Visualization Programming Guide

TM630

Perfection in Automation
www.br-automation.com

Prerequisites & Requirements

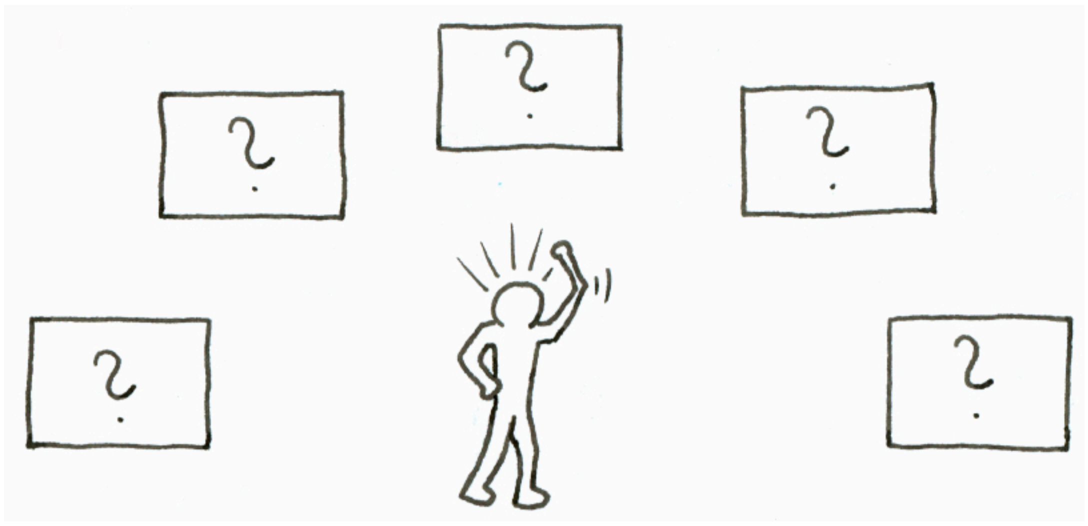
Training modules	TM600 – Introduction to Visualization
Software	None
Hardware	None

TABLE OF CONTENTS

1 INTRODUCTION

At some point or another, we have all asked ourselves "how should I get started with this new project"?

This is especially the case when there are no project guidelines or the guidelines have not yet been specified. The advice offered in this training module helps the user overcome the initial obstacles faced when beginning a visualization project, which in the end helps save time and expenses.



Visualization Programming Guide

Even experts may find some useful tips for developing a new visualization project.

When creating a visualization application, it helps to 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 challenge comes your way.

1.1 Training module objectives

This training module provides an overview of how to ...

- ... design more effective visualizations
- ... avoid poor design concepts
- ... create specifications
- ... improve programming style and project organization
- ... maintain and organize your project

2 PROJECT DEVELOPMENT PROCESS

A defined development process gives you a road map to ensure that your project stays organized and that nothing is overlooked.

"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 table provides an overview of the project development process.

Pre-project phase (presales, sales)	Project start	Project execution	Project end
⇨ Presentation ⇨ Clarification ⇨ Offers ⇨ ...	⇨ Project analysis ⇨ Project test ⇨ Team formation ⇨ Transfer of responsibility	⇨ Concept ⇨ Structure ⇨ Work packages ⇨ Communication	⇨ Achievement of objectives ⇨ Documentation ⇨ Backing up the results ⇨ Completion

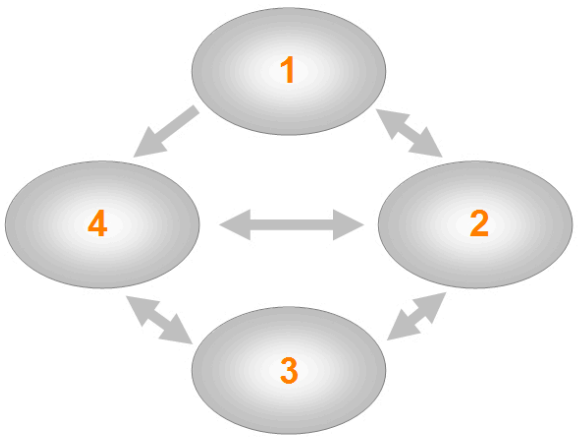
Table: Project phases

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:

Project sections and interaction between them:

1 Design specifications + Functional specifications
2 Project development, testing, integration
3 Machine commissioning
4 Documentation



Project sections and interaction between them:

3 SPECIFICATIONS

The basis of any visualization is formed by the customer's requirements and the resulting specifications for design and functionality.

Time spent [%] – As it is		Time spent [%] – As it should be	
Specification	0 - 20	Specification	30
Project development	80 - 100	Project development	20
Testing	20 - 0	Testing	50

Table: Distribution of time spent

As illustrated in this table, 1/3 of the time available for the project should be dedicated to defining the specifications. This reduces the time spent making changes during project development.

In practice, however, the specification step is often short-changed when developers feel they don't even have enough time for their "normal" development tasks. Nevertheless, it's worth considering that well-defined specifications can save time later in project development.

When creating specifications, it's important to find a balance between including all the necessary information and giving the programmer creative freedom. Defining every single bit would be overkill.

Aspects of the project to analyze when defining specifications:

- What type of visualization should be used?
  (price, runtime costs)
- What type of visualization should be used?
  (embedded, remote visualization, etc.)
- Connection to higher level systems
- What 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?
- What information is important?
- What operating method should be used?
  (keys, touch screen, etc.)

Design elements to consider:

- What is the main image?
- What elements are displayed?
- What sub-levels are required?
- What colors and shapes should be used?
- Navigation in the pages
- Navigation between the pages
- Who is permitted to operate which pages?
- How should the pages be laid out?
  (Password levels)
- How will the pages be grouped together?

Specifications

(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 establish the basic framework of the visualization.

Before you get into programming the individual pages, create some selected sample pages and present them to the customer. These samples can then be used as the template for the remainder of 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.

The customer knows the machine and the visualization requirements, while the programmer knows the programming tool and its capabilities, so it is essential that the two cooperate to define specifications that are realistic and feasible.

Customer / client:

- Knows the machine and the requirements, operates the machine

↑ Design specification ↓	→→→	Application

Service provider / programmer:

- Knows the programming tool and its capabilities

When defining the specifications, the programmer should already be familiar with the programming tool that will be used to create the visualization application.

If he or she isn't aware of the tool's capabilities until they begin programming, there is no way of knowing whether certain requirements are actually feasible.

4 SOFTWARE DESIGN

The statement "it just sort of happened that way" is commonly heard when referring to projects that were started at the programming stage instead of on paper.

Modifications made later in development are much harder to implement and maintain than functions that are integrated in the design from the very beginning.

Software design

4.1 Default visualization project template

A new project typically starts with a blank page. Then all of the necessary components such as bitmaps and texts are added. With this approach, however, the overall structure and the positioning of individual elements varies from one project to the next.

Using default projects or templates allows the programmer to start with the same basic structure every time. Regardless of how similar or different the projects are – a good framework always saves time when starting a new project.

Elements of a default project:

- Keyboard and touch screen configuration
  (for numeric and alphanumeric input)
- Unit groups
  (scaling and formatting)
- Bitmaps and bitmap groups
- Fonts and font sizes
- 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 application

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

Possible components of a visualization:

- Pages
- Static and dynamic text
- Alarms
- Data management, recipes
- Units and scaling
- Graphic elements
- Connections, variables, data points
- Button and keypad configurations

These components can be expanded as needed. Any functions not already contained in the application can be implemented with additional programming.

4.2.1 Page design

Pages make up the main part of a visualization. There are a few rules to consider when designing a page:

Element	Rule
Font	The number of different fonts should be kept to a minimum. You can vary the size of a font to focus attention on certain objects. Keep in mind that operators must be able to read the page from a distance of several meters.
Font	It's more important for a text to be clearly readable than for it to be "pretty".
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. Using graphics
Graphics	If used on multiple pages, an icon should always look the same.
Graphics	Use a logical, standardized color scheme (e.g. Green for OK, Blue for Cold, Red for Error)
Graphics	Use commonly known icons Save Copy Insert
Operation	The size of the input fields and buttons should be large enough that they can be easily operated by the operating personnel. Keep in mind that the operator might be required to wear gloves.
Operation	Non-functional or locked page elements must be clearly identifiable as such.
Layout	Avoid cluttering your pages. Important information on the page should be visible at first glance.
Layout	Identical functions on different pages should be in the same position on every page.

Table: Page design elements

Element	Rule
Layout	Be sure that the various elements on a page are aligned with one another.
Layout	Establish a strategy for key navigation (cursor control, page change) and apply this strategy consistently to all pages.
Layout	If certain machine options can be hidden, be sure that remaining elements are re-arranged appropriately. Avoid having large areas of empty space on the page. The options may need to be placed on separate pages, and if the programming tool allows it, you can position the elements dynamically.
Text	If text can be displayed in multiple languages, be sure that the text field is large enough to fit the longest version.

Table: Page design elements

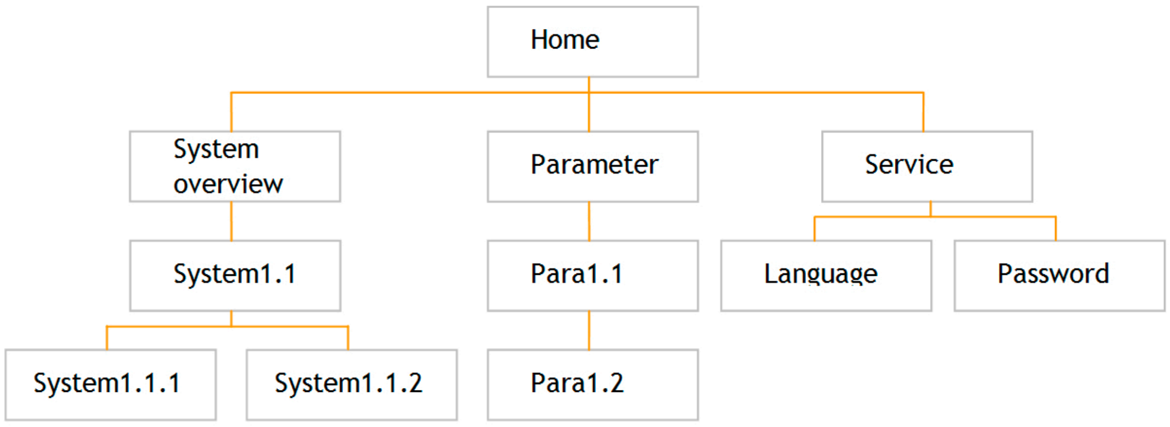
4.2.2 Page branching and grouping

Once you know the content of all the pages, you can arrange them into groups.

These groups can be structured to represent the:

- Process flowchart
- Operating levels
- Service and maintenance levels

Drawing an application map can help determine the best menu design and page navigation.



Example of an application map

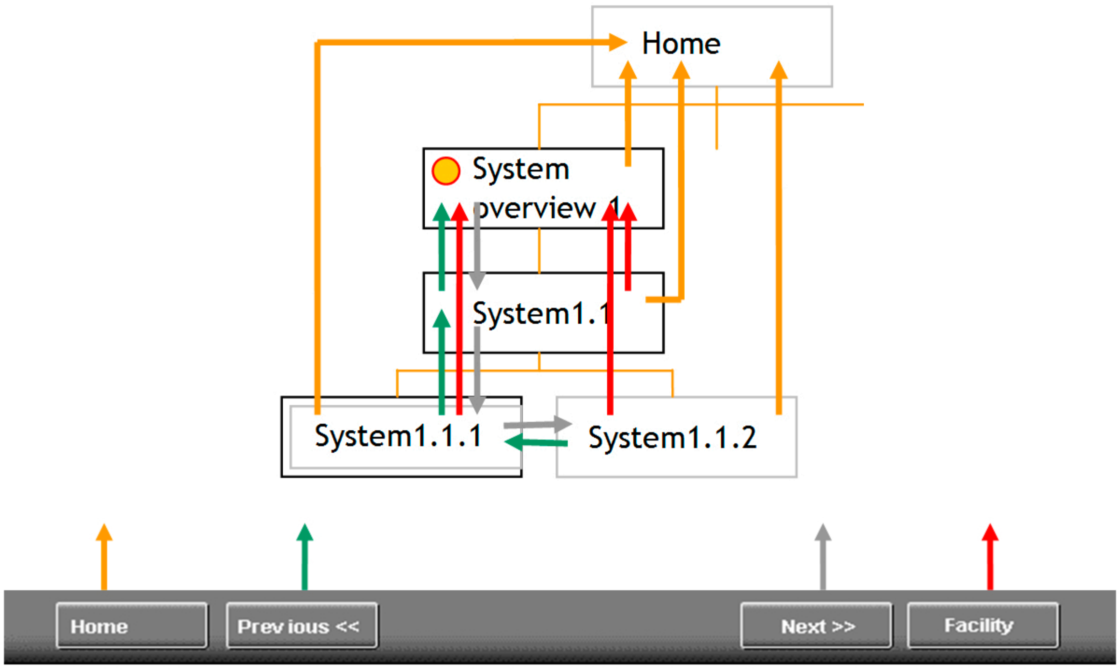
In this example, the grouping in the project page list could look something like this:

- 0000_StartPage
- 0100_SystemOverview
- 0110_System1.1
- 0111_System1.1.1
- 0112_System1.1.2
- 0120_System1.2
- 0200_Parameters
- 0210_Parameters1.1

- 0220_Parameters1.2
- 0300_Service
- 0310_Language
- 0320_Password

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

- Buttons for navigating should always be located in the same position on the page.
- When navigating in lower levels, the user must be able to return to the main level without having to go back page-by-page.



Page navigation in the footer

In this example, navigation between page levels is illustrated using colored arrows.

Arrow color	Description
Red arrow	Level "1" can be reached from any lower level page.
Green arrow	Within a lower level, this arrow brings you back within the same level. From the first page of a lower level, it brings you to the next level up.
Gray arrow	Within a level, this arrow brings you to the next page. From the last page of a level, it brings you to the first page of the next level down.
Orange arrow	This arrow brings you back to the start page from any level.

Table: Page navigation overview

There are of course many different navigation concepts out there. This example describes only one effective approach.

Suggestions for grouping pages:

- Place group numbers in front of the page names.
- Leave space between the group numbers so you can add pages later on.
- Indicate the level on each page. This allows users to easily identify what level they're on.

Navigating to a different page

Generally, page changes are triggered by pressing a key or touch screen button. The destination page should always be assigned directly to the key or touch action. It's best to avoid changing pages using a control variable, because the control program would have to be modified every time the page numbers are changed.

It is generally only necessary to use variables to change pages for specific actions triggered by the control program. This includes things such as alarms or password-dependent page changes.

It is also important to configure the application so that these page changes can be delayed. Users should be able to finish what they're doing without being interrupted by an automatic page change.

4.2.3 Layering

Using layers allows you to place related page elements on a layer and to use these elements as often as necessary. There are differences in how layers are configured, how they function and how they are used in different visualization programming environments.

Layers can be used for the following types of elements:

- Headers and footers
- Input fields
- Buttons
- Output fields
- Dialog boxes

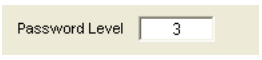
Advantages of layering:

- Image information used on multiple pages only has to be created one time.
- Changes made later only have to be made in one place.
- Layers can be configured to be shown or hidden dynamically during runtime.
- Layers can be used to implement individual dialog boxes.
- Control the visibility of machine options.

If the visualization tool supports layers, then you should always use them. Before you begin drawing up the pages, first decide what information belongs on each of the layers.

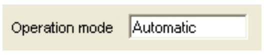
4.2.4 Text in a visualization application

A visualization application includes both static and dynamic text. Static text is a fixed part of a page, such as the description of a display element.


Password Level 3

Static description text for a display field

Dynamic text changes according to variables connected to the text element.



Displaying the operating mode as a dynamic text

Both static and dynamic texts are easiest to manage when grouped in functional text groups.

Advantages:

- Text used on different pages only has to be created once.
- Text changes made later on only have to be made in one place.
- Texts for different languages only have to be translated once for each language.

Language-dependent texts

If possible, texts should be managed in separate text files for each language.

Advantages:

- Only the files for the original and target language need to be passed on for translation. This makes it possible to obtain multiple translations in parallel.
- You can create a version of the visualization application that includes only the languages that were ordered. Additional languages can be then be added later as an option.

4.2.5 Graphic elements

Managing graphic elements

To improve clarity and organization, it helps to create logical groups or separate subdirectories containing related graphical elements, such as all bitmaps.

Dynamic graphic objects

Dynamic objects and constantly changing elements should be kept to a minimum. These can distract the operator from the more important information on the screen.

Dynamic objects

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

4.2.6 Using standards

An extensive selection of standard dialog boxes are available to programmers, especially for Windows.

There are a few things to consider, though:

- The text in standard dialog boxes is always displayed in the installed language.
  Generally, this language cannot be switched while the system is running.
- The buttons in these dialog boxes are not always large enough.
- It is not always possible to display UNICODE characters.

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

4.2.7 Giving elements descriptive names

When you add an element to the application it is given a default name. This name identifies the type of element, but when you have multiple elements of the same type it becomes impossible to tell them apart.

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

Default name	Descriptive name
• InputField_1	• nbSetPassword
• OutputField_1	• txtActPasswordLevel
• InputField_3	• nbSetCoolingDelay
• InputField_4	• nbMaxWaterTemp

Table: Default names vs. descriptive names

As you can see, the descriptive names identify what each of the elements is used for.

> It is always necessary to give visualization objects descriptive names. When programming the controller, the customer also uses standardized coding rules.

4.3 Operating concept

The visualization hardware used determines how a system will be operated. It is important to be aware of the limitations that result from the specific hardware that is used.

Touch screen operation

Touch screen systems

Different displays use different types of touch technology. Resistive touch screens do not allow multiple buttons to be pressed simultaneously. The touch controller always uses the average touch position. These systems can be used by operators wearing gloves or using a stylus.

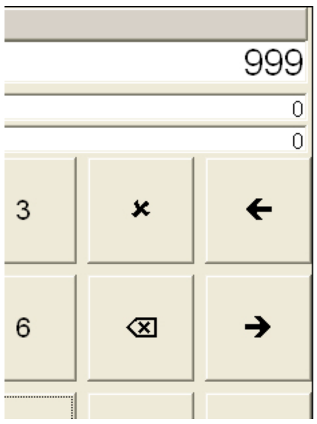
Multi-touch systems can differentiate between multiple touch positions on the screen. They usually do not allow operators to wear gloves or use a stylus.

Size of the operating elements

The design and size of operating elements should match the user's needs. Clarify with the customer exactly how the system is to be operated. If the system will mostly operated by people wearing gloves, you can't assume that the gloves will be taken off to operate the visualization application.

In this case, you need to make the operating elements large enough.

These considerations must be clarified with the customer right from the start. Making such a change in the project at a later point would be tedious.



Numeric touchpad

Keyboard

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

Real-time capability

Some applications require drive movements to be performed in jogging mode. This is only possible with very fast response times (< 50 ms). Not all visualization tools allow this, so it's important to check the technical limitations closely.

Mouse

The use of a mouse is not very common for machine operation. However, mouse usage is highly common in the control station or main office.

> Make sure the operating method you choose can be implemented with the visualization tool you'll be using.

4.4 Variables and data points

One of the most important issues when developing a visualization application is configuring the variables and data points.

Variables and data points

Description	Description
Process variable (PV)	Defines an address on the controller
Data point (TAG)	A data point is a variable with additional attributes. **Data point attributes:** • Format • Conversion • Refresh time • Data direction The configuration of a variable and its attributes can make a significant difference in the performance of a visualization application.

Table: Variables and data points

Property	Behavior
Polling and event-based	By default, all of the variables in an application are read cyclically (polled) by the controller. The more active variables that must be read, the longer it will take the display to update cyclically. If event-based operation is possible, then some of the load is taken off the cyclic communication because the controller only has to monitor changes to event variables. The remaining polled variables can be updated more quickly and more often.
Read cycle	A variable's refresh time determines how fast [ms] the variable should be read by the controller or at what interval the controller should monitor the event.

Table: Access types and times for variables

Property	Behavior
	Increasing the refresh time for variables that change slowly or only once frees up cyclic communication (temperature changes – read cycle > 2000 ms).

Table: Access types and times for variables

Which variables need to be read actively?

- Any variables displayed on the screen or connected to a graphic element.
- Variables read in the background by an alarm system or trend system.
  This data is usually displayed at a later time.
- Variables for which a value change isn't required should be set to "inactive".
  The status of "active + inactive" variables is updated when the page is changed.

Every page contains graphic elements with connected variables that are read actively by the controller.

Variables for elements and pages that are not currently displayed don't need to be read. This frees up cyclic communication and makes the variables for the current page update 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, the program waits for a response to confirm a task call. No other system operations are possible during this time. Definitely avoid calling multiple synchronous functions in a loop.

Asynchronous access is confirmed later on. The confirmation response tells the application whether the access was successful or if an error occurred.

Data consistency / Synchronization of data access procedures

Data consistency is guaranteed for scalable variables linked to input or output fields, such as DINT variables.

If large volumes of data (structures, arrays) are to be transferred during runtime, data consistency is no longer guaranteed. This is because the variables are transferred asynchronously between the visualization application and the controller.

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

An extra read/write image needs to be set up on the controller, which the visualization application can then access. Data exchange can be managed with trigger variables.

If the variables are accessed and modified from both the visualization application and the controller, then distinct value changes (0, 1, 2, 3, ..., 255, 0) should always be evaluated to ensure synchronization.

Avoid synchronizing with a BOOL variable (0,1,0), because the variable could overlook a value change of 0-1 due to the read cycle.

Priority for writing and reading variables

If the visualization application requires separate communication between the controller and the visualization hardware, make sure that a write request to a variable (visualization → controller) is executed with a higher priority than a read request.

4.5 Visualization runtime behavior

During runtime there are a number of things to consider that affect the quality of the visualization performance.

Pay special attention to the following:

- Page changes
  The page should change within a reasonable amount of time after a button is pressed.
  Keep in mind that the number of elements on a page affects the time needed to execute the page change.
- Blocking
  Actions that block operation should be avoided.
  If they cannot be avoided, these actions should be communicated to the user, e.g. with a progress bar.
- Reasonable default text
  Avoid leaving "default text" on a page.
  Until actual data is received, at least show a reasonable initial value.
- Initialization page
  If the visualization application requires initialization upon startup, this should be implemented with a separate "Start-up Screen".
  The first page required for operating the system is not displayed until initialization is complete.

4.6 Visualization programming interface

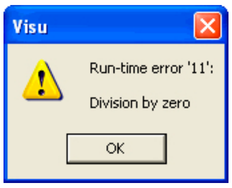
A visualization application often consists of more than simply drawing the graphics and connecting data points to graphic elements.

Functions that aren't supported by the visualization tool need to be programmed.

4.6.1 Evaluation of errors

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. Return values that represent errors need to be displayed on the alarm page or error log.


Visu

Run-time error '11':

Division by zero

Runtime error not evaluated

Every function needs to have an exception handler, which catches all types of runtime errors.

All Windows programming environments support handling of runtime errors.

(`On Error GoTo`, `Try...Catch...Finally`).

4.6.2 Language-dependent programming

When working with text, always be sure to manage texts using their reference numbers and not the text itself. This ensures that the correct text for the current language will always be displayed when the language is changed or when the text is re-edited.

When using UNICODE, be sure to use the corresponding data type in the controller and the programming environment.

Unicode characters

4.6.3 Visualization program

A visualization program involves constant interaction between the displayed process diagram and the processing on the controller.

There are different approaches to managing a visualization program:

- A single visualization program:
  All visualization processes and functions are managed in a single task. The disadvantage of this approach is that it can quickly become unwieldy.
- Page-based visualization program:
  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. Non time critical processes should be executed in the controller's idle time.

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

4.7 Pages for service and commissioning

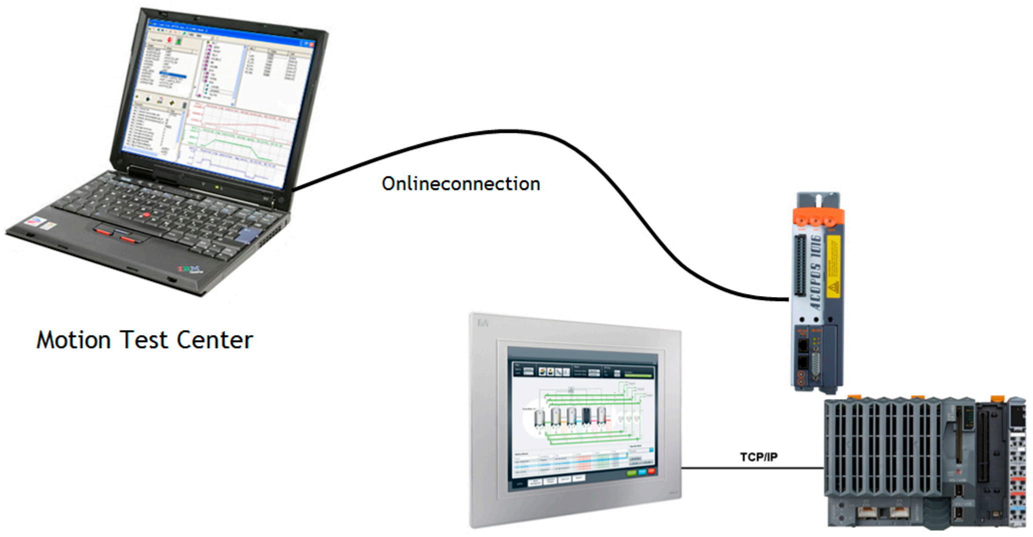
Often, a visualization is not only intended for "normal" operation of the system. Additional pages are also required for authorized personnel to commission or service the machine.

Normal operators are not allowed to access the service and maintenance pages, so they should be protected by a password.

Special tools are often needed for commissioning or for future service work. These tools must either be configured as part of the visualization application or existing software packages can be used.

Pay special attention to the following:

- Are there already tools or existing software packages that provide these services?
- Can these existing packages be integrated in the visualization?
  If yes, then 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?



Integration of external tools

4.8 Data and data management

Every visualization creates data during runtime that provides the user with information about the process.

"Record, Analyze, Archive"

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

Data management

4.8.1 Managing data

Related data should be kept in the same directories. This makes any future access for analysis much easier.

Depending on the type of data, the name and extension should also be assigned accordingly.

The names within existing alarm, trend and logging 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.

Different formats may even be used within a single visualization system.

Frequently used formats:

- ASCII file for initialization parameters
- Binary files for alarm/trend recordings
- XML for log systems

File format – The key to the content

ASCII file

An ASCII file is any file that contains only simple text in accordance with the ASCII standard. These files can be read by 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, it generally takes longer to search in large ASCII files, because the entire file must be searched.

An exception is the CSV file type. 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 a comma. 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 options that allow programs or the operating system to adjust dynamically for the user. This information includes the layout of the program, such as the preferred position of the window.

Registry entries are created in keys, which branch off from main keys located further up in the hierarchy.

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

XML file

XML (Extensible Markup Language) 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 structure of the respective documents needs to be defined specifically for any particular XML application. This involves defining the structure elements and their arrangement within the document tree.

XML provides the rules used to make these definitions.

Databases

Databases are used to manage large amounts of data. This data is logged, organized 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 "cleaned up". This means that a database needs more memory for each new entry even if data is deleted, and must therefore be compressed regularly in order to be reorganized.

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 text.

4.8.2.1 Saving and archiving data

There are several configuration issues to consider when archiving data:

- How much data is being managed?
  Is the existing memory sufficient?
- How will the data be subsequently processed?
  Are there external tools?
- Are any special software licenses required?
- How long does this data have to be archived?
- Does the data need to be stored externally?
  (network, storage media)
- What events need to be recorded?
  (alarms, trends, logs, etc.)
- How is data recorded?
  (cyclically, event-based, on change, etc.)
- Will data be evaluated based on the language?
  (In this case, a reference must be saved instead of the actual text.)

Once these conditions are known, then their dependencies and data format can also be determined.

4.8.2.2 Compatibility

If the format for logging data must be changed during development or after commissioning, make sure that "old" data is still compatible. This data must still be able to be processed in the visualization application (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 performed using either the visualization software or external tools.

Compatibility

If you are already aware of a future change in data format during development, you can use a version code in the file to help assign the different formats later on.

5 INTEGRATING A VISUALIZATION

Integrating the product (i.e. the function test) is an important part of development. If possible, the customer should be involved in planning and performing this test. Testing procedures should be specified at the beginning when putting together the design specification.

Initial tests of the visualization are performed in an office environment. Once these tests have been successfully completed, additional tests should be carried out on the actual system.

The following tests should be performed:

- Does communication with the control project work properly?
- Are all variables displayed and calculated correctly?
- Can values be entered properly and are input limits applied correctly?
- Are all key functions configured correctly?

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

It is also necessary to use simulation programs to test proper functionality in all steps of a project and under all conditions.

These simulations can be used to test "worst case" conditions and time-critical processes.

The visualization application should be tested not only on the development system, but also on a newly installed target system. This is the only way to ensure that the testing environment accurately represents the system being used by the customer.

6 PROJECT MAINTENANCE AND ORGANIZATION

"Aha, I should have done it like this..."

6.1 Changes to the project

There often comes a time in development when "Plan A" leads to a dead end or an unnecessarily complex solution.

When there are known solutions to these issues, make a note of them and apply them in a "clean-up" phase of the project. Don't make changes just before completion. Instead, changes should be made once the current phase has been completed.

Be sure to document any changes to the project, archive the previous version of the project and inform the customer if necessary.

Any changes made to the project must be logged, regardless of whether they were requested by the customer or made by the developer to correct errors. This makes it possible trace each step of the project in writing. The format and manner in which the log is kept should be arranged with the customer.

6.1.1 Cleaning up the project

Throughout the integration process and as changes are made, data will accumulate that is no longer needed once testing is complete. This data can be removed from the project.

The following should be removed:

- Simulation and test pages:
  These pages can be kept in the project for future testing; however, the user should not be able to navigate to these test pages.
- Bitmaps that are no longer required.
- Lines of code that have been commented out and are no longer needed.

Cleaning up the project

6.2 Software distribution and storage

Throughout development, a version of the project should be backed up before every major change. Backup versions should never be stored on the development PC, but rather on an external server or on suitable storage media.

The user can choose from a range of different databases to manage the versions of archived data.

Archiving

To avoid compatibility problems, all dependent software packages that are used and tested with the visualization should be archived together with the current version of the project.

6.3 Turning the project over to the customer

The current version of the project should always be turned over to the customer in the form of a complete Setup or a pre-installed storage medium. 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 software packages need to be installed in a specific order.

Turning over the project

6.4 Documentation

There a different types of documentation:

- Project documentation
- User documentation
- Online Help documentation in the application

Documentation is important

"Good documentation starts with well-defined specifications"

6.4.1 Project documentation

As the project is developed, every step along the way should be documented. The project documentation is used to record all additions and modifications made later on.

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

6.4.2 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 others who will be using the visualization application.

The day-to-day operator does not need any information about commissioning, service and the like. Furthermore, the user documentation should provide screenshots of each page necessary for operation.

6.4.3 Online Help

The Online Help can be viewed as a separate project. It serves as a supplement to the user documentation.

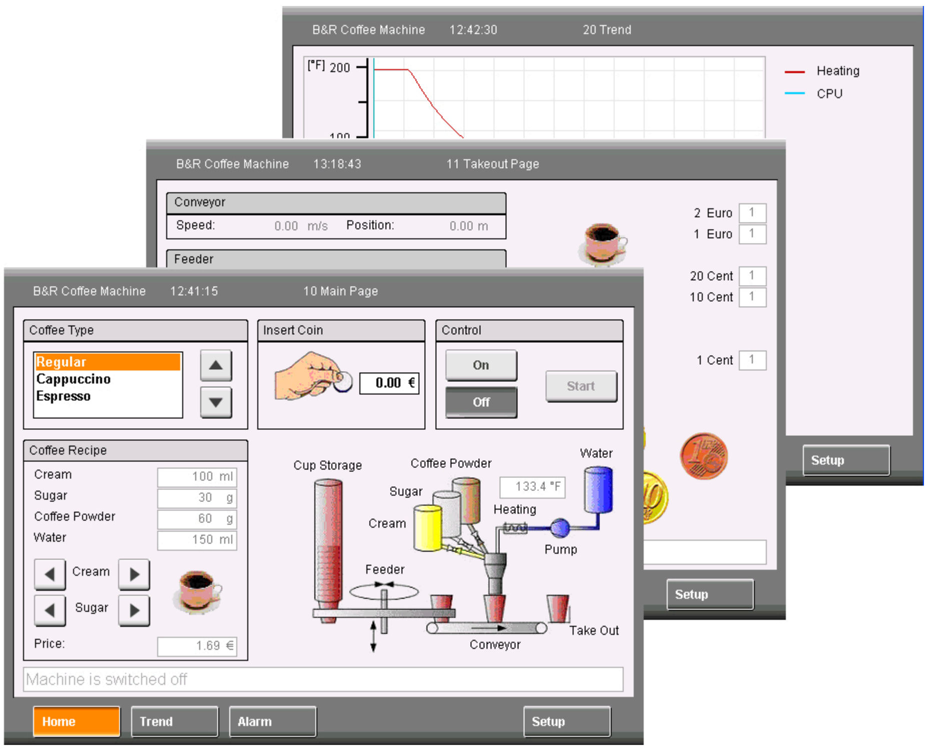
Creation of the online help files must be taken into consideration when planning the amount of work. When using Windows, there are different ways to create and call up the online help.

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

7 SUMMARY

There are many things to consider when planning and designing a visualization application. After all specifications have been fully defined, they are implemented in a software design phase. Subsequent testing and completion of all documentation are essential to success.

Several books could certainly be written on the subject of developing visualization systems.



B&R Coffee Machine 12:42:30 20 Trend

B&R Coffee Machine 13:18:43 11 Takeout Page

Conveyor
Speed: 0.00 m/s Position: 0.00 m

2 Euro 1
1 Euro 1
20 Cent 1
10 Cent 1
1 Cent 1

B&R Coffee Machine 12:41:15 10 Main Page

Coffee Type: Regular, Cappuccino, Espresso

Insert Coin: 0.00 €

Coffee Recipe
Cream 100 ml
Sugar 30 g
Coffee Powder 60 g
Water 150 ml
Price: 1.69 €

Heating 133.4 °F

Machine is switched off

Visualization Programming Guide

This training module has provided information showing the various aspects of a visualization system. Careful consideration of the different situations users experience improves ergonomics and therefore acceptance by operators.

User-friendly systems contribute greatly to profitable solutions – also with regard to machines and systems.

TRAINING MODULES

TM210 – Working with Automation Studio
TM213 – Automation Runtime
TM220 – The Service Technician on the Job
TM223 – Automation Studio Diagnostics
TM230 – Structured Software Development
TM240 – Ladder Diagram (LD)
TM241 – Function Block Diagram (FBD)
TM242 – Sequential Function Chart (SFC)
TM246 – Structured Text (ST)
TM250 – Memory Management and Data Storage
TM261 – Closed Loop Control with LOOPCONR
TM400 – Introduction to Motion Control
TM410 – Working with Integrated Motion Control
TM440 – Motion Control: Basic Functions
TM441 – Motion Control: Multi-axis Functions
TM450 – ACOPOS Control Concept and Adjustment
TM460 – Initial Commissioning of Motors
TM480 – The Basics of Hydraulics
TM481 – Valve-based Hydraulic Drives
TM482 – Hydraulic Servo Pump Drives
TM500 – Introduction to Integrated Safety
TM510 – Working with SafeDESIGNER
TM530 – Developing Safety Applications
TM540 – Integrated Safe Motion Control
TM600 – Introduction to Visualization
TM610 – Working with Integrated Visualization
TM630 – Visualization Programming Guide
TM640 – Alarms, Trends and Diagnostics
TM670 – Advanced Visual Components
TM800 – APROL System Concept
TM811 – APROL Runtime System
TM812 – APROL Operator Management
TM813 – APROL XML Queries and Audit Trail
TM830 – APROL Project Engineering
TM890 – The Basics of LINUX