# The Basics of ASiV

TM610

Perfection in Automation

**www.br-automation.com**

## Prerequisites

| | |
|---|---|
| Training modules: | TM210 – The Basics of Automation Studio |
| | TM600 – The Basics of Visualization |
| Software: | Automation Studio 2.5 |
| | Automation Runtime 2.90 |
| Hardware: | 4PP420.1043-75<br>PC with at least 256 MB (512 MB recommended)<br>Windows XP / Windows 2000 |

## Table of contents

# 1. INTRODUCTION

This training module will provide a quick introduction to visualization with Visual Components. B&R Visual Components is the right tool for creating simple or complex visualization applications.

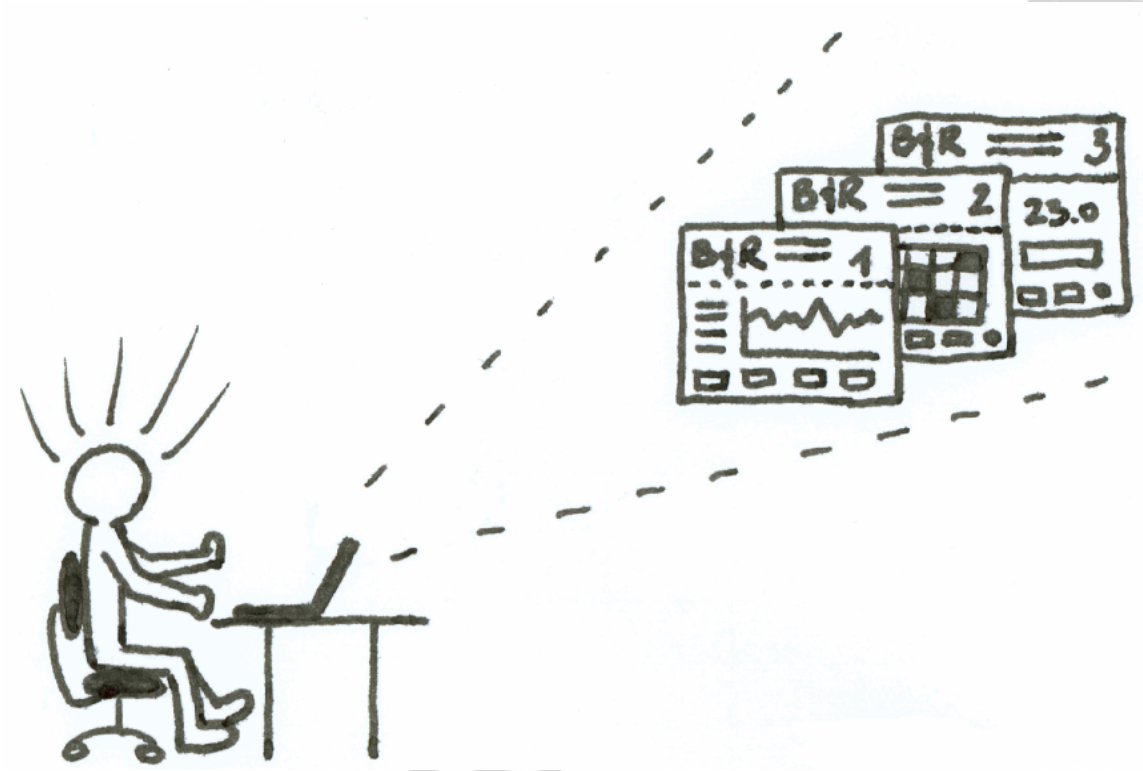



Fig. 1 Introduction – Visual Components Editor

With the help of a sample project, participants will learn which steps are necessary for creating a visualization application and how to effectively use the Editor components to create their own visualization application.

## 1.1 Objectives

By the end of this training module, the course participant should to be able to create a simple visualization application using the Visual Components Editor.

The course participant will receive a guide to learn about and understand the concept and possibilities of Visual Components.

This training module also demonstrates how to get a visualization project up and running on the target system within a few minutes.

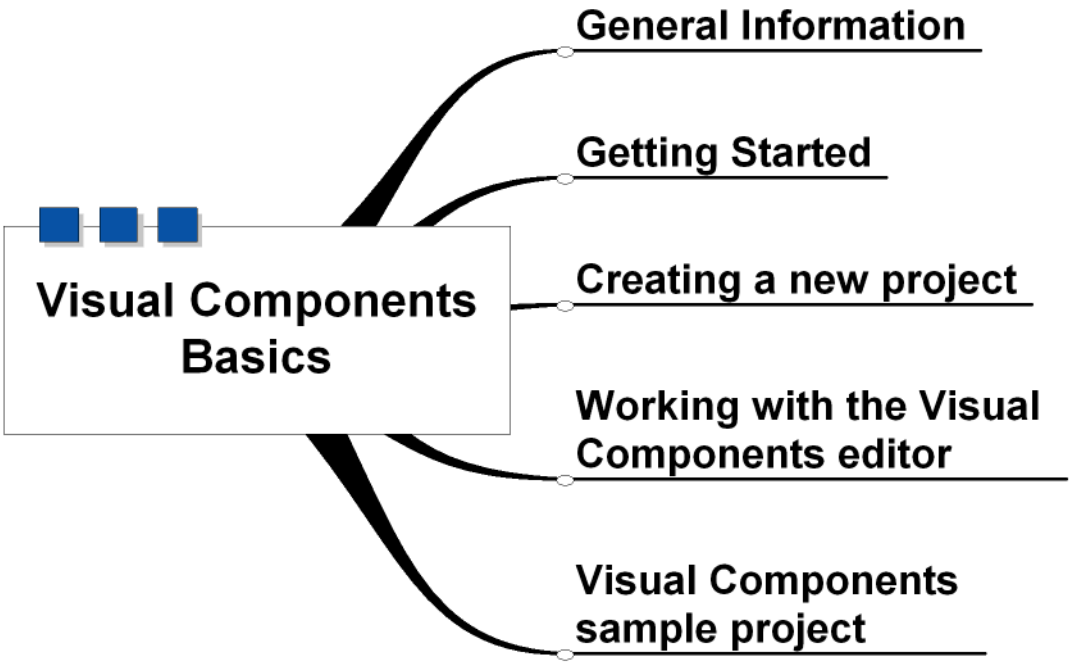



Fig. 2 Overview

# 2. GENERAL INFORMATION

## 2.1 What is Visual Components?

Visual Components is a visualization integrated in Automation Studio. This means that the visualization application (visualization objects) is managed, edited and executed together with the control application (control / positioning tasks). Depending on the target hardware being used, displays are supported ranging from 2x20 line displays to XGA displays (1024 x 768) with True Color (32bit).

## 2.2 Integrated visualization

The process images are displayed on the Automation Runtime Target.

Unlike ordinary visualization – where the visualization runs separate from the control unit and the displayed values are read by the controller via separate communication – the visualization runs together with the controller tasks.

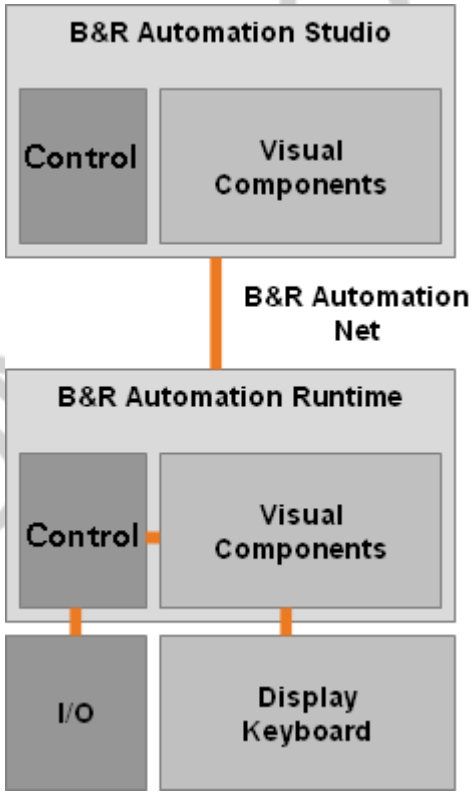



Fig. 3 Integrated visualization

## 2.3 Visual Components history

"In the beginning there was ..."

Historical background of Visual Components:

**BRRT Operator Panels**

These panels were "programmed" by the software engineer using ESC sequences. In the early days, the user had to program the visualization on his/her own. Later on, function blocks were provided for the mostly text-oriented displays and for user input.

**PCS** (PANELWARE Configuration Software) – ca. 1992

The MS-DOS based PCS software made it finally possible to create and manage visualization on a PC. The phrase "configuring instead of programming" became a reality for the first time ever. The project was then transferred to and started on an "intelligent" PANELWARE controller.

**Panel Studio** – ca. 1997

Implementation of PCS on a Windows basis.

**Visual Components** (starting with Automation Studio version 2.0) – ca. 2001

Integration of Panel Studio in Automation Studio ⇔ Integrated Visualization.

**Visual Components SG4** (starting with Automation Studio version 2.4.1) 2004/2005

Current version of Visual Components for all panels with VGA display. The Automation Studio version now also contains the "classic" Visual Components Editor for creating and editing SG3 targets.

# 3. GETTING STARTED

## 3.1 Installation requirements

**Required hardware /software:**

- PC with at least 256 MB RAM (512 MB to 1 GB recommended for extensive projects)
- 1 GHz or higher processor
- Operating system: Windows XP Professional / Windows 2000 Professional
- Automation Studio V2.4.1 or higher

**"Processing power is not as relevant to the processing speed of Windows programs as much as the RAM is."**

The Microsoft .NET Framework is required for running Automation Studio / Visual Components SG4. It will be installed automatically if required.

## 3.2 Installation of Visual Components

Visual Components is a part of Automation Studio.


Fig. 1: Automation Studio Setup

## 3.3 Additional resources

**"Help!"**

Help is an important tool when creating a Visual Components visualization. The Visual Components training documents constantly refer to the Help files, because they are available at all times and most importantly during the project creation (as long as they were installed!).

# 4. CREATING A NEW PROJECT

The selection of hardware and software is obviously a crucial part of creating a new project.

## 4.1 Preparing to create a project

It is important to define the hardware before creating a project. This training documentation is based on a PowerPanel with 10.4" diagonal (4PP420.1043-75).

Fig. 2 4PP420.1043-75

Any other SG4 Target can be used if this panel is not available.


Fig. 3 5CFCRD.0064-03

A CompactFlash card with at least 32 MB and 24V supply voltage is required to transfer the project.

We recommend transferring the project to the CompactFlash using a USB CF Reader (desktop/laptop) or a PCMCIA / CF Adapter (laptop) (more information about that later).

## 4.2 My first Visual Components project

The goal of this chapter is to create a new project and to transfer it right to the target hardware.

All required Visual Components runtime modules are transferred to the target system and a connection to the target hardware is established. Essentially, it does not matter whether the connection is made via the serial interface or via Ethernet – however we recommend using Ethernet to achieve optimum transfer rates.

**Exercise: Creating a new project**

Steps necessary for the first practice example:

- Create a new project with the target hardware 4PP420.1043-75
- Set Ethernet communication on the PC and PowerPanel
- Transfer the project to a CF using the Automation Software tool "PVI Transfer"

### 4.2.1 Selecting a new project with target hardware

We assume that each participant already knows how to create a new project using Automation Studio. It is important that a PowerPanel (i.e. an Automation Runtime Target with integrated visualization) is selected as target hardware.

The "**New Project**" wizard is opened via the menu [**File**] / [**New Project**] or the 🗋 button from the toolbar. The directory and project name must be defined first.

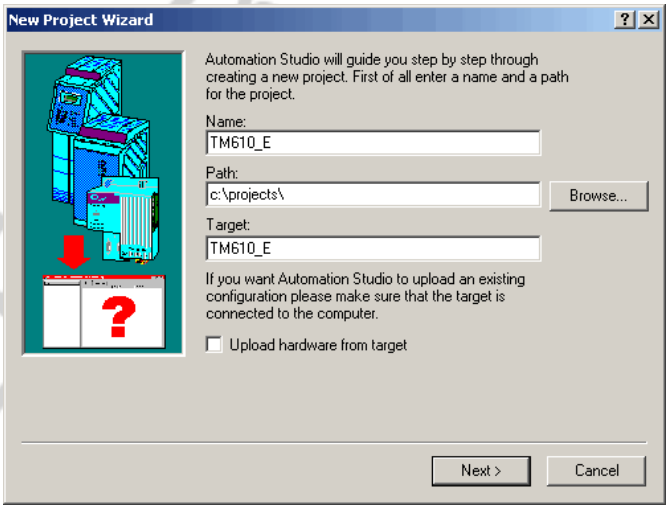



Fig. 4 New project

The product group "**PowerPanel**" must be selected after confirming the dialog box with the <**Next**> button.

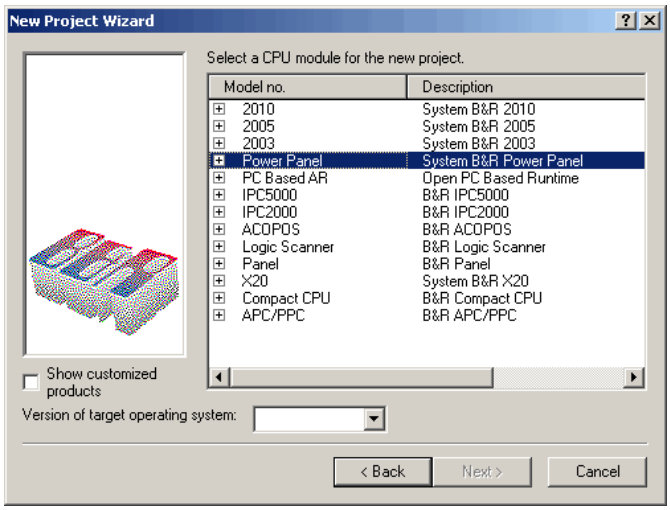



Fig. 5 New project – Product selection

The desired target hardware "**4PP420.1043-75**" is selected in this dialog box. The option "**Show customized products**" must be selected to choose a custom display.

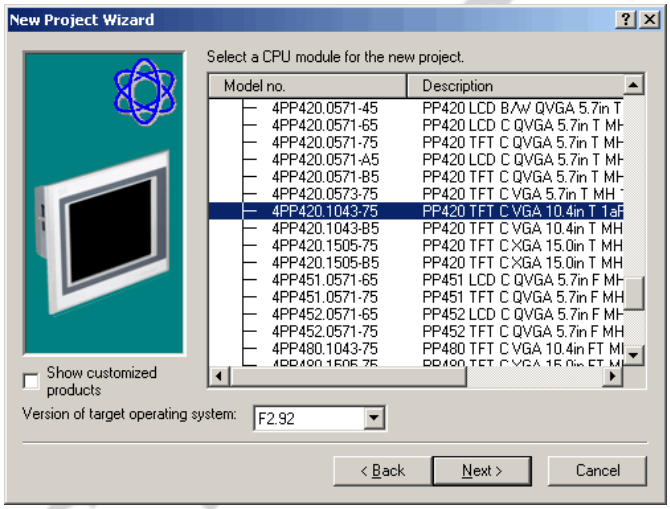



Fig. 6 New project – Component selection

After confirming the dialog box with <**Next**>, the project creation wizard can be closed in the next dialog box by clicking <**Finish**>.

The project structure of the new Automation Studio project has now been created on the hard disk.

The next step is to define the name of the visualization application. This name is limited to 6 characters. For this training, we will create a new panel object with the name "**visu**".

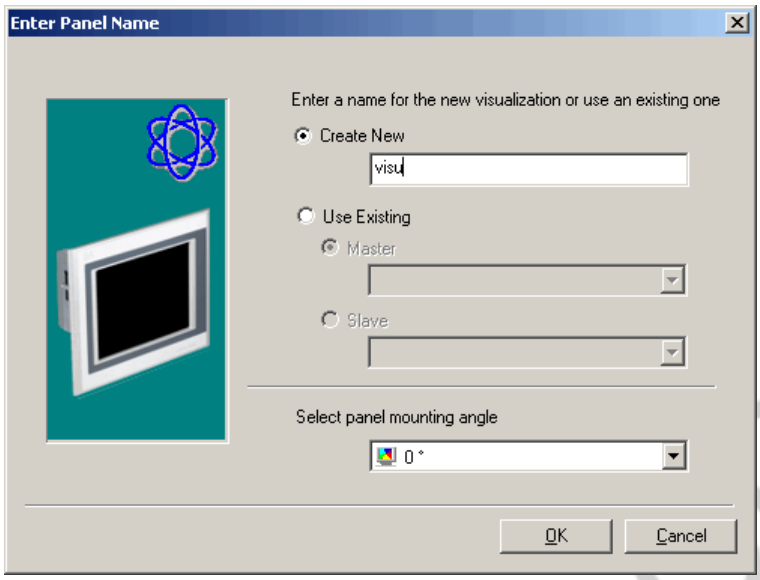



Fig. 7 Entering the name of the visualization application

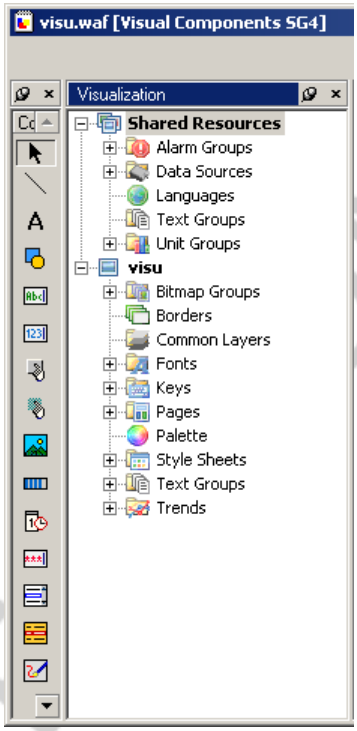



Fig. 8: Visual Components Editor – Project view

After clicking <OK>, all resources required for working with Visual Components are loaded and the Visual Components Editor is opened.

All questions regarding this view are answered under "Section 5: Working with the Visual Components Editor".

First we will close the Visual Components Editor (referred to from now as VC Editor).

**Question**

What is the goal of this exercise?

**Answer**

To create a new project and transfer it to the target system.

After closing the VC Editor, we will be in the Automation Studio project view.

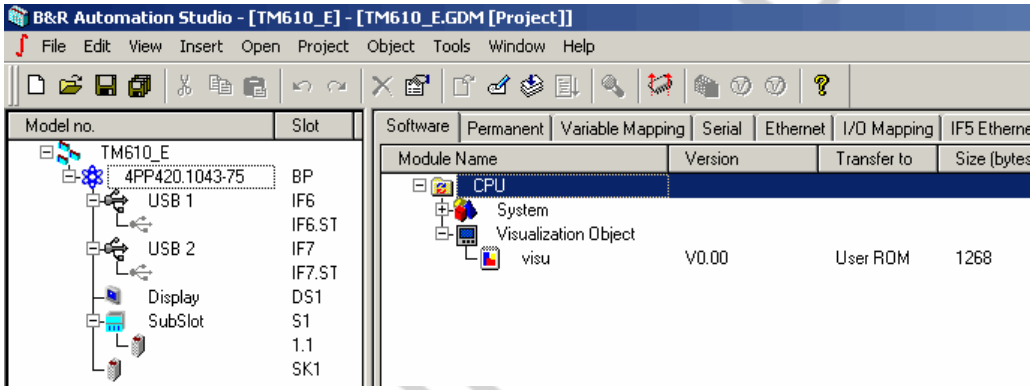



Fig. 9 Automation Studio project view

### 4.2.2 Transferring the project to CompactFlash

There are two ways to quickly transfer the project to the CompactFlash card:

- The AR operating system is already installed on the CompactFlash card (project transferred via an online connection)
- The project is transferred to the CompactFlash card using the Automation Software Tool "PVI Transfer"

**Creating a Compact Flash**

Select Tools: Generate Compact Flash from the main menu to create the Compact Flash:

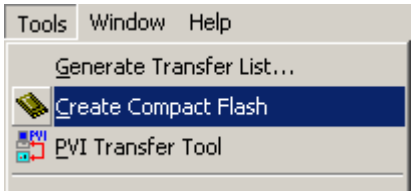



Fig. 10: Creating a Compact Flash

**Further steps in the PVI Transfer Tool:**

- **Select disk**
  Select a CompactFlash located in a corresponding Compact Flash adapter.
- **Generate disk**
  A bootable CompacFlash card is created within a minute depending on the size of the card.

**Get the Power Panel ready for operation:**

- Plug the CompactFlash card into the CompactFlash slot while PowerPanel is turned off
- Apply 24 volt supply to the PowerPanel

**Result:**

- The Power Panel boots up after the supply is connected
- A few seconds later, the boot screen changes to a blank, white page, which we will then add elements to in the next exercise.

**Goal of this exercise:**

- Create a new visualization project
- Transfer an entire Automation Studio project with visualization to a Compact Flash
- Test the target hardware

... With a little practice, this section should just take about 2 minutes, provided that Automation Studio is already started. The time was measured from "New Project" to "Create CompactFlash" ...

## 4.3 Adding elements to my first Visual Components project

In this section we will add a little life to the presently blank screen page. Section 5 will discuss working with the VC Editor.

**Exercise: Adding elements to the visualization**

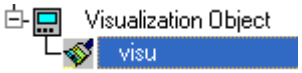
- Define a control variable
- Open the VC Editor
- Link the control variable with an input and output object
- Transfer the changes to the target system

### 4.3.1 Creating a user task and defining a variable

We will create a new user task "**panel**" using a local variable "**ActTemperature**" with the data type INT. Participants should already know how to do this (TM210).

### 4.3.2 Opening the visualization object "**visu**"

The VC Editor is opened by double clicking on the visualization object in the Automation Studio software tree, which we named "**visu**".




### 4.3.3 Drawing an input and output field and connecting with a variable

The goal of this exercise is to add a little life to the still empty page on the target system.

After opening the VC Editor, all of the components for a Visual Components visualization are displayed on the left side.

Each new Visual Components project automatically has a few pre-defined components (project template):

- Blank start page (we had a look at this in the last exercise on the target system)
- Operating element for numeric and alphanumeric entries on the touch screen

**The following steps will be performed in this exercise:**

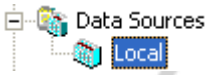
- All control variables created in Automation Studio are added to the VC Editor
- Search the blank page
- Insert two numeric fields
- Change the "**Input**" property in one of these fields to "True"; this property configures this field as an **input field**
- Link the two fields with the local control variable "ActTemperature"

**Add the variables defined in the control task to the VC Editor**

After being opened, the VC Editor does not yet have any information about the variables used in the Automation Studio project. These variables must first be added to the "**Data sources**".

There are now 2 steps necessary to add the variables:

- Select the "**Data sources**" node and double click on the item "**Local**"




- Add the data points which are not yet provided in the project by pressing the "**Update data points**" button now shown in the toolbar, or by pressing <**F5**>.

### Search empty page in the visualization project

The process images are managed in the "**pages**" node in the VC Editor. Similar to working with Windows Explorer, the deeper layers of the nodes can be opened or closed by clicking on the icon <+> or <->.

The **start page** is displayed in the center of the screen by double clicking on the "**InitPage**" layer.


The layer is opened in exclusive mode by double clicking on the "**Default**" layer.

### Inserting numeric fields

As long as nobody has hidden or closed the individual docking windows, the toolbar (i.e. all available tools necessary for creating a page) is displayed on the left side.

If the toolbar is not shown, it can be opened by using the menu [**View**] / [**Toolbar**].


Fig. 11: Basic tools

We will now select the numeric control "123" from this toolbar and place it on the page by clicking and dragging a rectangle to the desired size. After repeating this procedure a second time, our screen should appear as follows:

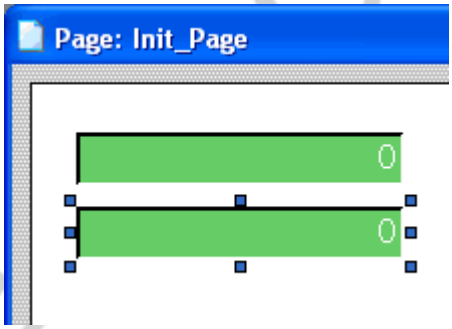



Fig. 12 Inserting an input and output field

In this exercise, it is not necessary for the alignment, size or design to be perfect.

### From the output to input field

If a numeric field were added, it would function as an output field during runtime.

The properties of this field must now be changed to achieve the function of an input field.

Each object placed on a page has "**Properties**", which affect the runtime behavior.

By default, the properties are displayed on the right side.

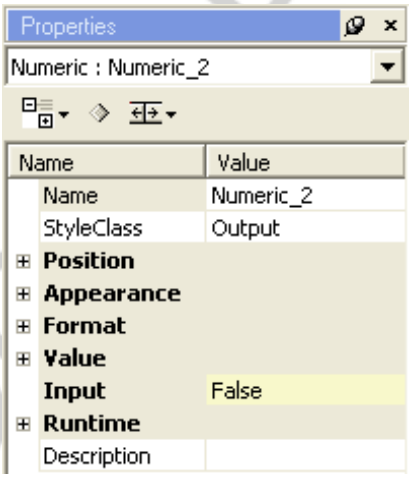



Fig. 13: Output field properties

The "**input**" property must now be changed to "**true**" in order to configure an output field for input. As a result of this change, the object receives additional input properties, from which we select the one for numeric touch input for the "**TouchPad**" property.

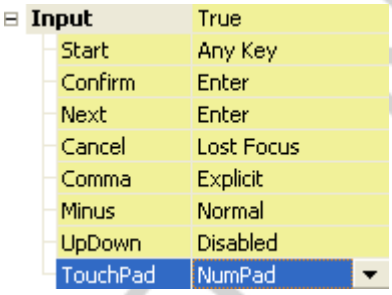
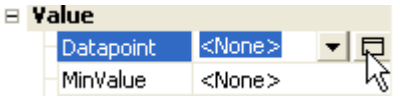



Fig. 14 Input field properties

### Linking the variable with the input and output fields

The control variable "VisuTemperature" must be linked with the fields in order to enter dynamic values or to make any changes to the values.

Linking is done using the "**Value**" – "**Data point**" property.




The control variable (referred to as data point from now on), must be selected using the <**Browse**> button (button on which the mouse is placed) since one has not yet been linked.

All data points then appear in the new dialog box sorted based on validity (local / global). Local data points are shown in the task in which they are used.

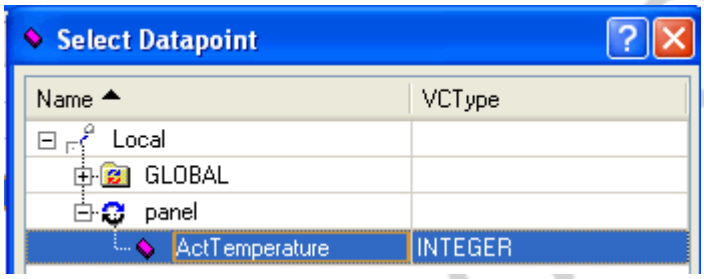



Fig. 15 Selected data point dialog box

Our local variable "**ActTemperature**" can be found in the "panel" task. Once the selection has been made, the dialog box is closed by clicking <OK>. The variable has now been linked with the numeric field.

This procedure must also be repeated for the second numeric field.

Ethernet communication must still be configured for the project transfer in Automation Studio before we can transfer the compiled changes to the target system.

**Note:**

Information about configuring Ethernet communication in Automation Studio and for the target system can be found in the training modules TM212 / TM213.

**Goal of this exercise:**

- Transfer changes made in a visualization project to the target system via Ethernet.
- Made your first entries using a touch system.

# 5. WORKING WITH THE VISUAL COMPONENTS EDITOR

Now that we have a run-capable project and know how to open the Visual Components Editor (see 4.3 Adding elements to my first Visual Components project), we can start learning about the Editor window and the visualization objects:

- Working with the Visual Components Help
- Using the VC Editor
- Using the components in the VC Editor

## 5.1 The Visual Components Help

The Visual Components Help (referred to as VC Help from now on) is an important aid when working with the VC Editor to find detailed descriptions for the corresponding functions and components.

This training module will also occasionally refer to the VC Help in order to answer questions and to learn how to work with the help.

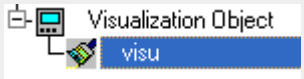
**Exercise: Visual Components editor window**

Open the Help and find the general description of the "Visual Components Editor window".

**Approach**

The VC Help is opened from the VC Editor by pressing the <F1> key.

The VC Editor is opened by double clicking on the visualization object.



**Result**

Each participant should have found the menu item "Finding your way in the editor".

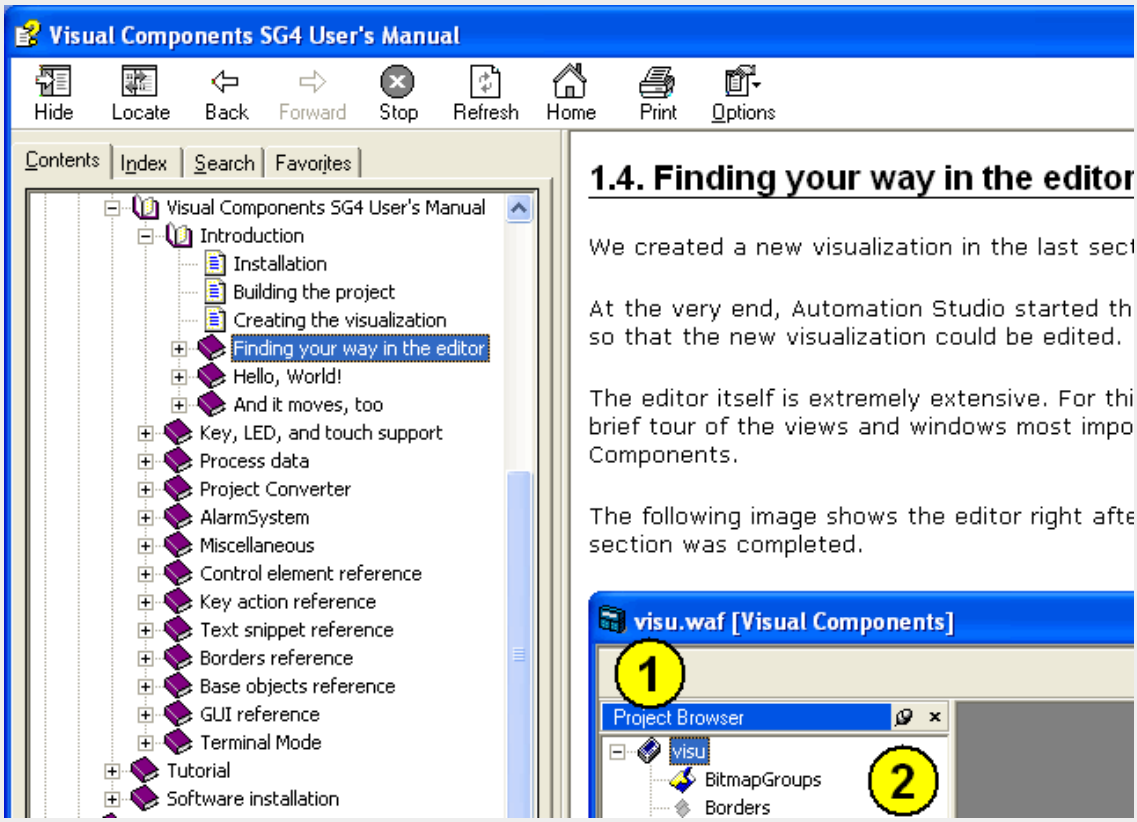



Fig. 16 Visual Components Help – Finding yourway in the editor

**Tip:**

We can leave the Help open and switch between Automation Studio and Help using <**ALT**> + <**TAB**>.

## 5.2 Using the Visual Components Editor

In this section, we will learn how to use the VC Editor. Those of you who have already worked with visualizations will feel right at home using this program.

The individual windows in the VC Editor can be edited by the user. This makes it possible to hide or to move any window to a desired position.

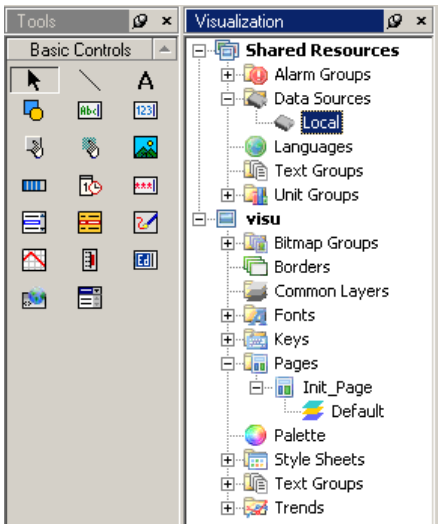



Fig. 17: dockable window

By default, the visualization tree and the main controls are displayed on the left side of the screen. You can hide a window by clicking on the "Pin" . You can release the window from its frame and place it individually on the screen by double clicking on the title bar.

**Exercise: Changing the design of the Editor window**

Over the next few minutes, we will try to change the design of the Editor window.

**Goal of this exercise**

Each participant should understand the docking behavior of the Editor.

## 5.3 The components in a visualization

### 5.3.1 The default project

When a new visualization is created it is immediately ready to be compiled and is run-capable (as we saw in the first exercise).

**What does our default project contain:**

- System alarms (TM640)
- Bitmap groups for touchpads and alarm system (TM640)
- Unit groups (TM650)
- Borders
- Default font (Bitstream Vera Sans)
- Start page (Init_Page)
- German and English languages (TM650)
- StyleSheets for color and monochrome
- Key configuration for touchpads
- Text groups for the alarm system (TM640)

**Note:**

The alarm system and the multi-lingual aspects are NOT discussed in this training module (TM640, TM650).

If you look back to our first practice example, you'll remember a blank page called the "Init_Page" was displayed here on the target system.

**Exercise: Opening the individual editors**

In this exercise, we will open the individual editors in the Visual Components project tree.

Goal of this exercise

Each participant should be familiar with the components in a Visual Components project. Working with these components will be the goal of the next section.

Throughout the course of this exercise, a few editors have been opened on the screen.

If too many windows are opened, the project will start to become a little unclear. The window dialog box can be opened using the menu [**Window**] / [**Window**] or the key combination <**CTRL**> + <**ALT**> + <**W**>.

This dialog box allows you to close one or all of the opened dialog windows.

**Note:**

Each opened window requires Windows memory. The program may slow down if too many windows are opened. The current view is saved when exiting the VC Editor and restored when the Editor is opened again. Therefore, the project may also require a little more time to open.

### 5.3.2 Conventions in Visual Components

A few guidelines must also be followed when working with the Visual Components Editor.

**Task: Visual Components conventions**

Learn the Visual Components conventions from the Visual Components Help. Why are conventions important? What conventions are there?

**Task help**

The conventions are located under "Miscellaneous".

# 6. VISUAL COMPONENTS SAMPLE PROJECT

In previous exercises, we have learned a few things about the VC Editor and its components.

In this section, a run-capable sample project created as a "real" application is used to explain the main components of VC.

Various tasks and exercises will be carried out in individual project steps to implement the sample project shown in the following image.

**"Learning by doing!"**

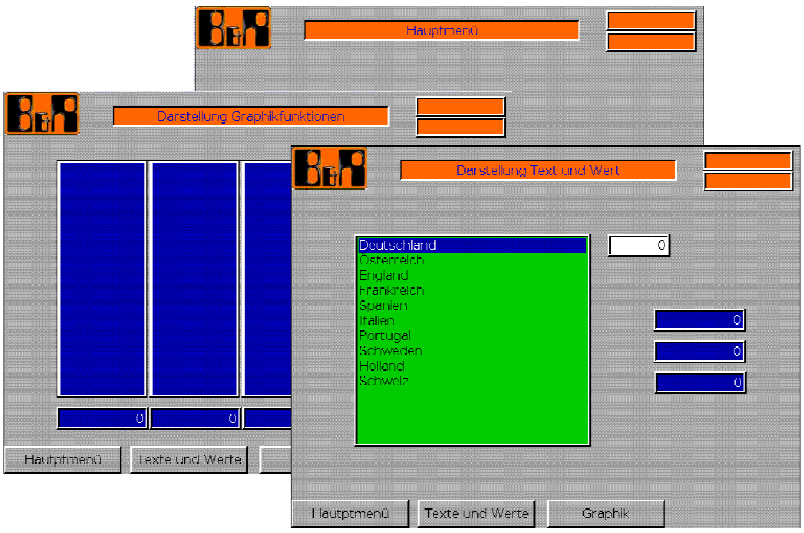



Fig. 18 Visual Components Editor – Sample program

**The following project sections will be explained and performed:**

- Visualization global settings
- Borders & style sheets
- Variable management
- The layering method in the page structure
- Page distribution
- Virtual keys and key actions
- Static and dynamic text
- Numeric and alphanumeric values
- Bars and graphic objects

## 6.1 Visualization global settings

A visualization application has global properties that can affect the runtime behavior of the visualization:

- Default language
- Color/monochrome setting for the start page and style sheets
- Screen saver (time definition and page selection)
- Runtime behavior (e.g. page change via variable, language switching via variable, etc.)

These settings are made right in the VC Editor at the highest layer of the selected visualization tree:

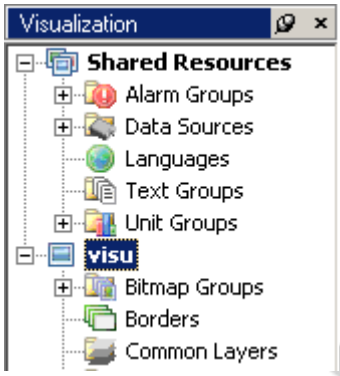



Fig. 19: selecting the visualization tree

Configuration takes place in the page properties:

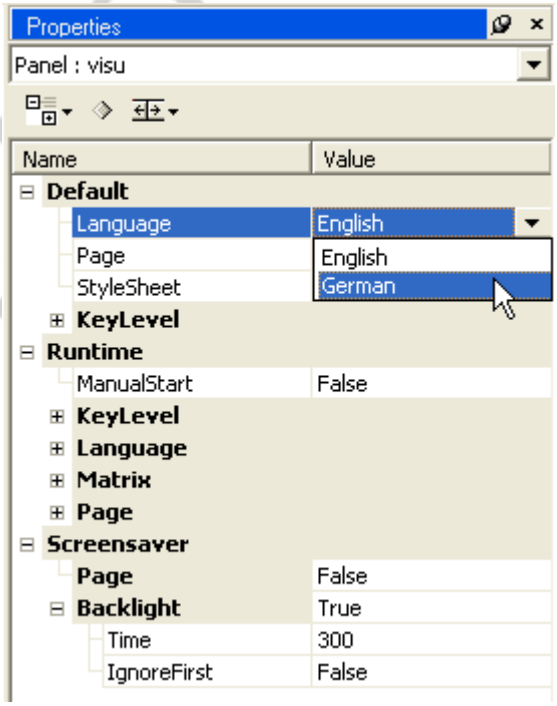



Fig. 20: Global project properties

**Task: Define default language**

Select the language "English".

After starting up, all of the texts are displayed on the target system in this language, unless the language is changed via a variable from the user task.

## 6.2 Borders and styles

### 6.2.1 Borders

Almost every visual object such as a text field, a numeric field or a button has a border. The appearance of these objects is not static. They can be individually designed by the user.

A few borders are already provided in the default project for most views. We will add a border without visible edges for displaying static text.

But what good is a border without visible edges?

In Visual Components, a border is not only determined by the border itself (style), but also by the "**Padding**", which defines the spacing of the text or graphic (the contents) to the edge of the border.

All borders in the project are shown by double clicking on the visualization node "**Borders**".

**Exercise: Borders for static text**

Create a new border without visible edges, which can be used to display static text.

**Solution**

A new "**Invisible**" border with the name "**Border_1**" is inserted using the <**Insert**> key or the context menu [Add border].

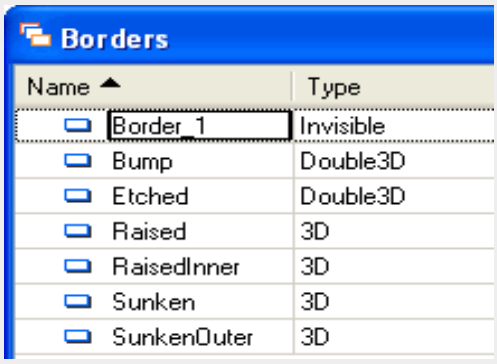



Fig. 21: Components – Borders

Now we will name this border "**Static_Text**" in the "**Name**" property. The type "**Invisible**" defines that the border will be invisible and the indentation will only be 2 pixels horizontal and 1 pixel vertical.

### 6.2.2 Style sheets

The appearance of the object (also called the template) is defined using a style sheet. As we saw in our second practice example (the one about input and output fields), these fields already have a few preset properties (e.g. foreground and background color).

The style sheets now offer the option to set an object's appearance, without having to adjust its properties after creating a page.

**Advantages of a style sheet:**

- Central management of properties
  E.g. Color setting for input and output fields
- Multiple style classes for one control
  E.g. various display bars for temperature, progress indicators
- Central management of color settings for color and monochrome displays
- Changing properties in one location for the entire project

**Task: Change the background color of the input field.**

In this exercise, we will change the background color of our input field.

This can be done directly in the input field's properties. If a second input field is added, its color properties must be changed again.

After making the change, you can check the results on the target system.

**Approach**

The background color is changed in the style sheets (we have a PP420 with color display!) for the numeric controls.

The properties group, "**Appearance**" must be changed because the appearance of the control was changed.

### 6.2.3 Style sheets for set values and actual values

**Task: Style for input and output field**

This task can be performed by each participant individually:

- Create a new style for displaying actual values (white background with black font)
- Create a new style for inputting set values (white font with blue background)

**Approach**

Add two new style classes to the style sheets for the control, "**Numeric**". "**OutputActualValue**" and "**InputSetValue**". The appearance and – don't forget – the input behavior are both defined in the properties.

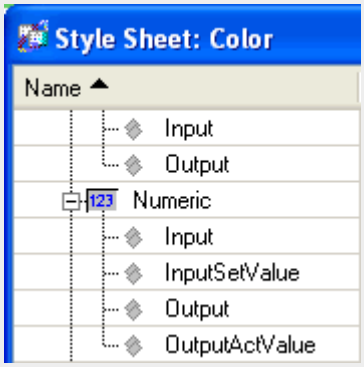



Fig. 22 Style Sheet – Input & Output

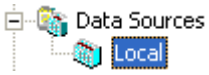
## 6.3 Variable management

The visualization application displays process sequences and allows the operator to intervene in the process. The process is displayed using a visual object such as a bar or a numeric value. Keys or buttons on the screen are used for actual operation.

The variable for a user task is managed in Visual Components as a data point and is the interface between the process sequence (control application) and the visualization.

### 6.3.1 Local data sources

In our second exercise, we have already inserted a data point to the visualization.




The variables in the control tasks are imported to the VC Editor as "data points" in the "**local data source**".

Now have a look at how the "**ActTemperature**" variable from the "panel" visualization task is displayed.

| Name | PLCTy... | VCType | Unit Group / Subtype | Limit | PLCUnit |
|---|---|---|---|---|---|
| GLOBAL | | | | | |
| panel | | | | | |
| ActTemperature | INT | INTEGER | None | None | None |

Fig. 23 Data source – Local

The variable is displayed directly in the corresponding task because it was defined with the scope "**local**".

If the variables in the control task have changed or added, the view can be updated anytime by pressing <**F5**> or by pressing the button from the toolbar.

**PLC type**

The data type defined in the user task of the control project is displayed under PLC type.

**VC type**

The VC type defines the data types which can be used in the visualization and are dependent on the data type of the control variable. Possible data types:

- <None>
  This setting specifies that the variable should not be used as a data point in VC.
- CHAR
  A single character, whereby the output behavior is defined by the controller data type (8 or 16 bit) (UNICODE = training module TM650).
- INTEGER
  A whole-number value where the value range depends on the data type of the control variable (maximum INT32).
- SCALED
  This data type should be selected when scaling a variable (training module TM650). Real variables are displayed directly as SCALED.
- STRING
  Field (array) of process variables which together make up a character string (see also CHAR).

We will be working with the VC data types INTEGER, CHAR and STRING in this training module.

**Further information:**

- Visual Components Help – Process data
- Training module TM650

**Note:**

The **UnitGroup, limitation and PLC** unit are covered in detail in the training module TM650.

## 6.4 The layering method in the page structure

The layering method means that multiple layers can be used on one page (e.g.: for headers, main page, footers, etc).

**Advantages of the layering method:**

- Reoccurring image information that should be displayed at the same position only has to be drawn one time.
- Changes to global image information only have to be made at one position.
- While editing and creating images, layers can be hidden or their objects can be displayed simply as frames.

Furthermore, there is a difference between global (common) and local layers.

There are no limitations regarding the number of or mixing of local and common layers.

**Local layers**

A local layer can only be used on the page where it was created.

**Common layers**

A global layer can be used repeatedly on multiple pages. This means that image information used on multiple pages only has to be drawn one time.

**Exercise: Common layers for header and footer**

We will now create two common layers for displaying a header and footer. At the same time, we will also get to know the objects which we add to the layers.

A new, blank layer is inserted and displayed in the center of the screen by selecting the "**Common Layers**" node and clicking on the <**Insert**> key.


This new layer will be called "**Header**" ("**Name**" property).

Now we will repeat this procedure and add a new layer called "**Footer**".

The node view now shows 2 new common layers.


**Which language will we be using?**

In the global project settings, we already determined that we will use English during runtime (remember that by default, a project contains two languages after project creation).

We are now at a layer in which we must determine which language to use for text entry in the VC Editor. To do this, we will select the "**active language**" (= language for project creation) from a combo box in the toolbar.


Fig. 24 Toolbar – Language selection

**What should appear in the header:**

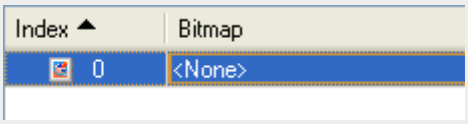
- Logo
- Date and time
- Page description

### 6.4.1 Displaying a Bitmap

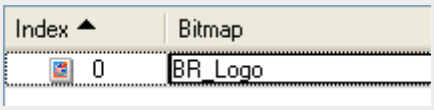
Let's start by displaying a bitmap (e.g. a company logo in the header).

**Exercise: Add a bitmap to the bitmap list**

- Add a new group called "**Logos**" to the "**Bitmap**" node using the <**Insert**> key.
- Insert the bitmap to the new bitmap group. To do this, we will click in the center of the screen on the editor field, which is still empty.
- Press the <**Insert**> key. A new bitmap element is added to the center of the screen, which must still be linked.

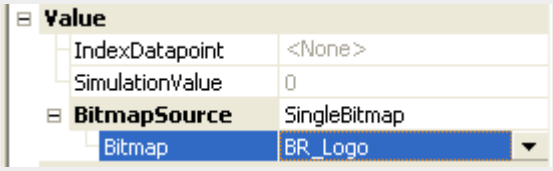



- A bitmap has not yet been assigned in the "**BitmapReference**" property. This is indicated with <**None**>.
- The "**Add bitmap**" dialog box is opened using the <**Browse**> button .
- We will then search for our logo on the hard disk and insert our bitmap list by clicking on the <**Browse**> button.



**Exercise: Displaying the logo in the header**

- Open the "**header**" layer by double clicking. A blank page is now shown in the editor window.
- Place a bitmap control on the upper left part of the page. The size does not yet matter.
- The bitmap is linked in the "**Value**" property. This property has another layer, "**Bitmap Source**". This is where the user can choose whether a single bitmap or multiple bitmaps, which can be switched dynamically by the control program using a data point, should be linked.
- In the "**Bitmap Source**" property, we will then select our logo from the ComboBox under the "**Bitmap**" property.




The bitmap is now displayed on the page with the corresponding size. All controls paced on the page are displayed in the "**header**" layer.


We will use the "**Name**" property to name the bitmap "**Logo**" so that we can identify the logo later on.

### 6.4.2 Displaying the date and time

The date and time should be separated in the header, (i.e. displayed on two lines).

**Exercise: DateTime Control**

The following steps are necessary to display the date and time:

- Place the "**DateTime**" control two times on the upper right part of the page.
- Change the appearance (i.e. the "**Appearance**" property) "**Border**" to "**Bump**" for both controls.
- Change the "**BackColor**" property to an orange color for both controls.
- Change the "**Format**" property; "**Alignment**" for "**Horizontal**" and "**Vertical**" to "**Center**" (i.e. centered text display).
- Change the "**Name**" property of both controls to "**Date**" and "**Time**".

**Question**

Why do we see different shades of yellow background in the properties?

**Answer**

Properties taken from the style sheet are shown in light yellow, while changed properties are shown in dark yellow.

Another possibility would be to change the control style sheet. This would mean you would only have to make the change in one location.

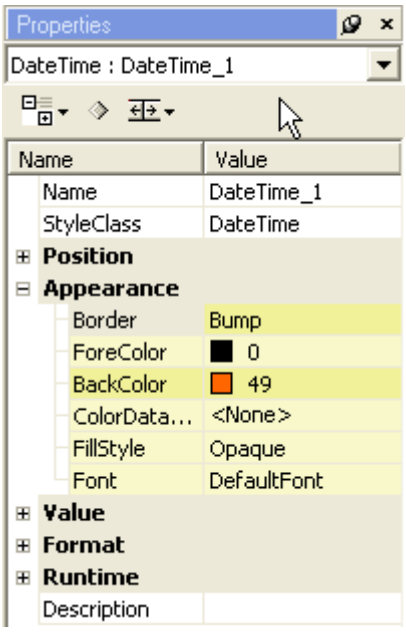



Fig. 25: Appearance – DateTime

We have now adjusted how the controls will be displayed. The next thing to do is to address the value (i.e. the "**Value**" property), with the **date** for the first control and the **time** for the second control.

**Assigning the date and time format**

When assigning the date and time, variables from the control task (i.e. reading the time and copying it to a structure) are not displayed as was just assumed.

This would result in display limitations during runtime because e.g. the language-dependent display (DD.MM.YY – MM/DD/YY) would not be possible.

There are now two ways to display the date and time:

- Addressing a control variable as data point for the "**Value**" property – seconds since 1.1.1970. The internal clock time is used if a data point is not linked to this property.
- Addressing a data/time format which can be freely defined by the user to the "**Format**" property – "**Source = Single Text**" – "**Text**".

We will just be using the second option, which can be set for any language.

**Question**

What does the "(**embedded**)" value from the "**TextGroup**" property mean?

**Answer**

Either a **text group** (management of multiple text entries in one group) or "**embedded**" text, (i.e. direct text input) can be entered in this property.

In our example, we use "**embedded**" text. This means that we specify the time and date format for the language "**English**" right in the text property.

**Time format as Hour:Minute:Second**

**%H:%M:%S** - Time format definition in capital letters

**Date format as Day.Month.Year**

**%d.%m.%y** - Date format definition in lower case letters

### 6.4.3 Displaying static text

The page description (page name) should be displayed in a text field.

**Possibilities for displaying text:**

- Embedded text (i.e. define the text directly during text control)
- Define text in a text group

We will also use the second option here since this has a few advantages during development:

- Central management of static and dynamic text
- The text can be used again in any location.

**Exercise: Creating a text group and inserting text**

The text for the page description is managed in a text group. Since we are starting with just one page, the text group will start off with just the text "**Main menu**".

A new text group can be created by clicking the <**Insert**> button in the text group node.

The editor dialog box now appears in the editor window. The individual lines of text are inserted here for each language used in the project.

Now rename the text group from "**TextGroup_1**" to "**PageDescription**" so that we can locate it quickly later on.

As a final step, we will add a new text in the Editor window by clicking the <**Insert**> key. The text "**MainMenu**" can be entered either right in the Editor window of the corresponding language or under the "**Text**" property.

**Note:**

We recommend making changes in the property list because that is where all of the properties are located. Text groups are however an exception because the Editor window must be used.

### Exercise: Drawing a text box and assigning text

Change the view back to our common layer "**header**". In the center of the header, enter a Text Control by dragging a border.

A text from a text group is now assigned in the Text property of the Text Control using "**Embedded**" text. First, select the text group for the "**PageDescription**" as shown in the image to the right.

The text with index "0" is shown as default text. The desired text can be selected from the ComboBox if the text group contains more than one text.

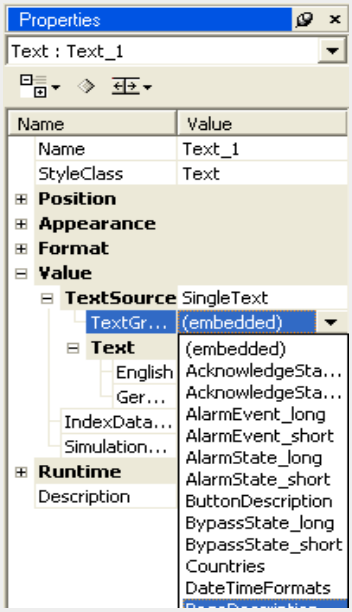



Fig. 26: Text goup in text control

### Task: Changing the text control properties

Each participant can change the appearance of the text as desired to learn how to work with the properties.

**Approach**

For example, the changes could appear as follows:


Fig. 28 Example – Common layer header

What should appear in the footer?

- Buttons for switching the page

The user can switch between pages using **buttons** in the footer. When we are finished, our training example will consist of three pages. Therefore, we will need 2 new text entries (preferably managed in a text group).

In the first step, we will just be drawing and designing the buttons. We will define the executable action, (page switching), later on.

### 6.4.4 Buttons and hotspots

There are now two possibilities for implementing a button:

- Visible button = Button
- Invisible button = Hotspot

**Difference between button ⇔ hotspot**

The functions (i.e. the executable actions), are identical in both types of buttons. The difference is that a visible button also has a visual component such as borders, color, and the option of displaying text or pictures.

A hotspot is used, when an action is to be carried out on an image area with static or dynamic objects.

**Exercise: Drawing buttons in the footer**

We will now enter three identically sized buttons next to each other in the common layer "**footer**". You can do this by copying and pasting or by using the "**Alignment icon**" in the toolbar.

Fig. 29 Toolbar – Arranging controls

**Task: Aligning the buttons**

After individually inserting the three button controls, we will attempt to make each button the same size by using the "**Alignment**" tool. Additionally, each button should be given a meaningful name because Button_1, Button_2, Button_3 are not too explanatory.

The color can also be selected by changing the properties of the button.

**Approach**

The design can be globally changed for a button by inserting a new style.

The footer should now look something like this:

Displaying text in the button

Now we will create a text group with the button descriptions and enter the following text:

- Main Menu
- Text and Value
- Graphics

**Note:**

The same text from the page description can also be used. However, a button description is usually shorter than the image description in the header.
We will assign these texts to each button in the order of the "**Text property**".

## Exercise: Assigning texts to the button control

Create a text group called "**ButtonDescription**" as well as the texts for the button description. These texts will be assigned to the individual buttons.

Assigning text to a button control is a little more complicated than a text control.

A button does not have a static appearance. A button detects a "**released**" and "**pressed**" state. Therefore, the appearance must be configured for each state.

"**TextSource**" from the "**Pressed**" property is set to <**SameAs**> if the text is the same for the pressed and released state.

The image to the right illustrates how the button must be configured for our requirements.

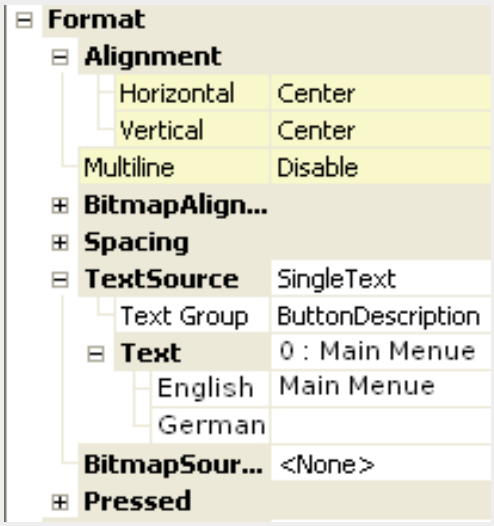



Fig. 31: Button properties Assign text

The footer should now look something like this:


Fig. 31 Footer – Button for page change

## 6.5 Page distribution

In this section, we will learn how to create the pages necessary for the example and to integrate the header and footer. We will then take a look at the result on the target system.

**Exercise: Adding global layers to the start page**

First we must select our start page.


This page, now called "**Init_Page**" also has a layer with the name "**Default**".

In our second example, we added a numeric input and output field to this layer.

A reference must now be added to these layers (the "**header**" and "**footer**") in the "**Context menu**" of the "**Init_Page**" in order to attach a global (i.e. common) layer to the start page.

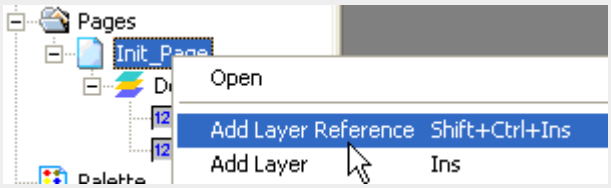
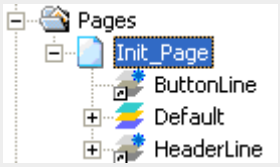



Fig. 33 Adding a layer reference

The common layers are displayed under the "**Default**" layer of the "**Init_Page**".



In the Editor window, our page should now appear as follows:

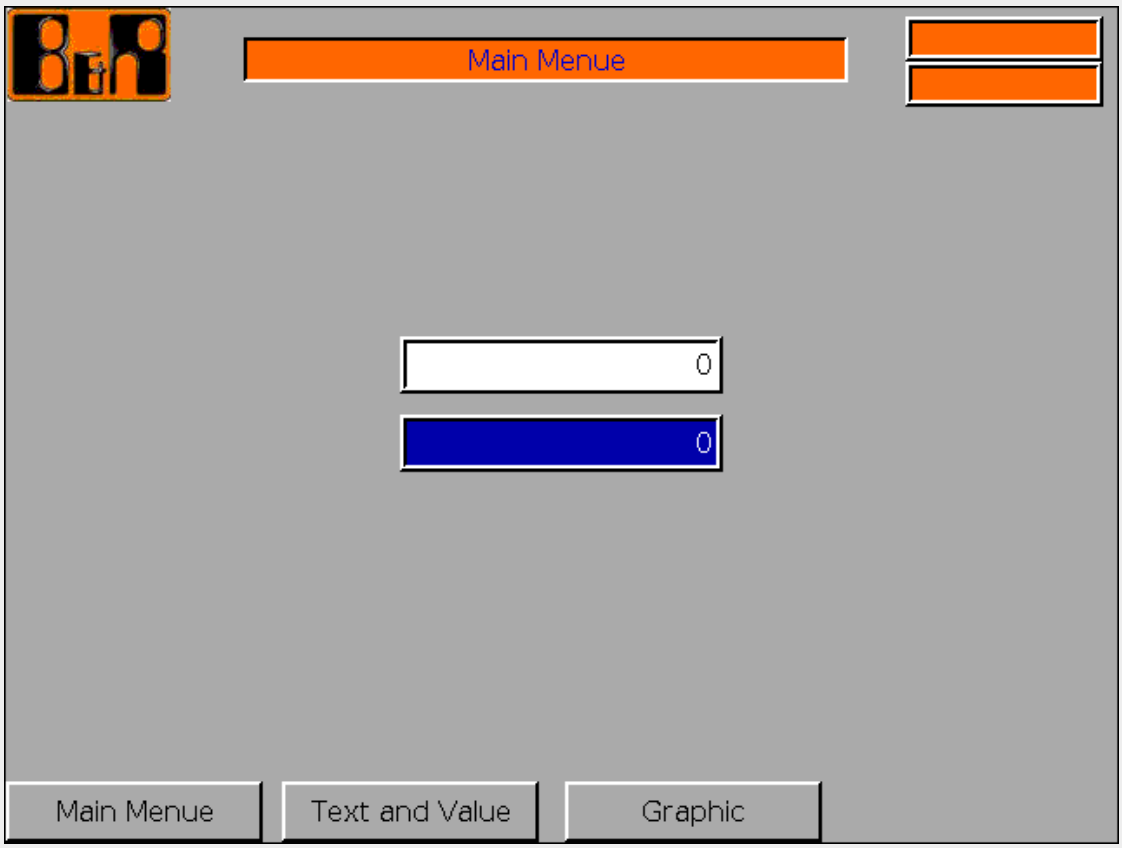



Fig. 34 Example program – Main menu

**Task: Compiling and transferring the changes to the target system**

Our start page will now be output to the target system. The buttons do not have a function yet because we have not created additional pages or instructed the button to perform any functions.

The time is displayed in both DateTime controls.

**Question**

Why don't the assigned foreground and background colors appear in the input field on the target system?

**Answer**

An input field also has a focus color. Since there is just one input field on the start page, it immediately has the focus and therefore displays the focus color. The focus color is set in the page properties and can also be turned off.

### 6.5.1 Page properties

Each page has special properties. The most important properties are those which affect the appearance of the page during runtime such as the background color and the global input behavior (e.g. the focus color on this page).

These properties are described thoroughly in the VC Help.

### 6.5.2 Creating new pages

Our next task is to create two additional pages, for "**Text and Value**" and "**Graphics**".

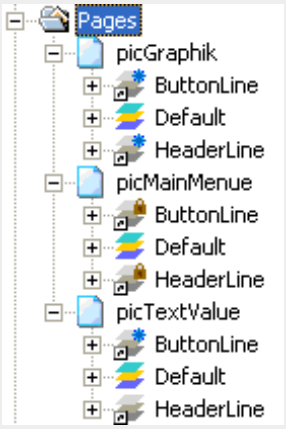
Starting from the "**Pages**" node, a new page can be added by clicking the <**Insert**> button.

**Task: Creating a new page and inserting global layers**

Create the page with corresponding name and insert the "**header**" and "**footer**" for each page

**Result**

The following pages should now appear in the project tree under "**Pages**".




As illustrated in this project tree, the name "**Init_Page**" was renamed to "**picMainMenu**".

### 6.5.3 Managing the layers on a page

A page consists of one "**Default**" layer or multiple layers. This depends on what it is being used for.

These layers are also individually managed in the VC Editor, (i.e. you can prevent objects on another layer from being shifted or deleted during editing).

Furthermore, each layer can be locked for editing, hidden or displayed as border using the toolbar.


Fig. 35 Toolbar – Layers

All layers available on the page are displayed when the ComboBox is opened.

The following actions are possible:

- Hide all objects on a layer using the light bulb icon
- Lock, release or freeze all objects on a layer

**Task: Changing the visibility / release of layers**

Now let's take some time to learn about the effects of the individual relationships by switching the visibility and release mechanisms on any page.

## 6.6 Virtual keys and key actions

Until now, several pages have been created and buttons added for switching between the pages. In this section, the function of "**virtual keys**" and a "**key action**" are described.

### 6.6.1 What is a "virtual key"?

**"Configuration of key actions, independent of hardware"**

A "virtual key" has no relation to a physical key or a button. It describes the behavior of a key using a key action. The location where this action takes place does not matter.

It can also be compared to a global layer. Image objects are also placed on this layer, but do not have a function by themselves. These layers are "brought to life" by linking them to a page.

This now also applies to virtual key. The assigned action is not executed until linked to a key or button.

All "**virtual keys**" apply to the entire visualization project.

This makes it possible to configure the key action just once and to assign it to multiple locations on a key (display with keys) or a button (display with touch screen).

**Exercise: Creating a "virtual key" for changing pages**

Double-clicking on the "keys" node opens the "**Virtual keys overview**" in the Editor window.

**Virtual keys** will be referred to as **VK** from now on.

All VKs already provided in the default project are displayed in the "Virtual key overview". These VKs are used for the TOUCH input dialog box provided.

A new VK can be added using the key combination **<SHIFT> + <CTRL> + <Insert>** or in the **context menu** of the overview.

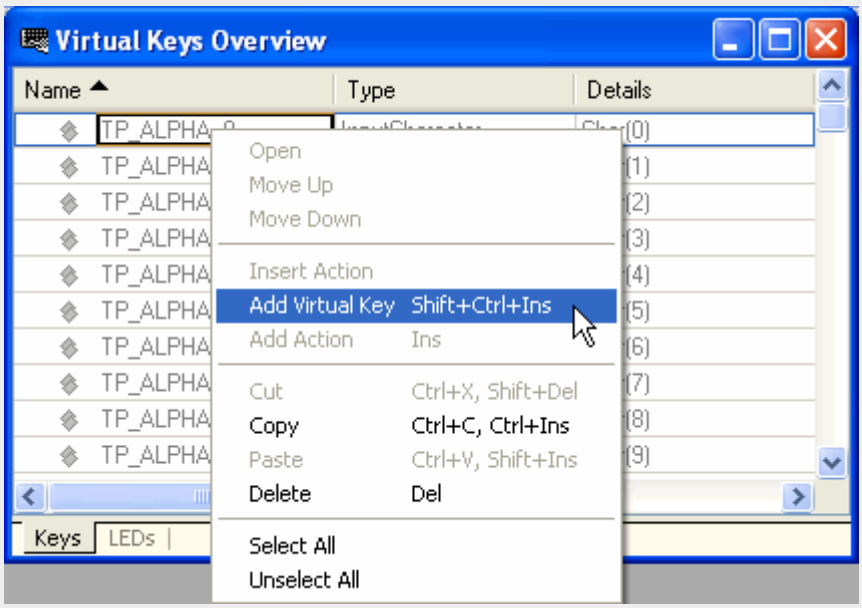



Fig. 37 Virtual keys overview

Now we will enter a new VK.

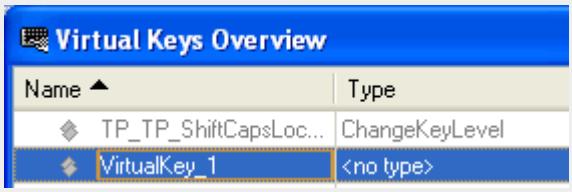



Fig. 37 New virtual key

This VK is configured in the **properties**.

This VK is given a default **name** and **action**.

We must define a page change in the action. More properties are shown depending on the assigned **action**.

In our example, we must specify the target of the page change.

**Task: Creating a VK for changing pages**

Three VKs with the action "**ChangePage**" should be created for the pages in the example project.

**Result**

| | | |
|---|---|---|
| VK_picGraphic | ChangePage | Target(Page, picGraphik) |
| VK_picMainMenue | ChangePage | Target(Page, picMainMenue) |
| VK_picTextValue | ChangePage | Target(Page, picTextValue) |

**Task: Key actions**

Each participant should take a minute to become familiar with the available key actions. Also take a few moments to look over the properties resulting from the action..

**Note:**

A detailed description of all key actions and their properties can be found in the Visual Components Help.

### 6.6.2 Assigning a VK to a button

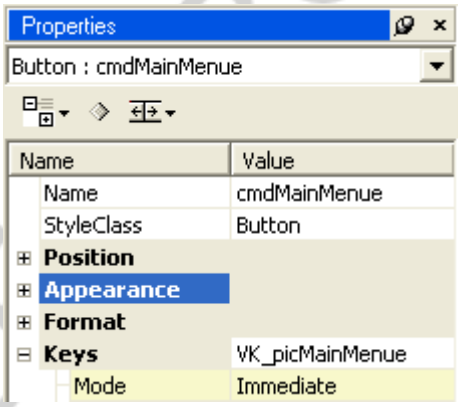



Fig. 38: Connecting a "virtual key" to a button

We must now assign the new VKs to the individual buttons in the "**footer**" layer so that the "**ChangePage**" action can be performed. This is done by assigning the VK to the "**Keys**" property.

### 6.6.3 Creating virtual keys directly on the control

It is not necessary to create a VK on the "**Key**" node. It can be created directly in the "**Keys**" property of a button, hotspot or a key.

A new local VK is created by clicking on the <**Browse**> button in the "**Keys**" property.

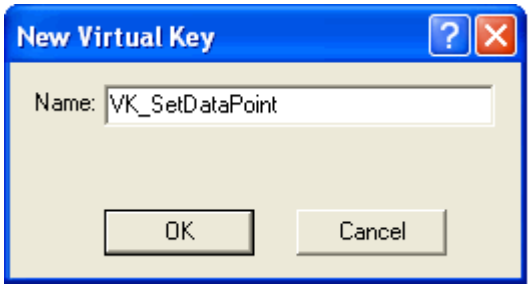



Fig. 39 New virtual key

In this case, the "**Action**" property must also be directly defined on the control.

When using this method, the VK is added to the "**Keys**" node without "**Type**". This VK can be used on another page, but the action must be reassigned on the control (local VK).

The key assignment of the global and local keys can be viewed in the image editor under the "**Keys**" tab.

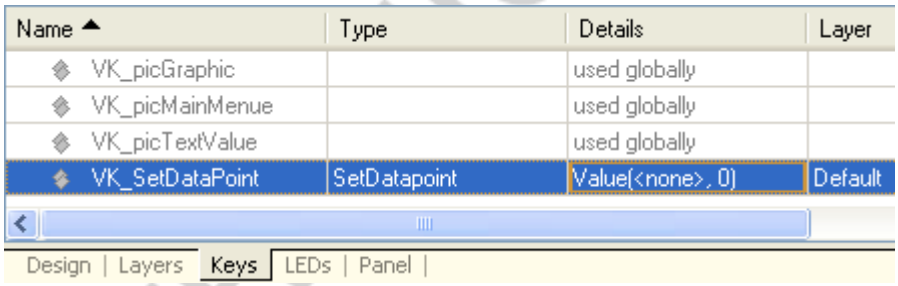



Fig. 40 Image editor - Keys

**Task: Assigning the VK to the buttons in the footer**

The three buttons for changing pages are now assigned the corresponding VKs.

The project can then be compiled and transferred to the target system.

**Result**

We can now change pages on the target system.

**Who can identify the first mistake?**

The answer is explained in the next section.

## 6.7 Static and dynamic text

In this section, we will have a look at the types of text and how to design texts.

**The types of text available:**

- Static text
  Text which does not change during runtime.
- Dynamic text
  Text which can change as a result of a process (change of a control variable).
- Listbox
  Text selected by the operator from a provided list.
- String
  Text provided by the control program (see 6.8 Numeric and alphanumeric display).

### 6.7.1 Static texts

We have already created a static text in the **header** as "**Page description**". Therefore, the user should not have any problems using additional static texts.

A dynamic text must now be used because this static text is displayed on each of the three pages.

### 6.7.2 Dynamic text

A dynamic text changes the display (the text) depending on a value.

Which steps are required now to change a static text to a dynamic text?

- More than one text in the "**Page description**" text group
- We need a control variable which represents the page number.
- A "**Multiple text**" is configured for the "**TextSource**" property instead of "**Single text**".
- The variable for the page number can be linked using the "**IndexDataPoint**" property.

**Task: Creating a control variable "PicNumber"**

A new control variable, "**PicNumber**" with the data type UINT will be created in the "**panel**" task and added to the VC Editor DataSource.

**Task: Global project setting – Linking current page numbers**

After transferring the variable to the VC Editor, it can then be linked to the global project property "**Runtime**" – "**Page**" – "**CurrentDatapoint**".

This variable is written with the page number during runtime.

**Question**

Where is the "**Page number**" defined in the VC Editor?

**Answer**

Each page has a properties **index**. This index is used to identify the page during runtime (e.g. page change via variable)

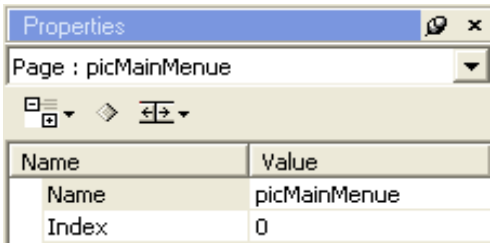
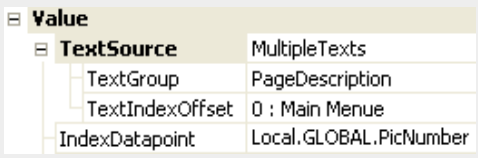



Fig. 41: Page properties - Index

**Task: Linking a variable to the text control in the "header" layer**

This task should be performed using the image shown here

We are still missing the text for the "**PageDescription**" text group because earlier we only defined text for the "**MainMenu**".



**Note:**

The page index is allocated after the pages are created. As a result, the index should also match the page description.

**Task: Creating the text for the page description**

We will now define the following text in the "**Page description**" text group:

| Index | English |
|---|---|
| 0 | Main Menue |
| 1 | Display Text and Value |
| 2 | Display Graphic Functions |

Fig. 42 Text group – Page description

The project is then transferred to the target system.

**Result**

The correct page description is shown on each page.

### 6.7.3 ListBox

Several texts are displayed in a ListBox. These texts can be provided in a "**Text group**" or from a "**Field of String variables**" from the controller task.

The ListBox is also used to select a text from the ListBox, to control the visibility of the individual texts via an option field and to execute entries in the ListBox with a user-defined input behavior.

**Task: Creating a text group with maximum 10 keys**

Create a new text group with a maximum of 10 texts (e.g. country names). The texts in this text group will be displayed in a ListBox in the next exercise.

### Result

| Index | English | German |
|---|---|---|
| 0 | Germany | |
| 1 | Austria | |
| 2 | United Kingdom | |
| 3 | France | |
| 4 | Spain | |
| 5 | Italy | |
| 6 | Portugal | |
| 7 | Sweden | |
| 8 | Netherlands | |
| 9 | Switzerland | |

Fig. 43 Text group – Countries

### Task: Creating a control variable for the ListBox

A new control variable "**ListboxIndex**" (UINT data type) is required in the "panel" user task for evaluating the ListBox index.

This variable must then be added to the Visual Components data sources.

We will now insert a ListBox on the "**Text and value**" page. The text groups "**Countries**" and the index data point "**ListboxIndex**" will be linked to this ListBox.

| Value | |
|---|---|
| Source | MultipleTexts |
| TextGroup | Countries |
| TextIndexOffset | 0 : Germany |
| IndexDatapoint | Local.GLOBAL.ListboxIndex |

**Task: Transfer the project and test the ListBox**

**Result**

The input touch pad "**NavigationPad_ver**" is opened if the ListBox is selected during runtime. This pad contains the keys for:

- Cursor up
- Cursor down
- Accept item at cursor position
- Cancel

**Input and output behavior of the ListBox**

By default, the ListBox is used as an "**Input**" ListBox (i.e. the "Input" property has the value "**True**"). Text is selected using the touch pad.

If the "**Input**" property is set to "**False**", text is only selected using the assigned variable "**ListboxIndex**".

## 6.8 Numeric and alphanumeric display

In this example, we will only be using numeric values for display and input. String or Array variables from the control task can also be displayed and input. These alphanumeric fields are covered in the training module TM650.

We have already created a numeric input and output field in the main menu.

**Task: Displaying the "ListboxIndex" variable in an output field**

Insert a numeric output field on the "**TextValues**" page and link the "**ListboxIndex**" variable.

**Result**

The index of the selected text is displayed in the output field.

### 6.8.1 Focus behavior for numeric input

If multiple input fields are displayed on a page, it is important (especially for entries via keyboard) to highlight the selected input field (Focus).

As we saw on the "**Main menu**" page, the "**Focus color**" is shown instead of the background color of the "**Input set value**" style.

The focus color is set globally for the page properties and applies to the entire page.

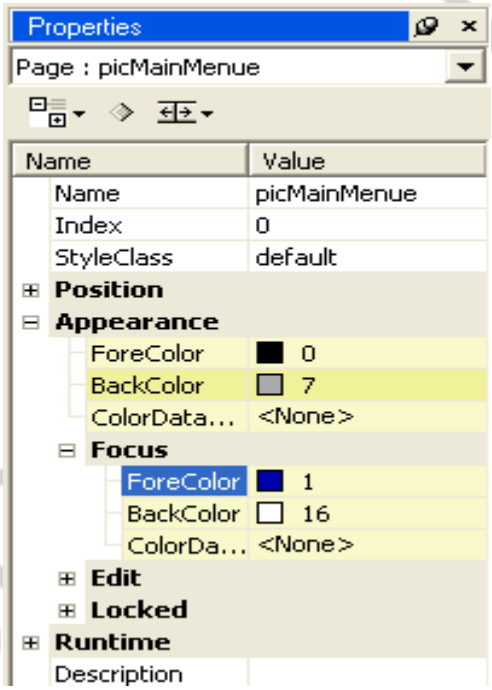



Fig. 44: Properties – Focus color

**Task: Testing the focus behavior with three new input fields**

Create three new variables "**SetTemperature1**" to "**SetTemperature3**" in the "**panel**" task with "**INT**" data type.

Three input fields are created on the "**TextValues**" page and the new variables are linked.

The changes are then transferred to the target system and the behavior is tested.

**Result**

The next input field receives the focus when an entry is completed.

Our page could now look something like this:

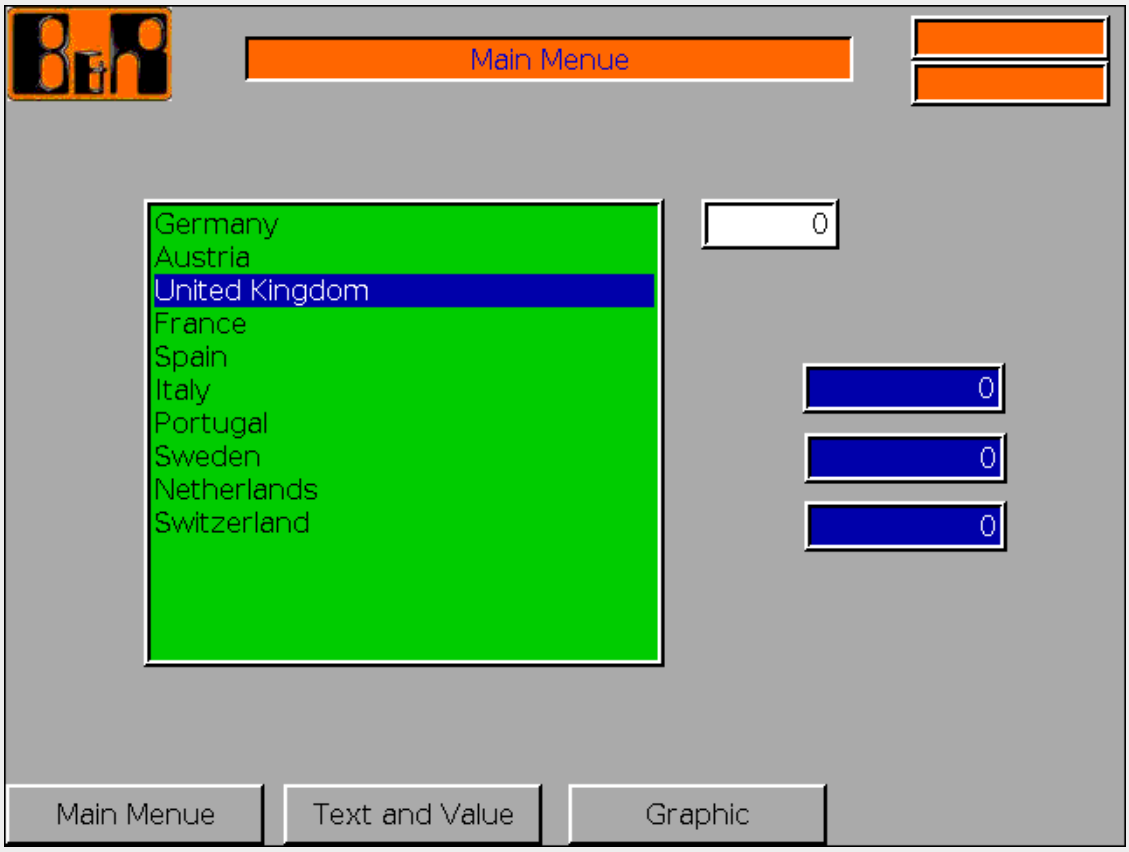



Fig. 45 Screen page – Text and Value

**Question**

What if the user does not want a focus when using a touch panel?

**Answer**

Change the focus color to the same foreground and background color as the input field and the style of the input field (as long as all input fields on the page have the same style).

If this is not the case, set the "**Foreground color**" and "**Background color**" for the focus to <**None**>.

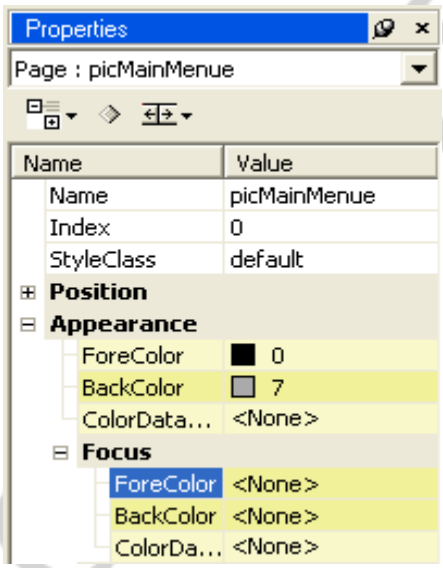



Fig. 46: Global properties – Focus

## 6.9 Bars and other graphic objects

Graphic objects are an important design feature of a visualization. The following graphic objects can be used in Visual Components:

- Static bitmaps
  Logos, background images, machine diagrams
- Dynamic bitmaps
  Display of various bitmaps with the same size at one position. Bitmaps are selected using a variable.
- Lines
- Shapes
  Rectangles, circles with or without filling
- Bars

In this practice example, we already integrated a "**Logo**", (i.e. a "**static bitmap**"). We do not have to learn about lines and rectangles separately. Each participant is able to use these objects on the page "**Graphic**".

### 6.9.1 Displaying bars

A bar can be used for the following functions:

- Progress indicator for a process sequence 0 – 100 %
- Displaying values in the bar diagram
- Displaying value ranges with colors

The most important bar properties:

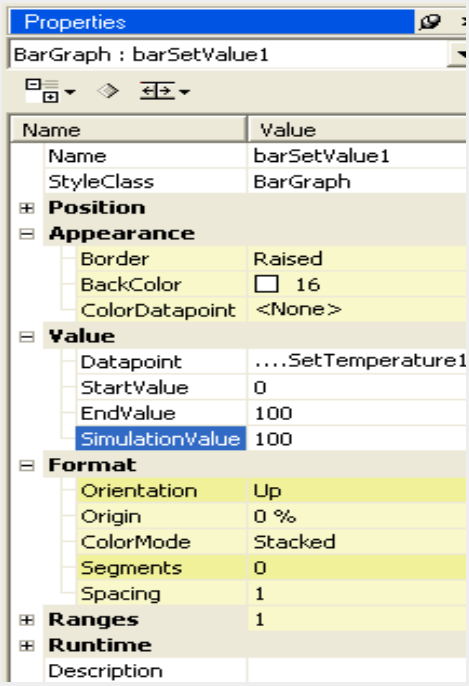
- Variable for values which can be displayed
- Alignment
- Color design
- Value range
- Start value for 0% to 100%

## Exercise: Displaying values as a bar diagram

In the previous task, we created three input fields with three variables. We will now connect these variables with three "**Bar Controls**" on the "**Graphics**" page.

The bar should display a value range between 0 and 100. The 0 point of the bar is at the bottom (i.e. the "**Orientation**" property is set to "**Up**").

The bar should be displayed un-segmented – "**Segments**" property = 0.



### Task: Adding three input fields for changing values

Insert the three input fields from the "**Text and Value**" page on the "**Graphics**" page.

Result

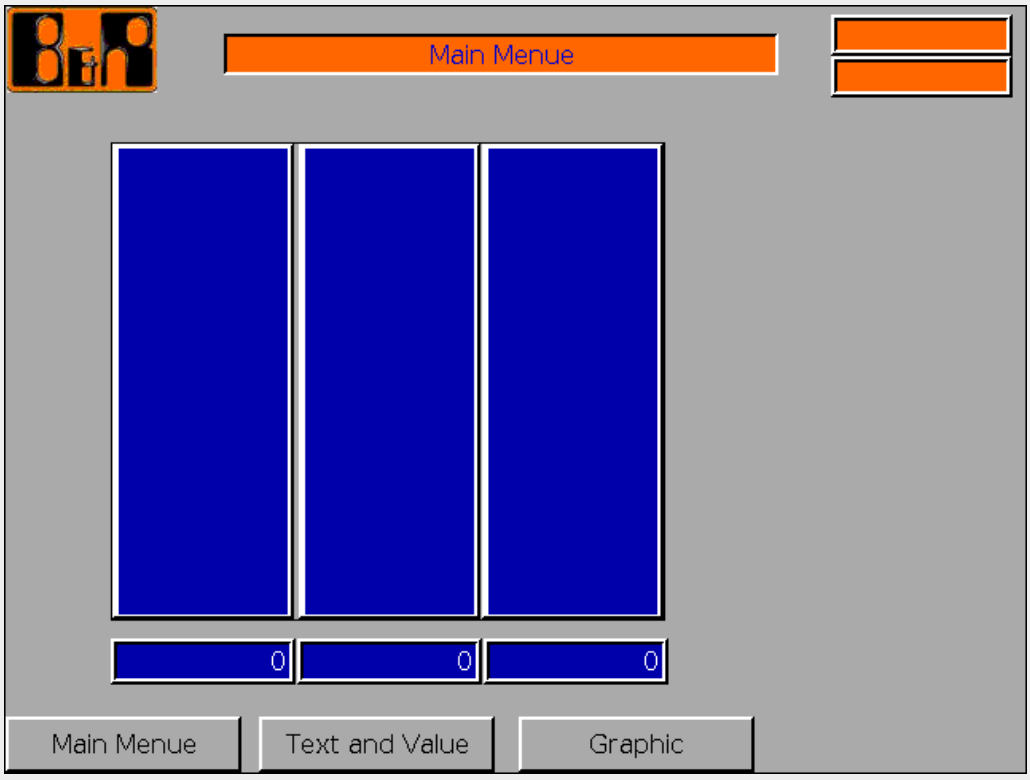



Fig. 48 Screen page - Graphics

### Task: Representing positive and negative values in bars

So far we have represented a value change between 0 and 100. We will now display a value change between -100 and +100, by which the bar is able to extend upwards and downwards.

**Approach**

Change the properties "**StartValue**" and "**Origin**".

The practice example intended for this training module is now finished once all of the exercises and tasks have been completed successfully.

## 7. SUMMARY

After completing this training module, each participant should now be able to create visualization applications using the Visual Components Editor.

We were not able to cover all of the functions and features in the Editor and its components during this Visual Components Basis training module due to the countless possibilities provided by the Visual Components Editor. Using a sample project to explain the most important features and elements of the editor provides enough information to understand how to use Visual Components.

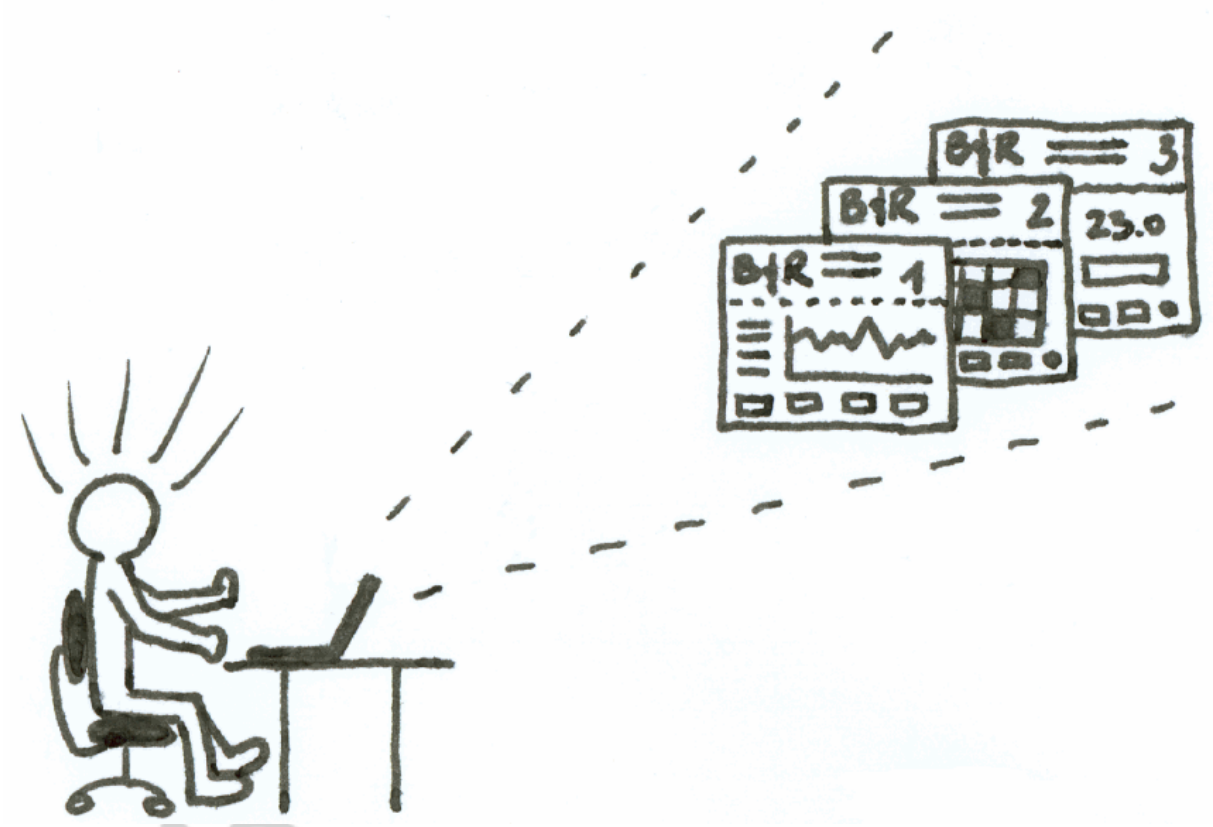



Fig. 49: Visual Components Editor

Since each visualization application has different demands, further information can be found in the Visual Components Help.

In order to gain thorough knowledge of Visual Components it would certainly be necessary to spend some time with the other Visual Components training module TM6xx.

## Overview of training modules

TM200 – B&R Company Presentation **
TM201 – B&R Product Spectrum **
TM210 – The Basics of Automation Studio
TM211 – Automation Studio Online Communication
TM212 – Automation Target **
TM213 – Automation Runtime
TM220 – The Service Technician on the Job
TM223 – Automation Studio Diagnostics
TM230 – Structured Software Generation
TM240 – Ladder Diagram (LAD)
TM241 – Function Block Diagram (FBD)
TM246 – Structured Text (ST)
TM247 – Automation Basic (AB)
TM248 – ANSI C
TM250 – Memory Management and Data Storage
TM260 – Automation Studio Libraries I
TM261 – Closed Loop Control with LOOPCONR

TM400 – The Basics of Motion Control
TM410 – The Basics of ASiM
TM440 – ASiM Basic Functions
TM441 – ASiM Multi-Axis Functions
TM445 – ACOPOS ACP10 Software
TM450 – ACOPOS Control Concept and Adjustment
TM460 – Starting up Motors

TM500 – The Basics of Integrated Safety Technology
TM510 – ASiST SafeDESIGNER

TM600 – The Basics of Visualization
TM610 – The Basics of ASiV
TM630 – Visualization Programming Guide
TM640 – ASiV Alarm System
TM650 – ASiV Internationalization
TM660 – ASiV Remote
TM670 – ASiV Advanced

TM700 – Automation Net PVI
TM710 – PVI Communication
TM711 – PVI DLL Programming
TM712 – PVIServices
TM730 – PVI OPC

TM800 – APROL System Concept
TM810 – APROL Setup, Configuration and Recovery
TM811 – APROL Runtime System
TM812 – APROL Operator Management
TM813 – APROL XML Queries and Audit Trail
TM830 – APROL Project Engineering
TM840 – APROL Parameter Management and Recipes
TM850 – APROL Controller Configuration and INA
TM860 – APROL Library Engineering
TM865 – APROL Library Guide Book
TM870 – APROL Python Programming
TM890 – The Basics of LINUX

**) see Product Catalog

## CORPORATE HEADQUARTERS

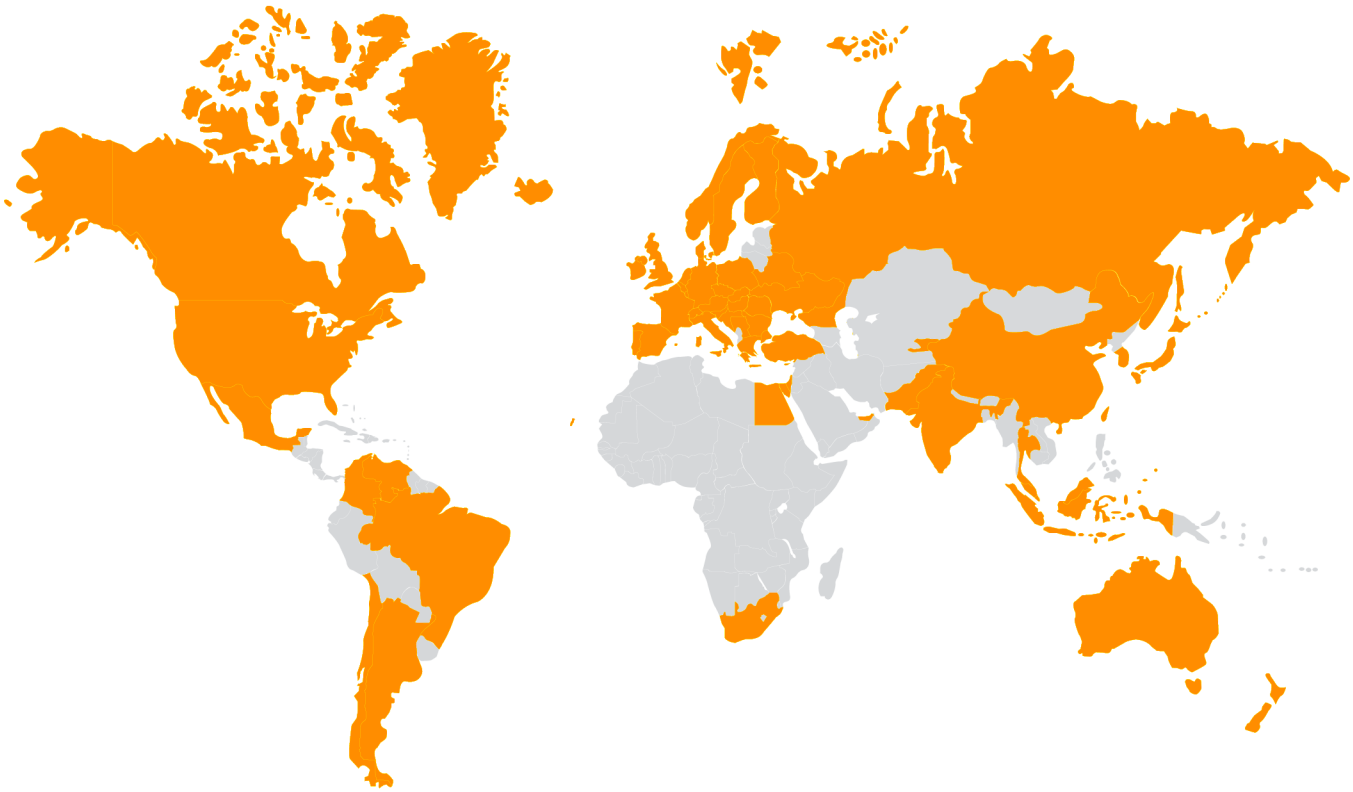
Bernecker + Rainer Industrie-Elektronik Ges.m.b.H.
B&R Strasse 1
5142 Eggelsberg
Austria
Tel.: +43 (0) 77 48 / 65 86 - 0
Fax: +43 (0) 77 48 / 65 86 - 26
info@br-automation.com
www.br-automation.com



TM610TRE.25-ENG 0907



140 offices in more than 55 countries - **www.br-automation.com/contact**



Australia • Argentina • Austria • Belarus • Belgium • Brazil • Bulgaria • Canada • Chile • China • Colombia • Croatia • Cyprus
Czech Republic • Denmark • Egypt • Emirates • Finland • France • Germany • Greece • Hungary • India • Indonesia
Ireland • Israel • Italy • Japan • Korea • Luxemburg • Kyrgyzstan • Malaysia • Mexico • The Netherlands • New Zealand
Norway • Pakistan • Poland • Portugal • Romania • Russia • Serbia • Singapore • Slovakia • Slovenia • South Africa
Spain • Sweden • Switzerland • Taiwan • Thailand • Turkey • Ukraine • United Kingdom • USA • Venezuela • Vietnam