

# Visualization Programming Guide

## TM630



Perfection in Automation  
[www.br-automation.com](http://www.br-automation.com)



## Requirements

Training modules: TM600 – The Basics of Visualization

Software: -

Hardware: -

---

## Table of Contents

1. INTRODUCTION	4
1.1 Training guide objectives	5
2. ROAD MAP	6
2.1 Project phases	6
3. SPECIFICATION	8
4. SOFTWARE DESIGN	11
4.1 The default project	11
4.2 The structure of a visualization	12
4.3 The operating concept	22
4.4 Variables and data points	24
4.5 Visualization runtime behavior	27
4.6 The Visualization Programming Interface	28
4.7 Screen pages for service and commissioning	30
4.8 Data and Data Management	31
5. INTEGRATING A VISUALIZATION	35
6. PROJECT MAINTENANCE AND ORGANIZATION	36
6.1 Changes to the project	36
6.2 Software distribution and storage	37
6.3 Turning the project over to the customer	37
6.4 Documentation	38
7. SUMMARY	40

## 1. INTRODUCTION

At some point or another, we have all been faced with the same problem – "where should I start with the new project"?

This is especially the case when there are no project guidelines or the guidelines have not yet been specified. This training module offers some advice, helping the user – mostly for a visualization project – to overcome the initial obstacles and to save time and expenses in the end.

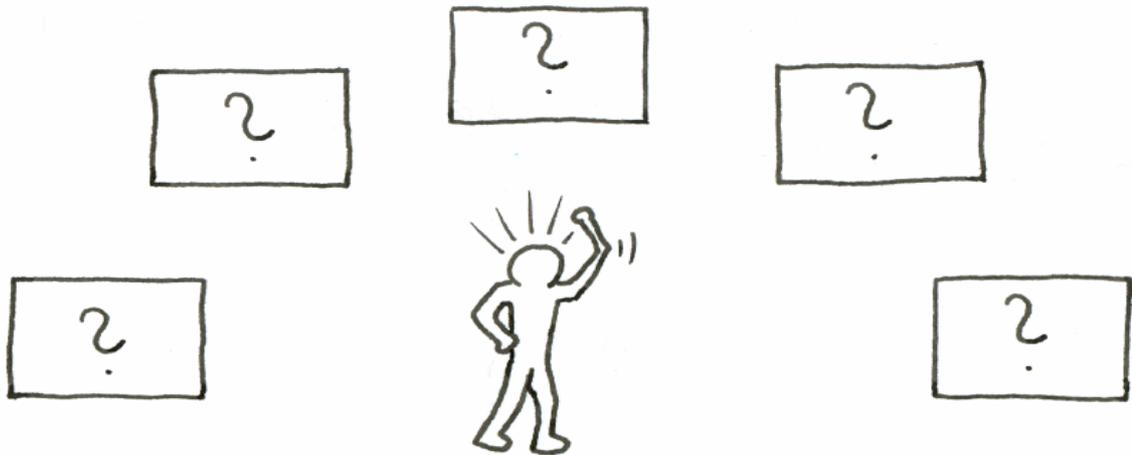


Fig. 1: Visualization Programming Guide

Perhaps even a few of the experts will find some useful information about programming a new visualization project.

When creating a visualization application, you must put yourself in the shoes of the visualization's target group (operator, service technician, advanced programmers) and understand the task-oriented process. Only then is it possible to find the optimal solution for whatever demand comes your way.

## 1.1 Training guide objectives

You will be able to design a more effective visualization project.

This should help prevent the "hello world" programming style whereby the programmer jumps right into programming without a solid plan or sufficient knowledge.

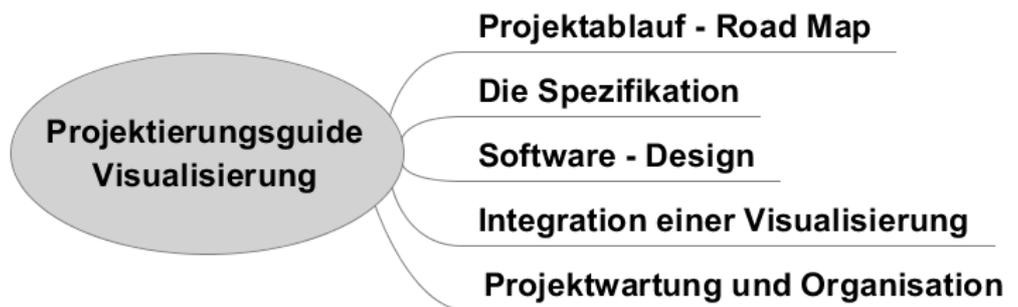


Fig. 2: Objective

## 2. ROAD MAP

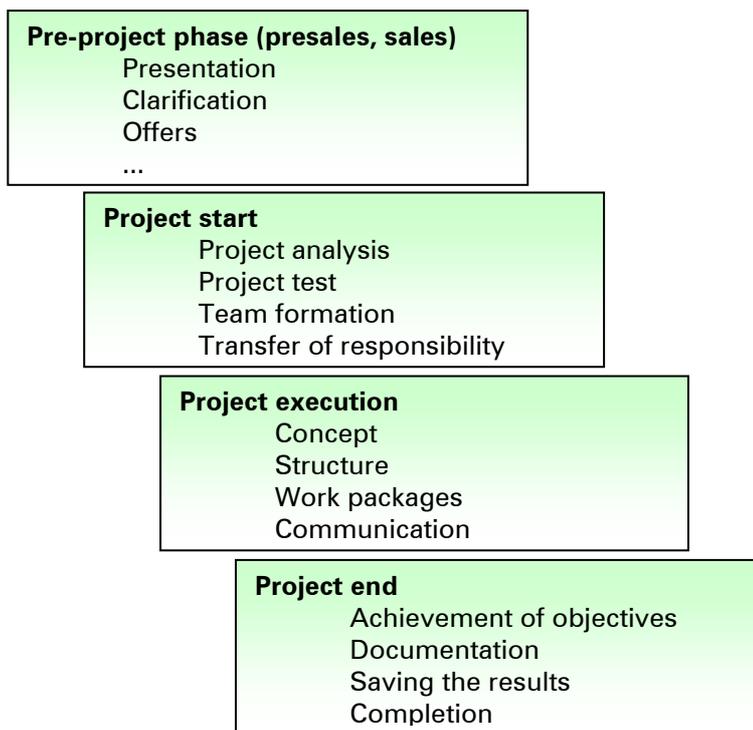
A road map guarantees that the project is carried out in an organized manner and that nothing is overseen.

**"Of course a visualization application must be finished yesterday because the machine was supposed to be operational the day before yesterday".**

### 2.1 Project phases

The customer's requirements must first be clarified before starting to execute the project.

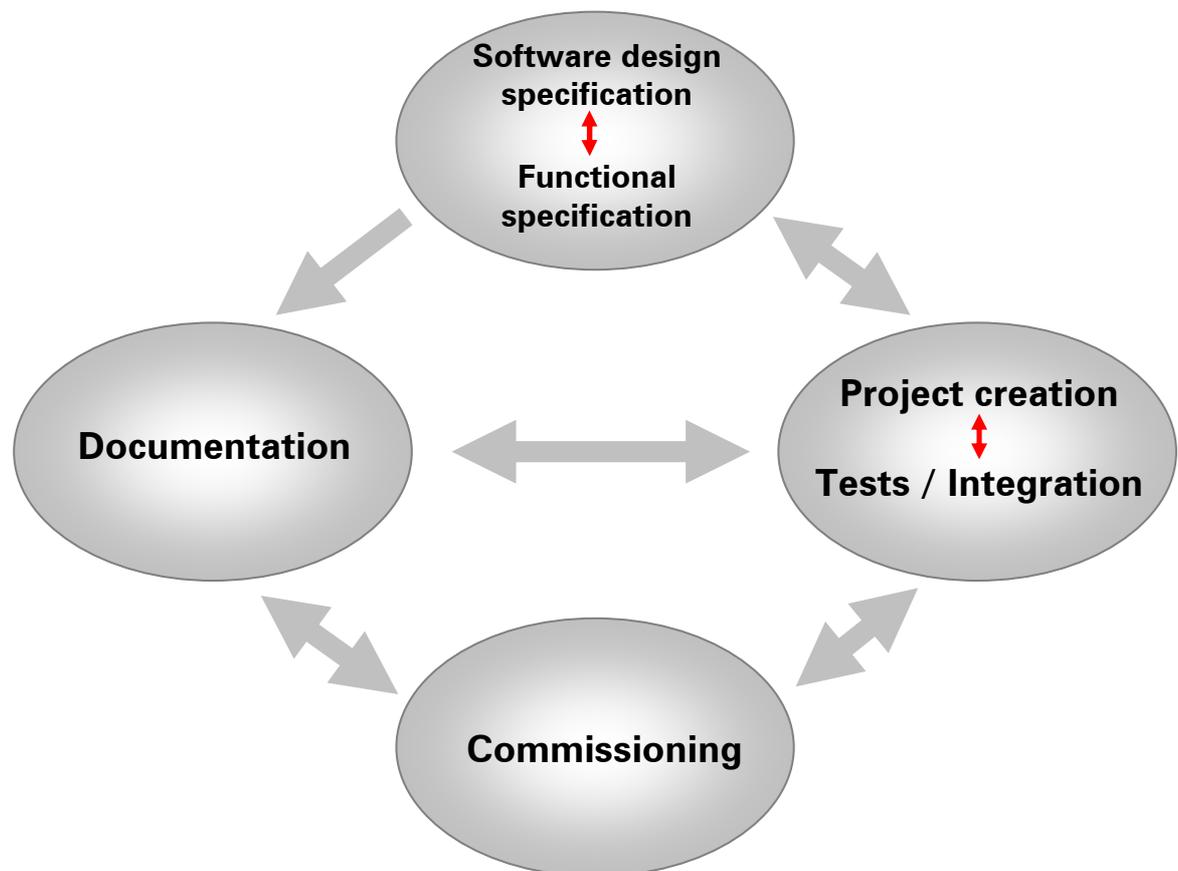
The following image offers an overview of the road map.



This training module deals with the project execution phase.

The project execution phase can be roughly divided into four sections, with considerable interaction between the individual sections:

- Creation of a software design specification ⇔ Assessment of expenditure ⇔ functional specification
- Configuration of the images, data management, etc.  
Tests and integration of the project versions
- Machine implementation
- Documentation



## 3. SPECIFICATION



Regardless of the visualization being used, the basis is determined by the customer's requirements and the resulting specifications or software design specification.

Expenditure of time [%] ACTUAL		Expenditure of time [%] TARGET	
Specifications	0 - 20	Specifications	30
Project creation	80 - 100	Project creation	20
Test	20 - 0	Test	50

As illustrated in this table, 1/3 of the time available for the project should be used for creating the specification or the software design specification. This results in a shorter project configuration time because fewer changes must be made later.

Unfortunately, the reality is always a little different because this time is usually not enough for "normal" project configuration. The programmer should consider whether or not an elaborate specification really saves times in the project configuration phase.

When creating the specification, a middle ground should be found between the necessary information and the programmer's freedom. You should avoid having to define every single bit.

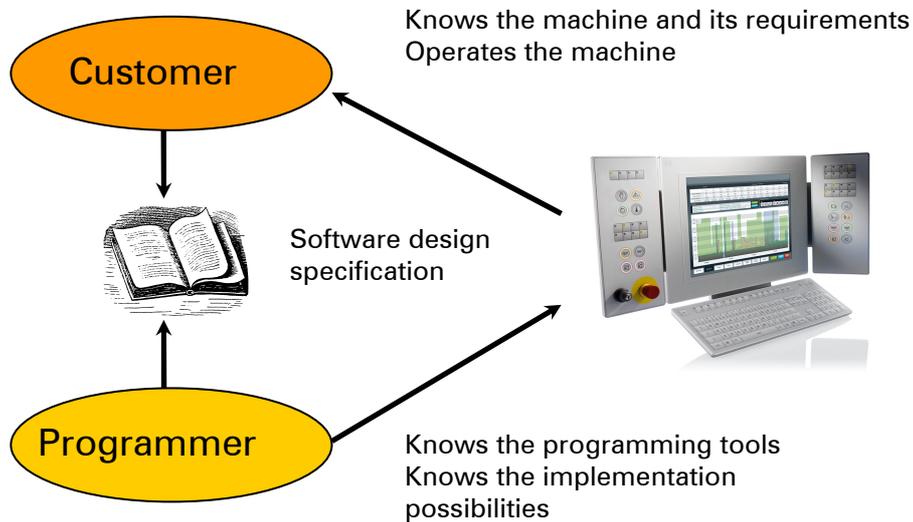
**The following questions should be answered when establishing the specifications for the visualization:**

- Which type of visualization should be used? (price category, runtime costs)
- Which visualization to use (embedded, remote visualization, etc.)
- Connection to higher level systems
- Which standards and safety functions are required?
- Data exchange, data formats and evaluation – also determined by the hardware being used
- Who will be operating the visualization application?
- What needs to be operated?
- Which information is important?
- Which operating concept should be used? (Tasten, Touch, ...)
- How should the pages be structured? Which elements should be displayed?
- What is the main page, which and how many sub-levels are necessary?
- Which colors and shapes should be used?
- Navigation in the pages
- Navigation between the pages
- Who is permitted to operate which pages? (Password levels)
- How will the pages be grouped together (page changes, menus)
- What fonts and font sizes are required?
- What languages are needed?
- What texts and graphics will be used?
- Password management

This information is used to create a foundation for the visualization.

A few samples must be created and presented to the customer before starting to setup and configure the pages. The sample chosen by the customer is then used as the template for the project. The customer must also be aware that any changes requested further into the project will require every page created to that point to also be re-edited.

It is necessary that the specification is always determined in cooperation with the customer because the user or programmer is familiar with the visualization software and the programming environment for creating the visualization.



This is the only way to satisfy and implement the customer's requirements with the tools being used.

The programmer should already be thinking about the project visualization when putting together the software design specification.

If the programmer is still learning how to use the tool while creating the project, then he cannot yet estimate if various requirements will be able to be met with the visualization software being used.

## 4. SOFTWARE DESIGN

The statement "**it just sort of happened**" is common to projects that were started at the programming stage instead of on paper.

Changes "thrown together" after the initial project setup are more difficult to implement and maintain than those made at the beginning while structuring the project.

### 4.1 The default project

A new project typically starts with a blank screen page. All of the necessary components such as bitmaps and texts must be added to the project afterwards. However, this also means that each project is structured differently and that elements are added to different positions each time.

Using a default project or template makes it possible to always start on the basis of the same project structure.

It does not matter whether each project is similar or altogether different – a good basis helps save time when starting a new project.

#### Elements of a default project

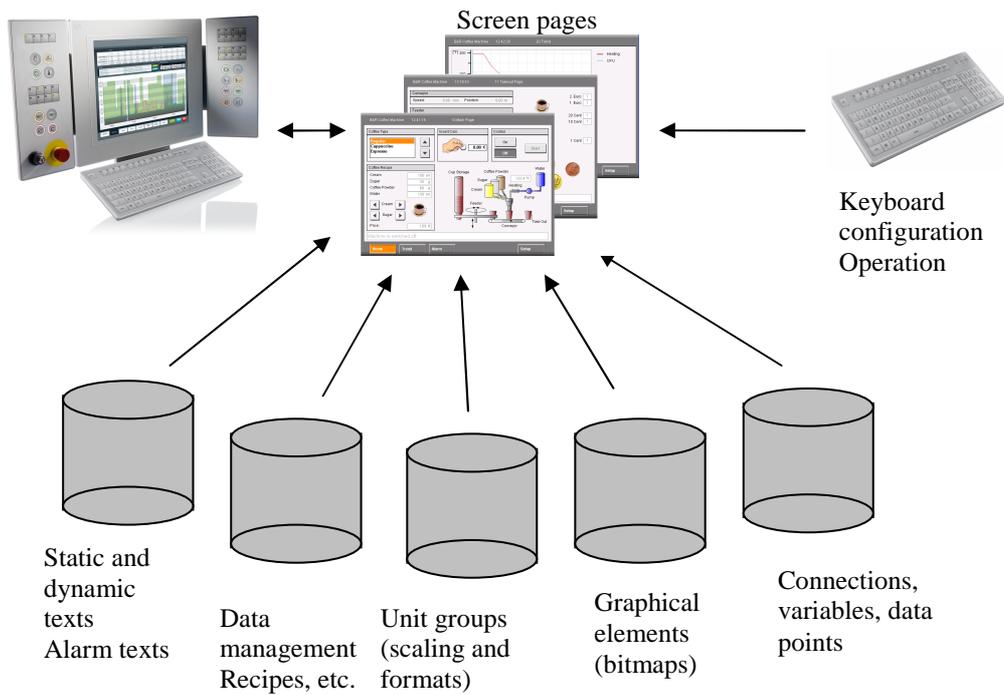
- Keyboard and touch configuration for numeric and alphanumeric entries
- Unit groups (scaling and formats)
- Bitmaps and bitmap groups
- Fonts and font sizes
- The default page, header, footer, templates
- Styles

Once the basis for a "New project" has been established, it should be used each time a project is started. All of the elements for the default project must be approved by the customer.

### 4.2 The structure of a visualization

A visualization project is divided into different project components. The distribution and configuration of these components is generally different for each visualization.

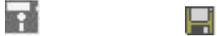
Possible components of a visualization:

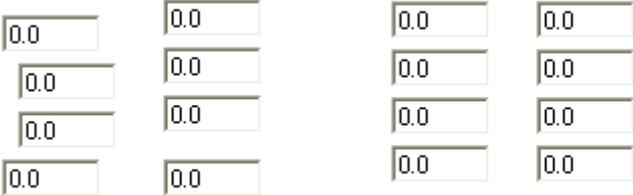
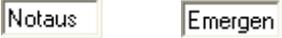


These components can be expanded as needed. The functions not contained in the visualization can be implemented with additional programming.

### 4.2.1 Image design

Screen pages make up the main part of a visualization. A few rules must be taken into consideration when designing a page:

Element	Comment
Font	The number of fonts should be kept to a minimum. Emphasis can be placed on objects using different font sizes (small, medium, large). Keep in mind that the visualization must be visible from a few meters distance.
Font	The texts should be easy to read, and not just " <i>pretty</i> ".
Font	If possible, the same font should be used for all of the languages in the visualization.
Graphics	A graphic or symbol can often say more than text can. Symbols do not need to be translated either. 
Graphics	If used on multiple pages, symbols should also have the same appearance. 
Graphics	Use standardized colors (e.g. green for OK., blue for cold, red for an error, warm or danger).
Graphics	Use commonly known symbols
Graphics	Keep in mind that operating personnel may be color blind. If so, they would not be able to differentiate between certain colors. The needs of all of the operators can be met by using a dynamic color configuration, which can be configured while the system is running.
Operation	The size of the input fields and buttons should be large enough that they can be easily operated by the operating personnel. You should also keep in mind that the operating personnel might be required to wear gloves.
Operation	Elements that do not contain any functions or for which the user does not have permission cannot be displayed on the page. These elements should be hidden or indicated with a "Disabled" status.
Structure	Avoid cluttering your pages. Important information on the page should be visible at first glance.
Structure	Identical functions on different screen pages should also be placed at the same position.

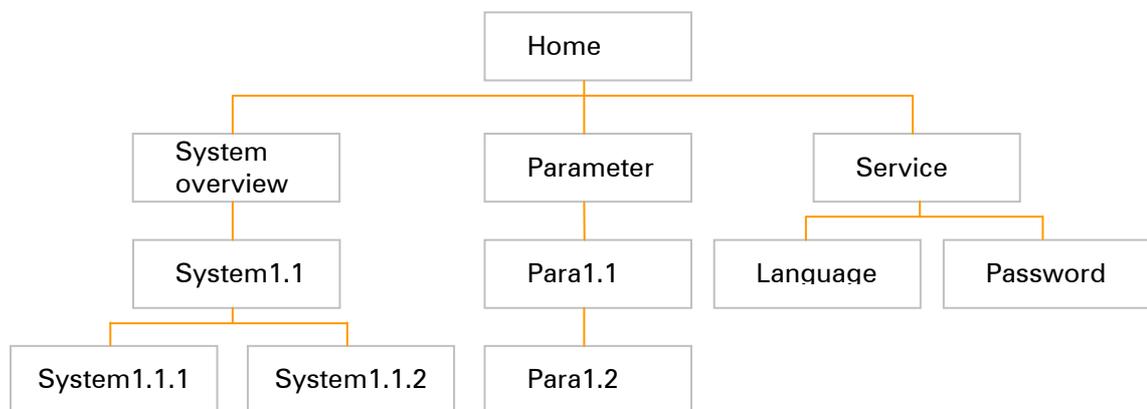
Structure	<p>Attention must be given to the alignment of the page elements.</p> 
Structure	<p>Establish a navigation strategy via key operation (cursor control, screen change) and apply this strategy equally to all pages.</p>
Structure	<p>If machine options are hidden, then attention must be given to the distribution of the remaining page elements. The page should not contain any large "holes".</p> <p>Different screen pages might have to be configured for the options. If possible in the visualization, dynamic page elements can also be placed.</p>
Text	<p>If texts are displayed in multiple languages then you must make sure that the maximum text length for a language also matches the length of the text field.</p> 

#### 4.2.2 Page branching and grouping

Once you know the content of every screen page, they can be put into groups. The groups can be based on the following structures:

- Process sequence
- Operating levels
- Service and maintenance levels

A tree structure can be used to determine the menu design and page changing on paper.



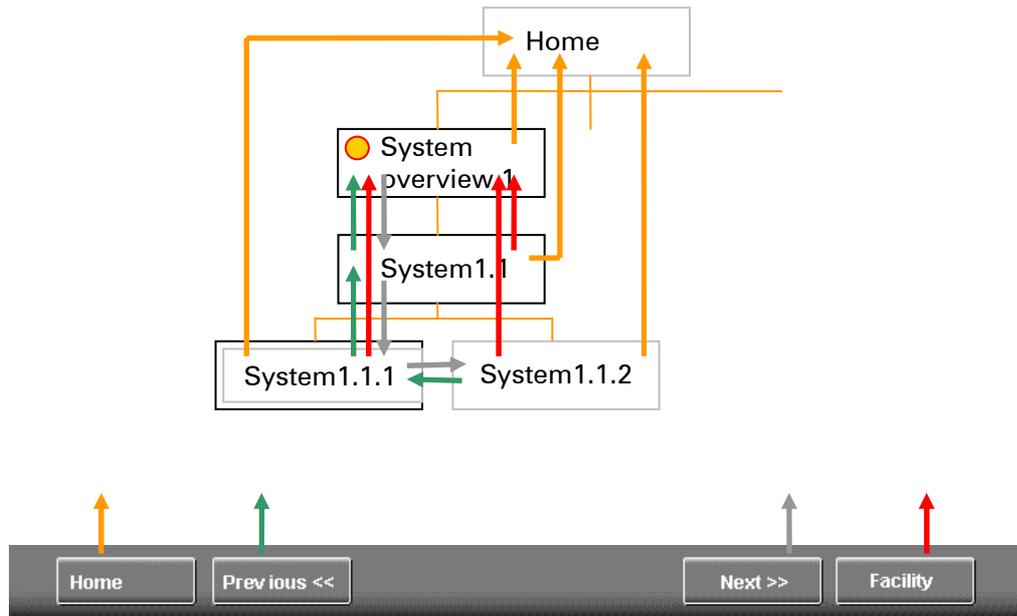
In this example, the page grouping in the project page list could look something like this:

```

0000_StartPage
0100_SystemOverview
0110_System1.1
0111_System1.1.1
0112_System1.1.2
0120_System1.2
0200_Parameter
0210_Parameter1.1
0220_Parameter1.2
0300_Service
0310_Language
0320_Password
  
```

The following rules must be followed when creating the page branching:

- A page should always be called up from the same position (touch button on the screen page or page changing key) on a screen page.
- When changing pages to a lower level, the user must be able to return to the main level without having to navigate back through several pages.



In this example the navigation between the page levels is displayed (starting at level "1" ●) using different colored arrows.

- Red arrow: Level "1" can be reached from any lower level page.
- Green arrow: From the lowest level, this arrow brings you back within the same level. From the first page of a lower level, this arrow brings you to the next level up.
- Gray arrow: In the same page level, this arrow brings you to the next page. At the end of a level, this arrow brings you to the first page of the next level down.
- Blue arrow: This arrow brings you back to the start page from any level.

There are of course many different concepts for a page change. This example describes only one possibility for an effective page changing sequence.

**Page grouping suggestion:**

- Assign group numbers before the page names.
- Space should be left between the group numbers so that pages can be added to a group afterwards.
- Display the page level on each page. This allows you to easily identify which level is currently being displayed.

**Changing a page:**

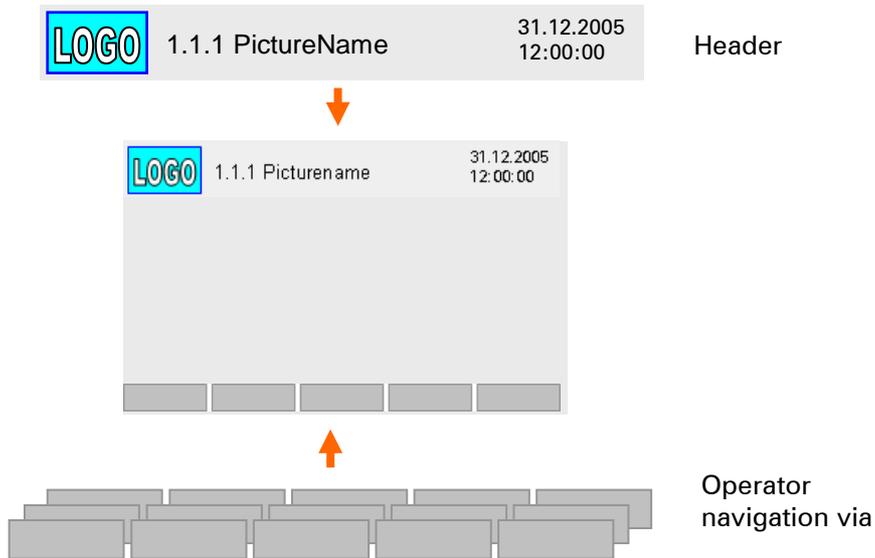
Generally, a page is changed using the keyboard or a touch device. The destination page should always be configured directly on the key or touch action. We do not recommend changing pages by writing a control variable, because the control program would have to be changed each time the page numbers are changed.

Changing a page using a variable is necessary either when certain actions that are initiated by the control program (e.g. when an alarm occurs) or when a page navigation with password is used (page is changed after the password has been checked).

Provisions in the visualization must also allow this page change to be delayed (e.g. if the user executes an action on a certain screen page, which cannot be interrupted).

## 4.2.3 Layering

The layering method allows you to place corresponding page elements on a layer and to use these elements as often as necessary. Once again, there are differences in the configuration, functionality and usage of the different visualization packets.



### Advantages of the layering method:

- Image information used on multiple pages only has to be created one time.
- Changes made afterwards only have to be performed at one position.
- Depending on the configuration, layers can be made visible or invisible during runtime.
- The layering method can be used to implement individual dialog boxes.
- Controlling the visibility of machine options.

Layers should definitely be used as long as they are supported. To do this, you must decide which screen information belongs in the different layers before drawing up the screen pages.

#### 4.2.4 Text in a visualization

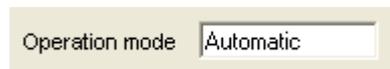
A visualization is made up of static and dynamic texts.

Static texts are a fixed part of a screen page such as a description text for a display element.



Static text example: Password Level

Dynamic texts change the actual text according to variables connected to the text element.



Dynamic text example: Operation mode

If possible, static as well as dynamic texts should be combined in easy-to-manage function-related text groups.

Advantages:

- Texts used on different screen pages only have to be created once.
- Text changes made afterwards only have to be performed at one position.
- Texts for different languages only have to be translated one time respectively (cost saving).

#### Language-dependent texts

If possible, texts from the same language should be managed in separate text files.

Advantages:

- Only the files for the language used during project setup and the languages for translation have to be translated. This makes it possible to translate the text for multiple locations just once.
- Only the languages which have been ordered can be delivered with the visualization. Additional languages can be also be added as an option.

### 4.2.5 Graphical elements

#### **Managing graphic elements**

To improve the overview and organization of the visualization, similar graphical elements, such as bitmaps, should be grouped together in logical groups or in separate subdirectories.

#### **Dynamic graphic objects**

Dynamic objects or constantly changing elements in a visualization should be kept to a minimum. Such objects can divert the operator's attention from the more important information on the screen.



If there are multiple objects on a screen that change cyclically (e.g. blinking alarm symbol and cursor), then make sure that all of the elements change at the same blinking frequency.

### 4.2.6 Using standards

An extensive amount of standard output dialog boxes (MessageBox, File Open Dialog, etc.) are available for the programmer especially when programming in Windows. However, a few things must be taken into consideration:

- Texts from standard dialog boxes are always displayed in the language of the libraries installed. Generally, this language cannot be switched while the system is running.
- The button sizes in these dialog boxes are not always dimensioned sufficiently.
- It is not always possible to output UNICODE characters.

To avoid these problems, all of the dialog boxes required in the visualization should be put on separate screen pages or screen levels.

#### 4.2.7 Each element has its own name

Each element used in a project is given a default name when added to the visualization at the project or component level. This name indicates the characteristics of the element but if several identical elements are used then it becomes difficult to differentiate each one.

This also makes it more difficult for a second person to work on the same project.

Working with default names	Working with descriptive names
Visualization Picture1 InputField_1 OutputField_1 Picture2 InputField_1 InputField_2 OutputField_1 Button_1	Vis_MainTerminal 000_StartPage txtSetPassword txtActPasswordLevel 010_Parameter txtSetCoolingDelay txtSetCoolingTemp txtCoolWaterTemp cmdPage_AlarmHistory

As seen in this example, the label names are needed in order to be able to identify what each of the elements is used for.

Visualization objects must always be given meaningful names – coding rules specified by the customer are used when programming the controller.

Working with default names	Working with descriptive names
Private Sub <b>Command1</b> _Click() End Sub	Private Sub <b>cmdStartHeating</b> _Click() End Sub
Private Sub <b>Command2</b> _Click() End Sub	Private Sub <b>cmdStopHeating</b> _Click() End Sub

### 4.3 The operating concept

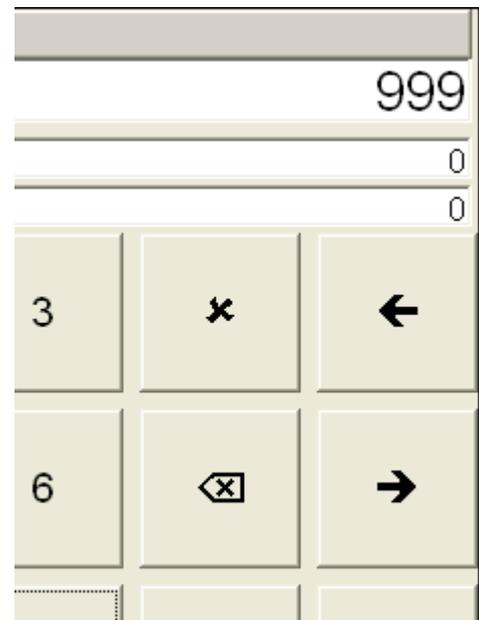
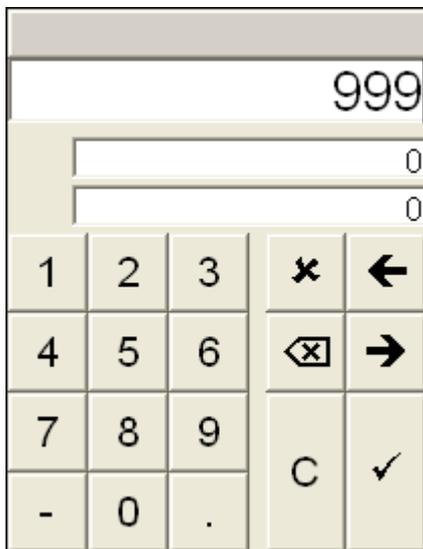


The visualization hardware being used determines how a system will be operated.

There are a few limitations regarding operation which result from the hardware that is being used:

**Touch:** In this system, it is not possible to operate several buttons at the same time. The touch controller always returns the average position, with the exception of systems that use a matrix touch, in which case the entire touch surface is divided into a fixed matrix.

**Size of the operating elements:** The design and size of operating elements should be made to match the user's needs. The customer should clarify how the system is to be operated. If the system is going to be mostly operated by people wearing gloves then the programmer should not assume that the gloves will be taken off to operate the visualization.



In these cases, the operating elements (buttons, numeric and alphanumeric touch input fields, touch pads, etc.) should be designed with the desired size.

These considerations must be planned for together with the customer right from the start. Making such a change in the project at a later point could only be done with a great deal of effort (each page would have to be changed).

**Keyboard:** You should also check whether the keyboard supports multiple simultaneous keystrokes.

**Real-time capability for the keys / touch:** In process engineering, a movement is started by pressing a key or touch field and stopped when the key or touch field is released = jog key mode. High speed reaction times are required to do this (< 50 ms).

Not every visualization application is able to guarantee such reaction times. A hardwired keyboard (hardware device) is recommended in this case.

**Mouse:** The use of a mouse is not very common for machine operation (on-site visualization). However, mouse usage is highly common on process control systems and visualizations at the control station or main office.

In order to implement a different operating philosophy in a visualization (e.g. mouse and touch), you must take into consideration that there will also be limitations – e.g. implementation of a shortcut menu (right mouse click).

#### 4.4 Variables and data points

One of the most important issues when setting up a visualization is the configuration of the variables or data points.

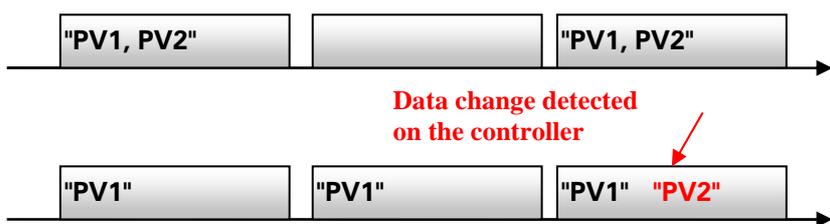
##### Variable (PV)

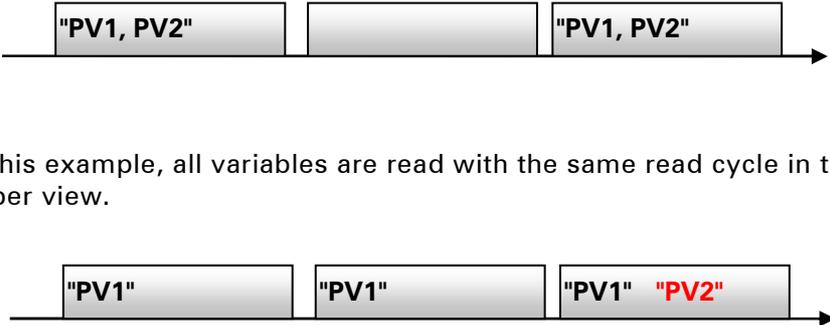
Definition of a controller address

##### Data point (TAG)

A variable is considered a data point if the visualization allows the output of additional attributes (read, write, event, scale, limit, format, convert, read cycle, etc.).

Suitable configuration of a variable and its properties / attributes can considerably increase a visualization's performance.

Property	Behavior
Event / polled	<p>By default, all of the variables in a visualization are read cyclically by the controller (polled).</p> <p>The more active variables that must be read, the slower the cyclic update of the display.</p> <p>If event operation is possible, then some of the load is taken off the cyclic communication because the controller handles the task of checking the event variable value change. As a result, the remaining, polled variables can be updated quicker and more often.</p> <p>Example:</p>  <p>In this example, a variable with the name "PV1" (fast data changes must be displayed) and another with the name "PV2" (slow temperature change) is only read in every second request (read access to the controller).</p> <p>The polled variable can now be read in each request if the variable "PV2" is used as event variable (and others of course).</p>

<p>Read cycle (refresh)</p>	<p>A variable's refresh time determines how fast [ms] the variable should be read by the controller or how fast the controller should perform the event monitoring.</p> <p>Changing the refresh time for variables that change to a higher value slowly or just one time can reduce the load on the cyclic communication (e.g: temperature changes – refresh value = 2000 ms).</p> <p>Example:</p>  <p>In this example, all variables are read with the same read cycle in the upper view.</p> <p>"Fast" variables with a short read cycle can also be read more frequently by changing the read cycle for "slow" variables to a higher value.</p>
-----------------------------	--

### Which variables must be read "actively" in a visualization?

- All variables displayed on the page or connected with a page element.
- Variables that must be read in the background by an alarm system or trend system. This data is usually displayed later.
- Variables that do not have to change their value should be switched to "inactive". The status change from "active ⇔ inactive" variables is executed when a page is changed.

Variables with page elements (e.g. numeric output field) are linked on each screen page and are "actively" read by the controller.

Variables on pages that are not displayed do not have to be read – this reduces the communication load and allows the variables on the active screen page to be updated faster.

### Methods for accessing a variable

In addition to read and write access for variables, "synchronous" and "asynchronous" access is also a common differentiation. Different visualization systems might use different terms for these methods, or they might support only one or the other.

- When using synchronous access (read / write), the program waits for the acknowledgement of a task call. No other system operations are possible during this time. Calling multiple synchronous functions in a loop should definitely be avoided.
- The acknowledgement for an asynchronous access procedure is provided at a later point. In the answer or acknowledgement, the application is notified of whether the access was successful or if an error occurred.

### Data consistency / Synchronization of data access procedures

Data consistency is guaranteed in scalable variables linked to input or output fields (e.g. INT, DINT, REAL, etc.).

Data consistency is no longer guaranteed if large amounts of data (structures, arrays) should be transferred during runtime because the variables are transferred between the visualization and the control unit asynchronous to the controller's program sequence.

In this case, the application must handle the data consistency by synchronizing the data exchange.

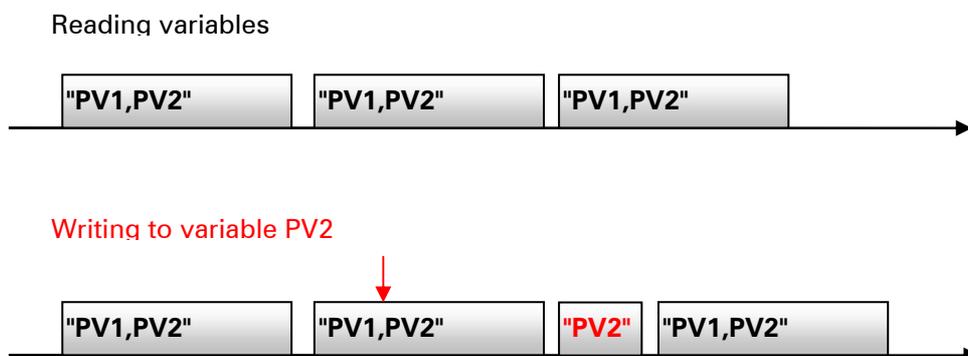
A separate **consistent write or read image** must be created on the controller for the visualization to access. The data exchange is controlled using trigger variables (start the transfer, transfer complete).

Distinct value changes (0, 1, 2, 3, ..., 255, 0) should always be evaluated for the synchronization as long as the access and the variable changes come from the visualization as well as the controller.

Performing a synchronization using a BOOL variable (0,1,0) should be avoided because the variable could overlook a data change 01 due to the read cycle.

### Priority for writing and reading variables

If separate communication between the controller and the visualization hardware must be programmed for the visualization, make sure that a write request to a variable (visualization => controller) is executed with a higher priority than reading variables.



At the moment when a write request is made, the current read request must be waited for. All write requests are then executed.

## 4.5 Visualization runtime behavior

During runtime, a few things must be taken into consideration, which determine the character of a visualization:

- A page change must be executed within a reasonable amount of time after pressing the page change key. You should also keep in mind that the number of page elements on a screen page affects the time needed to execute the page change.
- Actions that block operation should be avoided. If there is no other way around it (e.g. loading data, etc.), then such actions must be displayed for the user in a suitable manner (progress bar).
- "Default text" from a screen page element cannot be displayed on a screen page until the real data has been received. At least one "meaningful" init value must be displayed.
- If required at startup, the visualization should be initialized in a separate startup screen. The first screen page required for operating the system is then displayed after the initialization has completed.

## 4.6 The Visualization Programming Interface

A visualization consists only of drawing the pages and connecting variables / data points to a page element.

Functions not supported by the visualization must be implemented in the programming language supported by the visualization (script, control task, etc.).

### 4.6.1 Error evaluation

Any return value contained in a function must also be evaluated. This is the only way to detect and react to an error during runtime.

Each return value  $\neq 0$  must be displayed on a separate screen page in the visualization (alarm system, separate error protocol, etc.).



Each function must contain an exception handler, which intercepts all types of runtime errors.

All Windows programming environments support the handling of runtime errors (On Error GoTo, Try...Catch...Finally).

#### 4.6.2 Language-dependent programming

When editing texts, make sure that the text is only managed using its reference (text number) instead of the actual text (string). Therefore, the correct text from the current language will always be displayed when the language is changed or when the text is re-edited.

When using UNICODE languages (16-bit character set), the corresponding data type must be used in the controller and in the programming environment (2-byte data types).

#### 4.6.3 The visualization task

In the visualization, there is constant interaction between the displayed process diagram and the process sequence on the controller.

There are different ways to manage a visualization task:

- Just one visualization task: All of the processes and functions required for the visualization process are managed in just one task. This has the disadvantage that it can quickly become large and disorganized.
- Page-dependent visualization tasks: Multiple process-dependent tasks are created for the interaction with the visualization. The task is created in the desired task class according to the priority of the process sequence. Processes that are not time-critical are executed in the controller's idle time (controlling the page structure drawing, etc).

Take into consideration (in the user task) that the interaction is also dependent on the screen page being displayed (performance).

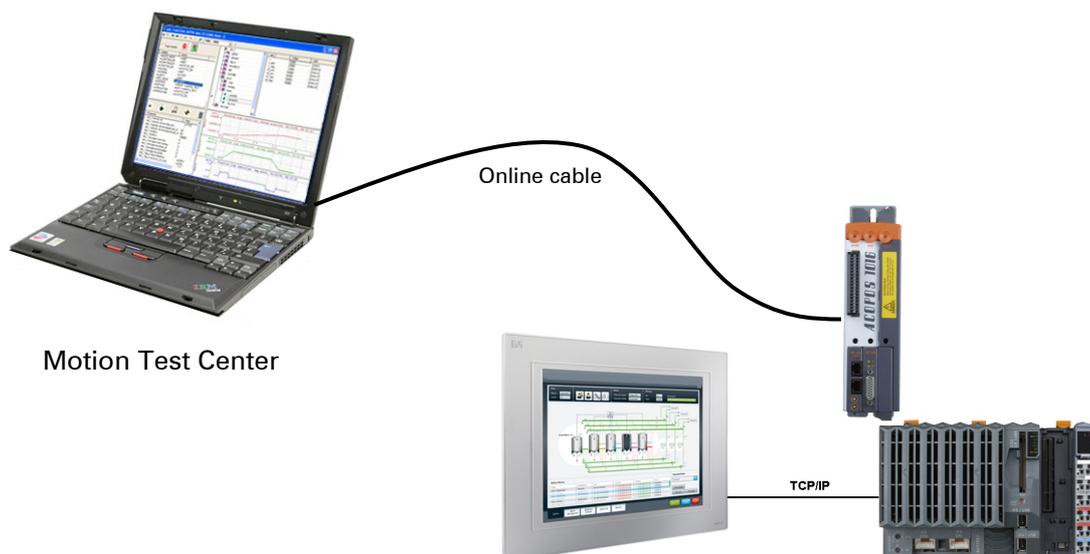
## 4.7 Screen pages for service and commissioning

The visualization is usually not only intended for "normal" operation of the system. Additional screen pages are also required for authorized personnel to commission or service the machine.

Normal operators are not allowed to access the service and maintenance pages which should be protected using a password.

Special tools are often needed for commissioning or for future service tasks. These tools must be either setup in the visualization (= project setup time) or be already available as independent software packages.

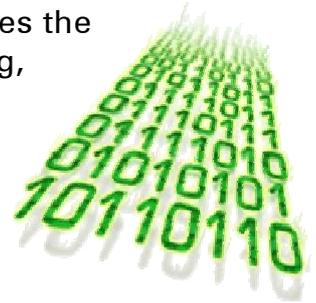
- Are there already tools or existing software packages that take care of these services?
- Can these existing packages be integrated in the visualization? If yes – how?
- Are software licenses required for these products?
- Is there a potential for compatibility problems when using new hardware?
- Where can I receive support when integrating the tools in my software?



## 4.8 Data and Data Management

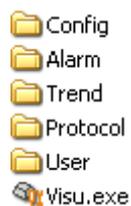
Every visualization creates data during runtime that provides the user with information about the process (logging, analyzing, archiving).

There is no wrong or right way to manage data. Either the customer specifies the method for managing the data or the programmer has a preferred method.



### 4.8.1 Managing data

If possible, data that belongs together should be managed in the same directory. This makes any future access for analysis much easier.



Depending on the type of data, the name and extension (e.g. Trend\_01012005\_001.trn) should also be assigned accordingly.

The names within existing alarm, trend and protocol systems are often determined by the system.

### 4.8.2 Data formats

There are different data formats that may be used depending on the hardware and software. Each of these formats has its advantages and disadvantages.

Various data formats could also be used in a visualization (e.g. ASCII file for initialization parameters, binary files for alarm and trend recording, XML for protocol systems).



**ASCII file:** An ASCII file is any file that contains only simple text in accordance with the ASCII standard. These files can be read out on just about any computer and are therefore easily transferred between different systems.

One disadvantage however, is that the structure of these ASCII files does not follow any rules or formats. Furthermore, performing searches in large ASCII files generally takes more time because the entire file must be searched.

An exception is the **CSV file**. A CSV file is a text file that contains data structured into tables. The abbreviation CSV stands for "*Character Separated Values*" or "*Comma Separated Values*", because the individual values are separated by a special separator - usually commas. However, the semicolon, colon, tab and other characters are also common. There is no official standard for this file format.

CSV files often use the file extension **.txt** instead of **.csv** and can also be created and edited in any text editor.

**Registry (Windows):** The registry does not contain documents, but user defined options, which allow the program or operating system to be dynamically adjusted for the user. This information can include the layout of the program, e.g. the preferred window position.

Name	Type	Data
 (Default)	REG_SZ	(value not set)
 frmMainHeight	REG_SZ	11190
 frmMainLeft	REG_SZ	-60
 frmMainPosition	REG_SZ	0
 frmMainState	REG_SZ	2
 frmMainTop	REG_SZ	-60
 frmMainWidth	REG_SZ	15480

Registry entries are created in keys, which branch off from main keys located further up in the hierarchy. Users should exercise extreme caution and only make changes to the registry when absolutely certain because mistakes made there can affect system functionality.

Saving dynamic parameters in the registry has the disadvantage that the customer cannot easily make any changes.

Making a change to the registry is complicated when using Windows Embedded operating systems with write protection on the system partition.

**XML file:** The **Extensible Markup Language**, abbreviated as **XML**, is a standard for creating documents in the form of a tree structure that can be read by humans and machines. XML defines the rules for the structure of these documents. The details of the respective document must be specified for a concrete application example ("XML application"). This particularly affects how the structured elements and their organization within the document tree are determined. To put it differently: XML provides the rules used when defining document types.

**Databases:** Databases are used when managing large amounts of data. This data is logged, ordered and stored based on specific characteristics and rules.

The user is provided with standardized functions for accessing these databases.

When using databases, you should be aware that they are not automatically sorted. This means that a database needs more memory for each new entry even if data is deleted. Therefore, a database must be compressed frequently (i.e. the database is reorganized). Keep in mind that the database cannot be accessed during this procedure.

**Binary files:** Unlike a simple text file, a binary file contains non-alphabetic characters and can use any byte value. Binary files are more commonly used for saving data, than for saving text.

#### 4.8.3 Saving and Archiving Data

A few things must be taken into consideration regarding the configuration when archiving data:

- How much data is being managed? Is there enough available memory (Compact Flash, etc.)?
- How will the data be further processed (external tools)?
- Are any special software licenses required?
- How long does this data have to be archived?
- Will this data be stored externally (network, storage media)?
- Which events must be recorded (alarm, trend, protocol, etc.)?
- How will this data be recorded (cyclically, event-controlled, when changed)?
- Will data be evaluated based on the language? In this case, a reference (text index) must be saved instead of the actual text.



Once these conditions have been established, then their dependencies and data format can also be determined.

### 4.8.4 Compatibility

If the format for logging data must be changed during the project setup or after commissioning, make sure that "old" data is still compatible (i.e. this data must still be able to be processed in the visualization - upward compatible).



If this is not possible, then this data must be converted so that it can also be analyzed in the future.

This conversion can be made using either the visualization software or external tools.

If, at the time of project setup, the programmer already knows that the data format will be changing (e.g. due to future expansions), a version indicator in the file can be helpful for future assignment of various formats.

## 5. INTEGRATING A VISUALIZATION

Integration of the product (i.e. the test) is an important part of the project setup. If possible, this test should always be carried out in consultation with the customer.

Testing procedures should be specified at the beginning when putting together the visualization's software design specification.

The visualization should be integrated on a newly installed target system as well as the development system. This is the only way to guarantee that the testing environment also corresponds to the system being used by the customer.



Development



Runtime

The initial tests on the visualization are performed in the office environment. After these tests have been successfully completed, then additional tests should be carried out on the system.

### Which tests must be carried out?

- Testing the communication with the controller project.
- Are all variables correctly displayed and calculated?
- Are the entries and input limits also correct?
- Are all key functions correctly configured?

For the initial tests, a dummy project, which contains all of the necessary variables, can be created on the controller.

Correct processing of the visualization must also be tested using simulation programs in order to test a project procedure in the visualization.

These simulation programs make it possible to test worst case scenarios (extreme loads) and time-critical program sequences.

### 6. PROJECT MAINTENANCE AND ORGANIZATION

#### 6.1 Changes to the project

**"I could have done that better this way"**

When creating a visualization, the time often comes when the initially adopted plan leads to a dead lock or complex solutions.

In known simpler solutions should be taken into consideration and changes should be made during a project reassessment phase. The changes should not be implemented right before the project completion. Instead, changes should be made once the current phase has been completed.

The project change must be documented, the previous project state must be archived and the customer must be informed if necessary.

Any changes made to the project must be logged whether the changes are based on a customer request or are changes and error corrections implemented by the programmer. This makes it possible in the future to review a written record of each project step. The format and manner in which the protocol is kept should be arranged with the customer.

##### 6.1.1 Cleaning up the project



Data that is no longer needed will start to amount over the course of the project whenever various changes are made or while integrating the visualization. This data can be removed from the project.

- Simulation or test pages: These pages can be kept in the project for future tests, however, the user should not be able to navigate to these test pages (page change).
- Bitmaps that are no longer required.
- Bitmaps that are no longer required.

## 6.2 Software distribution and storage

While creating the visualization, the current project state must be archived (backup copy) after each major change.



Backup copies should never be stored on the development PC. Backups should always be stored on an external server (backup) or on a suitable storage media.

To avoid compatibility problems, all dependent software packages that are used and tested with the visualization should be archived together with the current project state (communication drivers, control software, external service tools, etc.)

Different databases for version checking are available for the user to use when archiving.

## 6.3 Turning the project over to the customer



The current state of the project should always be turned over to the customer as a complete setup or a pre-installed storage medium (CompactFlash, etc.). You should avoid copying and distributing individual files.

The setup should contain all of the software packages used for integration as well as an installation guide. This is especially necessary when the installation sequence of the software packages contains dependencies.

### 6.4 Documentation

There are a few different types of documentation:

- Project documentation
- User documentation
- Online help in the visualization



**"A good software design specification is the foundation for good documentation"**

#### 6.4.1 The project documentation

Each stage of development should be documented when creating the project. The project documentation is used to record all expansions and changes.

It is important that the programmer and customer also use the same expressions and names of machine elements in the documentation from the time of project analysis until project completion.

#### 6.4.2 The user documentation

The user documentation provides the user with support for operating the machine.

The user documentation must differentiate between the various operating personnel, the service technicians and other persons who will be operating the visualization. The normal operator does not need any information about commissioning, service and similar issues.

Furthermore, the user documentation should provide screenshots of each page necessary for operation.

### 6.4.3 Online help

The online help can be looked at as its own project. The help files are used to supplement the user documentation. Creation of the online help files must be taken into consideration when planning the amount of work.

When using Windows, there are different ways to create and call up the online help.

Make sure that the visualization program supports context-sensitive help if desired.

## 7. SUMMARY

Unfortunately, the visualization is not yet finished after completing this training module.

Several books could certainly be written about creating visualization system projects.

The training module provides information showing the various aspects of a visualization system. Looking carefully at different situations users experience improves ergonomics and therefore acceptance by operators.

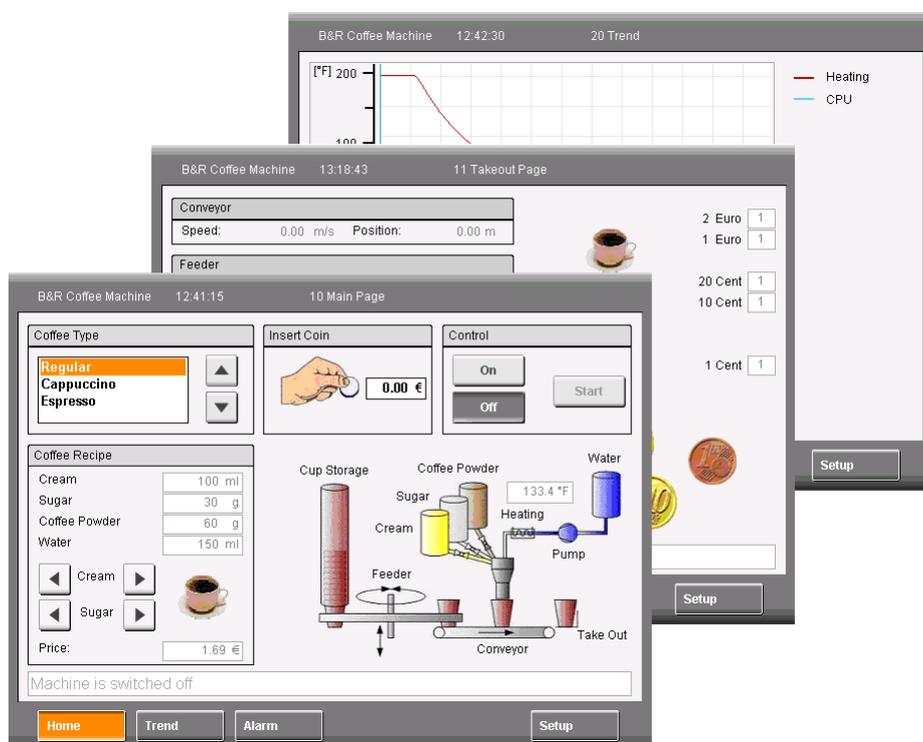


Fig. 3: Visualization Programming Guide

User-friendly equipment contributes considerably to efficient solutions - also with regard to machines and systems.

## Overview of 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 (LAD)  
TM241 – Function Block Diagram (FBD)  
TM246 – Structured Text (ST)  
TM250 – Memory Management and Data Storage  
TM261 – Closed Loop Control with LOOPCONR

TM400 – The Basics of Motion Control  
TM410 – The Basics of ASiM  
TM440 – ASiM Basic Functions  
TM441 – ASiM Multi-Axis Functions  
TM445 – ACOPOS ACP10 Software  
TM446 – ACOPOS Smart Process Technology  
TM450 – ACOPOS Control Concept and Adjustment  
TM460 – Starting up Motors  
TM480 – Hydraulic Drive Control

TM500 – The Basics of Integrated Safety Technology  
TM510 – ASiST SafeDESIGNER  
TM540 – ASiST SafeMC

TM600 – The Basics of Visualization  
TM610 – The Basics of ASiV  
TM630 – Visualization Programming Guide  
TM640 – ASiV Alarm System, Trend and Diagnostic  
TM670 – ASiV Advanced

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

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

## CORPORATE HEADQUARTERS

Bernecker + Rainer Industrie-Elektronik Ges.m.b.H.

B&R Strasse 1

5142 Eggelsberg

Austria

Tel.: +43 (0) 77 48 / 65 86 - 0

Fax: +43 (0) 77 48 / 65 86 - 26

info@br-automation.com

www.br-automation.com

TM630TRE-00-ENG 0610  
©2010 by B&R. All rights reserved.  
All registered trademarks are the property of their respective owners.  
We reserve the right to make technical changes.

155 offices in 60 countries - [www.br-automation.com/contact](http://www.br-automation.com/contact)



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