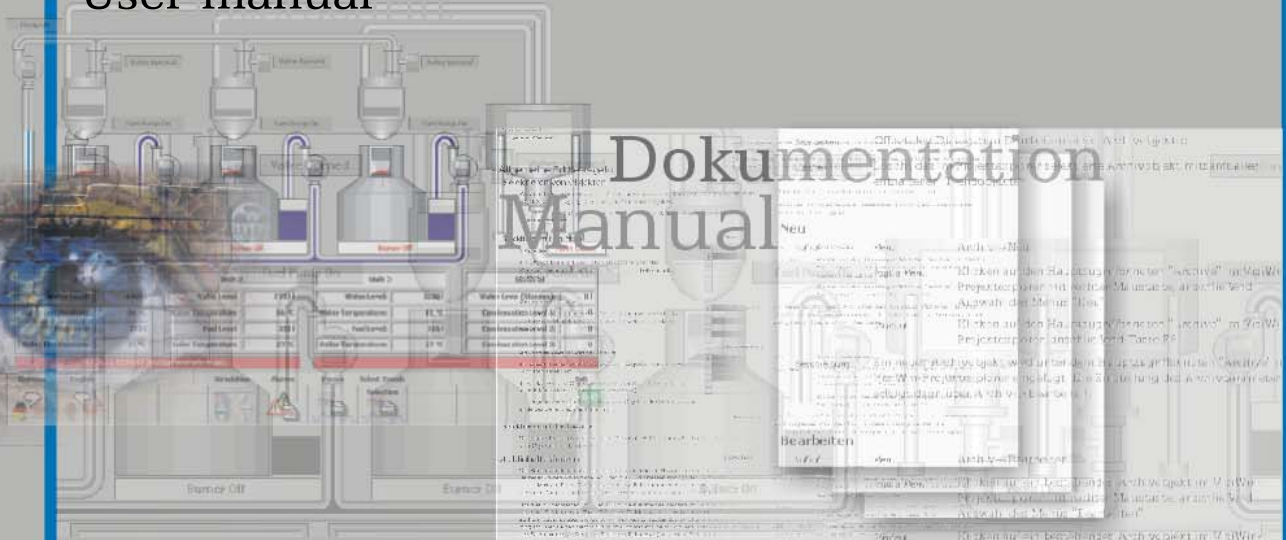


VisiWinNET 2005

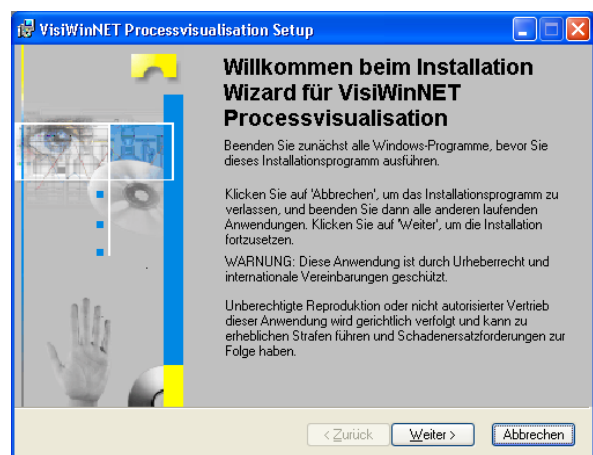
User manual



- **VisiWin**
- **VisiWinNET 2005**
- **Common**
- **Class Library**
- **Systems**
- **Tools**
- **Technical Informations**
- **Inosoft OPCServer**
- **Basics and helping tools**
- **Protocols**

VisiWinNET 2005

User manual



VisiWinNET Installation wizard







The contents of this manual must not otherwise be used without explicit written consent from INOSOFT GmbH.

We have checked the contents of this manual for compliance with the described software. Discrepancies can, however, not be ruled out. For this reason we cannot guarantee full compliance. The contents of the manual are subject to regular checking for necessary updates/amendments. Such amendments will be made in the subsequent edition.

Suggestions for improvement are welcome.

Legend

In order to point out particular paragraphs the following symbols are used in the INOSOFT documentations:

	Attention	Passages with this sign should be read – and observed – with particular attention.
	Hint	Important paragraph “additional information”
	Tip	Many roads lead to Rome; here a shortcut is to be found.
	In work	Functions that are in preparation or already implemented but not yet prepared for documentation.
	Example execute	Instructions to be carried out in an example
	Observe result	Results to be observed with carrying out the exemplary instructions

© / ™ / ®

Windows®, Windows NT®, Windows 2000®, Windows XP® are registered trademarks of the Microsoft company.

Further product names marked ® are trademarks of the appropriate manufacturers.

INOSOFT GmbH created on

VisiWinNET Version: from 6.04.000

created on 08.06.2010

Contents

1 Preamble	1
2 Welcome to VisiWinNET	2
2.1 VisiWinNET Professional Setup	5
2.1.1 Delivery	5
2.1.2 System requirements development environment	5
2.1.3 Setup instructions.....	6
2.1.4 Software protection	8
2.2 Quickstart.....	10
2.2.1 Control link with OPC.....	11
2.2.2 Control link with VisiWin drivers	15
2.2.3 Display and manipulation of process values in the application	20
2.2.4 Basic rules for application buildup	24
2.2.5 Localization.....	27
2.2.6 Alarm system.....	34
2.2.7 Archive system.....	44
2.2.8 Recipes.....	46
2.2.9 User administration	52
2.2.10 Logging.....	57
3 Projects under Windows CE	58
3.1 Project transmission from Visual Studio to the CE appliance.....	58
4 More references	62
4.1 Help for the systems and functions of VisiWinNET.....	62
4.1.1 Access	62
4.1.2 Research	64
4.2 Help for the VisiWinNET class library.....	65
4.2.1 Access	65
4.2.2 Contents of the Class Library help	67
4.3 Manual.....	71
4.3.1 PDF files	71

1 Preamble

About this manual

This manual contains introductory information on VisiWinNET, amongst others installation instructions, system requirements, and a first example explaining the interaction with the development environment.

Questions and Problems

For technical questions and problems please contact your responsible INOSOFT agent or the INOSOFT GmbH Support under +49 (5221) 16 66 02 or email: Support@INOSOFT.com

Frequent questions and problems are dealt with on our homepage under www.inosoft.com

There you will also find a support area for direct contact with our Main Office.

2 Welcome to VisiWinNET



VB.NET or C#: One programming language?

Welcome with VisiWinNET Professional, the process visualisation under Windows®.

VisiWinNET Professional is a development package for the individual design of different process visualisations. Due to the open and flexible concept very specific special functions can be implemented beside the usual standards.

The application span of VisiWinNET covers the entire panoply of commercial requirements: from the CE mini terminal via the classic single-workstation solution up to the complex client-server or SCADA/control station application everything is possible. The different product versions and the grading of runtime licenses (number of process values) allow maximum product utilization.

VisiWinNET Professional is a visualization system based on the .NET development environment. Fundamentally, the C# and VB.NET programming languages are offered. The question as to why VisiWinNET Professional requires a programming language to perform is easily answered:

A conventional visualization system provides fast and comfortable solutions to standard requirements. Process values from a PLC are to be displayed, input is to be passed on to the PLC. Additional modules allow further functions that have developed historically from the market requirements of the visualization industry, i.e. without which a visualization product cannot hold its own with the customer.

VisiWinNET meets these standards by its modular build-up. The variable kernel serves as a connection between the PLC and the visualization. The additional modules (alarm system, values recording, localization, ...) exchange data with the variable kernel if required, and make them available to the visualization surface in edited format. Display of this information is projected through the simple "placing and parameterizing" of control elements that is usual with visualization systems.

Visualization that is based on a programming language, however, enhances the extent of the functions considerably:

- The full syntactic extent of a modern programming language is available to the developer, particularly if not just the standard solution is requested.
- In view of an integrated process evaluation (vertical integration) it is important that an open interface with other systems can be used. C# and VB.NET consequently follow this approach. "Remoting" and "COM" allow network ability, data bank access is supported.

Working with VisiWinNET Professional

To all intents and purposes the programming language serves as an interface with the customer-specific project adaptation that with a standard product is either not possible at all or only through cost-intensive enhancements by the system manufacturer.

The components provided by VisiWinNET are specialized in data exchange with controls. They allow fast connection establishment between the hardware and the visualization project. Usual enhancements such as message administration, variable value recording or application language change can be projected simply and comfortably. Creating a visualization is fundamentally done in two steps:

Protecting the process database

The data required for the visualization are determined by VisiWin definitions in the VisiWinNET editors. Here it is ascertained which cells in the PLC memory are required for the visualization, and which special functions (alarm/trend recording...) they are to fulfill.

Operating surface

The design of the operating surface is determined by projecting forms and dialogs. The control elements provided by VisiWinNET have full access to the process database. Variables can be displayed or described. The alarms and trend recording special functions are supported by special control elements. In addition VisiWinNET control elements support the display of localizable texts. With this an inter-nationally usable application can be built up..

Improvements compared to VisiWinStudio

VisiWinNET is the consequent advancement of its predecessor VisiWinStudio. The .NET framework used as a basis offers, however, enhanced facilities:

- **Compatibility:** Microsoft promises platform independence to the greatest possible extent by the .NET technology. An individual framework is provided for each different platform with a uniform program interface that controls the adaptations to operating system and hardware. With this even Windows®-CE appliances can now for the first time be treated quasi seamlessly compatible with the same development tools as a standard PC.
- **Freedom of choice:** C# is now also available beside VB.
- **Version independence:** .NET does rigorously away with the version conflicts of the past. Components used by different applications can now again be installed in the application index. With this the flawed component registration loses its relevance. A common system index further exists in the "Global Assembly Cache" but is made less dangerous by the enhanced version management: Here files of the same identity can be parallelly installed with different versions. An

application is now tied much more closely to the appropriate component version than previously.

VisiWinNET was from the beginning developed to support the above mentioned advantages:

- The systems and components of VisiWinNET in its individual versions (Enterprise/Standard/Compact) are to the largest extent identical in function and operation.
- C# and VB.NET are supported as programming languages.
- Future versions with enhanced functions can be installed parallelly to existing versions. In particular with a longer-ranging development the projector can now decide whether to venture an update for new features or maintain the current project status.

2.1 VisiWinNET Professional Setup

2.1.1 Delivery

The VisiWinNET® program package includes:

- CD VisiWinNET Version 6.xx
- Software-protection (development dongle, licensed version only)

2.1.2 System requirements development environment

Computer	IBM-compatible PC
CPU	Intel Pentium/Celeron family or comparable CPU of min. 500 MHz clock rate; 1 GHz or more recommended-
RAM	Min. 512 MB or more recommended.
Graphics	Min. 800 x 600 pixels, High Color (16 bit); 1280 x 1024 pixels, True Color (32 bit) or higher recommended.
Hard Disc	200 MB free space, additionally 280 MB free space if the .NET framework 2.0 has not yet be installed.
CD-ROM Drive	For installation only.
Interfaces	For CE projects: Ethernet network card for communication with the target appliance; alternatively free serial port or USB interface. In addition a free parallel port or USB interface for licensing VisiWinNET through a hardware dongle.
Operating System	Microsoft Windows 2003 Server/XP SP2/Vista.
Software	Microsoft Visual Studio 2005 (from Visual Basic/C# Standard. Attention: the Express Edition is not sufficient). Optional*: Microsoft .NET Compact Framework 2.0 SP1, Microsoft .NET Framework SDK 2.0, Acrobat Reader, Microsoft ActiveSync.

*Optional software products that are not available in the installation computer can be obtained free of charge from the manufacturers' internet websites. If these products are not installed some enhanced functions of VisiWinNET will not be available.

For the installation further free space on the hard disc is required.

2.1.3 Setup instructions

After insertion of the VisiWinNET-CD in the drive respective, the setup program will automatically be started.



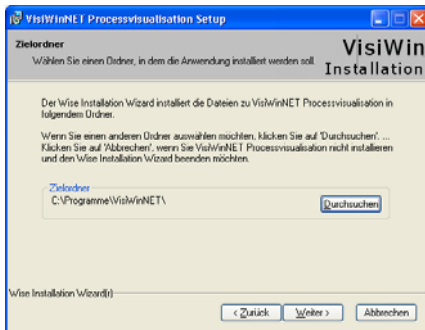
Please consider that Visual Basic is to be set up already! Here, the setup-program saves the path of the form design tool, which is to be called through the project manager. On the VisiWinNET-CD, in the directory "VB6Demo" a functionally restricted test version of Visual Basic is included for free. If Visual Basic was not set up before the VisiWinNET-Installation, so for testing this test version can be set up.

The setup wizard leads the user through the single steps of installation. There, the button "Next" leads to the next step.



Welcome

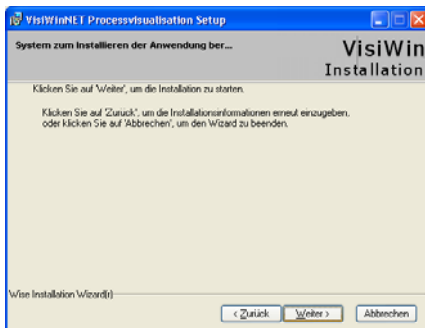
The "Welcome" dialog describes the functionality of the installation assistant and the "continue" button.



Select target indices

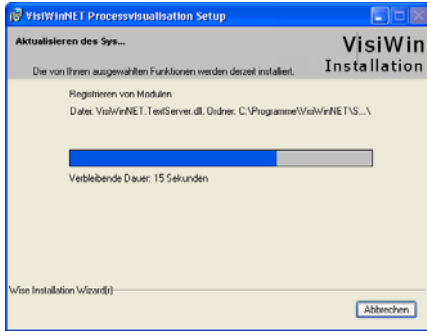
The installation path describes the index into which the VisiWinNET files are installed.

The "browse" buttons open a dialog for comfortable index selection.



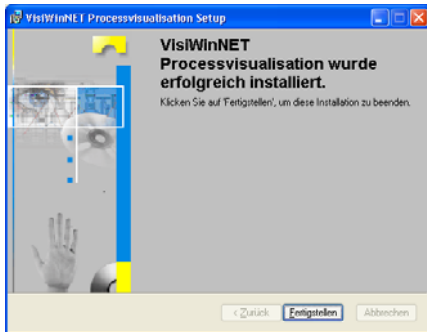
System ready for application installation

The program reports that is it ready for installation.



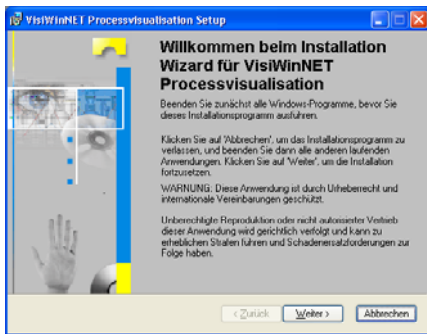
System update

During the installation the progress as well as the time remaining until completion of installation are displayed.



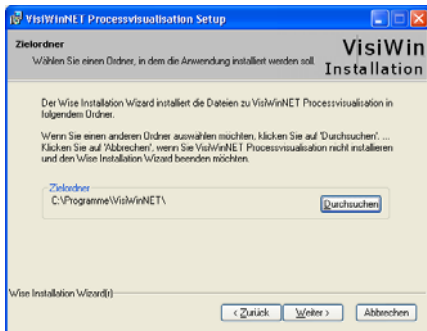
VisiWinNET process visualization was successfully installed

Completion of installation is reported.



Welcome

The "Welcome" dialog describes the functionality of the installation assistant and the "continue" button.



Select target indices

The installation path describes the directory into which the VisiWinNET files are installed.

The "browse" buttons open a dialog for comfortable index selection.

2.1.4 Software protection

VisiWinNET is subject to licence agreements. The development environment as well as the runtime is bound by number of pieces and efficiency extent charged. The software is protected from abuse as well as from using it free of charge. The customer can select between the various protection measures that differ with regard to hardware resources and large-scale serial production:

Protection	Description
LPT-Dongle	Hardwarelock for printer interface
Bus-Dongle	Hardwarelock for PCI-slot
USB-Dongle	Hardwarelock for USB-Interface
Lic-File	Contract-dependent release through licence file

The hardwarelocks differ in the efficiency extent

Development dongle Release of development environment and runtime system with following restrictions:

- 20000 Process values can be exchanged with the controls.
- A Client-Server-project runs 4 hours. Afterwards restart will be necessary.

Runtime dongle The runtime hardwarelock is staggered dependent to efficiency extent and project type. There, the number of process values to be exchanged between PLC and visualization, the client-server constellation and the Product Type (Standard, Compact) is decisive. The staggering is to be taken from the price list, together with the prices.

Toolbox option The tools option will be implemented in the runtime dongle. It enables project database change on the target system. Thus, changes of message definitions and multilingual text can be done directly by the customer.

Restrictions without Hardwarelock

Without hardwarelock VisiWinNET can be utilized restrictive only. The development environment as well as the runtime is in "Demo-Mode" then. This mode enables start and restricted editing of the Demo-projects supplied.

The demo-mode contains following restriction:

- Without Dongle only one project under the name "VWNTTest" can be created and edited in the development environment.
- The project explorer in the development environment displays the demo mode in the status line.
- With every start of the development environment a notice points out the missing software protection.
- The runtime system switches off after one hour.

2.2 Quickstart

The following chapter provides – as a practical example – an overview of the functions of the development environment, the runtime system, and the individual systems that carry out special visualization-typical tasks.

The test situation describes below allows the setup of a fully functional application. To keep this simple the structures partly follow the logical buildup of VisiWinNET, partly the practical basics of visualization development.

Control link with OPC servers or VisiWin drivers

Establishing communication with the PLC:
VisiWinNET supports two different kinds of communication components:

- OPC servers are now available as standardized communication components for nearly every control, and do not tie the projector to a specific visualization product as meanwhile nearly every visualization package handles the OPC standard.
- VisiWin drivers are the result of long years of development in particular for customers of long standing or those coming from older products (VisiWin32/VisiWinStudio). They continue being available as an alternative to OPC servers, where appropriate also for controls for which no OPC server exists, and for which no driver was created within the scope of a special development.

The example shows both link alternatives using the demonstration components supplied with the product.



The "Compact" and "Standard" runtime systems each support only part of the VisiWin drivers:

- The "classic" VisiWin drivers are not supported by the "Compact" runtime.
- The newer generation of VisiWinNET drivers can also be used under Compact.

Display and manipulation of process values in the application

The application is filled with life. First changing process values are displayed in different ways. Writing of process values is achieved using special control elements.

Basics of application buildup

Here it is shown how an application is built up modularly. Differentiation and change between the individual display modes is explained.

Localization

The localization interface is filled with data, and brought to life.

Alarm system

Detecting and displaying error states

Archive system

Meaningful long-run monitoring of process values

- Recipes** Display, optimization and transmitting to the machine of parameter sets
- Logging** Everything must be reproducible. Recording of user interaction and PLC values for a machine chronicle.
- User administration** Different users must only be able to obtain partial access to the components of the application.

2.2.1 Control link with OPC



In this chapter setting up a new OPC server and importing OPC items into the VisiWin test project is described.

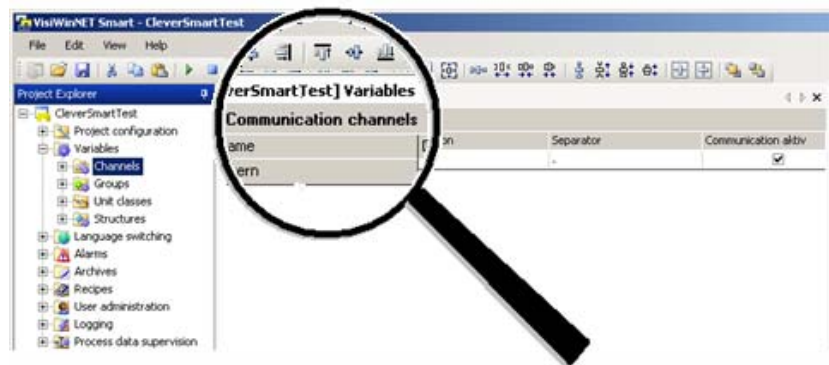
An important component of a visualization under VisiWinNET is the projection of the data access. Here it is determined which data from the PLC are required, with which communication settings they are exchanged, and, if required, how these data are named in the visualization. Data access is achieved in the variables editor. With the variables editor definitions are set that determine which process values are to be exchanged with the PLC. The variable kernel (pivotal component for the exchange of process values between PLC and visualization) interprets these definitions at runtime, and establishes the appropriate contact with the communication components. The variables editor is represented by the "Variables" node.



The "Variables" node in the VisiWinNET Project Explorer is to be expanded. Through a click on the "Channels" node beneath the table editor of the variable editor is to be activated.



If in the Project Explorer the "Channels" node is highlighted all communication channels of the project are listed in the table editor.






In the table editor the context menu is to be opened (click with r.h. mouse button), and the "New" entry to be selected.

This opens the "Add channel" dialog. Here basic properties of the new channel are determined:

- Name: A freely chosen name for the channel. The name must be unequivocal within the communication channels of the project. References to process variables in the applications incorporate this name (example: "Ch1.w0" nominates the "w0" variable in the communication channel with the name "Ch1").
- Channel type: either OPC-Server or Driver. This setting is to be set to "OPC-Server".

OPC Server / Driver: through the  button the dialog for the selection of a communication component is to be opened. Here the entry "VWOPC Server Demo Machine & Testing" is to be selected under the "Desktop→OPC Server" node.

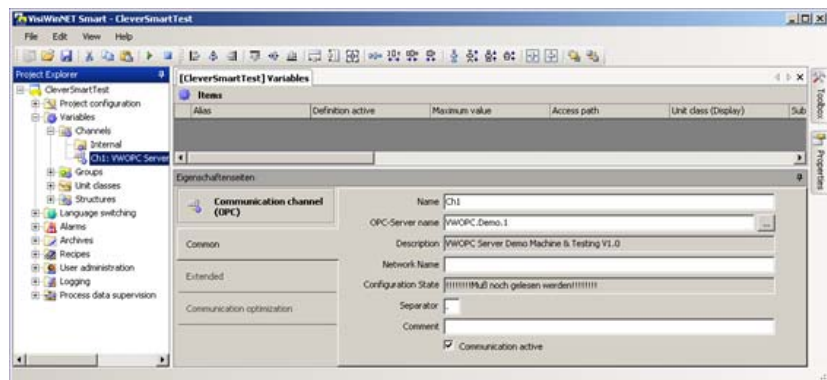
After closing the dialog click on the "Channels" node in the project explorer.



Highlighting the "Channels" node has the effect of the entered data being stored in the project database.

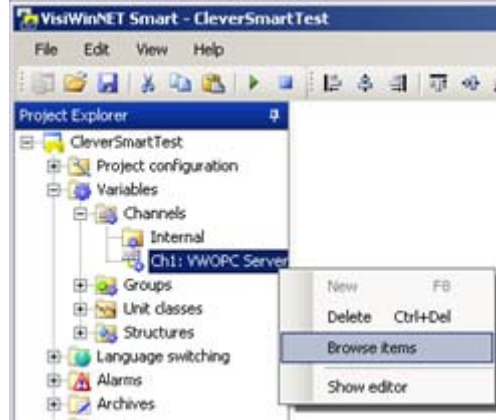
The new communication channel is now represented in the Project Explorer by a node.

If the node is highlighted the VisiWinNET properties page is automatically displayed. Here the full parameter set of the communication channel is provided for editing.





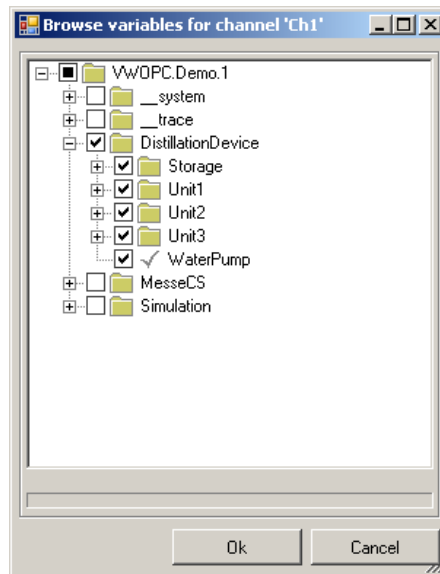
Through the context menu of the communication channel the "Browse" function is to be activated.



A window with the name "Browse variables for channel 'Ch1'" opens.



Through the "Start" button the reading of item definitions from the OPC server is to be initiated.



The item browser displays all item definitions in the OPC server hierarchically. The name of the OPC server is in the top hierarchy, the individual items are displayed in the bottom hierarchy. In between there are the so-called "Branches". These do not represent objects but only show the structuring of items in namespaces.

Item designators are separated by a separation character (similar to the "\" character in the full path definition of a file). Items that begin with the same designator parts are pooled under a common branch.

If a namespace is activated through the control box all namespaces and item designators beneath are also activated.

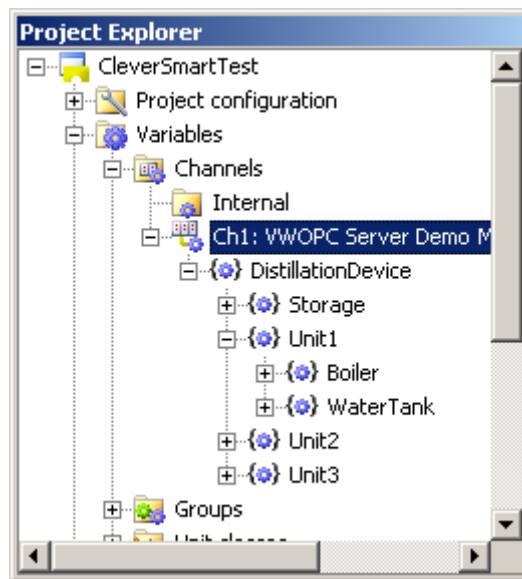
By selecting a variable designator the appropriate variable definition is assumed by the VisiWinNET project database.



The "Distillation Device" node is to be selected as shown in the graphic. The selection is to be confirmed through the "OK" button.



The selected items are assumed by the project database as variable definitions. The display in the Project Explorer is restricted to the hierarchic display of namespaces.



The variable definitions are displayed in the table editor together with their parameters.



A click on a namespace opens the table editor in the development environment (above the design area).

[CleverSmartTest] Variables					
Items					
Alias	Definition active	Maximum value	Access path	Unit class (Display)	Substitution value mode
DistillationDevice.Unit1.Boiler.BurnerOn	<input checked="" type="checkbox"/>	0		(no Unit class)	
DistillationDevice.Unit1.Boiler.FuelTank	<input checked="" type="checkbox"/>	0		(no Unit class)	
DistillationDevice.Unit1.Boiler.SwitchOn	<input checked="" type="checkbox"/>	0		(no Unit class)	
DistillationDevice.Unit1.Boiler.Temperature	<input checked="" type="checkbox"/>	0		(no Unit class)	

Here the variable definitions of the namespace just selected are listed in table form. For the example described here it is, however, sufficient to assume the variable definitions from the OPC server. Further adjustments are not necessary and the table editor can be closed again.

2.2.2 Control link with VisiWin drivers



In this chapter the setup of a new VisiWin driver together with the appropriate variable definitions is described. Also, the function of structures is explained.

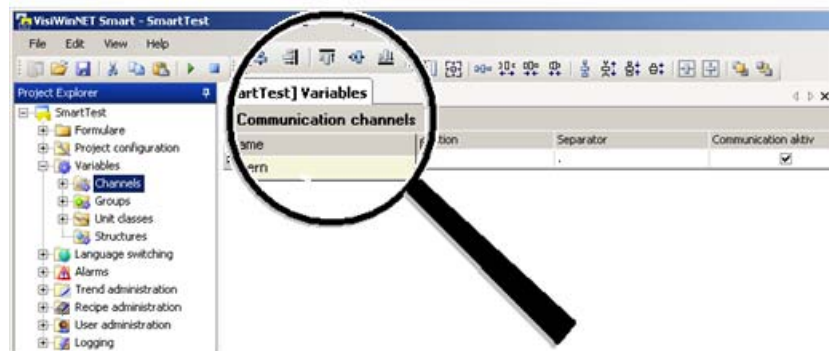
VisiWin drivers are an alternative to OPC servers. They are customarily used if no OPC server is available for a control, if very special customer-related requirements are to be fulfilled or if existing links from older VisiWin visualization packages are to be adopted without changes.



The "Variables" node in the VisiWinNET project explorer is to be expanded. Through a click on the "Channels" node beneath the table editor of the variable editor is to be opened.




If the "Channels" node is highlighted in the Project Explorer all communication channels of the project are listed in the table editor.



In the table editor the context menu is to be opened (click with r.h. mouse button), and the "New" entry to be selected.

This opens the "Add channel" dialog. Here basic properties of the new channel are determined:

- Name: A freely chosen name for the channel. The name must be unequivocal within the communication channels of the project. References to process variables in the applications incorporate this name (example: "Ch1.w0" nominates the "w0" variable in the communication channel with the name "Ch1").
- Channel type: either OPC-Server or Driver. This setting is to be set to "VisiWin-Driver".

OPC Server / Driver: through the  button the dialog for the selection of a communication component is to be opened. Here the entry "VWDummy SPS-Simulation" is to be selected.

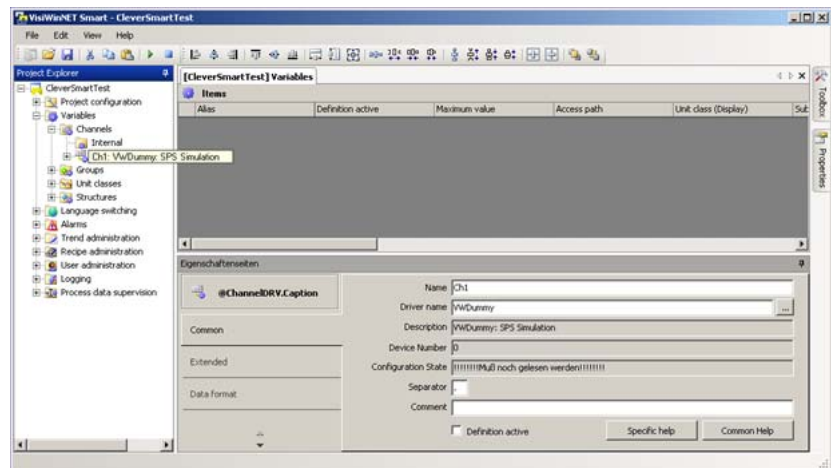
After closing the dialog click on the "Channels" node in the project explorer.



Highlighting the "Channels" node has the effect of the entered data being stored in the project database.

The new communication channel is now represented in the Project Explorer by a node.

If the node is highlighted the VisiWinNET properties page is automatically displayed. Here the full parameter set of the communication channel is provided for editing.



With this the selection of the communication component is finished. Through the definition of the communication channel the contact with the communication component has been established. Subsequently, the process values to be exchanged have to be determined. This is done by the definition of variables.

Variable definitions in drivers cannot be created automatically. They need to be defined manually by adding individual data sets in the table editor.

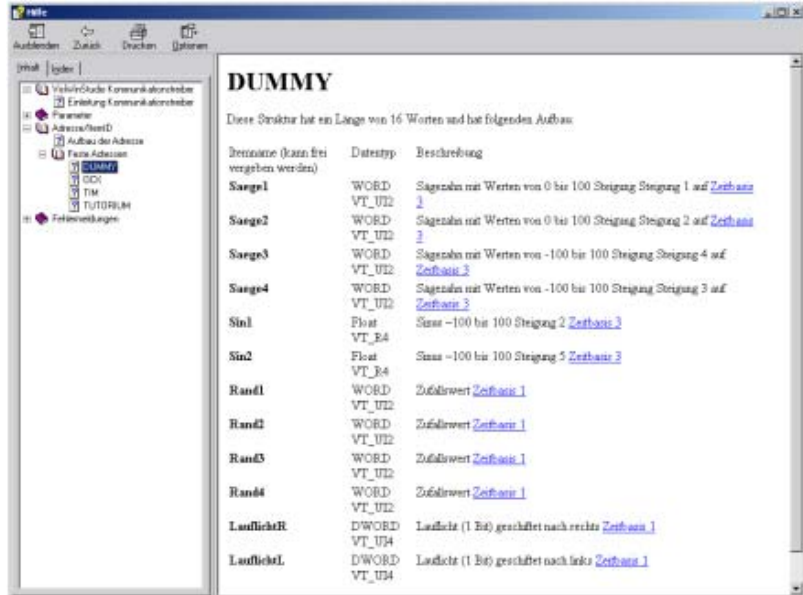
A specific characteristic are the addresses through which the driver exchanges data with the PLC. To every variable an address and (through the data type) a length must be allocated. In addition, addresses can be used – depending on the PLC type – symbolically or directly. Hence different drivers expect different address information. The information which syntax is followed by the specification of an address is documented in the specific driver help. The fastest way to this help is through the "Specific Help" button in the channel properties.



Through the "Specific Help" button on the "Standard" tab the driver help is to be opened.



All driver helps contain a paragraph about the buildup of the specific address indications (Address/ItemID)



What with a real PLC is only determined by programming is already firmly set with the dummy driver. Via the addresses specified here simulated variable values can be called up. The "Address/ItemID→firm addresses→Dummy" paragraph describes the data structure of the "Dummy" address.



Why "Data structure of the Dummy" address? Are "Saw1", "Saw2" etc. not addresses?

No, they are not. Drivers fundamentally have the possibility of exchanging data with the control, individually or by block. The definition of a variable as basic data type "VT_..." allows to access a single variable. The definition of a variable of the data type of a structure is a data exchange by block. Within a block there are the individual variables whose size is determined by the data type. A variable for example of the VT_14 data type has a firm size of four bytes to be transmitted. The transmission of a block only requires a control-specific start address and the size of the data block that with structures can be computed through the sizes of the individual elements.

A structure is formed from individual elements that can be again structures or basic data types. Structures offer the advantage that many data can be read or written with a minimum protocol overhead. This is in particular imperative if the PLC interface is slow (e.g. 9600 Baud). Through structures the communication can be optimized. There is no need to send a new protocol frame for every transmission of individual PLC variables. Through one single protocol frame for example a whole data block can be transferred with the appropriate projection. The breakdown into the individual elements or variables is done by the runtime system in the application.

Structures are defined in VisiWinNET.



Through the context menu of the "structure definitions" node a new structure is to be defined.

The name of the new structure is to be changed to "sDummy" on the VisiWinNET properties page.

In the table editor of the new definition 12 new structure elements are to be created (click on empty table, then press "F8" 12 times).

The parameters of the new structure elements are to be changed as indicated in the help:

Name	Datentyp
Saw1	VT_UI2
Saw2	VT_UI2
Saw3	VT_UI2
Saw4	VT_UI2
Sin1	VT_R4
Sin2	VT_R4
Edge1	VT_UI2
Edge2	VT_UI2
Edge3	VT_UI2
Edge4	VT_UI2
Running LightR	VT_UI4
Running LightL	VT_UI4



Finally it is to be checked whether the "Order number" parameter is ascending consecutively in the above sequence.



The structure built here describes the buildup of a data block. The structure itself is, however, not yet a block that itself exchanges data. It is only a type declaration. This declaration can be used as a data type in the variable definitions.



Change to the "Ch1: VWDummy..." node in the project explorer.



The display in the table editor shows the variable definitions of the channel. As no variables have been defined yet the table is empty.



Click on the empty table editor, and create a new variable via the "F8" key. The following values are to be changed in this new variable:

Parameter	Value
Name	DummyItem
Address	Dummy
Data type	sDummy



The variable name can be freely chosen. A purpose-related name pointing to the function of a variable makes the projection during the following development steps considerably easier.

The specified address relates to the driver function described in the help.

The defined structure is offered in the selection list as a data type. The definition of a variable of the structure data type effects the following procedure at runtime:

During the initializing phase the variable kernel loads the variable definitions from the project database, and builds up the appropriate core image. For this purpose it identifies the buildup and size of the structures specified as data types.

With the request of values from the application the variable kernel sends the start address and the size of the variable. In the above example the size of the requested data block equals 32 byte.

The driver reads the indicated number of bytes from the passed start address with a protocol access, and returns the byte block to the variable kernel.

The variable kernel now allocates the individual bytes internally to the appropriate variables in the core image.

2.2.3 Display and manipulation of process values in the application

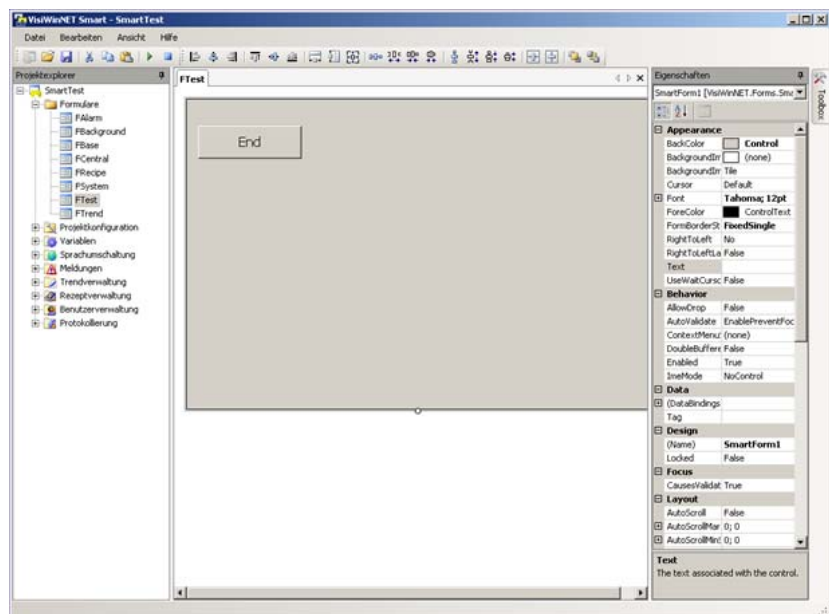


This chapter describes the interaction with control elements in general, and explains the connection with the variable definitions from the project database

After setting up a new project at least one form has already been created. A form represents a screen display or a dialog.



To open the design view of the form the "Forms" node is to be expanded. Under this the forms of the project are listed. A double click on a form name opens the designer.



The designer shows a screen display. The appearance of the screen display is determined through the placing and parameterization of control elements. Control elements are selected in the toolbox. In the basic setting the toolbox is displayed top right as a tab.



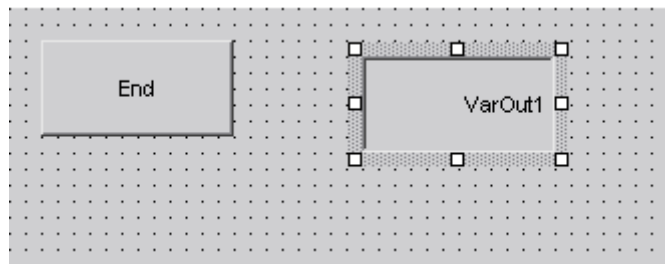


The toolbox is to be expanded through the appropriate tab. The "VarOut" control element type is to be marked in the toolbox.

Subsequently the position on the form where the control element is to be placed is to be selected by a click with the l.h. mouse button. If the mouse button is immediately released again the control element will be placed in this position with a predetermined size. If, however, the mouse pointer is moved with the button pressed the movement on the screen forms a frame. Releasing the mouse button effects the control elements being depicted on the form with the size and in the position of the displayed frame.



Subsequently the "VarOut" control element appears on the form.

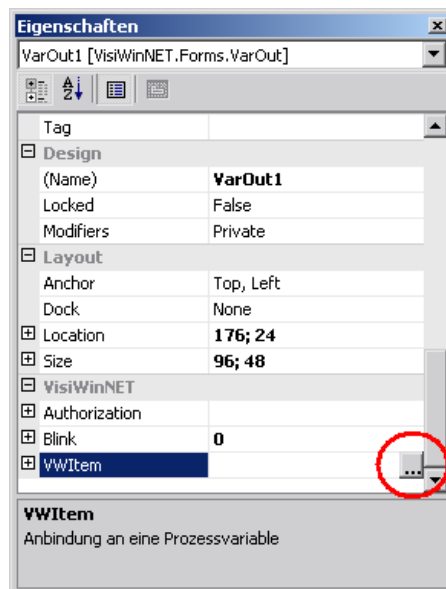


The selected control element type serves the numeric or alpha-numeric display of variable values. The essential determination which process value is to be displayed is made through the "VWItem" property.



The control element is to be marked with a mouse click so that – as shown above – the selection frame becomes visible.

Scroll within the properties window through the scroll bar until the "VWItem" property shows. The empty values field of the property is to be marked with a mouse click.



Within the values field of the property a small button appears through which a dialog for selecting a variable can be opened.

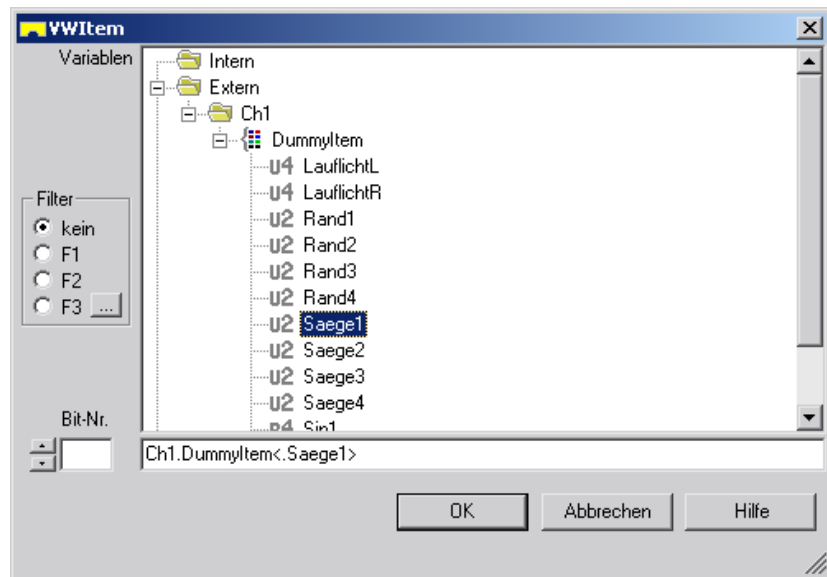


The selector button shown above is to be operated by a mouse click.

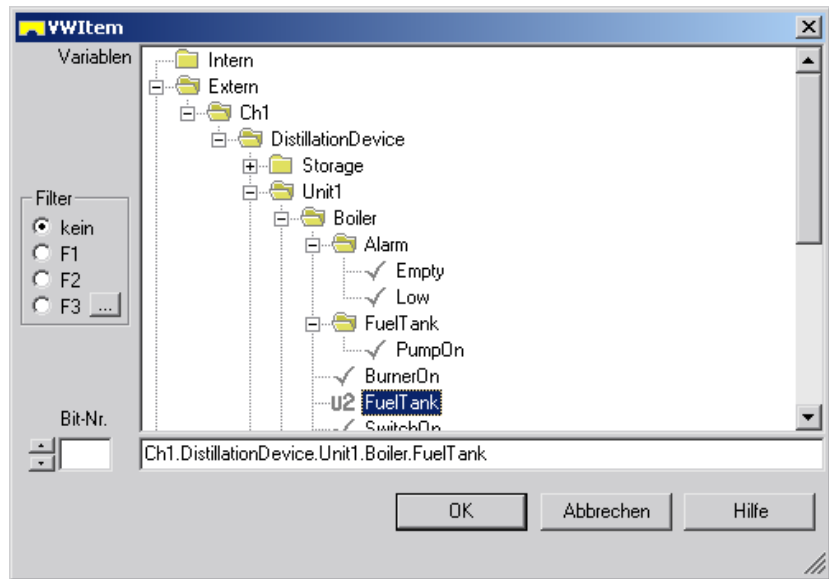


The previous chapters describe the linking through an OPC server and a VisiWin driver. Depending on the selection the dialog shows the following content:

For VisiWin drivers...



...or for OPC servers



The dialog shows the available process variables that were stored in the project database by browsing or own definition. Basically the buildup of the hierarchy shown here follows the buildup in the project explorer.

- "Internal" characterizes variables that are not allocated to a communication channel, i.e. that have no real image in a control. These variables can still be required within a visualization, e.g. to trigger variable-controlled processes or simply as caches for values from the visualization.

"External" contains the variables that have a PLC image. Under external the communication channels are listed. Hereunder are the namespaces that were also already used in browsing. They are followed by the variable definitions. With drivers the structure elements, too, are provided hereunder where appropriate.

The variables shown in the above images are to be selected. That selection is to be confirmed through the OK button.



By setting the 'VWItem' property the display in the "VarOut1" control element changes. With the determination of the variable to be displayed the data type is interpreted. The display in the control element jumps to a value that simulates the display at runtime. With this it is possible to check even at development time whether the control element can show the full value at all or whether the size of the control element has to be changed if appropriate.

A final test shows the operability of the link.

The test run of a standard project is simple:



In the tool bar the  button is to be activated.



The project starts. The display shown now corresponds with the display as shown to the visualization user. If the function is successful a changing value should now be shown in the display.

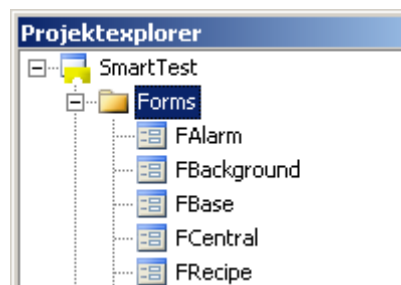
VisiWinNET provides a number of further control elements that allow display or input graphically or alpha-numerically. Common to all these control elements is the 'VWItem' property that selects the process variable to be displayed or written. This means that the procedure is the same or at least similar with all control elements. For an overview of the functions of a control element this can be put on a form, selected there, and the help function accessed through the "F1" key.

2.2.4 Basic rules for application buildup

Rarely a visualization will get by with only one screen display. Usual and useful is the division into different screen displays following the function or logic of the production line. Starting from a central overview page the user can then through buttons change to the application parts that he wants for display or input. To be observed in the process is the person-related release of functions. Usual practice in this connection is for example a service page that cannot be accessed by the operating personnel but only by a service engineer.



In practice the different screen displays are designed through forms. Forms are added in the project explorer via the "Forms" node.



A new form is to be added to a project. For this the "Forms" node in the project explorer is to be marked, and the context menu to be accessed with the r.h. mouse button. In the following dialog the name "MyFirstForm" is to be entered for the form.



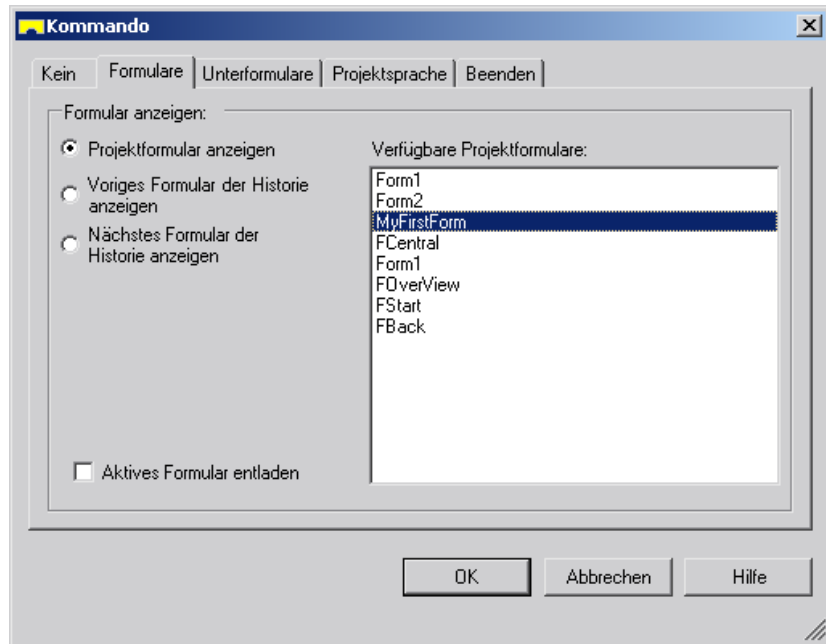
A new node with the name "MyFirstForm" is added under "Forms". An appropriate form designer is opened. In this designer control elements can again be placed and parameterized.

This leaves the answer to the question how the new form is displayed.



For this change to the first form of the application, and place a control element of the "CommandButton" type on this form.

Open the properties dialog of the "Command" property of the "CommandButton" control element. Select the entry "MyFirstForm" on the "Forms" index card in the forms list. The selection is to be confirmed through the OK button.



The function of the "CommandButton" established here effects the form "MyFirstForm" to displayed at runtime.



The test through the button shows that the form is displayed, however, with a title line and the usual function buttons.

The most used screen display of a visualization application is a full screen display without a title line.



After ending the application the "MyFirstForm" form is to make visible again as a form designer.

The following properties are to be changed in the properties window:

Property	Value
FormBorderStyle	None
WindowState	maximized



As the form with the properties set here no longer has a title line it cannot be closed without a button. It can also not be moved aside as it is maximized, and fills the whole screen.

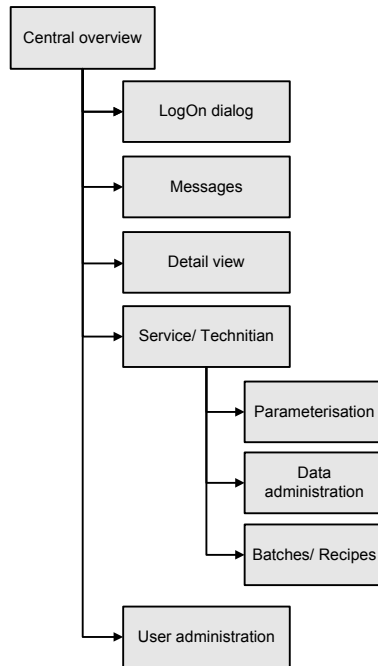


A "CommandButton" is to be placed on the form. Die "Command" property is to be parameterized so that it again calls up the first form of the application.



The test run shows that it is now to change between the two forms. Via the "CommandButton" the individual pages can be called up.

An essential aspect with the buildup of a visualization is the separation of the screen displays. A typical example for this is shown here:



Every one of the function blocks shown here is to be depicted by one or more screen displays. A hierarchy built up this way shows the basics of how the pages are to be linked, i.e. which pages are to be accessed from where. If for example access is to be made from the central overview page to the individual function blocks thereunder then at least 5 buttons are required. Access to the data by the service engineer does not have to be from the central overview page if one screen display contains access to the "Parameterization", "Data Maintenance" and "Recipes" blocks. It is down to personal taste to decide whether screen displays parallel to each other need to be linked with each other. An exception, however, is certainly made by the alarms display page that should in any case be accessible from all parts of the application.

2.2.5 Localization

The localization allows the change of all texts used in the application in dependence on the selected language. It is used for the following purposes:

- To make it possible to supply internationally usable applications. Here it would be sufficient to configure the language to be displayed with the project start or on a service page.
- To allow international operating personnel to work with the application. It can for example be imagined that the language is changed with a shift change as the nationality of the personnel changes.

Both is quasi possible by pressing a button. Even the input of non-Western characters is allowed through Unicode character sets.

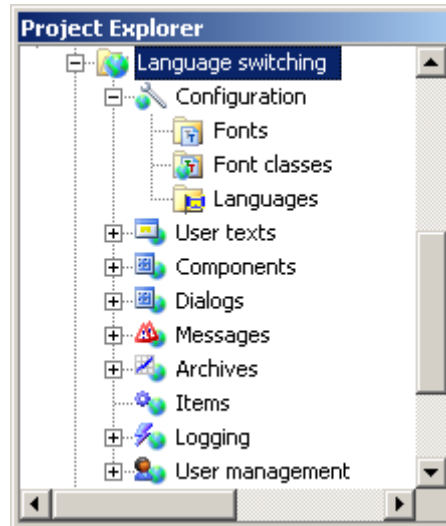
Beside the texts themselves that are used in the application further particularities are to be observed:

- Dependent on the selected language units (e.g. temperature °C/°F) need to be converted and changed.
- Date and time formats must be shown in the country-specific formats, again dependent on the selected language.
- Where applicable font sizes may be standardized differently with different font sets so that adjustments may be necessary with language changes.

Text specification is in the VisiWinNET localization. Linking with the control elements is through different properties.



The "Language switching" node in the project explorer is to be expanded.



The functions of the localization are separated into different areas:

- "User texts" are the texts that the projector can create to be displayed in the application.
- "Components" and "Dialogs" contain the texts that are used and expected by the control elements and dialogs of the VisiWin package. Here no texts can be created; the texts can be linguistically enhanced or changed application-related.
- In the "Variables", "Alarms", "Archives", "User Management" and "Logging" nodes no texts can be created. That is done by the editors of the appropriate systems. If for example an alarm is created in the alarm system the localizable text of the alarm appears under the indicated node. This complies with the central administration philosophy of the localization.

The "Configuration" node contain the following sub nodes

- "Font classes" and "Fonts" define the function of the font change.
- "Languages" contains the project languages as a collection. A new project contains German and English as project languages. Further languages can be added to the project.



The unit change is contained in the variables system. It has been conceptionally separated from the localization as the information to be defined contains conversion values as a main characteristic.



The "User texts" node is to be marked. Through the context menu a new text group is to be created. The text group is to be renamed "MyFirstForm".



Localizable texts can be created and administrated in individual groups. This has the following advantages:

- Through the name of the text group it can be specified where the texts are used. This means that a structuring is possible through text groups.
- The texts that are to be created and perhaps changed at a later stage can be retrieved more easily.
- If a text group is created for every form this makes life easier later. If the form is adopted by another project only the appropriate text group needs to be copied into the project. Then the form that uses the texts is again usable.




In the "Texts" table of the table editor a new text is to be added through the "F8" key.

The text is to be adapted as follows:

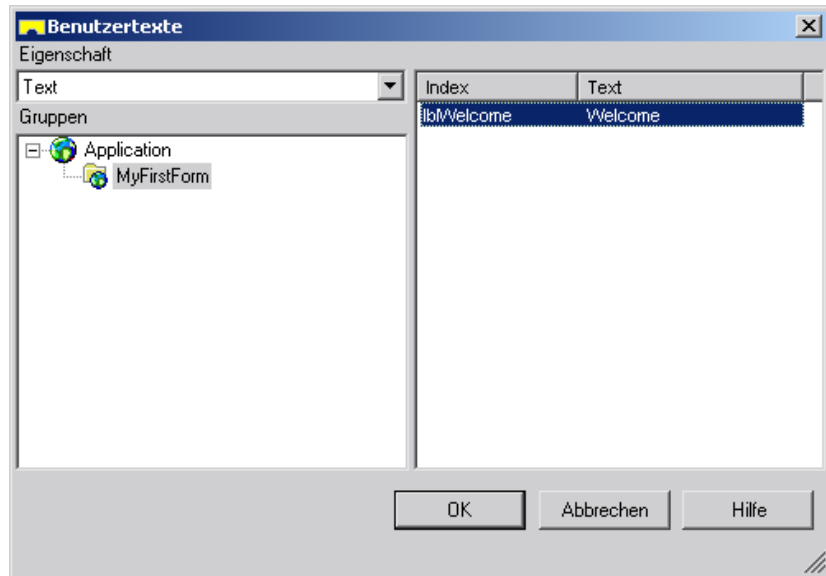
Parameter	Value
Name	lblWelcome
Deutsch (Deutschland) (1031)	Willkommen
English (USA) (1033)	Welcome

Via the appropriate node under group "Forms" in the project explorer the form "MyFirstForm" is to be opened. Here a 'Label' and two 'CommandButton' control elements are to be placed.

The 'Label' control element is to be marked. Through the  button of the 'LocalizedText' property the dialog to select a localizable text is to be opened.



Here all user texts provided in the project are offered for selection.




Navigate to the 'MyFirstForm' text group in the text groups display (l.h.). Mark the "lblWelcome" text in the text selection display (r.h.). Confirm the selection through "OK".



Through the selection it is determined which text is to be displayed by the control element at runtime. After adoption the text is displayed in the control element in the active language.

The two 'ComandButton' control elements are for changing the language at runtime..



The first 'CommandButton' control element is to be marked. Through the  button of the 'Command' property the function select dialog is to be opened. The entry "Deutsch (Deutschland)" is to be selected on the "Project language" index card. The selection is to be confirmed through the "OK" button.

The procedure is to be repeated for the second 'ComandButton' control element. In the process the language "English (USA)" is to be selected.



The so-called "Local Identifiers" (LCID) are shown as property values. Every language has such a key under Windows. "1031" stands for German, "1033" for English.

The test run shows that the language is changed in the "Label" control element through the buttons.

The language extent of texts can be expanded. New project languages are added through the "Localization→Configuration→Languages" node.

When a new language is added the text tables are also extended by one column each.



In the Project Explorer the "Localization→Configuration→Languages" node is to be highlighted. A new language is to be added to the table editor with the following parameters:

Parameter	Value
Language	Danish (Denmark)
Locale Input	Danish (Denmark) Danish (Denmark)



The specified input area scheme does not exist!?!

Input area schemes assist the operating system in changing the country-specific or special characters on the keyboard. A German keyboard for example contains the "ä", "ö" and "ü" characters that do not exist in the English language. With English keyboards these keys are engaged with other characters. The transmitted key code is, however, identical. The operating system controls the allocation of the keys to the characters displayed on the keys through a country-specific table, die "input area scheme".

Required input area schemes must be installed. For this, change to the system control of the computer (Windows Start Menu→Settings→Region and Language Options→"Languages" index card→"Details" button). Here the "Danish" language is to be added. Where appropriate, components need to be installed from the Windows CD on request.

Subsequently, the input area scheme is present in the selector list of the table editor field, too.



Muppet Lab: Please never show the following setting to a Dane!



The word "Sm'rebr'd" is to be entered into the Danish column of the 'lblWelcome' text.



In the process the " " character is replaced by the typical Danish character. As we can see setting an input scheme allows to display characters that can otherwise not be accessed through the keyboard.



To complete the example a third 'CommandButton' control element is to be placed on the "MyFirstForm" form, and the new language "Danish" to be configured through the 'Command' property.



If for example a particular font display becomes necessary for the Danish language this is possible through the "Fonts" and "Font classes" definition types.



Mark the "Font classes" node in the project explorer.




The font classes shown in the table editor have names, and contain a font for every project language.

By adding a new language there has also been added a new column "Danish" in the font classes table. The values are not filled but can be engaged with the font definitions..

If a control element is linked with a font class it changes to the appropriate font with a language change.



The field "Danish" of the "Labels" font class is to be marked. The font "Tahoma8BoldWestern" is to be selected in the selector list of the field.

The 'Label' control element is to be marked on the form "MyFirstForm". Via the  button of the 'FontClass' property the dialog to select a font class definition is to be opened. Here the "Labels" font class is to be selected. The project is to be started.




The test run shows that now also the font changes when the language is changed to Danish.

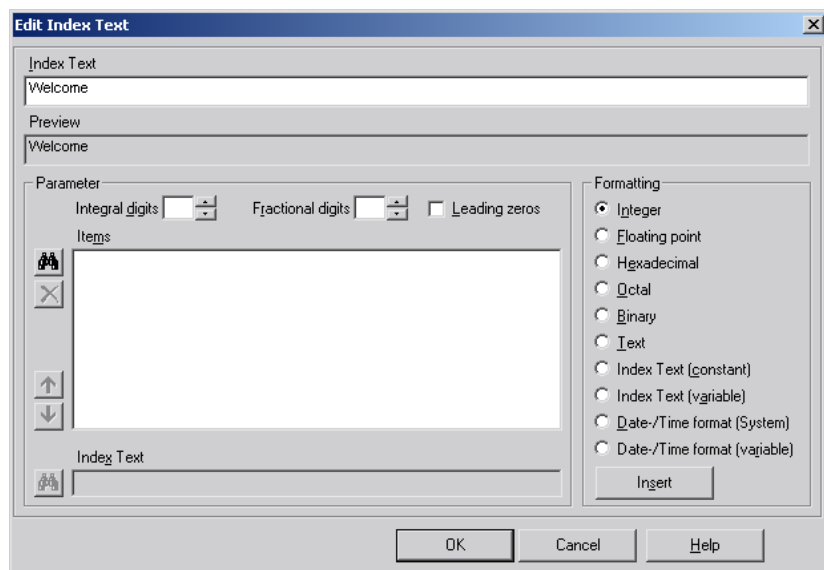
A further feature is the dynamic integration of additional information with texts:

- Process variable values
- Other texts
- Date/time formats.

At development time format character strings that are enveloped by "@" characters are integrated with the text. At runtime these areas are replaced by the appropriate values.




The text "lblWelcome" is to be shown in the text editor. Via the  button in the "English" column the dialog for text editing is to be opened.



This dialog contains a format character string generator whose functions allow to fit in the above mentioned functions.



Through the  button on the left of the "Variables" list the variable selection dialog is to be opened. Here a variable is to be selected.

In the formatting list the "Decimal" entry is to be selected. Via the "Insert" button the format character string is to be inserted in the text.



After the settings have been confirmed through the OK button the text field shows the following text:

"Welcome @1d@"

The formatting character string enveloped by "@" characters specifies that the first process value in the process variable list is to be shown in this place as a decimal value..

The test run shows that, indeed, the appropriate value of the variable is fitted into the text.

Beside the 'Label' control element other control elements, too, directly or indirectly support the display of localizable texts. In general, a 'LocalizedText' property marks the display of a localizable text.

Other control elements have no such property but obtain texts or formatting orders from the localization. The 'AlarmLine' control element for example contains the 'DateTimeFormat' property that allows specification of a text from the localization. The specification "@LongDate" as a property value means that the equally named text from the text group "Components.Time.DateFormats" contains a formatting character string for every project language that at runtime is used to format date/time values. This, too, is a function of the localization.

2.2.6 Alarm system

The alarm system serves for the central gathering and recording of error states or specific alarms. The appropriate definitions as to which variable triggers an alarm, how the alarm is to be presented and which additional information belongs to the alarm, are projected in the alarm editor. The variables that trigger alarms are registered for monitoring with the variable kernel by VisiWinNET. With a variable value change the appropriate alarm is sent to the application where it is displayed in special control elements.

VisiWinNET contains two control elements that allow direct display of current alarms:

AlarmList	All current alarms are displayed as a list. The control element also supports alarm acknowledgements.
AlarmLine	The space-saving variety that allows to watch current alarms at the top or bottom of all forms.

In addition to sending the alarm data to the surface VisiWinNET also allows storing alarms in files. This makes it possible to review the production process of a production line. The 'HistoricalAlarmList' control element serves to display these recorded data as a list.

A alarm is controlled by the change of a bit from the variable kernel. It can be selected whether the bit is set to "0" or "1" for the alarm to be triggered. By enhancement it is, however, also possible to monitor analog value for example for limit value exceedance. Alarms can be acknowledged by the user. Acknowledging feedback to the control can also be achieved.

The appearance of alarms in the 'AlarmLine' and 'AlarmList' control elements is parameterized through alarm classes. Alarm classes contain different parameters (colors, status texts and symbols) that describe the display of an alarm. Every alarm references an alarm class. This way every alarm is provided with the display parameters of an alarm class.

Alarm groups divide alarms. Similar as in the localization it makes sense to group the individual functional elements of a production line. This increases the clarity, and makes the project modular. Alarm groups, however, also contain parameters that allow group-wise deactivation or acknowledgement of alarms.

The following example shows:

- general projecting, triggering and display of alarms that are existent as bits in the variable kernel
- acknowledgement of alarms
- different display filters
- monitoring of analog values.



First of all, the sources of the alarms are to be defined. As alarms are variable-linked everything starts in the data access. However, since no real control data are available the triggering variables must be simulated by internal variables.



The "Variables" node in the project explorer is to be expanded. The table editor is to be opened through the "Channels→Internal" node. Here the following variables are to be defined:

Name	Data type
MessageSource	VT_I2
MessageState	VT_I2
GroupFunction	VT_I2



The variables defined here are sufficient as a functional data connection:

- MessageSource contains the bits that trigger the alarms.
- MessageStates contains the alarm states returned by the kernel functionality.
- GroupFunction serves for the control of alarm functionalities through the superior group.

Now follows setting the definitions in the alarm system:



The "Alarms" node in the project explorer is to be expanded, and the context menu of the inferior node "Alarm groups" to be accessed by a click with the r.h. mouse button. In the alarm group the following parameters are to be set:

Parameter	Value
Name	Grp1
Acknowledgement variable	GroupFunction
Acknowledgement variable	0

A second alarm group is to be created under the "Alarm groups" node.



Attention: Alarm groups are nestable. Therefore, always change to the "Alarm groups" node first.



The parameters of the second alarm groups are to be set to the following values:

Parameter	Value
Name	Grp2
Acknowledgement variable	GroupFunction
Bit number acknowledgement variable	1

In the table editor two new alarms are to be created in the "Alarms" table. After setting the cursor on a data set in the table editor the parameters of the alarm definitions are displayed on the VisiWinNET properties page.

Here the following settings are to be made:


For the 1st alarm

Parameter	Value
Name	Msg1
Event variable	MessageSource
Bit number	0
Message text	MessageSource Bit 0
Priority	1
Status variable	MessageState
Bit number status variable	0

For the 2nd alarm

Parameter	Value
Name	Msg2
Event variable	MessageSource
Bit number	1
Alarm text	MessageSource Bit 1
Priority	2
Status variable	MessageState
Bit number status variable	4



For the selection of variables there is the  button in the property page r.h. beside the appropriate parameter field.



Two further alarms are to be created under "Grp2". The parameters of these alarms are to be adapted in the same way as shown above.

For the 1st alarm

Parameter	Value
Name	Msg3
Event variable	MessageSource
Bit number	2
Alarm text	MessageSource Bit 2
Priority	3
Status variable	MessageState
Bit number status variable	8

For the 2nd alarm

Parameter	Value
Name	Msg4
Event variable	MessageSource
Bit number	3
Alarm text	MessageSource Bit 3
Priority	4
Status variable	MessageState
Bit number status variable	12



The values shown above will demonstrate the full function of the alarm system below. The above projected parameter "Status variable" is optional. Determining an alarm text is quasi compulsory. Finally, indication of the event variable is a must.

All newly created alarms point to the "Alarm" value with the "Alarm class" parameter. Behind this is a definition that hides under the "Alarm classes" node of the project explorer.



The "Alarm classes" node in the project explorer is to be marked. The cursor in the table editor is to be placed on the "Alarm" alarm class. In the VisiWinNET property page the parameter "Acknowledgement mode" is to be set to "Acknowledgement necessary".



VisiWinNET supports different acknowledgement modes. An acknowledgement mode determines how often an alarm must be acknowledged by the user or by the control before it vanishes from the control elements of the application. By this setting an alarm can assume different states at runtime. As an example the system distinguishes between the status "arrived" (event bit was set) and "arrived acknowledged" (after the event bit was set the user has acknowledged but the bit was not yet reset). Every status can be marked by a symbol, a status text and colors. These setting, too, are made in the alarm classes.

The status of an alarm can also be deposited bit-coded in a variable. For this purpose the status variables have in the alarms been provided with bit numbers. Three bits each represent the status of an alarm.

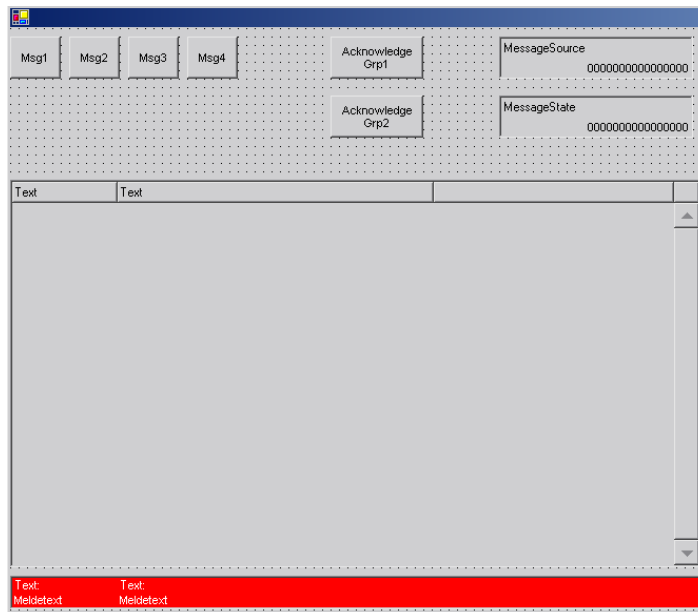
For all defined information to appear in the display the surface elements must now be projected. For better clarity this is done in an own form.



Via the "Forms" node in the project explorer a new form "MyAlarmForm" is to be created. With "CommandButton" control elements the appropriate page changes to the form are to be implemented.

On this form the following control elements (from top to bottom – see graphic) are to be placed:

- 2 x VarOut
- 4 x Switch
- 2 x Key
- 1 x AlarmList
- 1 x AlarmLine



The four 'Switch' control elements are to be connected with the "MessageSource" variable through the 'VWItem' property. The 'BitNumber' property in the VWItem object is to be set to the values 0, 1, 2 and 3 for the different control elements. These are the bits that trigger the alarms. Via the Text property the function of the button can be indicated.

The two 'Key' control elements are to be linked to the "GroupFunction" variable through the 'VWItem' property. The 'BitNumber' property in the 'VWItem' object is to be set to the value 1 for the second control element. The two connected bits are the acknowledgement bits of the alarm groups. Here, too, the functions of the buttons can be indicated through the 'Text' property.

The two 'VarOut' control elements are to be linked with the two internal variables "MessageSource" and "MessageState" through the VWItem property. In addition, the 'Format' property is to be set to 'binary'. Via the property Label→Text→Text the linked variable name is to be entered as information text.

The properties of the 'AlarmList' and 'AlarmLine' control elements need not be adapted for the first test run.

The application is to be started. The four lower bits of the "MessageSource" variable are to be set via the appropriate 'Switch' control elements.



The following changes become visible:

- All four alarms are displayed in the alarm list.
- In the alarm line the alarm "Msg4" is displayed. This is the current system alarm with the highest priority.
- The variable "MessageState" shows the status "100" (binary) for every of the alarms. If required this status could also be returned back to the control.



The four lower bits of the "MessageSource" variable are to be reset to "0" by a new click on the 'Switch' control elements.



- The alarm line shows no more alarms.
- In the alarm list the alarms remain. However, the status and with it the color changes. By the selected acknowledgement mode an acknowledgement is required to finally reset the alarms.
- The status bits of alarms and alarm groups now display the status "010".



The alarms are to be acknowledged one after the other by a double click on the appropriate entries in the alarm list.



- The alarms vanish from the alarm list.
- The status bits, too, in "MessageState" are reset.



The alarms are to be re-generated through the 'Switch' control elements.




The two 'Key' control elements are to be activated.

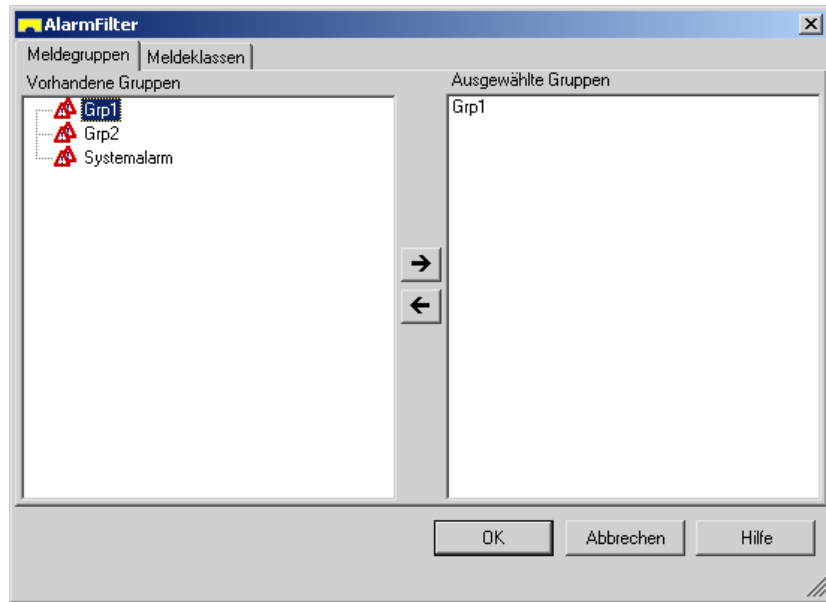
Through the acknowledgement bit of a group the contained alarms are commonly acknowledged.

Here the project can be stopped.

VisiWinNET contains the facility to filter the display of alarms in the control elements. By this it is for example possible to project an alarm list only with alarms regarding product line disruptions, and a further list with production information. In the Client-Server system this makes it also possible to only display the alarms that are important for that very workstation yet retain a central data recording.



Back in the design display the 'AlarmList' control element is to be marked. Via the  button of the 'AlarmFilter' property the dialog to set the filter is to be opened.



The filter allows the selection of alarm groups and alarm classes. Only the alarms in the groups selected here are displayed in the control element.



Through a double click on the entry "Grp1" the equally-named group is to be selected. After confirmation of the selection through the OK button the project is to be re-started. Again all four alarms are to be generated through "MessageSource".



Only the alarms in "Grp1" are displayed in the alarm list. The status bits show, however, that also the alarms in "Grp2" were generated. This means that the filter really only works for the display in the control element.

After hopefully cause and effect of alarms have now been sufficiently explained here now follows the description of analog value monitoring. A typical example is a temperature sensor whose value is divided into different areas (normal, critical, error).



The following example cannot be carried out with "Compact" or "Embedded" projects.



In the project explorer the node "Project configuration→ Server components" is to be marked. In the VisiWinNET property page the entry "Supervision/Process data supervision" is to be activated through the check box in the list of the server components.



In the project explorer a new node with the name "Process data supervision" becomes visible.

The procedure shows that VisiWinNET consists of individual components that can be individually activated or deactivated in a project. A deactivated component is not loaded at runtime.



Two new internal variables with the "VT_I2" data type are to be created in the variables editor. The names of the two variables are to be "AnalogValue" and "SupervisionOut".

In the editor of the alarm system a new alarm group with the name "AnalogValueAlarm" is to be created. In this group three alarms are to be defined with the following parameter values:

Parameter	Alarm 1	Alarm 2	Alarm 3
Name	Msg1	Msg2	Msg3
Text	AnalogValue Warning 1	AnalogValue Warning 2	AnalogValue Alarm
Event variable	SupervisionOut	SupervisionOut	SupervisionOut
Bit number event variable	3	4	5

Through the context menu of the "Process data supervision→Supervision groups" node a new supervision group is to be created. In the VisiWinNET property page the following settings are to be made:

Parameter	Value
Name	Analog
Type	Area supervision

In the table editor of the process data supervision a new supervision is to be added to the "Supervisions" table through the "F8" key.

The parameters of this supervision are to be filled with the following values:

Parameter	Value
Name	AnalogValueSupervision
Supervision variable	AnalogValue
Output variable	SupervisionOut
Bit number output variable	0
Limit HHH	100
Limit HH	90
Limit H	70

The form "MyFirstAlarm" is to be opened. Here a VarIn control element is to be added and to be connected with the "AnalogValue" variable through the 'VWItem' property.

Finally the group filter of the alarm list that was set in the previous steps is to be removed (take all groups out again).

The project is to be started. The value of the "AnalogValue" variable is to be set to the limit values "70/90/100" projected in the supervision through the appropriate 'VarIn' control element.



The appropriate alarms from the alarm system are triggered. Conceptionally, therefore, monitored analog values are not alarms themselves. The process data supervision only checks process values for limit value exceedance, and sets bits appropriately in other process values. These then serve to trigger bit-controlled alarms.



Attention: trap! Should the alarms not be displayed in the alarm list then that is due to the group filter set in the previous step.

In any case, however, the alarm should be visible in the alarm line.

2.2.7 Archive system

The VisiWinNET archive system allows the recording of process values. As a display component the 'TrendChart' control element is available. The necessary definitions on which process values are to be recorded and how, are made in the archive editor.



In the project explorer the "Trends→Archives" node is to be marked. Via the context menu a new archive definition is to be created. Through the VisiWinNET properties page the following parameters of the new definition are to be adapted:

Parameter	Value
Name	Archiv1
Type ("Format" index card)	Follow-on archive
Scan time ("Recording" index card)	10




An archive is a container for values to be recorded. That container

- ... pools multiple recorded process values in files
- ... determines the scan performance
- ... administrates the created files.

The determination of the process values to be recorded is described in the following step.




In the table editor a new trend definition is to be created through the "F8" key. Through the  button in the column "Trend variable" the variable selection dialog is to be opened. If an OPC server has been used with the previous steps the "Ch1.DistillationDevice.Unit1.WaterTank.Temperature" variable is to be selected.

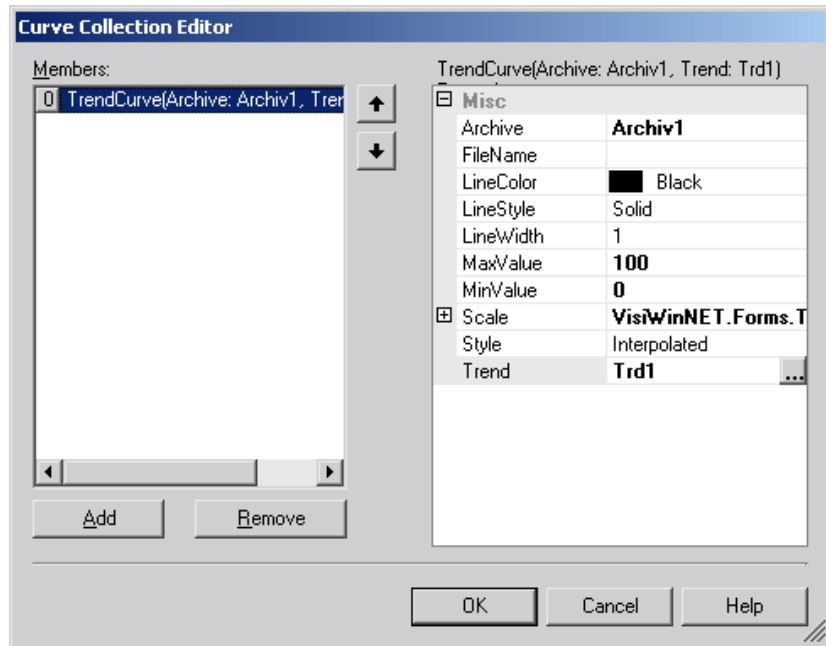
If a VisiWin driver was used with the previous steps the "Ch1.DummyItem<.Saege1>" variable can be used.



With this the determination of the process value, too, is finished. What is still missing is the display of the recorded values in the application.




A new form with the name "MyFirstTrend" is to be created, and to be made accessible through appropriately parameterized 'CommandButton' control elements from the other forms. On this form a control element of the 'TrendChart' type is to be placed.. Via the  button of the 'Curves' property the dialog for parameterization of the curves to be displayed is to be opened.



The list on the l.h. side contains the curves to be drawn. A curve is already inserted in the dialog. It has, however, no link with a trend definition yet.



The entry on the l.h. side is to be marked. Through the  button of the 'Archive' property on the r.h. side the dialog for the selection of archive and trend definitions is to be selected and "Trd1" is to be marked in the trend list. The selection is to be confirmed through the "OK" button. Subsequently, the 'MaxValue' property is to be set to 150.



The dialog selection is adopted into the 'Archive' and 'Trend' properties. 'MaxValue' parameterizes the upper visible value limit for this curve.



The dialog is to be closed through "OK". The 'TimeScale' property is to be expanded. In the inferior 'Resolution' property the value "00:02:00" is to be entered. The project is to be started.



The 'Resolution' property determines the visible time band in the control element. Through this the x-axis can be parameterized.

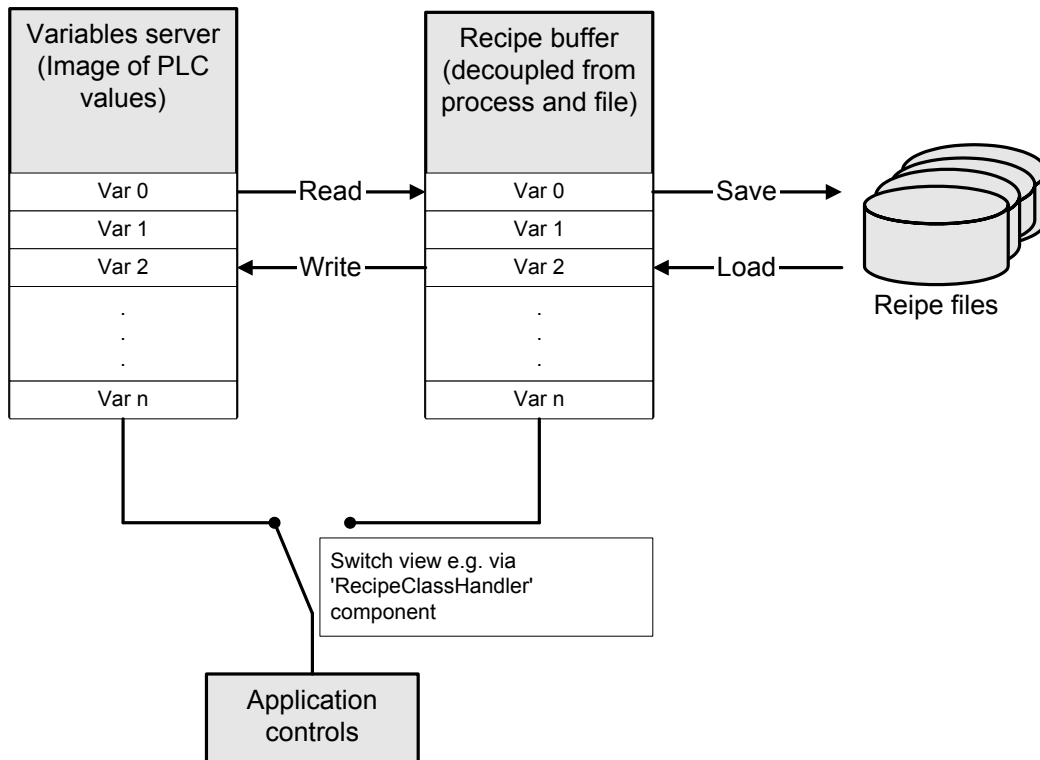
The current application shows the process value indicated in the trend definition as a curve.

2.2.8 Recipes

Visualization recipes describe sets of process values. Usually these determine together the settings of a machine to produce a product from a product selection. This makes switching between the individual products to be manufactured possible by writing recipes to the PLC.

At development time it is determined through recipe definitions which variables together describe the function, i.e. which variables are saved and transferred together.

At runtime the process value sets are filed in so-called recipe files. Between the process image and the recipe files there is the recipe cache. Through this cache recipe values in the application can be changed without affecting PLC data or recipe value data in the process. The recipe cache serves the parameterization and optimization of recipes. From the application for example recipe data are loaded from a file, adapted to the production target in the recipe cache, and subsequently transferred to the PLC. The opposite way allows to check value settings before a new recipe file is created.



The four transfer functions "Load", "Save", "Get" and "Set" control the individual transfer steps.

Typical input control elements such as 'VarIn' support the display of the process values in the variable kernel as well as in the recipe cache. If these values are to be made available for editing in the application the application does not have to be enhanced by new control elements. Rather, the display change that is centrally coupled with a recipe definition (for all variables used in the definition) is to be implemented.

Further functions such as the FDA-compliant gathering of recipe value changes or the saving of production comments are easily parameterizable.

The typical tasks for the developer in the recipe system are:

- Determination of the recipe definitions Here it is determined which variables describe the recipe. To preclude all misapprehensions: this does not refer to the variable values. These are only set at runtime.
- Design adaptation Basically, the existing control elements that are linked with the variables in the process can be used.
- Editing of the recipe values
- Design adaptation Typical tasks such as loading or saving recipe values from files in most cases require a collection of the existing recipe files.
- Administration of the recipe files

The following example illustrates the above mentioned steps based on a minimum recipe consisting of two variables:



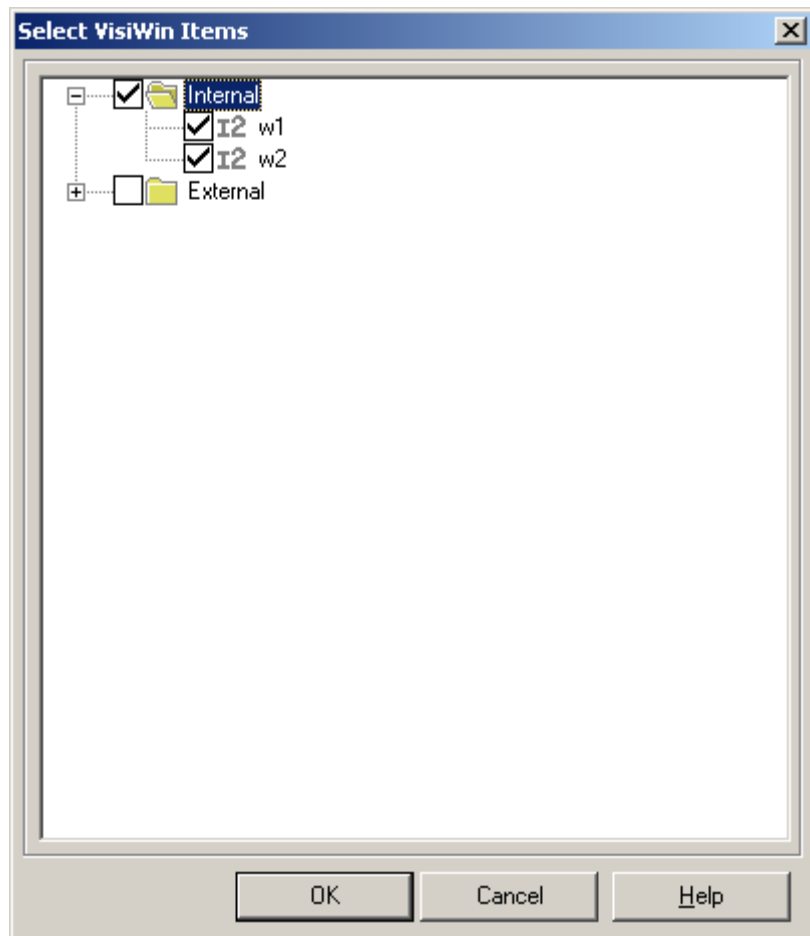
In the Project Explorer the "Variables" node is to be expanded. The table editor is to be opened through the "Channels→Internal" node. Here the following variables are to be defined:

Name	Data type
w1	VT_12
w2	VT_12

Mark the "Recipes→Recipe classes" node on the Project Explorer.

Through the context menu of the node a new recipe definition is to be created. The name of the definition is to be set to "Rec" through the VisiWinNET property page .

In the table editor of the new recipe definition the variable selection dialog is to be opened through the "New" context menu entry.



Here the variables belonging to the recipe definition are determined.



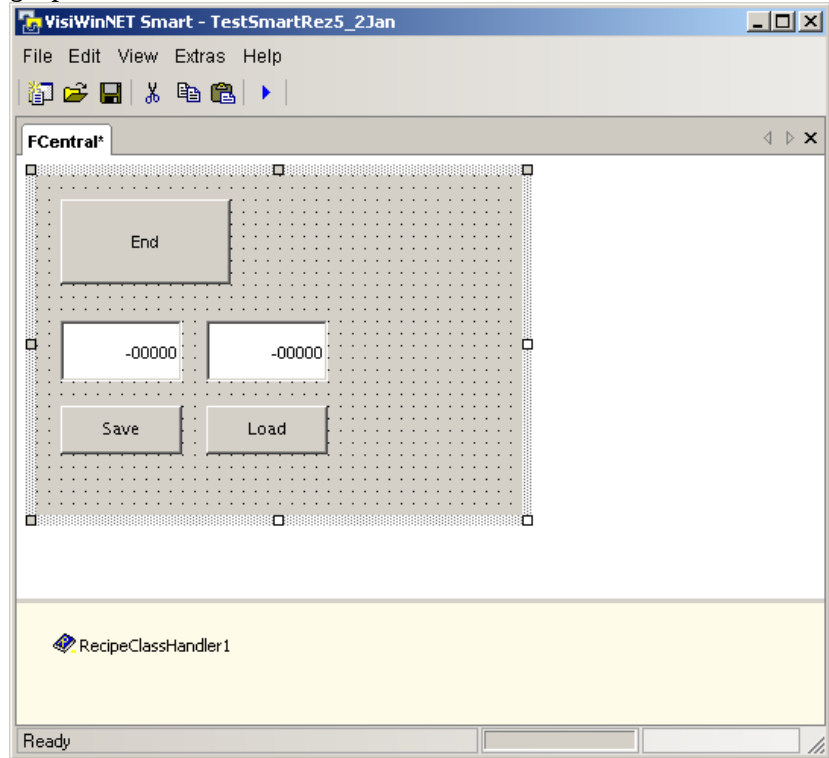
The two internal variables are to be marked. The dialog is to be left through the OK button.



The marked variables are now listed in the table editor as elements of the recipe.



On a form two 'VarIn', two 'CommandButton' and one 'RecipeClassHandler' component(s) are to be positioned (see graphic).



After the RecipeClassHandler component has been positioned on the form it moves automatically into the so-called "Component Compartment" underneath the design display of the form.




Here the following property values are to be changed:

- For VarIn1
- Property Value
- VWItem** w1
- For VarIn2
- Property Value
- VWItem** w2

For CommandButton1

Property Value

Text Save

Events The click event is to be linked with the access to the 'GetRecipe' function of the 'RecipeClassHandler' component. Through the  button in the 'Events' property the 'Event Management' dialog opens. Here the click event is to be selected, and the "Add" button activated. In the function selection dialog that now opens the above mentioned function is to be selected. After closing the function selection dialog make sure that the function appears in the functions list, and that its 'Provider' parameter refers to the 'RecipeClassHandler' component of the form.

For CommandButton2

Property Value

Text Load

Events The click event is to be linked with the access to the 'ShowLoadRecipeDialog' function of the 'RecipeClassHandler' component (see above).

For RecipeClassHandler1

Property Value

RecipeClass Rec (the name of the recipe definition from the project database)

Events Here the following events are used:

- The 'GetDoneSucceeded' event is to be linked with the access to the 'ShowSaveRecipeDialog' function of the 'RecipeClassHandler' component.
- The 'LoadDoneSucceeded' event is to be linked with the access to the 'SetRecipe' function of the 'RecipeClassHandler' component.



The such parameterized events cause the following actions when saved:

- The click event triggers the 'GetRecipe' function. Thereby the values of the variables determined as recipe elements are loaded into the recipe cache.
- The RecipeClassHandler reports the end of reading in the 'GetDoneSucceeded' event. This is in turn linked with the 'ShowSaveRecipeDialog' function. At runtime the saving of the recipe file is controlled through this. In the process a dialog for name and description input is shown prior to the saving process.

The opposite way is triggered through the Load button. First a recipe file is loaded into the recipe cache via a selection dialog. Subsequently it is from there transferred to the process.



This example does practically not use the recipe cache. With many applications it may be sufficient to create and optimize recipes directly in the process.

The use of the recipe cache does, however, make sense where production adjustments are to be made through the application that must not during the adjusting process change the values in the process. Only once the adjustments are finished the data can then together be transferred to the process.

In the "VWNRRecipeDemo" project the use of the recipe cache is shown as an example.

2.2.9 User administration

The VisiWinNET user administration is for allocating individual-related rights within the visualization application. Input control elements can be blocked, access to information about output control elements can be denied.

VisiWinNET supports two administration systems for users:

Level-oriented user administration Users and control elements can be allocated to one of 999 levels. If at runtime the level of a control element is higher than the level of the currently logged on user operation of the control element is denied.

Rights-oriented user administration A right is allocated to the control elements in the application that the user must also own in order to operate the control element (e.g. "edit recipe"). Subsequently the users are allocated the rights that they can exercise.

This makes particular sense in a Client-Server system if multiple employee hierarchies (groups of employees that are separated into rights level independently of each other) are active in the network. Release decisions are made through the comparison of user rights.

Selection of the system is made under the "User Administration→Configuration" node. After highlighting this node the "System" parameter is to be set on the VisiWinNET properties page.

User administration is effected through three definitions in the project database:

User group Pools users into a group, and contains the characteristic parameters for all users in this group. Through the user group the references to the rights required for the operation of the application are administrated.

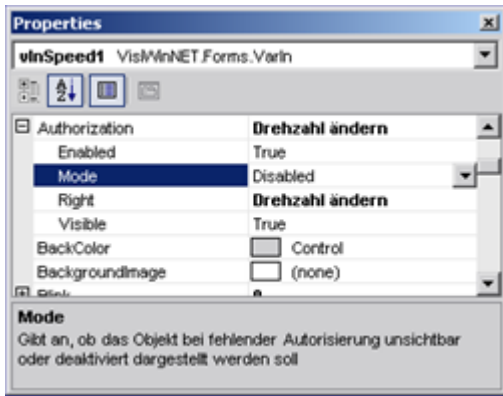
User Contains the access information (logon name and password) and the activation status of a user.

Right Rights define the information that is allocated to the control elements of the application. Through the "Authorization" property a right is allocated to a control element. A logged on user must have a reference to this right in his user group definition to be allowed to operate the control element.

Rights are only used in the rights-oriented user administration. In the level-oriented user administration the "Authorization" property of the control elements expects a figure.

All three definition types can at development time be projected through the User Administration Editor.

All VisiWinNET control elements contain an "Authorization" property. In this property the name of the right that is to make the control element operable at runtime is determined. Also it is specified how the control elements is to perform in case of a missing right. If the property is expanded in the properties window selection is possible in the 'Mode' adjustment beneath between 'disabled' and 'invisible'.



At runtime VisiWinNET provides special dialogs for administration of the user information. A typical example is the logon dialog that demands the user name and a password as input. Through this logon the control elements are automatically activated or deactivated depending on the allocated right. The access to the user administration dialog is supported by the "UserManagementDialog" component.

The following example shows the steps for implementing a rights-oriented user administration. In the example it is shown how linking the individual elements is to be done, and how user data can be adapted at runtime. There is an administrator as well as two users with different rights. Through the 'Authorization' property the appropriate rights are linked in the control elements.

Also, the example contains accessing the logon dialog and the administration dialog for editing user data.



The "User Administration→Configuration node is to be highlighted. On the VisiWinNET properties page the "System" setting is to be set to "rights-oriented".



In the Project Explorer the "Rights" node is displayed.



The "Rights" node is to be marked. In the table editor three rights with the names "Right1", "Right2" and "Administration" are to be registered.

Subsequently, mode to the "User Groups" node. Here three groups are to be created. The parameters are to be set as follows:

1st group

Here the name is to be adapted, and the link to "Right1" to be established.

2nd group

Like with the first group the name is to be adapted, and the link, as shown above, to be established.

3rd group

Administrators are permitted a lot more, in this case everything. Consequently, all rights have to be linked.



Allocation to rights is through the user group. This may at first glance seem a bit complicated, does, however, help in the real world: every user group can incorporate any number of users. The rights in the group can be commonly set for all contained users, hence commonly adapted.

In each of the three user groups a user is to be created in the table editor. The following parameters are to be defined in the process:

Parameter Value

Logon name <a name>

Password <a password that must in the initial setting contain at least four characters>

Status activated



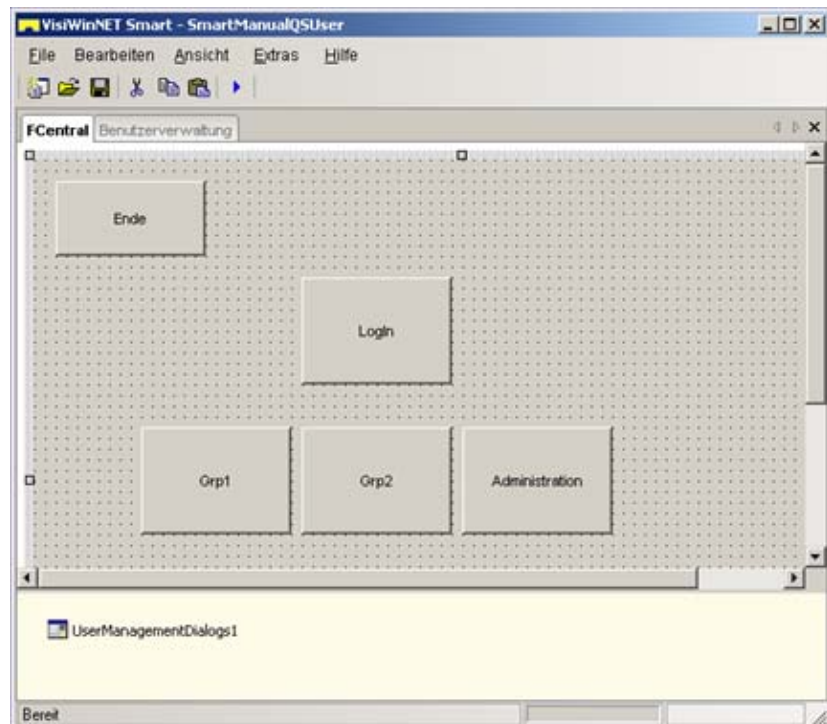
Logon name and password are later used as logon information. The status allows different settings that decide whether the user can log on, and whether he must change the password assigned by the projector with the first logon.



Do memorize the logon information you have entered for the further progress in the example.




In the design display a ,UserManagementDialogs' component and four control elements of the ,CommandButton' type are to be placed in a form.



Here the following properties are to be adapted:

For button 1

Text LogOn

Events The click event is to be linked with the access to the 'ShowLogOnDialog' function of the 'UserManagementDialogs' component. Through the  button a dialog opens. Here the click event is to be selected, and the "Add" button to be activated. In the function selection dialog that is now displayed the above mentioned function is to be selected. After closing the function selection dialog make sure that the function appears in the functions list, and its 'Provider' parameter refers to the 'UserManagementDialogs' component of the form.

For button 2

Text Grp1

Authorization Right1

For button 3

Text Grp2

Authorization Right2

For button 4

Text Administration

Authorization Administration

Events The click event is to be linked with the access to the 'ShowUserDialog' function of the 'UserManagementDialogs' component (see above).



The Text property here serves the display of the button function.

The 'Authorization' property determines when the button is to be activated: if a user is logged on, and if his user group is linked with the right specified here the button is activated.

The setting of the 'Events' property serves accessing the "Log on user" and "Administrate user" dialogs.



The project is to be started.



Following the start the "LogOn" button is released. Here no setting was made in the 'Authorization' property. A click on the button opens the dialog to log on a user.

The other three buttons are released by log on of the appropriate users. The two buttons "Grp1" and "Grp2" are without function. They are released by the users from the groups "Grp1" and "Administration" and "Grp2" and "Administration". If buttons parameterized that way are used to move to further forms entire application parts can be released depending on the current user.

The "administration" button is only available to users from the equally named user group. A click on the button opens the user administration dialog. Here new users can be created, and existing users administrated (deactivated, deleted or their settings changed).



User data that have been changed at runtime are not saved in the project database. Instead the system generates a file with the ending "rtn" in the project directory.

2.2.10 Logging



This chapter is in the construction phase.

3 Projects under Windows CE

3.1 Project transmission from Visual Studio to the CE appliance

Visualization applications that are to run on a Windows CE appliance are developed on a standard PC. For the correct function all required files must then be copied to the CE appliance. Visual Studio contains functions that configure and establish the connection with the target appliance, and also arrange the transfer of the executable project file as well as all references to the target system. With Visual Studio all projects can also be tested (debugged) on the CE appliance.

Due to the enhanced project structure of VisiWinNET visualization applications these functions are, however, not sufficient. Therefore, VisiWinNET enhances the Visual Studio functions for the transmission of projects to a Windows CE appliance.

The guidelines below describe how these enhanced functions are to be operated by the developer.

Preparation of the connection

Copy RemoteAccessManager to CE appliance and start

The connection with the target system is established through the "VisiWinNET RemoteAccessManager" auxiliary program. This program is part of the VisiWinNET installation, and must where appropriate be copied to the Windows CE appliance.

The appropriate file can be found in the VisiWin installation directory under:

```
<x>:\VisiWin\VisiWinNET\Compact\RemoteAccessManager\
```

The file can for example be copied via Windows network through authorization by the development computer. Under Windows CE open the "Start" menu, and select "Run". Then enter the computer name in the format of \\MyPC. "MyPC" here stands for the network name of the development computer. Subsequently, either a display of the PC authorizations opens or entering of user name and password is requested.

For permanent availability of the program it is advisable to save this file to the Compact Flash of the CE appliance.

Once the tool is successfully saved it can be started by a double click on the CE appliance. The VisiWinNET RemoteAccessManager then shows its readiness for the connection with the following message:

Waiting for connection at "IP Address".

"IP Address" here stands for the local IP Address of the Windows CE appliance.

The appliance is now ready to establish the connection with the development environment.

Configure the appliance in Visual Studio

The following steps are to be executed on the development computer in Visual Studio:

- Open the Visual Studio options dialog through the Extras→Options entry.
- Select the appliances configuration node "Appliance tools→Appliances" on the left hand side of the tree display.
- Duplicate the "Windows CE device" entry through the "Save as" button. Here, enter a name with a high recognition value (such as "VWNDevice").
- Open the properties dialog of the new appliance through the "Properties" button.
- Specify an existing output folder (e.g. "Root Folder\Storage"), the specification depends on the drive name on the CE appliance.
- Open the "TCP Transport Configuration". Select the "Use special IP Address" option, and specify the IP Address of the CE appliance that is to receive the project (is displayed in the RemoteAccessManager on the CE appliance).
- Close the "TCP Transport Configuration" and "Appliances Settings" dialogs.

Optionally, the new appliance can be determined as standard appliance in the options dialog.

The settings made as above are used by Visual Studio. The enhanced transmitting function of VisiWinNET is configured in the next step as described below.

Establishing the connection

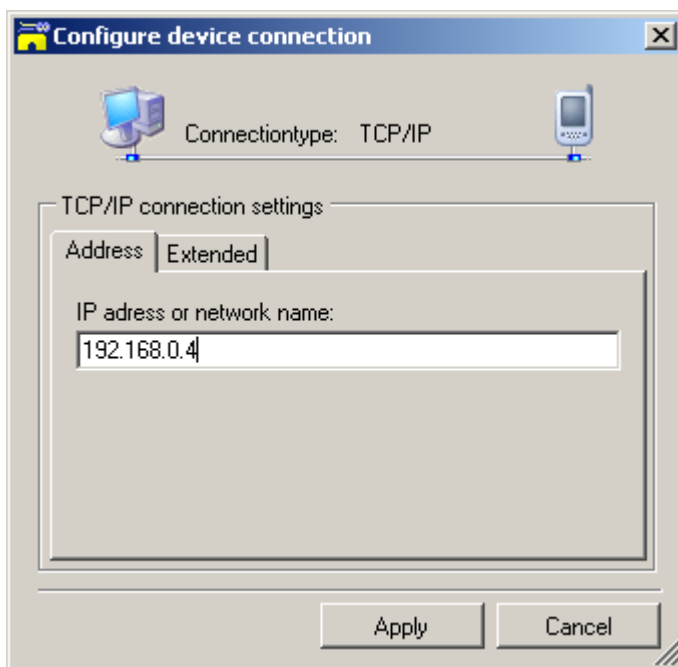
To establish the connection make sure that the appliance that was configured in the previous step is used by Visual Studio. For this, open the properties dialog of the project

through the "Project→<Project name> Properties" menu entry. On the "Appliances"

index card there is a "Target Appliance" combination list with the configured appliances. Here the appropriate appliance is to be selected.

With the start/transfer of the project the additional connection dialog provided by VisiWinNET is displayed.

Here the connection parameters are listed again:



The IP Address shown here originates from the appliance configuration administrated by Visual Studio. A change influences the impending connection only. The additional specification of a port on the "Enhanced" index card is administrated independently from Visual Studio.

The following connection is established in two steps:

- VisiWinNET functions: transfer of the specific project files (e.g. Project Databank, Language Files, Smart Forms, ...) and the connection files for Visual Studio
- Visual Studio: transfer of the references and the executable EXE. Where appropriate start of the application.

Additional VisiWin functions for appliance connections

VisiWinNET inserts a tool bar in Visual Studio.



The buttons of the tool bar contain the typical connection functions:



Start the project. In the process the entire project transfer and the project start are taken over by VisiWinNET alone. In this constellation debugging with Visual Studio does not work.



Stop the project. Different from the equally named Visual Studio function the project is terminated properly. All loaded project parts (OPC Server, application, drivers) are ended in the appropriate sequence through '*ExitApplication'.



Transfer project. Transfers the project and all required files with the specified settings without starting the project.



Establish connection. Established the connection with the specified settings without a data transfer taking place.



End the connection. Closes the connection.



Configure the connection. Opens the dialog to set the connection properties.

4 More references

This chapter introduces further support and sources of information contained in the product's scope of supply. Fundamentally, this information is split into the following categories:

Help for the systems and functions of VisiWinNET Contains the online help for the VisiWinNET-specific definitions, their parameters, and the projection.

Help for the class library of VisiWinNET Contains the reference of the design components (control elements and the appropriate classes and components).

4.1 Help for the systems and functions of VisiWinNET

Every VisiWinNET project contains a project database in which at development time special definitions are projected with the help of editors. Typical definitions are for example the variables, messages or localizable texts used in the application. Every definition contains a set of parameters to determine the runtime performance. The runtime components create by these parameters appropriate objects in the RAM that are available to the visualization.

The help for the systems and functions contains fundamentally:

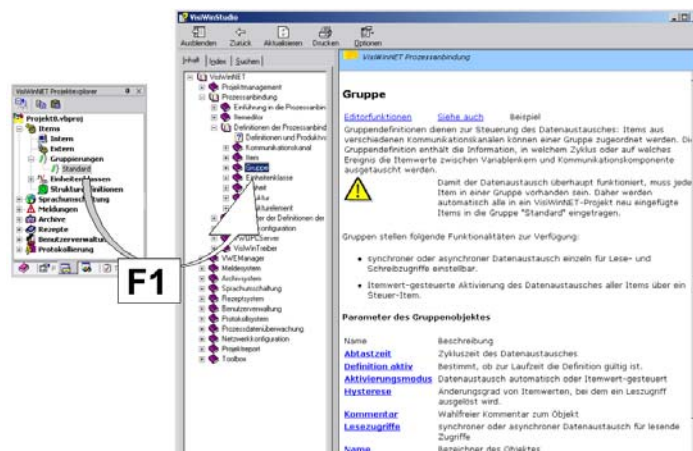
- The operation of the editors for creating definitions in the project database
- The description of the definitions and their parameters.

This help is available as an online help.

4.1.1 Access

Access to this help is through the Project Explorer or in the editors:

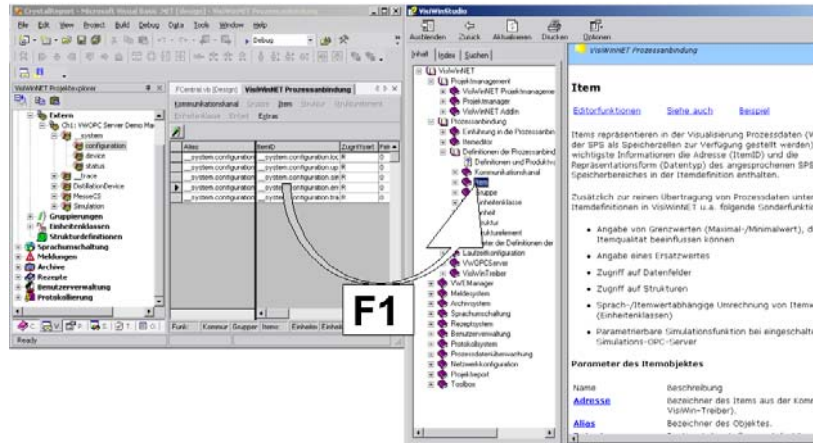
Project Explorer In the Project Explorer a definition can be selected. Via the "F1" key the help appropriate to the selected definition is accessed.



The help contains the description of the definition, and a reference list to its parameters. By a click on a reference the help displays the appropriate subject referring to the selected parameter.

Index text Editor

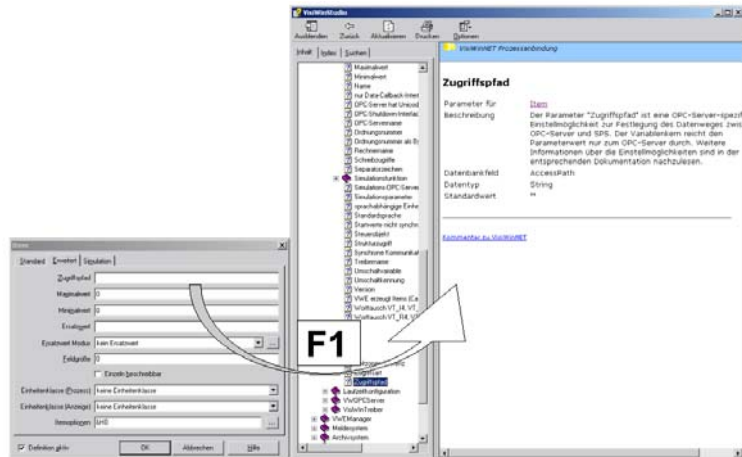
In the Index Editor, too, the help as to the displayed definition can be accessed via "F1".



Dialogs

The dialogs of the definitions allow direct access to the help relating to a parameter. When the dialog for editing a definition is opened the online help can be accessed in two ways:

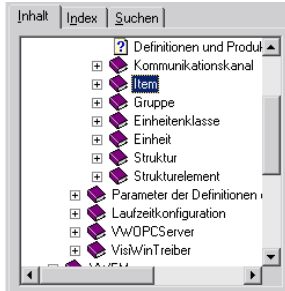
- Via the Help button the appropriate help relating to the definition is displayed.
- Subsequent to setting the input mark in a field displaying the parameter values the help relating to the appropriate parameter is displayed via "F1".



4.1.2 Research

Once the developer has accessed the help further research can be carried out. Through the references of the currently displayed page further subjects can be accessed that are thematically linked with the current page. The help system also contains further functions to help with the research. For this, the l.h. part of the help window contains three index cards:

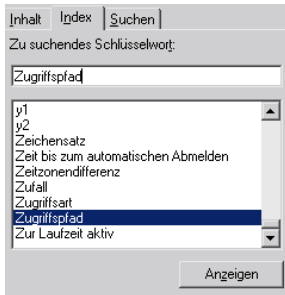
Contents



The content tree shows a hierarchic display of the table of contents. It contains the summary of all VisiWinNET systems. Every system fundamentally divides into the following chapters:

- Introduction** A description of the system with flow chart and introduction of the individual components
- Projection** The operation reference of the editor
- Definitions** The description of the definitions that can be used in the system.
- Parameters** The description of the parameters that are available to the definitions.

Index



The index contains an alphabetically sorted collection of all available keywords. Listed here are for example all parameter and definition names. The index is used if the term to be searched for is known but the direct access from the development environment and the position of a help page in the content tree are not.

Search



The "Search" function allows intensive search for a term. After specification of the search term the help conducts a full text search through all help pages. Help pages containing the term are listed as hits. Beside the specification of combined terms (via the "AND" keyword) wildcards (Wildcards "*") can be used, too.

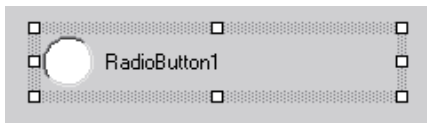
4.2 Help for the VisiWinNET class library

The help for the class library contains the description of all "Types" available in VisualStudio.NET. "Types" are here fundamentally control elements, components and classes that that be interpreted and compiled by the VB.NET and C# programming languages.

4.2.1 Access

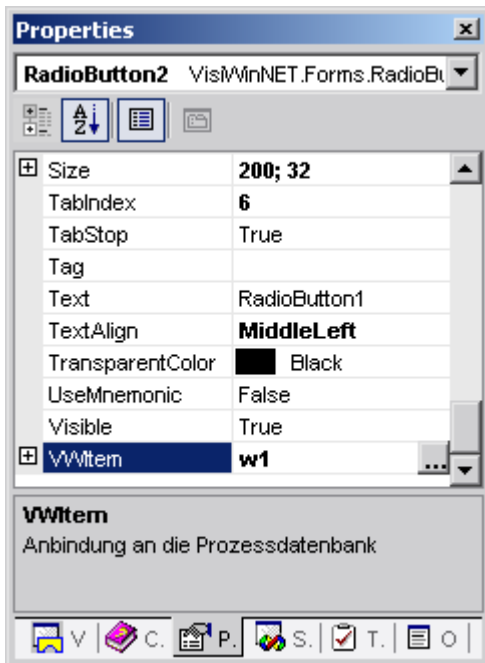
Access to the help is possible in several ways:

Select a control element in a form then "F1" key



The help page appropriate to the control element is opened.

Select a property in the properties window then "F1" key



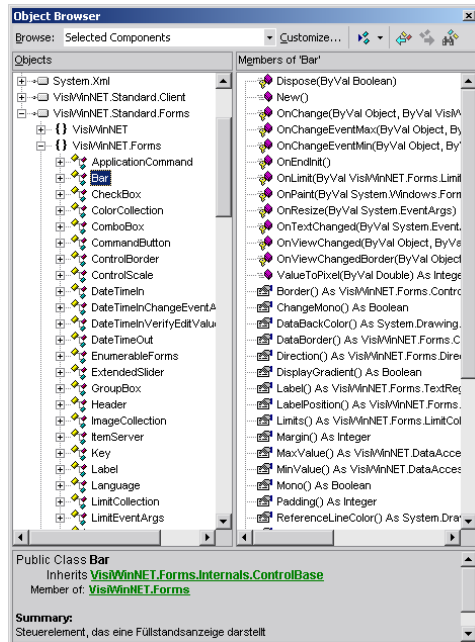
The help page appropriate to the property is opened. Even before the "F1" key is pressed a short description of the property is displayed in the lower window area.

Set input mark on a word in the source code then "F1" key

```
Private Sub CommandButton1_Click(  
    RadioButton1.StopEdit(True)  
End Sub
```

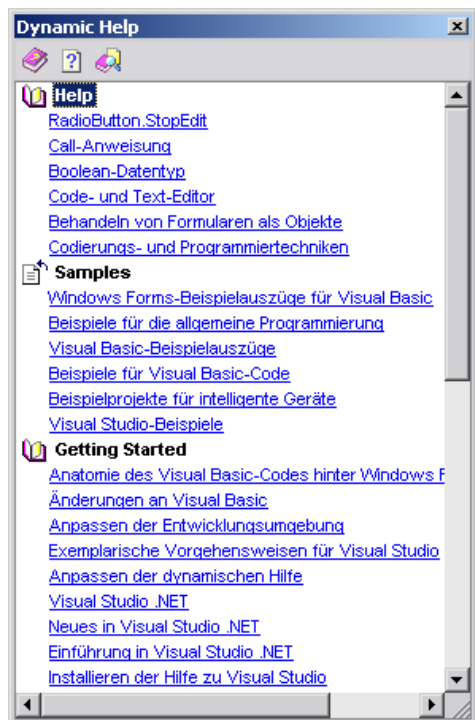
The help page appropriate to the type or keyword is opened.

Select an entry in the object catalog then "F1" key



The object catalog is displayed through the "Display→Object Catalog" menu item. It contains for all types integrated in the project a collection of the available "Members" (properties, events, methods or constants). In the lower window area a short description of the selected type/the selected member is displayed. The "F1" key opens the appropriate help page.

Dynamic help

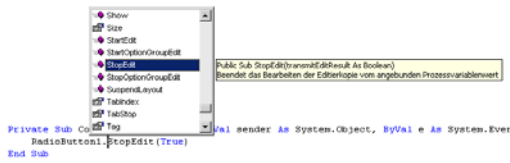


A further way to access the help is via the dynamic help window. This window displays a context-related (dependent on the elements currently visible/selected in the development surface) list of help subjects. The content of the window changes simply by selection in the development environment. If for example a window is opened the dynamic help window displays help subjects related to this window.

A click on a list item in the dynamic help window starts the appropriate help page.

The dynamic help window is displayed via the "Help→Dynamic Help" menu item.

Intellisense

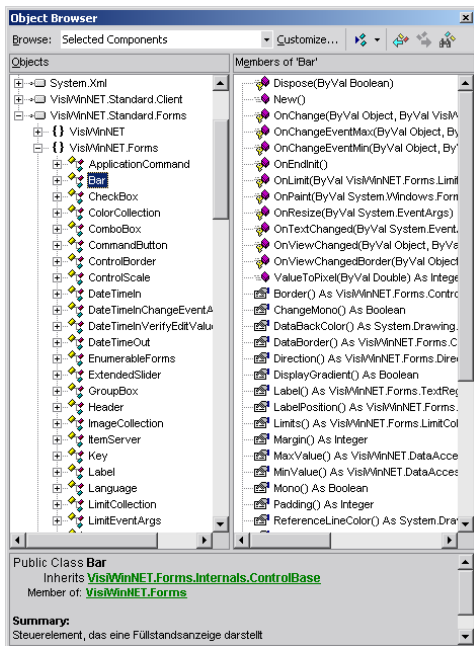


When writing source code the development environment supports the projector by two windows :

- By setting the "." operator a selection list opens containing the appropriate "Members" (properties, events, methods or constants) of the object specified ahead of the "." operator.
- Via the navigation keys (↑↓) alternation between the list entries is possible. In the process a tooltip is displayed showing the short description of the "Member" selected in the list.

4.2.2 Contents of the Class Library help

The structure of the Class Library help follows the .NET programming scheme. It is best demonstrated by the object catalog:



VisiWinNET provides two assemblies to the developer whose types and members can be used in the visualization application:

Client

Depending on the product version, e.g. "VisiWinNET.Standard.Client" or "VisiWinNET.Compact.Client"

Provides the programmatic interface with the system servers (variable kernel, alarm server, ...).

Forms

Depending on the product version, e.g. "VisiWinNET.Standard.Client" or "VisiWinNET.Compact.Client"

Contains the functional elements and classes for the application (amongst others the control elements). Internally the types contained here use the functions of the client.

The types of an assembly are split into so-called "Namespaces". Namespaces divide the types of an assembly. As an example the VisiWinNET systems (alarm/archive/data access) are divided in the client into appropriate namespaces (alarm/trend/data access).

For the developer the most important namespace for an application is "VisiWinNET.Forms". Here the most frequently used control elements and classes are listed. A first rough summary is provided via the help page of the namespace (select namespace from the object catalog then press "F1").

Klasse	Beschreibung
ApplicationCommand	Klasse, die die Funktion eines 'CommandButton'-Steuerelementes beschreibt
Bar	Steuerelement, das eine Füllstandsanzeige darstellt
CheckBox	Steuerelement, das ein Kontrollkästchen mit erweiterten Funktionen darstellt
ColorCollection	Auflistung von Farben, die den einzelnen Zuständen zugeordnet sind
ComboBox	Steuerelement, das ein Kombinationsfeld mit Prozessanbindung darstellt
CommandButton	Steuerelement, das eine Funktionsschaltfläche darstellt
ControlBorder	Klasse, die die äußere Umrandung eines Steuerelementes festlegt
ControlScale	Klasse, die eine Skala in einem Steuerelement beschreibt
DateTimeIn	Steuerelement, das die Eingabe von (sprachabhängig) formatierten Eingaben von Datums-/Zeitwerten ermöglicht
DateTimeInChangeEventArgs	Stellt Daten für das 'Change'-Ereignis bereit.
DateTimeInVerifyEditValueEventArgs	Stellt Daten für das 'VerifyEditValue'-Ereignis bereit.
DateTimeOut	Steuerelement zur Ausgabe von Datum und Uhrzeit
EnumerableForms	Auflistung der Formulare, die zur Zeit geladen wurden
ExtendedSlider	Steuerelement, das einen Schieberegler mit Hintergrund darstellt
GroupBox	Steuerelement, das einen Gruppenfeld darstellt
Header	Kapselt die Funktion eines Tabellenkopfes
ImageCollection	Auflistung von Grafiken, die den einzelnen Zuständen zugeordnet sind
ItemServer	Komponente zum Zugriff auf einen Prozesswert ohne bestimmte Interpretation
Key	Steuerelement, das eine Schaltfläche darstellt.
Label	Steuerelement, das einen Text (fest, variabelwertbezogen und/oder sprachumgeschaltet) darstellt
Language	Komponente zur Steuerung der Sprache in der Applikation

This help page lists all types of the selected namespace by keywords. The references in the l.h. column lead to the appropriate detailed descriptions.

Klasse Bar

VisiWinNET Klassenbibliothek

Bar

Vererbungshierarchie

[System.Object](#)
[System.MarshalByRefObject](#)
[System.ComponentModel.Component](#)
[System.Windows.Forms.Control](#)
[VisiWinNET.Forms.Internals.ControlBase](#)
VisiWinNET.Forms.Bar

Beschreibung

Das Steuerelement 'Bar' stellt eine Füllstandsanzeige dar.

Es hat Anbindungen an folgende Systeme:

- Prozessanbindung: Über die 'VWItem'-Eigenschaft wird eine Prozessvariable festgelegt, deren Wert als Füllstand dargestellt wird.
- Sprachumschaltung: Zusätzlich zur Füllstandsanzeige kann ein Textfeld im Steuerelement eingeblendet werden, das einen Text aus der Sprachumschaltung dargestellt
- Benutzerverwaltung: Die Eigenschaft 'Authorization' ermöglicht die 'Sperrung' des Steuerelementes in Abhängigkeit von den Freigaben des aktuell im System angemeldeten Benutzers
- Rezeptsystem: Zusätzliche Funktionen der 'VWItem'-Eigenschaft ermöglichen die Umschaltung zwischen Variablenkern und Rezeptserver. Dies ermöglicht sowohl die Ansicht des realen Prozesswertes als auch des entsprechenden Wert in der Rezeptdatei.
- Applikationsweites Blinken: Über die 'Blink'-Eigenschaft kann das Steuerelement an das applikationsweit synchronisierte Blinken angebunden werden. Das Steuerelement kann im Blinktakt unsichtbar geschaltet werden oder es kann eine selbstdefinierte Aktion im 'DoBlink'-Ereignis ausgeführt werden (z.B. das Umschalten einer Farbe im Steuerelement).

Weitere Funktionen des 'Bar'-Steuerelementes erlauben:

- die Unterteilung des Füllstandes in verschiedenfarbige Bereiche (Eigenschaft 'Limits')
- die wahlweise Darstellung in eine (von 'MinValue' bis 'MaxValue') oder in zwei Richtungen (vom 'ReferenceValue' in Richtung 'MinValue' bzw. 'MaxValue')
- das Einblenden einer Skala
- das Einblenden des numerischen Variablenwertes

Konstruktor

Mitglied	Beschreibung
New	Erzeugt eine neue Instanz der 'Bar'-Klasse.

Eigenschaften

Mitglied	Beschreibung
AccessibilityObject (geerbt von Control)	Ruft das dem Steuerelement zugewiesene AccessibleObject ab.
AccessibleDefaultActionDescription (geerbt von Control)	Ruft die Beschreibung der Standardaktion des Steuerelements für die Verwendung durch Eingabehilfen-Clientprogramme ab oder legt diese fest.
AccessibleDescription (geerbt von Control)	Ruft die Beschreibung des von Eingabehilfen-Clientprogrammen verwendeten Steuerelements ab oder legt diese fest.
AccessibleName (geerbt von Control)	Ruft den Namen des von Eingabehilfen-Clientprogrammen verwendeten Steuerelements ab oder legt diesen fest.
AccessibleRole (geerbt von Control)	Ruft die Eingabehilfenrolle des Steuerelements ab oder legt diese fest.
AllowDrop (geerbt von Control)	Ruft einen Wert ab, der angibt, ob das Steuerelement Daten annehmen kann, die vom Benutzer darauf gezogen wurden, oder legt diesen fest.

Help pages for "Types" (control element, component, class, enumeration) are divided into the following items:

Heredity hierarchy

Shows from which classes the type inherits, and which further classes inherit the properties of the type.

Description

Describes which use cases the type supports, and which special functions it is given where appropriate.

Member list

A list is provided of all members of the control element divided into "Constructor", "Properties", "Events", "Methods" and "Constants". Every member is listed with a short description. Inherited members are provided with an appropriate comment.

- Remarks** Remarks that where appropriate refer to peculiarities of the type.
- Example** An example of the utilization of the type is given if provided.
- See also** Finally, a reference list is offered allowing access to thematically related help pages. These are:
 - The namespace the type is in
 - Members that use the type as a parameter or return value
 - Related subjects whose utilization is associated with the type.



Help pages for "Members" (properties, events, methods or constants) are divided into the following items:

- Syntax** For the C# and VB.NET programming languages the syntax is shown (i.e. how the member is to be used in the source code). All used data types are offered as references. This is of particular importance when for example a property expects an object as data type. If the developer after having read the current help page wants further information as to the data type to be passed he can directly change to the help of the data type via the reference in the syntax description.
- Description** Provides a short description of the function.
- Remarks** Special remarks are provided where appropriate.
- Example** An example of the utilization is given if provided.
- See also** Finally, a reference list is offered allowing access to thematically related help pages. These are:
 - The namespace the type is in
 - The type implementing the "Member"
 - Related subjects whose utilization is associated with the "Member".

4.3 Manual

4.3.1 PDF files

The manuals are provided as a collection of PDF files on the CD in the "VisiWinNET\Common\Documentation" directory. The PDF files are divided into subjects. They can be read and printed out through the Acrobat Reader. The Acrobat Reader is provided as a download version on the CD in the "Acrobat Reader" directory (please observe the Adobe licence agreements when installing).



All information contained in the manual is also provided in the online help. It is, therefore, not urgently necessary to print out the entire manual with a volume of nearly 2000 pages.