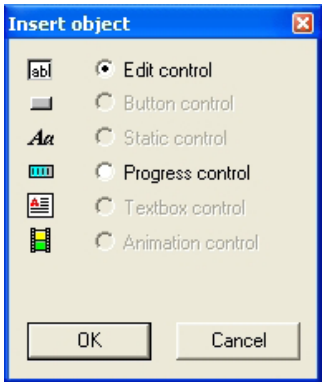
- explicit form = *@d.oi.os:type: d* = numerical ID of the device, *oi = object index*, *os = object subindex type= PLC type* (e. g. *@1.2010.0:UINT*);
- implicit form = *@dev.name: dev* = symbolic identifier of the device, *name* = symbolic name of the parameter (e. g. *Frigo.AIL1*).

The *d* (ID) field of the device is a numerical or symbolic identifier (to be defined at project creation). It refers to a specific device which may be local (the device which executes the pages itself) or on the fieldbus.

The *dev* field is a symbolic identifier of a device whose numerical ID can be retrieved by UserInterface.

2.9.7 LINKING AN EDIT BOX TO A VARIABLE BY DRAGGING AND DROPPING

You may add variables and parameters to the *Target vars and parameters* window by dragging and dropping them in the page. UserInterface will request to define the type of control to insert, to associate it with the variable.



2.10 INSERTING BUTTONS

Buttons are very versatile controls which play an essential role in the interaction between user and system, particularly in case of touchscreen systems without keyboard.

This chapter's aim is to show four kinds of use of the button control:

- LED-button to view the state of a boolean variable;
- command button of a boolean variable's state;
- opening button of a secondary page;
- activation button to start the execution of a customized procedure.

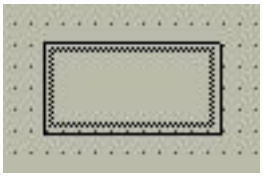
2.10.1 INSERTING A LED-BUTTON

The following paragraph teaches you how to use a button which shows an associated boolean variable's state.

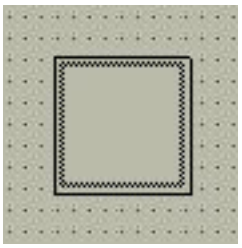
Insert a new button in the page by pressing the corresponding button in the *Page toolbar*.

Move the mouse in the active area of the page; a cross + appears. The object will be inserted in the grid near to the mouse cursor.

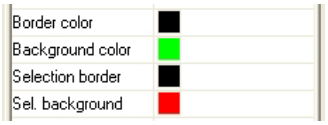
Confirm the insertion point by left-clicking. A new *Button* control appears. It has a default size.



You may edit both the dimensions by dragging one of the button's vertexes or one dimension at a time by dragging one of the button's sides.



The *Border color* and the *Background Color* properties determine the border and the background color when the button is inactive, whereas the *Selection Border* and *Selection Background* properties define the border and the background color when the button is selected(1).

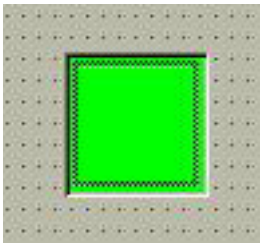


The *Selection variable* property determines the state of the button and, consequently, the couple of colors related to the control. This property may be associated either with a constant value (*FALSE* = the control is always inactive, *TRUE* = the control is always selected) or with a boolean variable whose value determines dynamically the selection state.

Declare a boolean global variable *b* and associate it with the control button as selection variable.

Transparent	FALSE

You may customize the button appearance through the *Appearance* property. For example, choose the *Sunken* option(1).



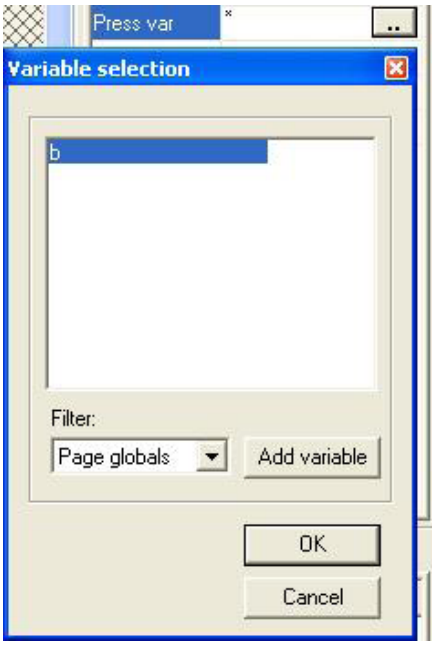
2.10.2 INSERTING A BOOLEAN VARIABLE COMMAND BUTTON

Insert a new button in the page by following the aforesaid instructions (see 2.10.1). Set it beside the LED button and let a text string show on it by means of the *Text property*(1).

The preview looks like as follows(1).

The *Press variable* property allows the user to associate a boolean variable with a button control. The boolean variable's value corresponds to the pressure state of the button.

For example, associate the button which you have just created with the global variable *b* which has been created paragraph 2.10.1.



At runtime, the LED-button (see 2.10.1) will be red when pressing the *Press* button. Otherwise it will be green.

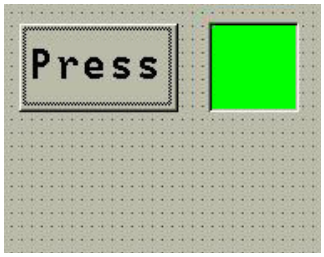
2.10.3 INSERTING A BUTTON TO OPEN A CHILD PAGE

Paragraph 2.4.1 showed you how to create a pop-up page.

The following paragraph explains how to invoke the *Pag2* page from the *Init* page by pressing a button.

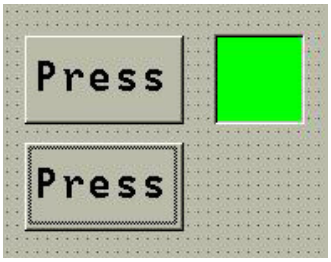
First of all insert a new button in *Init* and set it under the previously created *Press* button (see 2.10.2). As it should be exactly alike the previous one except the text string and the function, you can copy and paste the *Press* button and afterwards customize its properties.

Select the *Press* button by clicking once: the selection rectangle appears inside the control(1).

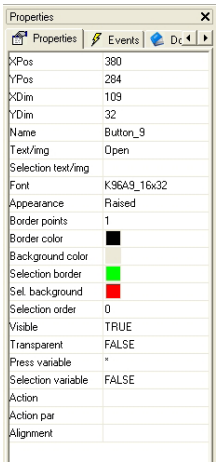


Press successively *Ctrl+C* and *Ctrl+V*. A cross + appears. The object will be inserted in the grid near the mouse pointer.

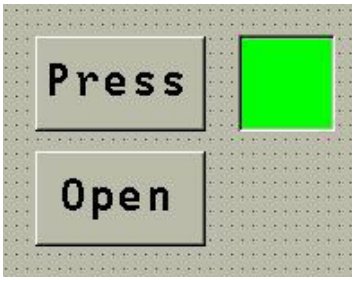
Confirm the insertion point by clicking under *Press*. A copy of the control appears(1); it is the same as the source button except its position and name.



You can access this new control's properties and customize them according to the relative purpose(1).



The preview looks like this(1).



The button control has got a very important attribute, which has not been represented in the properties grid above: the *Action* attribute allows the user to associate an action with the button pressure. Some actions require an additional parameter which you may specify in the *Action par* field.

In this case let us assume that you want that the pressure of the *Open* button opens the *Pag2* page. To obtain this select the *OpenPage* action in the *Action* field; then type the name of the child page *Pag2* in the *Action par* field.

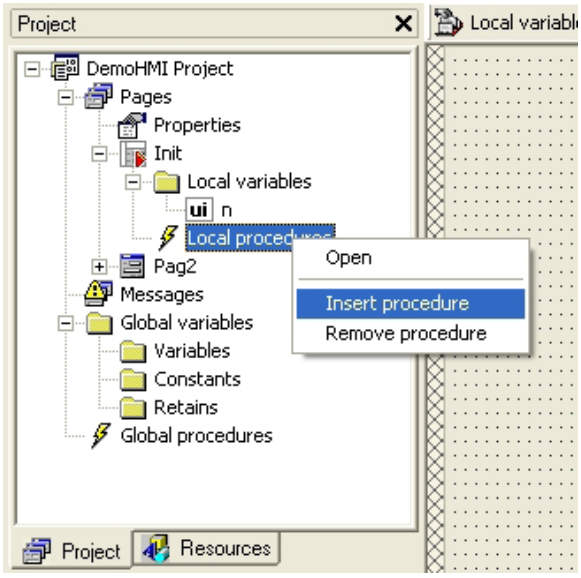
Action	OpenPage
Action par	Pag2

2.10.4 INSERTING A BUTTON AIMED AT LAUNCHING A PROCEDURE OF THE USER

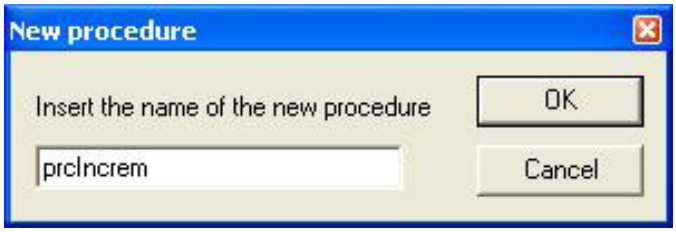
UserInterface enables the user to implement some procedures (see 4.8.4) through which it is possible to customize the HMI behaviour: this feature makes UserInterface projects very versatile.

Let us suppose that you want to create a procedure to increment the local variable *n* of the *Init* page. As this procedure applies on a local variable, it will be local in the *Init* page, too.

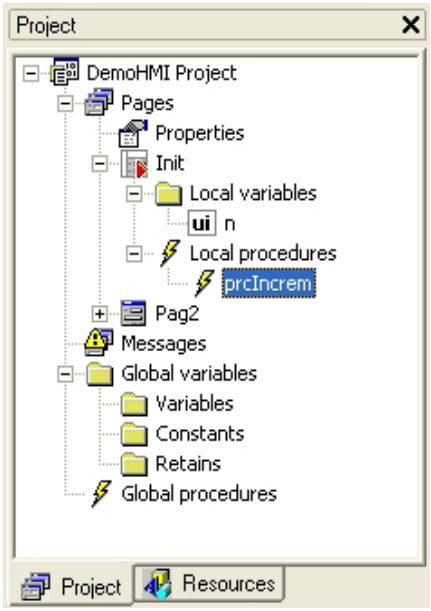
First of all create the procedure: expand the *Init* page tree, right-click on the *Local prcedures* item and select the *Insert procedure* command in the contextual menu which appears.



A little dialog window opens. The user is then required to type the new procedure's name. In this case, it may be *prcIncrem*.

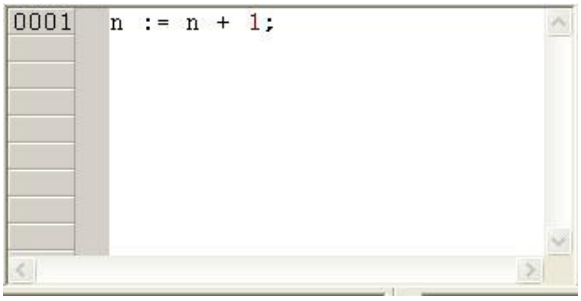


Press the *OK* button. Then UserInterface adds a new item in the page tree: it corresponds to the local variable which has been just declared.



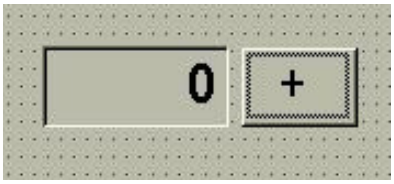
Double-click on the above mentioned new item: the ST language editor opens and lets you either implement or edit the selected procedure's code.

Write a procedure that applies a unit increment to the *n* variable.



Then close the document.

Insert a new button beside the edit box associated with the *n* variable and type the character + in the *Text* property(1).



Let us suppose that you want to execute the *prcIncrem* procedure by clicking the + button: select the *Call* action in the *Action* field and type the procedure's name in the *Action par* field.

Action	Call
Action par	prcIncrem

Every time the user will press the + button when executing the HMI, *n* will increase by one and the edit box will show the up-to-date value.

2.11 VISIBILITY AND UPDATING OF CONTROLS

As stated in the previous paragraphs, each control has its own properties which the user may customize through the properties table fields.

Some of these features are specifically related to a single type of control. Others may be included in the properties set of different objects. The following paragraphs concern two important properties which are common to some kinds of control.

2.11.1 THE VISIBILITY PROPERTY

Almost all of controls are endowed with the *Visibility* property, which determines whether the object is visible or not. This property can be associated either with a constant value (*FALSE* = the control is always hidden, *TRUE* = the control is always shown) or a boolean variable, whose value dynamically establishes the visibility state.

By following the instructions in paragraph 2.7.1 you have inserted the string: *Stringa di testo* in the *Init* page. At present this string is always visible, as you can deduce from the assigned value to its *Visibility* property.

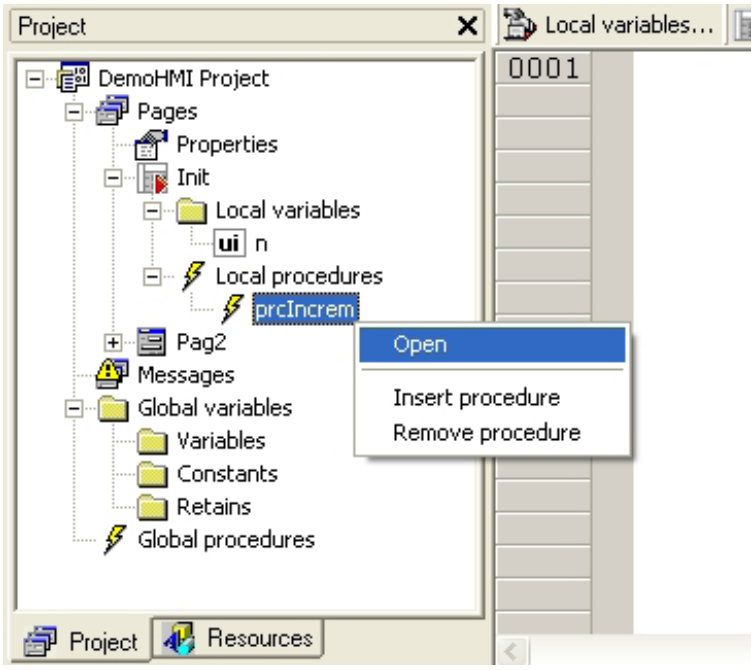
Visible	TRUE

Let us assume that you want to assign this control's visibility to the local variable *n*, which is displayed in the edit box created in paragraph 2.9.2 and managed by the *prcIncrem* procedure, which was implemented in paragraph 2.10.4 and started up by the + button. More precisely, let us suppose that you want the text string visible when *n* is even, whereas hidden when *n* is odd.

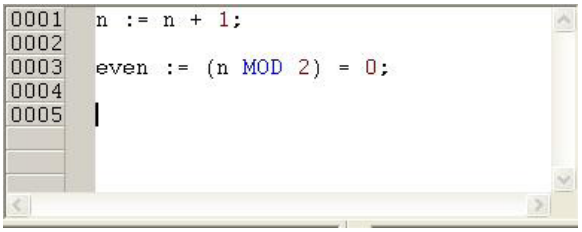
To this purpose, it is necessary to declare a new boolean local variable which indicates whether at present *n* is even.

	Name	Type	Array	Init value	Description
1	n	UINT	No	100	Variabile contatore
2	even	BOOL	No	TRUE	Variabile locale n è pari

Then it is necessary to edit the *prcIncrem* procedure so that, when it refreshes the *n* value, it evaluates again whether it is even or odd. In order to access the *prcIncrem* source code, select the corresponding item in the project tree by right-clicking. Afterwards, choose *Open* from the contextual menu which appears.

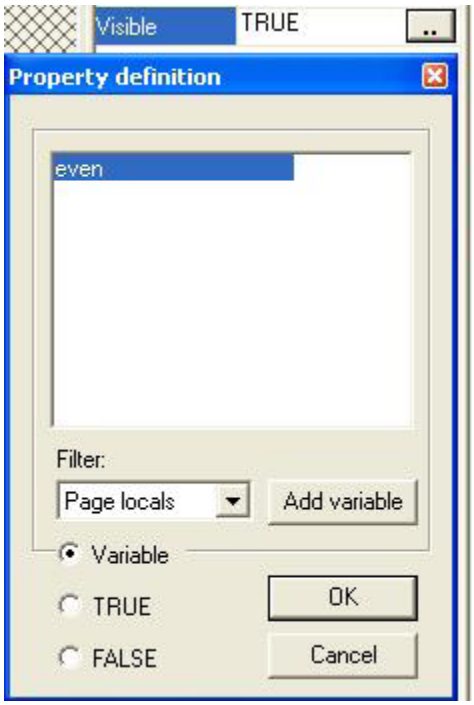


The ST language editor opens and the procedure's code may be extended as follows.



In order to associate the string's visibility state with the *even* boolean variable, select the text string and click the *Visibility* field: a button appears.

After clicking it, a dialog box opens. Select the *radio button Variable*, which enables the overhead variables list; change the filter *Filter* into *Page locals* and select the only local boolean variable that is *even*.



Confirm your choice by clicking *OK*. The result is the following.

2.11.2 THE REFRESH PROPERTY

When available, the *Refresh* property determines it the associated object has to be drawn once (when opening the page or coming back from a child page) or it needs to be constantly refreshed.

This property distinguishes, for example, the edit box and the text box.

With regard to the edit box, the refresh property has to be set when compiling and it cannot be edited at runtime. If you assign *Refresh = TRUE*, the associated variable's value is constantly read and refreshed, otherwise (*Refresh = FALSE*) the value is read and refreshed only when you open the page or when you come back from a child page.

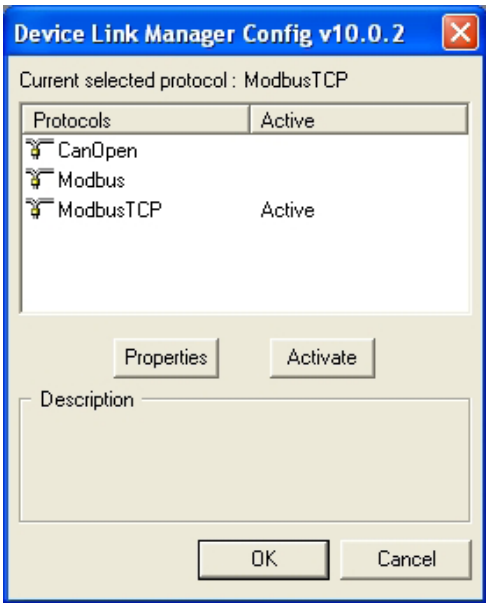
There is another option about text boxes: you can associate a boolean variable that is used as trigger for refresh: when the trigger variable becomes *TRUE*, the control's contents are refreshed then it is automatically reset by UserInterface to *FALSE*.

2.12 COMPILING AND DOWNLOADING THE PROJECT ON THE TARGET

The following paragraph shows you how to compile and download a HMI project on the target board that runs UserInterface.

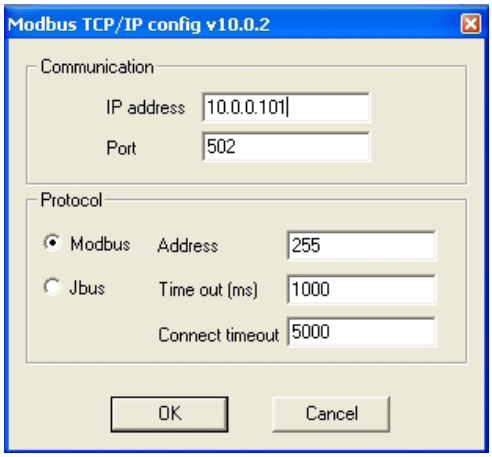
2.12.1 CONNECTING TO THE TARGET

Launch the *Communication settings* command from the *Project* menu. This causes the following dialog window to open.



The user is required to select a suitable communication protocol from the left column and to activate it by pressing the *Activate* button.

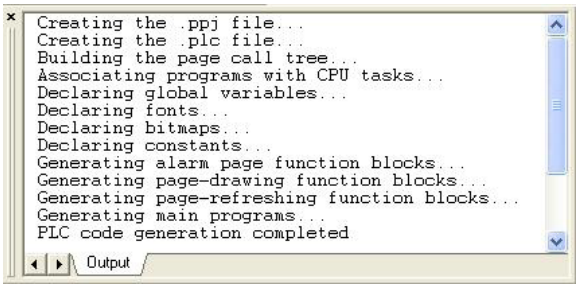
Then the *Properties* button becomes active: by clicking it the user accesses another dialog window which is different in accordance with the specific selected control and lets set the protocol's parameters. Let us consider the following example.



2.12.2 COMPILING PAGES FOR THE TARGET

You can start compiling the HMI project by clicking the corresponding button in the User-Interface's *Project toolbar*.

Compilation is composed of two phases: the first one consists in the PLC code generation which realizes the pages as they have been planned in UserInterface. The program shows in the *Output window* the progress level of the compilation and displays eventual errors.



The second one consists in the compilation of the PLC code which has been generated during the first phase. It can be started only if the first phase has been accomplished without any error.

This process is carried out by an external tool: the PLC command-line compiler *llc*, which UserInterface automatically invokes with the suitable parameters.

2.12.3 DOWNLOADING AND EXECUTING THE COMPILED PAGES ON THE TARGET

At the end of the compilation, if all the phases have been successfully accomplished, you will see the downloading button become active in the *Project toolbar*

Clicking it, you activate again the PLC command-line compiler *llc*, which this time just downloads the compiled code in the target.

The downloading permission management depends on the implementation of the on board firmware. Consequently it changes according to the destination target of the download.

2.12.4 SIMULATION

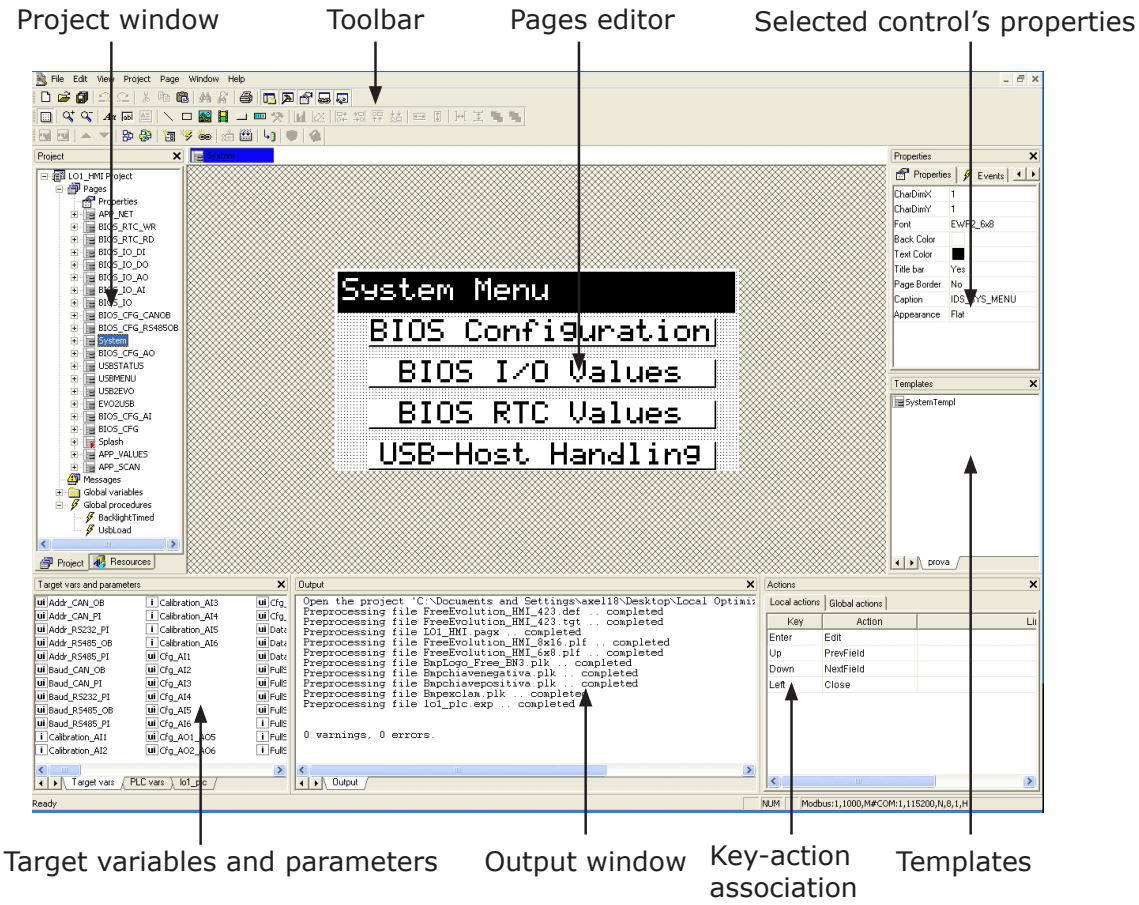
Depending on the target device you are interfacing with, you may be able to simulate the execution of the HMI application with UserInterface's integrated simulation environment: Simulation.

In order to start the simulation, just click on the appropriate item on the Project toolbar.

Refer to Simulation's manual to gain information on how to control the simulation.

3. USERINTERFACE LAYOUT

The following picture shows you the layout and the essential elements of UserInterface.



3.1 PROJECT WINDOW

This window includes two pages which are alternatively selectable by pressing the corresponding tab:

- Project: it shows the project tree and all the objects the project is composed of, hierarchically arranged. The pages node contains the project properties and the single pages. Each page contains the list of the local variables (visible and usable only in the page where they are declared) and the local procedures, which can be invoked only from the page where they are implemented. Moreover there is the node of the asynchronous messages, the node of the global variables (visible and usable from whatever page) and the node of the global procedures which you can invoke from whatever page.
- Resources: it shows the project resources, that is fonts, bitmaps, strings table, enumerated data types, images lists, and sets.

3.2 EMBEDDED EDITORS

UserInterface is endowed with three types of editor:

- Pages editor: in order to open this editor, double-click the name of the desired page in the project tree (see 3.1). This tool shows you a page preview and lets you edit it: you may either add or remove controls (see 4.4), customize properties, manage the events and the documentation.

- Variables editor: by double-clicking on a local or global variable in the project tree, you can see respectively the declaration table of the local variables or the global variable table.
- Procedures editor: it allows the user in implementing procedures to be associated to the events which are defined for the various project's objects (pages and controls) or generated from the user himself (see 4.8).

3.3 PROPERTIES WINDOW

Each time you select an object in the pages editor, the properties window automatically refreshes and shows the selected object's properties and events.

This window is composed of many pages which you may select alternatively by pressing the corresponding tag above.

- Properties: it shows a table including the selected object's properties either it is a whole page or it is a page's control. The user is enabled to customize this values through the right-hand column of the table.
- Events: it shows a table including the typical events of the currently selected object. The user may associate either a local or variable procedure with each event by typing its name on the corresponding row of the event in the right-hand column.
- Doc: it displays a table which shows the *Description* field of the currently selected object. The user may describe the object and this description will be included in the automatic documentation management (see 4.10).

3.4 TOOLBARS

The user can give commands to UserInterface through some useful toolbars. A toolbar can be defined as a collection of buttons which you may enable by left-clicking them and whose functions are intuitively represented by their icons.

The toolbars support tooltips, too. A tooltip is a small text frame containing a short description of the object which UserInterface automatically displays when you hover with the mouse over a button.

UserInterface is endowed with three essential toolbars:

- Main toolbar: it contains the commands to open and save the project, to cancel/restore the last changes, to print, to display or close other toolbars.
- Project toolbar: it allows you to add new elements to the project as variables, pages, events, actions, as well as to enable or prevent the simulation mode and to compile and download the whole project.
- Page toolbar: it allows you to choose a new type of control to be inserted in the active page, to align or equally space several controls, or to set the vertical order of the elements on the page.

3.5 THE OUTPUT WINDOW

UserInterface prints in this window some messages which indicate the progress and the output of the requested processes: opening and compilation of a project, resources importing/exporting, etc..

3.6 TARGET VARIABLES AND PARAMETERS

This window shows the list of external variables, available for UserInterface coding.

The window is composed of several pages which you may alternatively select by pressing the corresponding tab. One page contains the list of the available variables (file *.tgt*),

another page contains the list of the variables which have been exported from the PLC Application (file *.exp*). Other pages are optional, as many as the number of devices with external parameters linked to the project.

3.7 TABLE OF KEYS-ACTIONS ASSOCIATIONS

This table takes primary importance in case of traditional keyboards without touchscreen where the user interact with the system by pressing the relevant keys.

See in paragraph 4.8.5 the list of actions which may be associated to keys.

4. HMI PROJECT IN USERINTERFACE

UserInterface manages the creation (development) of pages for a specific application as projects.

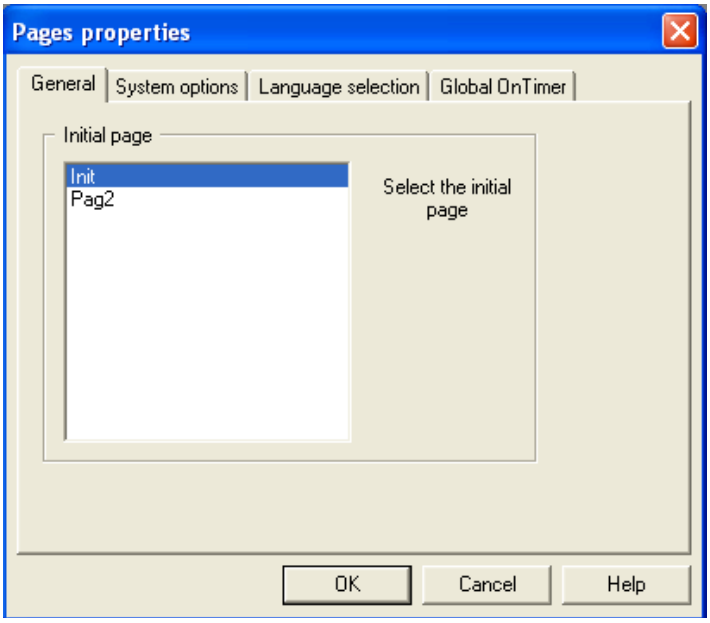
The UserInterface project is composed of several pages where the user may arbitrarily arrange the controls.

In each UserInterface project you have to specify the start page which will be displayed at the start of the system. Other pages will have at least a parent page from which they will be invoked and may have child page to invoke. The invoking/invoked relations implicitly give to the whole project a tree structure.

4.1 PROJECT PROPERTIES

In the project tree, click the *Pages* item and access the *Properties* item. By double-clicking the *Properties* item you open a dialog window which is composed of four pages. The following paragraphs show you the features of these pages.

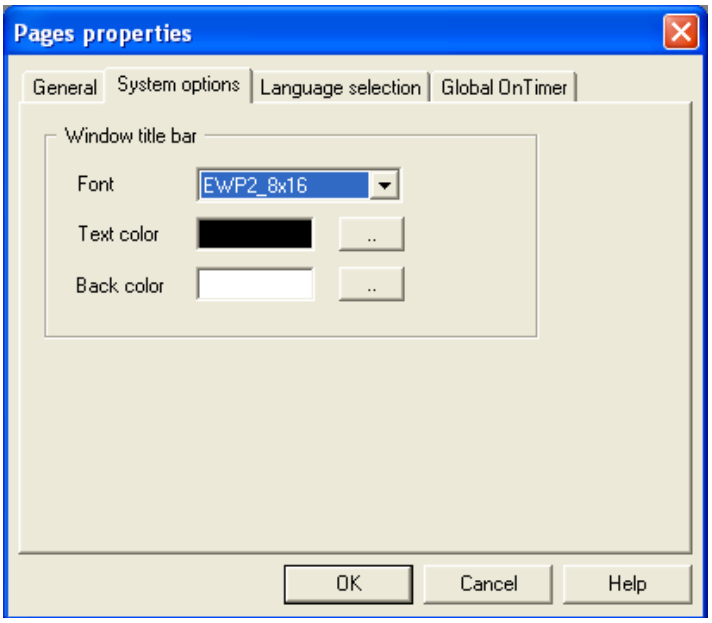
4.1.1 GENERAL



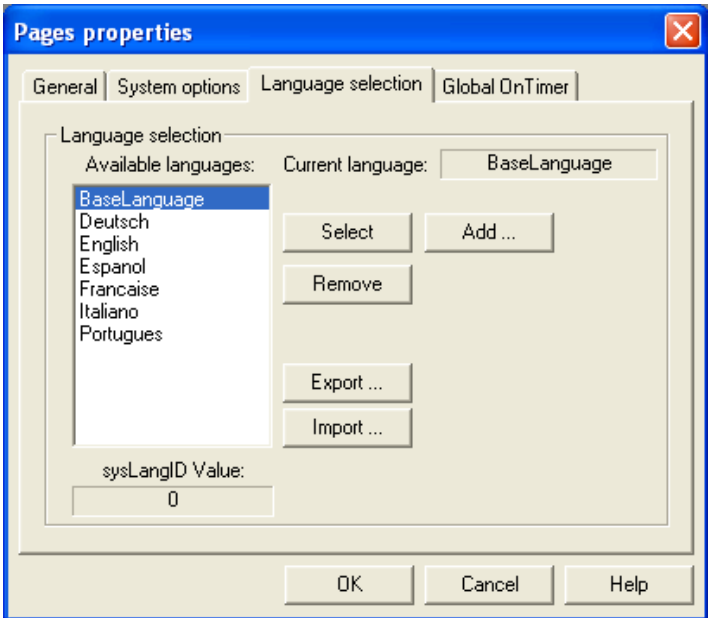
It allows to select the UserInterface project's start page among the implemented pages.

4.1.2 SYSTEM OPTIONS

It allows the user to customize the window's title bar features: the font, the text color and the background color.



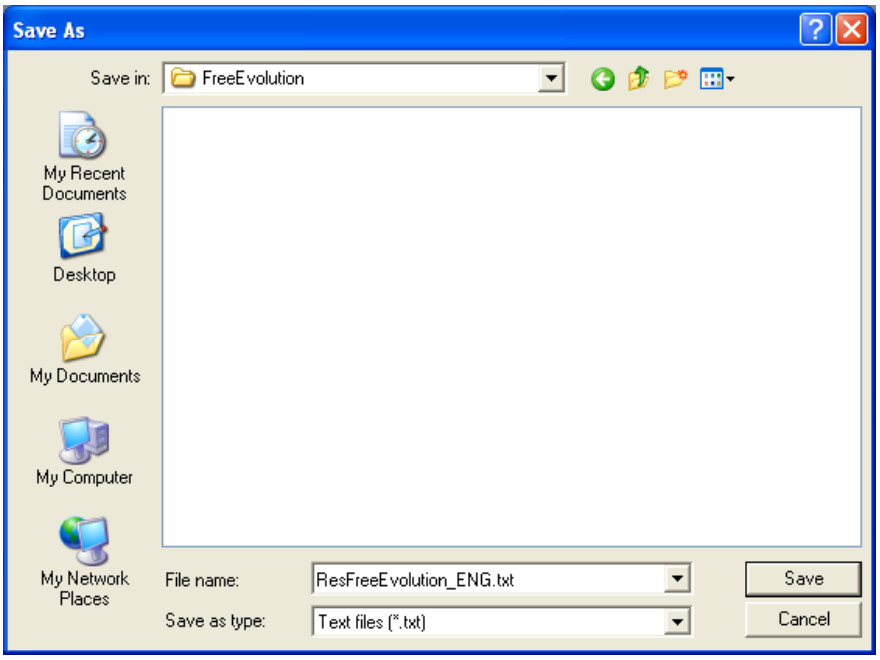
4.1.3 LANGUAGE SELECTION



It allows you to add, remove, export, import, and select the resources languages (see 4.9). The label: *sysLangID Value* indicates the value which the *sysLangID* target variables must take to display the pages in the selected language.

In order to add a language, apply the following procedure.

First of all export the language supported by the translator, choose Italian and press the *export...*button, which opens a window requiring the destination folder for the selected language file.

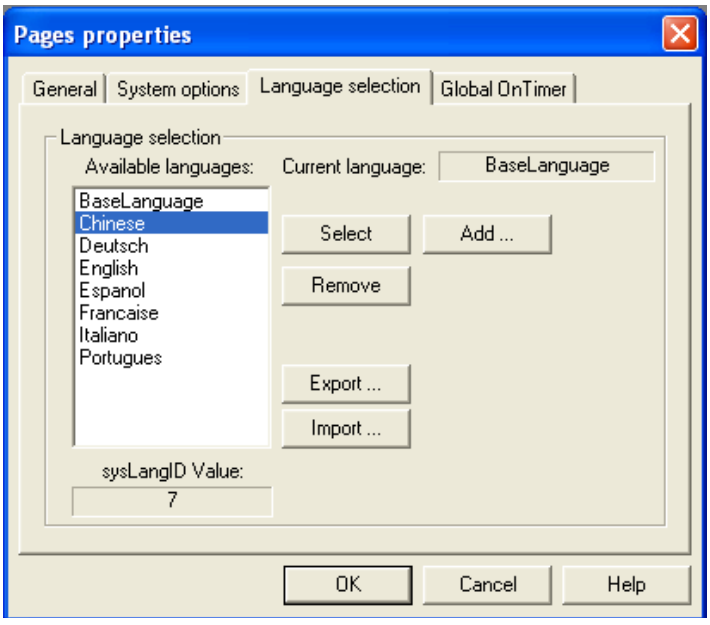


The program suggests a file name: *Res* + project title+'_' + first three characters of the language + extension *.txt*. At the end of the exportation the file is composed of all the project's resources which have to be translated:

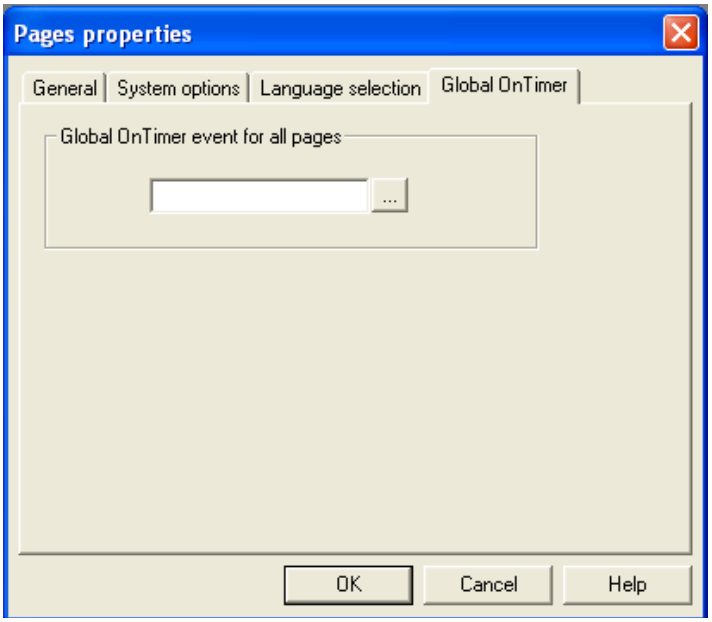
- strings
- enumeratives

Translate the file and replace the text under the *Language* tag with the one of the new language (for example, in this case change it into *Chinese*). In the *Language selection* panel choose the *Import...* button, then select the suitable file in the PC.

The new language appears in the list.

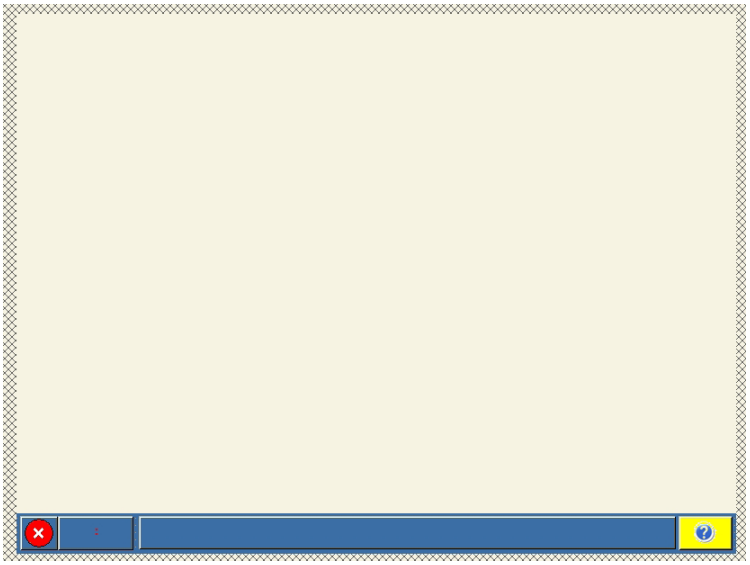


4.1.4 GLOBAL PERIODIC PROCEDURE



Global on timer allows you to specify the name of a global procedure to be periodically and independently executed on the active page. Such a procedure may be effectively used to constantly test one or more PLC variables and to emit alarm messages, for example through asynchronous messages (see 4.3.4).

4.2 FRAME SET



UserInterface allows to define areas which are called frames and are placed on the sides of the screen and are always active(1).

The user may set these frames' dimensions and insert some controls which are active whatever the currently loaded page. Consequently frames are useful to host the objects which have to appear in the whole project. In this way the user does not need to duplicate them in each page.

As regards to the above, there are two exceptions: the pop-up pages (see 4.3.3) when the *Modal* property is set to *Yes* and all the asynchronous messages. When these pages are active, the controls of the frame set are automatically disabled.

4.3 PAGES

4.3.1 NAVIGATING BETWEEN PAGES

UserInterface manages pages development for a specific application as projects.

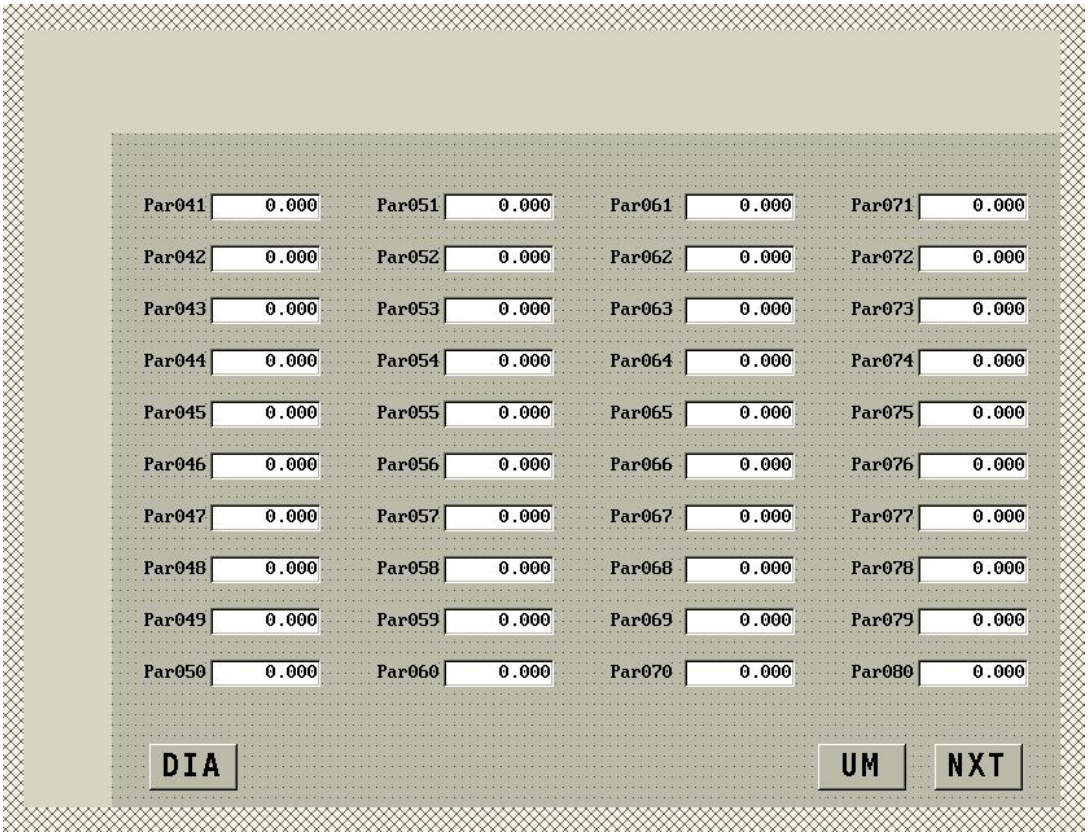
UserInterface project is composed of pages where the user can arbitrarily arrange controls.

In each UserInterface project it is necessary to define a start page which will be viewed at system startup. Other pages must have at least a parent page from which they are invoked and may have child page to invoke. The invoking-invoked relations of the pages give the whole project, even though in an implicit way, a multi-node tree structure.

A child page may be invoked in two ways:

- Through an action associated to a key: associate an *OpenPage* action with a physical key (if there is a keyboard) or with a virtual key (whose pressure is an event raised by software);
- Through an action associated with a button: insert in the parent page a *Button* control (see 4.4.7) and specify in the *Action* property that by pressing it the child page opens.

4.3.2 CHILD PAGES

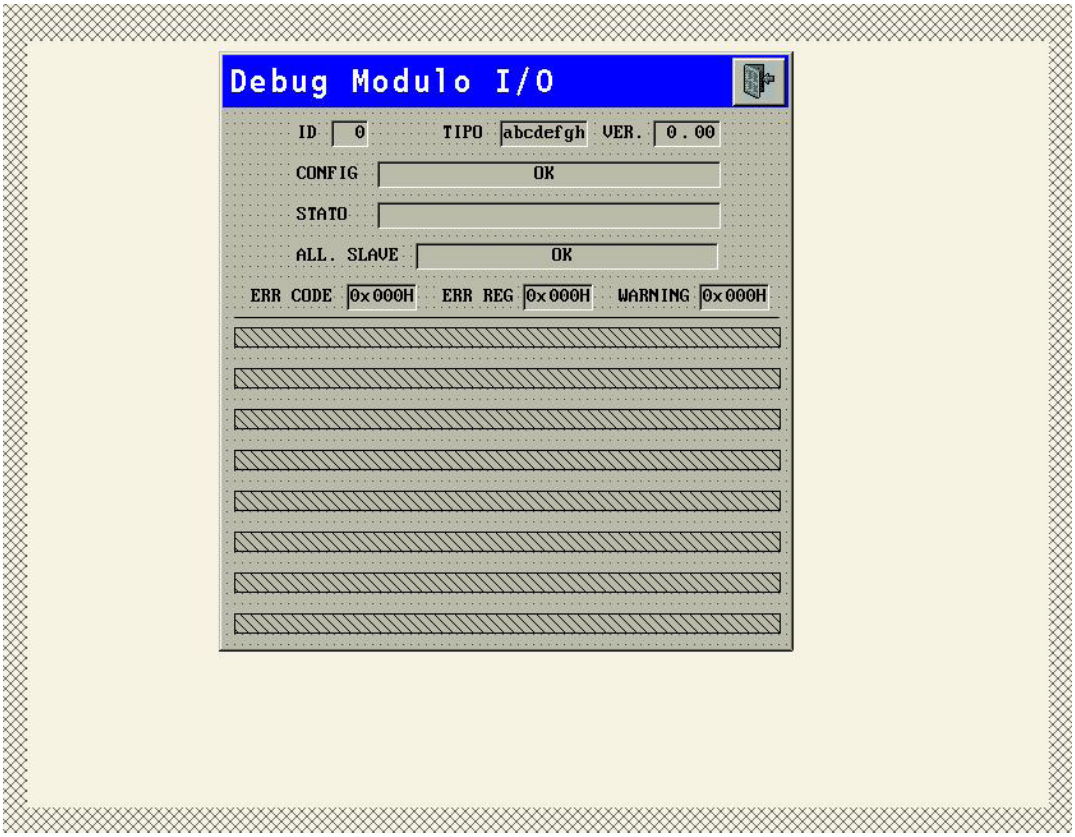


Let us assume that you want to add a page to a project(1).

UserInterface displays a dialog window which requests to insert the name you intend to assign to the new page. This dialog window contains a checkbox with the label: *Pop-up*. If you do not select it, the new page will be a child one.

A child page fits the whole screen or, alternatively if there are defined frames (see 4.2), it fits the free remaining area. Consequently, the user cannot define position and dimensions of a child page as they are automatically set according to the screen and the frame set.

4.3.3 POP-UP PAGES



When creating a new page, if the user selects the aforesaid checkbox with the *Pop-up* label, the new page will be a pop-up one(1).

There are no restrictions about position and dimension. In fact the user may superimpose a pop-up page on the frames: when activating this page, if it is not modal (property: *Modal*), the controls superimposed on the open page will be disabled; otherwise, all the controls will be inactive.

4.3.4 ASYNCHRONOUS MESSAGES

Asynchronous messages are similar to standard pages, except the following features:

- They have an additional property, that is the identifier of the associated message (*Msg ID*).
- They cannot contain invocations to child pages.
- They have no defined parent page nor a tree structure (see Introduction 4.), but they can be invoked from any other standard page.

An asynchronous message cannot be explicitly invoked; the system displays it whatever the active page when it intercepts a message containing the corresponding *Msg ID*. This message may be launched either by the firmware or by a procedure through the *Video_SendMessage* function (see 8.1.8) by using the following syntax:

```
Video_SendEvent( kWM_MSG, Msg ID );
```

4.4 CONTROLS

A control is a display element which is contained in a page. The following paragraphs shows you the controls which UserInterface supports.

4.4.1 STATIC

It displays a fixed string, whose contents cannot be edited when executing. In fact, you should specify the text of the string directly or by the association of the ID of a string defined as resource to support multi language management. For project resources and multi language support see paragraph 4.9.

In order to insert a *Static* control, press the corresponding button in the *Page toolbar*.

Then click the point where you want to insert the control.

You can get information on properties and events of the *Static* control in paragraph 5.4.

4.4.2 GRAPHIC ELEMENT

It displays a static line or rectangle. This means that their properties cannot be edited when executing.

In order to insert a *Line* control, press the corresponding button in the *Page toolbar*.

Then click the point where you want to insert the control.

In order to insert a *Rectangle*, press the corresponding button in the *Page toolbar.*

Then click the point where you want to insert the control.

You can get information on properties and events of the *Line* and *Rectangle* controls in paragraphs 5.5-5.6.

4.4.3 EDIT BOX

It displays the contents of an associated variable.

In order to insert an *Edit box*, click the corresponding button in the *Page toolbar.*

Then either click the point where you want to insert the control or drag a variable from the project tree or from the library window.

You can get information on properties and events of the *Edit box* control in paragraph 5.7.

4.4.4 TEXT BOX

It displays the contents of an associated string variable. It supports the formatting on several lines of the text which is contained in the string.

To insert a *Text box* control in the page, press the corresponding button in the *Page toolbar*.

Then either click the point where you want to insert the control or drag a variable from the project tree or the library window.

You can get information on properties and events of the *Text box* control in paragraph 5.8.

4.4.5 IMAGE

It displays a bitmap image.

In order to insert an *Image* press the corresponding button in the *Page toolbar*

Then click the point where you want to insert the control.

You can get information on properties and events of the *Image* control in paragraph 5.9.

4.4.6 ANIMATION

It displays a bitmap image which you select from a list of images depending on the value of an associated selection variable.

In order to insert an *Animation* press the corresponding button in the *Page toolbar*.

Then click the point where you want to insert the control.

You can get information on properties and events of the *Animation* control in paragraph 5.10.

4.4.7 BUTTON

You may use the *Button* control either to check a boolean variable's state or (press= *TRUE*, release = *FALSE*) or to send a command to the system.

In order to insert a *Button* press the corresponding button in the *Page toolbar.*

Then either click the point where you want to insert the control or drag a boolean variable from the project tree or the library window.

You can get information on properties and events of the *Button* control in paragraph 5.11.

4.4.8 CHART

Chart control draws the static diagram of one or more arrays of values associated.

In order to insert a *Chart* control, click the corresponding button in the *Page toolbar.*

Then click the point where you want to place the control.

You can get information on properties and events of the *Chart* control in paragraph 5.14.

4.4.9 TREND

After assigning up to 8 numerical variables, the object will automatically and periodically (once every a defined time) acquire their values and will draw the corresponding graphic in a dynamic and automatic way.

In order to insert a *Trend* control press the corresponding button *Page toolbar*.

Then click the point where you want to insert the control.

You can get information on properties and events of the *Trend* control in paragraph 5.15.

4.4.10 PROGRESS BAR

It represents the progress of an operation by showing a stained bar in a horizontal or vertical rectangle. The length of the bar, related to the bar's lenght, shows the percentage of the completed operation.

In order to insert a *Progress bar* control press the corresponding button in the *Page toolbar*.

Then click the point where you want to place the control.

You can get information on properties and events of the *Progress bar* control in paragraph 5.12.

4.4.11 CUSTOM CONTROL

This control is implemented in the firmware. You can have several types of custom controls which are marked by the *Control ID* property and each type of control may have several instances.

In order to insert a *Custom control*, press the corresponding button in the *Page toolbar.*

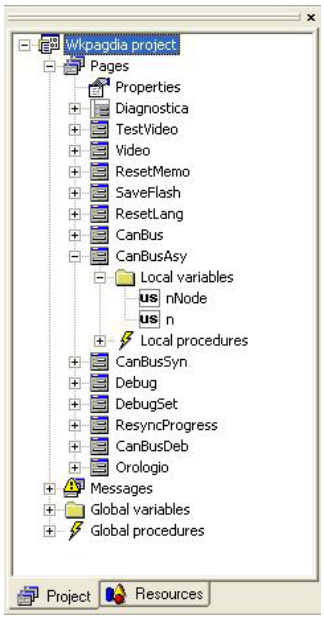
Then click the point where you want to insert the control.

You can get information on properties and events of the *Custom control* in paragraph 5.13.

4.5 VARIABLES

In a UserInterface project there are different classes of variables. The following paragraphs show you their features.

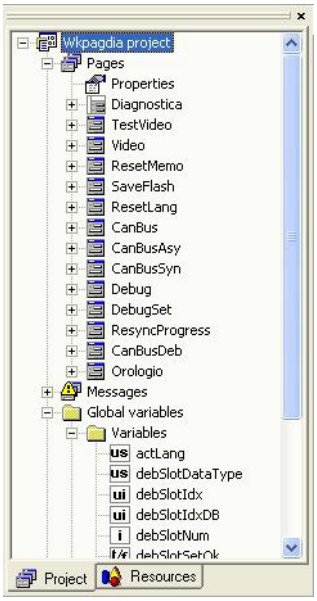
4.5.1 LOCAL VARIABLES



Local variables are variables of the UserInterface project. You can access them only through the page they were declared from.

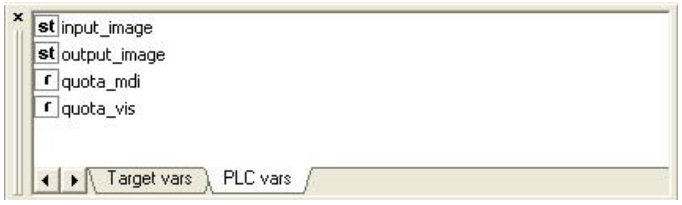
They are listed in the project tree, under the *Local variables* folder. Local variables can be used to carry out operations on PLC (for example to apply a different scale or to add an offset) or system variables or to implement local procedures.

4.5.2 GLOBAL VARIABLES



Global variables are declared in UserInterface and they are accessible from every page of the project. Global variables are listed in the *Global variables* folder in the project tree. The function of the global variables is similar to the local variable's one but the different visibility scope makes them unusable for the implementation of global procedures or for the parameters passing between distinct pages.

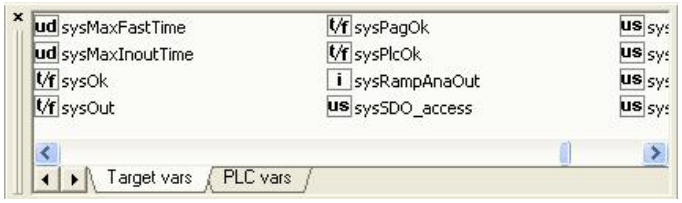
4.5.3 VARIABLES IMPORTED FROM PLC



A compiled UserInterface project consists in a PLC that, once downloaded on the target board, is executed by the actual PLC., which is implemented with Application. Variables exported from the Application PLC contained in the *.exp* file enable the interaction between these two distinct components. PLC variables which are not automatic, thus associated to a datablock are exported in the *.exp* file.

In order to include a *.exp* file in the UserInterface project, press the *Link PLC variables file* option from the *Project* menu, then search for the file in the PC resources. After linking a *.exp* file to the UserInterface project, you can get a list update of the exported variables by selecting the *Refresh PLC variables* option from the same menu.

4.5.4 SYSTEM VARIABLES



The interaction between UserInterface and target is enabled by system variables which the software publishes outside in a *.tgt* file.

You may access system variables in read/write or in read-only mode; if you try to access a read-only variable in write mode, an error will occur when compiling.

4.6 MULTIPLE PAGES MANAGEMENT

These functions allow to construct pages with data of different kind that must be represented on distinct pages for space reasons.

Sets (see 4.9.6) can be used with edit boxes or progress bars. Sets are ensemble of variables even of different type. Set definition can be done from the resource tree, they are implemented using a table with a series of variables that are dynamically associated to the control basing on the current index assigned to the page.

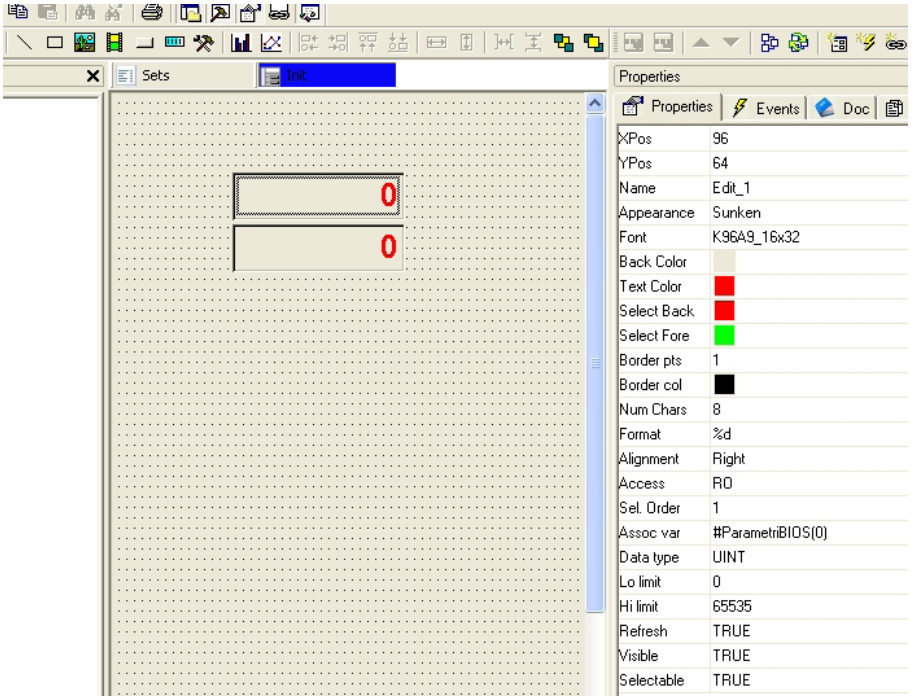
Let us see how to use a set.

4.6.1 ASSOCIATION OF ELEMENTS OF A SET

Elements of a set can be associated to a control using the following syntax: character # first of all, then the name of the set followed by the index of the position of the element in the page, between round brackets.

Position index is used to indicate the order in which elements are shown in case of more than one element in the same page.

A page contains one or more controls based on one (or more) set. At runtime, the page is replied in order to show all the elements contained in the set. In the last page, if any control cannot be filled with element value, that control is hidden(1).



We created before a set of five elements named *BIOSParameters*, now we can associate *#BIOSParemeters(0)* to the first edit box and *#BIOSParemeters(1)* to the second. So there are three pages: first page with the first two elements of the set, second page with elements 2 and 3 and third page showing the last element of the set. In the last page the second edit box is not visible.

4.6.2 NAVIGATION OF THE ELEMENTS OF A SET

Navigation of pages that represent a set of elements is automatically done using the *NextEdit* event of the last selectable control of the page and using *PrevEdit* event of the first selectable control of the page.

It is also possible to send special events to force the change of the page in this way:

```
Video_SendEvent( kEV_WM_CHANGESETPAGE, numpage );
```

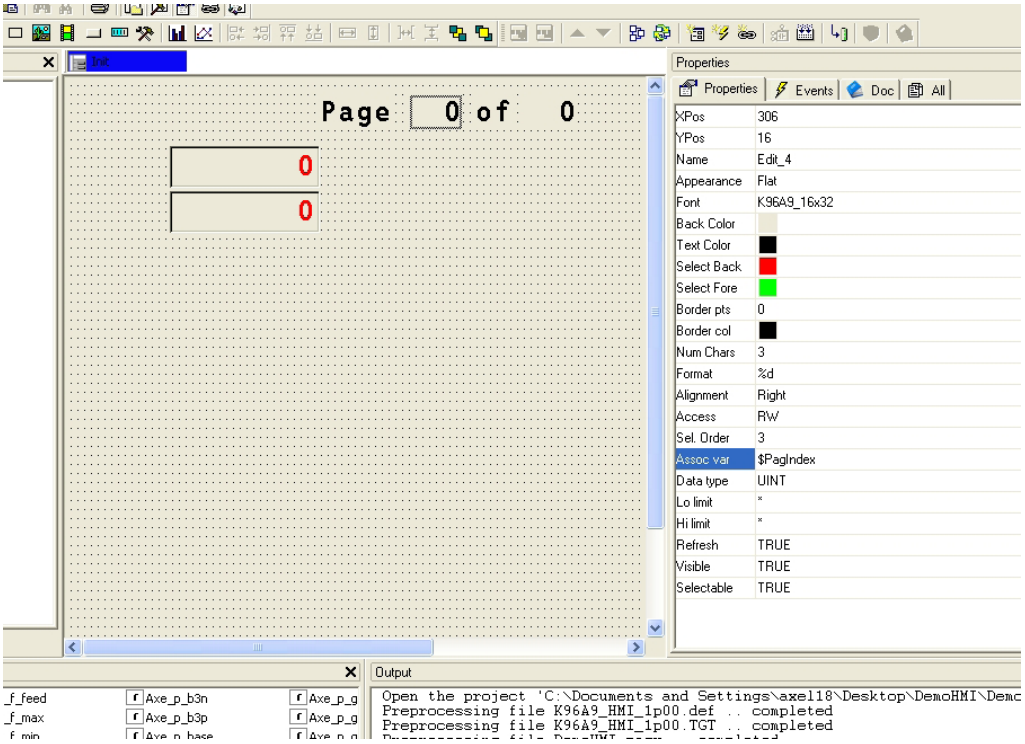
Where `numpage` is the number of the page of the set.

4.6.3 PAGES NUMBERING

UserInterface defines two variables related to pages numbering:

- *$PagIndex* = current index of the page containing controls based on a set;
- *$PagNumber* = number of pages that complete the visualization of the whole elements of the set.

These variables can be used in the page to show the numeration of the pages. In fact they can be used as variables associated to edit box controls in this way(1):



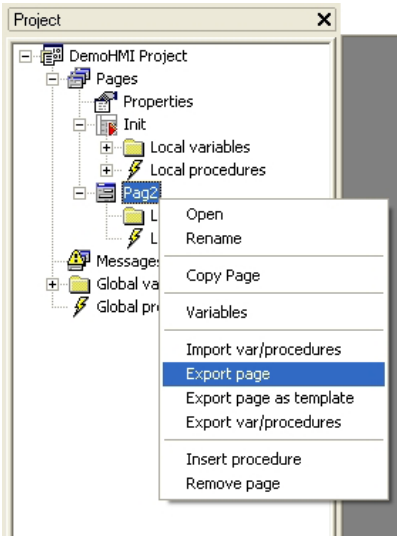
4.7 ADVANCED OPERATIONS ON PAGES

Advanced operations such as export/import, copy/paste and page based template management can be done with UserInterface. Next paragraphs show these arguments in details.

4.7.1 EXPORT/IMPORT OF PAGES TO/FROM FILES

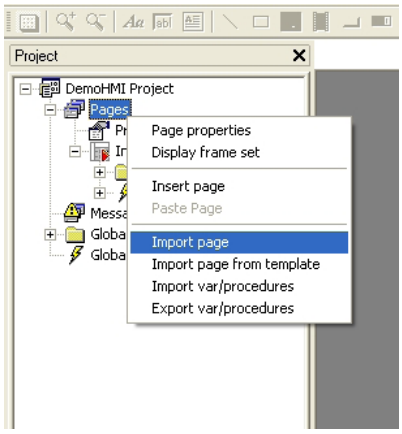
Each page, even if of a certain complexity, can be saved to be used later in other projects.

To do so click with the right button on the page node in the project window then select *Export page* from the menu:



Next, application asks user to insert the name of the file in which the page will be saved. This file assumes a *.pex* extension. Export file contains page info and local procedures.

Import operation is quite similar to the export operation. Select *Pages* node, click with right button and select *Import page* from the popup menu. User can then select the file of the page to import. Imported page takes the same name that it had when it was exported.

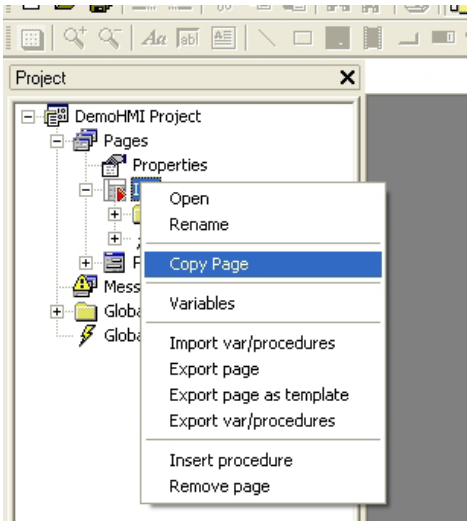


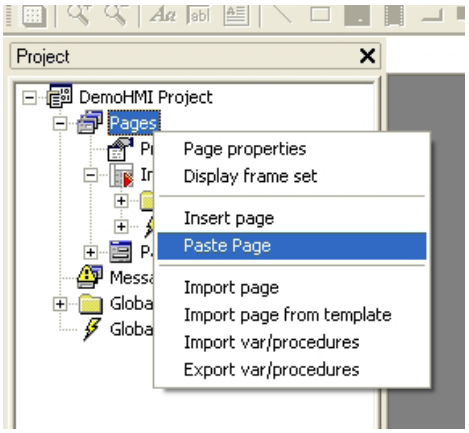
4.7.2 EXPORT/IMPORT PROCEDURES AND VARIABLES

It is also possible to export/import local or global variables and procedures using the menu commands *Export var/procedures* or *Import var/procedures*.

4.7.3 COPY/PASTE OF PAGES IN THE PROJECT

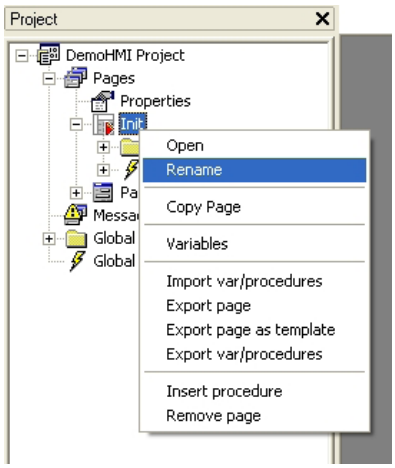
It is possible to copy and to paste a page inside the project. Select desired page, click with right button of the mouse and select *Copy Page* from the menu. Then, to paste page copied, select *Pages* node and select *Paste Page* from the menu.





4.7.4 RENAME PAGES

Select desired page from the project tree then click with the right mouse button and select *Rename* from the menu. This allows the user to change the name of the page.



N.B.: this operation changes only the name of the page, project references to the renamed page are not automatically updated.

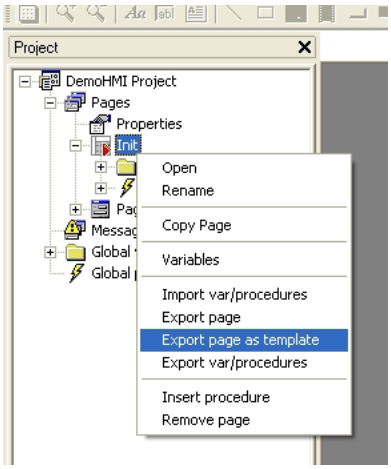
Templates of page management

4.7.5 TEMPLATES OF PAGE MANAGEMENT

Templates allow the user to save only the skeleton of the page and not the whole page. Templates can be described as pages without references to external variables. Templates can be grouped in libraries files (*.petx*) and can be linked into the project.

4.7.5.1 EXPORT PAGES INTO A TEMPLATE FILE

To export a page into a library of templates follow the procedure paragraph 4.7.1 initial steps then select *Export page as template* from the menu.



A library file with *.petx* extension (new or already existing) should be indicated. Template is appended to the existing templates and a name for the library is requested. If the template is already available in the library a message asks the user if he desires to rewrite the existing template or not.

Page is exported as template into the specified library with all its element but without any referenced variable.

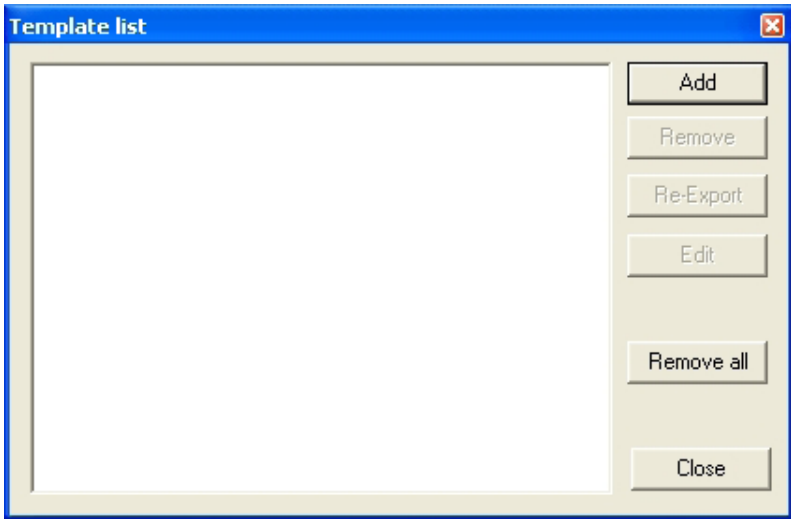
Scripts and local variables are exported without changes. References to variables contained in the scripts are not modified.

Child pages, popup and asynchronous messages can be treated as templates.

4.7.5.2 USAGE OF THE TEMPLATE LIBRARY IN A PROJECT

It is possible to include a template library in a project in order to use templates when desired.

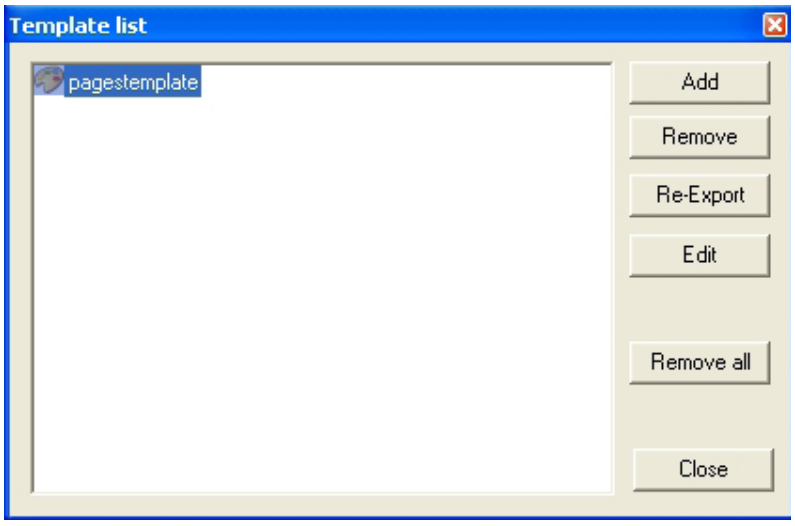
Select *Template Management* menu voice from *Project* menu. Following window will be shown:



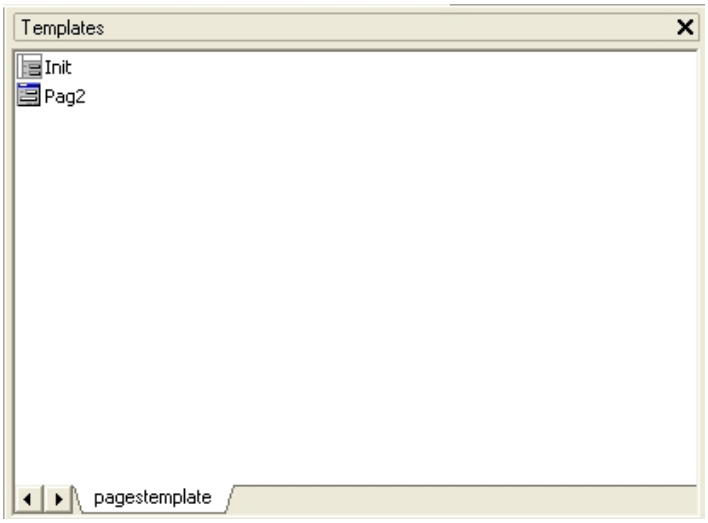
Available operations are listed here:

- Add: add a template library to the project. Including a library means that a reference to the library's *.petx* file is added to the current project, and that a local copy of the library is made.
- Remove: Remove template library from current project.
- Edit: To modify local copy of the template library removing no more used templates.
- Re-export: Export local copy of the template library into a new *.petx* library file.
- Remove All: Remove all template libraries from current project.

Now press the *Add* button to add a template library to the project. Once chosen one of the available libraries, *Template list* window appears as shown here.



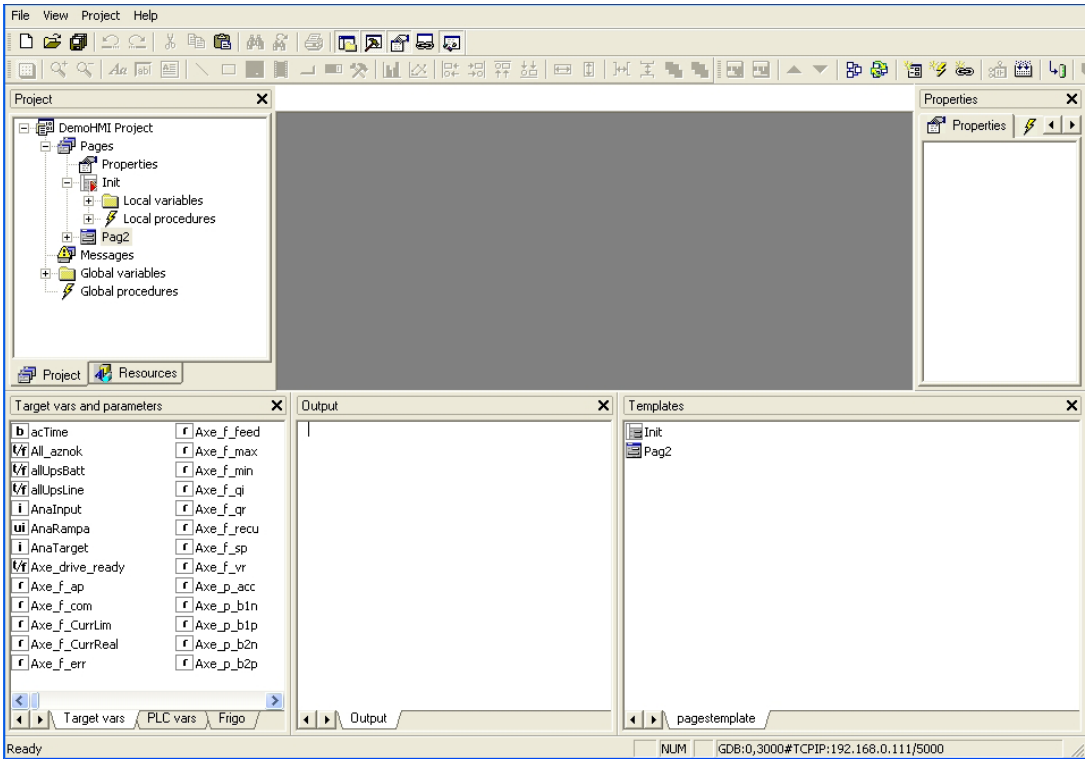
Template library has been included to the project. Press *Close* button, *Templates* window is shown: there is a tab for each library imported in current project.



Each tab shows the list of templates of the corresponding library.

4.7.5.3 USING A TEMPLATE

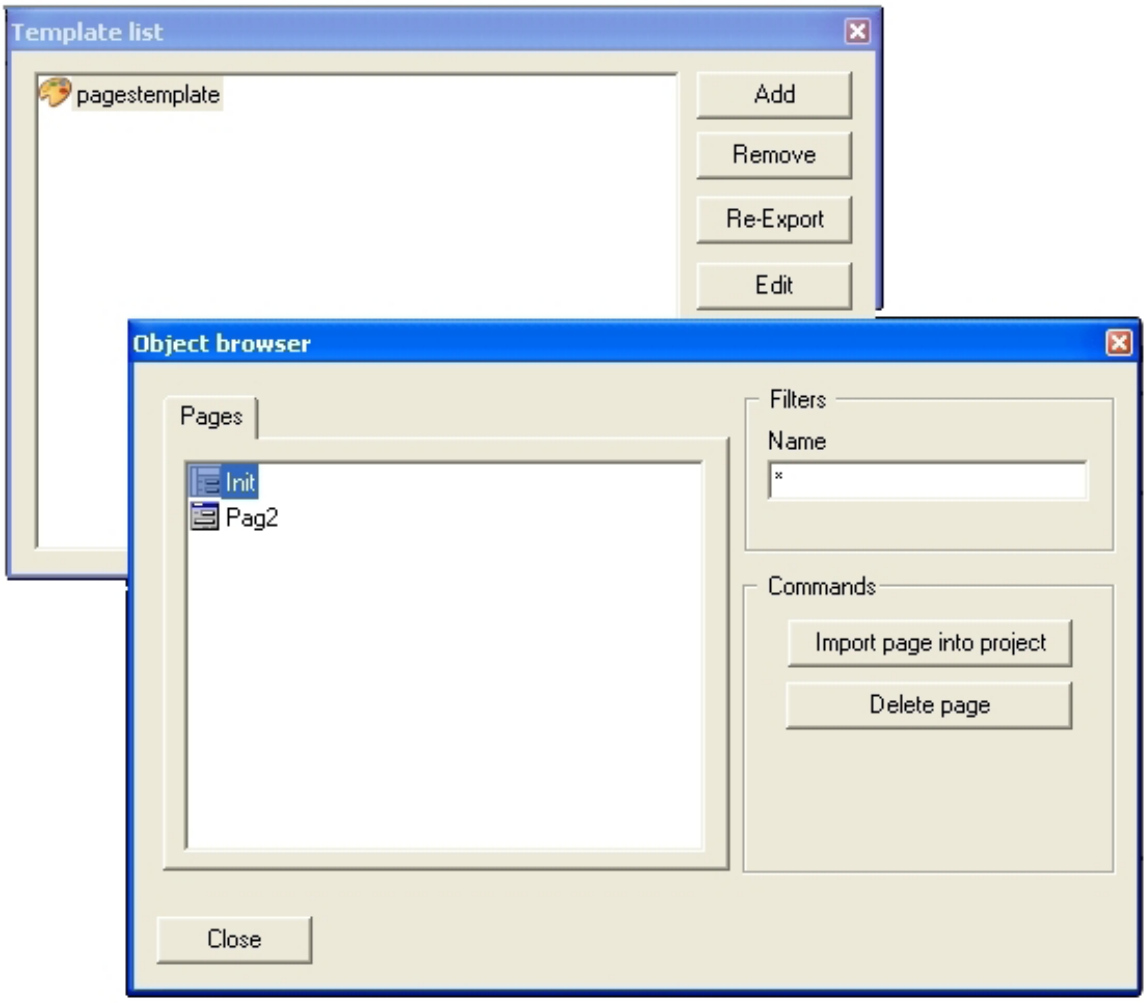
Once a template library has been added it is possible to use its elements simply dragging the chosen one from the template window and dropping it on the project tree.



Once the item has been dropped application asks the user for the name of the new page created (based on the template).

4.7.5.4 PROJECT TEMPLATE UPDATE

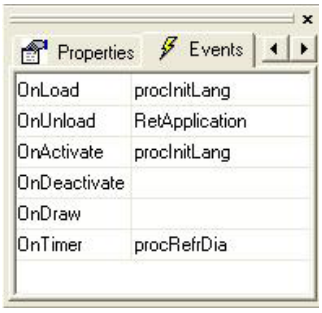
It is possible to delete templates from the (local) template library using *Edit* command in the *Template management* window.



4.8 EVENTS

There are different classes of events.

4.8.1 PAGE OR CONTROL EVENTS



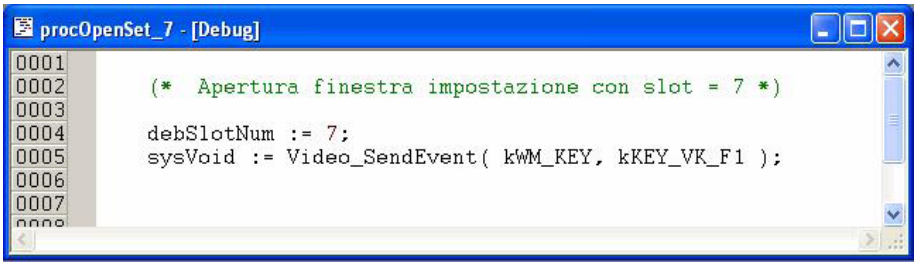
Each characteristic behaviour of a specific object can raise a specific event.

Each event can be associated to a procedure (see 4.8.4) that is executed each time the event takes place. The list of all available events for each UserInterface object (page or control) is reported in Chapter 5.

4.8.2 KEY PRESSURE EVENTS

These events take place when a key is pressed, the raising of the event starts the execution of the associated action (see 4.8.5) if it is. The pressure of a key can be also simulated by software, see next paragraph.

4.8.3 EVENTS RAISED BY SOFTWARE



Programmer can raise events by software using the function *Video_SendEvent* inside the target software or in the body of the procedure, using following syntax:

```
Video_SendEvent( event_id, param );
```

Where `event_id` is the identifier of the type of the event and param is an integer 16 bit parameter.

UserInterfacesupports software events defined in this table:

Event	Parameter	Description
kWM_NULL	Do not care	No event.
kWM_KEY	Key code	Simulates the pressure of the key specified as parameter then cause the associated action if it is.
kWM_MSG	Window ID	Causes a system message that, once got by the system, causes the instant opening of the alarm page that has *Window ID* as identifier.
kWM_SELECT	Edit box handle	In touchscreen systems simulates the pressure on the edit box whose handle is passed as parameter, causing its selection or its transition to edit mode.
kWM_PUSH	Button handle	In touchscreen systems simulates the pressure on the button whose handle is passed as parameter, causing the execution of the associated action if it is.
kEV_WM_ CHANGESETPAGE	Page number	Shows the page specified by the parameter (if the context is a page in which sets are used).

4.8.4 PROCEDURES THAT CAN BE ASSOCIATED TO EVENTS

A procedure is a program that is executed when the event that has been associated to it, takes place. Events have been deeply described in previous paragraphs (see 4.8).

There are two classes of procedures:

Local procedures

This kind of procedures can be called only within the scope of the page in which are declared. In particular, they can be associated to the events of the page itself and of all their controls. The same can be said for software events raised when the page they refer to is active. Procedure code can contains references to all the types of variables, with local variables of the page too.

Global procedures

This kind of procedures can be called from every page and can be also used as periodic asynchronous routine of alarm management. They cannot contain variables references.

Here follow the description of the syntax to get the properties of a control from a procedure; similarly to C language printf it is:

```
"fb%s%s.%s",   page_name, ctrl_name, prop_name
```

Where:

- `page_name` is the name of the page that has the control;
- `ctrl_name` is the name of the control;
- `prop_name` is the name of the property of the control.

So if we want to get the property *`Foreground color`* of the *`Static`* named *`String_26`* in *`Main`* page, we have to write:

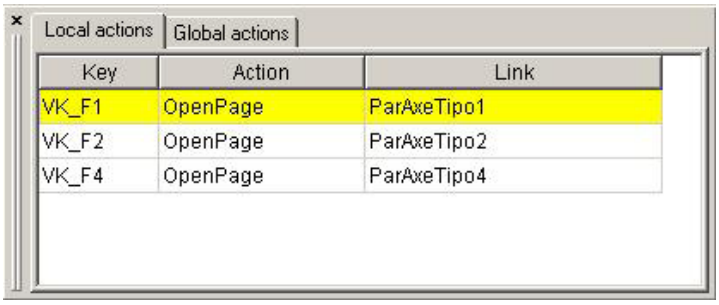
```
fbMainString_26.foreCol
```

N.B.: the name of the property to use in the scripts of the procedures is the name of the functional block exported by the software of the target (see 8.2), not the name in the properties window (see 3.3).

4.8.5 ACTIONS THAT CAN BE ASSOCIATED TO KEY PRESSURE

In common keyboard, not touchscreen systems, interaction between the user and the system is normally based on keys pressure.

UserInterfaceshows the following table to the user.



This table permits to associate a code of a key, defined in the file *`.def`*, to one of the actions listed in the following table. In this way the pressure of that key causes the specified action.

Also the names of the actions can be personalized by editing *`.def`* file.

Action	Link	Description
Call	Procedure name	Causes the invocation of the local or global procedure whose name is indicated in the *Link* field.
OpenPage	Page name	Causes the opening of the page whose name is indicated in the *Link* field.
Close	Do not care	Causes the closure of the current page
NextField	Do not care	Move the selection to the next edit box. If the system is not touchscreen moves selection to the buttons to allow their pressure.
PrevField	Do not care	Move the selection to the previous edit box.
Edit	Do not care	Access edit mode for the selected edit box. If the system is not touchscreen allows the user to simulate the pressure of the button.

There are two types of associations key-action:

- Local actions: local associations, valid only for the page currently open in the editor of the pages.
- Global actions: global associations, valid in any point of the project.

If the system has the touchscreen feature, normal interaction with user is made by the pressure of sensible area on the screen. However this table does not loss its meaning because allows the user to define virtual keys and to control their pressure by software causing in this way the dynamic execution of specific actions.

N.B.: if the same action is defined both at local and at global level, system does not give errors nor warnings because local declaration precedes global one.

4.9 RESOURCES

A resource is an interface element. User can get informations from resources or can use them to do actions.

UserInterface supports different categories of resources that are managed by *Tab Resources project* window. (see 3.1). Categories are explained in details in the following paragraphs.

4.9.1 FONTS

Fonts are the different types of characters supported for the output of text strings on the screen. In UserInterface there are two types of fonts:

- fonts imported: fonts declared in a *.plb* file included in the project and downloaded into the target board together with the pages code;
- fonts embedded: fonts already available on target board that have not to be downloaded with the project; this kind of fonts should be published to UserInterface in form of *.plf* file to let the application correctly draw the preview of the text strings in the page editor.

File with *.plf* extension defines one font; it has to be included in *.paj* project file with this synax:

```
FONT = "font_name"
```

At project opening time, if UserInterface finds this declaration, it searches in project folder for a file named `font_name.plf` and loads it in memory.

4.9.2 BITMAPS

Bitmaps are pictures to associate to image controls (see 4.4.5). Bitmaps had been managed by UserInterface old versions as text files with *.plb* extension and structured with the same syntax of the initialization definition of an array variable in IEC. Now, images are saved and loaded in binary format to optimize loading time on images of big size.

Bitmap definition files have to be included in project *.paj* file using this syntax:

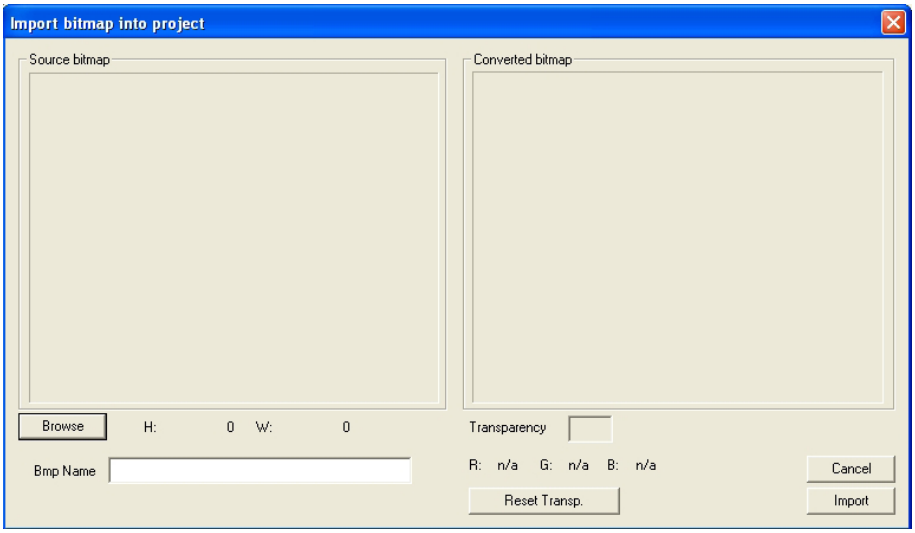
```
BITMAP = "bitmap_name.plk"
```

At project opening time, if UserInterface finds this declaration, it searches in project folder for a file named *bitmap_name.plk* and loads it in memory.

UserInterface provides a tool to convert bitmaps from Windows format to UserInterface format.

To start this tool click on *Import resource* > *Bitmap* from *Project* menu, it is also possible to click on *Import bitmap* from context menu that can be shown by right click on *Bitmaps* node of resources tree.

This dialogue box will be shown as follows.



Click on *Browse* button to navigate computer resources to select desired source file.

In *Bmp Name* field user can personalize bitmap name that will be shown on resource tree; bitmap name is constituted by file name without extension and with *Bmp* prefix by default.

Transparency color field allows the user to specify transparency color, so a color that is not really drawn on the screen but a transparent color zone that does not cover elements previously drawn.

Transparency color can be personalized by choosing it by mouse from *Converted bitmap* window.

RGB indicates transparency color Red, Green, Blue components. *n/a* value indicates that no transparency color has been selected.

Reset Transp. button allows the user to undo last selected transparency color.

Once finished these operations it is possible to confirm bitmap importation by clicking on *Import* button.

4.9.3 STRINGS TABLE

In a UserInterface project it is always possible to explicitly write the text to show on a text string or on a title of the page. It is also possible to refer to one of the strings of the resources specifying its ID.

In first case text will be always the same, in second case the text that correspond to the active language will be shown.

So, English language string table contains the following record.

ID_GDB_RXNAK	Bad RX packets

And Italian language string table contains the following record.

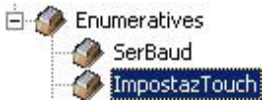
DB_RXNAK	Pacchetti RX errati

If we refer to the identifier *ID_GDB_RXNAK* from a page control or from a page, if current active language is English *Bad RX packets* will be shown, if current active language is Italian *Pacchetti RX errati* will be shown instead.

4.9.4 ENUMERATIVES

An enumerative is a data type defined by user, it is a set of constants named by user. Each element of an enumerative is treated as a constant and can be translated in all available languages of the project.

e.g.: we defined *ImpostazTouch* enumerative that is shown on resource tree as follows.



Enumerative records are shown by double clicking on *ImpostazTouch* node.

Value	Description
0	Attesa impostazione ...
1	Memorizzazione in corso ..
2	Touch screen impostato

Now we introduce an *Edit box* control (see 4.4.3) and insert the name of the enumerative *ImpostazTouch* in its *Format property* field. Control will show the string associated to the value as it is in the table above, not the numeric value of the variable associated to the control.

If the numeric value of the variable does not match with any record of the enumerative table, an error string `########` is shown instead.

Even enumerative are supported by multi-language feature. In fact it is possible to personalize the name of the enumerative.

And its record values.

Value	Description
0	Awaiting settings ...
1	Saving settings ...
2	Touch screen set up

4.9.5 IMAGES LISTS

An images list is very similar to an enumerative but with the following differences:

- intervals of constants are supported, not only simple values;
- each value has an image associated;
- a list of images determines the content shown by an *Animation* control, while an enumerative can be associated to an Edit box.

e.g.: now we have an images list *ListBulbs* that is shown on the resource tree.

It is possible to see all the records of the list by double-clicking the node.

Init Value	End Value	Bitmap
-10	0	BmpBulb5
0	5	BmpBulb6
5	10	BmpBulb7
*	*	BmpClose

If we introduce an *Animation* control (see 4.4.6) in the page, and we set its property *Image* list with the name of the enumerative *ListBulbs*, the control will show the image whose specified interval includes the value of a variable associated to the control.

If the numeric value of the associated variable does not match any record in the list a default image (with init and end value set to *) will be shown if it is. If no default image is specified no image will be drawn.

4.9.6 SETS

As it is described above (see 4.6) sets are ensemble of global variable even of distinct type.

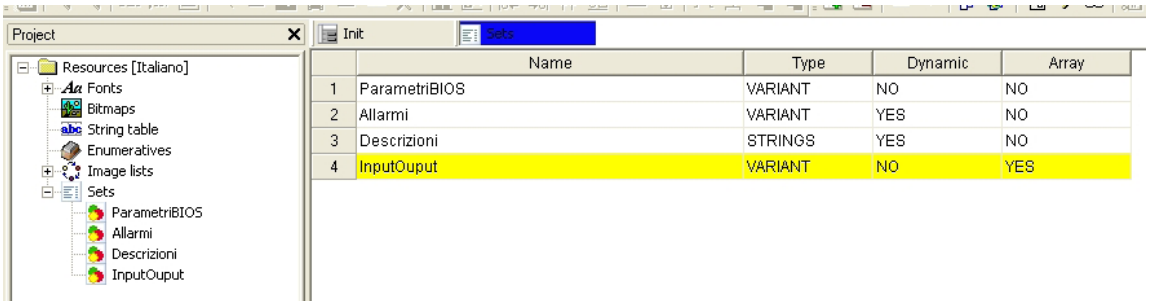
In particular there are two types of set:

- Variable/parameter sets even of not equal type (*VARIANT*);
- Strings sets (*STRINGS*).

The sets of the first type are defined indicating VARIANT as type. This kind of set has the following attributes:

- Dynamic: indicates that every n execution cycles target automatically reloads the elements of the set and hide those elements that have no visibility (boolean constant *FALSE* or associated visibility variable set to false at that moment).
- Array: indicates that the unique element of this set is a variable of type array.

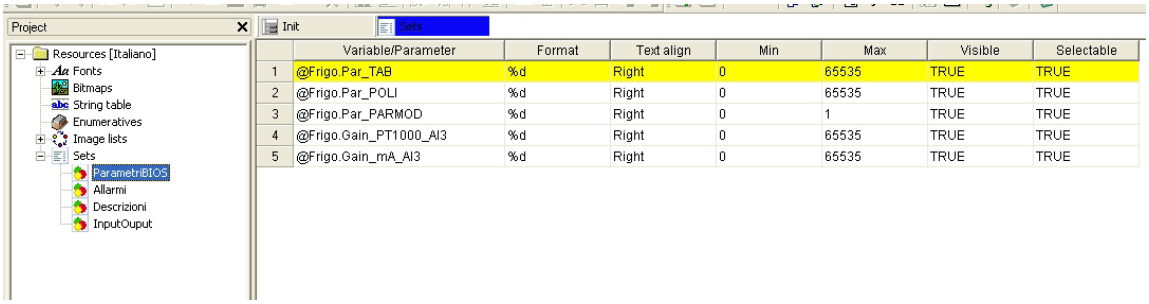
N.B.: this kind of set can be assigned only to an edit-box control.



In this example four sets with different characteristics have been defined.

Once defined a set, each element of the set can be added via drag & drop from *Target vars and parameters* or can be manually inserted by user.

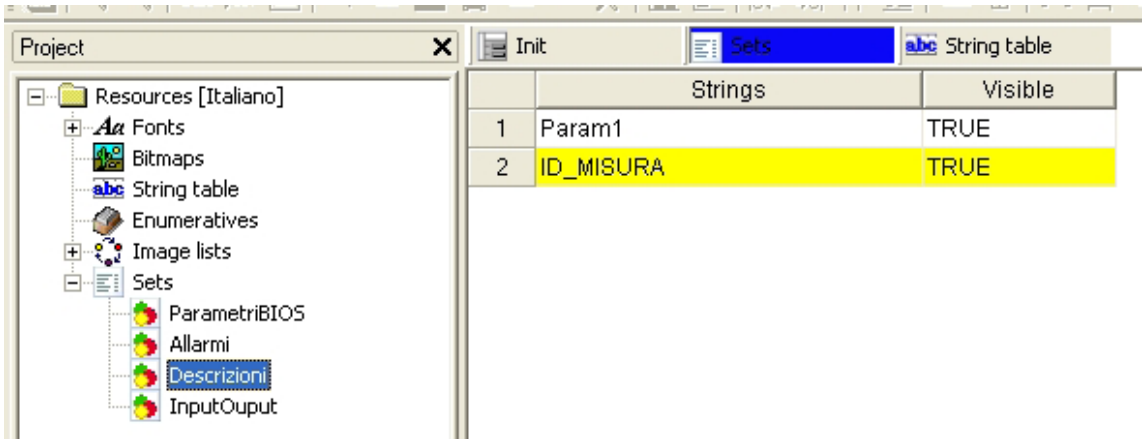
Lets see how to manage *ParametriBIOS* set.



Following attributes can be defined:

- Variable/Parameter = variable/parameter name.
- Format = indicates how to show associated variable value specifying a syntax analogous to C language printf (see paragraph 5.7.2).
- Text Align = the alignment of the text to show.
- Min/Max = minimum and maximum value for the element of the set.
- Visible = boolean variable or constant that defines the visibility of the element. If dynamic feature of the set is active the variable is periodically checked to hide or show the element.
- Selectable = indicates that the element can be selected. In this case a boolean variable or constant can be assigned too.

For a set of type STRING each element of the set is quite simple as it is shown in the next figure:



We have to define only two attributes, the string or the ID of a string resource (see 4.9.3) and the variable/constant of visibility. As we said an element not visible will not be shown on the screen.

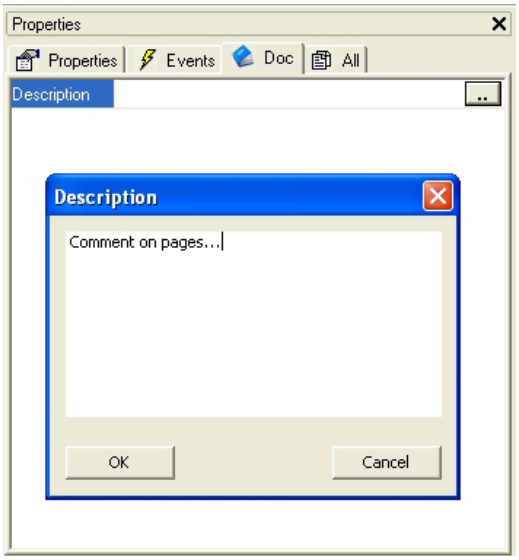
N.B.: this kind of set can be used with Static control only.

4.10 AUTOMATIC DOCUMENTATION

During project development it is usually necessary to write comments for each page in order to explain how the page works.

UserInterface integrates into its development environment the automatic documentation feature that consists in the generation of a graphical report with all the previously inserted comments followed by the pages they refer to.

Comments related to controls and pages should be inserted in the *Doc* tab of the properties window.



Documentation is generated when the apposite button is pressed.

At the end of the process the following dialogue is shown. By clicking on the *Open documentation* link it is possible to view the generated report using the browser.

It is also possible to manually open the *.html* file generated. This file is created in the project folder and is named *project name.html*.

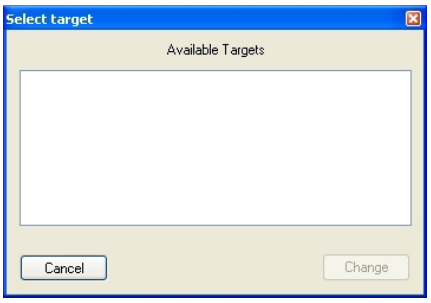
N.B.: documentation generation process requires the file *Documentation.xsl* to be in the project folder. This file can be personalized by user to redefine report style.

4.11 MANAGING PROJECTS

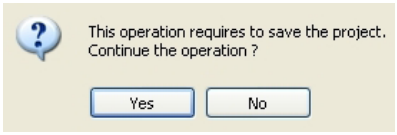
4.11.1 SELECTING THE TARGET DEVICE

You may need to port a PLC application on a target device which differs from that you originally wrote the code for. Follow the instructions below to adapt your UserInterface project to a new target device.

1) Click *Select target* in the *Project* menu of the UserInterface main window. This causes the following dialog box to appear.



2) Select one of the target devices listed in the combo box.
3) Click *Change* to confirm your choice, *Cancel* to abort.
4) If you confirm, UserInterface displays the following dialog box.



Press *Yes* to complete the conversion, *No* to quit.

If you press *Yes*, UserInterface updates the project to work with the new target.

It also makes a backup copy of the project file(s) in a sub-directory inside the project directory, so that you can roll-back the operation by manually (i.e., using Windows Explorer) replacing the project file(s) with the backup copy.

5. APPENDIX I: PAGE PROPERTIES AND OBJECT PROPERTIES

5.1 FRAME SET

5.1.1 PROPERTIES

Properties	Available values	Description
TopDim	>= 0	Top-height of the frame (#pixel).
BottomDim	>= 0	Bottom-height of the frame (#pixel).
LeftDim	>= 0	Left-width of the frame (#pixel).
RightDim	>= 0	Right-width of the frame (#pixel).
CharDimX	> 0	Horizontal space among grid points (#pixel).
CharDimY	> 0	Vertical space among grid points (#pixel).
Font	Name found in *Resources*	Default font used when inserting new objects in page.
Background Color	...	Background color selectable from palette. In addition this color is also set when inserting new objects in the frame.
Text Color	...	Foreground color selectable from palette. This color is set when inserting new objects in the frame.
Title bar	*Yes*, *No*	Title bar, settings can be found in *System options* dialog: - *Yes*: page has title; - *No*: page has not title.
Page Border	*Yes*, *No*	- *Yes*: page with outer border; - *No*: page without outer border.
Caption	Text otherwise *Resource ID*	Text on title bar or *Resource ID*. This property is not sensible if *Title Bar* field is set to *No*.
System menu	*Yes*, *No*	If *Yes* denotes that there is a button with 'X' image on it and the behaviour is similar to *Windows Dialog*: - *Yes*: page has close button; - *No:* page has not close button.
Appearance	*Flat*, *Raised*, *Sunken*	- Flat - Raised - Sunken

5.2 CHILD PAGE

5.2.1 PROPERTIES

Properties	Available values	Description
CharDimX	> 0	Horizontal space among grid points (#pixel).
CharDimY	> 0	Vertical space among grid points (#pixel).
Font	Name found in *Resources*	Default font used when inserting new objects in page.
Background Color	...	Background color selectable from palette. In addition this color is also set when inserting new objects in the frame.
Text Color	...	Foreground color selectable from palette. This color is set when inserting new objects in the frame.
Title bar	*Yes*, *No*	Title bar, settings can be found in *System options* dialog: - *Yes*: page has title; - *No*: page has not title.
Page Border	*Yes*, *No*	- *Yes*: page with outer border; - *No*: page without outer border.
Caption	Text otherwise *Resource ID*	Text on title bar or *Resource ID*. This property is not sensible if *Title Bar* field is set to *No*.
System menu	*Yes*, *No*	If *Yes* denotes that there is a button with 'X' image on it and the behaviour is similar to *Windows Dialog*: - *Yes*: page has close button; - *No:* page has not close button.
Appearance	*Flat*, *Raised*, *Sunken*	- Flat - Raised - Sunken

5.2.2 EVENTS

Event	Description
OnLoad	On loading this page, i.e. when calling from parent page.
OnUnload	On closing this page, when the page returns and the parent page will be restored.

Event	Description
OnDeactivate	On calling a child page and the current page is no more active. This event does not exist in main page.
OnActivate	When the previous opened child page will be closed. This event does not appear in leaf page, i.e in the pages which do not call child pages.
OnDraw	When the page starts drawing all the objects. The page has just drawn border, background, and title.
OnTimer	Asynchronous event. The user can link a procedure and it will be executed cyclically.

5.3 POP-UP PAGE

5.3.1 PROPERTIES

Properties	Available values	Description
XPos	>= 0	Top-left 'x coordinate' edge of full page.
YPos	>= 0	Top-left 'y coordinate' edge of full page.
XDim	> 0	Width of the page (#pixel).
YDim	> 0	Height of the page (#pixel).
CharDimX	> 0	Horizontal space among grid points (#pixel).
CharDimY	> 0	Vertical space among grid points (#pixel).
Modal	*Yes, No*	- *Yes*: all the parent page objects will be disabled; - *No*: all the parent page objects will be enabled if they are completely visible.
Font	Name found in *Resources*	Default font used when inserting new objects in page.
Background Color	...	Background color selectable from palette. In addition this color is also set when inserting new objects in the frame.
Text Color	...	Foreground color selectable from palette. This color is set when inserting new objects in the frame.
Title bar	*Yes, No*	Title bar, the settings can be found in *System options* dialog: - *Yes*: page has title; - *No*: page has not title.
Page Border	*Yes, No*	- *Yes*: page with outer border; - *No*: page without outer border.
Caption	Text otherwise *Resource ID*	Text on title bar or *Resource ID*. This property is not sensible if the *Title Bar* field is set to *No*.

Properties	Available values	Description
System menu	*Yes, No*	If *Yes* denotes that there is a button with *X* image on it and the behaviour is similar to *Windows Dialog:* - *Yes*: page has close button; - *No*: page has not close button.
Appearance	*Flat, Raised, Sunken*	Flat Raised Sunken

5.3.2 EVENTS

Event	Description
OnLoad	On loading this page, i.e. when calling from parent page.
OnUnload	On closing this page, when the page returns and the parent page will be restored.
OnDeactivate	On calling a child page and the current page is no more active. This event does not exist in main page.
OnActivate	When the previous opened child page will be closed. This event does not appear in leaf page, i.e in the pages which do not call child pages.
OnDraw	When the page starts drawing all the objects. The page has just drawn border, background and title.
OnTimer	Asynchronous event. The user can link a procedure and it will be executed cyclically.

5.4 STATIC

5.4.1 PROPERTIES

Properties	Available values	Description
XPos	>= 0	Top-left 'x coordinate' edge relative to page.
YPos	>= 0	Top-left 'y coordinate' edge relative to page.
Name	Not empty	Name of object.
Text	Text otherwise *Resource ID*	Text or *Resource ID* shown in the object.
Font	Name found in *Resources*	Font used for drawing the text in object.
Background Color	...	Background color selectable from palette.
Text Color	...	Text color selectable from palette.
Sel. Background	...	Background color selectable from palette when the object is chosen. This property is not sensible if the *Select* field is constant *FALSE*.

Properties	Available values	Description
Sel. Foreground	...	Text color selectable from palette when the object is chosen. This property is not sensible if the *Select* field is constant *FALSE*.
Appearance	*Flat, Raised, Sunken*	- Flat - Raised - Sunken
Border points	>= 0	Border thickness (#pixel). This property is sensible only if *Appearance* is set to *Flat*.
Border color	...	Border color selectable from palette. This property is sensible only if *Appearance* is set to *Flat*.
Number of Chars	> 0	Number of chars that this object can show. If the value is *0* the object will show the complete text. Otherwise with another value it can be truncated or extended.
Alignment	*Right, Center, Left*	Text alignment in the object.
Refresh	*TRUE, FALSE*	Continuous redraw of the object: - *FALSE*: the *Text value* is read from memory and updated only when opening the page or when a child page is closed; - *TRUE:* the *Text value* is read from memory and always updated.
Select	*TRUE, FALSE, var_name*	Selected status of the object. It can be constant (*TRUE* or *FALSE*) or linked with a boolean variable *var_name:* if *var_name* is *TRUE* the object is selected and so it will show the colors *Select Back*, *Select Fore*.
Visible	*TRUE, FALSE, var_name*	Visible status of the object. It can be constant (*TRUE* or *FALSE*) or linked with a boolean variable *var_name*: if *var_name* is *TRUE* the object is visible, otherwise it is hidden.

5.4.2 EVENTS

Event	Description
BeforeUpdate	Before the object is redrawn.
AfterUpdate	Immediately after the object is redrawn.

5.5 LINE

5.5.1 PROPERTIES

Properties	Available values	Description
XPos	>= 0	Top-left 'x coordinate' edge relative to page.
YPos	>= 0	Top-left 'y coordinate' edge relative to page.

Properties	Available values	Description
X2Pos	> 0	Bottom-right 'x coordinate' edge relative to page.
Y2Pos	> 0	Bottom-right 'y coordinate' edge relative to page.
Name	Not empty	Name of object.
Thickness pts	> 0	Line thickness (#pixel).
Border col	...	Line color selectable from palette.

5.6 RECTANGLE

5.6.1 PROPERTIES

Properties	Available values	Description
XPos	>= 0	Top-left 'x coordinate' edge relative to page.
YPos	>= 0	Top-left 'y coordinate' edge relative to page.
XDim	> 0	Width (#pixel).
YDim	> 0	Height (#pixel).
Name	Not empty	Name of object.
Border points	> 0	Border thickness (#pixel).
Border color	...	Border color selectable from palette.
Background Color	...	Background color selectable from palette. This property is sensible only if *Transparent* is set to *TRUE*.
Transparent	*TRUE*, *FALSE*	Transparency: - *TRUE:* transparent background; - *FALSE:* solid background where color is *Back Color*.

5.7 EDIT BOX

5.7.1 PROPERTIES

Properties	Available values	Description
XPos	>= 0	Top-left 'x coordinate' edge relative to page.
YPos	>= 0	Top-left 'y coordinate' edge relative to page.
Name	Not empty	Name of object.
Appearance	*Flat*, *Raised*, *Sunken*	- *Flat:* plain with use of *Border pts* and *Border col;* - *Raised*; - *Sunken*.
Font	Name found in *Resources*	Font used for drawing the text in object.

Properties	Available values	Description
Background Color	...	Background color selectable from palette.
Text Color	...	Text color selectable from palette.
Sel. Background	...	Background color selectable from palette when the object is chosen. This property is not sensible if the *Selectable* field is constant *FALSE*.
Sel. Foreground	...	Text color selectable from palette when the object is chosen. This property is not sensible if the *Selectable* field is constant *FALSE*.
Border points	>= 0	Border thickness (#pixel). This property is sensible only if *Appearance* is set to *Flat*.
Border color	...	Border color selectable from palette. This property is sensible only if *Appearance* is set to *Flat*.
Number of Chars	> 0	Chars visible in the object. Width of entire object is calculated among this value and the size of *Font*. If *NumChar* are less than the value, the object shows this error string: `#####`.
Format	String as `printf` or `.enum_name`	The format can be numeric, to define as `printf` of C language (see 5.7.2), enumerative, if in this field there is `enum_name` defined in *Resources* (see 4.9).
Alignment	*Right, Center, Left*	Text alignment in the object.
Access	*RO, RW*	Accesses variable *Assoc var* used in object: - *RO* = read only; - *RW* = read/write.
Selection Order	>= 0	Selection order of the object. It can be selected by pressing a key or by means of a procedure. In this case the selection moves from the current object to the previous or next *Sel. Order* object.
Variable	Not empty	Name of the variable that can be shown and edited with this object. It can be any variable of the project, (local, global, imported from PLC or target - see 2.9.2), a parameter (see 2.9.2) or an element of a set (see 4.6).
Data type	*UNDEF, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, STRING*	Type of *Assoc var*. If it is a variable, the type is defined automatically. This property is sensible if *Assoc var* is an explicit parameter.

Properties	Available values	Description
Low limit	*CONSTANT, var_name*	Name of variable or numeric constant. This is the least number that the object can show. It can be any variable of the project, (local, global, imported from PLC or target - see 2.9.2). This object shows an error string (!!!!!!!) if condition does not holds. The * symbol means that there is no low limit.
High limit	*CONSTANT, var_name*	Name of the variable or numeric constant. This is the maximum number that the object can show. It can be any variable of the project, (local, global, imported from PLC or target - see 2.9.2). This object views an error string (*!!!!!!!*) if condition does not hold. The * symbol means that there is no high limit.
Refresh	*TRUE, FALSE*	Enables continuous update of the value: - *FALSE:* the *Assoc var* value is read from memory and updated only when open page or when a child page is closed; - *TRUE*: the *Assoc var* value is read from memory and always updated.
Visible	*TRUE, FALSE, var_name*	Visible status of the object. It can be constant (*TRUE* or *FALSE*) or linked with a boolean variable *var_name*: if *var_name* is *TRUE* the object is visible, otherwise hidden.
Selectable	*TRUE, FALSE, var_name*	Selected status of the object. It can be constant (*TRUE* or *FALSE*) or linked with a boolean variable *var_name*: if *var_name* is *TRUE* the object is selected and so it will show the colors *Select Back*, *Select Fore*. If this field is *FALSE* the *Access* property is not sensible.

5.7.2 FORMAT SPECIFICATION - PRINTF

If the object has not any enumerative format, the format string is composed as follows:

```
%[flags][width][.precision]type
```

The field has one or more characters, that describe the specification. The simplest format contains only percentage symbol and one char as type (for example: *%s*).

Next table explains in details functions and values.

Field	Available values	Description
flags	- + prints always the sign, even if the number is positive. - 0 prints zeros in head until *width* (if specified) or *NumChar*.	This char is an option for chars order, print sign, number of decimal digit. This field may have more than one flag.

Field	Available values	Description
width	> 0, <= *NumChar*	Maximum chars can be printed. Allows to view values that do not fill *NumChar* fully.
precision	>= 0	Decimal digits after the point. If the field is an integer and there is a precision the object shows a decimal point. E.g. the value is 102 integer, and precision is 2, with %.2d, the number is shown as 1.02.
type	- %d: Integer with sign. - %f: Real. - %x: Hexadecimal with lowercase chars. - %x: Hexadecimal with uppercase chars. - %s: String. - %@sdf: Password. - [%d,u,f,x]: Custom measure unit format.	Mandatory field.

5.7.3 EVENTS

Event	Description
BeforeUpdate	Before the object is redrawn.
AfterUpdate	Immediately after the object is redrawn.
OnGotFocus	Whenever object is selected.
OnLostFocus	Whenever object loses the selection.
OnEnter	Whenever the object is selected and receives the command for entering in edit-mode.
OnClick	Whenever HMI receives a pressure on the object, valid only for touchscreen systems.
OnChange	Whenever the user confirms the modifications and the value is different from start.

5.8 TEXT BOX

5.8.1 PROPERTIES

Properties	Available values	Description
XPos	>= 0	Top-left 'x coordinate' edge relative to page.
YPos	>= 0	Top-left 'y coordinate' edge relative to page.
Name	Not empty	Name of object.
Appearance	*Flat, Raised, Sunken*	- *Flat*: plain with use of *Border pts* and *Border col*; - *Raised*; - *Sunken*.

Properties	Available values	Description
Font	Name found in *Resources*	Font used for drawing the text in the object.
Background Color	...	Background color selectable from palette.
Text Color	...	Text color selectable from palette.
Border points	>= 0	Border thickness (#pixel). This property is sensible only if *Appearance* is set to *Flat*.
Border color	...	Border color selectable from palette. This property is sensible only if *Appearance* is set to *Flat*.
Number of Chars	> 0	Chars visible in the object. Width of entire object is calculated among this value and the size of *Font*.
Number of Rows	> 0	Rows visible in the object. Height of entire object is calculated among this value and the size of *Font*.
Show line number	*TRUE, FALSE*	Flag for viewing number of lines.
Access	*RO, RW*	Access on variable *Assoc string* used in object: - *RO:* read only; - *RW*: read/write.
Selection order	>= 0	Selection order on which the object can be selected with the pressure of a key or with a procedure. In this case the selection moves from the current object to the previous or next *Sel.Order object*.
String variable	*Not empty*	Name of variable that can be shown and edited with this object. It can be any string variable of the project, (local, global, imported from PLC or target - see 2.9.2).
Refresh trg	*TRUE, FALSE, var_name*	Enables update of the value: - *FALSE*: the *Assoc string* value is read from memory and updated only when opening page or when a child page is closed. - *TRUE*: the *Assoc string* value is read from memory and always updated. The runtime sets automatically the value to *FALSE*.
Visible	*TRUE, FALSE, var_name*	Visible status of the object. It can be constant (*TRUE* or *FALSE*) or linked with a boolean variable *var_name*: if *var_name* is *TRUE* the object is visible, otherwise hidden.

5.8.2 EVENTS

Event	Description
BeforeUpdate	Before the object is redrawn.
AfterUpdate	Immediately after the object is redrawn.
OnClick	Whenever HMI receives a pressure on the object, valid only for touchscreen systems.
OnChange	Whenever the user confirms the modifications and the value is different from start.

5.9 IMAGE

5.9.1 PROPERTIES

Properties	Available values	Description
XPos	const >= 0, variable	Top-left 'x coordinate' edge relative to page. It is possible to assign a variable only if *Style* is set to *Floating*.
YPos	const >= 0, variable	Top-left 'y coordinate' edge relative to page. It is possible to assign a variable only if *Style* is set to *Floating*.
XDim	> 0	Width (#pixel).
YDim	> 0	Height (#pixel).
Name	Not empty	Name of object.
Appearence	*Flat, Raised, Sunken*	- *Flat* = plain with use of *Border pts* and *Border col*; - *Raised*; - *Sunken*.
Border points	>= 0	Border thickness (#pixel). This property is sensible only if *Appearance* is set to *Flat*.
Border color	...	Border color selectable from palette. This property is sensible only if *Appearance* is set to *Flat*.
Bitmap	Name found in *Resources*	Bitmap used for drawing the image in object.
Background image	Image object in the page	Name of another object that is redrawn when *Style* is set to *Floating*. It is sensible only if it is overlapped with this image.
Visible	*TRUE, FALSE, var_name*	Visible status of the object. It can be constant (*TRUE* or *FALSE*) or linked with a boolean variable *var_name*: if *var_name* is *TRUE* the object is visible, otherwise it is hidden.
Style	*Docking, Floating*	- *Docking*: fixed position; - *Floating*: variable position, according to *XPos* variable and *Ypos* variable.

5.10 ANIMATION

5.10.1 PROPERTIES

Properties	Available values	Description
XPos	>= 0	Top-left 'x coordinate' edge relative to page.
YPos	>= 0	Top-left 'y coordinate' edge relative to page.
XDim	> 0	Width (#pixel).
YDim	> 0	Height (#pixel).
Name	Not empty	Name of object.
Appearance	*Flat, Raised, Sunken*	- *Flat* = plain with use of *Border pts* and *Border col*; - *Raised*; - *Sunken*.
Border points	>= 0	Border thickness (#pixel). This property is sensible only if *Appearance* is set to *Flat*.
Border color	...	Border color selectable from palette. This property is sensible only if *Appearance* is set to *Flat*.
Image list	Name found in *Resources*	It contains the images that the object can view and the value range.
Animation variable	*var_name*	Name of the variable that is compared with value range in *Image list*.
Data type	*SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD*	Type of *Animation var*. If it is a variable, the type is automatically defined.
Visible	*TRUE, FALSE, var_name*	Visible status of the object. It can be constant (*TRUE* or *FALSE*) or linked with a boolean variable *var_name*: if *var_name* is *TRUE* the object is visible, otherwise hidden.

5.10.2 EVENTS

Event	Description
BeforeUpdate	Before the object is redrawn.
AfterUpdate	Immediately after the object is redrawn.

5.11 BUTTON

5.11.1 PROPERTIES

Properties	Available values	Description
XPos	>= 0	Top-left 'x coordinate' edge relative to page.
YPos	>= 0	Top-left 'y coordinate' edge relative to page.
XDim	> 0	Width (#pixel).
YDim	> 0	Height (#pixel).
Name	Not empty	Name of object.
Text/Img	Empty or explicit text or Resource ID or Bitmap	Text or image to view in the button: - string; - Resource ID; - bitmap.
Selection Text/Img	Empty or explicit text or Resource ID or Bitmap	Text or image to view in the button when it is selected: - string; - Resource ID; - bitmap.
Font	Name found in *Resources*	Font used for drawing the text in object. This field is not sensible if it shows a bitmap.
Appearance	Flat, *Raised*, *Sunken*	- *Flat*: plain with use of *Border pts* and *Border col*; - *Raised*; - *Sunken*.
Border points	>= 0	Border thickness (#pixel). This property is sensible only if *Appearance* is set to *Flat*.
Border color	...	Border color selectable from palette. This property is sensible only if *Appearance* is set to *Flat* or *Text* is not empty.
Background color	...	Background color selectable from palette. This property is sensible only if *Transparent* is set to *TRUE*.
Selection border	...	Border color when the object is selected. This property is not sensible if *Selection var* is *FALSE* fixed.
Sel. background	...	Background color when the object is selected. This property is not sensible if *Selection var* is *FALSE* fixed.

Properties	Available values	Description
Selection order	>= 0	Selection order on which the object can be selected with the pressure of a key or with a procedure. In this case the selection moves from the current object to the previous or next *Sel. Order* object.
Visible	*TRUE*, *FALSE*, *var_name*	Visible status of the object. It can be constant (*TRUE* or *FALSE*) or linked with a boolean variable *var_name*: if *var_name* is *TRUE* the object is visible, otherwise it is hidden.
Transparent	*TRUE*, *FALSE*, *var_name*	Transparency. It can be constant (*TRUE* or *FALSE*) or linked with a boolean variable *var_name*: if *var_name* is *TRUE* the object is transparent.
Press variable	Empty or *var_name*	When the button is pressed *var_name* is set to *TRUE*. When the button is not pressed, *var_name* is set to *FALSE*.
Selection variable	*TRUE*, *FALSE*, *var_name*	Selected status of the object. It can be constant (*TRUE* or *FALSE*) or linked with a boolean variable *var_name*: if *var_name* is *TRUE* the object is selected and so it will show the colors *Select Back*, *SelectBord*. If this field is *FALSE SelectBord* and *Select Back* properties are not sensible.
Action	*Call*, *OpenPage*, *Close*, *NextField*, *PrevField*, *Edit*	Action executed on button pressure.
Action par	*page_name* *proc_name*	Parameter associated with the action executed on button pressure. It is sensible only if *Action* is *OpenPage* (*Action par* = name of the page to open) or *Call* (*Action par* = name of the procedure to execute).
Alignment	*Right*, *Center*, *Left*	Text alignment in the object.

5.11.2 EVENTS

Event	Description
OnClick	Whenever HMI receives a pressure on the object, valid only for touchscreen systems.
OnRelease	Whenever HMI releases the pressure on the object, valid only for touchscreen systems.

5.12 PROGRESS BAR

5.12.1 PROPERTIES

Properties	Available values	Description
XPos	>= 0	Top-left 'x coordinate' edge relative to page.
YPos	>= 0	Top-left 'y coordinate' edge relative to page.
XDim	> 0	Width (#pixel).
YDim	> 0	Height (#pixel).
Name	Not empty	Name of object.
Appearance	*Flat*, *Raised*, *Sunken*	- *Flat:* plain with use of *Border pts* and *Border col*; - *Raised*; - *Sunken*.
Border points	>= 0	Border thickness (#pixel). This property is sensible only if *Appearance* is set to *Flat*.
Border color	...	Border color selectable from palette. This property is sensible only if *Appearance* is set to *Flat* or *Text* is not empty.
Bar color	...	Color of step bar, selectable from palette.
Background color	...	Background color selectable from palette.
Visible	*TRUE*, *FALSE*, *var_name*	Visible status of the object. It can be constant (*TRUE* or *FALSE*) or linked with a boolean variable *var_name*: if *var_name* is *TRUE* the object is visible, otherwiseit is hidden.
Refresh trigger	*TRUE*, *FALSE*, *var_name*	Object redraw: - *FALSE:* the *Progress var* value is read from memory and updated only when opening page or when a child page is closed. - *TRUE:* the *Progress var* value is read from memory and always updated. - *var_name:* the *Progress var* value is read from memory and updated only when the variable becomes *TRUE*. After the update the runtime sets it to *FALSE*.
Progress variable	Not empty	Step variable. This is the filling percentage of bar in relation with the range assigned by *Lo limit* and *Hi limit*. It can be any string variable of the project (local, global, imported from PLC or target) or a parameter (see 2.9.2).
Data type	UNDEF, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, LWORD, REAL, LREAL, STRING	Type of *Progress var*. If it is a variable, the type is automatically defined. This property is sensible if *Progress var* is an explicit parameter.

Properties	Available values	Description
Low limit	Constant or *var_name*	Name of the variable or numeric constant. This is the least value for step bar. It can be any variable of the project, (local, global, imported from PLC or target - See. §) with type specified by *Data type*.
High limit	Constant or *var_name*	Name of the variable or numeric constant. This is the maximum value for step bar. It can be any variable of the project, (local, global, imported from PLC or target - See. §) with type specified by *Data type*.
Orientation	Horizontal, vertical	Direction of step bar.

5.12.2 EVENTS

Event	Description
BeforeUpdate	Before the object is redrawn.
AfterUpdate	Immediately after the object is redrawn.

5.13 CUSTOM CONTROL

5.13.1 PROPERTIES

Properties	Available values	Description
XPos	>= 0	Top-left 'x coordinate' edge relative to page.
YPos	>= 0	Top-left 'y coordinate' edge relative to page.
XDim	> 0	Width (#pixel).
YDim	> 0	Height (#pixel).
Name	Not empty	Name of object.
Control ID	> 0	Identifier of custom control type.
Visible	*TRUE*, *FALSE*, *var_name*	Visible status of the object. It can be constant (*TRUE* or *FALSE*) or linked with a boolean variable *var_name*: if *var_name* is *TRUE* the object is visible, otherwise it is hidden.
Refresh	*TRUE*, *FALSE*	Continuous redraw of the object: - *FALSE*: the body of the runtime object is updated only when opening page or when a child page is closed. - *TRUE:* the body of the runtime object is always updated.

5.13.2 EVENTS

Event	Description
BeforeUpdate	Before the object is redrawn.
AfterUpdate	Immediately after the object is redrawn.

5.14 CHART

5.14.1 PROPERTIES

Properties	Available values	Description
XPos	>= 0	Top-left 'x coordinate' edge relative to page.
YPos	>= 0	Top-left 'y coordinate' edge relative to page.
XDim	> 0	Width (#pixel).
YDim	> 0	Height (#pixel).
Name	Not empty	Name of object.
Track 1	*var_name*	Opens a dialog with the following options: - the array with data of track (*Data Source*); - the visibility condition of the track (*TRUE* or boolean variable); - *Color* of the track; - the scale factor (range among two horizontal divisions); - the offset (displacement of the track 0-Y); - step of print label for Y axis; - three horizontal bars with name, value and colors. If *var_name* is empty the track is not defined and not drawn.
Track 2	*var_name*	As *Track 1*, but for track 2.
Track 3	*var_name*	As *Track 1*, but for track 3.
Track 4	*var_name*	As *Track 1*, but for track 4.
Track 5	*var_name*	As *Track 1*, but for track 5.
Track 6	*var_name*	As *Track 1*, but for track 6.
Track 7	*var_name*	As Track 1, but for track 7.
Track 8	*var_name*	As *Track 1*, but for track 8.
Track Left	>=0, *var_name*	Integer value that is the track for Y axis left. It is admitted a constant value (ex 1). If empty means that the chart must not draw the label for Y axis left.
Track Right	>=0, *var_name*	As *Track Left* for the right side.
Format Left	String as *printf*	Format of Y axis left as c *printf* function.
Format Right	String as *printf*	As *Format Left* for the right side.
XLabel	>=0, *var_name*	Step for X-axis labels. How many divisions of horizontal bar must have labels.

Properties	Available values	Description
X Scale	*var_name*	Scale factor of x-Axis. Value range among two divisions of horizontal bars. An empty value indicates that the chart is in autoscale mode.
Len Data	*var_name*	Name of the variable that contains the array index samples. The chart adds the sample values when the *Refresh* is *TRUE*. If the value remains unchanged the chart does not add new values. The runtime maintains the last value of this field. Constant values are not allowed.
X Offset	*var_name*	Variable name for the deviation of 0 for x-Axis, left or right. A positive value moves the chart values to left.
Grid	*Yes, No*	Visibility of the grid.
X Div. Grid	value	Number of division on horizontal bar, used with scale factor and offset for drawing the chart tracks.
Y Div. Grid	value	Number of division on vertical bar, used with scale factor and offset for drawing the chart tracks.
Background color	...	Background color selectable from palette.
Appearance	*Flat, Raised, Sunken*	- *Flat*: plain with use of *Border pts* and *Border col*; - *Raised*; - *Sunken*.
Border points	>= 0	Border thickness (#pixel). This property is sensible only if *Appearance* is set to *Flat*.
Border color	...	Border color selectable from palette. This property is sensible only if *Appearance* is set to *Flat* or *Text* is not empty.
Font	Name found in *Resources*	Font used for drawing the label in object.
Refresh	*TRUE, FALSE, var_name*	Continuous redraw of the object: - *FALSE*: the chart is updated only when opening page or when a child page is closed; - *TRUE*: the chart object is always updated, synchronized with others objects; - *var_name*: the chart is drawn on rising edge of boolean variable. This value *TRUE* of this variable is the moment when the char adds the samples. (Len Data <> internal HMI index of data).
Visible	*TRUE, FALSE, var_name*	Visible status of the object. It can be constant (*TRUE* or *FALSE*) or linked with a boolean variable *var_name*: if *var_name* is *TRUE* the object is visible, otherwise it is hidden.
Format X	String as *printf*	Format of X axis as c *printf* function.

Properties	Available values	Description
X Data	>=0, *var_name*	X-Axis array values. If there is a constant value in this property each Y sample has a X value equal to the product among X Data and the index in array. Ex. Y= track[3] = 20 X = 3* X Data
X Color	...	Color of X-Axis label.
Grid Step	Value	Space among two points of grid in pixel. The property is sensible if the grid is visible.
Sample Buffer	Value	Number of samples that the runtime can store. The older are deleted if the size has exceeded.
Grid Color	...	Color of grid if it is visible.
Bord. Color	...	Color of border grid.
Vertical Bar 1	>=0, *var_name*	Name of variable for drawing a vertical fixed bar on chart. It is allowed also a constant value. The * symbol means that there is not this vertical bar.
Color bar 1	...	Color of vertical bar 1 if different from *.
Vertical Bar 2		As *Vert. Bar 1*, but relative to bar 2.
Color bar 2		As *Color bar 1*, but relative to bar 2.
Vertical Bar 3		As *Vert. Bar 1*, but relative to bar 3.
Color bar 3		As *Color bar 1*, but relative to bar 3.
Clear Data	*var_name*	Boolean variable. If it is *TRUE* and *Refresh* is *TRUE* the chart deletes all the previous data.

5.14.2 EVENTS

Event	Description
BeforeUpdate	Before the object is redrawn.
AfterUpdate	Immediately after the object is redrawn.

5.15 TREND

5.15.1 PROPERTIES

Properties	Available values	Description
XPos	>= 0	Top-left 'x coordinate' edge relative to page.
YPos	>= 0	Top-left 'y coordinate' edge relative to page.
XDim	> 0	Width (#pixel).
YDim	> 0	Height (#pixel).
Name	Not empty	Name of object.

Properties	Available values	Description
Track 1	*var_name*	Open a dialog with the following options: - the variable will be sampled each *Sampling Time* seconds; - the visibility condition of the track (*TRUE* or boolean variable); - color of the track; - the scale factor (range among two horizontal divisions); - the offset (displacement of the track 0-Y); - step of print label for Y axis; - three horizontal bars with name, value and colors. If *var_name* is empty the track is not defined and not drawn.
Track 2	*var_name*	As *Track 1*, but for track 2.
Track 3	*var_name*	As *Track 1*, but for track 3.
Track 4	*var_name*	As *Track 1*, but for track 4.
Track 5	*var_name*	As *Track 1*, but for track 5.
Track 6	*var_name*	As *Track 1*, but for track 6.
Track 7	*var_name*	As *Track 1*, but for track 7.
Track 8	*var_name*	As *Track 1*, but for track 8.
Track Left	>=0, *var_name*	Integer value that is the track for Y axis left. It is admitted a constant value (ex. 1). If empty means that the chart must not draw the label for Y axis left.
Track Right	>=0, *var_name*	As *Track Left* for the right side.
Format Left	String as *printf*	Format of Y axis left as c *printf* function.
Format Right	String as *printf*	As *Format Left* for the right side.
XLabel	>=0, *var_name*	Step for X-axis labels. How many divisions of horizontal bar must have labels.
X Scale	*var_name*	Scale factor of x-Axis. Value range among two divisions of horizontal bars. An empty value indicates that the chart is in autoscale mode.
Sampling Time	>0	Sampling time measured in seconds. Every *Sampling Time* seconds the trend sample the value even if the chart is not shown.
X Offset	*var_name*	Variable name for the deviation of 0 for x-Axis, left or right. A positive value moves the chart values to left.
Grid	*Yes, No*	Visibility of the grid.
X Div. Grid	Value	Number of division on horizontal bar, used with scale factor and offset for drawing the chart tracks.
Y Div. Grid	Value	Number of division on vertical bar, used with scale factor and offset for drawing the chart tracks.
Background color	...	Background color selectable from palette.

Properties	Available values	Description
Appearance	*Flat, Raised, Sunken*	- *Flat*: plain with use of *Border pts* and *Border col*; - *Raised;* - *Sunken.*
Border points	>= 0	Border thickness (#pixel). This property is sensible only if *Appearance* is set to *Flat*.
Border color	...	Border color selectable from palette. This property is sensible only if *Appearance* is set to *Flat* or *Text* is not empty.
Font	Name found in *Resources*	Font used for drawing the label in object.
Refresh	*TRUE, FALSE, var_name*	Continuous redraw of the object: - *FALSE*: the chart is updated only when opening page or when a child page is closed; - *TRUE*: the chart object is always updated, synchronized with others objects; - *var_name*: the chart is drawn on rising edge of boolean variable. This value *TRUE* of this variable is the moment when the char adds the samples. (Len Data <> internal HMI index of data)
Visible	*TRUE, FALSE, var_name*	Visible status of the object. It can be constant (*TRUE* or *FALSE*) or linked with a boolean variable *var_name*: if *var_name* is *TRUE* the object is visible, otherwise it is hidden.
Format Time	*Default list*	Format for X-axis label. The choices are: - *ss*: seconds; - *mm.ss*: minutes, seconds; - *hh.mm*: hours, minutes; - *hh.mm.ss*: hours, minutes, seconds.
X Color	...	Color of X-Axis label.
Grid Step	Value	Space among two points of grid in pixel. The property is sensible if the grid is visible.
Sample buffer	Value	Number of samples that the runtime can store. The older are deleted if the size has exceeded.
Grid Color	...	Color of grid if it is visible.
Bord. Color	...	Color of border grid.
Vertical Bar 1	>=0, *var_name*	Name of variable for drawing a vertical fixed bar on chart. It is allowed also a constant value. The * symbol means that there is not this vertical bar.
Color bar 1	...	Color of vertical bar 1 if different from *.
Vertical Bar 2		As *Vert. Bar 1*, but relative to bar 2.
Color bar 2		As *Color bar 1*, but relative to bar 2.

Properties	Available values	Description
Vertical Bar 3		As *Vert. Bar 1*, but relative to bar 3.
Color bar 3		As *Color bar 1*, but relative to bar 3.
Clear Data	*var_name*	Boolean variable. If it is *TRUE* and *Refresh* is *TRUE* the chart deletes all the previous data.

5.15.2 EVENTS

Event	Description
BeforeUpdate	Before the object is redrawn.
AfterUpdate	Immediately after the object is redrawn.

6. APPENDIX II: FILE FOR TARGET DESCRIPTION

The *.def* files contain some definitions of target environment. UserInterface uses this information for generating custom code.

The *.def* file consists of two sections. It is allowed comment, that starts with a semicolon.(;).

This file is included in *pajx* file.

6.1 TARGET PROPERTIES

6.1.1 DESCRIPTION

This section consist of five records, which support one or more parameters. Each record is on new line and the elements must be separated with spaces or tabs.

Record Structure			
Header	**Param. 1**	**Param. 2**	**Description**
SCREEN	*dimX*	dimY	Screen dimension of target measured in pixel: - *DimX*: width; - *DimY*: height.
SAVESCREEN	0/1	---	Target board can save and restore video memory: - 0: no save; - 1: save and restore.
TOUCHSCREEN	0/1	---	Target board has touchscreen, i.e. can use the pressure events: - 0: no touchscreen; - 1: exists touchscreen.
REFRESH	msec	---	Refresh time of all objects in page, measured in milliseconds.
FONT_FORMAT	"HH"/"VH"	---	Font encoding.
ColorSET	"RGB"	---	
BMP_FORMAT	"SIMULAB"	---	Image encoding.
UNICODE	0/1	---	Target board has support for unicode fonts.
JOYPAD	0/1	---	Target board has a joypad that can be used for moving among elements of page and can be connected to actions.
INIT	0/1	---	If set to 1 says that HMI run-time has *Video_InitHMI()*, invoked on target start-up. Typically it is used for custom commands on start-up.
BMPFULL	0/1	---	If set to 1 generates PLC code extended for bitmap instead binary bitmap.

6.2 OBJECT VERSION

The graphical objects (editbox, textbox, static, bitmap, etc.) can have a version, or cannot exist. The syntax is:

```
CTRL "Name" "Version"
```

where:

- *Name*: name of graphical object. Ex. *Editbox*;
- *Version*: version of HMI run-time objects.

  If this value is set to -1, UserInterface does not make available this object.

6.3 SYSTEM ENUMERATIVES

Enumeratives of *.def* file are maps for binding among numeric values and strings, or other numeric values.

Each enumerative has an identifier, that specifies a function in the map with this syntax:

```
ENUM id   en_key   en_val
```

where:

- `id`: enumerative identifier;
- `en_key`: value-key of record, must be a number;
- `en_val`: value of value-key, can be a number or string.

6.3.1 DESCRIPTIONS

This paragraph describes the values for system enumeratives.

- Enumerative 100

  With this key you can define new buttons (the names will be shown in the *Key* field of actions table (see 4.8.5).

  The number of lines is not limited. But the user must define at least all the elements of 102 enumerative.

  - `id`: 100;
  - `en_key`: key encoding, one byte;
  - `en_val`: string with key name.

- Enumerative 101

  With this key you can define new actions (the names will be shown in *Action* field of actions table).

  - `id`: 101;
  - `en_key`: action identifier;
  - `en_val`: string with action name.

  This enumerative has a well defined number of lines. The following table shows you the corresponding actions.

en_key	Action
0	Calls local or global procedures.
1	Opens child page.
2	Closes current page.
3	Selects next object *Edit Box*, *Button*, etc..

en_key	Action
4	Selects previous object *Edit Box*, *Button*, etc..
9	Enters editing-mode (*Edit Box*, *Button*).
10	Leaving (not implemented).

The string `en_val` is arbitrary.

- Enumerative 102

  Selection and edit functions:

  - `id`: 102;
  - `en_key`: identifier of edit function;
  - `en_val`: string with the name of associated string.

The name of this field `en_val` must be the same of `en_val` of 100 enumerative, so that UserInterface associates an edit function with a key.

This enumerative has a well defined number of lines. See the actionsin the table below:

en_key	Edit function
0	Confirms modifications and leaves editing-mode.
1	Loses modification and leaves editing-mode.
2	Deletes selected character.
3	Moves cursor left.
4	Moves cursor right.
5	Selects the previous element of an enumerative associated with an Editbox.
6	Selects the next element of an enumerative associated with an Editbox.
7	Deletes the first character on the left.
8	Inserts tab character.
9	Switching to uppercase alphanumeric characters for a single character.
10	Transition to permanent uppercase alphanumeric characters.

- Enumerative 103

  Define a color palette, the encoding is RGB:

  - `id`: 103;
  - `en_key`: index of the color inside palette;
  - `en_val`: RGB color encoding.

  RGB encoding represents 24 bit of colors: `0x00bbggrr` where `bb` (1 byte) intensity of blue, `gg` (1 byte) the green and `rr` (1 byte) the red. The intensity is at least 0 and at most `0xff`.

  The number of lines is not limited. The user can define which colors he wants.

- Enumerative 104

  Names of object styles (shown on *Appearance* property):

  - *id:* 104;
  - *en_key*: style;
  - *en_val:* string with the name of style.

  This enumerative contains at most 3 records, supported by UserInterface.

en_key	Style
0	Flat, plane.
1	Raised.
2	Sunken.

6.3.2 EXAMPLE

```
;
;    Target properties
;
SCREEN     128    64
SAVESCREEN 1
REFRESH 50
FONT_FORMAT      "VH"
JOYPAD 1
INIT    1
BMPFULL 1
UNICODE 1
;
;    Versions of controls
;
CTRL"Static"           1
CTRL"EditBox"          1
CTRL"TextBox"          -1
CTRL"Button"           2
CTRL"Progress"         0
CTRL"Animation"        0
CTRL"Image"            0
CTRL"CustomCtrl"-1
CTRL"Chart"            -1
CTRL"Trend"            -1


;
;    Enumeratives
;
; ENUM 100: key codes
;
ENUM100    13     "Enter"
ENUM100    8      "Left"
ENUM100    12     "Right"
ENUM100    11     "Up"
ENUM100    10     "Down"
```

```
ENUM100    19     "LongEnter"
ENUM100    15     "LongLeft"
ENUM100    16     "LongRight"
ENUM100    17     "LongUp"
ENUM100    18     "LongDown"
ENUM100    30     "VK_F1"
ENUM100    31     "VK_F2"
ENUM100    32     "VK_F3"
ENUM100    33     "VK_F4"
ENUM100    34     "VK_F5"
ENUM100    35     "VK_F6"
ENUM100    36     "VK_F7"
ENUM100    37     "VK_F8"
ENUM100    38     "VK_F9"
ENUM100    39     "VK_F10"
;
; ENUM 101: key-related actions
;
ENUM101    0      "Call"
ENUM101    1      "OpenPage"
ENUM101    2      "Close"
ENUM101    3      "NextField"
ENUM101    4      "PrevField"
ENUM101    9      "Edit"
;
; ENUM 102: editing-mode keys
;
ENUM102    0      "Enter"
ENUM102    1      "LongLeft"
ENUM102    3      "Left"
ENUM102    4      "Right"
ENUM102    5      "Up"
ENUM102    6      "Down"
;
; ENUM 103: color codes
;                      BBGGRR
ENUM103    0      "0x00000000"  ; Bianco
ENUM103    1      "0x00FFFFFF"  ; Nero
;
; ENUM 104: controls appearance
;
ENUM104    0      "Flat"
ENUM104    1      "Raised"
ENUM104    2      "Sunken"
```

7. APPENDIX III: DESCRIPTION OF PARAMETER FILE

As described in section 2.8.4 it is possible to link in UserInterface some variables from external device.

In some objects you can define an explicit or implicit syntax in order to use the parameter mode.

To use the implicit syntax, `@Device.Parametro`, UserInterface requires a *.PARX* file in *xml* format.

For example:

```
<parameters>
<par ipa="10100" name="Par_TAB" descr="Tab (map code)" defval="0" min="0" max="65535" um="num" typetarg="unsignedShort">
 <protocol name="Modbus" commaddr="15716" commsubindex="0"/>
 <protocol name="CanOpen" commaddr="15716" commsubindex="0"/>
</par>
<par ipa="10001" name="Gain_Ntc_AI2" descr="NTC calibration gain AI2" defval="32768" min="0" max="65535" um="num" typetarg="unsignedShort">
 <protocol name="Modbus" commaddr="15617" commsubindex="0"/>
 <protocol name="CanOpen" commaddr="15617" commsubindex="0"/>
</par>
<par ipa="11308" readonly="false" name="Modem_InitStr1" defval="" descr="Init String (1st part)" typetarg="string" strsize="19">
 <protocol name="Modbus" commaddr="15821" commsubindex="0"/>
 <protocol name="CanOpen" commaddr="15821" commsubindex="0"/>
</par>
</parameters>
```

Where each parameter has these fields.

- *ipa:* parameter index used as input value of *Video_SetParam()*, *Video_GetParam()*. If there are nodes with protocol type, they have more priority than *ipa*, so UserInterface uses them.
- *Name*: parameter name.
- *descr*: complete description of parameter.
- *defval*: default value of parameter.
- *min*: minimum value of parameter.
- *max*: maximum value of parameter.
- *um*: measure unit of parameter.
- *typetarg*: type of parameter.

  The available values with the translation in PLC are:

  - *char*: SINT;
  - *unsignedChar*: USINT;
  - *short*: INT;
  - *unsignedShort*: UINT;
  - *int*: DINT;
  - *unsignedInt*: UDINT;
  - *boolean*: BOOL;

  - *digitalInput*: BOOL;
  - *digitalOutput*: BOOL;
  - *float*: REAL;
  - *double*: REAL;
  - *string*: STRING.
- *strsize*: number of character if it is a string type.

8. APPENDIX IV: ELEMENTS OF HMI RUNTIME

8.1 FUNCTIONS

This chapter lists all the functions that HMI run-time exports to UserInterface and so the user can use them into script and procedures.

These functions are divided into several categories which are shown in details in the following paragraphs.

8.1.1 SYSTEM FUNCTIONS: HARDWARE AND OPERATING SYSTEM

```
unsigned char Video_InitHMI (unsigned char dmy)
```

Function of initialization for HMI runtime

Parameter	Description
dmy	Reserved. Set 0.
Return Value	**Description**
Video_InitHMI	*TRUE* if successful, *FALSE* otherwise.

```
unsigned char Video_Switch (unsigned char on);
```

Turn on/off the display

Parameter	Description
on	*TRUE*: turns on the display. *FALSE*: turns of the display.
Return Value	**Description**
Video_Switch	Not sensible (always *TRUE*).

```
unsigned char Video_LCDContrast( unsigned char more );
```

Display contrast

Parameter	Description
more	*TRUE*: increases display contrast. *FALSE*: decreases display contrast.
Return Value	**Description**
Video_LCDContrast	Not sensible (always *TRUE*).

```
unsigned char Video_SaveRect( unsigned short x1, unsigned short y1, unsigned short x2, unsigned short y2 );
```

Save display area to memory

Parameter	Description
x1	Top-left 'x coordinate' edge relative to full page.
y1	Top-left 'y coordinate' edge relative to full page.
x2	Bottom-down 'x coordinate' edge relative to full page.
y2	Bottom-down 'y coordinate' edge relative to full page.

Return Value	Description
Video_SaveRect	Not sensible (always *TRUE*).

```
unsigned char Video_WriteFromBuff( unsigned short x1,unsigned short y1, unsigned short x2,unsigned short y2 );
```

Restore display area from memory (previously saved with Video_SaveRect).

Parameter	Description
x1	Not sensible (saved area has the original coordinates).
y1	Not sensible (saved area has the original coordinates).
x2	Not sensible (saved area has the original coordinates).
y2	Not sensible (saved area has the original coordinates).
Return Value	**Description**
Video_WriteFromBuff	Not sensible (always *TRUE*).

```
unsigned char Video_Lock( unsigned char res );
```

Lock the display resources for exclusive access

Parameter	Description
res	Reserved. Set 0.
Return Value	**Description**
Video_Lock	Not sensible (return input parameter res).

```
unsigned char Video_Unlock( unsigned char res );
```

Unlock the display resource after exclusive access

Parameter	Description
res	Reserved. Set 0.
Return Value	**Description**
Video_Unlock	Not sensible (return input parameter res).

```
unsigned char Video_Sleep( unsigned short msec );
```

Suspend the task where the function is used

Parameter	Description
msec	Suspends time measured in milliseconds.
Return Value	**Description**
Video_Unlock	Not sensible (always *TRUE*).

8.1.2 FUNCTION FOR MANAGING PROJECT RESOURCES AND COMMON PROPERTIES

```
unsigned char Video_SetWndSysProps( unsigned long pFont, unsigned long colFore, unsigned long colBack );
```

Set common properties for all pages in the project

Parameter	Description
pFont	Address of font for printing text in title bar (the font must be added with Video_AddFont function).
colFore	Text color of *Title Bar*.
colBack	Background color of *Title Bar*.
Return Value	**Description**
Video_SetWndSysProps	Not sensible (always *TRUE*).

```
unsigned char Video_SetEditKey( unsigned char id, unsigned char code );
```

Set key-code for editing functions

Parameter	Description
id	Identifier of editing function (see. Enumerative table 102, 6.3.1).
code	Key code associated with editing function.
Return Value	**Description**
Video_SetEditKey	Not sensible (always *TRUE*).

```
unsigned char Video_AddFont( unsigned long pFont, unsigned char charLen,
unsigned char charHei, unsigned char offs );
```

Publish a new font in HMI run-time

Parameter	Description
pFont	Address of first byte of font.
charLen	Character width of font (#pixel).
charHei	Character height of font (#pixel).
offs	Byte offset of a font that starts with ASCII 0x00 (subset of characters).
Return Value	**Description**
Video_AddFont	*TRUE* if successful, *FALSE* otherwise.

```
unsigned char Video_AddFontUnicode( unsigned long pFont, unsigned char char-
Len, unsigned char charHei );
```

Publish a new unicode font in HMI run-time

Parameter	Description
pFont	Address of first byte of font.
charLen	Character width of font (#pixel).
charHei	Character height of font (#pixel).
Return Value	**Description**
Video_AddFontUnicode	*TRUE* if successful, *FALSE* otherwise.

```
unsigned char Video_LoadLanguage( unsigned long pResStrings, unsigned long
pEnums );
```

Load strings and enumeratives of any language

Parameter	Description
pResStrings	Address of first resources string for current language.
pEnums	Address of first resources string for current language.
Return Value	**Description**
Video_LoadLanguage	*TRUE* if successful, *FALSE* otherwise.

```
unsigned char Video_DrawFrames(unsigned short left, unsigned short top, unsigned short right, unsigned short bottom,
  unsigned long colBack, unsigned char fBar,
  unsigned long pTitle, unsigned char fResStr,
  unsigned char fSysBtn, unsigned char style );
```

Function for draw frame-set

Parameter	Description
left	Width of left frame (#pixel).
top	Height of top frame (#pixel).
right	Width of right frame (#pixel).
bottom	Height of bottom frame (#pixel).
colBack	Background color.
fBar	- *TRUE*: shows title bar; - *FALSE*: hides title-bar.
pTitle	Text of title bar: NULL: No string in title.
fResStr	- *TRUE*: pTitle is a resource string; - *FALSE*: pTitle is an address of constant string.
fSysBtn	- *TRUE*: shows system; - *FALSE*: hides system button.
style	- 0: Flat; - 1: Raised; - 2: Sunken.
Return Value	**Description**
Video_DrawFrames	Not sensible (always *TRUE*).

8.1.3 FUNCTIONS FOR OPERATING WITH PAGES

```
unsigned char Video_InitPage( unsigned short x1, unsigned short y1, unsigned short x2, unsigned short y2,
  unsigned long pTitle, unsigned short wData );
```

Show a page on display

Parameter	Description
x1	Top-left 'x coordinate' edge relative to full page.
y1	Top-left 'y coordinate' edge relative to full page.
x2	Bottom-down 'x coordinate' edge relative to full page.

Parameter	Description
y2	Bottom-down 'y coordinate' edge relative to full page.
pTitle	Address of *Text* of title bar: - *NULL*: no text in title bar.
wData	Feature declaration: b0..b7: - 0: Flat; - 1: Raised; - 2: Sunken. b8: - 0: no title bar; - 1: shows title bar. b9: - 0: pTitle is an address of constant string; - 1= pTitle is a resource string. b10: - 0: no system button; - 1: shows system button. b11: - 0: window not modal; - 1: modal window (sensible only for pop-ups windows).
Return Value	**Description**
Video_InitPage	Not sensible (always *TRUE*).

```
unsigned char Video_SetPageColors( unsigned long colFore, unsigned long colBack );
```

Assign all colors for current page

Parameter	Description
colFore	Color of the text of page.
colBack	Background color of page.
Return Value	**Description**
Video_SetPageColors	Not sensible (always *TRUE*).

```
unsigned char Video_ClrScreen( );
```

Delete entire display area and fill with background color defined with Video_SetPageColors

Return Value	Description
Video_ClrScreen	*TRUE* if successful, *FALSE* otherwise.

```
unsigned char Video_ClrRect( unsigned short x1, unsigned short y1, unsigned short x2, unsigned short y2 );
```

Delete only a portion of display and fill with background color defined with Video_SetPageColors

Parameter	Description
x1	Top-left 'x coordinate' edge relative to full page.
y1	Top-left 'y coordinate' edge relative to full page.
x2	Bottom-down 'x coordinate' edge relative to full page.
y2	Bottom-down 'y coordinate' edge relative to full page.
Return Value	**Description**
Video_ClrRect	*TRUE* if successful, *FALSE* otherwise.

```
unsigned char Video_SetFont( unsigned long fontPtr );
```

Load a font as current font for drawing objects. To correctly execute this function, the font must be declared with `Video_AddFont`.

Parameter	Description
fontPtr	Address of first byte of font.
Return Value	**Description**
Video_SetFont	*TRUE* if successful, *FALSE* otherwise.

```
unsigned char Video_SetColors( unsigned long colForeTxt, unsigned long colBackTxt, unsigned long colForeSel, unsigned long colBackSel );
```

Assign the current colors for drawing objects

Parameter	Description
colForeTxt	Text color.
colBackTxt	Background color.
colForeSel	Text color for selection.
colBackSel	Background color for selection.
Return Value	**Description**
Video_SetColors	*TRUE* if successful, *FALSE* otherwise.

```
unsigned char Video_ResetMaps( unsigned char res );
```

Delete the maps saved for every object. The maps are created adding an object at once, with access mode `kACS_INIT`.

Parameter	Description
res	Reserved. Set 0.
Return Value	**Description**
Video_ResetMaps	Not sensible (return input parameter `res`).

8.1.4 FUNCTION FOR OBJECTS

`unsigned char Video_NextEdit( unsigned char fRWOnly );`

Enable selection for next objects identified by *Sel. Order* attribute.

Parameter	Description
fRWOnly	Limit for selecting the next edit-box: - *FALSE*: next edit-box must be selectable; - *TRUE*: the next edit-box must be selectable and writable.
Return Value	**Description**
Video_NextEdit	Handle of selected objects; if -1 the function has an error.

`unsigned char Video_PrevEdit( unsigned char fRWOnly );`

Enable selection for previous objects identified by *Sel. Order* attribute.

Parameter	Description
fRWOnly	Limit for selecting the next edit-box: - *FALSE*: the next edit-box must be selectable; - *TRUE*: the next edit-box must be selectable and writable.
Return Value	**Description**
Video_PrevEdit	Handle of selected objects; if -1 the function has an error.

`unsigned char Video_EnterEdit( unsigned short wHnd );`

Enter edit-mode of an Edit Box otherwise execute the action for a button. The object holds the task until exits from edit-mode.

Parameter	Description
wHnd	Handle of object that must be edited or execute his action.
Return Value	**Description**
Video_EnterEdit	Return pressed key code for exiting edit-mode. If return -1 is an error only if the object is an edit-box.

`unsigned char Video_EnterEditSel( unsigned short wHnd, unsigned char onlySelect )`

Select object or enter edit-mode of an Edit box otherwise execute the action for a button. The object holds the task until exit from edit-mode.

Parameter	Description
wHnd	Handle of object that must be edited or execute his action.
OnlySelect	- *FALSE*: as VideoEnterEdit(); - *TRUE*: enables only the selection without entering edit-mode.

Return Value	Description
Video_EnterEditSel	Return pressed key code for exiting edit-mode. If return -1 is an error only if the object is an edit-box.

```
unsigned char Video_PushButton( unsigned short wHnd );
```

Enter press-mode for buttons. The object holds the task until exit from press-mode. This function is sensible only for touchscreen systems.

Parameter	Description
wHnd	Handle of button.
Return Value	**Description**
Video_PushButton	- *TRUE*: last pressure event was in button area; - *FALSE*: last pressure event was outside button area; - *-1*: error.

```
short Video_FirstLastEdit( unsigned char rwReq, unsigned char last )
```

Return the handle of first or last selectable controls.

Parameter	Description
rwReq	Boolean parameter. It indicates if the function checks for the objects that have read-write access mode.
last	- *TRUE:* last selectable object; - *FALSE*: first selectable object.
Return Value	**Description**
Video_FirstLastEdit	Handle of the object; -1 if errors or do not exist selectable objects

8.1.5 DRAWING FUNCTIONS

```
unsigned char Video_Line( unsigned short x1, unsigned short y1, unsigned
short x2, unsigned short y2, unsigned char pts, unsigned long color );
```

Draw a line

Parameter	Description
x1	Top-left 'x coordinate' edge relative to full page.
y1	Top-left 'y coordinate' edge relative to full page.
x2	Bottom-down 'x coordinate' edge relative to full page.
y2	Bottom-down 'y coordinate' edge relative to full page.
pts	Thickness.
color	Line color.
Return Value	**Description**
Video_Line	*TRUE* if successful, *FALSE* otherwise.

```
unsigned char Video_Rectangle( unsigned short x1, unsigned short y1, un-
signed short x2, unsigned short y2, unsigned char pts, unsigned char transp,
unsigned long bordCol, unsigned long fillCol );
```

Draw a rectangle

Parameter	Description
x1	Top-left 'x coordinate' edge relative to full page.
y1	Top-left 'y coordinate' edge relative to full page.
x2	Bottom-down 'x coordinate' edge relative to full page.
y2	Bottom-down 'y coordinate' edge relative to full page.
pts	Border thickness.
transp	- *TRUE*: transparent square; - *FALSE*: solid square.
bordCol	Border color.
fillCol	Fill color. The value is not sensible if *transp* is *TRUE*.
Return Value	**Description**
Video_Rectangle	*TRUE* if successful, *FALSE* otherwise.

```
unsigned char Video_DrawBorder( unsigned char style, unsigned short x1, unsigned short y1, unsigned short x2, unsigned short y2, unsigned char pts, unsigned char color );
```

Draw a border outside the rectangle area

Parameter	Description
style	- 0: flat; - 1: raised; - 2: sunken.
x1	Top-left 'x coordinate' edge relative to full page.
y1	Top-left 'y coordinate' edge relative to full page.
x2	Bottom-down 'x coordinate' edge relative to full page.
y2	Bottom-down 'y coordinate' edge relative to full page.
pts	Border thickness. It is sensible only if *style* = 0.
color	Border color. It is sensible only if *style* = 0.
Return Value	**Description**
Video_DrawBorder	*TRUE* if successful, *FALSE* otherwise.

```
unsigned char Video_DelBorder( unsigned char style, unsigned short x1, unsigned short y1, unsigned short x2, unsigned short y2, unsigned char pts );
```

Delete a border outside the rectangle area. The color of fill is the page color assigned with `Video_SetPageColors`

Parameter	Description
style	- 0: flat; - 1: raised; - 2: sunken.
x1	Top-left 'x coordinate' edge relative to full page.
y1	Top-left 'y coordinate' edge relative to full page.
x2	Bottom-down 'x coordinate' edge relative to full page.

Parameter	Description
y2	Bottom-down 'y coordinate' edge relative to full page.
pts	Border thickness It's sensible only if *style* = 0
Return Value	**Description**
Video_DelBorder	*TRUE* if successful, *FALSE* otherwise.

```
unsigned char Video_PrintBitmap( unsigned long ptrBmp, unsigned short x,
unsigned short y );
```

Print a bitmap coded with run-time HMI format

Parameter	Description
ptrBmp	Address of first byte of bitmap.
x	Top-left 'x coordinate' edge relative to full page.
y	Top-left 'y coordinate' edge relative to full page.
Return Value	**Description**
Video_PrintBitmap	Not sensible (always *TRUE*).

```
unsigned char Video_DelBitmap( unsigned long ptrBmp, unsigned short x, un-
signed short y );
```

Delete a bitmap where it is not transparent, coded with run-time HMI format

Parameter	Description
ptrBmp	Address of first byte of bitmap.
x	Top-left 'x coordinate' edge relative to full page.
y	Top-left 'y coordinate' edge relative to full page.
Return Value	**Description**
Video_DelBitmap	Not sensible (always *TRUE*).

```
unsigned long Video_InitBmpTreeRefresh( unsigned short x1,
unsigned short y1, unsigned short x2, unsigned short y2 );
```

Switch context of drawing area. With this call all the next drawing functions uses the invisible device context

Parameter	Description
x1	Top-left 'x coordinate' edge relative to full page.
y1	Top-left 'y coordinate' edge relative to full page.
x2	Bottom-down 'x coordinate' edge relative to full page.
y2	Bottom-down 'y coordinate' edge relative to full page.
Return Value	**Description**
Video_InitBmpTreeRefresh	Address of invisible device context.

```
unsigned long Video_EndBmpTreeRefresh( unsigned short pDC,
unsigned short x1, unsigned short y1,
unsigned short x2, unsigned short y2 );
```

Restore original device context and copy the area from invisible context to display context

Parameter	Description
pDC	Address of invisible device context.
x1	Top-left 'x coordinate' edge relative to full page.
y1	Top-left 'y coordinate' edge relative to full page.
x2	Bottom-down 'x coordinate' edge relative to full page.
y2	Bottom-down 'y coordinate' edge relative to full page.
Return Value	**Description**
Video_EndBmpTreeRefresh	Not sensible (always *TRUE*).

8.1.6 FUNCTIONS FOR TEXT

```
unsigned char Video_PrintStr( char * str, unsigned short x, unsigned short y );
```

Print a string using the current font set with *SetFont* and current colors set with *SetColors()*

Parameter	Description
str	Text to print.
x	Top-left 'x coordinate' edge relative to full page.
y	Top-left 'y coordinate' edge relative to full page.
Return Value	**Description**
Video_PrintStr	Number of chars printed.

```
unsigned char Video_PrintResStr( unsigned short idRes, unsigned short x, unsigned short y );
```

Print a resources string using the current font set with *SetFont* and current colors set with *SetColors()*

Parameter	Description
idRes	Identifiers of resource.
x	Top-left 'x coordinate' edge relative to full page.
y	Top-left 'y coordinate' edge relative to full page.
Return Value	**Description**
Video_PrintResStr	Number of chars printed.

```
unsigned char Video_PrintNChar( char * str, unsigned char accMode, unsigned short x, unsigned short y, unsigned char nChar, unsigned long format );
```

Print at most *nChar* characters of a string, using the current font set with *SetFont* and current colors set with *SetColors()*. It uses also a format for drawing the text.

If *nChar* is less than string length, it truncates the string; otherwise apply the alignment.

Parameter	Description
str	Text to print.
accMode	- kACS_PRINT: print with colForeTxt and colBackTxt colors. - kACS_SELECT: print with colForeSel and colBackSel colors.
x	Top-left 'x coordinate' edge relative to full page.
y	Top-left 'y coordinate' edge relative to full page.
nChar	Maximum number of chars to print.
format	Alignment of text. It is sensible only if *nChar > length of str*: - 0x08 = right alignment; - 0x10 = center alignment; - 0x20 = left alignment.
Return Value	**Description**
Video_PrintNChar	Number of chars of truncated string.

8.1.7 FUNCTIONS FOR PARAMETER ACCESS

```
unsigned short Video_GetParam( unsigned char idxDevice, unsigned short idx-
Param, unsigned char subIdxParam, unsigned long pVal, unsigned char type )
```

Read a parameter from a device

Parameter	Description
idxDevice	Index of device connected.
idxParam	Index of parameter.
subIdxParam	Sub-index of parameter.
pVal	Address of variable that contains the read value.
type	Parameter type. Available values: tyBool, tySInt, tyUSInt, tyByte, tyInt, tyUInt, tyWord, tyDInt, tyUDInt, tyDWord, tyReal,tyString.
Return Value	**Description**
Video_GetParam	Integer values: - 0 = successful; - 1 = index of parameter not found; - 2,8,9 = system errors; - 3 = type not valid.

```
unsigned short Video_SetParam( unsigned char idxDevice, unsigned short idx-
Param, unsigned char subIdxParam, unsigned long pVal, unsigned char type )
```

Write a parameter to a device.

Parameter	Description
idxDevice	Index of device connected.
idxParam	Index of parameter.
subIdxParam	Sub-index of parameter.
pVal	Address of variable that contains the value to write.
type	Parameter type. Available values: tyBool, tySInt, tyUSInt, tyByte, tyInt, tyUInt, tyWord, tyDInt, tyUDInt, tyDWord, tyReal, tyString.
Return Value	**Description**
Video_SetParam	Integer values: - 0 = successful; - 1 = index of parameter not found; - 2,8,9 = system errors; - 3 = type not valid; - 4 = read-only parameter; - 5 = cannot write now; - 6 = the value is less than the min value; - 7 = the value is more than the max value.

8.1.8 FUNCTIONS FOR EVENTS

```
unsigned char Video_SendEvent( unsigned short msgID, unsigned short wParam
);
```

Send an event from code

Parameter	Description
msgID	Available values: - kWM_NULL = no event; - kWM_KEY = key pressure; - kWM_MSG = open message; - kWM_SELECT = select an edit-box, a button; - kWM_PUSH = pressure on button.
wParam	Event parameter. It has a different meaning according to msgID: - if kWM_NULL= not sensible; - if kWM_KEY= pressed key. For the key a constant value exists. The syntax is: kKEY_<key> Ex. LongLeft -> kKEY_LongLeft - if kWM_MSG =ID of message page to open; - if kWM_SELECT= handle of selected edit-box, button; - if kWM_PUSH= handle of pressed button.

Return Value	Description
Video_SendEvent	*TRUE* if successful, *FALSE* otherwise.

`unsigned long Video_GetEvent( unsigned char dmy );`

Pop an event from queue

Parameter	Description
dmy	Reserved. Set 0.
Return Value	**Description**
Video_GetEvent	Double word with inside the encoding. 16 low bit = type of event: - kWM_NULL = no event; - kWM_KEY = key pressure; - kWM_MSG = open message; - kWM_SELECT = select an edit-box, a button; - kWM_PUSH = pressure on button. 16 high bit = event parameter: - if kWM_NULL= not sensible; - if kWM_KEY= pressed key; - if kWM_MSG= ID of message page to open; - if kWM_SELECT= handle of selected edit-box, button; - if kWM_PUSH= handle of pressed button.

8.2 FUNCTION BLOCKS

`FUNCTION BLOCK: Video_GetPageColors`

Get the page colors of the page where called

Frame structure: FB_VIDEO_GETPAGEColorS		
Local variables	**Type**	**Description**

Input variables	**Type**	**Description**

Output variables	**Type**	**Description**
color	word32	Text color in the page.
back	word32	Background of the page.

`FUNCTION BLOCK: Static01`

Text strings with variable visibility

Frame structure: FB_STATIC01		
Local variables	**Type**	**Description**
memVis	byte	Visibility status of the previous execution.
Input variables	**Type**	**Description**
wHnd	word16	Handle of the object. Must be unique among static objects.
x	word16	Top-left 'x coordinate' edge relative to full page.
y	word16	Top-left 'y coordinate' edge relative to full page.
accMode	byte	- kACS_IDLE = no effect; - kACS_INIT = first draw on display; - kACS_PRINT = update draw on display.
fResStr	byte	Boolean value: - *FALSE* = pString is the address of string to draw; - TRUE = pString is the identifier of resource string.
pString	word32	Text to draw. It is different according to fResStr.
pFont	word32	Address of font for drawing text. The font must be initialized with Video_AddFont.
foreCol	word32	Text color.
bckCol	word32	Background color.
pVisVar	word32	Visibility. Available values: - *FALSE* = text not visible; - *TRUE* = text always visible; - var_addr = address of boolean variable.
format	word16	Format for numeric values, encoded in 32 bit: b3 1= right alignment b4 1= center alignment b5 1= left alignment
style	byte	- 0 = flat - 1 = raised - 2 = sunken
bordPts	byte	Border thickness. It is sensible only if *style* = 0
bordCol	word32	Border color. It is sensible when *style* = 0 bordPts > 0 and not pSelVar = 1 fixed.
selBackCol	word32	Background color when object is selected.It is not sensible if pSelVar = 0 fixed.

Frame structure: FB_STATIC01		
selForeCol	word32	Text color when selected. It is not sensible if pSelVar = 0 fixed.
pRefrVar	word32	Variable for update: - *FALSE* = the object is redrawn only when the page is opening or when returning from child page; - *TRUE* = the object is always redrawn.
pSelVar	word32	Selection flag for the object. Suggest if the object must uses {'selBackCol' } and {'selForeCol'}. Available values: - *FALSE* = object is never selected; - *TRUE* = object is always selected; - *var_addr* = address of boolean variable.
numChars	word16	Number of max characters. 0 indicates that the string is drawn with the entire value of pString.
Output variables	**Type**	**Description**

FUNCTION BLOCK: Image

Image object

Frame structure: FB_IMAGE		
Local variables	**Type**	**Description**
memVis	byte	Visibility status of the previous execution.
memSel	byte	Selection status of the previous execution.
Input variables	**Type**	**Description**
wHnd	word16	Handle of the object. Must be unique among image objects.
x1	word16	Top-left 'x coordinate' edge relative to full page.
y1	word16	Top-left 'y coordinate' edge relative to full page.
px1	word32	Address of variable for moving image on X-Axis. It is sensible only if *floating* = *TRUE*
py1	word32	Address of variable for moving image on Y-Axis. It is sensible only if *floating* = *TRUE*

Frame structure: FB_IMAGE		
type_x	byte	Type for px1. Available values: tySInt; tyUSInt; tyByte; tyInt; tyUInt; tyWord; tyDInt; tyUDInt; tyDWord. It is sensible only if *floating* = *TRUE* and px1 <> *NULL*
type_y	byte	Type for *py1*. Available values: tySInt; tyUSInt; tyByte; tyInt; tyUInt; tyWord; tyDInt; tyUDInt; tyDWord. It is sensible only if *floating* = *TRUE* and *py1* <> *NULL*
dx	word16	Width (#pixel).
dy	word16	Height (#pixel).
style	byte	- 0 = flat - 1 = raised - 2 = sunken
floating	byte	Position of object: - *FALSE* = docking - *TRUE* = floating
bordPts	byte	Border thickness. It is sensible only if style = 0
bordCol	word32	Border color. It is sensible when *style* = 0 bordPts > *0* and not pSelVar = 1 fixed.
bordSelCol	word32	Border color for selected object. It is sensible when style = *0* and bordPts > 0 and not pSelVar = 0 fixed
accMode	byte	- kACS_IDLE = no effect - kACS_INIT = first draw on display - kACS_PRINT = update draw on display - kACS_QUERY = request for updating output variables - kACS_BCKQUERY = request for updating output variables when the object is in background pages - kACS_DELETE = delete object
pBmp	word32	Address of first byte of bitmap to view. It is not sensible if pSelBmp = 1 fixed.
pSelBmp	word32	Address of first byte of bitmap to view when selected. It is not sensible if pSelBmp = 0 fixed.

Frame structure: FB_IMAGE		
pSelVar	word32	Selection flag for the object. Suggest if the object must uses {'bordCol', 'pBmp'} or {'bordSelCol', 'pSelBmp'}. Available values: - *FALSE* = object is never selected - *TRUE* = object is always selected - var_addr = address of boolean variable
pVisVar	word32	Flag of visibility. Available values: - *FALSE* = image not visible - *TRUE* = image always visible - var_addr = address of boolean variable
Output variable	**Type**	**Description**
reqRefr	byte	Request refresh, updated when the object is called with accMode = kACS_QUERY or accMode = kACS_BCKQUERY.
abs_x1	word16	Top-left 'x coordinate' edge relative to full page obtained with the sum among 'x1' and 'px1'. The value is updated when the object is called with accMode = kACS_INIT or accMode = kACS_QUERY.
abs_y1	word16	Top-left 'y coordinate' edge relative to full page obtained with the sum among 'y1' and 'py1'. The value is updated when the object is called with accMode = kACS_INIT or accMode = kACS_QUERY.
mem_x1	word16	Value read from abs_x1 when the object is called with accMode = kACS_INIT or accMode = kACS_PRINT.
mem_y1	word16	Value read from abs_y1 when the object is called with accMode = kACS_INIT or accMode = kACS_PRINT.

FUNCTION BLOCK: Animation

Animation object

Frame structure: FB_ANIMATION		
Local variables	**Type**	**Description**
memBmp	word32	Address of bitmap of the previous execution
Input variables	**Type**	**Description**
wHnd	word16	Handle of the object. Must be unique among animation objects.
x1	word16	Top-left 'x coordinate' edge relative to full page.
y1	word16	Top-left 'y coordinate' edge relative to full page.
x2	word16	Bottom-right 'x coordinate' edge relative to full page.

Frame structure: FB_ANIMATION		
y2	word16	Bottom-right 'y coordinate' edge relative to full page
style	byte	- 0 = flat - 1 = raised - 2 = sunken
bordPts	byte	Border thickness. It is sensible only if style = 0
bordCol	word32	Border color. It is sensible when style = 0 bordPts > 0 and not pSelVar = 1 fixed
accMode	byte	- kACS_IDLE = no effect - kACS_INIT = first draw on display - kACS_PRINT = update draw on display
pBmpArr	word32	Address of first image to view.
pCaseArr	word32	Address of first element of selection.
nArrEl	byte	Number of elements in image list.
pBmpDef	word32	Address of bitmap to view pSelVar not in pCaseArr.
pSelVar	word32	Address of variable for selection.
type	byte	Type of pSelVar. Available values: tyBool; tySInt; tyUSInt; tyByte; tyInt; tyUInt; tyWord; tyDInt; tyUDInt; tyDWord.
pVisVar	word32	Flag of visibility. Available values: - *FALSE* = image not visible - *TRUE* = image always visible - var_addr = address of boolean variable
Output variable	**Type**	**Description**

FUNCTION BLOCK: Button02

Button object

Frame structure: FB_BUTTON02		
Local variables	**Type**	**Description**
memVis	byte	Visibility status of the previous execution.
memTransp	byte	Transparency status of the previous execution.
memSel	byte	Selection status of the previous execution.
Input variables	**Type**	**Description**
wHnd	word16	Handle of the object. Must be unique among buttons objects.
x1	word16	Top-left 'x coordinate' edge relative to full page.
y1	word16	Top-left 'y coordinate' edge relative to full page.

Frame structure: FB_BUTTON02		
x2	word16	Bottom-right 'x coordinate' edge relative to full page.
y2	word16	Bottom-right 'y coordinate' edge relative to full page.
fResStr	byte	Boolean value: - *FALSE* = pString is the address of string to draw - *TRUE* = pString is the identifier of resource string
pText	word32	Text to draw on the button. It has different meaning according to fResStr. If this field is *NULL*, no text is drawn.
pFont	word32	Address of font for drawing text. The font must be initialized with Video_AddFont.
style	byte	- 0 = flat - 1 = raised - 2 = sunken
bordPts	byte	Border thickness. It is sensible only if style = 0
bordCol	word32	Border color and text color. It is sensible only if style = 0 and bordPts > 0, or pString different as *NULL*, and not pSelVar = 1 fixed.
fillCol	word32	Color of button area. It is sensible only if pTransp different as 1 fixed, and not pSelVar = 1 fixed.
bordSelCol	word32	Border color and text color when selected. It is sensible only if style = 0 and *bordPts* > 0, or pString different as NULL, and not pSelVar = 0 fixed.
fillSelCol	word32	Color of button area when selected. It is sensible only if pTransp different as 1 fixed, and not pSelVar = 0 fixed.
accMode	byte	- kACS_IDLE = no effect - kACS_INIT = first draw on display - kACS_PRINT = update draw on display
pVisVar	word32	Flag of visibility. Available values: - *FALSE* = image not visible - *TRUE* = image always visible - var_addr = address of boolean variable
pTransp	word32	Flag of transparency. Available values: - *FALSE* = button always solid - *TRUE* = button always transparent - var_addr = address of boolean variable

Frame structure: FB_BUTTON02		
pPressVar	word32	Address of a boolean variable. Pressed button= *pPressVar = *TRUE* Released button= *pPressVar = *FALSE*. If the field is *NULL* there is no variable.
pSelVar	word32	Selection flag for the object. Suggest if the object must uses {'bordCol', 'fillCol'} or {'bordSelCol', 'fillSelCol'}. Available values: - *FALSE* = object is never selected - *TRUE* = object is always selected - var_addr = address of boolean variable
format	word16	Format of numeric values encoded with 16 bit: b4 1= right alignment b5 1= center alignment b6 1=left alignment
order	word16	Number for establishing a sequential selection
Output variable	**Type**	**Description**

FUNCTION BLOCK: EditBox01

Edit object

Frame structure: FB_EDITBOX01		
Local variables	**Type**	**Description**
memVis	byte	Visibility status of the previous execution.
Input variables	**Type**	**Description**
wHnd	word16	Handle of the object. Must be unique among edit-box objects.
x1	word16	Top-left 'x coordinate' edge relative to full page.
y1	word16	Top-left 'y coordinate' edge relative to full page.
x2	word16	Bottom-right 'x coordinate' edge relative to full page.
y2	word16	Bottom-right 'y coordinate' edge relative to full page.
pFont	word32	Address of font for drawing text. The font must be initialized with Video_AddFont.
style	byte	- 0 = flat - 1 = raised - 2 = sunken
foreCol	word32	Text color.

Frame structure: FB_EDITBOX01		
bckCol	word32	Background color.
foreSelCol	word32	Text color when selected. It is sensible only if pCanSel is not 0 fixed.
bckSelCol	word32	Background color when selected. It is sensible only if pCanSel is not 0 constant.
bordPts	byte	Border thickness. It is sensible only if style = 0
bordCol	word32	Border color. It is sensible when style = 0 bordPts> 0
rw	byte	- *FALSE* = read-only mode - *TRUE* = read-write mode
refr	byte	Request refresh: - *FALSE* = the object is redrawn only when the page is opening or return from child page - *TRUE* = the object is always redrawn
pVar	word32	Address of variable or parameter according to format. It cannot be *NULL*. If it is a parameter is encoded in this way: b0..b7 = Subindex parameter b8..b23 = IPA parameter b24..b32 = Device address
type	byte	Type of data. Available values: tyBool; tySInt; tyUSInt; tyByte; tyInt; tyUInt; tyWord; tyDInt; tyUDInt; tyDWord,tyReal
pVarMin	word32	Min value for edit-box variable. If bit b16-b17 (LSB) of field *format* contains 0 the limit is not set, if contains 1 is a constant limit, if contains 2 it is a variable limit.
pVarMax	word32	Max value for edit-box variable. If bit b14-b15 (LSB) of field *format* contains 0 the limit is not set, if contains 1 is a constant limit, if contains 2 it's a variable limit.
enumId	int16	Identifier of enumerative. If 0 no enumerative associated with this field exists.

Frame structure: FB_EDITBOX01		
format	word32	View format encoded in 32 bit: b0 - 0 = draw sign only if number is negative - 1 = draw sign also for positive numbers b1 - 0 = does not print most significant null digits - 1 = draw zeroes on most significant null digits b2 - 0 = 'pVar' is a variable - 1 = 'pVar' is a parameter b3 1 = right alignment b4 1 = center alignment b5 1 = left alignment b10 Exadecimal format, with a..f lowercase b11 Exadecimal format, with A..F uppercase b14..b15 - 0=no max limit - 1=constant max limit - 2=variable max limit b16..b17 - 0=no min limit - 1=constant min limit - 2=variable min limit b24..b26 Precision (real numbers) b27..b31 Width (cfr. § 1.7.2)
pVisVar	word32	Flag of visibility. Available values: - *FALSE* = object not visible - *TRUE* = object always visible - var_addr = address of boolean variable
pCanSel	word32	Available values: - *FALSE* = object not selected - *TRUE* = object always selected - var_addr = address of boolean variable
order	byte	Number for establish a sequential selection.

Frame structure: FB_EDITBOX01		
accMode	byte	- kACS_IDLE = no effect - kACS_INIT = first draw on display - kACS_PRINT = update draw on display - kACS_SELECT = update draw on display when selected - kACS_MODIFY = enter in editing mode
Output variable	**Type**	**Description**
outKey	char	Key code for exiting editing-mode.

FUNCTION BLOCK: TextBox

Text box object

Frame structure: FB_TEXTBOX		
Local variables	**Type**	**Description**
memVis	byte	Visibility status of the previous execution.
base	word16	Number of first line seen in object.
Input variables	**Type**	**Description**
wHnd	word16	Handle of the object. Must be unique among textbox objects.
x1	word16	Top-left 'x coordinate' edge relative to full page.
y1	word16	Top-left 'y coordinate' edge relative to full page.
x2	word16	Bottom-right 'x coordinate' edge relative to full page.
y2	word16	Bottom-right 'y coordinate' edge relative to full page.
pFont	word32	Address of font for drawing text. The font must be initialized with Video_AddFont.
style	byte	- 0 = flat - 1 = raised - 2 = sunken
foreCol	byte	Text color.
bckCol	byte	Background color.
bordPts	byte	Border thickness It is sensible only if style = 0
bordCol	byte	Border color. It is sensible when style = 0 bordPts > 0
LineNr	byte	- *FALSE* = hide line number - *TRUE* = show line number
rw	byte	- *FALSE*= read-only mode - *TRUE*= read-write mode
pVar	word32	Address of string variable. It cannot be *NULL*.
szpVar	word32	Size of pVar.

Frame structure: FB_TEXTBOX		
pVisVar	word32	Flag of visibility. Available values: - *FALSE*= object not visible - *TRUE*= object always visible - var_addr= address of boolean variable
order	byte	Number for establishing a sequential selection.
accMode	byte	Access mode. Available values: - kACS_IDLE = no effect - kACS_INIT = first draw on display - kACS_PRINT = update draw on display - kACS_SELECT = update draw on display when selected - kACS_MODIFY = enter editing mode - kACS_SCROLLUP= scroll up one line - kACS_SCROLLDW= scroll down one line
rqCursPos	word16	Char Index where move the cursor.
rqCursRow	word16	Row to select.
dispCurs	byte	- *TRUE*= the cursor is always visible even if it is not enabled editing mode - *FALSE*= the cursor is visible only if it is enabled editing mode.
dispRow	byte	- *TRUE*= the row selection is always visible even if it is not enabled editing mode - *FALSE*= the row selection is visible only if it is enabled editing mode
bckSelCol	word32	Future developments.
wParam	word32	Future developments.
lParam	word32	Future developments.
Output variable	**Type**	**Description**
outKey	char	Key code for exiting editing-mode.
outCursPos	word16	Char index where there is the cursor.
outCursRow	word16	Index of selected row.

FUNCTION BLOCK: Progress

Progress bar object

Frame structure: FB_PROGRESS		
Local variables	**Type**	**Description**
memVis	byte	Visibility status of the previous execution
memVal	word32	Progress status of the previous execution
Input variables	**Type**	**Description**
wHnd	word16	Handle of the object. Must be unique among progress objects.
x1	word16	Top-left 'x coordinate' edge relative to full page.
y1	word16	Top-left 'y coordinate' edge relative to full page.

Frame structure: FB_PROGRESS		
x2	word16	Bottom-right 'x coordinate' edge relative to full page.
y2	word16	Bottom-right 'y coordinate' edge relative to full page.
style	byte	- 0 = flat - 1 = raised - 2 = sunken
barCol	word32	Color of step bar.
bckCol	word32	Background color.
bordPts	byte	Border thickness. It is sensible only if style = 0
bordCol	word32	Border color. It is sensible when style = 0 bordPts > 0
pVar	word32	Step variable. This is the filling percentage of bar in relation with the range assigned by pMin and pMax.
type	byte	Type of pVar. Assigned values: tyBool; tySInt; tyUSInt; tyByte; tyInt; tyUInt; tyWord; tyDInt; tyUDInt; tyDWord
pMin	word32	Min value for edit-box variable. If bit b0 (LSB) of field *format* contain 0 is a constant limit, if contain 1 it is a variable limit.
pMax	word32	Min value for edit-box variable. If bit b1 (LSB) of field *format* contain 0 is a constant limit, if contain 1 it is a variable limit.
format	word32	View format encoded in bit: b0 - 0 = pMin contains a constant value of Type 'type' - 1 = pMin contains the address of variable of Type 'type' b1 - 0 = pMax contains a constant value of Type type - 1 = pMax contains the address of variable of Type type b2 - 0 = horizontal orientation - 1 = vertical orientation
pVisVar	word32	Flag of visibility. Available values: - *FALSE* = object not visible - *TRUE* = object always visible - var_addr = address of boolean variable
accMode	byte	Access mode. Available values: - kACS_IDLE = no effect - kACS_INIT = first draw on display - kACS_PRINT = update draw on display

Frame structure: FB_PROGRESS		
Output variable	**Type**	**Description**

FUNCTION BLOCK: CustomCtrl

Embedded function block which implements custom control

Frame structure: FB_CUSTOMCTRL		
Local variables	**Type**	**Description**
memVis	byte	Visibility status of the previous execution.
ptrFunct	word32	Address of function that implements *Type* wCtrlID.
data0	word32	Local variable.
data1	word32	Local variable.
data2	word32	Local variable.
data3	word32	Local variable.
Input variables	**Type**	**Description**
wHnd	word16	Handle of the object. Must be unique among custom control objects.
x1	word16	Top-left 'x coordinate' edge relative to full page.
y1	word16	Top-left 'y coordinate' edge relative to full page.
x2	word16	Bottom-right 'x coordinate' edge relative to full page.
y2	word16	Bottom-right 'y coordinate' edge relative to full page.
wCtrlID	word16	Identifier of custom control.
pVisVar	word32	Flag of visibility. Available values: - *FALSE* = object not visible - *TRUE* = object always visible - var_addr = address of boolean variable
refr	byte	Request refresh: - *FALSE* = the object is redrawn only when the page is opening or return from child page - *TRUE* = the object is always redrawn
accMode	byte	Access mode. Available values: - kACS_IDLE = no effect - kACS_INIT = first draw on display - kACS_PRINT = update draw on display The value greater than 200 can be used for custom purpose.
wParam	word16	16 bit data without sign, used for custom purpose
lParam	int32	32 bit data with sign, used for custom purpose
rParam	float	32 bit real data with sign, used for custom purpose
Output variable	**Type**	**Description**

FUNCTION BLOCK: Chart

Chart object

Frame structure: FB_CHART		
Local variables	**Type**	**Description**
memVis	byte	Visibility status of the previous execution
grx1	word16	Top-left 'x coordinate' edge relative to full page.
gry1	word16	Top-left 'y coordinate' edge relative to full page.
grx2	word16	Bottom-right 'x coordinate' edge relative to full page.
gry2	word16	Bottom-right 'y coordinate' edge relative to full page.
pChart	word32	Handle of the chart created after ACS_INIT.
lastIdxSamples	word32	Actual index of inserted track data.
Input variables	**Type**	**Description**
wHnd	word16	Handle of the object. Must be unique among chart objects.
x1	word16	Top-left 'x coordinate' edge relative to full page.
y1	word16	Top-left 'y coordinate' edge relative to full page.
x2	word16	Bottom-right 'x coordinate' edge relative to full page.
y2	word16	Bottom-right 'y coordinate' edge relative to full page.
pFont	word32	Address of font for drawing text. The font must be initialized with Video_AddFont.
style	byte	- 0 = flat - 1 = raised - 2 = sunken
bordPts	byte	Border thickness. It is sensible only if *style* = 0
bordCol	byte	Border color. It is sensible when *style* = 0 *bordPts* > 0
backCol	byte	Background color.
pNSamples	word32	Address of the number of available samples to add in the chart. This value is used only when refresh is *TRUE*.
tyNSamples	byte	Type of the number of samples.
tyXOffset	byte	Type of the offset of X-axis.
pXOffset	word32	Address of the offset of X-axis. (move right-left the chart in order to 0).
tyTrackRight	word16	Type of default track for right Y-Axis
pTrackRight	word32	Address of the track of right Y-Axis. (if 0 the right label will not drawn).
tyTrackLeft	word16	Type of default track for left Y-Axis.

Frame structure: FB_CHART		
pTrackLeft	word32	Address of the track of left Y-Axis. (if 0 the left label will not drawn).
formatLeft	word32	Label format of left Y-Axis.
formatRight	word32	Label format of right Y-Axis.
format	Word32	Label format of X-Axis.
iDivGridX	word16	Number of division on horizontal bar , used with scale factor and offset for drawing the chart tracks (Ex. scale X=1, iDivGridX = 5 value between 0 and 5). Sensible even if the grid is not visible.
iDivGridY	word16	Number of division on vertical bar , used with scale factor and offset for drawing the chart tracks (Ex. scale Y=1, iDivGridY = 5 value between 0 and 5). Sensible even if the grid is not visible.
fGrid	byte	Draw grid: - *FALSE* = grid not visible - *TRUE* = grid visible
iXLabelDiv	word16	Step for X-axis labels. How many division of horizontal bar must have labels.
tyXScaleType	byte	Type of X-Axis scale.
pXScale	word32	Address of Scale factor of x-Axis. Value range among two division of horizontal bars. 0 value indicate that the chart is in auto-scale mode.
pClearVar	Word32	Address of boolean variable. If it is *TRUE* the chart delete all the previous data.
accMode	byte	Access mode. Available values: - kACS_IDLE = no effect - kACS_INIT = first draw on display - kACS_PRINT = update draw on display - kACS_CLOSE = close the chart and delete all the data
pXData	word32	Address or constant for X-Axis definition. Available values: - constant: number of samples * constant start with 0 - variable = array that contains pNSamples samples with X-axis value
tyXData	byte	Type of 'pXData' array. If tyXData = tyUndefined is a constant.
XlabelCol	word32	Color of X-Axis label.
iDotStep	word16	Space among two points of grid in pixel. The property is sensible if the grid is visible.
iSampleBuffer	word16	Number of samples that the run-time can store. The older ones are deleted if the size is exceeded.

Frame structure: FB_CHART		
arXBars	word32[3]	Array of addresses of vertical bars. If 0 the vertical bar is not defined, otherwise the address of variable or constant value.
arXBarsType	word16[3]	Type of variable that indicates the value of vertical bars. If arXBarsType[n] = tyUndefined and arXBars[n] is not *NULL*, the value of arXBars[n] is a numeric costant.
arXBarsCol	word32[3]	Colors of vertical bars.
GridCol	word32	Color of grid.
BorderGridColor	word32	Color of border of grid.
pVisVar	word32	Flag of visibility. Available values: - *FALSE* = object not visible - *TRUE* = object always visible - var_addr = address of boolean variable
arTrkData	word32[8]	Array of addresses of data. The nth of arTrkData contains the address of first elements of array of nth track. If address is *NULL* the track is not define.
arTrkType	byte[8]	Array of data. The nth of arTrkType contains the type of nth elements of arTrkData. This value is sensible only if the element of arTrkData is not *NULL*.
arTrkCol	byte[8]	Array of track colors. This value is sensible only if the element of arTrkData is not *NULL*.
arTrkVis	word32[8]	Array of visibility flags. The nth element of arTrkMinY determines the visibility of the track: - *FALSE* = track not visible - *TRUE* = track always visible - var_addr = address of boolean variable This value is sensible only if the element of arTrkData is not *NULL*.
arTrkScaleY	word32[8]	Array of Y-axis scale. The range of samples for every horizontal division.
arTrkScaleType	word16[8]	Type of variable of Y-Axis scale. If constant value arTrkScaleType[n] = tyUndefined.
arTrkOffset	word32[8]	Array of offset of Y-Axis for every track. The displacement of the track from 0 high and low.
arTrkOffsetType	word16[8]	Type array of offset of Y-Axis for every track. If constant value arTrkOffsetType[n] = tyUndefined.
iYLabelDiv	word16[8]	Array that contains on every step draw the Y-Axis label.
arTrkBarValue	word32[8*3]	Array of addresses of variables for horizontal bars.
arTrkBarValueType	word16[8*3]	Array of types of variable for horizontal bars.

Frame structure: FB_CHART		
arTrackBarName	word32[8*3]	Array of names for horizontal bars.
arTrkBarCol	word32[8*3]	Array of colors for horizontal bars.
Output variable	**Type**	**Description**

FUNCTION BLOCK: Trend

Trend object

Frame structure: FB_TREND		
Local variables	**Type**	**Description**
memVis	byte	Visibility status of the previous execution.
grx1	word16	Top-left 'x coordinate' edge relative to full page.
gry1	word16	Top-left 'y coordinate' edge relative to full page.
grx2	word16	Bottom-right 'x coordinate' edge relative to full page.
gry2	word16	Bottom-right 'y coordinate' edge relative to full page.
pChart	word32	Handle of the chart created after ACS_INIT.
FirstSamplingTimeS	word32	Sampling time in seconds take on ACS_INIT or when cleared.
FirstSamplingTimeMS	word16	Sampling time in milli-seconds take on ACS_INIT or when cleared.
LastSamplingTimeS	word32	Sampling time in seconds take every acquisition.
LastSamplingTimeMS	word16	Sampling time in millisecond take every acquisition.
InitDraw	Byte	If *TRUE* the trend is just drawn.
Input variables	**Type**	**Description**
wHnd	word16	Handle of the object. Must be unique among chart objects.
x1	word16	Top-left 'x coordinate' edge relative to full page.
y1	word16	Top-left 'y coordinate' edge relative to full page.
x2	word16	Bottom-right 'x coordinate' edge relative to full page.
y2	word16	Bottom-right 'y coordinate' edge relative to full page.
pFont	word32	Address of font for drawing text. The font must be initialized with Video_AddFont.
style	byte	- 0 = flat - 1 = raised - 2 = sunken

Frame structure: FB_TREND		
bordPts	byte	Border thickness It is sensible only if *style* = 0
bordCol	byte	Border color. It is sensible when *style* = 0 bordPts > 0
backCol	byte	Background color.
pNSamples	word32	Inherited from chart but contains acquisition time in seconds.
tyNSamples	byte	Not used.
tyXOffset	byte	Type of the offset of X-axis.
pXOffset	word32	Address of the offset of X-axis (move right-left the chart in order to 0)
tyTrackRight	word16	Type of default track for right Y-Axis.
pTrackRight	word32	Address of the track of right Y-Axis (if 0 the right label will not drawn).
tyTrackLeft	word16	Type of default track for left Y-Axis.
pTrackLeft	word32	Address of the track of left Y-Axis (if 0 the left label will not drawn).
formatLeft	word32	Label format of left Y-Axis.
formatRight	word32	Label format of right Y-Axis.
format	Word32	Label format of X-Axis. Available values: - 0 = ss - 1 = mm.ss - 2 = hh.mm - 3 = hh.mm.ss
iDivGridX	word16	Number of division on horizontal bar, used with scale factor and offset for drawing the chart tracks. (Ex. scale X=1, iDivGridX = 5 value between 0 and 5). Sensible even if the grid is not visible.
iDivGridY	word16	Number of division on vertical bar, used with scale factor and offset for drawing the chart tracks (Ex. scale Y=1, iDivGridY = 5 value between 0 and 5). Sensible even if the grid is not visible.
fGrid	byte	Draw grid: - *FALSE* = grid not visible - *TRUE* = grid visible
iXLabelDiv	word16	Step for X-axis labels. How many division of horizontal bar must have labels.
tyXScaleType	byte	Type of X-Axis scale.
pXScale	word32	Address of Scale factor of x-Axis. Value range among two division of horizontal bars. 0 value indicate that the chart is in auto-scale mode.
pClearVar	Word32	Address of boolean variable. If it is *TRUE* the chart delete all the previous data.

Frame structure: FB_TREND		
accMode	byte	Access mode. Available values: - kACS_IDLE = no effect - kACS_INIT = first draw on display - kACS_PRINT = update draw on display - kACS_CLOSE = close the chart and delete all the data
XlabelCol	word32	Address or constant for X-Axis definition. Available values: - constant: number of samples * constant start with 0 - variable: array that contains pNSamples samples with X-axis value
iDotStep	word16	Type of pXData array. If tyXData = tyUndefined is a constant.
iSampleBuffer	word16	Color of X-Axis label.
arXBars	word32[3]	Space among two points of grid in pixel. The property is sensible if the grid is visible.
arXBarsType	word16[3]	Number of samples that the run-time can store. The older ones are deleted if the size exceeds.
arXBarsCol	word32[3]	Array of addresses of vertical bars. If 0 the vertical bar is not defined, otherwise the address of variable or constant value.
GridCol	word32	Type of variable that indicates the value of vertical bars. If arXBarsType[n] = tyUndefined and arXBars[n] is not *NULL*, the value of arXBars[n] is a numeric costant.
BorderGridColor	word32	Colors of vertical bars.
pVisVar	word32	Color of grid.
arTrkData	word32[8]	Color of broder of grid.
arTrkType	byte[8]	Flag of visibility. Available values: - *FALSE* = object not visible - *TRUE* = object always visible - var_addr = address of boolean variable
arTrkCol	byte[8]	Array of addresses of data. The nth of arTrkData contains the address of first elements of array of nth track. If address is *NULL* the track is not define.
arTrkVis	word32[8]	Array of data. The nth of arTrkType contains the type of nth elements of arTrkData. This value is sensible only if the element of arTrkData is not *NULL*.
arTrkScaleY	word32[8]	Array of track colors. This value is sensible only if the element of arTrkData is not *NULL*.

Frame structure: FB_TREND		
arTrkScaleType	word16[8]	Array of visibility flags. The nth element of arTrkMinY determines the visibility of the track: - *FALSE* = track not visible - *TRUE* = track always visible - *var_addr* = address of boolean variable This value is sensible only if the element of arTrkData is not *NULL*.
arTrkOffset	word32[8]	Array of Y-axis scale. The range of samples for every horizontal division.
arTrkOffsetType	word16[8]	Type of variable of Y-Axis scale. If constant value arTrkScaleType[n] = tyUndefined
iYLabelDiv	word16[8]	Array of offset of Y-Axis for every track. The displacement of the track from 0 high and low.
arTrkBarValue	word32[8*3]	Type array of offset of Y-Axis for every track. If constant value arTrkOffsetType[n] = tyUndefined
arTrkBarValueType	word16[8*3]	Array that contains on every step draw the Y-Axis label.
arTrackBarName	word32[8*3]	Array of addresses of variables for horizontal bars.
arTrkBarCol	word32[8*3]	Array of types of variable for horizontal bars.
Output variable	**Type**	**Description**
---		

(1) Available figures and colors depend on target's features.