Working with Integrated Visualization

TM610

Perfection in Automation

www.br-automation.com

Prerequisites	
Training modules:	TM210 – Working with Automation Studio TM600 – Working with Integrated Visualization
Software	Automation Studio 4.0
Hardware	[optional] Power Panel with 10.4" touch screen

TABLE OF CONTENTS

1 INTRODUCTION

This training module provides a quick introduction to visualization with Visual Components.

B&R Visual Components provides an environment for creating simple or complex visualization applications.

Visual Components editor

This training module shows how to develop a visualization project in Automation Studio and how to utilize the components of the visualization editor effectively.

1.1 Training module objectives

Selected examples will help you learn how to simply and effectively create a visualization application in Automation Studio's Visual Components environment.

You will learn how to…

- ... Use Visual Components
- ... Create a visualization application
- ... Apply visualization elements
- ... Configure the visualization hardware
- ... Arrange static screen elements
- ... Arrange dynamic screen elements
- ... Implement localization through language and unit switching

2 VISUAL COMPONENTS

Visual Components is a visualization environment integrated in Automation Studio.

This means that the visualization application is managed, edited and executed together with the control project.

Process images can be displayed on a target system with a display, separately from the controller on a remote terminal (e.g. Power Panel) or on a virtual VNC display (**V**irtual **N**etwork **C**omputing).

Integrated visualization

Unlike ordinary visualization – where the application runs separately from the control unit with the controller relying on separate communication to the values displayed – Visual Components is integrated with the controller tasks.

All the examples in this training module are executed using Automation Runtime simulation (ARsim). The visualization is displayed using a VNC viewer. Because the software elements are managed modularly, any SG4 target system[1] can be used.

Visualization \ Visual Components VC4

[1] Any target system that uses a CompactFlash card as application memory can be used.

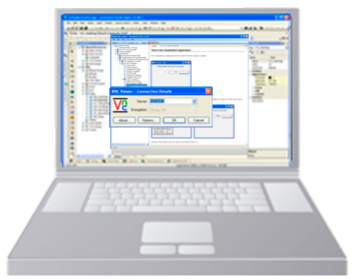
3 FIRST VNC VISUALIZATION

In this section, we will use the Automation Studio help documentation to create a new project with a visualization application, transfer it to the Automation Runtime simulation (ARsim) and then test the program using Automation Studio.

Exercise: Creating your first project using the help system

Create a new Automation Studio project with a VNC visualization application.

Use the Getting Started chapter of the Automation Studio Help system as a guide to complete the example.



Creating your first project

Automation Software \ Getting started \ Creating visualizations with Automation Studio \ First VNC visualization

1) Create a new project.
2) Insert a program.
3) Insert a visualization.
4) Insert a VNC server object and configure it.
5) Open the Visual Components editor and configure the image objects.
6) Compile the project and transfer it to the PC-based Automation Runtime simulation.
7) Open the VNC viewer and connect to the Automation Runtime simulation (ARsim).
8) Test the visualization in the VNC viewer.

With the support of the online help system, you were able to create your first visualization application using Visual Components.

In the next few sections, we will explain the structure of an Automation Studio project using a sample project.

4 THE VISUAL COMPONENTS EDITOR

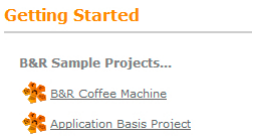
An Automation Studio sample project is used to describe the Visual Components editor and the structure of a visualization project.

4.1 "CoffeeMachine" sample project

An Automation Studio installation includes several sample projects. One of these sample projects is used in this training module to describe the Visual Components environment and the extensive functions it provides.

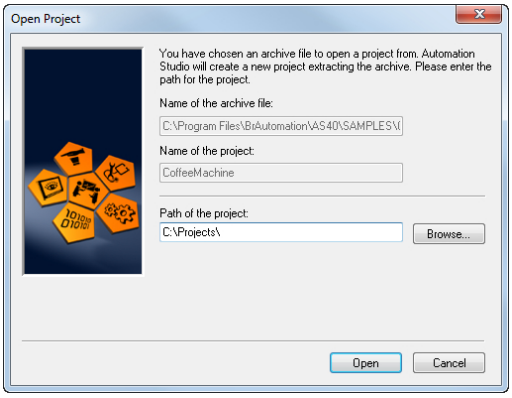
Exercise: Sample Automation Studio project - "CoffeeMachine"

The sample project can be opened from the start page.



Choosing a sample project

1) Open the sample Automation Studio project from the start page.
2) Open the visualization object **"Visu"** from the **Logical View**.

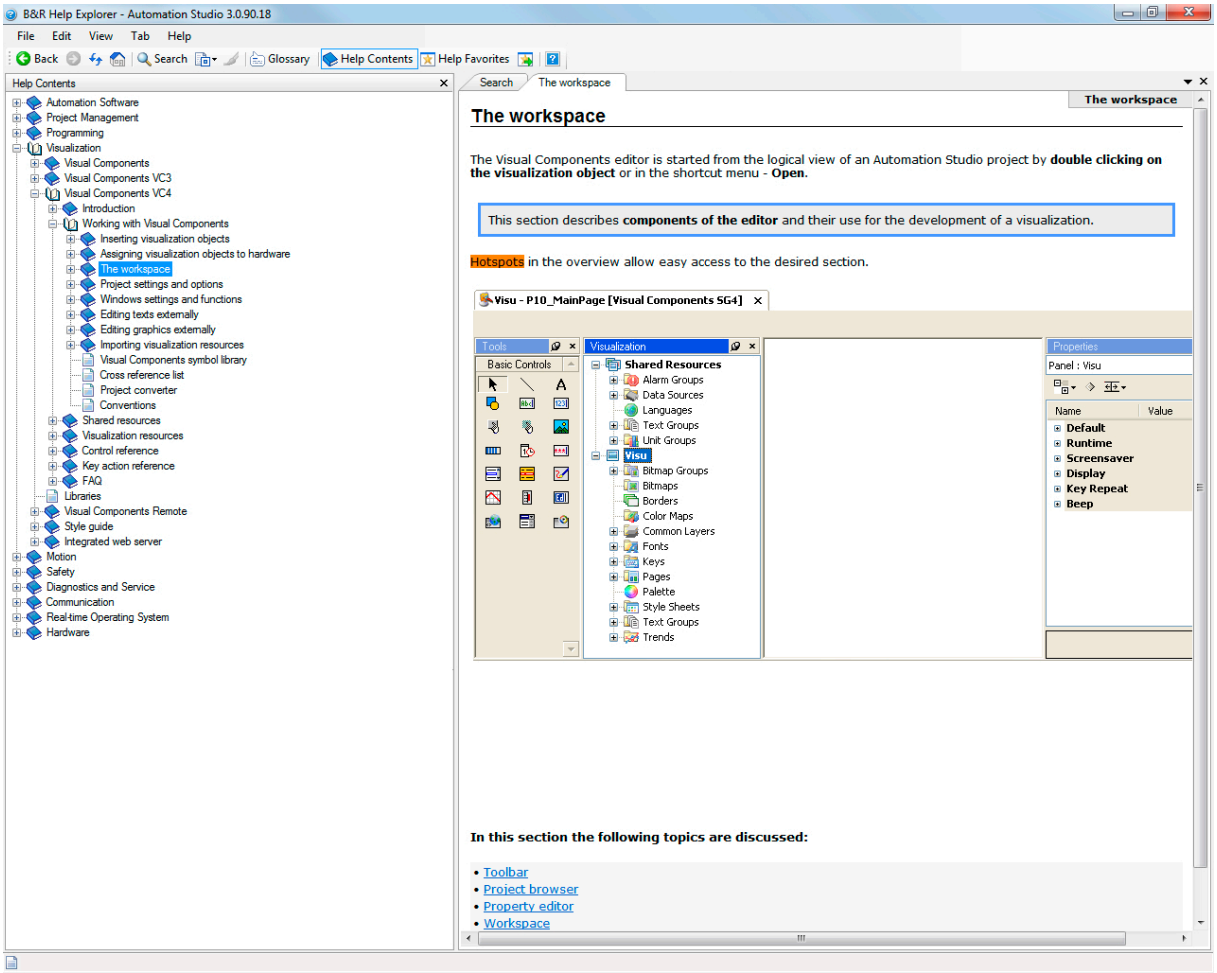


Opening a project with Automation Studio

4.2 Visual Components help system

The Automation Studio help system is your constant companion throughout the development, configuration and commissioning of a project.

It serves as a reference guide for using Automation Studio and its editors, for creating a program or visualization application, for configuring drives, and also provides access to all hardware documentation.



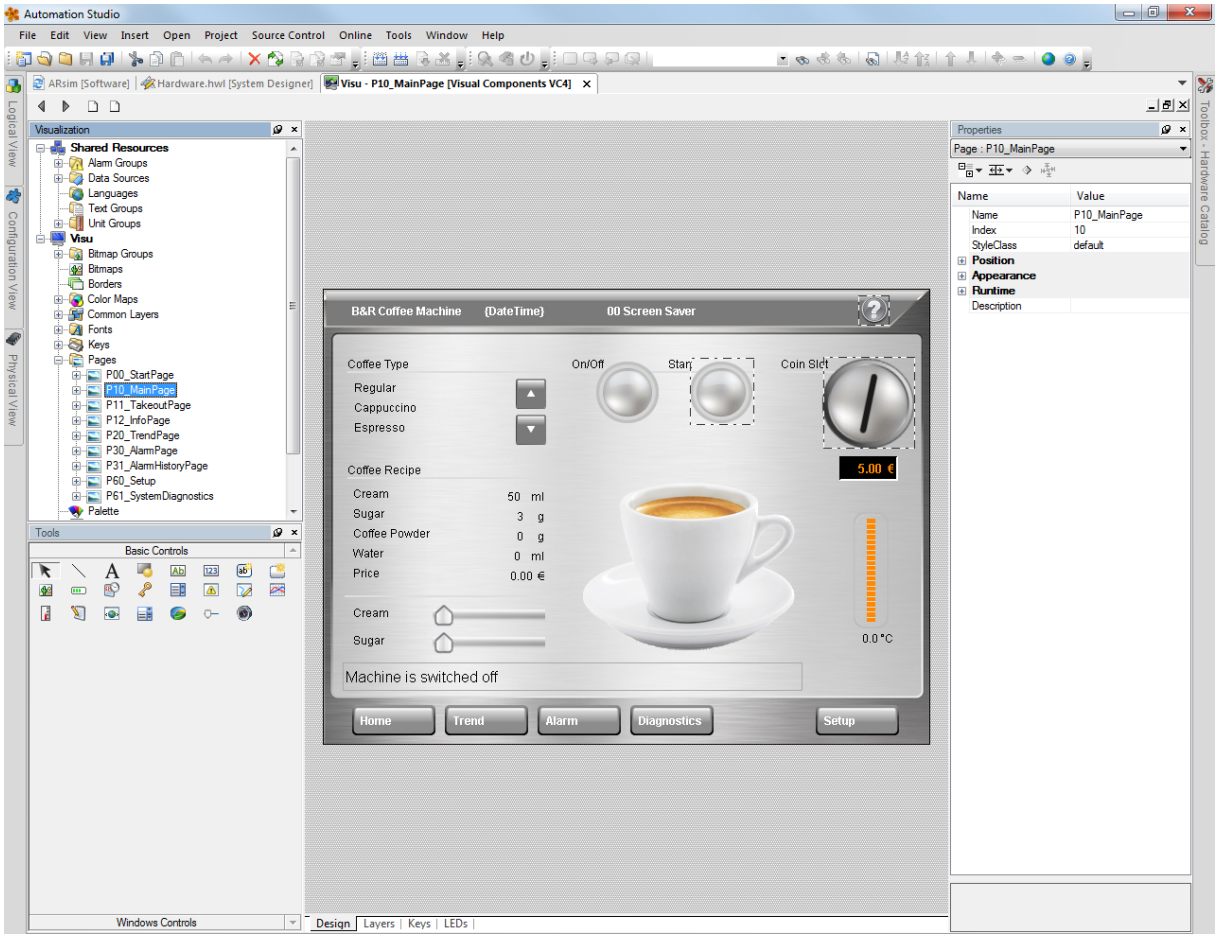
Visual Components help system

The Visual Components training documents frequently refer to the help files because they are available at all times and – most importantly – during project creation.

All help references in this training module refer to the Automation Studio help sections "Visualization \ Visual Components VC4" or "Visualization \ Visual Components Remote".

4.3 The workspace

The Visual Components editor is divided into several areas, each with a different function.



The Visual Components workspace

- The menus and toolbars provide access to the functions of the Visual Components editor.
- The components of a visualization are created, managed and edited in the respective nodes on the left.
- The editor for configuring a component is shown in the workspace in the middle.
- The properties of a component or a selected object are shown on the right.

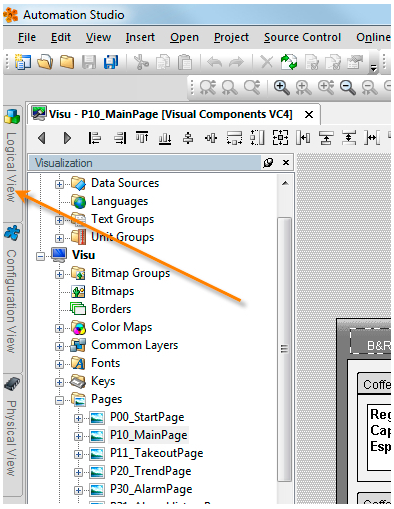
Working with Visual Components \ Workspace

Exercise: Working with Visual Components

The goal of this exercise is to become familiar with using the Visual Components editor.

Open each node of the visualization project explorer, and then double-click on the desired object to open the corresponding editor. The properties of the object are listed on the right side of the screen.

Hiding the project explorer opens up more space on the screen for creating the Visual Components application.



Hiding the project explorer

Working with Visual Components \ The workspace \ Toolbar

Working with Visual Components \ The workspace \ Project browser

Working with Visual Components \ The workspace \ Property editor

Working with Visual Components \ The workspace \ Workspace

Working with Visual Components \ The workspace \ Controls

4.4 The structure of a visualization application

A visualization application consists of multiple pages, or process images, and each of these pages has a specific task for displaying the process sequence.

Interaction between the operator and the machine takes place using a keypad or touch screen.

Process images are displayed as text, values and graphics. In Visual Components, these display elements are called controls.

Controls have various properties that determine their appearance. A control may have static properties, which can't be changed during runtime, or dynamic properties, which can.

Process variables represent the interface between a control and a control program.

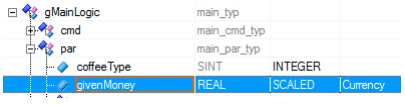
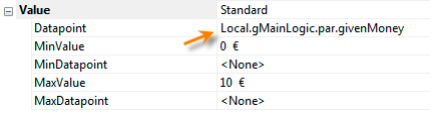
Control	Data Source	Control properties
 Select control	 Organization of variables in the data sources	 Assignment of data source to the control's "Value" property

Table: Relationship between process variables and controls

Data points are process variables that can be configured in Visual Components with additional properties, such as units or limits.

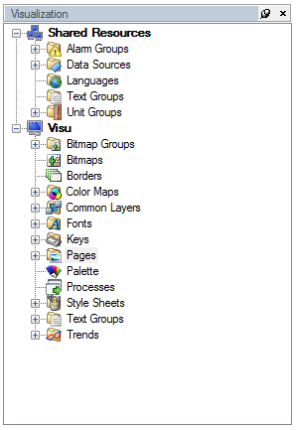
When a control receives **input**, it triggers an action in the visualization application or changes the value of a process variable. How this input takes place is configured using properties.

4.4.1 Global and local visualization elements

In addition to containing local elements, a visualization application is also composed of global elements for all objects in the Automation Studio project.

Global resources, such as data points, languages and alarms, are managed in the project explorer under **"Shared resources"**.

Local resources only apply for one visualization and are managed under the component with the **"name of the visualization object"**.



Resources of the visualization object

4.4.2 Conventions in Visual Components

Just like when programming in Automation Studio, there are some guidelines to follow when creating a visualization application in Visual Components.

Working with Visual Components \ Conventions

4.4.3 The visualization template

When a new object is inserted in Visual Components, it has a number of predefined elements that you can use in your project.

When a new object is inserted in Visual Components, it has a number of predefined elements that you can use in your project.

- The start page (Init_Page)
- Fonts and two languages - English and German
- Unit groups (SI units)
- Predefined style classes for all controls
- Key configurations for touchpads
- Basic alarm system
- Bitmap groups for touchpads and the alarm system
- Borders and buttons

5 DEVELOPING A VISUALIZATION PROJECT

Visual Components sample program

The previous exercises illustrated the structure of the Visual Components editor and its components using an Automation Studio sample program.

In this chapter, you will create a visualization application in Visual Components based on the **"CoffeeMachine"** sample program.

The following project sections will be explained and performed

Exercise: Executing the "CoffeeMachine" sample program

Start the "CoffeeMachine" sample program using Automation Studio.

The goal is to transfer the example program to the Automation Runtime simulation (ARsim) and operate the visualization application in the VNC viewer. This will make it possible to complete the remaining tasks.

Automation Software \ Getting started \ Creating programs with Automation Studio \ CoffeeMachine example

The "Visu" visualization object found in the sample program can be used as a reference at any time. Keep in mind that only one visualization can be opened in Visual Components at a time.

5.1 Exercise

Your task is to recreate the existing program for the "CoffeeMachine" in your own visualization application.

5.1.1 Process flowchart

The coffee machine is turned on first; then the water is heated up. After a certain temperature has been reached, preparation of the selected coffee type can begin.

Payment for the coffee is simulated by entering an amount. If the amount matches the price of the respective type of coffee, preparation can begin.

Once preparation is complete, the coffee can be removed and the change dispensed.

5.1.2 Process variables used

The following process variables are used for communication between the visualization and user applications.

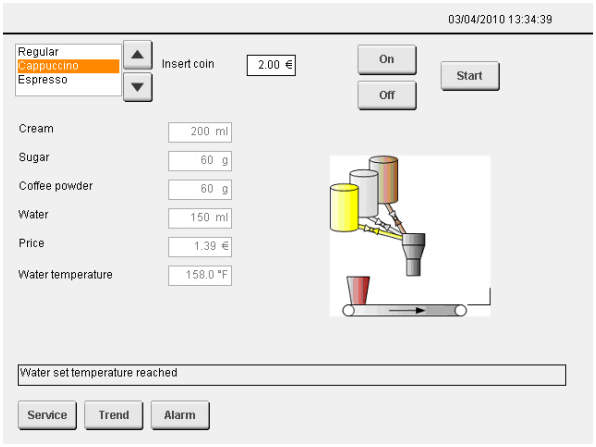
Action	VC Components	Process variable
Select coffee type	Listbox, button	gMainLogic.par.coffeeType
Coffee ingredients	Numeric field	gMainLogic.par.recipe.coffee gMainLogic.par.recipe.milk gMainLogic.par.recipe.sugar gMainLogic.par.recipe.water
Coffee price	Numeric field	gMainLogic.par.recipe.price
Payment	Numeric field	gMainLogic.par.givenMoney
Switching on/off	Button	gMainLogic.cmd.switchOnOff
Starting the preparation phase	Button	diStartCoffee
Water temperature	Numeric field	gHeating.status.actTemp
Status indicators	Text field	gMainLogic.cmd.vis.messageIndex

5.1.3 Image generation

The image should be structured as follows:

Main page

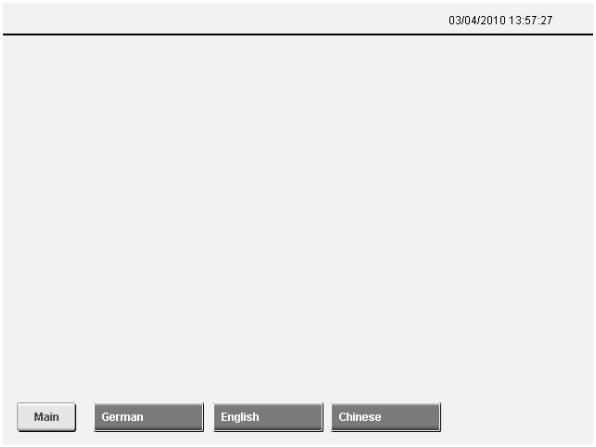
The main page is structured similarly to the template. There are no borders or design elements. The recipe data is changed here by entering the values in numeric fields.



Structure of the main page

Service page

In this training module, the service page is used to switch languages. This page is opened manually from the main page.



Structure of the service page

5.2 Inserting a new visualization object

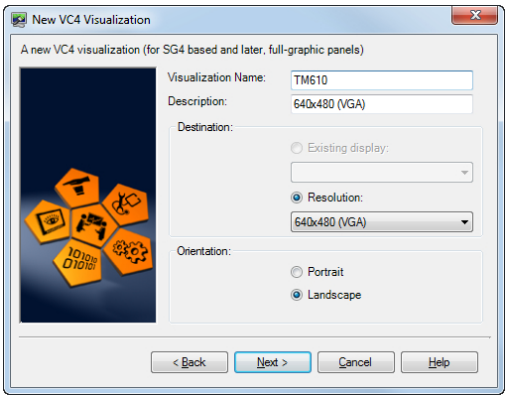
New visualization objects are created and managed in the Logical View.

This process was completed when creating the first project.

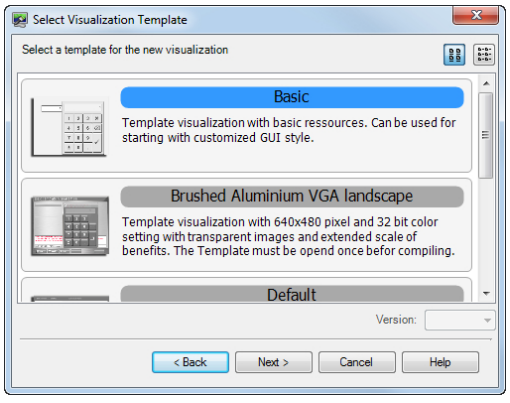
Exercise: Adding a visualization object - "TM610"

Insert a new visualization object in the Logical View and name it "TM610". Select a resolution of 640 x 480 in landscape format.

1) Close the open visualization.
2) Right-click on the "Visualization" package and select <**Add object**> from the shortcut menu to insert a new VC4 visualization with the category "**Visualization**".
3) Select the standard template "Basic"
4) Add this object to the active configuration.

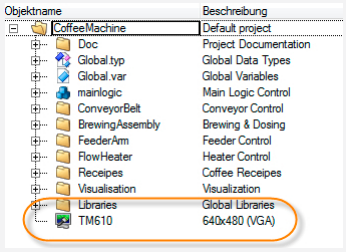


Inserting a new visualization object



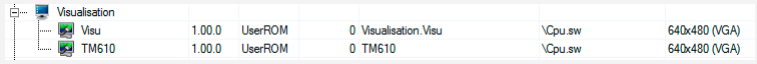
Selecting the standard template "Basic"

A new visualization object has been added in the Logical View.



TM610 in the Logical View

The visualization object is displayed in the corresponding positions in the software configuration.



TM610 in the software configuration

Working with Visual Components \ Inserting visualization objects

The original "CoffeeMachine" visualization application contains not only the default languages English and German, but also Chinese for displaying UNICODE text. This language is added to the new visualization application automatically when it is opened (4.4.1 "Global and local visualization elements").

This is indicated by a message in the output window.

```
Warning: New language "zh(CN)" was added to the project.
```

5.3 Assigning the visualization to the hardware

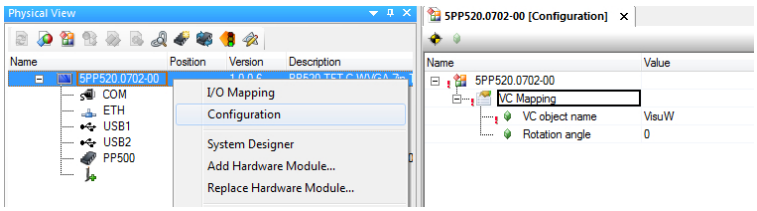
Before the visualization can be configured, it must first be assigned to the corresponding hardware.

A visualization object can be used for the following types of visualization:

- Visualization on a local display
- Visualization using a VNC viewer
- Visualization on a terminal connected via Ethernet

5.3.1 Visualization on the local display

For every target system with a display, visualization objects can be assigned in the Physical View. In the shortcut menu for the "Display" entry, select **Configuration** .



Assigning a visualization object to a display

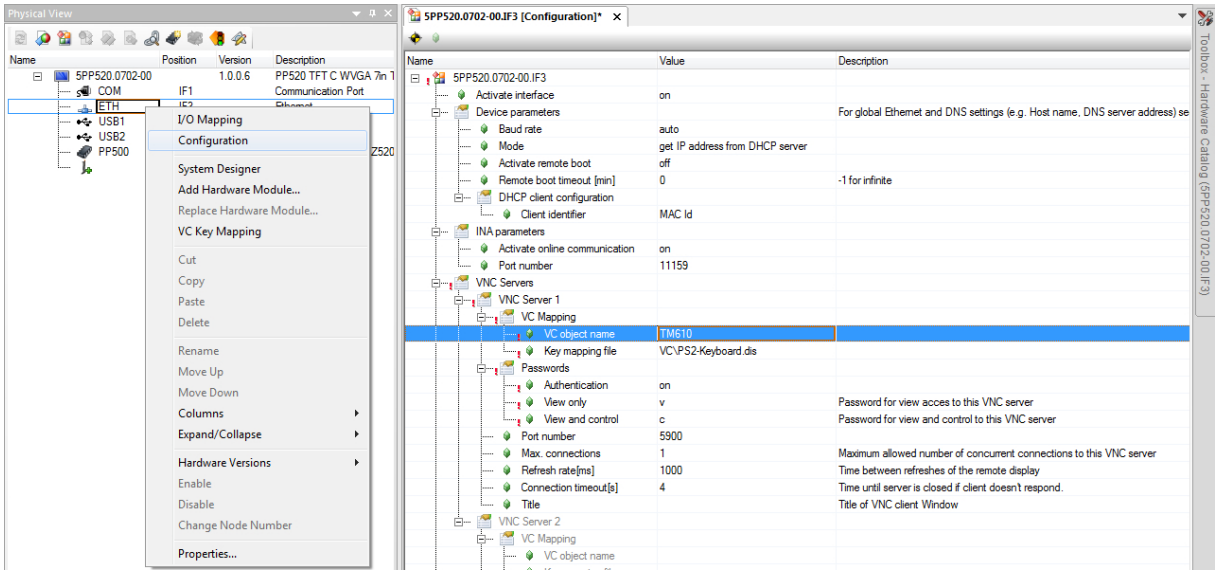
In the VC mapping, any visualization object assigned in the software configuration that fits the resolution of the display can be selected.

Local visualization on a Power Panel PP520

5.3.2 Visualization via VNC

A visualization object can be assigned to the VNC server in the properties of the Ethernet interface. In the shortcut menu for the Ethernet interface, select **Settings** to open the configuration. The visualization object can be assigned under "**VNC Servers**". This allows multiple VNC servers to be configured with different visualization objects.

Below the assignment of a visualization object there are additional settings for the VNC server, such as passwords and connection settings.



Mapping visualization object "TM610" to the VNC server

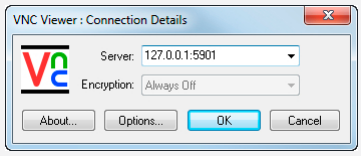
Exercise: Inserting and configuring a VNC viewer

Your task is to insert a new VNC viewer object in the Physical View for the "Simulation" configuration.

Then you will assign the visualization object "TM610" to the new VNC viewer object. A password for read and write access must be defined in the **VNC server configuration**.

1) In the Physical View, open the configuration settings for the Ethernet interface
2) Below the entry "VNC server", assign the visualization object "TM610"
3) Define a password for read and write access
4) Set the port number to 5901
5) Transfer the project to the target system and open the VNC viewer

In the VNC viewer, the IP address and port 5901 need to be entered for the connection. After the password is entered, an empty visualization page is displayed.



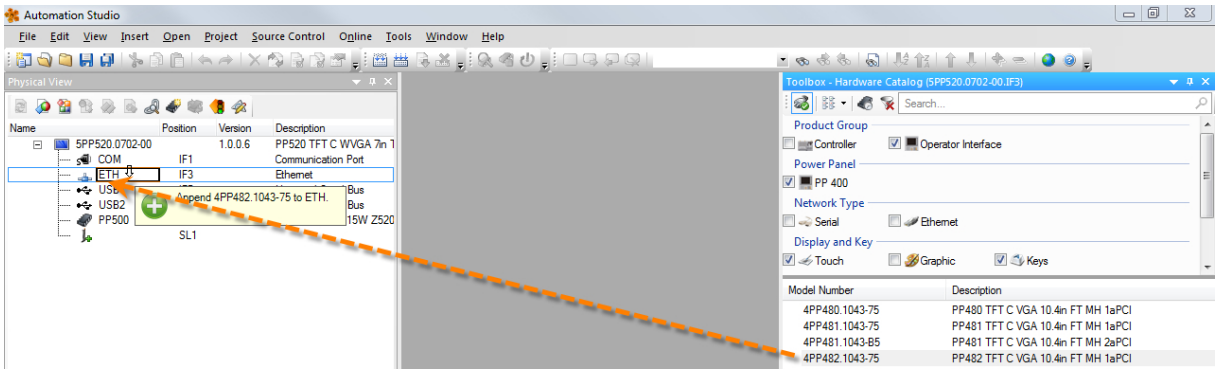
Displaying a visualization in the VNC viewer

Visual Components Remote \ VNC \ VNC project development

5.3.3 Visualization on a terminal

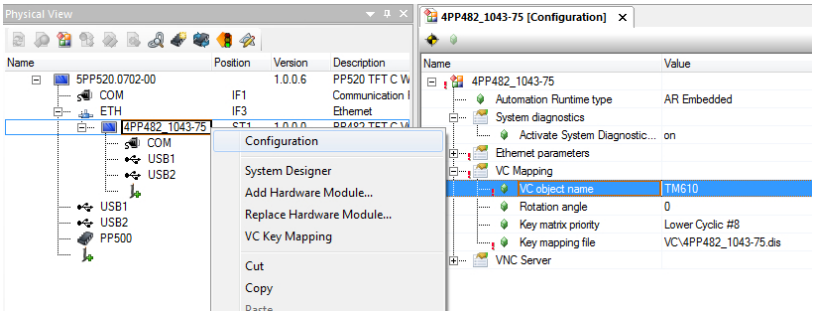
Like a VNC visualization, a terminal visualization is also displayed remotely from the CPU.

A terminal is connected to the CPU's Ethernet interface, i.e the terminal server. The terminal can be dragged and dropped from the Hardware Catalog to the Ethernet interface in the Physical View.



Connect a terminal to the Ethernet interface

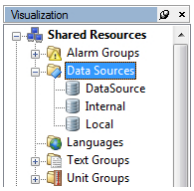
Once the terminal has been added, you can go to its properties and assign the visualization object and configure the network connection settings.



Configuration settings for a terminal

Visual Components Remote \ Terminal Mode \ Terminal server project development

5.4 Managing variables and data points



Data sources

A screen can be "given life" using the variables programmed in Automation Studio.

Variables in the application are managed in Visual Components as **data points** that form the interface between the process sequence and the visualization.

In the Visual Components editor, data points are created and managed under the **"Data sources"** node.

5.4.1 Relationship between process variables and data points

A process variable deals with physical values in the application.

However, a data point in Visual Components has several properties. In addition to the value of the process variable, the data point can also be scaled, limited or displayed with decimal places.

Data point = Variable with additional properties.

5.4.2 Data points

The sample project already contains the data points for the existing application program. Double-clicking on the **Local** data source displays all the variables found in the Automation Studio project as data points.

Name	PLCType	VCType	Unit Group / Subtype	Limit	PLCUnit	UpdateTime	UserID
gBrewing	brewing_typ						
gConveyor	conveyor_typ						
gFeeder	feeder_typ						
gHeating	heating_typ						
gMainLogic	main_typ						
cmd	main_cmd_typ						
par	main_par_typ						
status	main_status_typ						
curLanguage	UINT	INTEGER				<Default>	
curPage	UINT	INTEGER				<Default>	
money	main_status_money_...						
progressStep	USINT	INTEGER				<Default>	
startProgressStep	USINT	INTEGER				<Default>	
mainlogic							
Visualisation							

Variables as data points in Visual Components

If the variables in the application program change or if new variables are added, the view in the data sources editor can be refreshed by pressing <F5> or using the button on the toolbar.

Shared resources \ Data sources

5.4.3 Displaying a value with its units

If a data point is connected to a control, the control takes on all of the properties of that data point.

This means that the properties of a data point (e.g. decimal places and/or unit text) are configured for the data point and not for the control. In the Visual Components editor, unit groups are created and managed under the "Unit groups" node.

Unit change - °F to °C

Data points with units will be described further on.

Shared resources \ Unit groups

5.5 Global properties of a visualization object

A visualization object has global properties that can be used to influence the runtime behavior of the visualization application.

A visualization object has global properties that can be used to influence the runtime behavior of the visualization application.

- The language in which the visualization starts
- The page displayed after startup
- Screensaver configuration
- Data points for switching and monitoring the image number and switching languages in the application program
- Color depth of the visualization
- Key reaction configuration

In the Visual Components editor, these settings are made under the node of the visualization object.

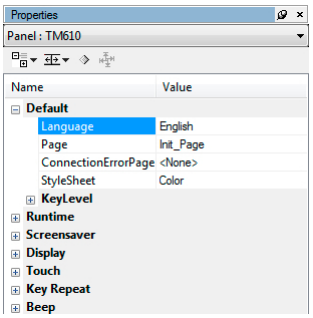
Exercise: Setting the default language

A new project automatically contains two languages (English / German). Create the visualization in your local language.

Set the desired default language in the global properties.

1) Select the node "TM610".

2) Change the language in the default properties.



Select the default language

5.6 The layering method when designing pages

A process image can be made up of multiple layers.

Repeated global image information is created once centrally and can be used by an unlimited number of process images throughout the application.

- Template for common image areas
- The various layers are combined to form the overall image.
- Layers can be locked, hidden and shown individually during runtime.

The layering method when designing pages

Before the individual images are created, it is important to determine what information will be shown in all the images.

In this example, the date and time are displayed in the header at the top right of the screen.

The header is separated from the rest of the image area by a horizontal line.

Sample global page

Exercise: Inserting a global layer to display the time

Your task is to create a new global image layer named **"globalArea"** under the **"Common Layers"** node. This image layer is used to display the time in the header of a page.

The time format is determined by selecting the format as text in a text group named "DateTimeFormat".

Date and time formats can be configured independently of the language using a flexible **format string**.

1) Create a local text group named **"DateTimeFormat"**.
2) Insert text with these formats:

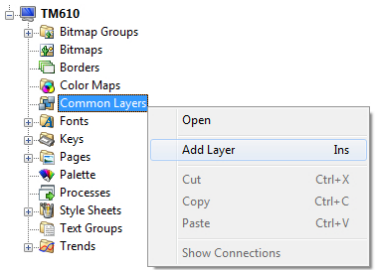
German	English
%d.%m.%Y %H:%M:%S	%m/%d/%Y %H:%M:%S

Table: Date and time format

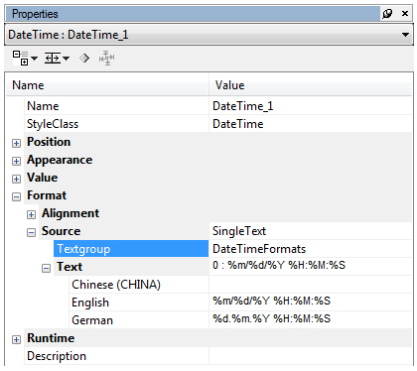
3) Insert a global image layer.
4) Insert a **"DateTime"** control in the top right of the page and connect the text from the text group **"DateTimeFormat"**.
5) Draw a horizontal line with a width of 2 pixels.

Approach:

The following images show the steps in the Visual Components editor. As the training module progresses, there will be fewer and fewer of these images and they will only show new functions.



Insert a global layer



Configuration for the date & time control

It is recommended to give each control in a visualization a name that corresponds to its function. A large project can quickly become confusing if only the default names are used.

The background color of the global layer is not applied to the overall process image.

Regardless of your local language, it is best to use English names for the objects in your project. This makes it easier for the project to be edited by international colleagues. This recommendation is practiced throughout this training manual.

A global layer contains all the information that remains the same throughout multiple process images.

In the next step, you will create a process image that uses this global image layer.

Visualization resources \ Common layers

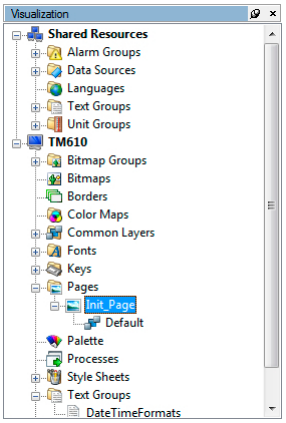
5.7 Process images

Process images consist of one or more image layers.

An image layer contains controls used to display text or values or to enable interaction between the user and the system. When a new visualization project is created, it already contains an image with a local image layer.

This example requires the following pages:

- Machine operation
- Service page
- Display of temperature curves
  (see TM640 Alarms, Trends and Diagnostics)
- Display of alarms and messages
  (see TM640 Alarms, Trends and Diagnostics)



Managing visualization pages

A new process image can be added by right clicking on the "**Pages**" node and selecting <**Add Page**> from the shortcut menu, or using the <**Insert**> button.

A reference to the global layer can be added by right-clicking on the node for the page, selecting **<Add Layer Reference>** from the shortcut menu and then selecting the global layer.

Exercise: Creating the main and service pages

The main page is used to operate the machine. The service page displays the process sequence during preparation.

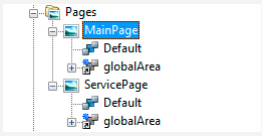
Add the global layer "globalArea" to each of these pages.

Approach:

- Rename the **"Init_Page"** to **"MainPage"**.
- Create a new page and name it **"ServicePage"**.
- Add the global image layer to both of the pages.

The properties of a page define the appearance of the control itself as well as the input fields. For example, if you would like to define a focus color for a touch screen display with active input fields, you can do so in the appearance properties.

There are now two pages under the Pages node, each with a local **"default"** layer and a reference to the global layer.

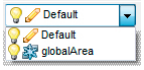


Local levels and level references

In the image editor, both the local and the referenced global layers are shown overlapping.

If an image contains layers that will be shown and hidden dynamically during runtime (e.g. dialog or message boxes), they can be hidden during editing in order to make it easier to work on your project.

This is done using the visibility icon in the image editor toolbar.



Visibility of image layers

Exercise: Changing visibility of image layers in the editor

Open the combo box for each image layer in the status bar of the image editor and change the visibility of the layers using the visibility icon.

Visualization resources \ Pages

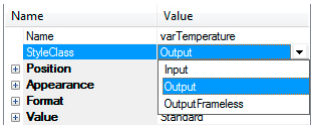
5.8 Styles and style sheets

A **style** defines the appearance of an object by allowing its basic properties to be defined.

A **style sheet** is a collection of styles. Assigning a style sheet for the visualization is an easy way to give it a uniform appearance.

Advantages of a style sheet

- The properties used to design an object are managed centrally (e.g. colors of an input or output field).
- Multiple styles can be created for the same object.
- Properties only need to be changed in one place and are applied to all objects in the project with the same style.



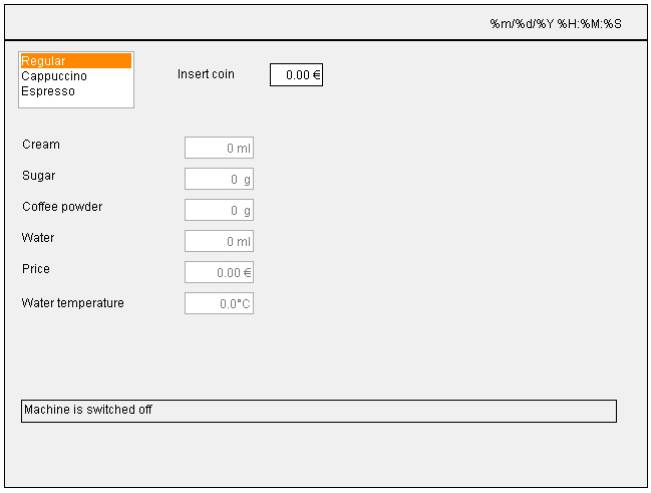
Example of a style for numeric fields

In this training module, different styles will be used to differentiate between input and output controls.

Visualization resources \ Style sheets

5.9 Static and dynamic text

In Visual Components, text can be added directly on the **Text** control where it is to be shown or as a reference to a text group.



Static and dynamic text

Managing static and dynamic text centrally in a text group allows it to be used multiple times throughout the project.

For the sample program, this means that all text will be created and managed in text groups. The text, button and listbox controls will only contain a reference to the corresponding property.

Text will be required for the following controls:

- Type of coffee – "Listbox" control
- Preparation of coffee – "Text" control
- Payment – "Text" control
- Water temperature display – "Text" control
- Message texts – "Text" control as dynamic text

Visualization resources \ Text groups

Control reference \ Text

Exercise: Creating text groups for description texts

Create the following local text groups:

- Control
- Recipe
- Messages
- CoffeeTypes

Enter all text for the languages English and German. The values in parentheses indicate the index. Insert the text groups in the local node named "Text Groups".

1) Text group "**Control**"

Index	German	English
0	Ein	On
1	Aus	Off
2	Start	Start
3	Geld einwerfen	Insert coin

Table: Texts in the "Control" text group

2) Text group "**Recipe**"

Index	German	English
0	Preis	Price
1	Milch	Cream
2	Zucker	Sugar
3	Kaffee Pulver	Coffee powder
4	Wasser	Water
5	Wassertemperatur	Water temperature

Table: Texts in the "Recipe" text group

3) Text group "**Messages**"

Index	German	English
0	Maschine ist ausgeschaltet	Machine is switched off
1	Wassertemperatur noch nicht erreicht	Water temperature not yet stable
2	Wassertemperatur erreicht	Water set temperature reached

Table: Texts from the "Messages" text group

4) Text group "**CoffeeTypes**"

Index	German	English
0	Normal	Regular
1	Cappuccino	Cappuccino
2	Espresso	Espresso

Table: Texts from the "CoffeeTypes" text group

The texts can now be used to display static or dynamic text on controls in either English or German.

A third language, Chinese, has been defined in the shared resources. We will discuss this language under 5.10 "Languages and fonts".

When an object is inserted, it is given a default name (e.g. TextGroup_1). Text groups should always been given meaningful names, so that they can be easily identified later.

The possible types of coffee are shown in a list.

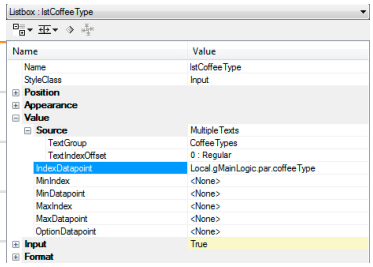
Exercise: Selecting coffee type from a listbox

As shown, the available coffee types are displayed in a listbox.

Insert a "**Listbox**" control on the "MainPage" and connect the "CoffeeType" text group to it.

Property	Value
Name	lstCoffeeType
Style class	Output
Value / Source	MultipleTexts
Value / Source / Textgroup	CoffeeType
Value / Source / IndexDatapoint	gMainLogic.par.CoffeeType
Format / alignment / Horizontal	Left
Format / Buttons / Slider	Never

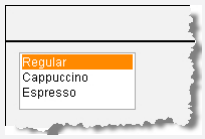
Table: Listbox properties



Listbox properties

1) Drag and drop the listbox control from the list of controls into the image editor.
2) Adjust the size of the control by clicking and dragging the points on its border.
3) Edit the properties.

There is now a listbox for selecting the type of coffee that looks like this:



"Listbox" control

Control reference \ Listbox

To check whether the texts for all languages fit into the control at its current size (text length, height and content), use the icon in the toolbar to switch the language.

Language switching in the visualization editor

A **"Text"** control is used to display a description of an object or text that changes during runtime.

Text control

Control reference \ Text

Exercise: Displaying description texts

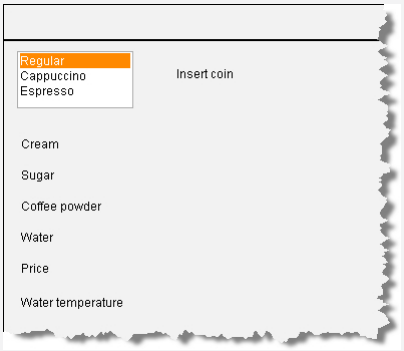
For the recipe data, the simulation of payment and the water temperature display, create description texts on the **"MainPage"** using **"Text"** controls.

The procedure is the same for all of the texts. The text for the water temperature display is used here as an example.

Property	Value
Name	txtWaterTemperature
Value / Source	SingleText
Value / Source / TextGroup	Recipe
Value / Source / Text	5: Water temperature

Table: Text field properties

The description texts have been created using **"Text"** controls and now look like this:



"Text" control

If multiple controls are selected by holding the **<CTRL>** key and clicking on them, they can be aligned using the alignment tool in the toolbar.

Alignment of controls

The last step for the texts is to display a dynamic text in a "Text" control.

Unlike a static text, a dynamic text is controlled by a process variable.

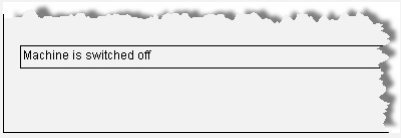
Exercise: Displaying message text

Create a message text on the „**MainPage**“ using a „**Text**“ control. A border around the text.

Property	Value
Name	txtMessage
Value / Source	MultipleTexts
Value / Source / TextGroup	Messages
Value / Source / IndexDatapoint	Local.gMainLogic.cmd.vis.messageIndex
Appearance / Border	Flat_Black

Table: Text field properties

The message texts have been created using "Text" controls and now look like this:



"Text" control for dynamic texts

5.10 Languages and fonts

A **language** refers to a specific language such as German or Japanese.

Spoken languages are divided into language families. Using **"Language Codes"** (according to ISO[2] 649-1 and 639-2) allows any international language to be identified clearly.

Languages can be added and managed under the shared resources in the **Languages** node.

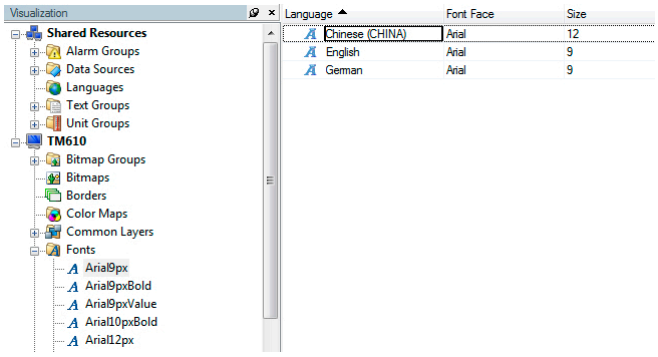
Shared resources \ Languages

A **fall-back language** can be defined for all texts that can't be translated during development. For these texts, any empty text field is filled with the fall-back language when the project is built.

The fall-back language is defined under the "**Languages**" node using the "FallBackLanguage" property.

2 ISO: International Organization for Standardization

In the field of typography, **fonts** refer to the graphic design of a set of characters. In Visual Components, fonts are managed under the "Fonts" node.



Visual Components fonts

So far we've been using the fonts defined by the styles.

A font must be configured for each language used in the project.

A font in Visual Components manages the font itself for each language as well as its properties.

- TrueType fonts
- Font size (in pixels)
- Bold and/or italics

The default project contains the fonts shown above. Specifying a font allows you to define its size and type when it is used in a property of a control.

Switching the language can also affect the font type and size (e.g. Asian text). If this is the case, the control must be configured to accommodate both font sizes.

Different font sizes depending on language

Exercise: Changing the font size of a language

For the language **"Chinese"**, change the size of the font "Arial9px" from 9 to 12 pixels.

Apply the Chinese text to the **"Messages"** text group. In the image editor for the "MainPage", switch the language to check the text size.

Switching the language in the visualization editor

Check the control named "txtMessage" (now we'll see who named their controls) or "Text_n" to see whether the font is set correctly:

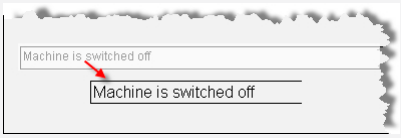
Property	Value
Name	txtMessage

Table: Text field properties

Property	Value
Appearance / Font	Arial9px

Table: Text field properties

The text in the messages' text field is larger. If the height and width of the text doesn't fit within the borders, the size of the text field must be increased.



"Text" control after switching the language

In section 5.12 "Touch and keypad control", the language is switched during runtime.

In addition to the fonts contained in Windows, a number of fonts are installed for Visual Components when Automation Studio is installed (licensing agreement must be accepted during installation):

In addition to the fonts contained in Windows, the following fonts are installed for Visual Components when Automation Studio is installed (licensing agreement must be accepted during installation):

- Arial and Arial Unicode
- Bitstream Vera Sans

These fonts do not require an additional license during runtime.

Many Windows fonts are protected by copyright. There may be certain licensing conditions limiting their use in Visual Components. Not all fonts can be used free of charge. This legal aspect should therefore always be checked before using a font!

Unicode is an international standard with the goal of defining a digital code for each graphic character or element in all text and character systems.

Unicode

Visualization resources \ Fonts

Visualization resources \ Fonts \ Using fonts

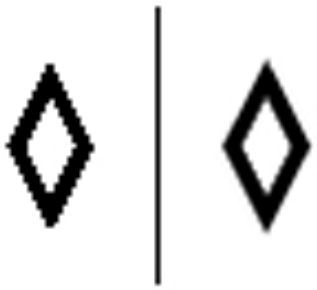
Shared resources \ Languages \ Unicode

Visualization resources \ Fonts \ Unicode fonts

Working with Visual Components \ Editing texts externally

Anti-aliasing smooths the edges of a font.

Anti-aliasing can be enabled or disabled for all texts in the project in the properties of the visualization object.

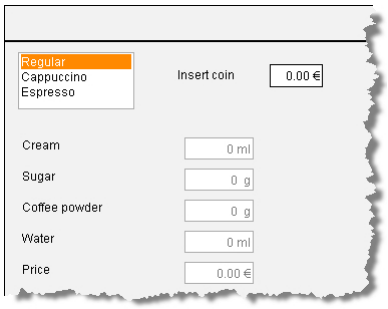


Anti-aliasing

Visualization resources \ Visualization object \ Anti-aliasing

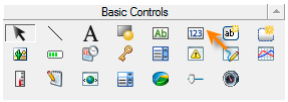
5.11 Displaying and modifying process values

In Visual Components, a numeric or alphanumeric control can be used to display or edit the value of a process variable.



Displaying and modifying process values

In the sample program, the process values are displayed and input using a numeric control.



Numeric control

The following elements should be displayed as numerical values:

- Simulation of payment – Input in a numeric field
- Preparation of the coffee – Display as a numeric field
- Insertion of money – Displayed as text
- Water temperature display – Displayed as a **scaled** numeric field

Action	VC components	Process variable
Coffee ingredients	Numeric field	gMainLogic.par.recipe.coffee gMainLogic.par.recipe.milk gMainLogic.par.recipe.sugar gMainLogic.par.recipe.water
Coffee price	Numeric field	gMainLogic.par.recipe.price

Action	VC components	Process variable
Payment	Numeric field	gMainLogic.par.givenMoney
Water temperature	Numeric field	gHeating.status.actTemp

Exercise: Simulating payment – Numeric input

Input takes place by clicking or touching inside the numeric control.

A **"NumPad"** touchpad found in the Visual Components project is used for the input. The input is limited to a value between 1 and 10.

Property	Value
Name	valueInsertCoin
Style class	Input
Value / Datapoint	gMainLogic.par.givenMoney
Value / MinValue	0
Value / MaxValue	10
Format / UnitText	Abbreviation

1) Drag and drop the numeric control from the list of controls into the image editor next to the description text "Insert money".
2) Change the StyleClass to "Input".
3) Connect the process variables, the limits and the unit text.

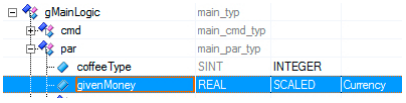
Changing the **"StyleClass"** property from **Output** to **Input** automatically sets the properties needed for inputs such as "Input = True", "Input/TouchPad = NumPad" and the border for differentiating between input and output fields.

Connecting the process variable **"gMainLogic.par.givenMoney"** and a unit text will display two decimal places and the limits with a unit.

Control reference \ Numeric

Value with unit text

The data point **"gMainLogic.par.givenMoney"** is a variable of type **"Real"**. Visual Components automatically displays data points with this data type with decimal places.



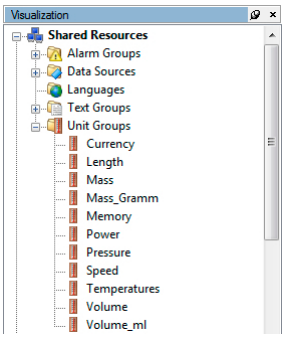
Data type "gMainLogic.par.givenMoney"

To display this data point with a defined number of decimal places and/or with a unit text (e.g. Euro €), the data point must be connected to a unit group.

Units and data points (data sources) are global resources. These have already been configured in the "Visu" visualization object and can also be used in this visualization. The following description shows how they are created.

Units are created and managed under the **"Unit groups"** node.

A unit text and the number of decimal places are defined in the unit group called **"Currency"**.



Unit groups

Index	Name	Unit Abbreviation	Unit Description	Default Precision	D
0	Euro	€	Euro	2	

"Currency" unit group

The Currency unit group only has one unit - Euro (€).

Since this unit group is not switched during runtime, the unit is automatically given Index 0 when it is assigned to a data point.

Since this unit only requires the unit short text, no scaling is necessary. Only the properties of the short text and the default precision are defined.

The Currency unit group is assigned to the data point of the **gMainLogic.par.givenMoney** process variable in the **Data Sources** view. The assignment can only take place if the setting **VCType = SCALED** has been made.

Name	PLCType	VCType	Unit Group / Subtype	Limit
gMainLogic	main_typ			
cmd	main_cmd_typ			
par	main_par_typ			
coffeeType	SINT	INTEGER		
givenMoney	REAL	SCALED	Currency	<Default>
receipe	main_par_re...			
status	main_status...			

Connect the "Currency" unit group to a scaled data point

Shared resources \ Unit groups

FAQ \ Display \ Displaying a value with unit text

The units and a scaling value are not configured for the control, but instead on the data point (process variable + properties).

Exercise: Outputting recipe values – Numeric output

The values for the recipe data are displayed in a numeric control.

The properties are described for the process variable "Sugar" and should be configured similarly for the rest of the recipe data.

Property	Value
Name	valSugar
Value / Datapoint	gMainLogic.par.receipe.sugar
Format / UnitText	Abbreviation

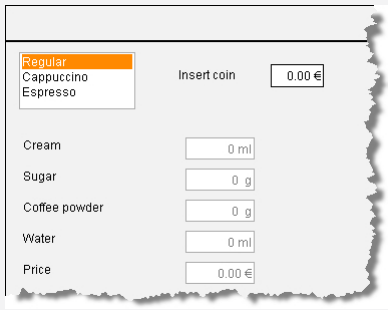
Table: Properties of the numeric output

1) Place the numeric controls next to the description texts.

2) Change the properties to match the ingredients.

For each process variable used to display the recipe data, a unit group containing the unit text is connected to the data point.

The values for the recipe data are displayed with units.



"Numeric" control

Value with unit switching

In the "CoffeeMachine" example, the current temperature of the water is mapped to the data point **"gHeating.status.actTemp"**.

The temperature is specified with a resolution of 0.1°C.

This causes the temperature sensor to return a temperature using a resolution of 0.1 degrees Celsius. Not all countries use Celsius, however. For some countries, the display needs to be in degrees Fahrenheit.

In order to display the water temperature, the following requirements must therefore be met:

- Scaling of raw value in °C
- Scaling of raw value in °F
- Scaling must switch depending on the language setting
- Display with one decimal place
- Display of the unit text °C or °F

Temperature

Before we continue explaining how to fulfill these requirements, we'll begin by creating the control for displaying the temperature.

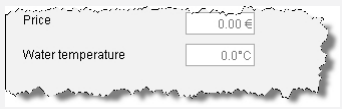
Exercise: Outputting water temperature – Numeric output

The water temperature is displayed using a numeric control.

Property	Value
Name	valTemperature
Value / Datapoint	gHeating.status.actTemp
Format / UnitText	Abbreviation

Table: Properties of the numeric output

The scaled temperature value is displayed with one decimal place and the unit text (°C or °F).



"Numeric" control with a scaled value and unit text

Scaling converts the physical value of the process variable into a value that can be displayed (process value) according to the linear equation **y=k*x+d**.

With a resolution of 1/10 degree, a physical value of 200 would correspond to a displayable value of 20.0°C.

K = 0.1 (incrementation by decimal)

d = 0 (Offset)

y = 0.1 * 200 + 0 = 20.0

The following formula is used to display degrees Fahrenheit:

°F = °C * 9/5 + 32

Scaling formulas based on the same physical value can be grouped into a **unit group**.

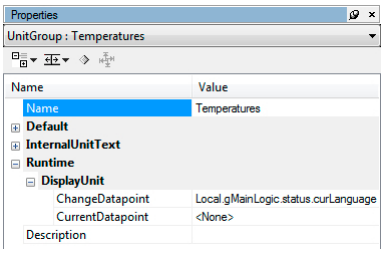
To switch the unit displayed with the temperature, the "**Temperatures**" unit group is used.

This group contains two units, **Celsius** and **Fahrenheit**. Each unit has a name, an abbreviated text, a long text and a precision (decimal places).

Index	Name	Unit Abbreviation	Unit Description	Default Precision	D
0	Fahrenheit	°F	fahrenheit	1	
1	Celsius	°C	celsius	1	

"Temperatures" unit group

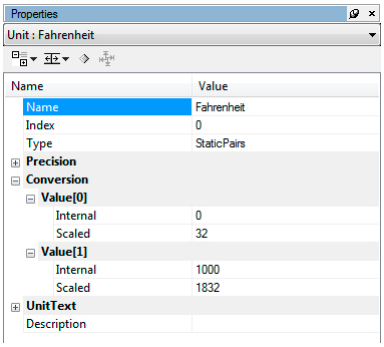
A process variable connected to the properties of the units group is used to switch the units during runtime.



Switching the units

Fahrenheit scaling

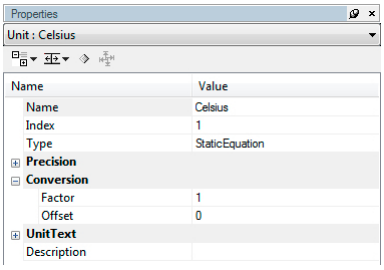
In the properties of each unit, the scaling is determined according to the linear equation. For Fahrenheit, the scaling uses a static value pair (Type = StaticPairs). Using the formula °F = 9/5 x °C + 32 we get internal values of 0°C = 32°F and for any other point on the line, e.g. 1000°C = 1832°F.



"Fahrenheit" units

Celsius scaling

No conversion is necessary for Celsius because the process variable already provides the value in degrees Celsius. Therefore, it is scaled by a factor of 1.



"Celsius" units

The unit group is assigned to the data point of the process variable **"gHeating.status.actTemp"**. The assignment can only take place if the setting **VCType = SCALED** has been made.

Name	PLCType	VCType	Unit Group / Subtype	Limit
gHeating	heating_typ			
cmd	heating_cm...			
par	heating_par...			
status	heating_stat...			
actTemp	REAL	SCALED	Temperatures	<None>
setTempOK	BOOL	BOOL		
gMainLogic	main_typ			

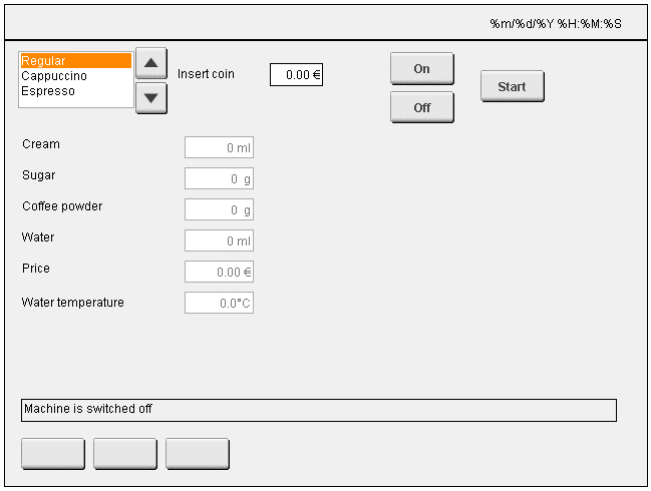
"Temperatures" unit group connected to data point

Shared resources \ Unit groups

FAQ \ Display \ Displaying a value with unit text

5.12 Touch and keypad control

Input takes place on a touch screen using buttons, hotspots or hardware keys.



Touch and keypad control

In Visual Components, each **key action** is mapped to a **virtual key**. This makes it possible to configure a key action independent of what hardware is used.

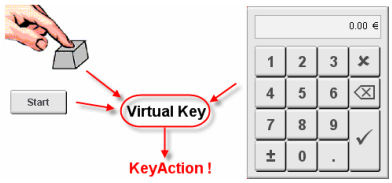
In the current exercise, operation will look like this:

- Selecting the coffee type
- Turning the coffee machine on and off
- Starting the preparation phase
- Switching the page on the screen
- Switching the language

A virtual key can be compared to a global image layer: It also has no function on its own until it is connected to the actual process image.

Virtual keys and key actions

A **virtual key** has no relation to a physical key or field on the touch screen.



Using virtual keys

It describes the behavior of a key using a key action. The location where this action takes place does not matter. The assigned action is not executed until the virtual key is linked to a physical key.

Hardware-independent configuration of key actions

Visualization resources \ Keys

A **key action** refers to the action that is performed when a key or touch screen button is pressed.

For the current example, this means that a data point is changed by a button in the following cases:

- When changing the type of coffee selected
- When writing 0/1 when turning off/on
- When writing 1 to start the preparation phase
- When writing to switch the language

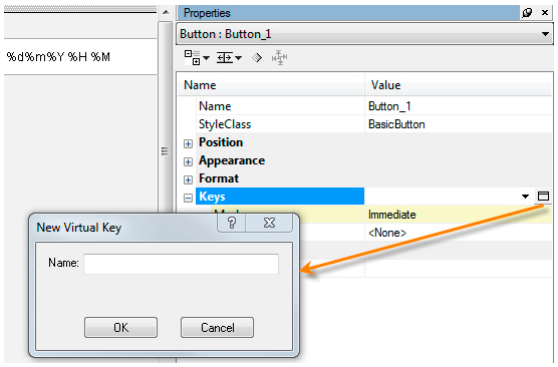
A page change is not performed using a data point, but rather using a separate key action.

Visual Components differentiates between local and global key actions.

Local key actions are only designed for one layer and only apply when this layer is active.

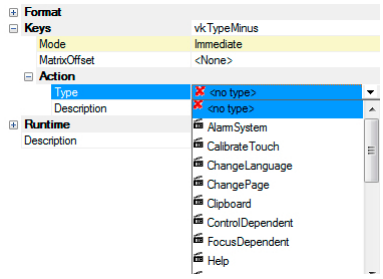
A local key action is created by inserting a button or hotspot (invisible button) in the image editor.

Before a key action is assigned to a virtual key, the virtual key must first be created. A virtual key is created using the "Keys" property of a button or hotspot by pressing the "Browse" button.



Creating a new virtual key

Enter a name that describes the action, then select a key action.



Selecting key actions

Visualization resources \ Keys \ Key actions

Global key actions apply for the whole visualization and are available on every image layer.

Virtual keys with global key actions are created in the workspace by double clicking on the **"Keys"** node.

Exercise: Changing the coffee type using two buttons

Use two buttons to change the type of coffee. The buttons should switch the process variable **"gMainLogic.cmd.coffeeType"** between the values 0 and 2.

The key action **"UpDownDatapoint"** for one button should increase the value of the data point from 0 to 2, and the key action for the other button should decrease the value from 2 to 0.

Select a style to differentiate between the two buttons.

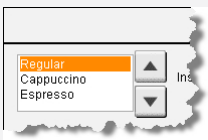
Property	Value
Name	cmdCoffeeTypeMinus
Style class	ScrollUpButton
Keys	vkTypeMinus
Keys / Action / Type	UpDown data point
Keys / Action / Repeat	False
Keys / Action / Value / Datapoint	gMainLogic.par.coffeeType
Keys / Action / Value / MinValue	0
Keys / Action / Value / MaxValue	2
Keys / Action / Value / StepValue	-1

Table: Settings for the scroll button

1) Insert two buttons next to the listbox.
2) Arrange them and adjust their sizes (e.g. 32x32 pixels).
3) Insert a new virtual key for each button.
4) **"UpDownDatapoint"** key action
5) Configure the properties for each button.

A negative value for the **"StepValue"** property reduces the value of the process variable, a positive value increases the value.

Setting the "StepValue" property to -1 and 1 allows the process variable to be decreased or increased.



"UpDownDatapoint" key action

Selecting appropriate styles helps differentiate between the two buttons visually.

Additional local key actions can be set for a virtual key with a global key action. When doing so, remember that there are also **exclusive key actions**, which prevent additional local key actions from being created.

Key action reference \ UpDownDatapoint

Visualization resources \ Keys \ Key actions

In this training manual, only local key actions are used.

Next, we will create controls to turn the CoffeeMachine on and off and start preparation.

Exercise: Buttons for turning on and off

Use two buttons to turn the machine on and off. The buttons should write the values 0 (off) and 1 (on) to the process variable **"gMainLogic.cmd.switchOnOff"**.

A button's **"SetDatapoint"** key action writes the defined value to the process variable.

Properties	Value
Name	cmdCoffeeOn
Keys	vkCoffeeOn
Keys / Action / Type	Set data point
Keys / Action / Value / Datapoint	gMainLogic.cmd.switchOnOff
Keys / Action / Value / SetValue	1

Table: Properties for the "On" button

1) Insert two buttons.
2) Arrange them and adjust their size.
3) Insert a new virtual key for each button.
4) "SetDatapoint" key action
5) Configure the properties for each button.

When the **"gMainLogic.cmd.switchOnOff"** data point is written with the value 1, the heating phase begins and the temperature rises. A message appears in the message line when the set temperature for the selected coffee type has been reached.

To have a button remain in a pressed state after being pressed, use the "ToggleDatapoint" key function.

Key action reference \ SetDatapoint

Key action reference \ ToggleDatapoint

A button is pressed to begin preparing the coffee.

The following states are queried in the application program so that the preparation phase can be started:

- The "CoffeeMachine" must be turned on.
- The water temperature must be correct.
- Payment must have taken place.

Exercise: Creating a button to start preparation

Create a button that starts preparation of the coffee. It must write the value 1 to the local process variable **"visCtrl.cmdStartCoffee"** as long as the button is pressed.

The **"MomentaryDatapoint"** key action writes a value to the process variable when the button is pressed and a different value when it is released.

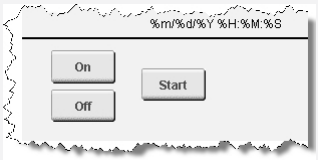
Property	Value
Name	cmdStartCoffee
Keys	vkStartCoffee
Keys / Action / Type	Momentary data point
Keys / Action / Value / Datapoint	visCtrl.cmdStartCoffee
Keys / Action / Value / SetValue	1
Keys / Action / Value / ResetValue	0

Table: Properties for the "Start" button

1) Insert a button
2) Arrange them and adjust their size.
3) Insert a new virtual key.
4) "MomentaryDatapoint" key action
5) Configure the properties.

Key action reference \ MomentaryDatapoint

Pressing the button begins preparation of the coffee. If all requirements are not met, the application program prevents preparation from starting.



Buttons with key actions

The next step will be to configure a page change between the main and service pages. The service page is used to switch the language.

These tasks should be performed on your own, using the Visual Components help system as a reference.

Exercise: Page changing and language switching

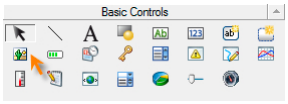
Configure a page change between the main page and the service page, as shown in the overview image (see section 5.1 "Exercise"). On the service page, a button can be used to switch the language.

1) Use the **"ChangePage"** key action.
2) Use the **"ChangeLanguage"** key action.

5.13 Using graphic objects

The graphic elements in a visualization are inserted, edited and managed under the **"Bitmaps"** node.

The graphic elements can be displayed in a process image using a **Bitmap** control.



Bitmap control in the toolbar

In this example the progress of the preparation is displayed graphically in three steps.

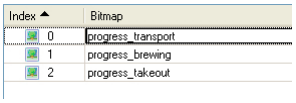
Under the **"Bitmap groups"** node, multiple graphic elements with the same size can be placed in logical groups.

- Moving the coffee cup into filling position
- Filling coffee ingredients and water into cup
- Moving the coffee cup into dispense position

Exercise: Graphic display of preparation

Import the following graphics files from the original "CoffeeMachine" project into a new bitmap group named **"ProgressStep"**.

These bitmaps are animated from the application program using the process variable **"gMainLogic.status.progressStep"**.

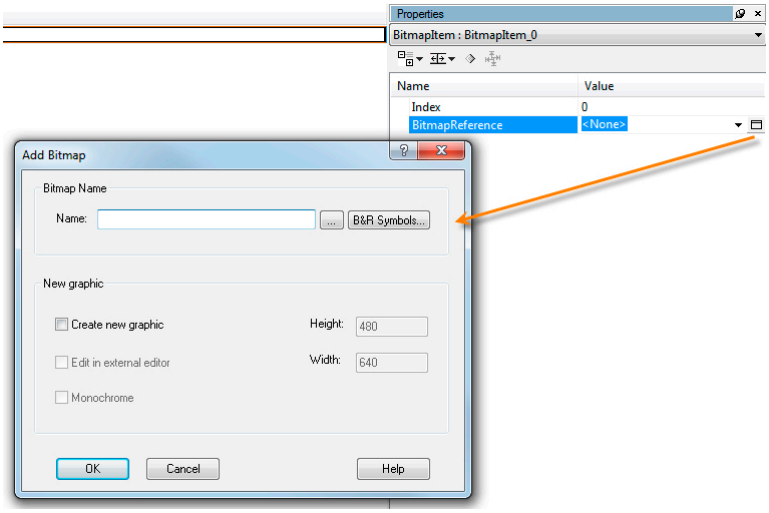


"ProgressStep" bitmap group

1) Create a bitmap group.
2) Import the bitmaps **"progress_transport.bmp"**, **"progress_brewing.bmp"** and **"progress_takeout.bmp"**.
3) Insert a bitmap control on the **"MainPage"**.
4) Connect a **"Multiple bitmap"** control to the new bitmap group.

5) Connect a data point for switching the bitmap.

Insert the bitmaps from the original "CoffeeMachine" project (visualization object **"Visu"**) using the Browse button in the **"Add bitmap"** dialog box.



Inserting bitmaps

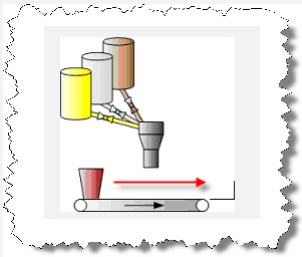
The bitmaps for the visualization object "Visu" are found in the following directory: "**..\Logical\Visualisation\Visu\Bitmaps**"

Visualization resources \ Bitmap groups

Visualization resources \ Bitmaps

Control reference \ Bitmap

The preparation process is displayed graphically by changing graphics after the "Start" button is pressed.



"Bitmap" control

So far, we have seen many of the features of Visual Components.

With the support of the Automation Studio help system, you can create your own visualization applications according to specific requirements.

The **FAQ** section included in the Visual Components VC4 help documentation provides approaches for completing various tasks.

FAQ

6 SUMMARY

Now that you've completed this training module, you are able to create a visualization application using the Visual Components editor.

Due to the countless possibilities provided by the Visual Components editor, we are not able to cover all of the editor's functions and features in this introductory training module.

Visual Components editor

A visualization object is configured independently of the hardware being used. This allows the application to be displayed either on a local display, on a remote terminal or in a VNC viewer.

The features for creating and managing one or more visualization projects in Automation Studio thus make it considerably easier to develop your projects.

TRAINING MODULES

TM210 – Working with Automation Studio
TM213 – Automation Runtime
TM220 – The Service Technician on the Job
TM223 – Automation Studio Diagnostics
TM230 – Structured Software Development
TM240 – Ladder Diagram (LD)
TM241 – Function Block Diagram (FBD)
TM242 – Sequential Function Chart (SFC)
TM246 – Structured Text (ST)
TM250 – Memory Management and Data Storage
TM400 – Introduction to Motion Control
TM410 – Working with Integrated Motion Control
TM440 – Motion Control: Basic Functions
TM441 – Motion Control: Multi-axis Functions
TM450 – ACOPOS Control Concept and Adjustment
TM460 – Initial Commissioning of Motors
TM500 – Introduction to Integrated Safety
TM510 – Working with SafeDESIGNER
TM530 – Developing Safety Applications
TM540 – Integrated Safe Motion Control
TM600 – Introduction to Visualization
TM610 – Working with Integrated Visualization
TM630 – Visualization Programming Guide
TM640 – Alarms, Trends and Diagnostics
TM670 – Advanced Visual Components
TM800 – APROL System Concept
TM811 – APROL Runtime System
TM812 – APROL Operator Management
TM813 – APROL XML Queries and Audit Trail
TM830 – APROL Project Engineering
TM890 – The Basics of LINUX
TM920 – Diagnostics and Service for End Users

TM1010 – B&R CNC System (ARNC0)
TM261 – Closed Loop Control with LOOPCONR
TM480 – The Basics of Hydraulics
TM481 – Valve-based Hydraulic Drives
TM482 – Hydraulic Servo Pump Drives

www.br-automation.com

TM610TRE.40-ENG / V2.0.0.4
©2013/04/11 by B&R. All rights reserved.
All registered trademarks are the property of their respective owners.
We reserve the right to make technical changes.