TM670

Advanced Visual Components

PERFECTION IN AUTOMATION
www.br-automation.com

Prerequisites and requirements

Training modules	TM610 – Working with Integrated Visualization TM640 – Alarm System, Trends and Diagnostics
Software:	Automation Studio 4.2 Coffee machine sample project
Hardware:	Automation Runtime simulation (ArSim) or Power Panel

Table of contents

1 Introduction

Visual Components includes a wide range of combinable and easy-to-configure standard components. The exercises in this training module are designed to deepen your existing basic knowledge of Visual Components. The additional information, configuration options and combination of various controls open up entirely new possibilities for designing a visualization application. The exercises in this training module are divided into several categories. The main areas of focus include dynamization of the visualization application itself, touch and key operation, adapting the visualization to the user and using the Visual Components programming interface.

The level of difficulty varies for the exercises in this training module. There is usually more than one way to arrive at a solution. The notes in this training module provide an overview of the range of functions and flexibility of Visual Components.

1.1 Learning objectives

This training module uses exercises to expand and reinforce your existing basic knowledge of Visual Components.

- You will learn the procedures and options for creating dynamic graphics, texts and dialog boxes in Visual Components.
- You will learn how to create multilingual keyboard layouts and assign multiple functions to the touch screen and hardware keys.
- You will learn about how the appearance of controls and the Visual Components configuration can be optimized.
- You will learn about the Visual Components programming interface and its corresponding documentation.
- You will learn how to run the sample programs for the Visual Components programming interface and adapt them to the needs of your application.

2 Dynamization of control elements

The configuration of static screen elements, whose color schemes and appearance can be determined in the visualization editor, already makes it possible to create many elements of an attractive visualization application. Going a step further, process variables representing a machine or process state can be used to change the color scheme of controls or show and switch between process graphics.

The following section shows how to modify the color scheme of controls at runtime, create dialog box fields and enter values indirectly.

2.1 Status data points and locking input fields

In Visual Components, data points can be used to deliver information about activities to the visualization application. It is also possible to use data points to influence the behavior of controls. All of the data points for evaluating status and runtime control are grouped together under the "Runtime" category for the respective control.

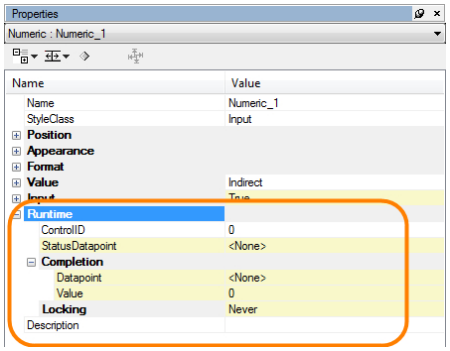


Figure 1: Runtime data points for the Numeric control

Completion data point

The Completion data point is available for all controls that allow input. After input has taken place, a data point is switched to a predefined value to indicate that confirmation has taken place. This value can then be read out in the control program. Input that is canceled has no effect on the value of the Completion data point.

Visualization \ Visual Components VC4 \ Control reference \ Numeric \ Runtime \ Completion

Locking data point

The Locking data point can be used to lock input fields. Any data point from the control program can be connected to the Locking data point for this. It is also possible to define limit values that determine when input is locked or unlocked for a control. In many cases, the Locking property is combined with the Password control.

Visualization \ Visual Components VC4 \ Control reference \ Numeric \ Runtime \ Locking

Status data point

Each control has a Status data point. The Status data point is a 16-bit value for checking and operating controls and layers. Bits 0 to 7 are used to operate the control. Bits 8 to 15 can be used to observe the control. A range of functions are available depending on the functionality of the control. Bit 0 is used to show or hide a control or layer. Bit 1 can be used to lock a control or layer. More information can be found in the Automation Studio help system.

Visualization \ Visual Components VC4 \ Control reference \ Numeric \ Runtime \ Status data point

Focus and edit colors

Different background colors are used depending on the state of the control. It is possible to define a color for the focus, a color for editing and a color for identifying locked controls for each screen in a visualization application. These colors are configured in the properties of the respective visualization screen.

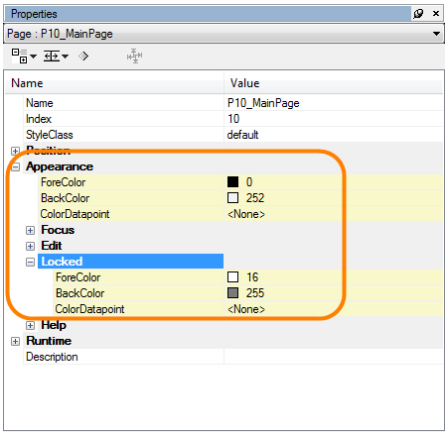


Figure 2: Global color settings for focus, editing and input locking

2.2 Switching colors at runtime

The foreground and background of controls can be changed at runtime using the value of "ColorDatapoint". The foreground and background colors each take up one bit. This is made easier through the use of a color map. A color map is a list of the colors that can be assigned to a control. "ColorDatapoint" can be used to switch the index of the color table – and therefore the foreground and background colors of the control.

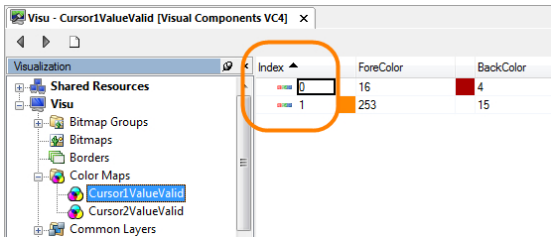


Figure 3: Configuring a color table, index for switching

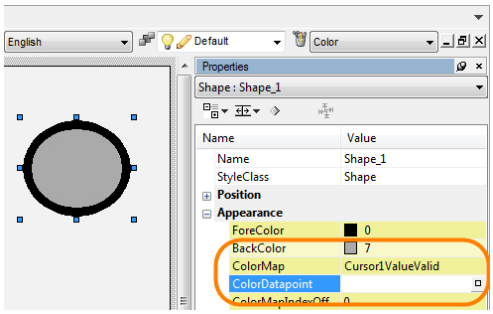


Figure 4: Connecting the control with the color map and attaching ColorDatapoint

Visualization \ Visual Components VC4 \

- Control reference \ Shape
- Visualization resources \ Color maps
- Visualization resources \ Palette

Exercise: Creating a traffic light using color tables and shapes

This exercise is based on the CoffeeMachine example project.

The goal is to create and "bring to life" a traffic light as a way to visualize the status of the coffee machine. The traffic light should turn red for the "Machine off" status, yellow for "Heating" and green for "Ready".

The Shape control should be used to represent the colors of the traffic light; color maps should be used to switch the colors.

1) Construct a traffic light using several Shape controls.
2) Create the corresponding color map(s).
3) Assign the color maps to the individual shapes.
4) Use ColorDatapoint to switch the color at runtime.

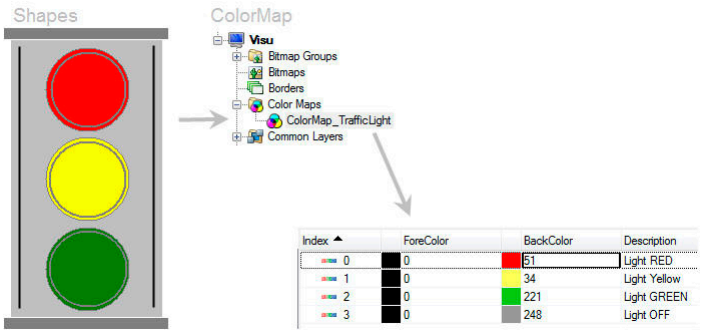


Figure 5: Overview image of shapes and the color map

Fill areas in bitmaps

Fill areas are an alternative to simply coloring in image areas. They allow monochromatic enclosed surfaces or lines to be colored using a ColorDatapoint. The assignment of color values and colors can be configured using palettes.

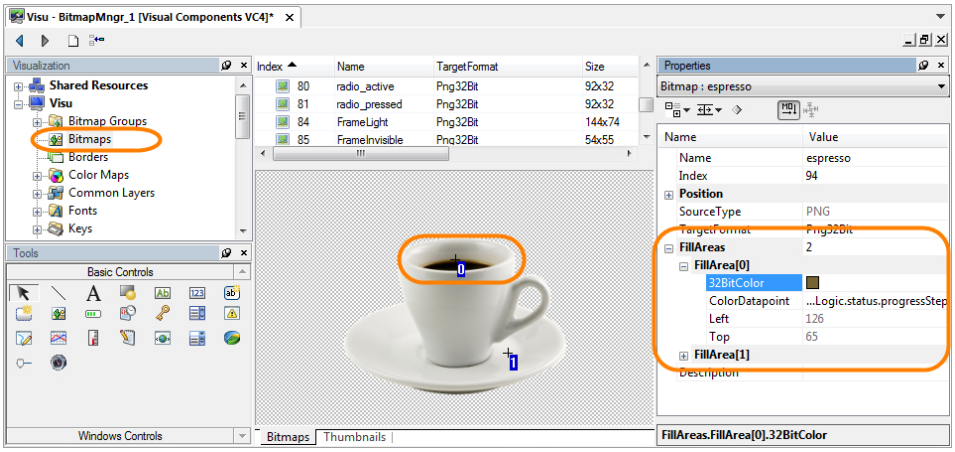


Figure 6: The color of consistent surfaces can then be changed at runtime with the fill area.

With the color palette that is being used, setting the ColorDatapoint to "10" changes the marked area to bright green.

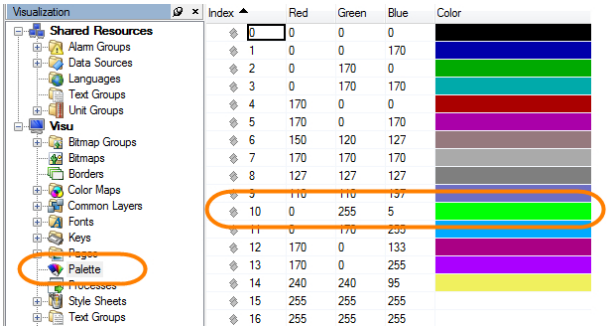


Figure 7: Color palette in the Visual Components project

Visualization \ Visual Components VC4 \ Visualization resources \

- Bitmaps \ Configuring bitmaps \ FillAreas
- Palette

2.3 Creating dialog boxes

In Visual Components, runtime data points are available for controls, layers and the visualization object (Runtime group). These data points make it possible to check user input as well as control the visualization object from the control program. For example, runtime data points can be used to trigger a screen change from the control program or show/hide a layer (see also "Status data points and locking input fields" on page 5).

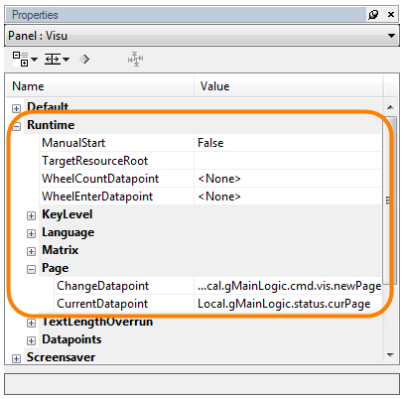


Figure 8: Runtime data points for the visualization object

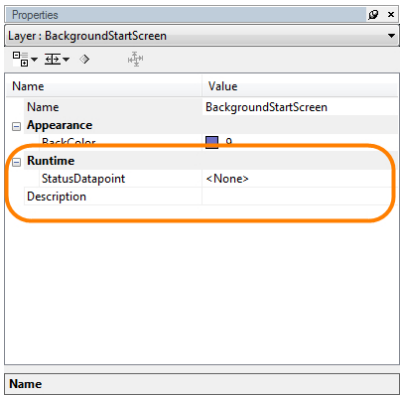


Figure 9: Runtime data points for a common layer

Visualization \ Visual Components VC4 \ Visualization resources \

- Visualization object \ Page data point
- Common layers \ Configuring layers \ Status data point

Exercise: Creating a dialog box using a common layer and list box

This exercise will create a button that functions like the Start button in Windows. Pressing this button should open a dialog box that lists all of the screens included in the visualization application. A list box will be used for navigation and selection. Pressing Enter will switch to the corresponding screen and close the dialog box.

1) Create a new common layer.
2) Add a Listbox control to the new layer.
3) Connect the Listbox control to a list of the visualization screens

   The Listbox control makes it possible to select from a list of elements. The Listbox's index data point can be used set the ChangePageDatapoint of the visualization. A text group can be created to list all of the visualization screens.

   A destination screen can be selected in the Listbox. When Enter is used to confirm the selection, the screen changes. The dialog box containing the Listbox should then be hidden.

4) Use the status data point of the layer to control visibility.

5) Add a layer reference to all visualization screens.

6) Add the Start button to the screen or footer.

   Pressing the Start button changes the status data point of the layer via the "SetDatapoint" key action, making the layer visible.

To complete this exercise, it will be necessary to configure the status data point and variables for switching the screen. These variables must also be used in the control program so that they can be created by the compiler.

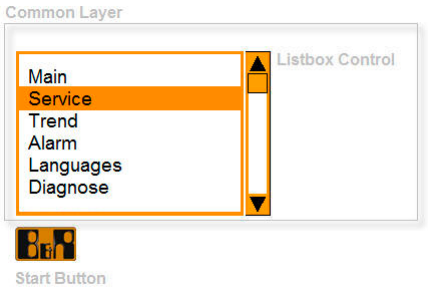


Figure 10: The dialog box for switching pages can be made visible by tapping on the B&R logo.

Visualization \ Visual Components VC4 \

- FAQ \ Layering \ Creating dialog boxes
- Visualization resources \ Common layers
- Visualization \ Visual Components VC4 \ Control reference \ Listbox
- Visualization \ Visual Components VC4 \ Control reference \ Button

2.4 Indirect value input

A user ID can be assigned in the data sources for each data point. This makes it possible to change the variable affected by the input at runtime. For this to work, the user ID must be unique in the Visual Components project.

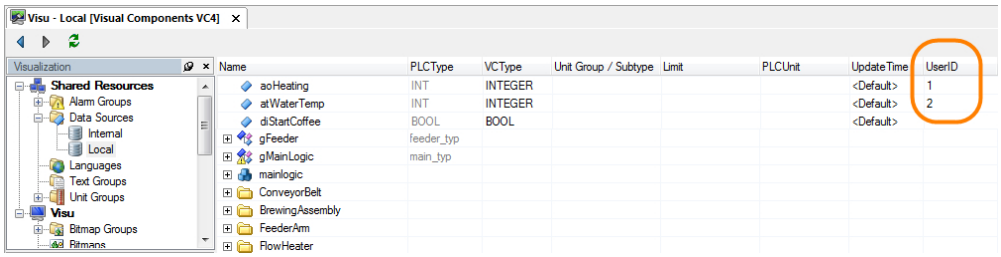


Figure 11: Configuring the user ID in the data sources

This user ID can be connected for each input field via the "Indirect" mode. The data point connected to the input field switches the user ID at runtime. The user ID can be used to implement toggled input fields. In addition, displaying a descriptive text dependent on the user ID is available as an expansion for touchpads.

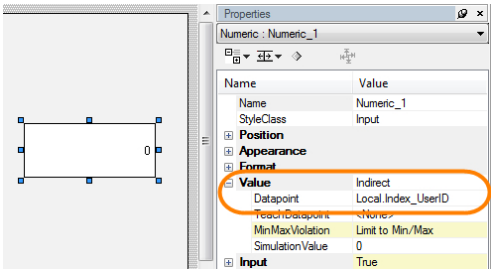


Figure 12: Input field configured in "Indirect" mode

Exercise: Configuring an input field and entering values indirectly

This example will test the function of the user ID to deepen understanding of this feature.

1) Create a new program called "advanced".
2) Declare variables

Variable	Data type	UserID
input_value1	INT	1
input_value2	INT	2
Index_UserID	USINT	-

Table 1: List of variables and data types

3) Enter a different user ID in each of the data sources.
4) Place a new numeric input field.
5) Change the mode from "Standard" to "Indirect".
6) Connect the "Index_UserID" variable under the "Datapoint" property.
7) Download and test the project.

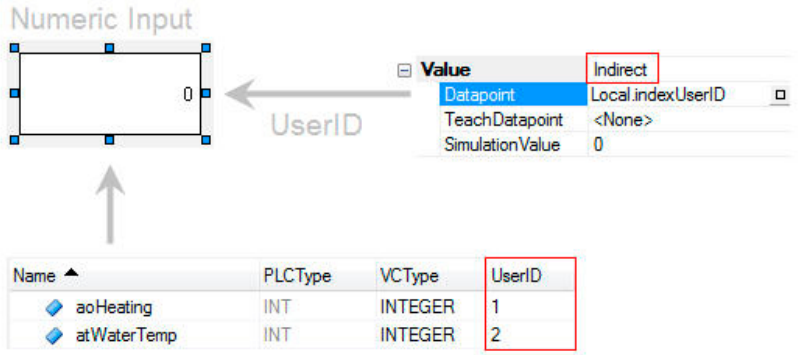


Figure 13: The relationship between the user ID in the data sources and the "Indirect" property of the Numeric control

Visualization \ Visual Components VC4 \

- Control reference \ Numeric
- Shared resources \ Data sources \ Configuring data points \ UserID
- FAQ \ Input and touch screen operation \ Displaying a description text on the touch-pad

3 Keys, touch operation and touchpads

Visual Components allows an HMI application to be operated using touch-operated buttons and areas. Predefined touchpads are provided for entering values and text. Depending on the use case, it may be necessary to adapt existing touchpads or create new ones. The key actions used for this can also be used for hardware keys on the visualization device or optional keypads.

A wide range of frequently used key actions are predefined in Visual Components. The image shows the list of key actions in the Automation Studio help system.

> Visualization \ Visual Components VC4 \ Key action reference \ CalibrateTouch

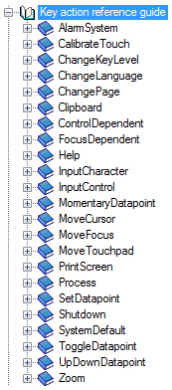


Figure 14: Key action reference in the Automation Studio help system

The next section discusses how to configure key actions for different control technologies.

3.1 Configuring keyboards

A Power Panel 500 with hardware keys is used in the example configuration. This will be used to show how to configure hardware keys and add a USB keyboard. How a PC keyboard for remote operation via VNC can be used will also be explained.

Virtual keys offer many advantages. Depending on the application, they can be linked with software keys (buttons on the touch display), hardware keys (integrated display keys) or even external keyboards such as the USB keyboard in this example. Different key layers make it possible to have a key output more than one character. For example, switching to a different layer allows for keys to show both uppercase and lowercase letters.

Displays with hardware keys

A key mapping file is added automatically when creating a hardware configuration for a device with hardware keys. This file is saved in the Configuration View. The mapping for the key mapping file can be seen in the configuration for the display unit.

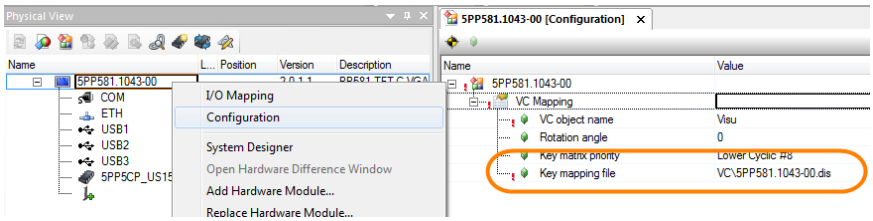


Figure 15: Mapping the visualization object and key mapping file

The key mapping file can be opened for editing from the display unit's shortcut menu. This opens an image of the device. Virtual keys can then be added by selecting a key.

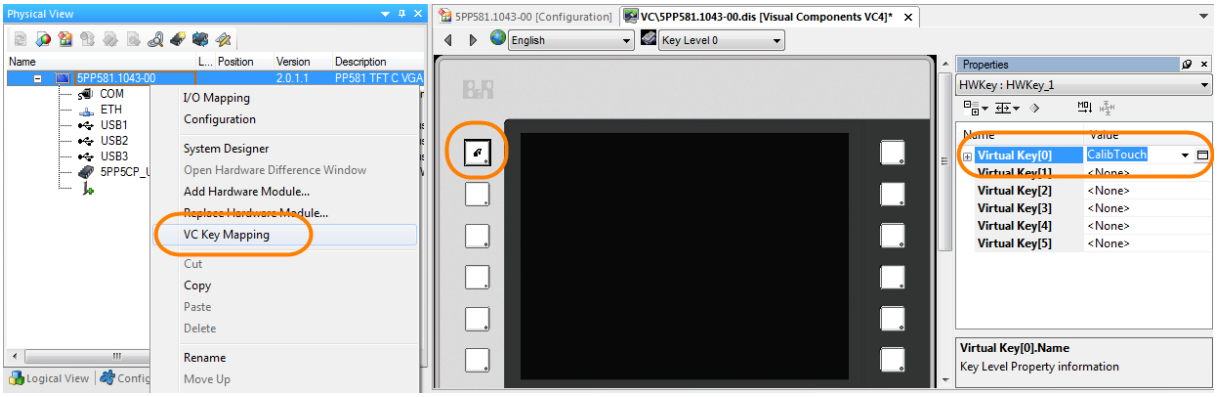


Figure 16: Opening the key mapping file for editing

Configuring a USB keyboard

A USB keyboard can be added to a project by first selecting the USB interface and then moving a USB keyboard to it from the Hardware Catalog using drag-and-drop. A key mapping file is automatically created for the USB keyboard in the Configuration View. This key mapping file can be opened for editing from the USB keyboard's shortcut menu.

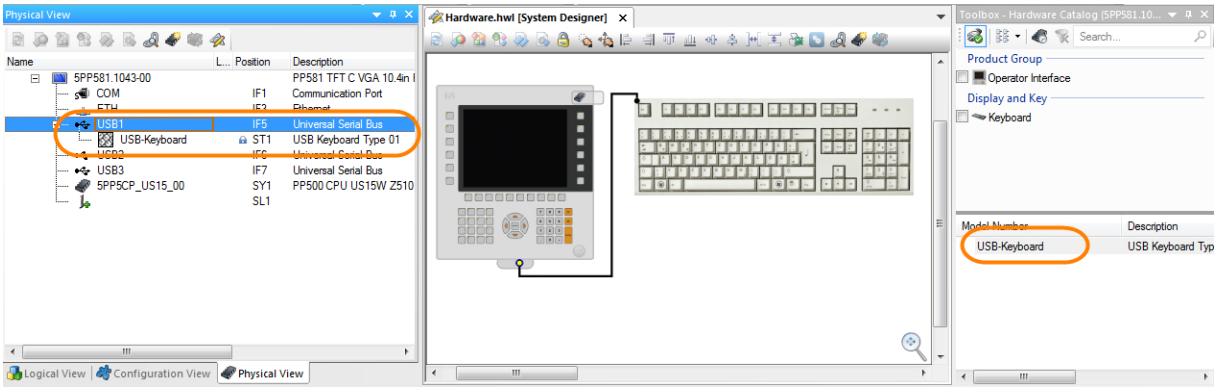


Figure 17: Adding a USB keyboard in the Physical View

Keys with the VNC Viewer

A VNC server can be configured in order to control a visualization application remotely. The settings for this care made in the controller's Ethernet configuration. When a VNC server is configured, a key mapping file is automatically created in the Configuration View. The key mapping file can then be opened from the shortcut menu in the VNC server's configuration.

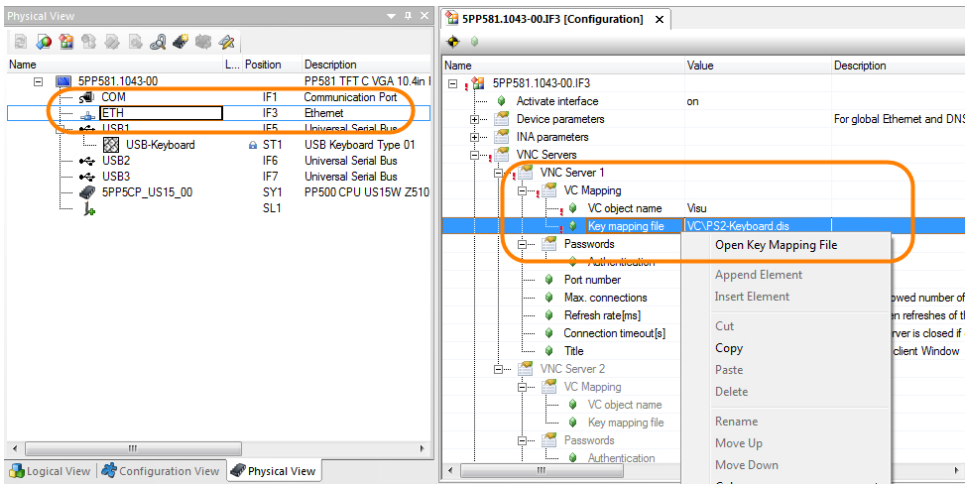


Figure 18: Opening the key mapping file in the VNC configuration

Exercise: Assigning virtual keys to a USB keyboard

A USB keyboard can also be added for on-site operation. In this case, the F1 key should be used to start touch calibration. The USB keyboard is only connected to the controller when it is needed.

1) Add the USB keyboard.
2) Open the key mapping file from the USB keyboard's shortcut menu.
3) Assign virtual keys to the keys

Visualization \ Visual Components VC4 \ Visualization resources \ Keys

- Mapping physical keys
- Switching the keyboard layout
- Configuring virtual keys

Visualization \ Visual Components Remote \ VNC

3.2 Calibrating the touch screen

External influences such as temperature fluctuations or improper installation of the display (distortion) may make it necessary to recalibrate the touch screen. The calibration data is stored on the CompactFlash card and retained until the next recalibration or until the memory is deleted.

If the touch screen is so incorrectly calibrated that it cannot be used to start calibration, then calibration must be started from the application.

Visual Components offers several different ways to start touch screen calibration.

Starting calibration using a touch button

The "CalibrateTouch" key action can be used to start touch screen calibration from buttons and keys (including a USB or VNC keyboard).

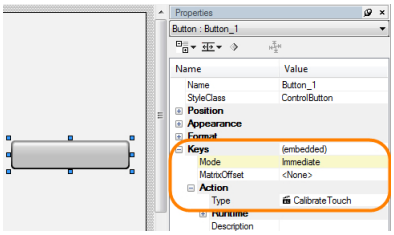


Figure 19: Starting touch screen calibration with the "CalibrateTouch" key action

Visualization \ Visual Components VC4 \

- Control reference \ Button
- Key action reference \ CalibrateTouch
- FAQ \ Input and touch screen operation \ Calibrating a touch screen
- FAQ \ General information \ Storing the VC configuration file

Starting calibration from a hardware key or USB keyboard

The previous section explained how to add a USB keyboard. The virtual key for touch screen calibration can be connected to the USB keyboard. The USB keyboard can be connected to the controller if this is needed.

Visualization \ Visual Components VC4 \

- FAQ \ Input and touch screen operation \ Calibrating a touch screen
- Control reference \ Button
- Visualization resources \ Keys \ Mapping physical keys

Starting calibration from the application

The control program itself can also be used to start touch screen calibration or read the status of the calibration. Function blocks included in the VISAPI library can be used to do this. Calling the calibration functions in the VISAPI library is already part of a prepackaged library example.

Programming \ Libraries \ Visualization \ VISAPI

Examples \ Libraries \ Visualization \ Display and touch

Exercise: Triggering touch calibration using a certain node number

Touch screen calibration should be started after the controller is booted – but only if the controller's node number is set to "16#F1". The controller's node number must therefore be read during booting. Depending on the configured node number, calibration can be started using the VISAPI library. A separate library sample exists for reading the node number and calibrating the touch screen.

1) Reading the node number using the function blocks in the AsARCfg library
2) Starting touch screen calibration using the function blocks in the VISAPI library

Programming \ Examples \ Libraries \

- Visualization \ Display and touch
- Configuration, system information and runtime control \ Managing network settings

3.3 Creating touchpads

Visual Components makes it possible to create new touchpads or modify those that already exist. Each touchpad consists of bitmaps and a grid used to map the virtual keys. The grid determines which key actions are executed at which position of the touchpad.

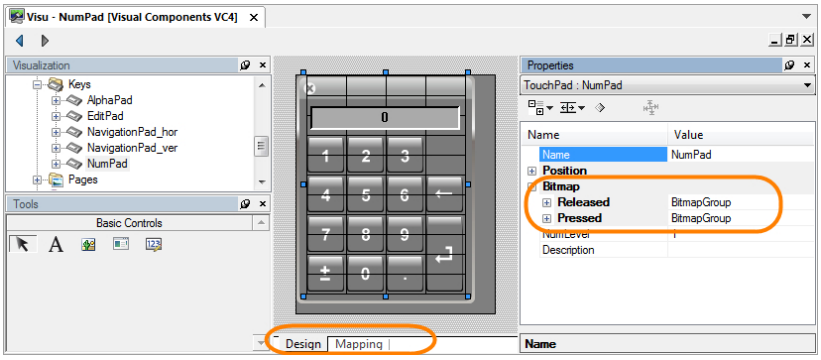


Figure 20: Editor for touchpads: The bitmaps and grid determine the structure and function.

Exercise: Creating a touchpad for trend operation

This exercise will create a touchpad with special functions for the trend control. This touchpad will make it possible to zoom into and out of trends.

1) Open the visualization object.
2) Add a new touchpad to the resources under Keys (right-click / Add touchpad).
3) Add a corresponding bitmap for the touchpad layout under Properties.
4) Scale the grid accordingly and place it over the bitmap.
5) Open the "Mapping" tab (lower left of the editor).
6) Link the individual grid elements with their respective virtual keys (zoom in / zoom out).

7) Set "Input" to TRUE and link the touchpad to the trend.

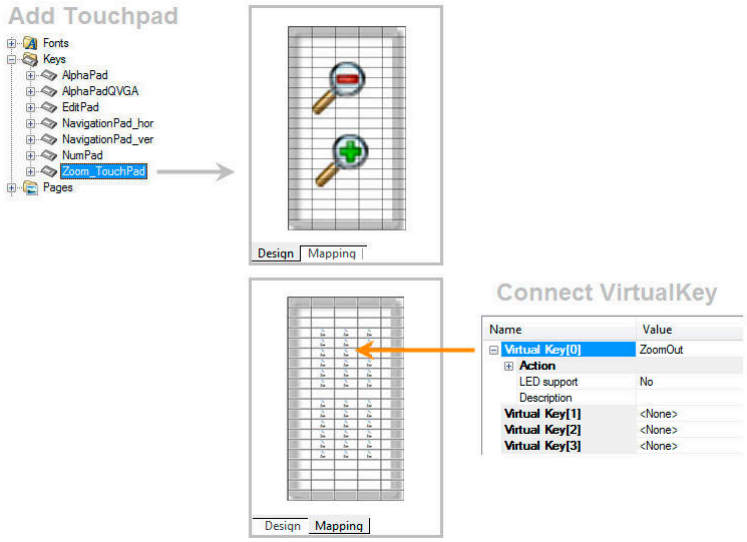


Figure 21: Overview image - Creating a touchpad

Visualization \ Visual Components VC4 \ Visualization resources \ Keys \ Creating touchpads

FAQ \ Input and touch screen operation \ Displaying a description text on the touchpad

4 Optimizing the display and operation

For sophisticated operating concepts, it is essential to account for the demands of the operator at the time when the visualization is configured. Only those areas of the visualization that are actually necessary for operation by the respective user should be active and accessible. User-optimized display of texts and values is also important.

The following section shows how controls and layers can be hidden and locked, for example, using password input. Filter functions for alarm lists and displaying multi-line texts will be demonstrated further on.

4.1 Using password protection

The password control is used to enter and encrypt passwords. Passwords can be stored in the project as a text group or in the controller as a string array.

"LevelDataPoint" represents the return value that provides the password level depending on the password entered. "LevelDataPoint" is reset to the value "0" if the password entered is incorrect. The value of "LevelDataPoint" could be used, for example, to lock input to all controls with the "Locking" property. Depending on the configuration, input can be allowed or blocked depending on the password entered.

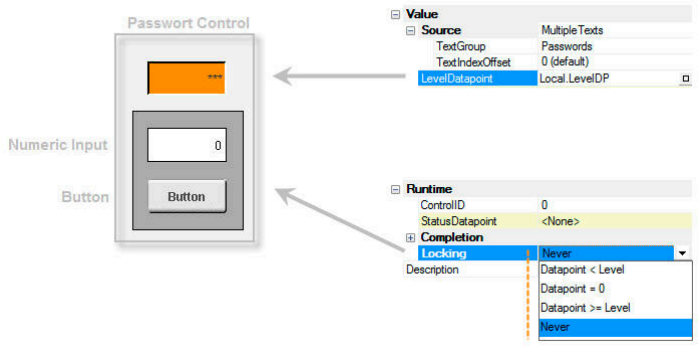


Figure 22: Password control - Overview image

Visualization \ Visual Components VC4 \ Control reference \ Password

Visualization \ Visual Components VC4 \ Control reference \ ... \ Runtime \ Locking

Exercise: Configuring the password control

This exercise will demonstrate how to set up password management in the visualization application. Three different users with different levels of access (password levels) will be created.

- User 1 can only use the button. The input field is locked.
- User 2 can only use the input field. The button is locked.
- User 3 has sufficient access to operate both controls.

It is possible to complete this exercise using the Password control. This is where the users enter their password. Depending on the password level, the controls are unlocked and access is allowed.

1) Add the Password control, button and Numeric control on one side.
2) Create a text group for the different passwords.
3) Configure the Password control and read "LevelDataPoint".
4) Configure the button and Numeric input control and connect the "Locking" data point.

4.2 Filtering in the alarm history

Filter options are a good way to improve the organization and layout of the alarm list. It is possible to filter by alarm, group, priority or time. Multiple data points are used to set up filtering. For example, a time and a priority can be specified for the filtering. A filter control byte can be used to enable and combine various filter options.

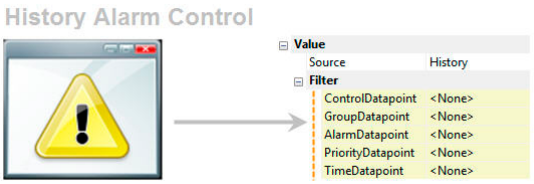


Figure 23: Filtering options - Overview image

Visualization \ Visual Components VC4 \ Shared resources \ Alarm system \ Viewing alarms during runtime \ Filtering alarms

Exercise: Filtering historical alarms

The CoffeeMachine sample project is used for this exercise. An alarm control has already been added to show historical alarms. The current date should be used for the filter. The time for the alarm filtering can be entered in the visualization application. The alarm list should be filtered so that all alarms older than the entered time are displayed.

1) Declare variables for filter types in the control program.
2) Declare variable for filter control byte.
3) Connect variables with alarm control.
4) Test and combine filters.

4.3 Displaying multi-line texts

There are a wide variety of configurable views for the texts in the Text control. Line breaks can be inserted automatically or manually for longer texts. Control characters are also supported for line breaks. The behavior of line breaks can be configured using the "Multiline" property.

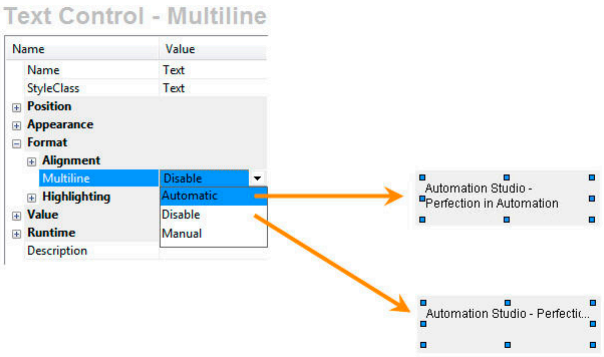


Figure 24: Overview image - Text control settings - Multiline

Exercise: Configuring multi-line texts in the Text control

Depending on the visualization concept and the space available on the display, it may be necessary to display text across multiple lines. The goal of this exercise is to determine and test the different options for texts.

1) Add the Text control.

2) Try out the different "Multiline" options.

A line break for alarm texts can be forced using the "\n" control character. Line breaks are automatically inserted for longer texts in the Alarm control.

Visualization \ Visual Components VC4 \

- Control reference \ Text \ Format \ Multiline
- FAQ \ Display \ Text with control characters

Using text snippets

Text snippets can be used in order to more flexibly configure texts. At runtime, text snippets make it possible to compose texts that are made up of variable values, strings and texts from other text groups. Text snippets can be used in the Visual Components alarm system as well as in all text groups. The text snippet editor can be accessed via the "TextSnippets" tab. Text snippets can be added to a text in a text group using the shortcut menu.

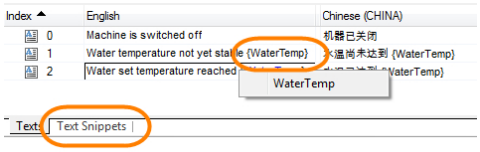


Figure 25: Adding text snippets within a text group

Visualization \ Visual Components VC4 \ Visualization resources \ Text groups \ Texts in a text group \ Text snippets

5 Additional functions

5.1 Resizing a visualization application

A visualization application can be resized at a later time. This is done by selecting "Change resolution" from the visualization object's shortcut menu. A dialog box will open that can be used to resize the application as needed.

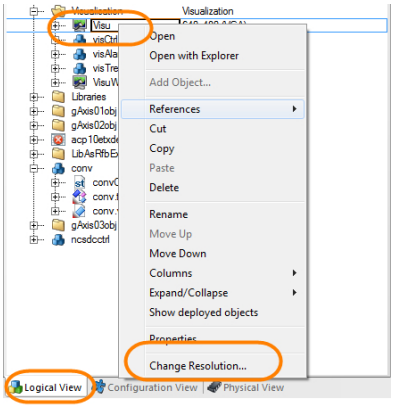


Figure 26: Changing the size from the shortcut menu in the Logical View

Visualization \ Visual Components VC4 \ Working with Visual Components \ Resize visualization

5.2 Additional controls

Visual Components includes a wide array of additional controls that have not yet been discussed. The following is a brief overview of additional useful controls as well as references to their usage in the Automation Studio help system.

Edit control

The Edit control makes it possible to load and save files. For example, it allows CNC programs to be edited. This control also supports syntax highlighting and can be operated using predefined key actions and touchpads.

```
(Tasse)
%L1

G1 Z40 F10000
G170 M51
$IF EXW98==1
G1 Z-25 F10000
M40
G4 0
G19
G3 X16.0  Y-20 Z-23 R=100

G1 Y-22 F25000
G1 Y-20 F25000
M41
G2 X37.0 Y14  Z-23 R=100 F8000
M56

G3 X16.0  Y-20 Z-18 R=100
G1 Z-5
```

Figure 27: CNC program in the Edit control

Slider control

Value changes can be illustrated with the help of the Slider control. The slider can also be used to enter values.

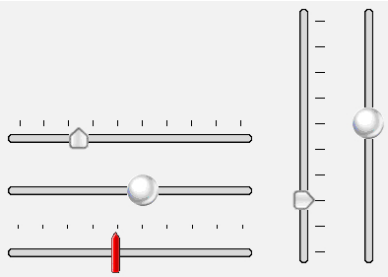


Figure 28: Various implementations of the slider control

Gauge control

The Gauge control is used to display gauges. The background, scale and appearance of the pointer can be configured individually.

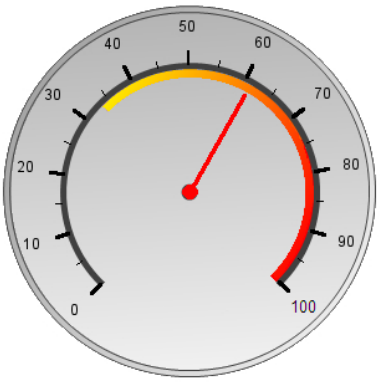


Figure 29: Gauge control

Media control

The Media control can be used to play various types of video and audio formats or to embed webcams on Windows terminals.

Visualization \ Visual Components VC4 \ Control reference

- Edit
- Slider
- Gauge
- Media

5.3 Internal data sources

A series of internal data points for shared resources is available under the "Data sources" node. This is where device information is provided, for example. These data points are read-only.

Visualization \ Visual Components VC4 \ Shared resources \ Data sources \ Internal data points

5.4 Importing and exporting text

All texts from the Visual Components object can be exported for translation. To do this, an export language and reference language must be selected. The fully translated texts can then be put back into the Visual Components project using the import function. If translations are missing, a default language can be selected.

If not all texts have been translated by the time commissioning takes place, then it is still possible to copy the outstanding texts directly to the machine controller's CompactFlash card using export files. The target language must be created and compiled in the project.

Visualization \ Visual Components VC4 \ Working with Visual Components.

- Editing texts externally
- The Export Texts wizard
- The Import Texts wizard

Visualization \ Visual Components VC4 \ FAQ

- Problems with text width and translation
- Editing export files

Visualization \ Visual Components VC4 \ Shared Resources

- Configuration \ TargetResourceRoot
- Languages \ Fall-back language

6 The Visual Components programming interface

Programming interfaces provide expanded functionality to the already extensive range of configuration functions offered in Visual Components. Access to the visualization application via programming is mainly handled using the VISAPI, VCScrsht, AsRfbExt and VcLib libraries. The following section will provide basic information about the areas where these libraries can be used.

6.1 Reading and storing the alarm history

It is enormously advantageous to use the library examples included in Automation Studio. Important basic functions are already available as a completed solution and can very quickly be imported into another Automation Studio project. All it takes to combine these examples with more complex applications are minor adjustments. This makes it much easier and faster to create applications.

Exercise: Reading and saving the alarm history to a file

The CoffeeMachine sample project can be used for this exercise.

It already contains a configured alarm control and list of historical alarms. The task here is to use the function blocks in the VISAPI library to read the individual alarms and save them to a file with the help of the FileIO library.

To do this, the two already existing program examples "LibVisApi1_ST" and "LibFileIO1_ST" can be used and adapted accordingly.

1) Add the "LibVisApi1_ST" and "LibFileIO1_ST" library examples.
2) Adjust the program components for reading and saving the alarm history to a file.

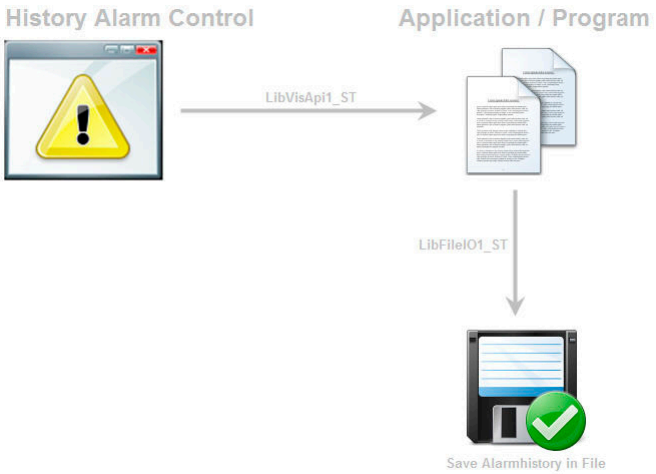


Figure 30: Reading and saving the alarm history to a file - Overview image

Visualization \ Visual Components VC4 \ Control reference \ Alarm

Programming \ Examples \ Libraries

- Visualization \ Trends and diagnostics
- Data access and data storage \ Data storage

6.2 Drawing in the DrawBox control

The VISAPI library is a collection of programming functions for the visualization application. Among other things, it contains graphics functions that can be used to create graphics objects at runtime. Drawing functions make it possible to output lines directly to the visualization application, for example. The point of origin is the upper left corner of the display. Using the DrawBox control creates a container with its own point of origin. The DrawBox can be moved to the Visual Components project at a later time without having to make any changes to the program.

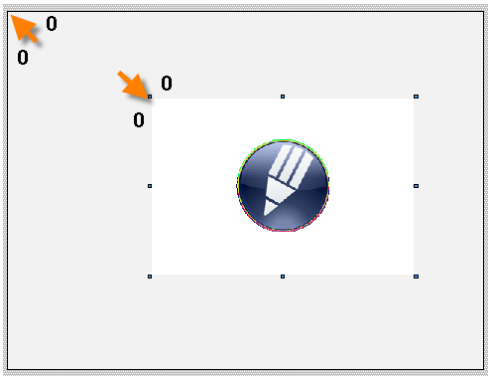


Figure 31: Point of origin of the visualization object and DrawBox

Exercise: Outputting graphics elements at runtime

The first part of the exercise will output various graphics elements (lines, rectangles, circle, bitmaps, texts) directly in the visualization application.

The second part of the exercise will use the DrawBox control to show the graphics elements programmed in the first part of the exercise.

1) Creating a new program
2) Use the VISAPI library to establish a connection to the visualization object.

   The connection to the visualization object is established with the "VA_Setup" function. This procedure only has to be done once after the controller is booted.
3) Request visualization resources.

   The "VA_Saccess" function is used to request visualization resources. Drawing functions can only be called if access has been carried out successfully.
4) Output graphic elements.

   The "VA_Line" and "VA_Rectangle" functions can be used to output a variety of graphic elements to the visualization applications.

5) Deallocate visualization resources

Calling the "VA_Srelease" function causes the application to deallocate the necessary visualization resources, and the Visual Components runtime environment begins executing again.

As long as the VISAPI library is active accessing the visualization application ("VA_Saccess"), the visualization will be frozen.

Exercise: Using the DrawBox control

The previous solution has the disadvantage that the point of origin is always in the top left corner and access takes place regardless of the active screen page. The DrawBox control can be used to define its own point of origin.

1) Add a DrawBox to the page.

2) Expand the program from exercise 1.

To access the DrawBox control, it is necessary to call the "VA_Attach" function after the "VA_Saccess" function call. The name of the page and layer as well as the DrawBox control is transferred.

The graphics output can then take place.

The DrawBox is closed using the "VA_Detach" function.

The visualization resources are deallocated with "VA_Srelease.

Visualization \ Visual Components VC 4 \ Control reference \ DrawBox

Programming \ Libraries \ Visualization \ VISAPI

6.3 Taking screenshots

The VCScrsht library makes it possible to take screenshots of the currently displayed visualization page. A screenshot can be taken by pressing a button, which calls the corresponding program function for creating the screenshot. The resulting bitmap is saved to the CompactFlash card. The screenshot can then be copied from the controller to a PC, for example, via an FTP connection[1].

Exercise: Saving screenshots to a USB flash drive

The VCScrsht library consists of three function blocks. The "LibVcScrSht_ST" library example can be used to take a screenshot. By specifying a file device, screenshots not only can be stored on a CompactFlash card, but also on a USB mass storage device.

1) Configure the file device.

[1] File Transfer Protocol is a standard network protocol specified in RFC 959 for transferring files via TCP-based networks. Source: Wikipedia

A file device represents a system variable that addresses a physical memory location. The access to the file device takes place in the program. The advantage of this is that the memory location in the system configuration can be changed without having to change the program.

2) Import the sample program for taking a screenshot.

3) Add a button for triggering a screenshot in the visualization.

4) Test the project.

Programming \ Libraries \

- Visualization \ VCScrsht
- Data access and storage \ FileIO \ General information \ File devices, files and directories

Programming \ Examples \ Libraries \ visualization \ Graphics and text \ Generating a screenshot

6.4 Reading user actions

The functions in the VcLib library can be used to read information from the visualization at runtime. This includes things like determining the process variables currently connected to a visualization page or detecting events involving user input. In order to use VcLib, it is necessary to configure the necessary user IDs in the data sources. The record of user events can be configured in the properties of the shared resources. The basic functions have already been implemented in the "libvclib1_ST" library example.

Programming \ Libraries \ Visualization \ VcLib

Programming \ Examples \ Libraries \ Visualization \ Process variables

Visualization \ Visual Components VC4 \ Shared resources \ Configuration \ EnableEvents

6.5 Controlling VNC visualization applications

The AsRfbExt library enables access to a VNC server. It is then possible to read out device-specific data[2] from visualization devices connected to the controller via VNC. It is also possible to determine the number of connected clients, to remove connections or change the title bar of the VNC viewer's window.

Exercise: Changing the VNC window title bar

The goal of this exercise is to call the functions in the AsRfbExt library. The title bar of the window on the VNC client should be changed.

1) Import the sample program.

2) Adjust the name of the VC object.

[2] Device-specific data refers to data from B&R devices that are equipped with hardware keys, handwheels, etc. and connected to the controller via VNC. This would include the operating elements on the MP40/50, for example.

3) Change the window text to "Visual Components Advanced".

4) Testing the function

The window title has been changed by calling the program code.

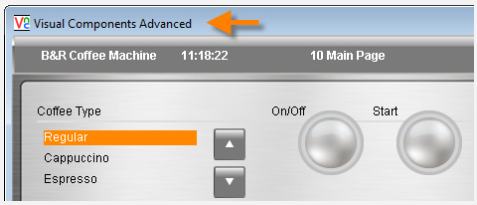


Figure 32: Changed window title for the VNC viewer

Programming \ Libraries \ Visualization \ AsRfbExt

Programming \ Examples \ Visualization \ Remote visualization

6.6 mapp technology file management

The MpFile library provide complete file management functionality. It has been designed for use with Visual Components.

The UIConnect structure is given a list of file devices, which can be switched between using an index datapoint. The MpFileManagerUI component automatically makes a file list available for the selected file device. This list is linked to the Visual Components visualization application using text output fields. Elements of the UIConnect structure are used to allow scrolling and paging up and down in the list. Many file and folder operations are available in the same way.

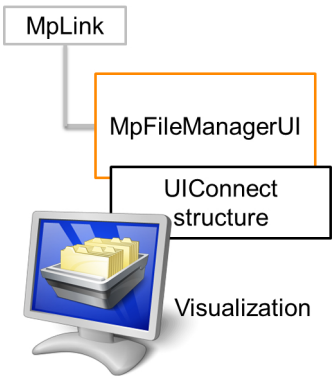


Figure 33: File management with the MpFileManagerUI component; UIConnect structure for interaction with the visualization application

Application layer - mapp technology \ Components \ Infrastructure \ MpFile - File management system

Exercise: Enable the file management system component

The file management component is available to display the folder contents of the file system. It allows files and folders on selected file devices to be listed. A UIConnect structure is available for displaying data and carrying out operations in a Visual Components visualization application. Command variables to scroll in the list and also file operations have already been made available here.

1) Determine file device in the CPU configuration
2) Add and call the MpFileManagerUI component.
3) Specify the length of the file list in the HMI application

   The value can be set using a structure of type "MpFileManagerUISetupType". However, the standard value can be left for this exercise.

4) Preconfigure used file devices

   The used file devices can be transferred to the subelement "DeviceNames" using a structure of type "MpFileManagerUIConnectType".

5) Transfer the program
6) Check the status ID.
7) Read the value of the "File.PathInfo.FileCount" subelement in the MpFileManagerUIConnectType structure to check the number of files on the active file device.

The call is displayed as follows in Ladder Diagram programming language.

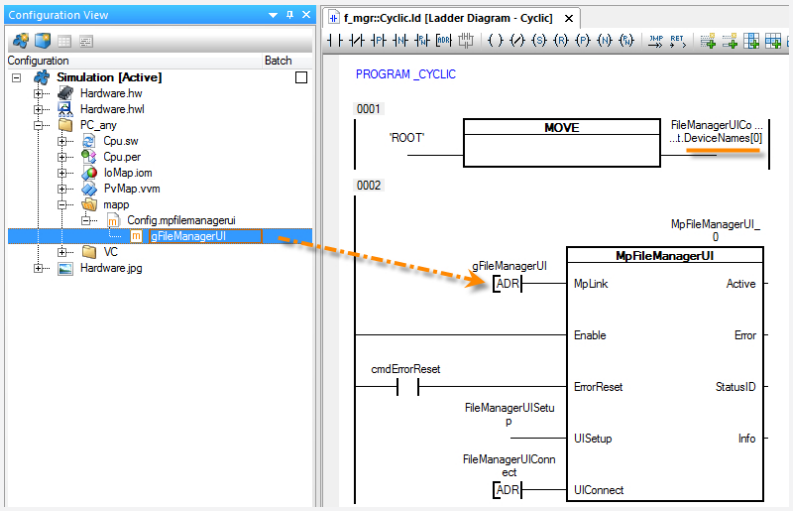


Figure 34: Left: Configuration View with mapp configuration with MpLink; right: Ladder Diagram program with MpFileManagerUI component and configuration structures

Exercise: Configure the visualization application for a file browser

The objective of this task is to create a visualization application for the file management component. For this purpose, the mapp technology template for Visual Components can be implemented as the fastest method. The preinstalled visualization page for the MpFileManagerUI component is connected with the UIConnect structure of your own application using the Visual Components Refractor function.

1) Insert a new visualization in the Logical View
2) Select mapp technology template

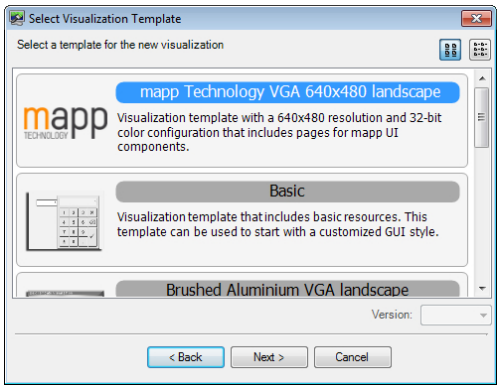


Figure 35: mapp technology template for Visual Components

3) Update data sources

4) Reconnect the FileManagerUIConnect structure using the Refactor function

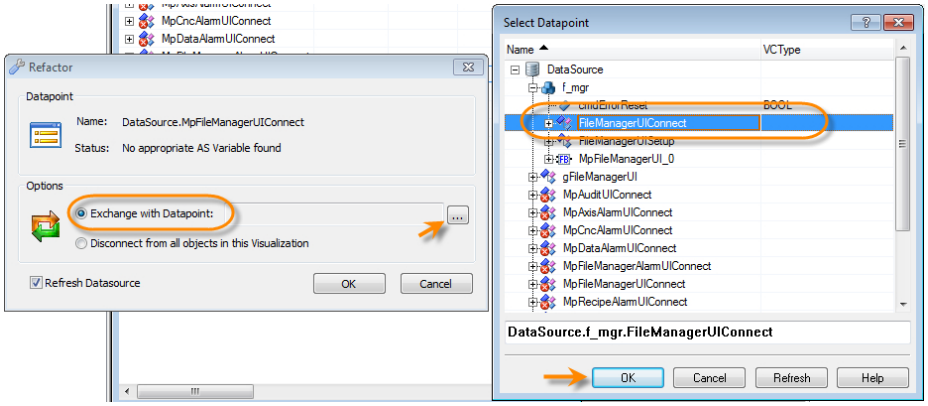


Figure 36: UIConnect structure connection using the Refractor function

5) Building and transferring the project

6) Test the file manager visualization

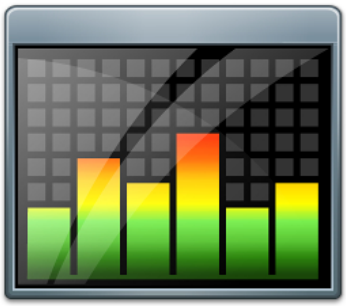
Visualization \ Visual Components VC4 \ Shared resources \ Data sources \

- Refreshing data points
- Replacing data points

Application Layer - mapp Technology \ VC4 templates

7 Summary

Elaborate and complex visualization concepts can be achieved through the connection of several controls and visualization resources. With the help of the tasks and exercises in this training module, your basic knowledge of Visual Components has been significantly expanded. Additional information, configurations and the combination of various controls allow many new concepts for designing a visualization application.



The instructions in this training module served to provide deeper understanding in addition to offering an overview of possible expansions in a visualization application. The references to the help system included in this module show where additional information can be found for expanding configurations.

Training modules

TM210 – Working with Automation Studio
TM213 – Automation Runtime
TM223 – Automation Studio Diagnostics
TM230 – Strukturierte Softwareentwicklung
TM240 – Kontaktplan (LD)
TM241 – Funktionsplan (FBD)
TM242 – Ablaufsprache (SFC)
TM246 – Strukturierter Text (ST)
TM250 – Speichermanagement & Datenhaltung
TM400 – Introduction to Motion Control
TM410 – Arbeiten mit Integrierter Antriebstechnik
TM440 – Motion Control: Basic Functions
TM441 – Antriebstechnik Mehrachsfunktionen
TM450 – ACOPOS Regelungskonzept und Einstellung
TM460 – Erstinbetriebnahme von Motoren
TM500 – Übersicht Integrierte Sicherheitstechnik
TM510 – Working with SafeDESIGNER
TM540 – Integrated Safe Motion Control
TM600 – Übersicht Visualisierung
TM610 – Arbeiten mit Integrierter Visualisierung
TM630 – Visualization Programming Guide
TM640 – Alarm System, Trends and Diagnostics
TM670 – Advanced Visual Components
TM800 – APROL Systemkonzept
TM811 – APROL Runtime System
TM812 – APROL Operatorverwaltung
TM813 – APROL XML Abfragen und Audit Trail
TM830 – APROL Projekt Engineering
TM920 – Diagnostics and service
TM923 – Diagnostics and Service with Automation Studio
TM950 – POWERLINK Configuration and Diagnostics

TM1010 – B&R CNC–System (ARNC0)
TM1110 – Integrated Motion Control (Axis Groups)
TM1111 – Integrierte Antriebstechnik - Bahngesteuerte Bewegungen
TM261 – Regelungstechnik mit LOOPCONR
TM280 – Condition Monitoring für Schwingungsmesstechnik
TM480 – Hydraulik Basis
TM481 – Valve-based Hydraulic Drives
TM482 – Hydraulische Servopumpenantriebe