Visual Components Advanced

TM670

Perfection in Automation

www.br-automation.com

Requirements

Training modules:	TM610, TM640
Software	Automation Studio 3 Coffee machine sample project
Hardware	Any (PC-based Automation Runtime Simulation (ARsim) , Power Panel)

TABLE OF CONTENTS

1 INTRODUCTION

The following tasks are intended to reinforce the participant's basic knowledge by providing practical challenges and additional information.

The tasks are divided into categories, which indicate the general area of use.

The difficulty varies with the particular tasks. Direct solutions will not be provided, because there are several ways to implement the tasks. However, references will be provided to the corresponding help files as well as step by step instructions aimed at directing the participants toward the right approach.

Learning by doing

1.1 Training module objectives

Existing basic knowledge will be expanded upon and strengthened by working through a variety of tasks independently.

You will learn ...

... how to solve tasks with a structured approach

... how to implement a variety of different solutions on your own

... a holistic view on the subject of visualization through a series of related exercises

2 DYNAMIC ELEMENTS

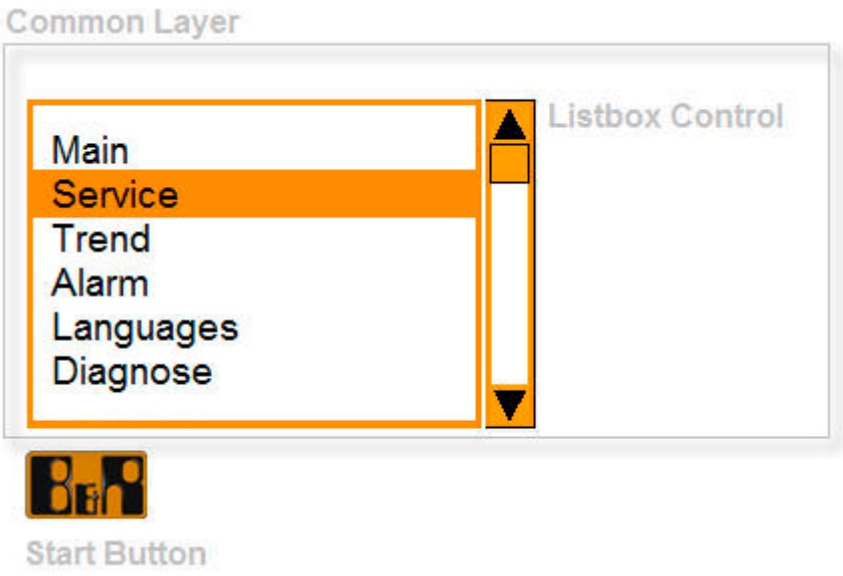
2.1 Creating dialog boxes

We will now implement a start button, similar to the Windows Start Button. Pressing this button will expand a menu list from which any page of the visualization can be selected.

A listbox is used for navigation / selection. Confirming the selection with the Enter key should cause that page to be opened. The menu will disappear only after pressing the Enter key.

Task: Creating a dialog box using a common layer and listbox

1) Common layers - Create new layer
2) Add listbox control to the created layer
3) All of the visualization pages should be available for selection in the listbox control (use the "ChangePageDatapoint" key action)
4) Use the status data point of the layer to control visibility
5) Add Start button to page
6) When pressing the start button, the key action changes the status data point of the layer



Overview image - Result

The listbox control provides a simple listing and selection of list items. Many functions can be performed quickly and easily by using the listbox index data point and the action buttons such as "ChangePage-DataPoint".

The status point can be used to set the visibility, as well as other effects. The majority of VC controls contain this status data point interface.

Visualization \ Visual Components VC4 \ Visualization resources \ Common Layers

Visualization \ Visual Components VC4 \ Control reference \ Listbox

Visualization \ Visual Components VC4 \ Control reference \ Button

2.2 Implementing a traffic light

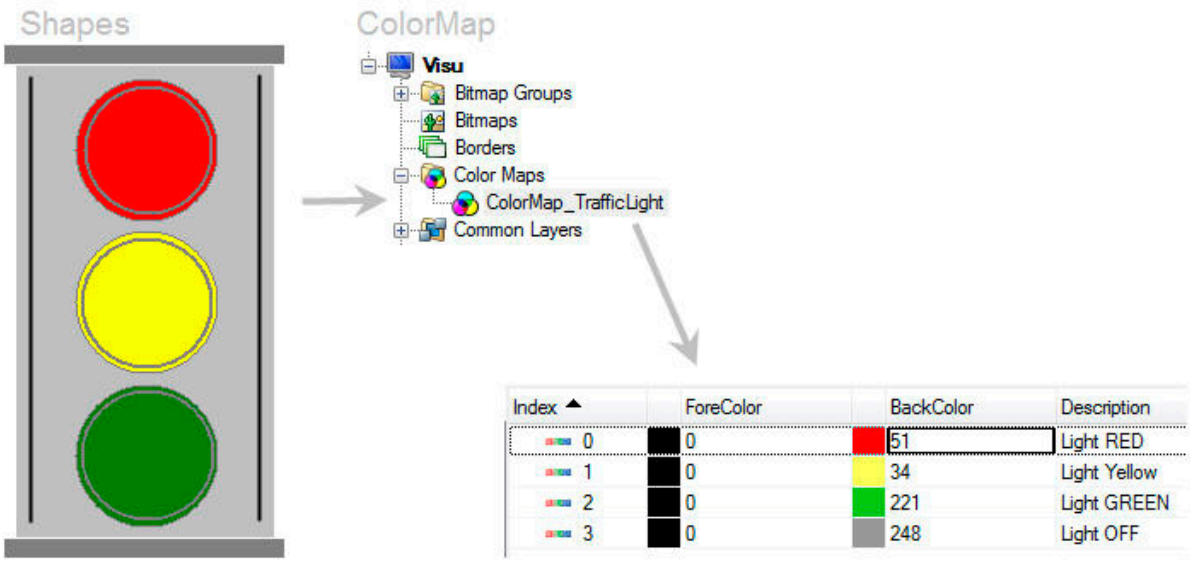
We will use the coffee machine sample project.

We will now implement and illustrate a traffic light as a visualization of the coffee machine status. It should turn red for the "Machine off" status, yellow for "Heating" and green for "Ready".

The Shape control is used for the shape and the color map function for dynamically changing the colors.

Task: Creating a traffic light using color table values and shapes

1) Draw a traffic light using different shape controls
2) Create corresponding color map
3) Assign color values to the various shapes
4) The color data point can be used to switch the color during runtime - establish connection with coffee machine states



Overview image - Shapes and color map

The advantage of color palettes is that individual color palettes can be compiled. The individual color indices can then be switched again during runtime. This makes it possible for various elements to change color by switching the index data point.

Visualization \ Visual Components VC4 \ Control reference \ Shape

Visualization \ Visual Components VC4 \ Visualization resources \ Palette

2.3 Creating dynamic elements

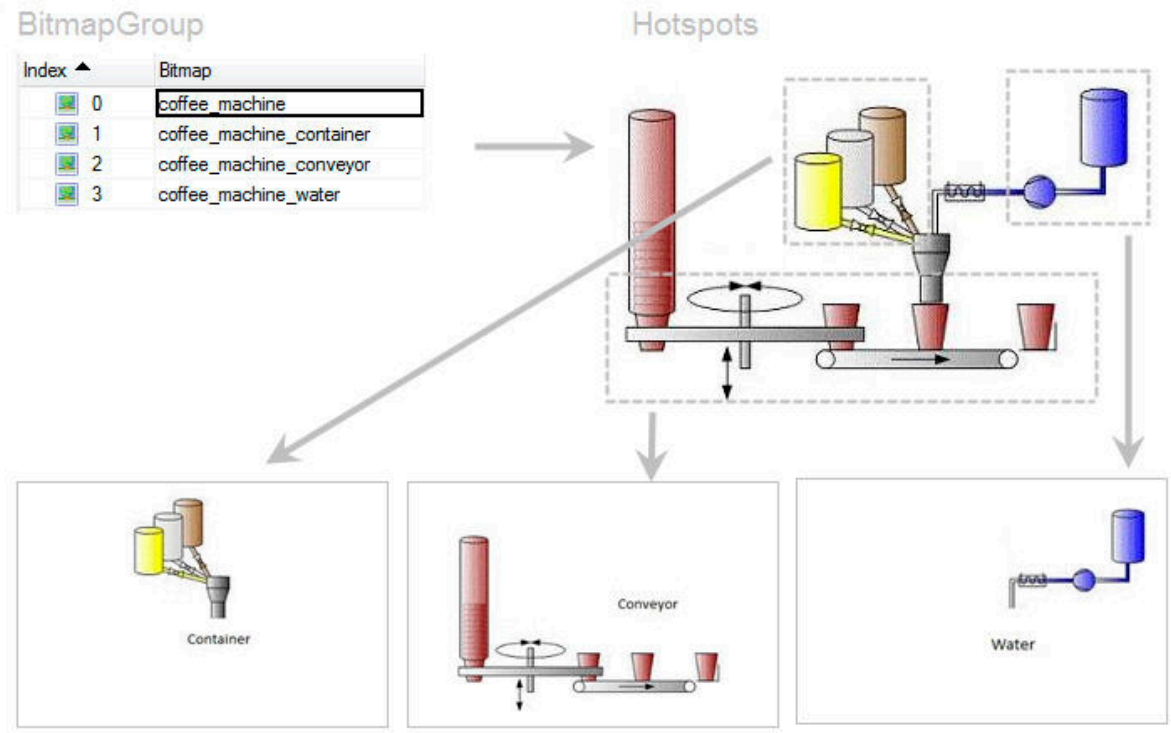
We will use the coffee machine sample project.

The overview image "coffee_machine" will be animated as follows:

Clicking on a hotspot on the overview image, for displaying the respective sub-image (conveyor belt, water tank, recipe container). Clicking again will return the user to the overview screen.

Task: Combining a bitmap group and animation using a hotspot

1) Bitmaps - Make several copies of the "coffee_machine" overview image and rename each copy (conveyor, water, container)
2) Edit the created bitmaps using an external graphics editor (right click "in External Editor") to emphasize the respective sub-image
3) Create bitmap group with the overview and sub-images (index data point)
4) Create graphic and assign bitmap group
5) Create hotspots with appropriate key actions to toggle to the respective bitmap index and place them over the sub-images of the overview image
6) Clicking on the sub-image hotspot will switch to the overview image on the respective image



Overview image - Bitmap group and multiple bitmaps

The multiple bitmap image function can be used to create simple image animations. This is based on multiple images that can be switched during runtime using an index data point.

These animated images can be made interactive by using the Hotspot control. A hotspot is an invisible button that can be assigned a function (key action).

When placing multiple hotspots on an overview image, different effects can be implemented to emphasize certain elements in the image.

Visualization \ Visual Components VC4 \ Visualization resources \ Bitmaps

Visualization \ Visual Components VC4 \ Visualization resources \ Bitmaps groups

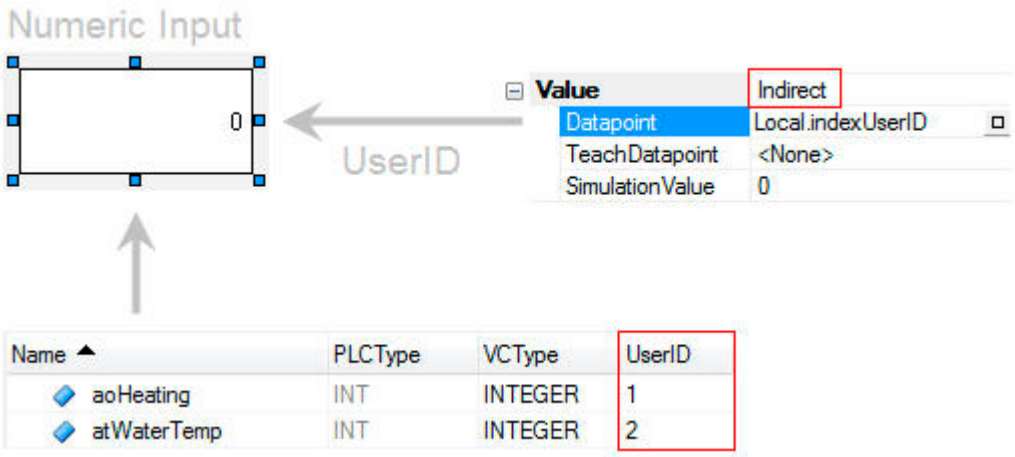
Visualization \ Visual Components VC4 \ Control reference \ Hotspot

2.4 Indirect value input

An additional user ID can be assigned in the data sources for each data point. This user ID can be associated with each input field via the "Indirect" interface. This function enables you to switch the data point where the value is being entered during runtime.

Task: Configuring the input field

1) Create two variables for input
2) Enter a different user ID in each of the data sources
3) Create input field (numeric input)
4) Change value from "standard" to "indirect"
5) A data point is now connected, which refers to the user ID and indicates the variable being accessed



Overview image - Numeric Input and UserID

The user ID feature is useful if you want to write several variables during runtime using just a few input fields.

Visualization \ Visual Components VC4 \ Control reference \ Numeric

Visualization \ Visual Components VC4 \ Shared resources \ Data sources \ Configuring data points \ UserID

The data point associated with the "Datapoint" value is the user ID. The value is then written to the data point that refers to the user ID.

3 INPUT AND OPERATION

3.1 Triggering a touch calibration

There are several ways to calibrate a touch trigger.

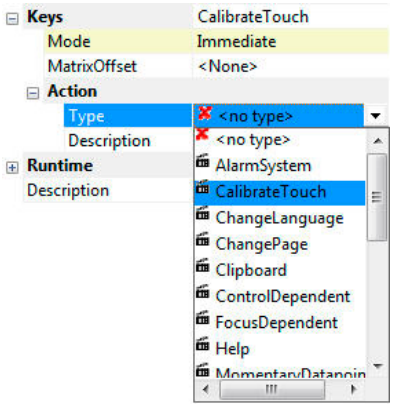
Buttons and keys (as well as USB / VNC keyboard) can be easily assigned using the "CalibrateTouch" VirtualKey action. Calibration is started when pressed.

The application can also be used to start touch calibration and to read calibration data. The Visapi Library contains several functional blocks for this.

We will take a closer look at assigning hot keys for a USB keyboard in the next example, "Configuring a USB keyboard".

Task 1 - Button for touch calibration

1) Insert a button
2) Create a new virtual key in the button's properties under "Keys"
3) Select the "CalibrateTouch" action under "Action \ Type"



Task 2 - Touch calibration during start-up with specific node number

1) Insert a new program
2) The node number can be read using a function block from the AsARCfg library
3) The touch calibration can be started using a function block from the Visapi library

Why calibrate the touch screen?

External influences such as temperature fluctuations or tight mounting of the panel can reduce the accuracy of the calibration.

In such cases, it is necessary to re-calibrate the touch screen. The calibration data is stored on the CompactFlash card and is retained until re-calibration or until manually deleted.

If the touch screen calibration has deteriorated to such an extent that for example it is no longer possible to actuate the touch calibration button, then the calibration method available via the system/application can be useful.

Visualization \ Visual Components VC4 \ Control reference \ Button

Programming \ Libraries \ Configuration, system information, runtime control \ AsARCfg

Programming \ Libraries \ Visualisation \ Visapi

Visualization \ Visual Components VC4 \ FAQ \ Input and touch screen operation \ Calibrating a touch screen

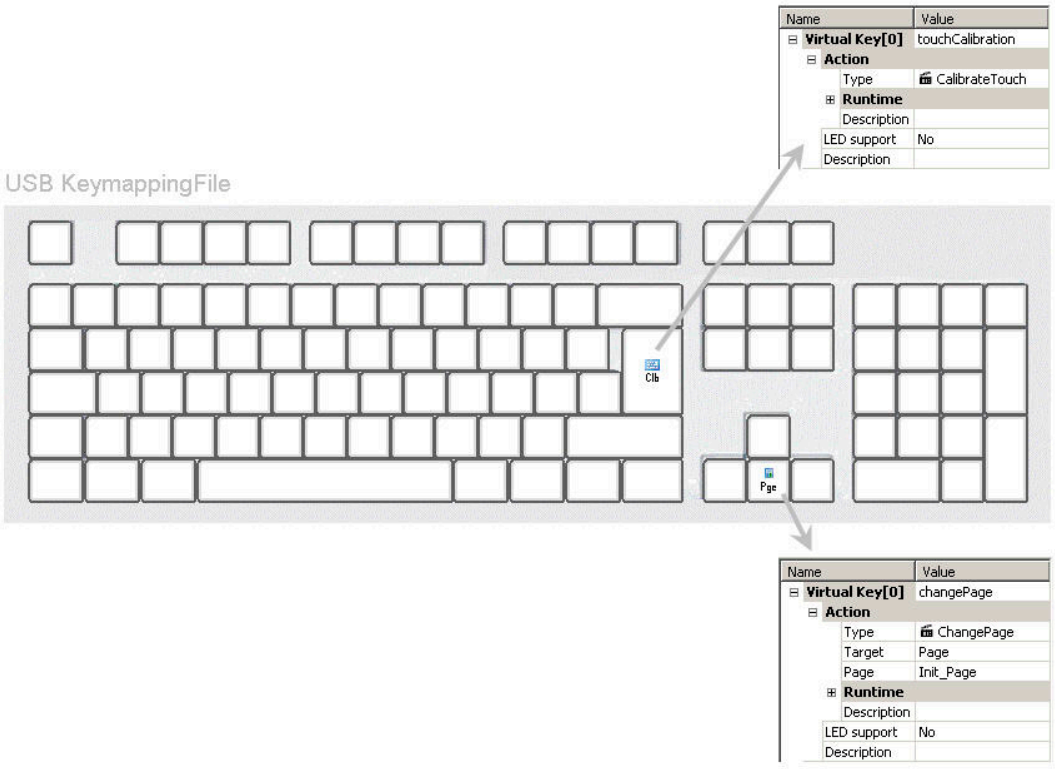
3.2 Configuring a USB keyboard

We will now configure and add a USB keyboard.

A separate key mapping file is provided for this in the form of a blank keyboard. The respective keys can now be linked with existing virtual keys to provide them with key actions such as change page, calibrate touch, etc.

Task: Assigning virtual keys to a USB keyboard

1) Connect a USB keyboard to a USB port on the PC
2) Open the VC Mapping table in the Configuration view and assign a visualization to the USB keyboard
3) Open the USB key mapping file in the Physical view
4) Create and assign virtual keys to the keyboard keys



Overview image - Sample USB keyboard layout

This illustrates the advantage of virtual keys. Depending on the application, these can be linked with software keys (buttons on the touch display), hardware keys (integrated display keys) or even external keyboards, like the USB keyboard in this example.

Configuring different key levels on the USB keyboard can allow multiple layers to be used with different functions. For example capital letters - the shift key can be used to change key levels if configured accordingly.

Keep in mind that the entries made on the USB keyboard will only appear in the respectively connected visualization application.

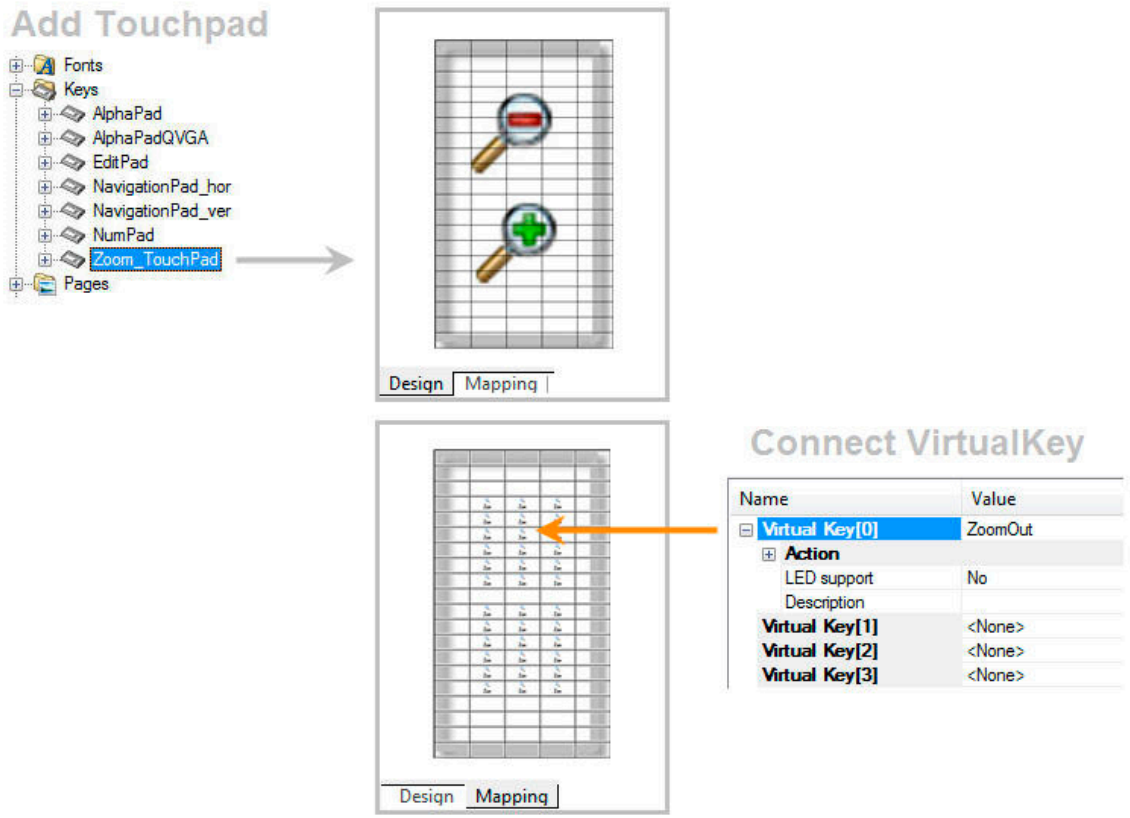
Visualization \ Visual Components VC4 \ Visualization resources \ Keys \ Mapping physical keys

3.3 Create touchpad for trend operation

In addition to the standard touchpads such as AlphaPad or NavigationPad, we will also be creating a touchpad with special functions that will allow you to implement a zoom in/out function for a trend.

Task: Creating a touchpad for trend functions

1) Open visualization object
2) Add a new touchpad to the resources under Keys (right-click -> Add TouchPad)
3) Insert 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) If there is an existing trend, or when a new one is created, set "Input" to TRUE and link the touchpad.



Overview image - Creating a touchpad

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

4 OPTIMIZING THE APPEARANCE WITH AN IMPROVED LAYOUT

4.1 User management with password protection

We will now set up a user management system. There are three different users with different access rights (password levels).

User 1 can only use the button. The input field is locked.

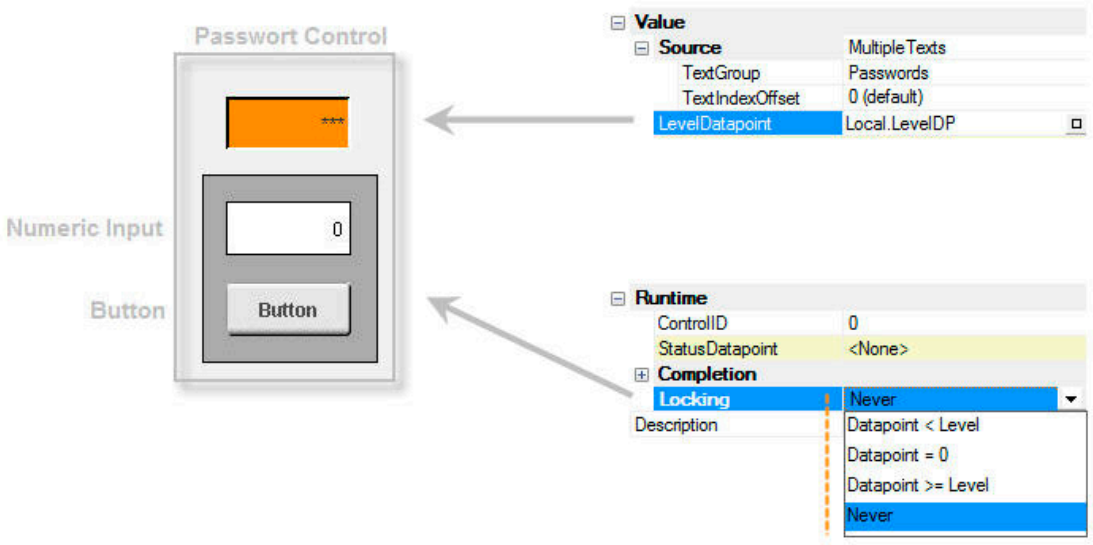
User 2 can only use the input field. The button is locked.

User 3 has access to and can use both controls.

This is implemented using the password control, in which users enter their respective password. Access to the subsequent controls is then unlocked or locked according to the password level.

Task: Configuring the password control

1) Create the password control, button and numeric input on the same page
2) Create a text group and enter the different passwords
3) Configure password control and read "LevelDataPoint"
4) Configure button and numeric input and set "locking" parameter



Overview image - Password Control

The password control is used for encrypted input of passwords that have already been defined and stored in a text group in the system.

The "LevelDataPoint" serves as a return parameter that returns the user's password level if the entered password is also present in the text group.

The "Locking" function, which is included in most of the controls, is an easy way to lock or unlock the respective control (Button, Input etc.).

Visualization \ Visual Components VC4 \ Control reference \ Password

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

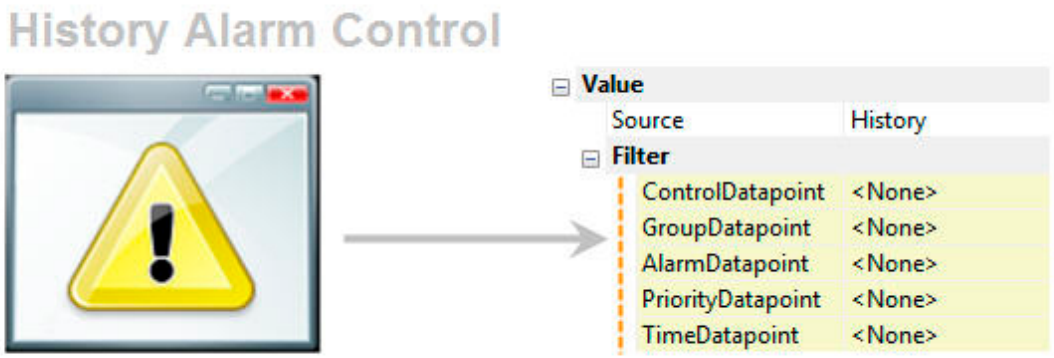
4.2 Filters in the alarm history

We will use the coffee machine sample project.

An alarm control has already been added with history configuration. The task is to filter this alarm list using filter control bytes. (Filter / Control Data Point)

Task: Filtering the historical alarms

Configure settings for the filtering properties of the alarm control element



Overview image - Filtering options

Filter options are often a good way to improve the organization and layout of the alarm list. You can filter by alarm, group, priority and time.

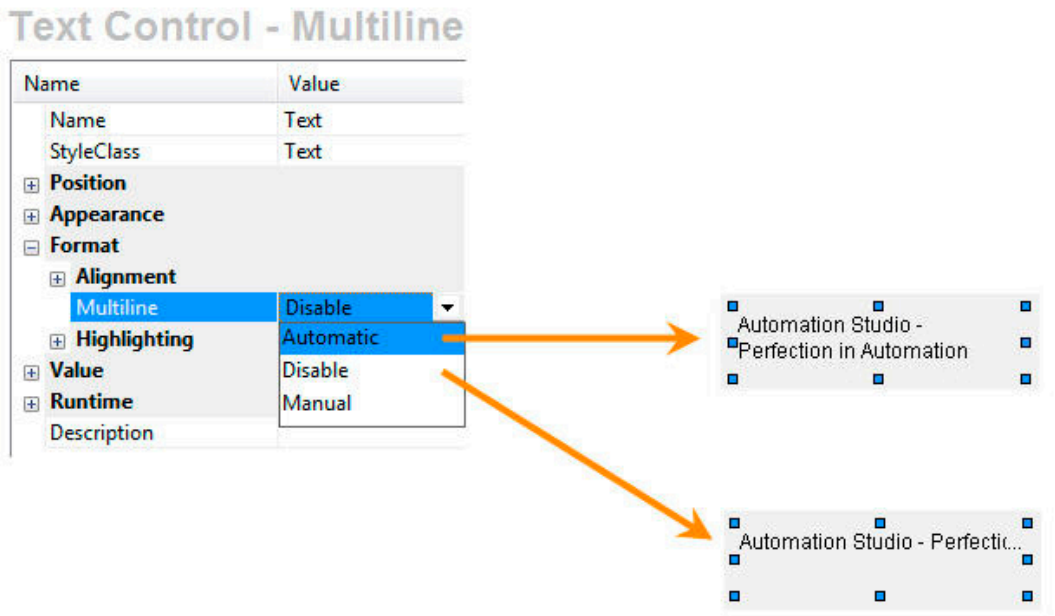
Visualization \ Visual Components VC4 \ Shared resources \ Alarm system \ Viewing alarms during runtime \ Filtering alarms

4.3 Displaying multi-line texts

The text control is used for outputting static or dynamic text. The "Multiline" property allows text to be displayed on multiple lines.

Task: Configuring the text control

1) Insert text control
2) Display text
3) Try out the different "Multiline" options



Overview image - Text control settings - Multiline

There are different ways to display the text in a text control. Line breaks can be initiated automatically or manually. This behavior can be defined using the "Multiline" property.

Visualization \ Visual Components VC4 \ Control reference \ Text \ Format \ Multiline

5 PROGRAMMING

5.1 Reading and storing the alarm history

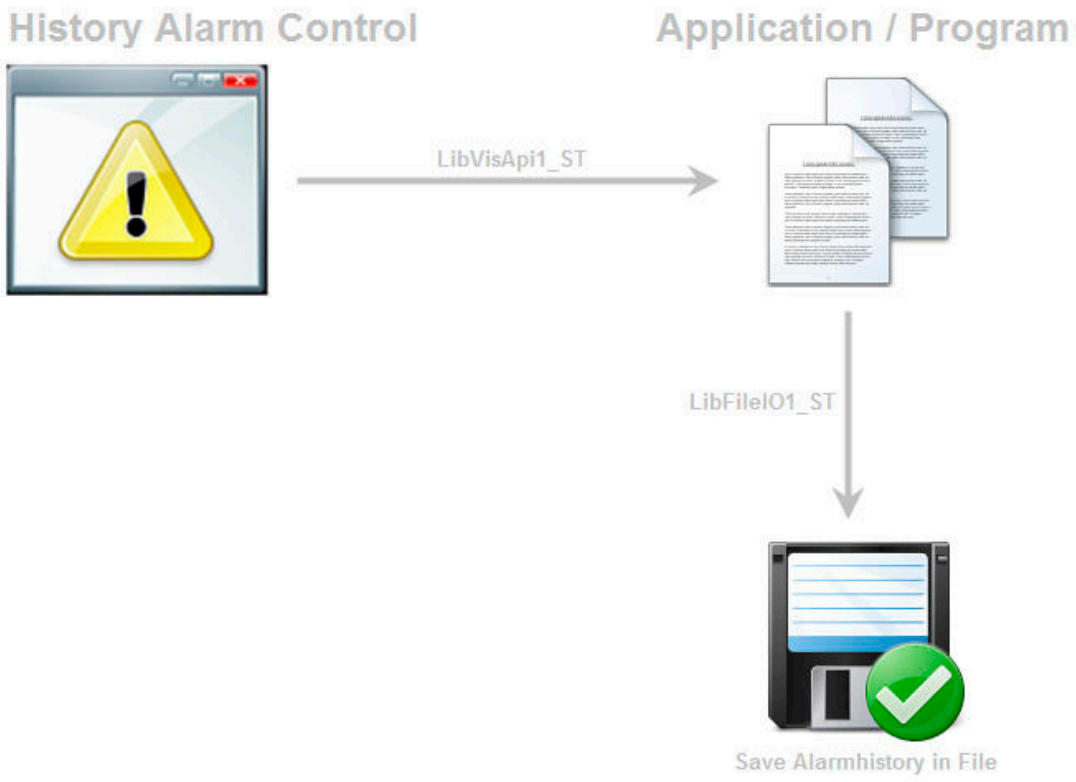
We will use the coffee machine sample project.

An alarm control has already been added with history configuration. The aim is to read the individual alarms using the Visapi library and to save the the information to a file using the FileIO library.

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

Task: Using and adjusting library examples

1) Insert library examples "LibVisApi1_ST" and "LibFileIO1_ST"
2) Adjust program components for reading and storing the alarm history in a file



Overview image - Reading and storing alarm history read in file

The advantage of using library examples, is that they can be easily added and used in Automation Studio. Minor adjustments make it easier to create and implement custom applications.

Visualization \ Visual Components VC4 \ Control reference \ Alarm

Programming \ Examples \ Libraries \ Visualization \ Alarm system

Programming \ Libraries \ Visualization \ Visapi \ Examples

Programming \ Libraries \ Data access and data storage \ FileIO \ Examples

5.2 Graphical elements with the DrawBox control

The Visapi library is a collection of programming functions. Among other things, the library contains graphics functions that are created dynamically in the application instead of being static in the visualization.

First part of the exercise is to output different graphics directly to the visualization (lines, rectangles, circles, bitmaps, text).

In the second part of the exercise, we will use the DrawBox control, which acts like a "window in a window" for displaying the graphics that we programmed earlier.

Task 1 - Outputting text and graphics to the visualization

1) Create a program
2) Establish a connection to the visualization object using the Visapi library
3) Reprogram several graphics/text functions provided by the Visapi library for outputting to the visualization

Other objects/controls are frozen on the page as long as the Visapi library is actively (cyclically) accessing the visualization.

Task 2 - Graphics functions in connection with DrawBox

1) Add DrawBox to page
2) Expand the program from Task 1 - Establish connection to DrawBox for outputting the graphics/text functions

Task 3 - Graphics functions in connection with multiple DrawBoxes

1) Insert a second DrawBox on the same page
2) Expand the program from Tasks 1 + 2 - Output the same graphics/text functions to both DrawBoxes

The advantage of the DrawBox control is in its "window in a window" functionality. It has its own coordinate origin, which no longer refers to the absolute zero point of the screen.

If a graphic element that was implemented using Visapi is shifted, then the application code would have to be modified without using a DrawBox control (changing x / y position).

Programming \ Libraries \ Visualisation \ Visapi

Visualization \ Visual Components VC4 \ Control reference \ DrawBox

The library examples for DrawBox and Visapi library contained in Automation Studio can be used for these tasks.

5.3 Screenshot Library

Use the VCScrsht library to create a screenshot of the visualization page currently being displayed. The screenshot is triggered by clicking on a button that starts the appropriate application. The resulting bitmap will be stored on the CompactFlash card. The screenshot can then be copied from the controller to the PC via an FTP connection.

Task: Working with the VCScrsht library

1) Create file device
2) Create corresponding program using VCScrsht library
3) Add button for triggering screenshot in visualization

The VCScrsht library consists of three function blocks. The library example "LibVcScrSht_ST" will help you to create a screenshot of the visualization page. The final screeshots can then be stored on the CompactFlash card or any other storage media (e.g. USB) using FileDevice.

Programming \ Libraries \ Visualisation \ VCScrsht

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

6 SUMMARY

The topic of visualization is more easily understood by combing the individual controls and functions for a clearer overview.

Learning by doing

Providing general, step-by-step guidance toward a solution without strictly defined end results encourages the trainee to contemplate the tasks and to think more on their own as they work toward a solution. This process is accompanied by the Automation Studio online help, which provides further guidance with numerous examples.

TRAINING MODULES

TM210 – The Basics of Automation Studio
TM211 – Automation Studio Online Communication
TM213 – Automation Runtime
TM220 – The Service Technician on the Job
TM223 – Automation Studio Diagnostics
TM230 – Structured Software Generation
TM240 – Ladder Diagram
TM241 – Function Block Diagram (FBD)
TM242 – Sequential Function Chart (SFC)
TM246 – Structured Text
TM250 – Memory Management and Data Storage
TM261 – Closed Loop Control with LOOPCONR
TM400 – The Basics of Drive Technology
TM410 – ASiM Basis
TM440 – ASiM Basic Functions
TM450 – ACOPOS Control Concept and Adjustment
TM460 – Starting up Motors
TM500 – Basics of Integrated Safety Technology
TM510 – ASiST SafeDESIGNER
TM540 – ASiST SafeMC
TM600 – The Basics of Visualization
TM630 – Visualization Programming Guide
TM640 – ASiV Alarms, Trends and Diagnostics
TM670 – Visual Components Advanced
TM700 – Automation Net PVI
TM710 – PVI Communication
TM711 – PVI DLL Programming
TM712 – PVIServices
TM810 – APROL Setup, Configuration and Recovery
TM811 – APROL Runtime System
TM812 – APROL Operator Management
TM813 – APROL XML Queries and Audit Trail
TM830 – APROL Project Engineering
TM890 – The Basics of LINUX

www.br-automation.com

TM670TRE.25-ENG / V0.1