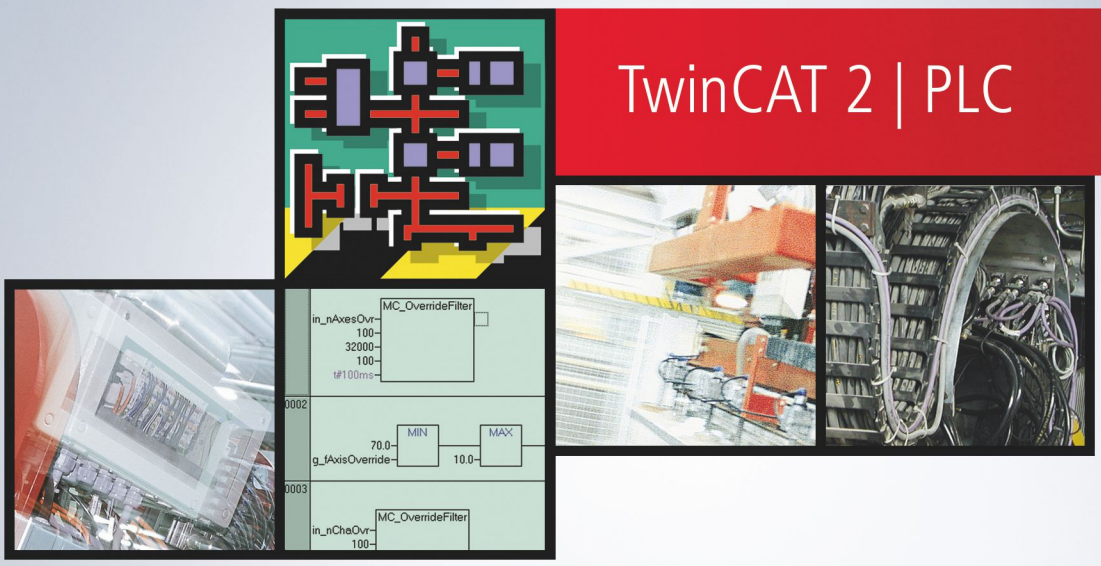
BECKHOFF New Automation Technology

Manual | EN

TX1200

PLC Control



TwinCAT 2 | PLC

2025-08-12 | Version: 1.3

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
The documentation and the following notes and explanations must be complied with when installing and commissioning the components.
The trained specialists must always use the current valid documentation.

The trained specialists must ensure that the application and use of the products described is in line with all safety requirements, including all relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been compiled with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without notice.
Claims to modify products that have already been supplied may not be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar®, and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.



EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.
The distribution and reproduction of this document, as well as the use and communication of its contents without express authorization, are prohibited.
Offenders will be held liable for the payment of damages. All rights reserved in the event that a patent, utility model, or design are registered.

Third-party trademarks

Trademarks of third parties may be used in this documentation. You can find the trademark notices here: https://www.beckhoff.com/trademarks.

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER

Hazard with high risk of death or serious injury.

⚠ WARNING

Hazard with medium risk of death or serious injury.

⚠ CAUTION

There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE

The environment, equipment, or data may be damaged.

Information on handling the product



This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

Table of contents

1 Foreword ... 3
1.1 Notes on the documentation ... 3
1.2 For your safety ... 3
1.3 Notes on information security ... 5

2 A Brief Introduction to TwinCAT PLC Control ... 9
2.1 Project Components ... 10
2.2 Languages ... 16
2.2.1 Programming Languages ... 16
2.2.2 Instruction List (IL) ... 16
2.2.3 Structured Text (ST) ... 18
2.2.4 Sequential Function Chart (SFC) ... 23
2.2.5 Function Block Diagram (FBD) ... 27
2.2.6 The Continuous Function Chart Editor (CFC) ... 27
2.2.7 Ladderdiagram (LD) ... 28
2.2.8 Debugging, Online Functions ... 29
2.2.9 IEC 61131-3 ... 30

3 Sample Program ... 32

4 Individual Components ... 41
4.1 The Main Window ... 41
4.2 Options ... 43
4.3 Managing Projects ... 66
4.4 Objects ... 87
4.5 Editing Functions ... 99
4.6 Online Functions ... 103
4.7 Windows ... 112
4.8 Help System ... 113
4.9 Log ... 113

5 Editors ... 115
5.1 Declaration Editor ... 116
5.2 Text Editors ... 125
5.3 Instruction List Editor ... 128
5.4 Structured Text Editor ... 129
5.5 Graphical Editors ... 130
5.6 Function Block Diagram Editor ... 132
5.7 Ladder Diagram Editor ... 136
5.8 Continuous Function Chart Editor (CFC) ... 141
5.9 Sequential Function Chart Editor ... 152

6 Resources ... 161
6.1 Global Variables ... 161
6.2 Alarm Configuration ... 164
6.2.1 Alarm system, Terms ... 165
6.2.2 Alarm classes ... 166
6.2.3 Alarm groups ... 169

6.2.4 Alarm saving ... 170
6.2.5 'Extras' Menu: Settings ... 171
6.3 PLC Configuration ... 172
6.4 Task Configuration ... 173
6.5 Sampling Trace ... 175
6.6 Watch and Recipe Manager ... 179
7 Reference programming ... 182
7.1 Data Types ... 182
7.1.1 Standard Data Types ... 182
7.1.2 User Data Types ... 186
7.2 Operators ... 190
7.2.1 Overview IEC Operators ... 190
7.2.2 Numeric Operators ... 194
7.2.3 Arithmetic Operators ... 197
7.2.4 Bitstring Operators ... 199
7.2.5 Bitshift Operators ... 201
7.2.6 Selection Operators ... 204
7.2.7 Comparison Operators ... 206
7.2.8 Selection of different Operators ... 208
7.2.9 Type Converting Operators ... 210
7.3 Operands ... 213
7.3.1 Constants ... 213
7.3.2 Variables ... 215
7.4 System Functions ... 218
7.4.1 CheckBounds ... 218
7.4.2 CheckDivByte : BYTE ... 219
7.4.3 CheckDivReal : REAL ... 219
7.4.4 CheckDivWord : WORD ... 220
7.4.5 CheckDivDWord : DWORD ... 220
7.4.6 CheckRangeSigned : DINT ... 221
7.4.7 CheckRangeUnsigned : UDINT ... 222
7.5 Compiler Errors ... 223
7.6 Programming conventions for creating the IEC61131-3 ... 249
8 Library Management ... 252
9 Engineering Interface (ENI) ... 254
10 Visualization ... 257
10.1 Visualization Editor ... 258
10.1.1 Create a new Visualization ... 259
10.1.2 Insert Visualization element ... 259
10.1.3 Positioning of Visualization Elements ... 263
10.1.4 Configure Visualization ... 266
10.2 Language switching in the Visualization ... 309
10.2.1 Static ... 311
10.2.2 Dynamic ... 312
10.2.3 Language dependend Online Help ... 316

10.3 Placeholder Concept ... 316
10.4 Online Mode ... 318
10.5 Libraries ... 319
10.6 TwinCAT PLC HMI Visualization ... 319
10.6.1 Installation, Start and Operating ... 320
10.7 Target Visualization ... 320
10.7.1 Feature overview of visualisation with TwinCAT ... 321
10.7.2 Requirements ... 324
10.7.3 Creating a Target Visualization ... 325
10.7.4 Starting the Target Visualization ... 328
10.7.5 Scan of mouse-clicks and dynamic texts ... 328
10.7.6 Restrictions ... 330
10.8 System Variables ... 330

11 Appendix ... 332
11.1 Using the Keyboard ... 332
11.2 Command Line/Command File Commands ... 334

2 A Brief Introduction to TwinCAT PLC Control

What is TwinCAT PLC Control?

TwinCAT PLC Control is a complete development environment for your PLC. TwinCAT PLC Control puts a simple approach to the powerful IEC language at the disposal of the PLC programmer. Use of the editors and debugging functions is based upon the proven development program environments of advanced programming languages.

TwinCAT PLC Control Overview

How is a project structured?

A project is put into a file named after the project. First a default Task Configuration is opened. The name of the task is 'Standard'. The first POU (Program Organization Unit) created in a new project will automatically be named MAIN. You can rename this POU in the task configuration. TwinCAT PLC Control differentiates between the different kinds of objects in a project: POUs, data types and ressources. The Object Organizer contains a list of all the objects in your project.

How do I set up my project?

First you should select the target system. Then configure the task. You can create the POUs needed to solve your problem. Now you can program the POUs you need in the desired languages. Once the programming is complete, you can compile the project and remove errors should there be any.

How can I test my project?

Once all errors have been removed, log in to the PLC and "load" your project in the PLC. Now TwinCAT PLC Control is in Online mode. Test your project for correct sequence. To do this, enter input variables manually and observe whether outputs are as expected. You can also observe the value sequence of the local variables in the POUs. In the Watch and Recipe Manager you can configure data records whose values you wish to examine.
In case of a programming error you can set breakpoints. If the process stops at such a breakpoint, you can examine the values of all project variables at this point in time. By working through sequentially (single step) you can check the logical correctness of your program.
An additional TwinCAT PLC Control debugging function: You can set program variables and inputs and outputs at certain values. You can use the flow control to check which program lines have been run. The Sampling Trace allows you to trace and display the actual course of variables over an extended period of time.
A Log records operations, user actions and internal processes during an online session in a chronological order.
The entire project can be documented or exported to a text file at any time.

More Features:

The entire project can be documented or exported to a text file at any time. Also it can be translated into another language.

ENI: The ENI interface ('Engineering Interface') allows to connect the Programming system to an external data base. There the data which are needed during creation of an automation project can be stored. The usage of an external data base guarantees the consistency of the data, which then can be shared by several users, projects and programs.

Summary

TwinCAT PLC Control is a complete development tool used to program your PLC which will save you a measurable amount of time setting up your applications.

2.1 Project Components

A project contains all objects of a control program. A project is saved in a file with the name of the project. A project includes the following objects:
function blocks, data types, resources and libraries.

POU (Program Organization Unit)

Functions, function blocks, and programs are POUs, which can be supplemented by actions.
Each POU consists of a declaration part and a body. The body is written in one of the IEC programming languages which include IL, ST, SFC, FBD, LD or CFC. TwinCAT PLC Control supports all IEC standard POUs. If you want to use these POUs in your project, you must include the library standard.lib in your project.
POUs can call up other POUs. However, recursions are not allowed.

Function

A function is a function block that returns exactly one data element (which can also be multi-element, such as fields or structures) when executed, and whose call in textual languages can occur as an operator in expressions.

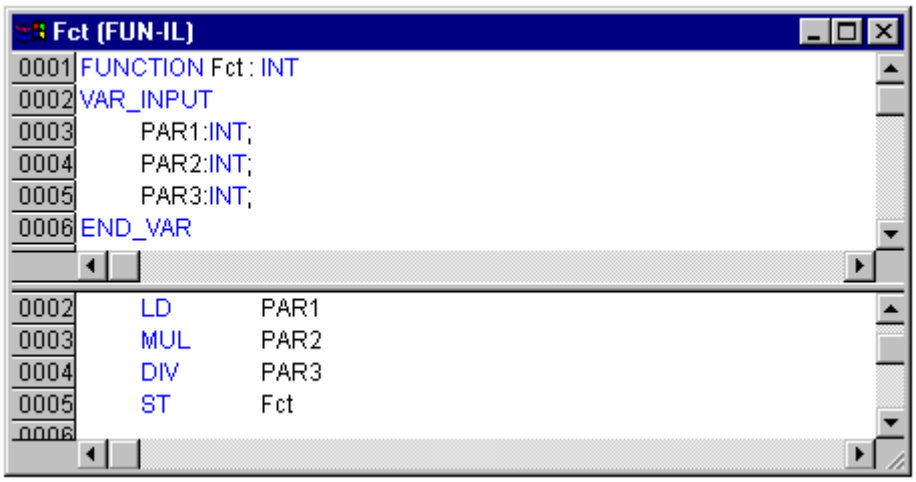
When declaring a function, it is important to note that the function must be given a type. I.e. after the function name a colon followed by a type must be entered.

A correct function declaration looks like this, for example:

```
FUNCTION Fct: INT
```

In addition, a result must be assigned to the function. The function name is used like an output variable.

Sample in IL for a function that takes three input variables and returns as result the product of the first two divided by the last one:



The call to a function can occur in ST as an operand in expressions.

Functions do not have internal states. I.e. calls of a function with the same arguments (input parameters) always return the same value (output).



If a local variable is declared as RETAIN in a function, this has no effect. The variable is not stored in the retain area.

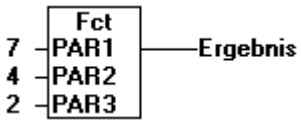
Samples of calling the function described above:
in IL:

```
LD 7
Fct 2,4
ST Ergebnis
```

in ST:

```
Ergebnis := Fct(7, 2, 4);
```

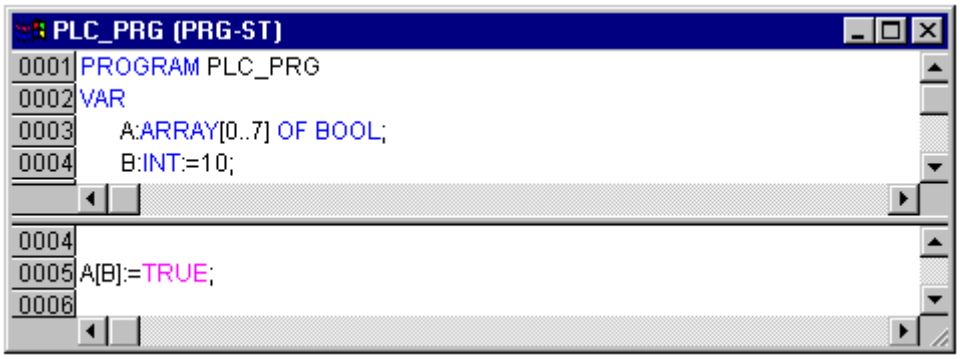
in FBD:



In SFC, a function call can only be made within a step or transition.

If you define a function in your project with the name CheckBounds [▶ 218], you can use it to check for range overflows in your project! The name of the function is defined and may have only this identifier.

The following typical program for testing the CheckBounds function goes beyond the boundaries of a defined array. The CheckBounds functions makes sure that the value TRUE is not assigned to the position A[10], but rather to the upper area boundary A[7] which is still valid. Therefore, the CheckBounds function can be used to correct extensions beyond array boundaries.



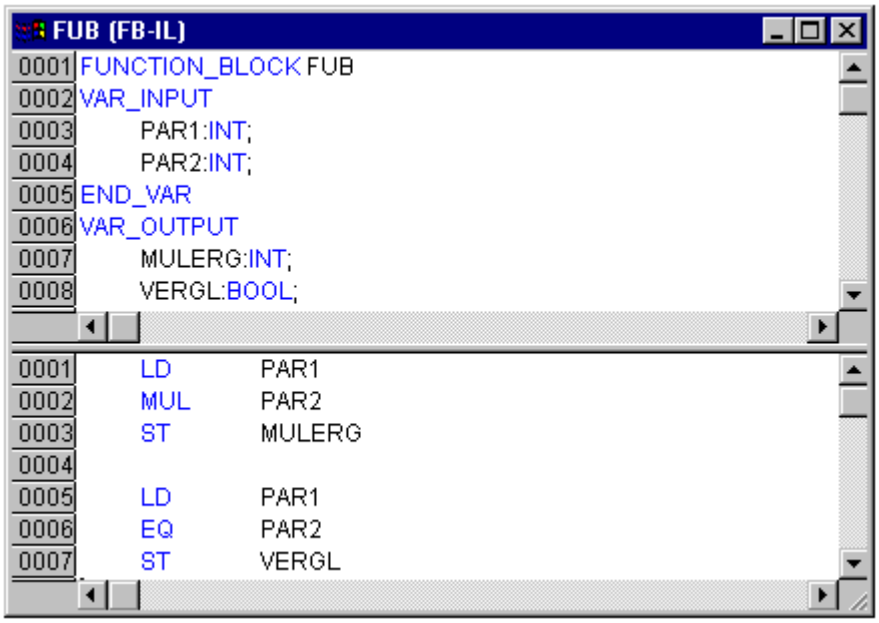
If you define functions in your project with the name **CheckDivByte**, **CheckDivWord**, **CheckDivDWord** and **CheckDivReal**, you can use them to check the value of the divisor if you use the operator DIV, for example to avoid a division by 0. The name of the function is defined and may have only this identifier.

If you define functions with the names **CheckRangeSigned** and **CheckRangeUnsigned**, then range exceeding of variables declared wit subrange types can be intercepted.
All these check function names are reserved for the described usage.

Function block

A function block is a POU which provides one or more values during the procedure. As opposed to a function, a function block provides no return value.

Example in IL of a function block with two input variables and two output variables. One output is the product of the two inputs, the other a comparison for equality:



Function block Instances

Reproductions or instances (copies) of a function block can be created. Each instance possesses its own identifier (the Instance name), and a data structure which contains its inputs, outputs, and internal variables. Instances are declared locally or globally as variables, whereas the name of the function block is indicated as the type of an identifier.

Example of an instance with the name INSTANCE of the FUB function block:

```
INSTANCE: FUB;
```

Function blocks are always called through the instances described above.

Only the input and output parameters can be accessed from outside of a function block instance, not its internal variables.

Example for accessing an input variable: The function block FB has an input variable in1 of the type INT.

```
PROGRAM prog
  VAR
    inst1:fb;
  END_VAR

  LD 17
  ST inst1.in1
  CAL inst1
END_PROGRAM
```

The declaration parts of function blocks and programs can contain instance declarations. Instance declarations are not permitted in functions.

Access to a function block instance is limited to the POU in which it was declared unless it was declared globally. The instance name of a function block instance can be used as the input for a function or a function block.



All values are retained from one execution of the function block to the next. Therefore, calls of a function block with the same arguments do not always return the same output values!



If the function block contains at least one retain variable, the entire instance is stored in the retain area.

Calling a function block

The input and output variables of a function block can be accessed from another POU by setting up an instance of the function block and specifying the desired variable using the following syntax:

```
<Instance name>.<Variable name>
```

If you would like to set the input parameters (thus the value of the inputvariables) when you open the function block, you can do this in the text languages IL and ST by assigning values to the parameters after the instance name of the function block in parentheses (this assignment takes place using ":=" just as with the initialization of variables at the declaration position).

Please note, that the input/output variables (VAR_IN_OUT) of a POU will be turn over as a pionter. Therefore no constants can be assigned to them at a call, and an external read or write access is impossible. Example for the call of a VAR_IN_OUT variable inout1 of the POU fubo in ST:

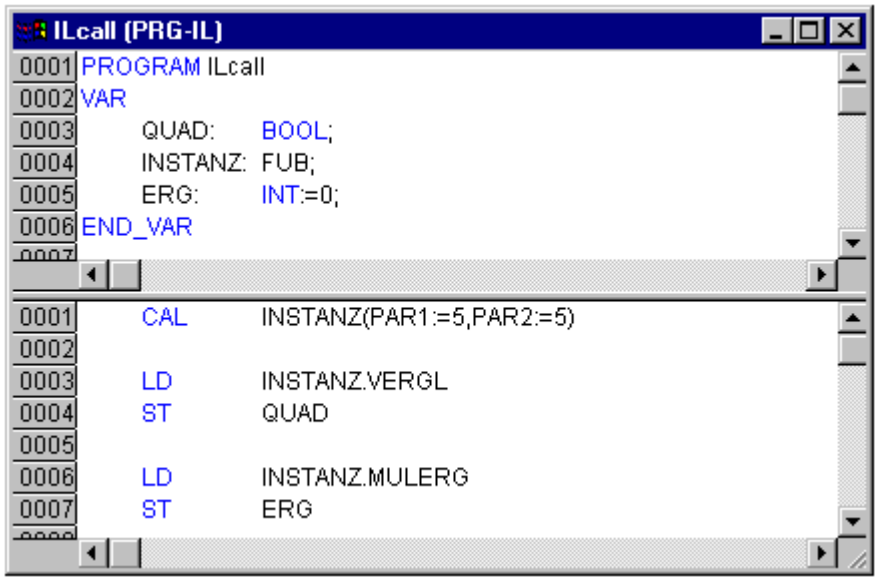
```
VAR
    inst:fubo;
    var1:int;
END_VAR

var1:=2;
inst(inout1:=var1);
```

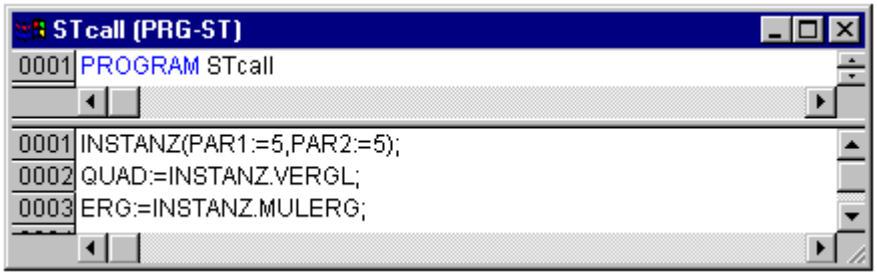
Not allowed: inst(inout1:=2); or. inst.inout1:=2;

Examples for calling function block FUB described above. The multiplication result is saved in the variable ERG, and the result of the comparison is saved in QUAD. An instance of FUB with the name INSTANCE is declared:

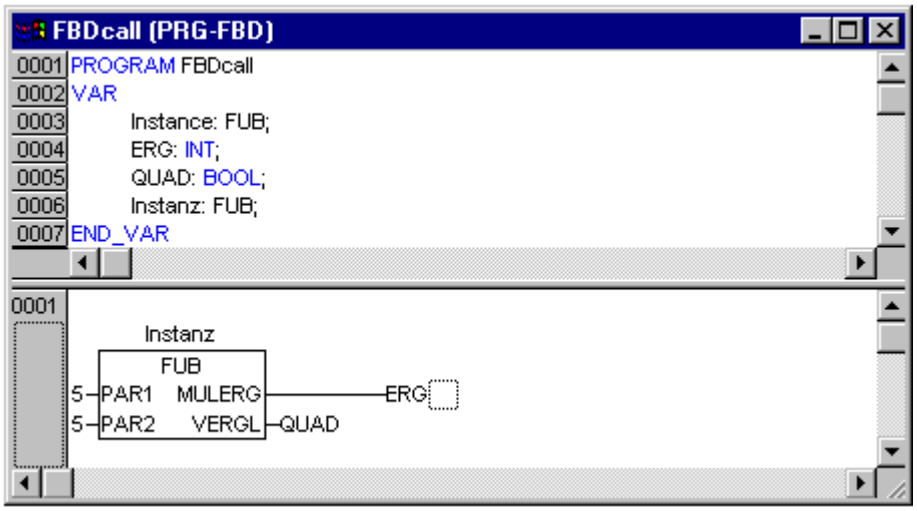
In IL the function block is called as follows:



In the example below the call is shown in ST. The declaration part is the same as with IL:



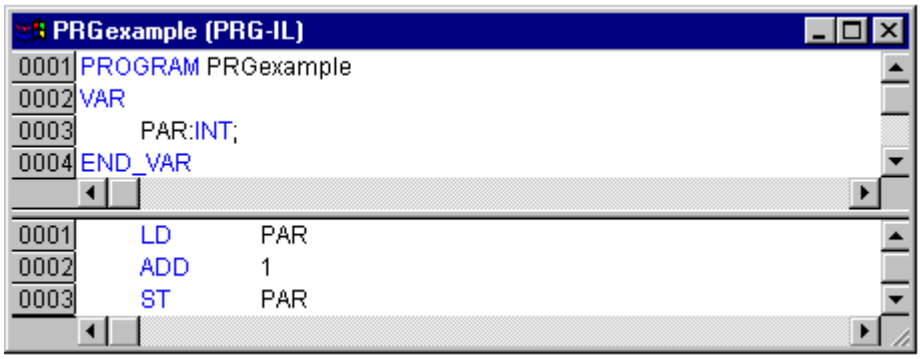
In FBD it would look as follows (declaration part the same as with IL):



In SFC function block calls can only take place in steps.

Program

A program is a POU which returns several values during operation. Programs are recognized globally throughout the project. All values are retained from the last time the program was run until the next.



Programs can be called. A program call in a function is not allowed. There are also no instances of programs. If a POU calls a program, and if thereby values of the program are changed, then these changes are retained the next time the program is called, even if the program has been called from within another POU. This is different from calling a function block. There only the values in the given instance of a function block are changed. These changes therefore play a role only when the same instance is called. A program declaration begins with the keyword PROGRAM and ends with END_PROGRAM.

Examples of calls of the program described above:

In IL:

```
CAL PRG Example
LD PRGexample.PAR
ST ERG
```

in ST:

```
PRGExample;
Erg := PRGexample.PAR;
```

In FBD:

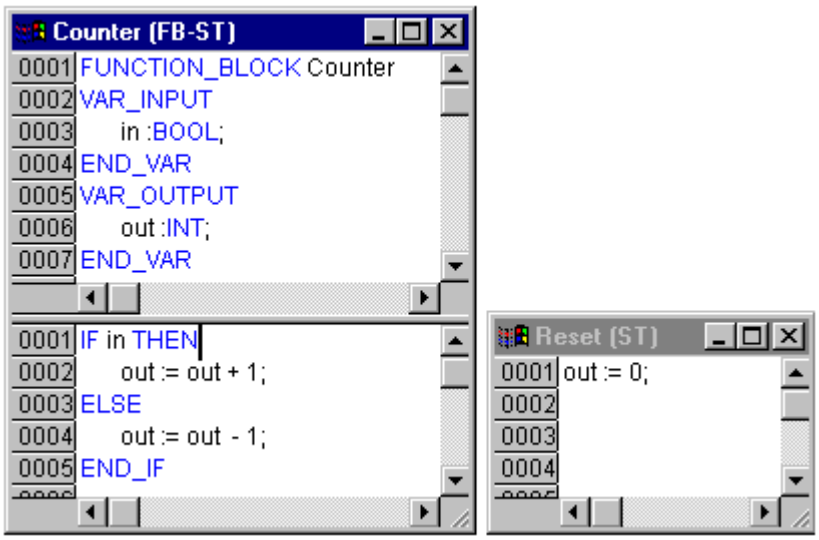


If the variable PAR from the program PRGexample is initialized by a main program with 0, and then one after the other programs are called with above named program calls, then the ERG result in the programs will have the values 1, 2, and 3. If one exchanges the sequence of the calls, then the values of the given result parameters also change in a corresponding fashion.

Action

As addition to function blocks and programs you can define actions. An action is an further implementation. It can be written in a different language than the 'normal' implementation. Each action gets a name.

An action works with the data of the accompanying function block or program. It uses the same input/output variables and local variables like the 'normal' implementation..



When function block Counter is called, the output variable gets increased or decreased depending on the input variable 'in'. When action Reset, affiliated to the function block is called, the output variable is set to 0. In both cases the same variable out is written.

An action is called with <program name>.<action name> respectively <instance name>.<action name>. If the action should be called within the 'mother' module, in text editors only the action name is used, in graphical editors the function block call without instance name.

Example

Declaration for all Examples:

```
PROGRAM PLC_PRG
VAR
  Inst : Counter;
END_VAR
```

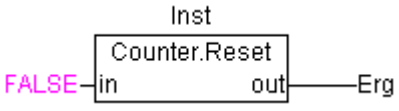
In AWL:

```
CAL Inst.Reset(In := FALSE)
LD Inst.out
ST ERG
```

In ST:

```
Inst.Reset(In := FALSE);
Erg := Inst.out;
```

In FBD:



See Chapter SFC [▶ 23] for more information about SFC Actions.

In IEC 61131-3 there are only actions for SFC described.

Ressourcen

You need the ressources for configuring and organizing your project and for tracing variable values:

- Global Variables which can be used throughout the project
- PLC Configuration for configuring your hardware
- Task Configuration for guiding your program through tasks
- Sampling Trace for graphic display of variable values
- Watch and Recepe Manager for displaying variable values and setting default variable values

See the chapter called "The Ressources [▶ 161]".

Libraries

You can include in your project a series of libraries whose POUs, data types, and global variables you can use just like user-defined variables. The library "standard.lib" is a standard part of the program and is always at your disposal.

See the chapter called "Library Manager [▶ 252]".

Data types

Along with the standard data types the user can define his own data types. Structures, enumeration types and references can be created.

See 'Standard [▶ 182]-' and 'User Defined Datatypes [▶ 182]' in the appendix.

2.2 Languages

2.2.1 Programming Languages

TwinCAT PLC Control supports all IEC 61131-3 languages. There are two textual languages and four grafical languages.

texual languages

- Instruction List (IL) [▶ 16]
- Structured Text (ST) [▶ 18]

grafical languages

- Function Block Diagram (FBD) [▶ 27]
- Ladder Diagram (LD) [▶ 28]
- Continous Function Chart (CFC) [▶ 27]
- Sequential Function Chart (SFC) [▶ 23]

2.2.2 Instruction List (IL)

An instruction list (IL) consists of a series of instructions. Each instruction begins in a new line and contains an operator and, depending on the type of operation, one or more operands separated by commas. In front of an instruction there can be an identification mark (label) followed by a colon (:).

A comment must be the last element in a line. Empty lines can be inserted between instructions.

Example:

```
LD      17
ST      lint      (* comment *)
GE      5
JMPC    next
LD      idword
EQ      istruct.sdword
STN     test
next:
```

Modifiers and operators in IL

In the IL language the following operators and modifiers can be used.

Modifiers:

- C with JMP, CAL, RET: The instruction is only then executed if the result of the preceding expression is TRUE.
- N with JMPC, CALC, RETC: The instruction is only then executed if the result of the preceding expression is FALSE.
- N otherwise: Negation of the operand (not of the accumulator)

Below you find a table of all operators in IL with their possible modifiers and the relevant meaning:

Operator	Modifiers	Meaning
LD	N	Make current result equal to the operand
ST	N	Save current result at the position of the Operand
S		Put the Boolean operand exactly at TRUE if the current result is TRUE
R		Put the Boolean operand exactly at FALSE if the current result is TRUE
AND	N, (	Bitwise AND
OR	N, (	Bitwise OR
XOR	(	Bitwise exklusive OR
ADD	(	Addition
SUB	(	Subtraction
MUL	(	Multiplication
DIV	(	Division
GT	(	>
GE	(	>=
EQ	(	=
NE	(	<>
LE	(	<=
LT	(	<
JMP	CN	Jump to label
CAL	CN	call function block
RET	CN	Return from call of a function block
)		Evaluate deferred operation

You find a list of all IEC operators in the appendix.

Example of an IL program while using some modifiers:

```
LD      TRUE        (*load TRUE in the accumulator*)
ANDN    BOOL1       (*execute AND with the negated value of the BOOL1 variable*)
JMPC    label       (*if the result was TRUE, then jump to the label "label"*)
LDN     BOOL2       (*save the negated value of *)
ST      ERG         (*BOOL2 in ERG*)
label:

LD      BOOL2       (*save the value of *)
ST      ERG         (*BOOL2 in ERG*)
```

It is also possible in IL to put parentheses after an operation. The value of the parenthesis is then considered as an operand.

For example:

```
LD 2
MUL 2
ADD 3
ST Erg
```

Here is the value of Erg 7. However, if one puts parentheses:

```
LD 2
MUL( 2
ADD 3
)
ST Erg
```

Here the resulting value for Erg is 10, then the operation MUL is only then evaluated if you come to ")"; as operand for MUL 5 is then calculated.

2.2.3 Structured Text (ST)

The structured text consists of a series of instructions which, as determined in high level languages, ("IF..THEN..ELSE") or in loops (WHILE..DO) can be executed.

Example:

```
IF value < 7 THEN

    WHILE value < 8 DO

    value := value + 1;

    END_WHILE;

END_IF;
```

Expressions

An expression is a construction which returns a value after its evaluation. Expressions are composed of operators and operands. An operand can be a constant, a variable, a function call, or another expression.

Valuation of expressions

The evaluation of expression takes place by means of processing the operators according to certain binding rules. The operator with the strongest binding is processed first, then the operator with the next strongest binding, etc., until all operators have been processed. Operators with equal binding strength are processed from left to right.

Below you find a table of the ST operators in the order of their binding strength:

Operation	Symbol	Binding strength
Put in parentheses	(expression)	Strongest binding
Function call	Function name (parameter list)	
Exponentiation	EXPT	
Negate Building of complements	- NOT	
Multiply Divide Modulo	* / MOD	
Add Subtract	+ -	
Compare	<,>,<=,>=	
Equal to Not Equal to	= <>	
Boolean AND	AND	
Boolean XOR	XOR	
Boolean OR	OR	Weakest binding

There are the following instructions in ST, arranged in a table together with example:

Instruction	Example
Assignment	A:=B; CV := CV + 1; C:=SIN(X);
Calling a function block and use of the FB version	CMD_TMR(IN := %IX5, PT := 300);A:=CMD_TMR.Q;
RETURN	RETURN;
IF	IF D<0.0 THEN C:=A; ELSIF D=0.0 THEN C:=B; ELSE C:=D; END_IF;
CASE	CASE INT1 OF 1: BOOL1 := TRUE; 2: BOOL2 := TRUE; ELSE BOOL1 := FALSE; BOOL2 := FALSE; END_CASE;
FOR	FOR I:=1 TO 100 BY 2 DO IF ARR[I] = 70 THEN J:=I; EXIT; END_IF; END_FOR;
WHILE	WHILE J<= 100 AND ARR[J] <> 70 DO J:=J+2; END_WHILE;
REPEAT	REPEAT J:=J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;
EXIT	EXIT;
Empty instruction	;

Instruction in structured text

The name already indicates, the structured text is designed for structure programming, i.e. ST offers predetermined structures for certain often used constructs such as loops for programming. This offers the advantages of low error probability and increased clarity of the program.

For example, let us compare two equally significant program sequences in IL and ST:

A loop for calculating powers of two in IL:

```
Loop:
LD    Counter
EQ       0
JMPC
end

LD
Var1
MUL
2
ST    Var1

LD
Counter
SUB
1
ST    Counter
JMP
Loop
End:
LD    Var1
ST    ERG
```

The same loop programmed in ST would produce:

```
WHILE Counter<>0 DO

    Var1:=Var1*2;

    Counter:=Counter-1;

END_WHILE
Erg:=Var1;
```

You can see, the loop in ST is not only faster to program, but is also significantly easier to read, especially in view of the convoluted loops in larger constructs.

The different structures in ST have the following significance:

Assignment operator

On the left side of an assignment there is an operand (variable, address) to which is assigned the value of the expression on the right side with the assignment operator :=

Example: Var1 := Var2 * 10;

After completion of this line Var1 has the tenfold value of Var2.

Calling function blocks in ST

A function block is called in ST by writing the name of the instance of the function block and then assigning the values of the parameters in parentheses. In the following example a timer is called with assignments for the parameters IN and PT. Then the result variable Q is assigned to the variable A.

The result variable, as in IL, is addressed with the name of the function block, a following point, and the name of the variable:

CMD_TMR(IN := %IX5, PT := 300);
A:=CMD_TMR.Q

2.2.3.1 FOR loop

With the FOR loop one can program repeated processes.
Syntax:

```
INT_Var :INT;


FOR <INT_Var> := <INIT_VALUE> TO <END_VALUE> {BY
<stepsize>} DO

    <instructions>

END_FOR;
```

The part in braces {} is optional.

The <Instructions> are executed as long as the counter <INT_Var> is not greater than the <END_VALUE>. This is checked before executing the <Instructions> so that the <instructions> are never executed if <INIT_VALUE> is greater than <END_VALUE>. When <Instructions> are executed, <INT_Var> is always increased by <Step size>. The step size can have any integer value. If it is missing, then it is set to 1. The loop must also end since <INT_Var> only becomes greater.

Example:

```
FOR counter:=1 TO 5 BY 1 DO

    Var1:=Var1*2;

END_FOR;

Erg:=Var1;
```

Let us assume that the default setting for Var1 is the value 1. Then it will have the value 32 after the FOR loop.

The <END_VALUE> can not be the limit of the counter <INT_VAR>.E.g. if the variable counter is of type SINT, and if <END_VALUE> is 127, you get an endless loop.

2.2.3.2 REPEAT loop

The REPEAT loop is different from the WHILE loop because the break-off condition is checked only after the loop has been executed. This means that the loop will run through at least once, regardless of the wording of the break-off condition.

Syntax:

```
REPEAT
    <instructions>
UNTIL <Boolean expression>
END_REPEAT;
```

The <Instructions> are carried out until the <Boolean expression> returns TRUE. If <Boolean expression> is produced already at the first TRUE evaluation, then <Instructions> are executed only once. If <Boolean_expression> never assumes the value TRUE, then the <Instructions> are repeated endlessly which causes a relative time delay.

The programmer must make sure that no endless loop is caused. He does this by changing the condition in the instruction part of the loop, for example by counting up or down one counter.

Example:

```
REPEAT
    Var1 := Var1*2;
    Counter := Counter-1;
UNTIL
    Counter=0
END_REPEAT
```

2.2.3.3 WHILE loop

The WHILE loop can be used like the FOR loop with the difference that the break-off condition can be any Boolean expression. This means you indicate a condition which, when it is fulfilled, the loop will be executed.

Syntax:

```
WHILE <Boolean expression> DO
    <instruction>
END_WHILE;
```

The <Instructions> are repeatedly executed as long as the <Boolean_printout> returns TRUE. If the <Boolean_printout> is already FALSE at the first evaluation, then the <Instructions> are never executed. If <Boolean_printout> never assumes the value FALSE, then the <Instructions> are repeated endlessly which causes a relative time delay.

The programmer must make sure that no endless loop is caused. He does this by changing the condition in the instruction part of the loop, for example, by counting up or down one counter.

Example:

```
WHILE counter<>0 DO
    Var1 := Var1*2;
```

```
    counter := counter-1;

END_WHILE
```

The WHILE and REPEAT loops are, in a certain sense, more powerful than the FOR loop since one doesn't need to know the number of cycles before executing the loop. In some cases one will, therefore, only be able to work with these two loop types. If, however, the number of the loop cycles is clear, then a FOR loop is preferable since it allows no endless loops.

2.2.3.4 IF instruction

With the IF instruction you can check a condition and, depending upon this condition, execute instructions.
Syntax:

```
IF <Boolean_printout1> THEN

<IF_instructions>

{ELSIF <Boolean_printout2> THEN

<ELSIF_instructions1>

.

.

ELSIF <Boolean_printout n> THEN

<ELSIF_instructions n-1>

ELSE

<ELSE_instructions>}

END_IF;
```

The part in braces {} is optional.

If the <Boolean_printout1> returns TRUE, then only the <IF_Instructions> are executed and none of the other instructions. Otherwise the Boolean expressions, beginning with <Boolean_printout2>, are evaluated one after the other until one of the expressions returns TRUE. Then only those instructions after this Boolean expression and be-fore the next ELSE or ELSIF are evaluated. If none of the Boolean expressions produce TRUE, then only the <ELSE_instructions> are evaluated.

Sample:

```
IF temp<17

THEN heating_on := TRUE;

ELSE heating_on := FALSE;

END_IF;
```

Here the heating is turned on when the temperature sinks below 17 degrees. Otherwise it remains off.

2.2.3.5 CASE instruction

With the CASE instructions one can combine several conditioned instructions with the same condition variable in one construct.
Syntax:

```
CASE <Var1> OF
<Value 1>:
    <instruction 1>
<Value 2>:
    <instruction 2>
    ...
<Value n>:
    <instruction n>
```

```
ELSE
    <ELSE-instruction>
END_CASE;
```

A CASE instruction is processed according to the following model:

- If the variable in <Var1> has the value <Value i>, then the instruction <Instruction i> is executed.
- If <Var1> has none of the indicated values, then the <ELSE Instruction> is executed.
- If the same instruction is to be executed for several values of the variables, then one can write these values one after the other separated by commas, and thus condition the common execution.

Example:

```
CASE INT1 OF
1, 5:
    BOOL1 := TRUE;
    BOOL3 := FALSE;
2:
    BOOL2 := FALSE;
    BOOL3 := TRUE;
ELSE
    BOOL1 := NOT BOOL1;
    BOOL2 := BOOL1 OR BOOL2;
END_CASE;
```

2.2.3.6 RETURN instruction

The RETURN instruction can be used for ending a function, depending upon a condition.

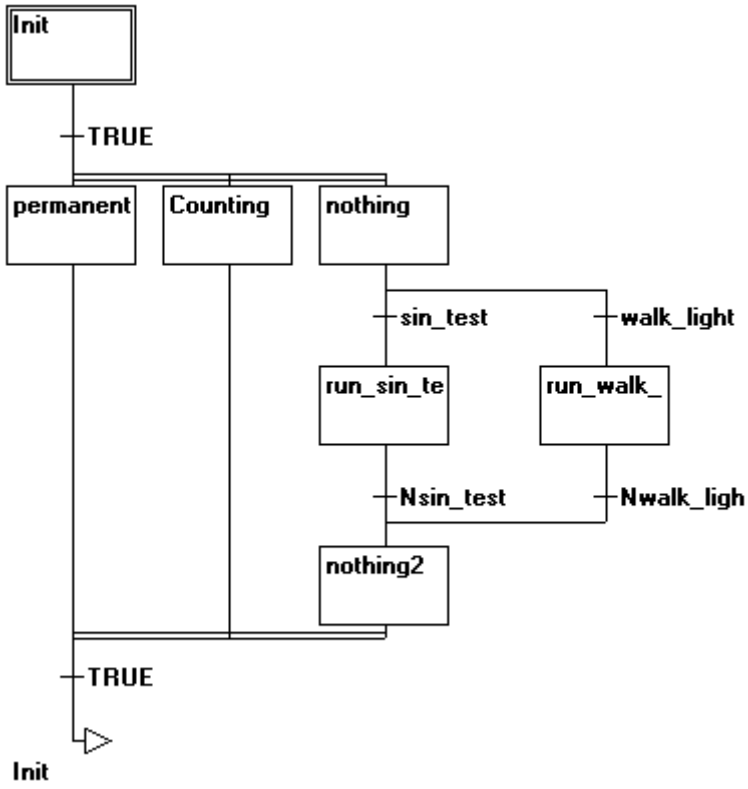
2.2.3.7 EXIT instruction

If the EXIT instruction appears in a FOR, WHILE, or REPEAT loop, then the innermost loop is ended, regardless of the break-off condition.

2.2.4 Sequential Function Chart (SFC)

The sequential function chart a graphically oriented language which makes it possible to describe the chronological order of different actions within a program.

Example for a network in the sequential function chart:



Step

A POU written in a sequential function chart consists of a series of steps which are connected with each other through directed connections (transitions).
There are two types of steps.

- The simplified type consists of an action and a flag which shows if the step is active. If the action of a step is implemented, then a small triangle appears in upper right corner of the step.
- An IEC step consists of a flag and one or more assigned actions. The associated actions appear to the right of the step.

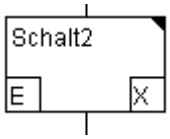
Action

An action can contain a series of instructions in IL or in ST, a lot of networks in FBD or in LD, or again in sequential function chart (SFC). With the simplified steps an action is always connected to a step. In order to edit an action, click twice with the mouse on the step to which the action belongs. Or select the step and select the menu command "Extras" "Zoom Action/Transition". In addition, one input or output action per step is possible.
Actions of IEC steps hang in the Object Organizer directly under their SFC-POU and are loaded with a doubleclick or by pressing <Enter> in their editor. New actions can be created with "Project" "Add Action"..

Entry or exit action

Additional to a step action you can add an entry action and an exit action to a step. An entry action is executed only once, right after the step has become active. An exit action is executed only once before the step is deactivated. A step with entry action is indicated by an "E" in the lower left corner, the exit action by an "X" in the lower right corner. The entry and exit action can be implemented in any language. In order to edit an entry or exit action, doubleclick in the corresponding corner in the step with the mouse. An entry and exit action can only be defined to a simplified step, but not to an IEC step.

Example of a step with entry and exit action:



Transition/Transition condition

Between the steps there are so-called transitions. A transition condition must have the value TRUE or FALSE. Thus it can consist of either a boolean variable, a boolean address or a boolean constant. It can also contain a series of instructions having a boolean result, either in ST syntax (e.g. (i<= 100) AND b) or in any language desired (see 'Extras" "Zoom Action/Transition"). But a transition may not contain programs, function blocks or assignments!

Note:

Besides transitions, inching mode can also be used to skip to the next step; see SFCtip and SFCtipmode.

Active Step

After calling the SFC POU, the action (surrounded by a double border) belonging to the initial step is executed first. A step, whose action is being executed, is called active. If the step is active, then the appropriate action is executed once per cycle. In Online mode active steps are shown in blue. To each step belongs a flag which saves the condition of the step. The step flag (active or inactive condition of the step) is shown by the logical value of a Boolean structure element <StepName>.x. This Boolean variable has the value TRUE if the appropriate step is active and FALSE if it is inactive. This variable is implicitly declared and can be used in any action and transition of the SFC POU.In a control cycle all actions are executed which belong to active steps. Thereafter the respective following steps of the active steps become active if the transition conditions of the following steps are TRUE. The currently active steps will be executed in the next cycle.

If the active step contains an output action, this will only be executed during the next cycle, provided that the transition following is TRUE.

IEC step

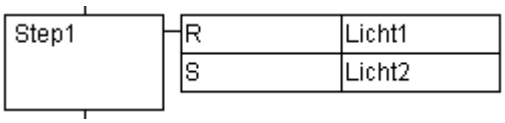
Along with the simplified steps the standard IEC steps in SFC are available. In order to be able to use IEC steps, you must link the library **TcSystem.Lib** into your project.
Any number of actions can be assigned to an IEC step. IEC actions are not fixed as input, step or output actions to a certain step as in the simplified steps, but are stored separately from the steps and can be reused many times within a POU. For this they must be associated to the single steps with the command **'Extras"Associate action'**.

Along with actions, Boolean variables can be assigned to steps.
The activation and deactivation of actions and boolean variables can be controlled using so-called qualifiers. Time delays are possible. Since an action can still be active, if the next step has been processed, for example through the qualifier S (Set), one can achieve concurrent processes.

An associated boolean variable is set or reset with each call of the SFC block. That means, that with each call the value changes from TRUE or FALSE or back again.

The actions associated with an IEC step are shown at the right of the step in a two-part box. The left field contains the qualifier, possibly with time constant, and the right field contains the action name.

An example for an IEC step with two actions:



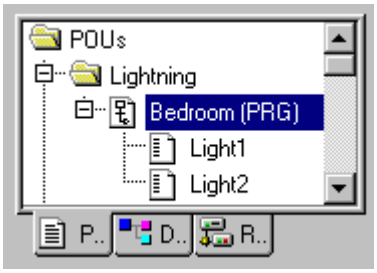
In order to make it easier to follow the processes, all active actions in on-line mode are shown in blue like the active steps. After each cycle a check is made to see which actions are active.

Pay attention here also to the restrictions on the use of time-qualifiers in actions that are repeatedly re-used within the same cycle.

If an action has been inactivated, it will be executed once more. That means, that each action is executed at least twice (also an action with qualifier P). In case of a call first the deactivated actions, then the active actions are executed, in alphabetical order each time.

Whether a newly inserted step is an IEC step depends upon whether the menu command "**Extras" "Use IEC-Steps**" has been chosen.

In the Object Organizer the actions hang directly underneath their respective SFC POUs. New actions can be created with "Project" "Add Action". In order to use IEC steps you must include in your project the special SFC library lecsfc.lib .



In order to associate the actions with IEC steps the following qualifiers are available:

N	Non-stored	The action active as long as the step.
R	overriding Reset	The action is deactivated.
S	Set (Stored)	The action is activated and remains active until a Reset.
L	time Limited	The action is activated for a certain time.
D	time Delayed	The action becomes active after a certain time as long as the step is still active.
P	Pulse	The action is executed just one time if the step is active.
SD	Stored and time Delayed	The action is activated after a certain time and remains active until a Reset.
DS	Delayed and Stored	The action is activated after a certain time as long as the step is still active and remains active up to a Reset.
SL	Stored and time Limited	The is action is activated for a certain time.

If the same action is used with qualifiers in two straight sequently steps, which influence the time flow, the time qualifier can´t become operative at the second use. To avoid this, an intermediate step has to be inserted. Thus the action status can be initialised anewd in the additional passing through cycle.

Implicit variables in SFC

There are implicitly declared variables in the SFC which can be used. A flag belongs to each step which stores the state of the step. The step flag (active or inactive state of the step) is called <StepName>.x for IEC steps or <StepName> for the simplified steps. This Boolean variable has the value TRUE when the associated step is active and FALSE when it is inactive. It can be used in every action and transition of the SFC block. One can make an enquiry with the variable <ActionName>.x. as to whether an IEC action is active or not.

Alternative branch

Two or more branches in SFC can be defined as alternative branches. Each alternative branch must begin and end with a transition. Alternative branches can contain parallel branches and other alternative branches. An alternative branch begins at a horizontal line (alternative beginning) and ends at a horizontal line (alternative end) or with a jump.

If the step which precedes the alternative beginning line is active, then the first transition of each alternative branch is evaluated from left to right. The first transition from the left whose transition condition has the value TRUE is opened and the following steps are activated.

For IEC steps the implicit variables <StepName>.t can be used to enquire about the active time of the steps. Implicit variables can also be accessed by other programs. Example: boolvar1:=sfc.step1.x; Here, step1.x is the implicit boolean variable representing the state of IEC step step1 in POU sfc1.

Parallel branch

Two or more branches in SFC can be defined as parallel branches. Each parallel branch must begin and end with a step. Parallel branches can contain alternative branches or other parallel branches. A parallel branch begins with a double line (parallel beginning) and ends with a double line (parallel end) or with a jump.

If the parallel beginning line of the previous step is active and the transition condition after this step has the value TRUE, then the first steps of all parallel branches become active (see active step). These branches are now processed parallel to one another. The step after the parallel end line becomes active when all previous steps are active and the transition condition before this step produces the value TRUE.

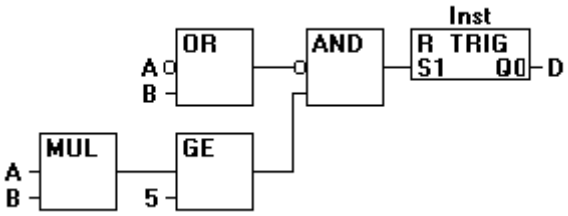
Jump

A jump is a connection to the step whose name is indicated under the jump symbol. Jumps are required because it is not allowed to create connections which lead upward or cross each other.

2.2.5 Function Block Diagram (FBD)

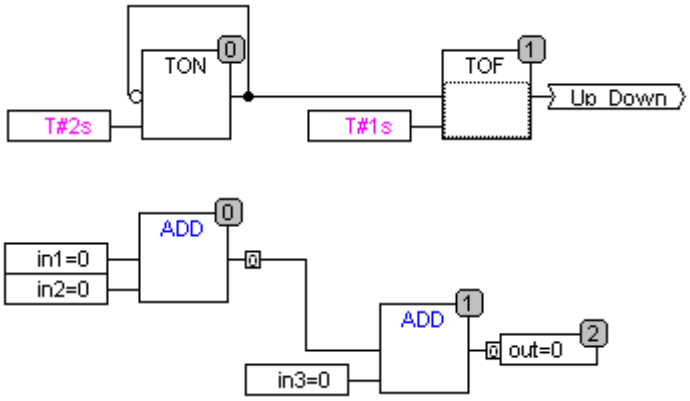
The function block diagram is a graphically oriented programming language. It works with a list of networks whereby each network contains a structure which represents either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

An example of a typical network in the function block diagram as it could appear in TwinCAT PLC Control :



2.2.6 The Continuous Function Chart Editor (CFC)

The continuous function chart editor does not operate like the function block diagram FBD with networks, but rather with freely placeable elements. This allows feedback, for example. Two examples of a network in the continuous function chart editor, as they would typically appear:



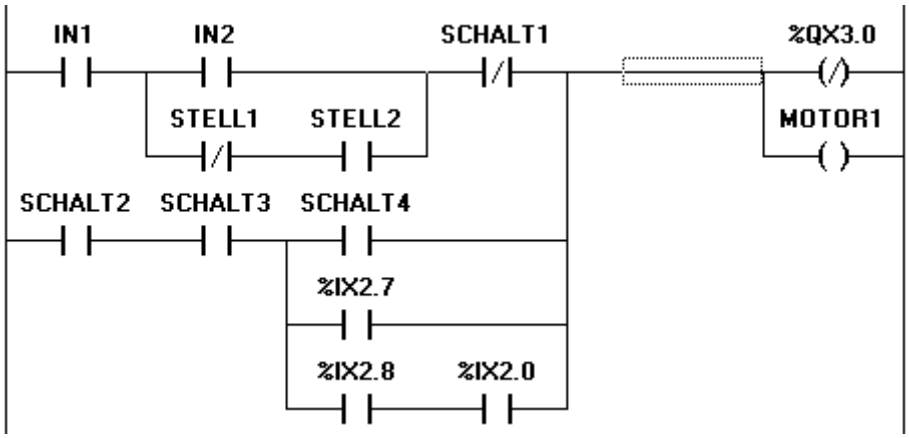
2.2.7 Ladderdiagram (LD)

The ladder diagram is also a graphics oriented programming language which approaches the structure of an electric circuit.

On the one hand, the ladder diagram is suitable for constructing logical switches, on the other hand one can also create networks as in FBD. Therefore the LD is very useful for controlling the call of other POUs. More about this later. The ladder diagram consists of a series of networks. A network is limited on the left and right sides by a left and right vertical current line. In the middle is a circuit diagram made up of contacts, coils, and connecting lines.

Each network consists on the left side of a series of contacts which pass on from left to right the condition "ON" or "OFF" which correspond to the Boolean values TRUE and FALSE. To each contact belongs a Boolean variable. If this variable is TRUE, then the condition is passed from left to right along the connecting line. Otherwise the right connection receives the value OFF.

Example of a typical network in the ladder diagram as it could appear in TwinCAT PLC Control :



Contact

Each network in LD consists on the left side of a network of contacts (contacts are represented by two parallel lines: | |) which from left to right show the condition "On" or "Off". These conditions correspond to the Boolean values TRUE and FALSE. A Boolean variable belongs to each contact. If this variable is TRUE, then the condition is passed on by the connecting line from left to right, otherwise the right connection receives the value "Out". Contacts can be connected in parallel, then one of the parallel branches must transmit the value "On" so that the parallel branch transmits the value "On"; or the contacts are connected in series, then contacts must transmit the condition "On" so that the last contact transmits the "On" condition. This therefore corresponds to an electric parallel or series circuit. A contact can also be negated, recognizable by the slash in the contact symbol: |/|. Then the value of the line is transmitted if the variable is FALSE.

Coil

On the right side of a network in LD there can be any number of so-called coils which are represented by parentheses:( ). They can only be in parallel. A coil transmits the value of the connections from left to right and copies it in an appropriate Boolean variable. At the entry line the value ON (corresponds to the Boolean variable TRUE) or the value OFF (corresponding to FALSE) can be present. Contacts and coils can also be negated (in the example the contact SWITCH1 and the coil %QX3.0 is negated). If a coil is negated (recognizable by the slash in the coil symbol: (/)), then it copies the negated value in the appropriate Boolean variable. If a contact is negated, then it connects through only if the appropriate Boolean value is FALSE.

Function blocks in the ladder diagram

Along with contacts and coils you can also enter function blocks and programs. In the network they must have an input and an output with Boolean values and can be used at the same places as contacts, that is on the left side of the LD network

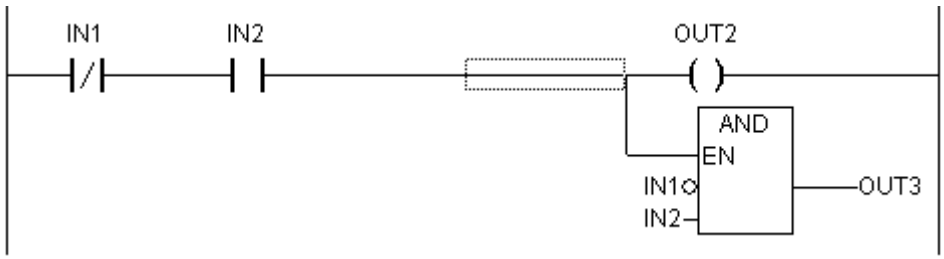
Set/Reset coils

Coils can also be defined as set or reset coils. One can recognize a set coil by the "S" in the coil symbol: (S)) It never writes over the value TRUE in the appropriate Boolean variable. That is, if the variable was once set at TRUE, then it remains so. One can recognize a reset coil by the "R" in the coil symbol: (R)) It never writes over the value FALSE in the appropriate Boolean variable: If the variable has been once set on FALSE, then it remains so.

LD as FBD

When working with LD it is very possible that you will want to use the result of the contact switch for controlling other POUs. On the one hand you can use the coils to put the result in a global variable which can then be used in another place. You can, however, also insert the possible call directly into your LD network. For this you introduce a POU with EN input. Such POUs are completely normal operands, functions, programs, or function blocks which have an additional input which is labeled with EN. The EN input is always of the BOOL type and its meaning is: The POU with EN input is evaluated when EN has the value TRUE. An EN POU is wired parallel to the coils, whereby the EN input is connected to the connecting line between the contacts and the coils. If the ON information is transmitted through this line, this POU will be evaluated completely normally. Starting from such an EN POU, you can create networks similar to FBD.

Part of a LD Network with an EN POU



2.2.8 Debugging, Online Functions

Sampling Trace [▶ 175]

The sampling trace allows you to trace the value sequence of variables, depending upon the so-called trigger event. This is the rising edge or falling edge of a previously defined Boolean variable (trigger variable). TwinCAT PLC Control permits the tracing of up to 20 variables. There is a 64 kB ring buffer.

Debugging [▶ 103]

The debugging functions of TwinCAT PLC Control make it easier for you to find errors.

Breakpoint

A breakpoint is a place in the program at which the processing is stopped. Thus it is possible to look at the values of variables at specific places within the program.
Breakpoints can be set in all editors. In the text editors breakpoints are set at line numbers, in FBD and LD at network numbers, and in SFC at steps.
No breakpoints can be set in function block instances.

Single step

Single step means:

- In IL: Execute the program until the next CAL, LD or JMP command.
- In ST: Execute the next instruction.
- In FBD, LD: Execute the next network.
- In SFC: Continue the action until the next step.
- In CFC: Execute the next POU (Box) in the CFC program.

By proceeding step by step you can check the logical correctness of your program.

Single cycle

If Single cycle has been chosen, then the execution is stopped after each cycle.

Change values online

During operations variables can be set once at a certain value (write value) or also described again with a certain value after each cycle (force value). In online mode one also can change the variable value by double click on the value. By that boolean variables change from TRUE to FALSE or the other way round, for each other types of variables one gets the dialog **Write Variable xy**, where the actual value of the variable can be edited.

Monitoring

In Online mode, all displayable variables are read from the controller and displayed in real time. You will find this display in the declarations and program editor; you can also read out current values of variables in the watch and recipe manager and in a visualization. If variables from instances of function blocks are to be monitored, the corresponding instance must first be opened.

In **monitoring VAR_IN_OUT variables**, the de-referenced value is output. With a simple click on the cross or a double-click on the line, the display is either expanded or truncated.

In monitoring pointers, both the pointer and the de-referenced value are output in the declaration portion. In the program portion, only the pointer is output:

+ --pointervar = '<'pointervalue'>'

POINTERs in the de-referenced value are also displayed accordingly.
With a simple click on the cross or a double-click on the line, the display is either expanded or truncated.

Monitoring of ARRAY components: In addition to array components indexed by a constant, components are also displayed which are indexed by a variable:
anarray[1] = 5
anarray[i] = 1
If the index consists of an expression (e.g. [i+j] or [i+1]), the component can not be displayed.

Simulation

During the simulation the created PLC program is not processed in the PLC, but rather in the calculator on which TwinCAT PLC Control is running. All online functions are available. That allows you to test the logical correctness of your program without PLC hardware.

Simulation mode is only available for the Buscontroller (BCxx00). If you use TwinCAT on the PC (Code generation i386) you can simulate the program direct in the runtime system without having physical I/O.

Log

The log chronologically records user actions, internal processes, state changes and exceptions during Online mode processing. It is used for monitoring and for error tracing.

2.2.9 IEC 61131-3

The standard IEC 61131-3 is an international standard for programming languages of Programmable Logic Controllers. The programming languages offered in TwinCAT PLC Control conform to the requirements of the standard. According to this standard, a program consists of the following elements:

- Structures
- POUs
- Global Variables

Literature related to IEC 61131-3:

- Flavio Bonifatti et al.: IEC 1131-3 Programming Methology, Seyssins: CJ International, 1997. ISBN 2-9511585-0-5
- Karl-Heinz John, Michael Tiegelkamp: Iec 61131-3 Programming Industrial Automation Systems, Berlin: Springer-Verlag, 2001. ISBN 3-540677526

• R. W. Lewis: Programming industrial control systems using IEC 1131-3, London: IEC Publishing, 1998. ISBN 0 85296 950 3

3 Sample Program

Let us now start to write a small example program. It is for a small traffic signal unit which is supposed to control two trafficsignals at an intersection. The red/green phases of both trafficsignals alternate and, in order to avoid accidents, we will insert between the phases yellow or yellow/red transitional phases. The latter will be shorter than the former. In this example you will see how time dependent programs can be shown with the language ressources of the IEC61131-3 standard, how one can edit the different languages of the standard with the help of TwinCAT PLC Control.

Create POU

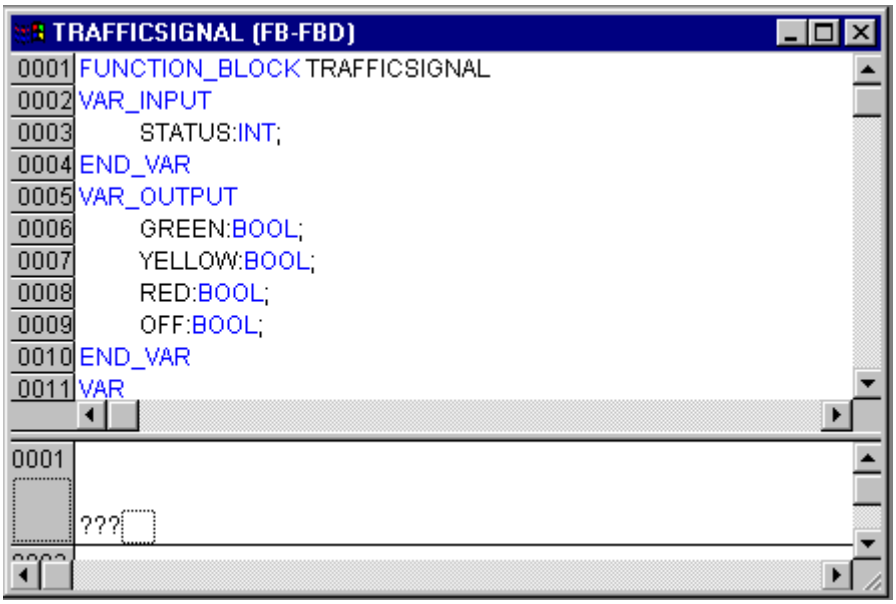
First start TwinCAT PLC Control and choose "File" "New". In the dialog box which appears, the first POU should have the name PLC_PRG. The type of POU should definitely be a program. In this case we choose as the language of this POU the sequential function chart (SFC). Now create two more objects with the command "Project" "Object Add" with the menu bar or with the context menu (press right mouse button in the Object Organizer). A function block in the language Function Block Diagram (FBD) by the names of TRAFFICSIGNAL, along with a POU WAIT, also of the type function block, which we want to program as an instruction list (IL).

In the POU TRAFFICSIGNAL we will assign the individual trafficsignal phases to the lights, i.e. we will make sure that the red light is lit red in the red phase and in the yellow/red phase, the yellow light in the yellow and yellow/red phases, etc.

In WAIT we will program a simple timer which as input will receive the length of the phase in milliseconds, and as output will produce TRUE as soon as the time period is finished.
PLC_PRG will combine everything at the end so that the right light lights up at the right time for the desired time period.

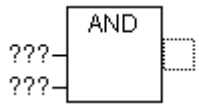
Let us now turn to the POU TRAFFICSIGNAL. In the declaration editor you declare as input variable (between the keywords VAR_INPUT and END_VAR) a variable named STATUS of the type INT. STATUS will have five possible conditions, that is one for the TRAFFICSIGNAL phases green, yellow, yellow/red, red, and off.

Therefore our TRAFFICSIGNAL has four outputs, that is RED, YELLOW, GREEN, and OFF. You should declare these four variables. Then the declaration part of our function block TRAFFICSIGNAL will look like this:

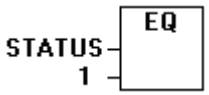


Function block TRAFFICSIGNAL, declaration part

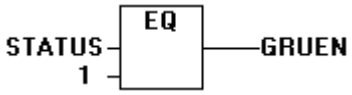
Now it is time to determine from the input STATUS of the POU the values of the output variables. For this go into the body of the POU. Click on the field to the left beside the first network (the gray field with the number 1). You have now selected the first network. Now choose the menu item "Insert" "Operator". In the first network a box is inserted with the operator AND and two inputs:



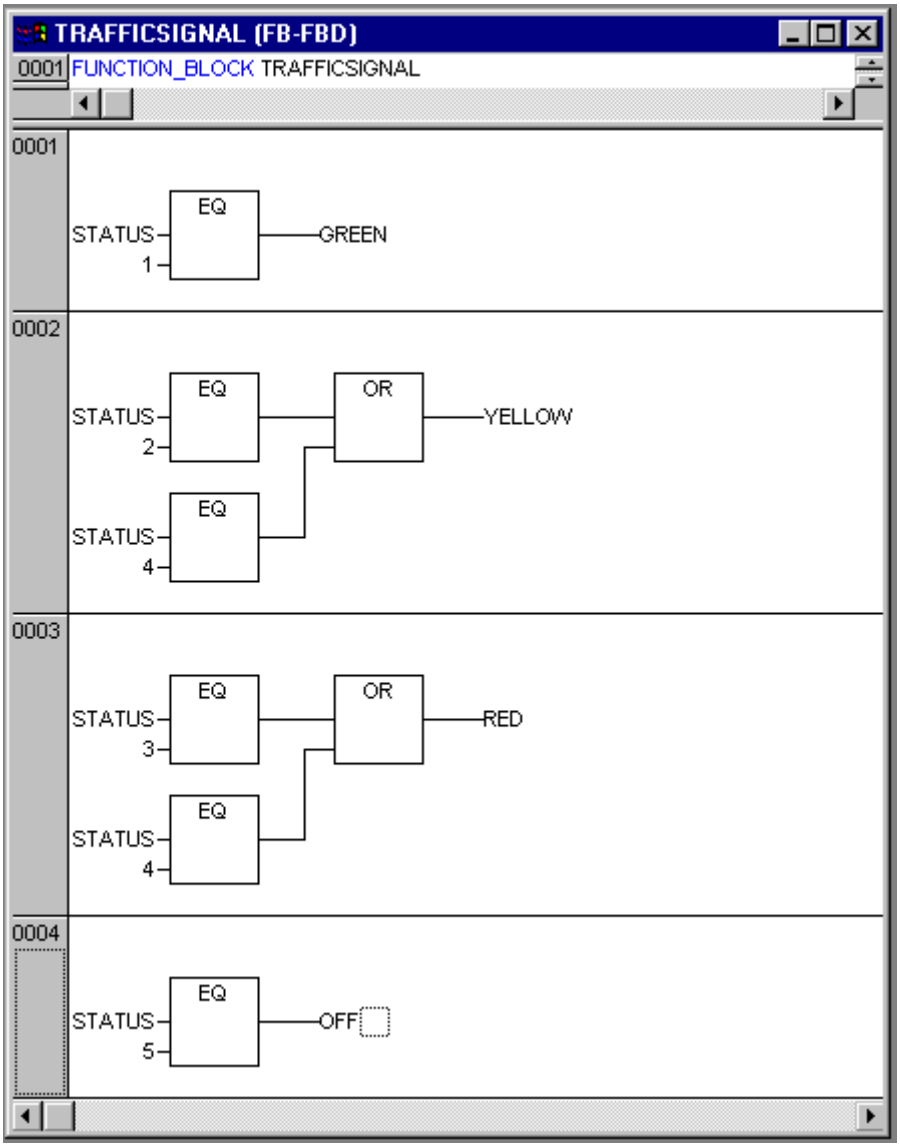
Click on the text AND with the mouse pointer and change the text into EQ. Select the three question marks from the upper of the two inputs and enter the variable STATUS. Then select the lower of the three question marks and put a 1 underneath it. You get the following network:



Click now on a place behind the EQ Box. Now the output of the EQ operation is selected. Choose "Insert" "Assignment". Change the three question marks ??? to GREEN. You now have created a network with the following structure:



STATUS is compared with 1, the result is assigned to GREEN. This network thus switches to GREEN if the preset state value is 1.For the other TRAFFICSIGNAL colors or for OFF we need three more net-works. You create them with the command "Insert" "Network (after)". These networks should be set up as in the example. The finished POU now is as follows:



In order to insert an operator in front of another operator, you must select the place where the input to which you want to attach the operator feeds into the box. Then use the command "Insert" "Operator". Otherwise you can setup these networks in the same way as the first network. Now our first POU has been finished. TRAFFICSIGNAL, according to the input of the value STATUS, controls whichever light color we wish.

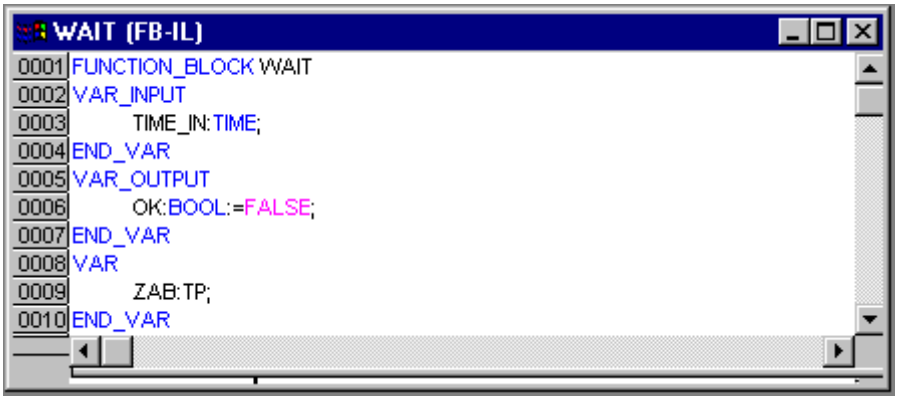
For the timer in the POU WAIT we need a POU from the standard library. Therefore, open the library manager with "Window" "Library Manager". Choose "Insert" "Additional library". The dialog box appears for opening files. From the list of the libraries choose standard.lib.

Now let us turn to the POU WAIT. This POU is supposed to become a timer with which we can determine the length of the time period of each TRAFFICSIGNAL phase. Our POU receives as input variable a variable TIME of the type TIME, and as output it produces a Boolean value which we want to call OK and which should be TRUE when the desired time period is finished. We set this value with FALSE by inserting at the end of the declaration (before the semicolon, however) " := FALSE ".

For our purposes we need the POU TP, a clock generator. This has two inputs (IN, PT) and two outputs (Q, ET). TP does the following:

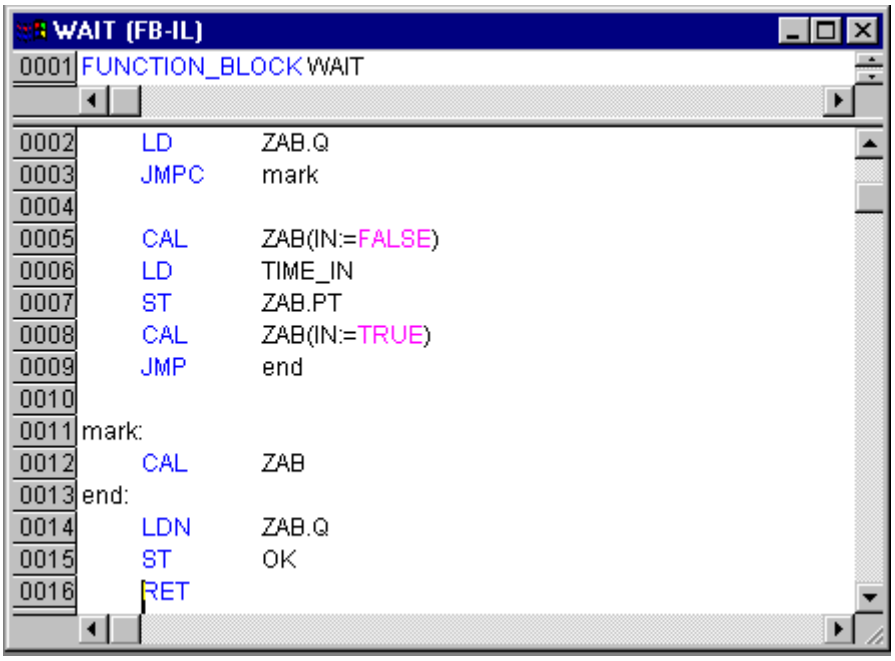
As long as IN is FALSE, ET is 0 and Q is FALSE. As soon as IN provides the value TRUE, the time is calculated at the output ET in milliseconds. When ET reaches the value PT, then ET is no longer counted. Meanwhile Q produces TRUE as long as ET is smaller than PT. As soon as the value PT has been reached, then Q produces FALSE again. In addition you will find a short description of all POUs from the standard library in the appendix.

In order to use the POU TP in the POU WAIT we must create a local instance from TP. For this we declare a local variable ZAB (for elapsed time) of the type TP (between the keywords VAR, END_VAR). The declaration part of WAIT thus looks like this:



Function Block WAIT, Declaration Part

In order to create the desired timer, the body of the POU must be programmed as follows:



Function Block WAIT, Instruction Part

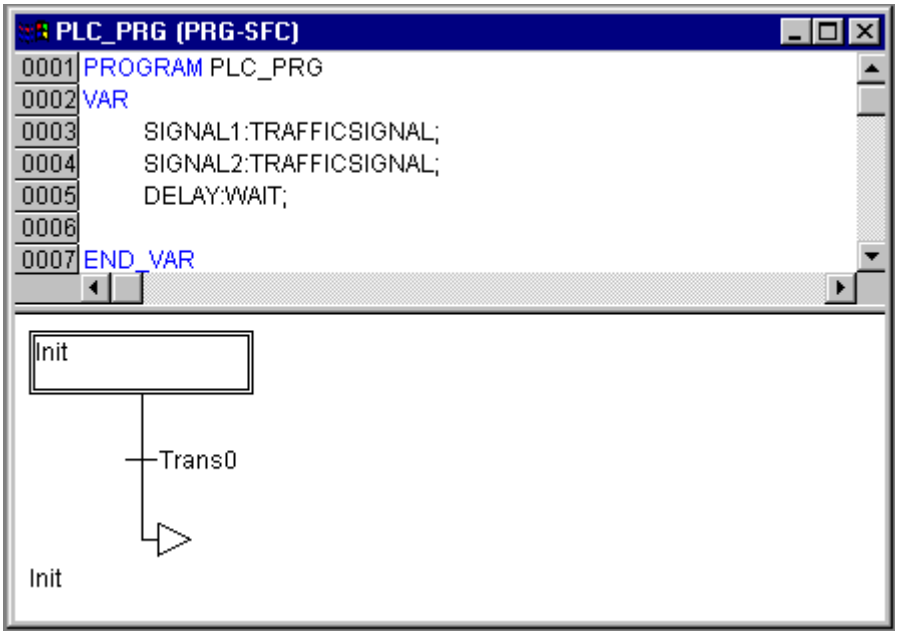
At first the question is posed whether Q has already been set at TRUE (as though the counting had already been executed), in this case we change nothing with the occupation of ZAB, but rather we call the function block ZAB without input (in order to check whether the time period is already over).

Otherwise we set the variable IN in ZAB at FALSE, and therefore at the same time ET at 0 and Q at FALSE. In this way all variables are set at the desired initial condition. Now we assign the necessary time from the vari-able TIME into the variable PT, and call ZAB with IN:=TRUE. In the function block ZAB the variable ET is now calculated until it reaches the value TIME, then Q is set at FALSE.

The negated value of Q is saved in OK after each execution of WAIT. As soon as Q is FALSE, then OK produces TRUE.

The timer is finished at this point. Now it is time to combine our two function blocks WAIT and TRAFFICSIGNAL in the main program PLC_PRG.

First we declare the variables we need. They are two instances of the function block TRAFFICSIGNAL (TRAFFICSIGNAL1, TRAFFICSIGNAL2) and one of the type WAIT (DELAY as delay). PLC_PRG now appears as follows:



Program PLC_PRG, First Expansion Level, Declaration Part

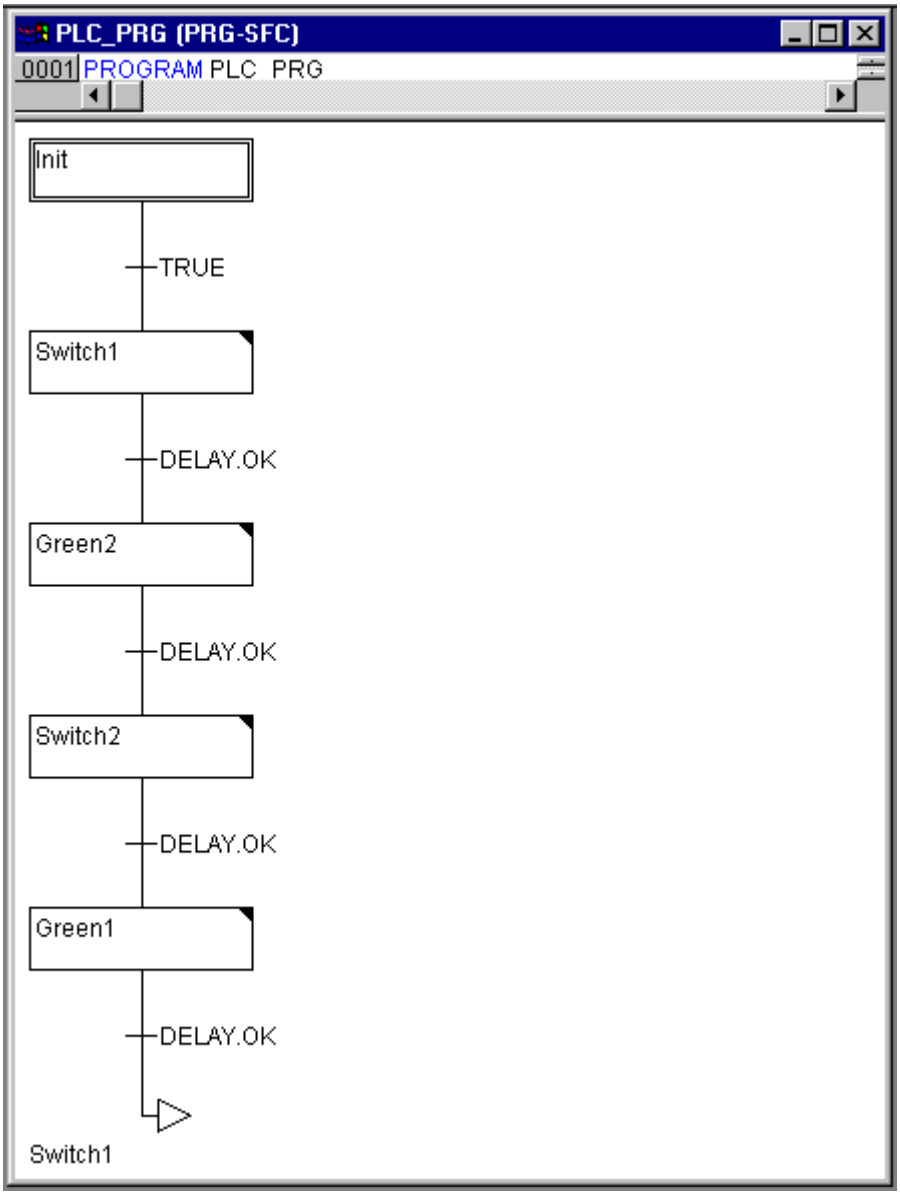
The beginning diagram of a POU in SFC always consists of an action "Init" of a following transition "Trans0" and a jump back to Init. We should expand that somewhat.

First let us determine the structure of the diagrams before we program the individual action and transitions. First we need for each TRAFFICSIGNAL phase a step.. Insert it by marking Trans0 and choosing "Insert" "Step transition (behind)". Repeat this procedure three more times.

If you click directly on the name of a transition or a step, then this is marked and you can change it. Name the first transition after Init "TRUE", and all other transitions "DELAY.OK".

The first transition always switches through, and all others switch through when DELAY in OK produces TRUE, i.e. when the set time period is finished.

The steps (from top to bottom) receive the names Switch1, Green2, Switch2, Green1, whereby Init of course keeps its Name. "Switch" should mean each time a yellow phase, at Green1 TRAFFICSIGNAL1 at Green2 TRAFFICSIGNAL2 will be green. Finally change the return address of Init after Switch1. If you have done everything right, then the diagram should be as follows:

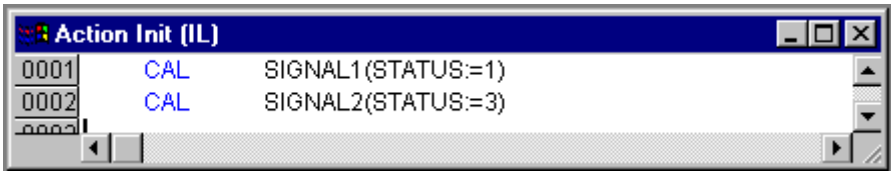


Program PLC_PRG, First Expansion Level, Instruction Part

Now we have to finish programming the individual steps. If you doubleclick on the field of a step, then you open a dialog to open a new action. In our case we will use IL (Instruction list).

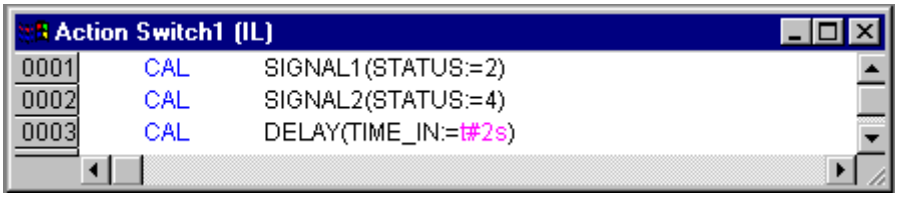
Actions and transition conditions

In the action of the step Init the variables are initialized, the STATUS of TRAFFICSIGNAL1 should be 1 (green). The state of TRAFFICSIGNAL2 should be 3 (red). The action Init then is as follows:



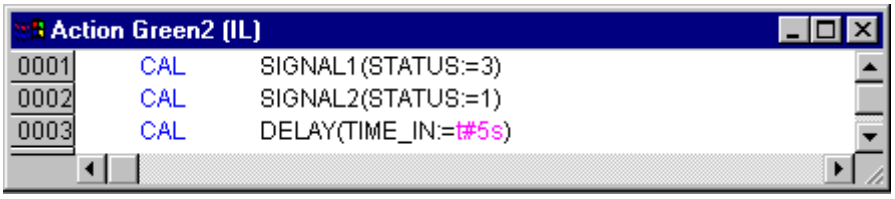
Action Init

Switch1 changes the State of TRAFFICSIGNAL1 to 2 (yellow), and that of TRAFFICSIGNAL2 to 4 (yellow-red). In addition, a time delay of 2000 milli-seconds is now set. The action is now as follows:



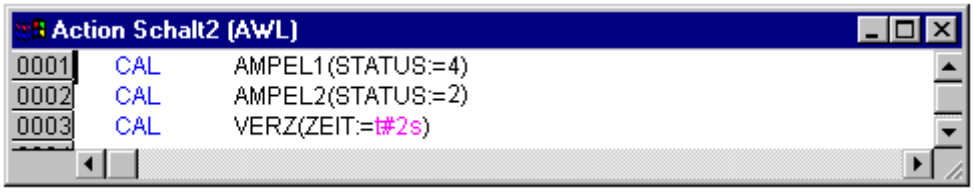
Action Switch1

With Green2 TRAFFICSIGNAL1 is red (STATUS:=3), TRAFFICSIGNAL2 green (STATUS:=1), and the delay time is at 5000.



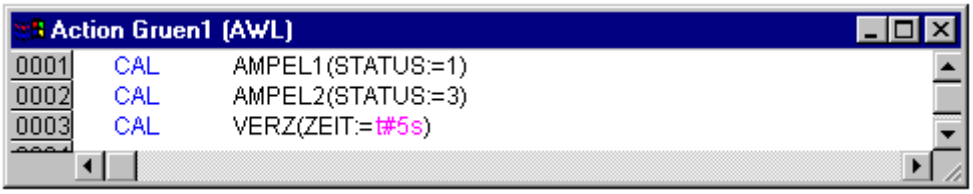
Action Green2

At Switch2 the STATUS of TRAFFICSIGNAL1 changes to 4 (yellow-red), that of TRAFFICSIGNAL2 to 2 (yellow). A time delay of 2000 milliseconds is now set.



Action Switch2

With Green1 TRAFFICSIGNAL1 is green (STATUS:=1), TRAFFICSIG-NAL2 is red (STATUS:=3), and the time delay is set at 5000.

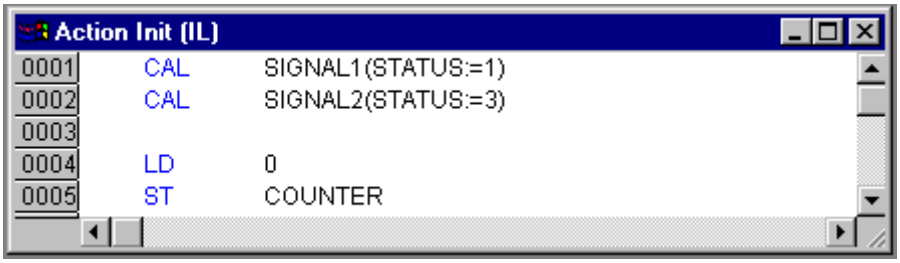


Action Green1

The first expansion phase of our program is completed. Now you can com-pile it and also test the simulation.

In order to ensure that our diagram has at least one alternative branch, and so that we can turn off our traffic light unit at night, we now include in our program a counter which, after a certain number of TRAFFICSIGNAL cycles, turns the unit off.

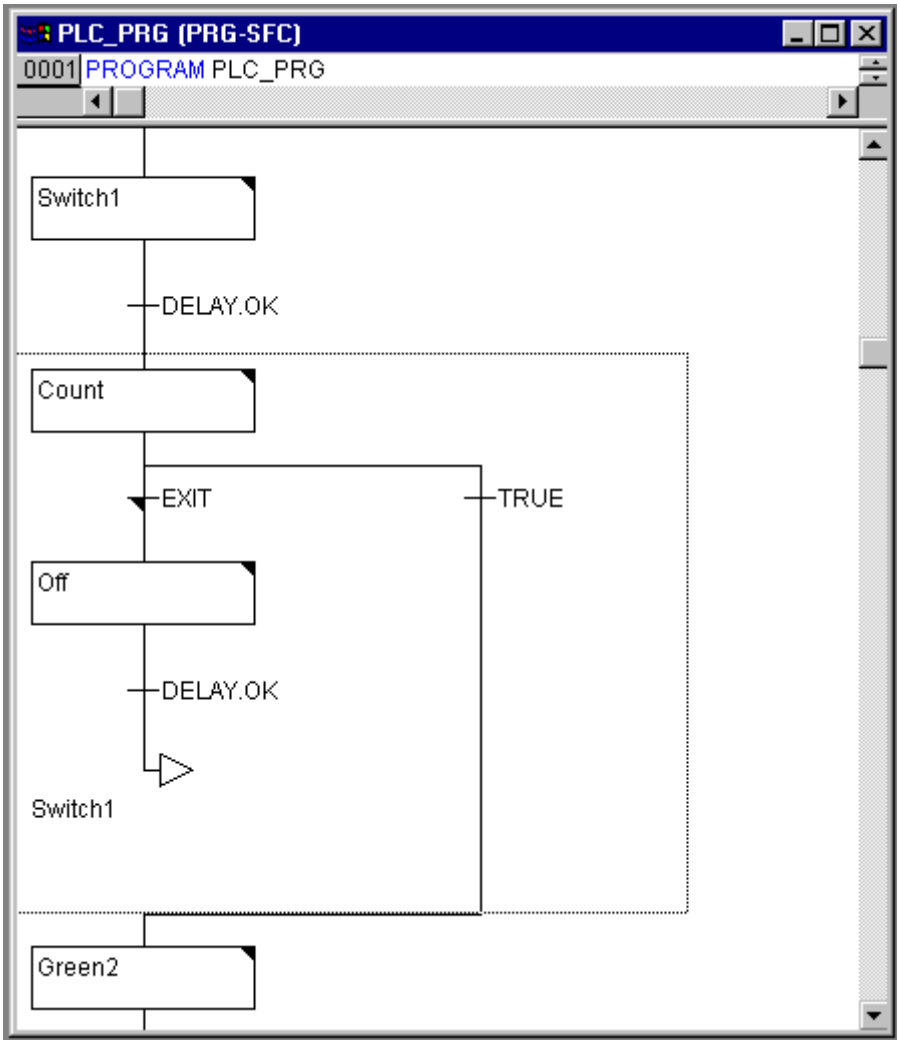
At first we need a new variable COUNTER of the type INT. Declare this as usual in the declaration part of PLC_PRG, and initialize it in Init with 0.



Action Init, Second Version

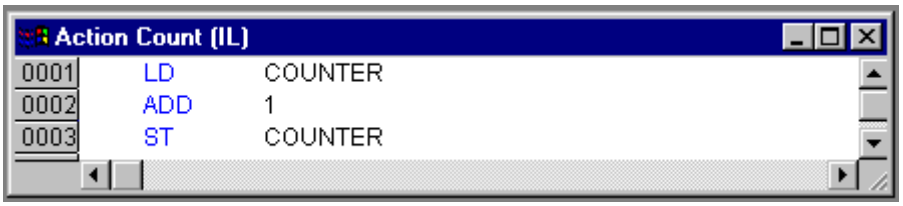
Now select the transition after Switch1 and insert a step and then a transition. Select the resulting transition and insert an alternative branch to its left. After the left transition insert a step and a transition. After the resulting new transition insert a jump after Switch1. Select the transition below the new step. Open the context menu and select "Zoom action/transition". Select IL for the language of the transition.

Name the new parts as follows: the upper of the two new steps should be called "Count" and the lower "Off". The transitions are called (from top to bottom and from left to right) EXIT, TRUE and DELAY.OK. The new part should therefore look like the part with the black border:



Traffic light unit

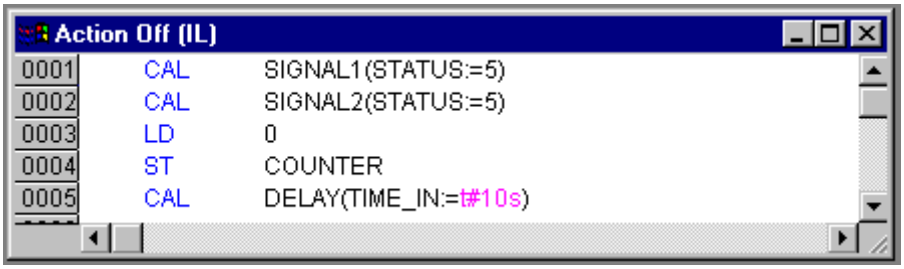
Now two new actions and a new transition condition are to be implemented. At the step Count, the only thing that happens is that COUNTER is increased by one:



The EXIT transition checks whether the counter is greater than a certain number, let us say 7:



At Off the state of both lights is set at (OFF), the COUNTER is reset to 0, and a time delay of 10 seconds is set:

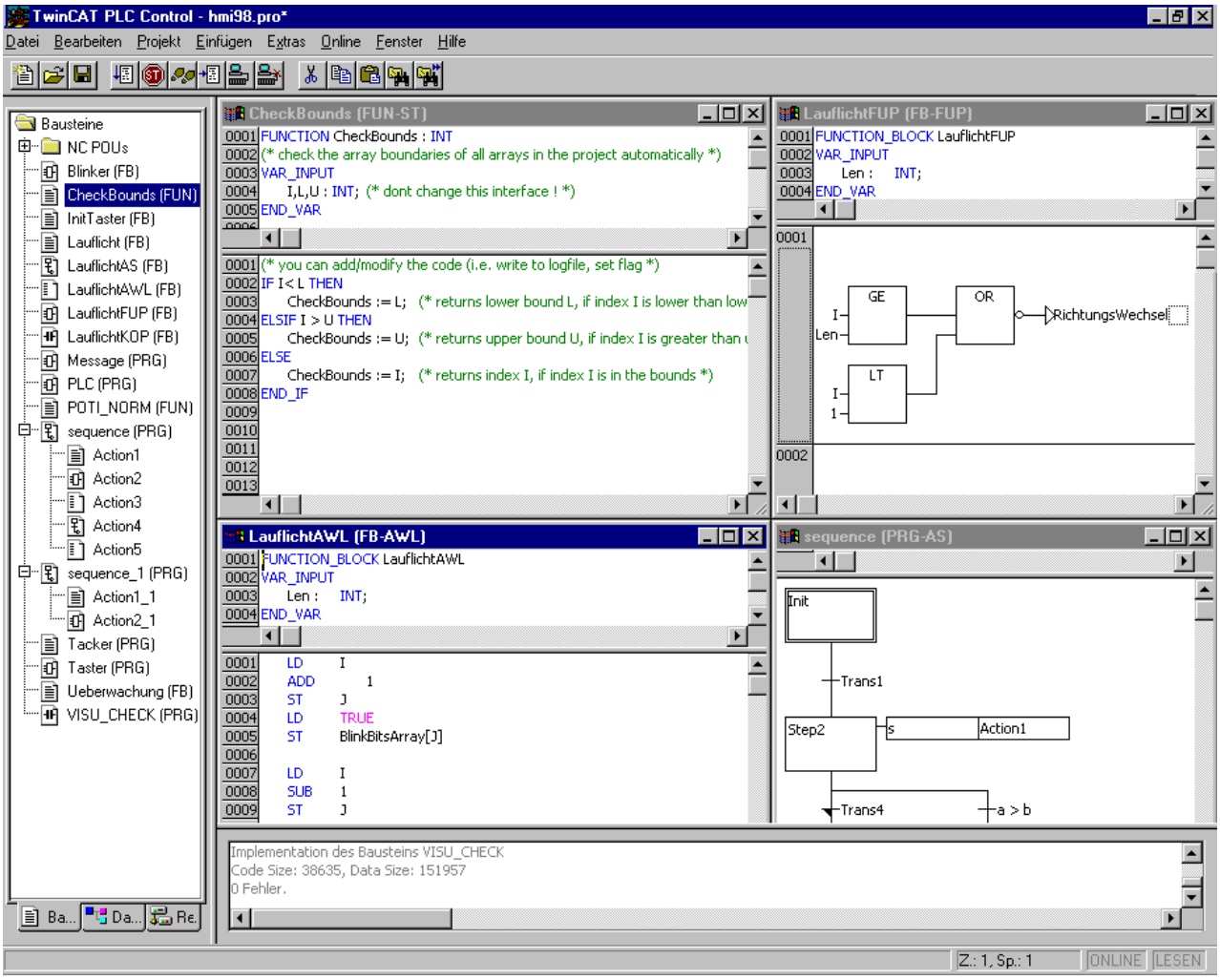


In our hypothetical situation, night falls after seven TRAFFICSIGNAL cycles, for ten seconds the TRAFFICSIGNAL turns itself off, then we have daylight again, the traffic light unit turns itself on again, and the whole process starts again from the beginning.

Now test your program. For this you must compile it ("Project" "Rebuild all") login ("Online" "Login" and then load it "Online" "Download"). If you now select "Online" "Run", the chronological order of the individual steps of your main program can be followed. The window of the POU PLC_PRG has now changed to the monitor window. Click twice on the plus sign in the declaration editor, the variable display drops down, and you can see the values of the individual variables.

4 Individual Components

4.1 The Main Window

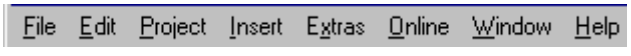


The following elements are found in the main window of TwinCAT PLC Control (from top to bottom):

- The menu bar
- The tool bar (optional); with buttons for faster selection of menu commands.
- The Object Organizer with register cards for POUs, Data types, and Ressources
- A vertical screen divider between the Object Organizer and the Work space of TwinCAT PLC Control
- The Work space in which the editor windows are located
- The message window (optional)
- The Status bar (optional); with information about the current status of the project

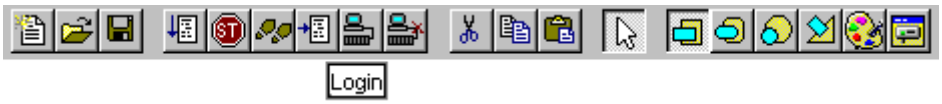
Menu bar

The menu bar is located at the upper edge of the main window. It contains all menu commands.



Tool bar

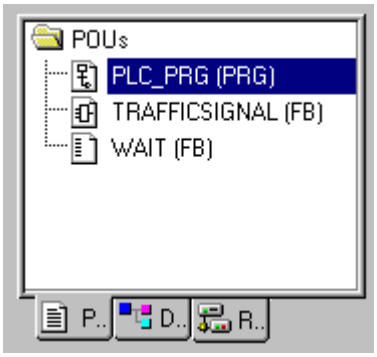
By clicking with the mouse on a symbol you can select a menu command more quickly. The choice of the available symbols automatically adapts itself to the active window. The command is only carried out when the mouse button is pressed on the symbol and then released. If you hold the mouse pointer for a short time on a symbol in the tool bar, then the name of the symbol is shown in a Tooltip. In order to see a description of each symbol on the tool bar, select in Help the editor about which you want information and click on the tool bar symbol in which you are interested. The display of the tool bar is optional (see "Project" "Options" category Desktop).



Object Organizer

The Object Organizer is always located on the left side of TwinCAT PLC Control. At the bottom there are four register cards with symbols for the four types of objects POUs, Data types, and Ressources. In order to change between the respective object types click with the mouse on the corresponding register card or use the left or right arrow key.

You will learn in chapter Objects [▶ 87], how to work with the objects in the Object Organizer.



Object Organizer

Screen divider

The screen divider is the border between two non-overlapping windows. In TwinCAT PLC Control there are screen dividers between the Object Organizer and the Work space of the main window, between the interface (declaration part) and the implementation (instruction part) of POUs and between the Work space and the message window.

You can move the screen divider with the mouse pointer. You do this by moving the mouse with the left mouse button pressed.

Make sure the the screen divider always remains at its absolute position, even when the window size has been changed. If it seems that the screen divider is no longer present, then simply enlarge your window.

Work space

The Work space is located on the right side of the main window in TwinCAT PLC Control. All editors for objects and the library manager are opened in this area. You find the description of the editors in the chapter Editors in TwinCAT PLC Control, The Ressources, and Library Manager. Under the menu item "Window" you find all commands for window management.

Message window

The message window is separated by a screen divider underneath the work space in the main window. It contains all messages from the previous compilations, checks , or comparisons. If you doubleclick with the mouse in the message window on a message or press <Enter>, the editor opens with the object. The

relevant line of the object is selected. With the commands "Edit" "Next error" and "Edit" "Previous error" you can can quickly jump between the error messages. The display of the message window is optional (see "Window" "Messages").

Status bar

The status bar at the bottom of the window frame of the main window in TwinCAT PLC Control gives you information about the current project and about menu commands. If an item is relevant, then the concept appears on the right side of the status bar in black script, otherwise in gray script. When you are working in online mode, the concept Online appears in black script. If you are working in the offline mode it appears in gray script. In Online mode you can see from the status bar whether you are in the simulation (SIM), the program is being processed (RUNS), a breakpoint is set (BP), or variables are being forced (FORCE).
With text editor the line and column number of the current cursor position is indicated (e.g. Line:5, Col.:11). If you have chosen a menu command but haven't yet confirmed it, then a short description appears in the status bar. The display of the statusbar is optional (see "Project" "Options" category Desktop).

Context menu Shortcut: <Shift>+<F10>

Instead of using the menu bar for executing a command, you can use the right mouse button. The menu which then appears contains the most frequently used commands for a selected object or for the active editor. The choice of the available commands adapts itself automatically to the active window.

4.2 Options

About TwinCAT PLC Control there can be of course only one viewpoint. In TwinCAT PLC Control, however, you can configure the view of the main window (and have more than one viewpoint). In addition you can make other settings. For this you have the command "Project" Options" at your disposal. The settings you make thereby are, unless determined otherwise, saved in the file "TwinCAT PLC Control.ini" and restored at the next TwinCAT PLC Control startup.

'Project' 'Options'

With this command the dialog box for setting options is opened. The options are divided into different categories. Choose the desired category on the left side of the dialog box by means of a mouse click or using the arrow keys and change the options on the right side.

You have at your disposal the following categories:

- Load & Save [▶ 44]
- User information [▶ 45]
- Editor [▶ 45]
- Desktop [▶ 48]
- Color [▶ 49]
- Directories [▶ 50]
- Log [▶ 51]
- Build [▶ 51]
- Passwords [▶ 53]
- Sourcedownload [▶ 54]
- Symbol configuration [▶ 55]
- Database connection [▶ 57]
- Macros [▶ 62]
- TwinCAT [▶ 64]

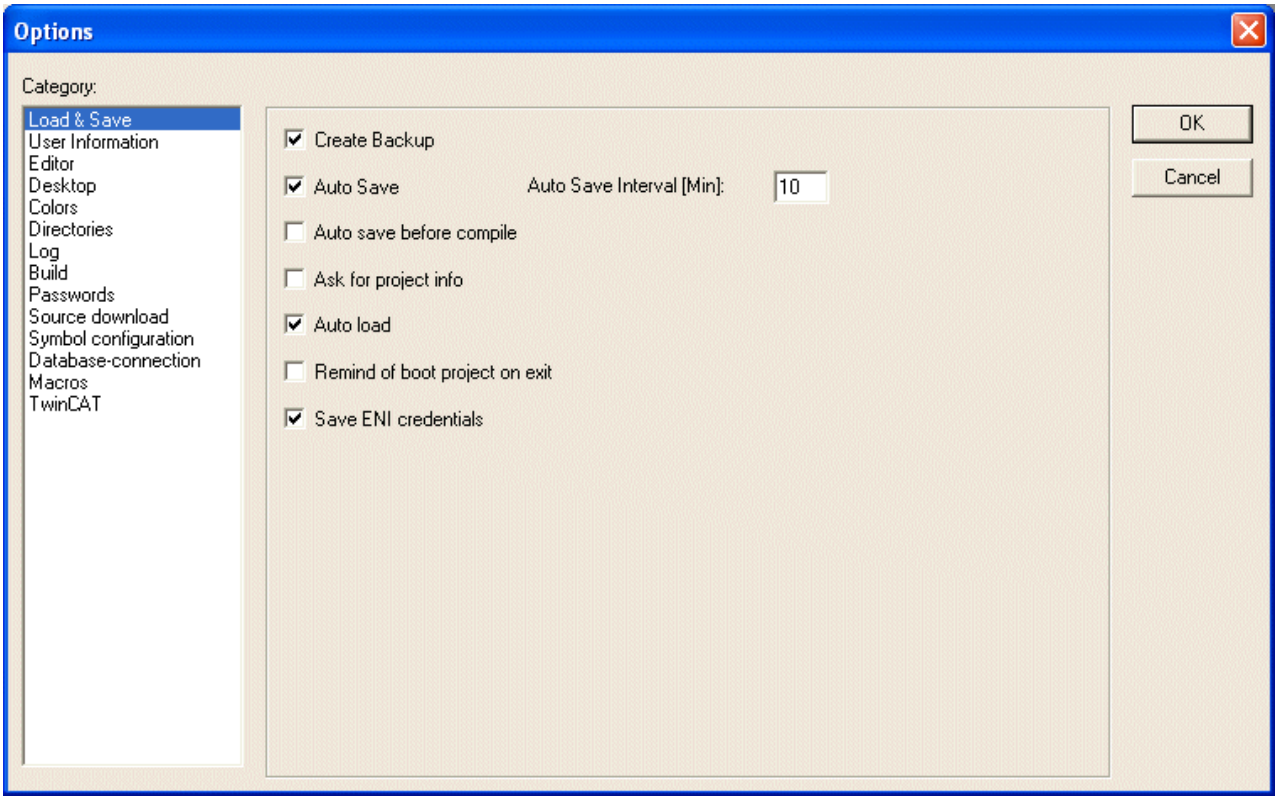
If the runtime system is a bus controller, then the TwinCAT category will be replaced by these options:

- Controller Settings [▶ 65]

- Controller Advanced

Load & Save

If you choose this category, then you get the following dialog box:



Option dialog box of the category Load & Save

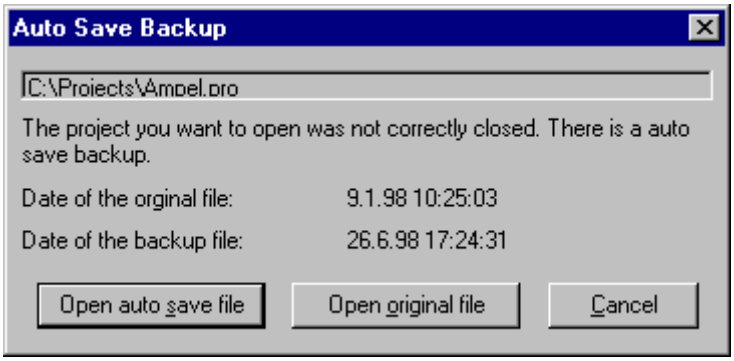
Create Backup

When activating an option, a check appears before the option. If you choose the option Create Backup, then TwinCAT PLC Control creates a backup file at every save with the extension ".bak". In this way you can always restore the versions before the last save.

Auto Save

If you choose the option Auto Save, then while you work your project is constantly saved to a temporary file with the extension ".asd" according to a set time interval **(Auto Save Interval)**. This file is erased at a normal exit from the program. If for any reason TwinCAT PLC Control is not shut down "normally" (e.g. due to a power failure), then the file is not erased.

When you open the file again the following message appears:



You can now decide whether you want to open the original file or the auto save file.

Auto Save before compile

The project is saved before each compilation.

Ask for project info

If you request the option Ask for project info, then when saving a new project, or saving a project under a new name, the project info is automatically called. You can visualize the project info with the command "Project" "Project info" and also process it.

Auto load

If you choose the option Auto Load, then at the next start of TwinCAT PLC Control the last open project is automatically loaded. The loading of a project at the start of TwinCAT PLC Control can also take place by entering the project in the command line.

Remind of boot project on exit:

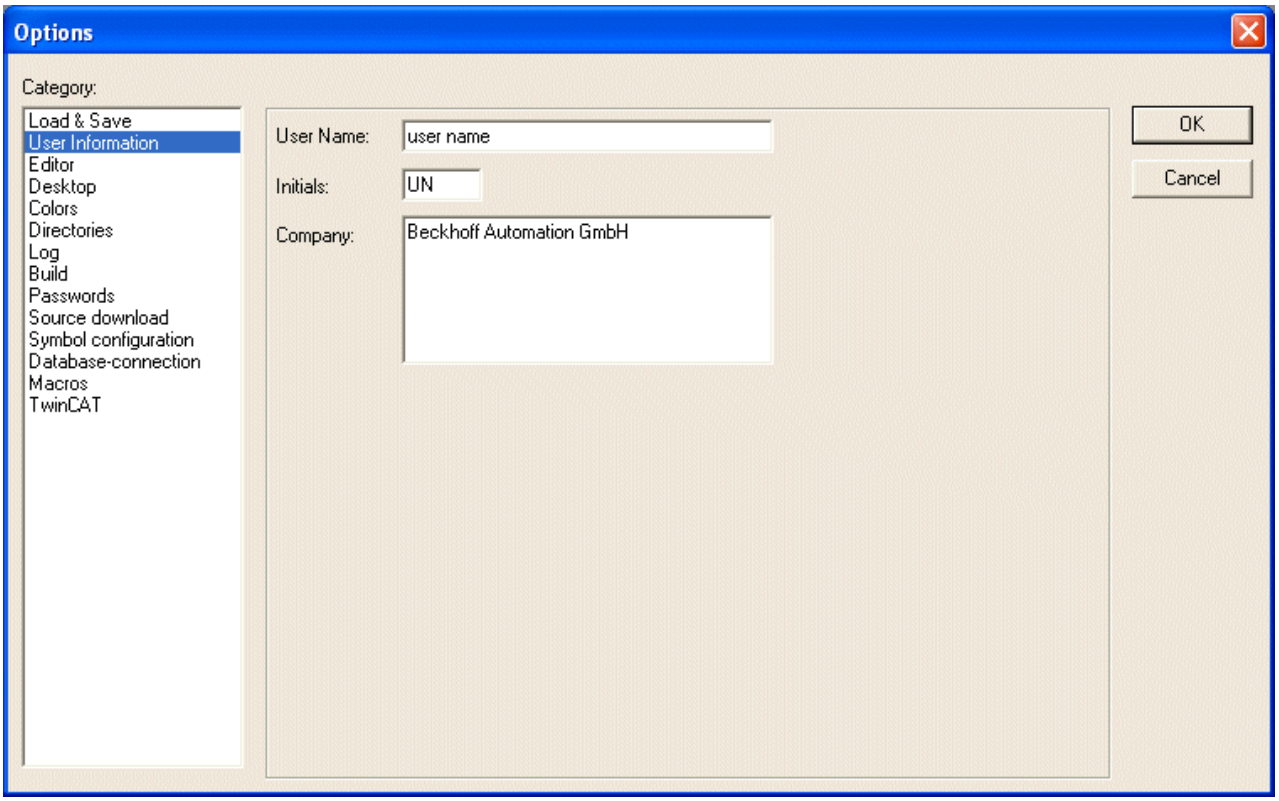
If the project has been modified and downloaded without creating a new boot project since the last download of a boot project, then a dialog will advise the user before leaving the project: "No boot project created since last download. Exit anyway?"

Save ENI credentials:

User name and Password, as they might be inserted in the Login dialog for the ENI data base, will be saved with the project.

User Information

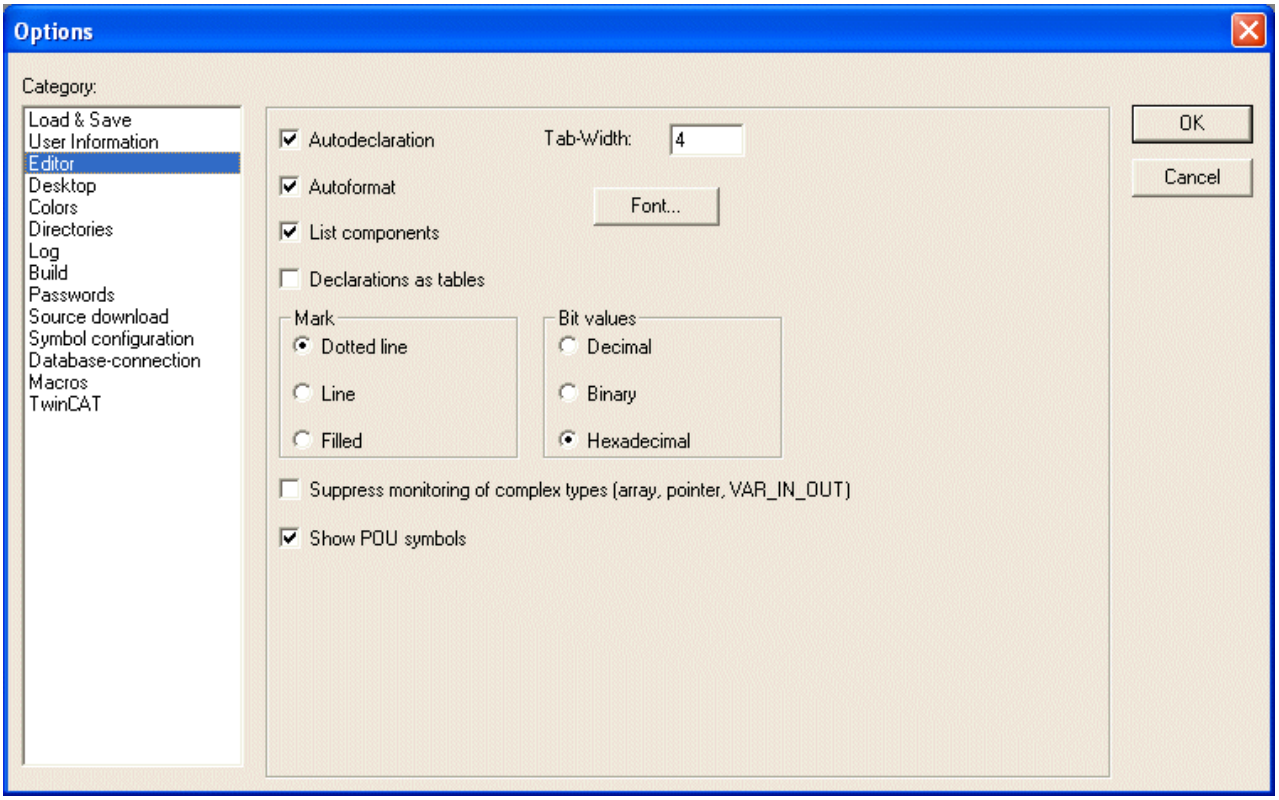
If you choose this category , then you get the following dialog box:



To User information belong the Name of the user, his Initials and the Company for which he works. Each of the entries can be modified and is saved with the project.

Editor

If you choose this category, then you get the following dialog box:



When activating an option, a check (ü) appears before the option. You can make the following settings for the Editors:

- Autodeclaration
- Autoformat
- List components
- Declaration as tables
- Tab width
- Font
- Display of the text selection (Mark)
- Display of the Bitvalues
- Suppress monitoring of complex types (array, pointer, VAR_IN_OUT)
- Show POU symbols

Autodeclaration

If you have chosen the Autodeclaration, then (following the input of a not-yet-declared variable) a dialog box will appear in all editors with which this variable can be declared.

Autoformat

If the option Autoformat in the category Editor of the options dialog box has been chosen, then TwinCAT PLC Control executes automatic formatting in the IL editor and in the declaration editor.

When you finish with a line, the following formatting is made:

- Operators written in small letters are shown in capitals;
- Tabs are inserted to that the columns are uniformly divided.

List components:

If this option is activated, then the **Intellisense** functionality will be available. This means that if you insert a dot at a position where an identifier should be inserted, then a selection list will open, offering all global variables which are found in the project. The Intellisense function is available in editors, in the Watch- and Receiptmanager and in the Sampling Trace.

Declarations as tables

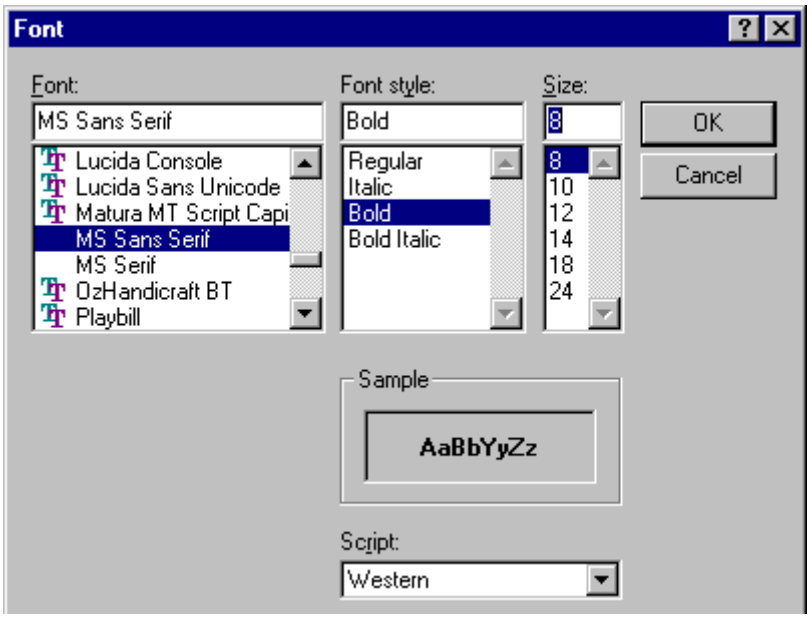
If the option Declarations as tables in the Editor category in the Options dialog box is selected, then you can edit variables in a table instead of using the usual declaration editor. This table is arranged as a card-index box in which there are register cards for input, output, local, and input/output variables. For each variable you have available the fields Name, Address, Type, Initial, and Comment.

In the field **Tab-Width** in the category Editor of the Options dialog box you can determine the width of a tab as shown in the editors. The default setting is four characters, whereby the character width depends upon the font which is chosen.

Font

By clicking on the button Font in the category Editor of the Options dialog box you can choose the font in all TwinCAT PLC Control editors. The font size is the basic unit for all drawing operations. The choice of a larger font size thus enlarges the printout, even with each editor of TwinCAT PLC Control.

After you have entered the command, the font dialog box opens for choosing the font, style and font size:



Dialog box for setting the font

Mark:

You can choose whether the current selection in your graphic editors should be represented by a dotted rectangle (Dotted), a rectangle with continuous lines (Line) or by a filled-in rectangle (Filled). The selection is activated in front of which a point appears.

Bitvalues:

You can choose whether binary data (type BYTE, WORD, DWORD) during monitoring should be shown Decimal, Hexadecimal, or Binary. The selection is activated in front of which a point appears.

Suppress monitoring of complex types (array,pPointer, VAR_IN_OUT):

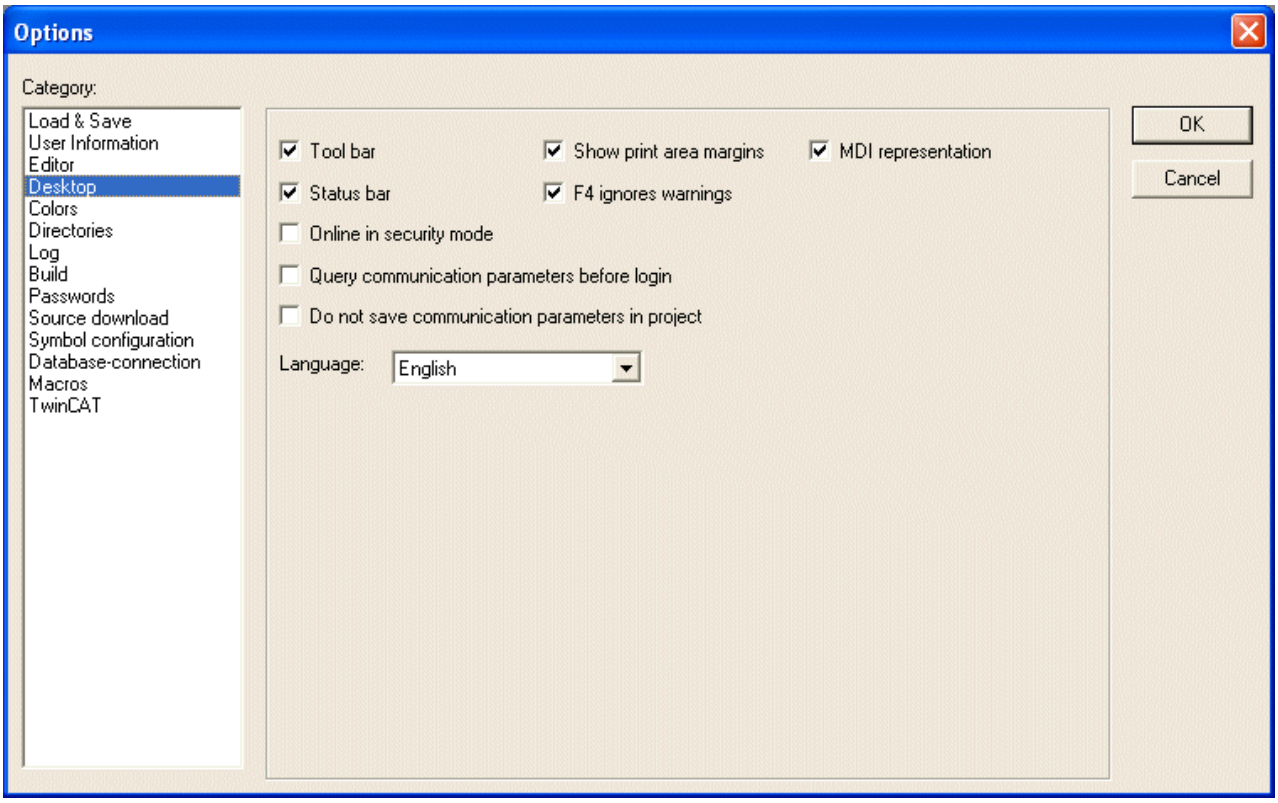
If this option is activated, then complex data types like arrays, pointer, VAR_IN_OUTs will not be displayed in the monitoring window in online mode.

Show POU symbols:

If this option is activated, then symbols will be displayed in POU boxes, provided that those are available as bitmaps in the library folder. The name of the bitmap file must be composed of the POU name and the extension ".bmp". Example: The symbol file for POU TON must be named TON.bmp:

Desktop

If you choose this category, then you get the following dialog box:



Options dialog box of the category Desktop

Tool bar

If the option tool bar has been chosen, then the tool bar with the buttons for faster selection of menu commands becomes visible underneath the menu bar.

Status bar

If the option status bar has been chosen, then the status bar at the lower edge of the TwinCAT PLC Control main window becomes visible.

Online in security mode

In Online mode with the commands 'Run', 'Stop', 'Reset' 'Toggle Breakpoint', 'Single cycle', 'Write values', 'Force values' and 'Release force', a dialog box appears with the confirmation request whether the command should really be executed.

Query communication parameters before login

As soon as the command 'Online' 'Login' is executed, first the communication parameters dialog will open. To get in online mode you must first close this dialog with OK.

Do not save communication parameters in project

The settings of the communication parameters dialog ('Online' 'Communication Parameters') will not be saved with the project.

Show print area margins

In every editor window, the limits of the currently set print range are marked with red dashed lines. Their size depends on the printer characteristics (paper size, orientation) and on the size of the “Content” field of the set print layout (menu: 'File” “Documentation Settings”).

F4 ignores warnings

After compilation, when F4 is pressed in a message window, the focus jumps only to lines with error messages; warning messages are ignored.

MDI representation

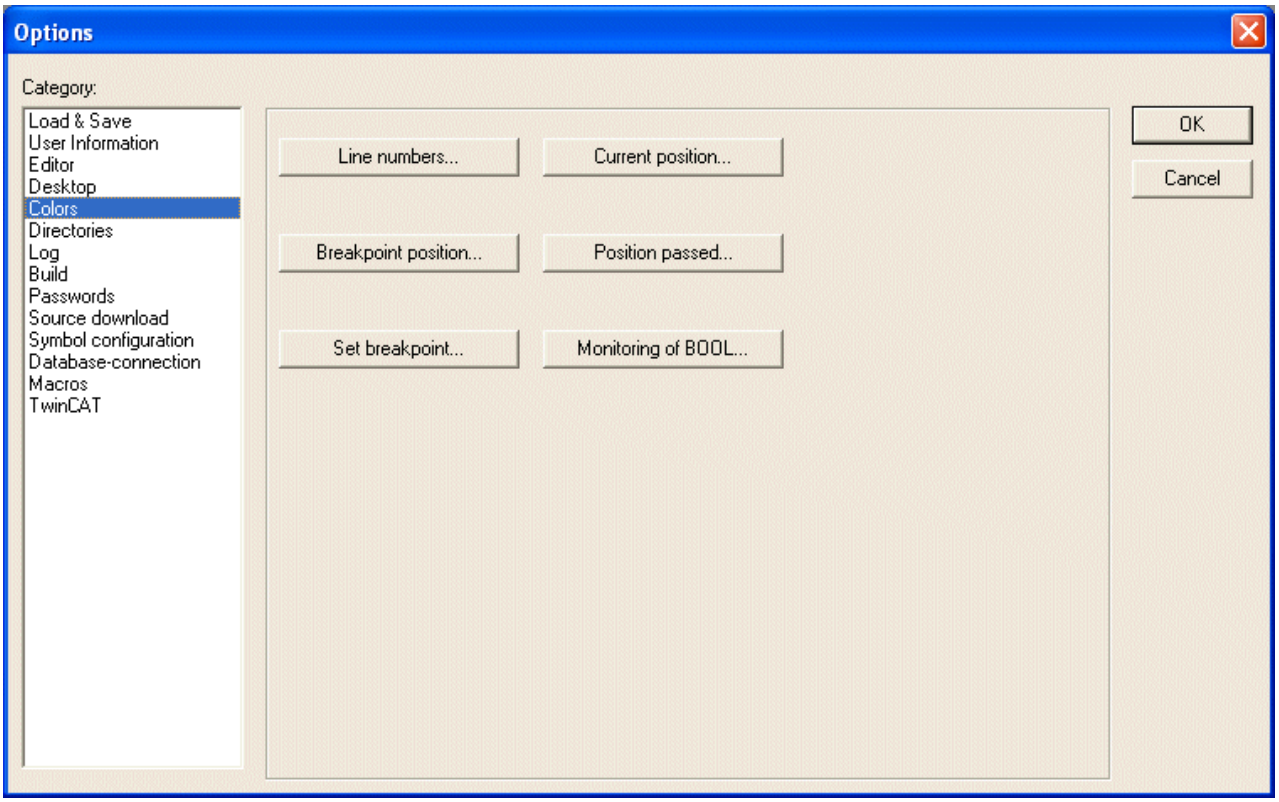
Per default this option (Multiple-Document-Interface) is activated and thus several windows can be opened at the same time. If the option is deactivated (SDI mode) only one window can be opened and will be displayed in full screen mode.
Exception: The action of a program and the program itself can be displayed side by side even in MDI mode.

Language

Define here, in which language the menu and dialog texts should be displayed.

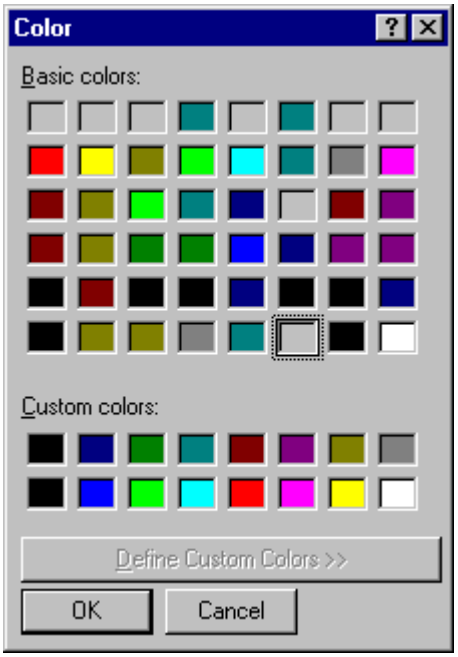
Colors

If you choose this category, then you get the following dialog box:



Options dialog box of the category Color

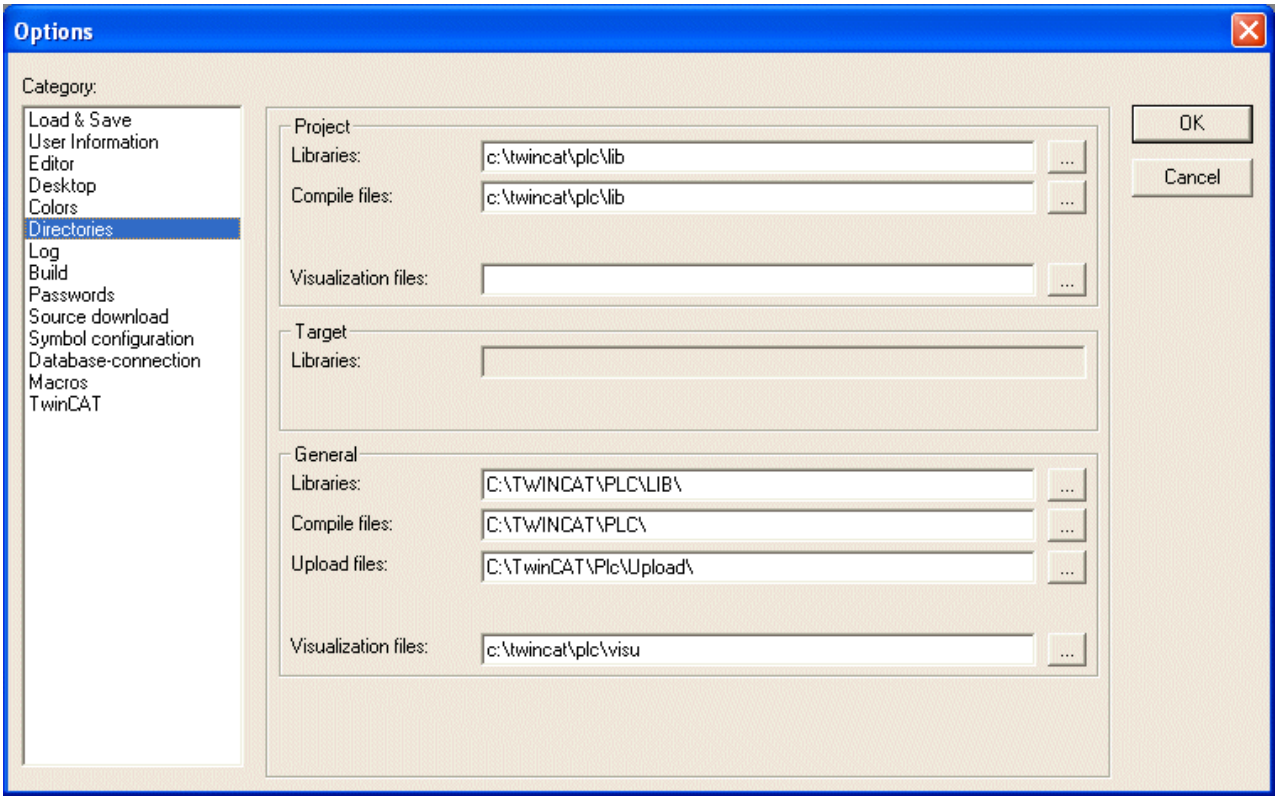
You can edit the default color setting of TwinCAT PLC Control. You can choose whether you want to change the color settings for Line numbers (default setting: light gray), for Breakpoint positions (dark gray), for a Set breakpoint (light blue), for the Current position (red), for the Position passed (green) or for the Monitoring of Boolean values (blue). If you have chosen one of the indicated buttons, the dialog box for the input of colors opens.



Dialog box for setting colors

Directories

If you choose this category, then you get the following dialog box:



Directories can be entered in the **Project** and **General** areas for TwinCAT PLC Control to use in searching for libraries and Visualization files, as well as for storing compile and source-upload files. If you activate the button (**...**) behind a field, the directory selection dialog opens. For library files, several paths can be entered for each, separated by semicolons ";".

Library paths can be entered relative to the current project directory, by using a prefixed ".".
If, for example, ".\libs" is entered, the libraries are searched in the directory C:
\Programs\projects\libs', even if the current project is stored in the directory 'C:\Programs\projects'

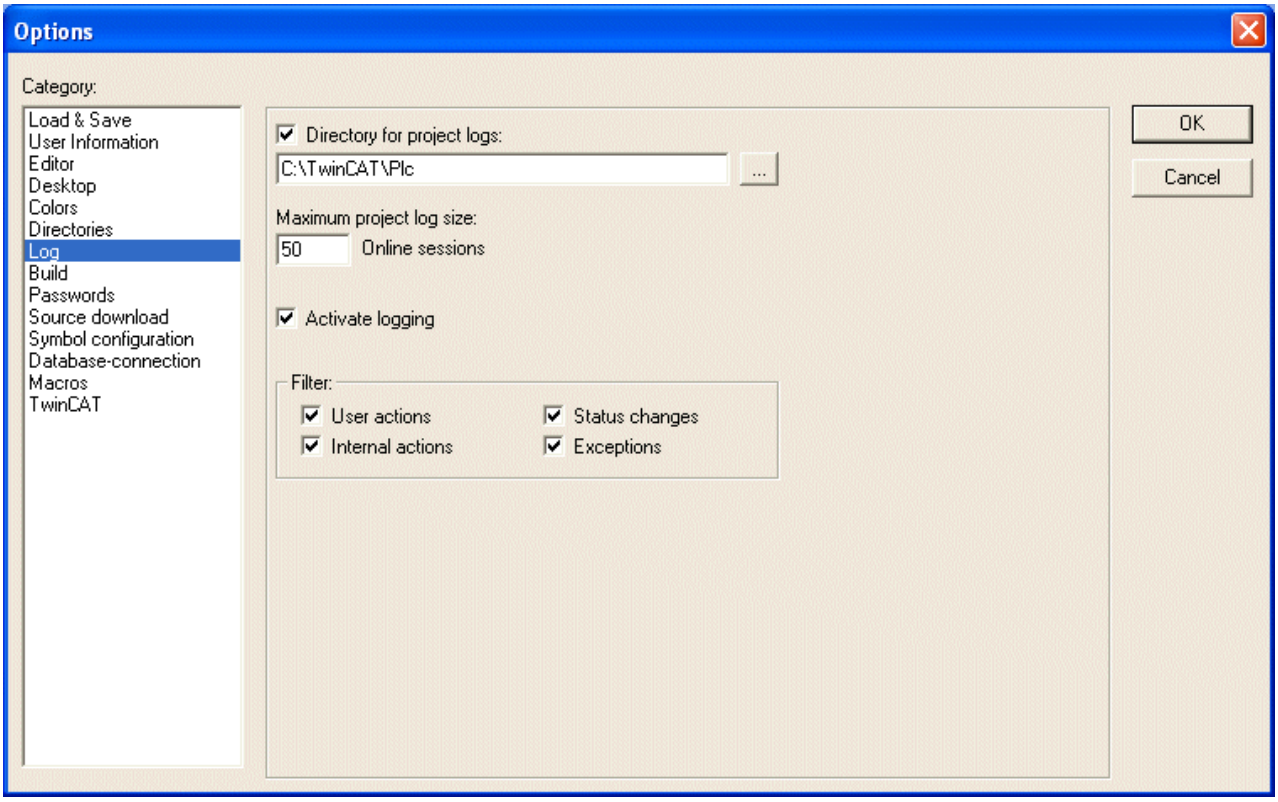
The information in the **Project** area is stored with the project; information in the **General** area is written to the ini file of the programming system and thus apply to all projects.

The **Target** area just displays the directories for libraries in the Target file. These fields cannot be edited, but an entry can be selected and copied (right mouse button context menu).

TwinCAT PLC Control generally searches first in the directories entered in **'Project'**, then in those in **'Target System'** (defined in the Target file), and finally those listed under **'General'**. If two files with the same name are found, the one in the directory that is searched first will be used.

Log

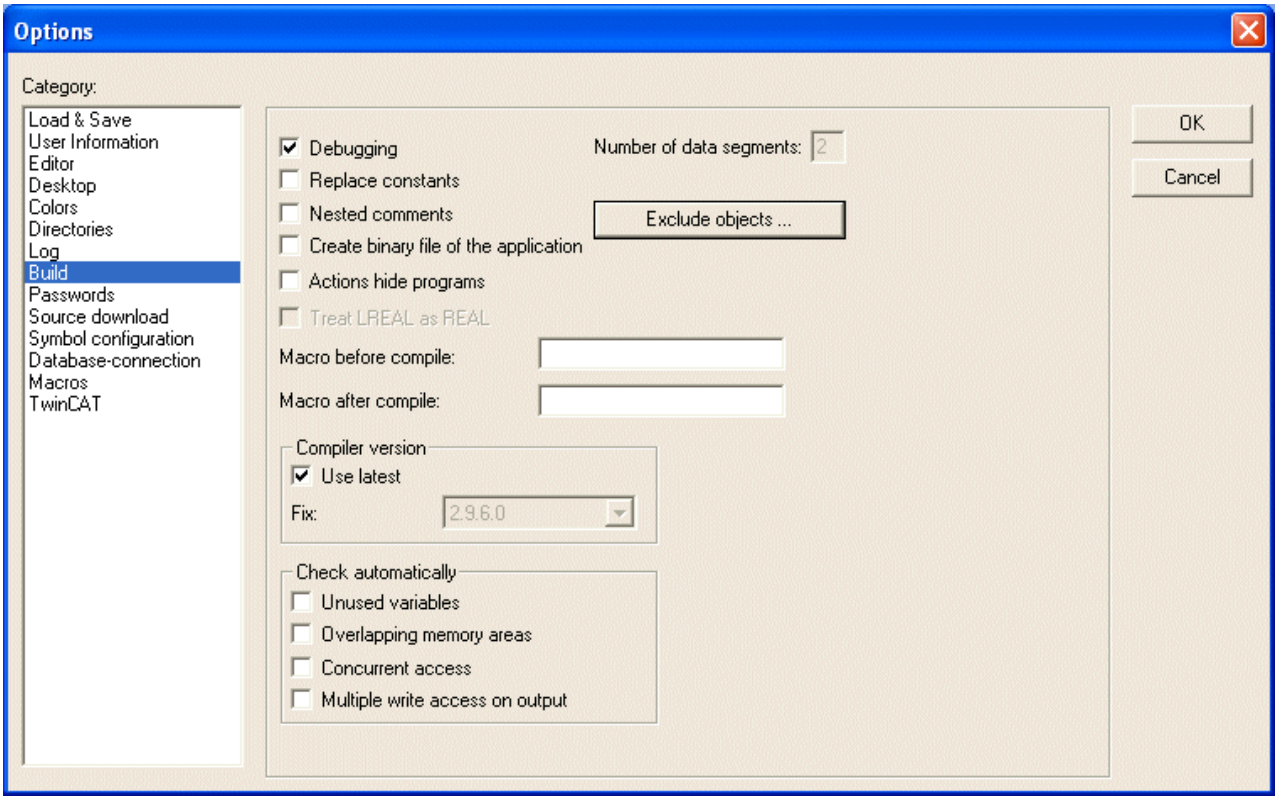
If you choose this category, then you get the following dialog box:



In this dialog, you can configure a file that acts as a project log, recording all user actions and internal processes during Online mode processing.
If an existing project is opened for which no log has yet been generated, a dialog box opens which calls attention to the fact that a log is now being set up that will receive its first input after the next login process.
The log is automatically stored as a binary file in the project directory when the project is saved.
If you prefer a different target directory, you can activate the option **Directory for project logs**: and enter the appropriate path in the edit field. Use the button to access the "**Select Directory**" dialog for this purpose.
The log file is automatically assigned the name of the project with the extension .log. The **maximumnumber** of **Online sessions** to be recorded is determined by **Maximum project log size**. If this number is exceeded while recording, the oldest entry is deleted to make room for the newest.
The Log function can be switched on or off in the Option field **Activate logging**. You can select in the **Filter** area which actions are to be recorded: **User actions, Internal actions**, **Status changes**, **Exceptions**. Only actions belonging to categories checked here will appear in the Log window and be written to the Log file.

Build

If you choose this category, then you get the following dialog box:



Options dialog box of the category Build

The Selecting and Deselecting of the options **debugging** and online changes has no effects in TwinCAT for the PC at the moment. These settings can be made under TwinCAT options! With TwinCAT for Bus Terminal Controller this setting causes that no more breakpoints are possible. In exchange more program memory is available.

When the **Replace constants** option is selected, the value of each constant is loaded directly, and in Online mode the constants are displayed in green. Forcing, writing and monitoring of a constant then is no longer possible. If the option is deactivated, the value is loaded into a storage location via variable access (this does in fact allow writing the variable value, but implies longer processing time).

If the **Nested comments option** is active, comments can be placed within other comments.

Example:

```
(*
a:=inst.out; (* to be checked *)
b:=b+1;
*)
```

Here, the comment that begins with the first bracket is not closed by the bracket following “checked,” but only by the last bracket.

Create binary file of the application

If the option Create binary file of the application is activated, a binary image of the generated code (boot project) is created in the project directory during compilation. File name: <project_name>.bin. By comparison, the command 'Online' 'Create Boot project' sets up the boot project on the controller. When an option is activated, a check appears next to the option.

Actions hide programs

This option is activated per default, when a new project is created. It means: If a local action has the same name like a global variable or a program, the following hierarchy is valid: local variable before local action before global variable before program.



If a project is opened that was created with an earlier TwinCAT PLC Control version, the option is disabled by default. Thus, the previous ranking (local variable before global variable before program before local action) valid at creation is maintained.

Treat LREAL as REAL:

If this option is enabled (default: not enabled), LREAL values will be treated as REAL values when compiling the project. This can be used to develop projects in a platform-independent way.

Data segments:

By specifying the number of data segments, you can define how many memory segments in the controller should be reserved for the data of your project. This place is necessary so that an online change can also be performed when new variables have been added. If during compilation you get the message "The global variables need too much memory, increase the number entered here. This setting is only relevant for the BCxxxx Bus Controller. In TwinCAT for the PC the number of data segments is constant. However, the size can be set under the TwinCAT options.

Exclude objects:

This button leads to the Exclude objects from build dialog:
Here, select the project function blocks in the displayed tree that are not to be regarded during a compilation run and activate the 'Do not build' option. The excluded function blocks are then displayed in green in the tree. To automatically exclude all function blocks that are not used in the program, press the 'Exclude unused' button.

Macro:

To influence the compiling process, two macros can be specified: the macro in the Macro before compile field is executed before the compilation, the macro in the Macro after compile field is executed after. However, the following macro commands cannot be executed here: file new, file open, file close, file saveas, file quit, online, project compile, project check, project build, debug, watchlist
All settings specified in the Build dialog are saved with the project.

Compiler version:

The compiler version to be used for the translation process can be defined here. In TwinCAT PLC Control versions after V2.9, both the current and the previous compiler versions (for each version / each service pack / each patch) going back to V2.9 are available. If you want a project to be always compiled with the latest compiler version, activate the 'Use latest' option. If you want the project to be automatically compiled with a specific version, set it via the selection field at 'Fix'.

Check automatically:

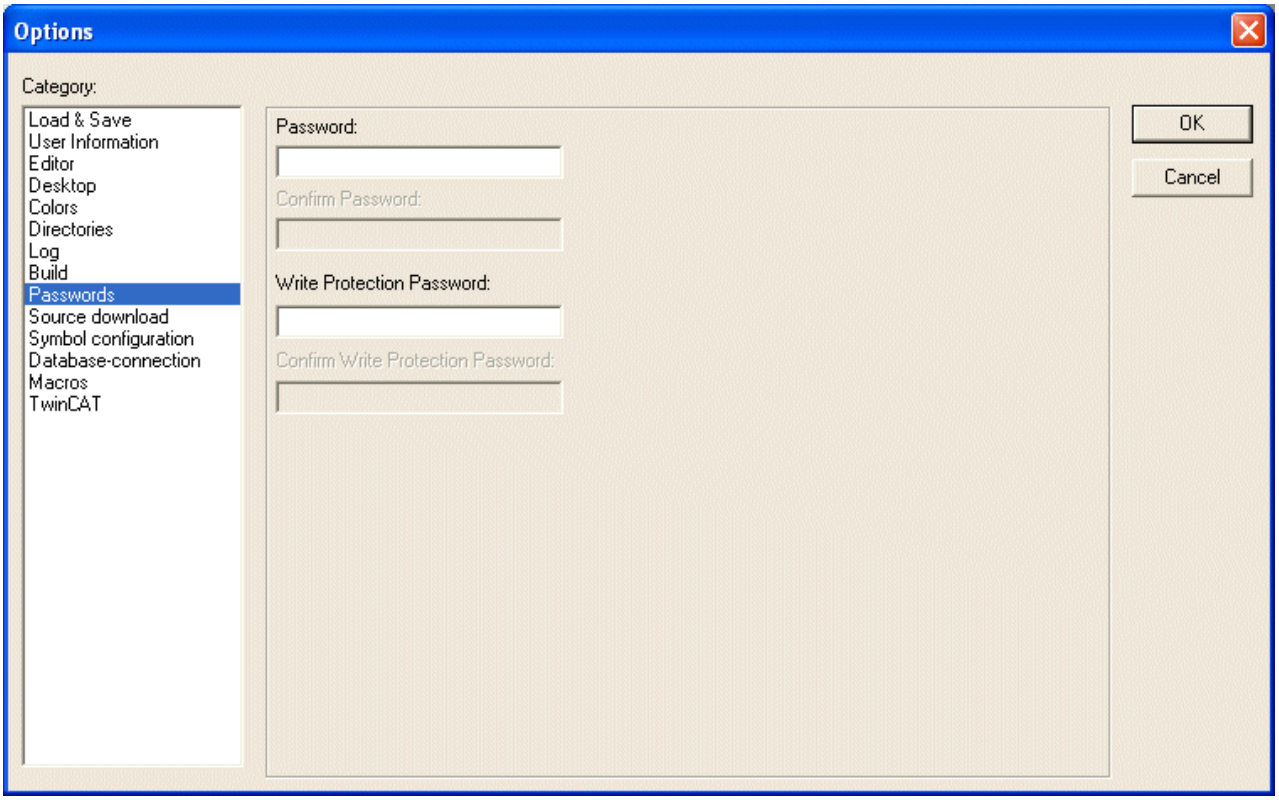
To check semantic correctness during each compilation of the project, the following options can be enabled:
- Unused variables
- Overlapping memory areas
- Concurrent access
- Multiple write access on output

The results are displayed in the message window. These checks can also be specifically triggered via the 'Check' command menu in the 'Project' menu

All entries in the Build Options dialog are stored with the project.

Passwords:

If you choose this category, then you get the following dialog box:



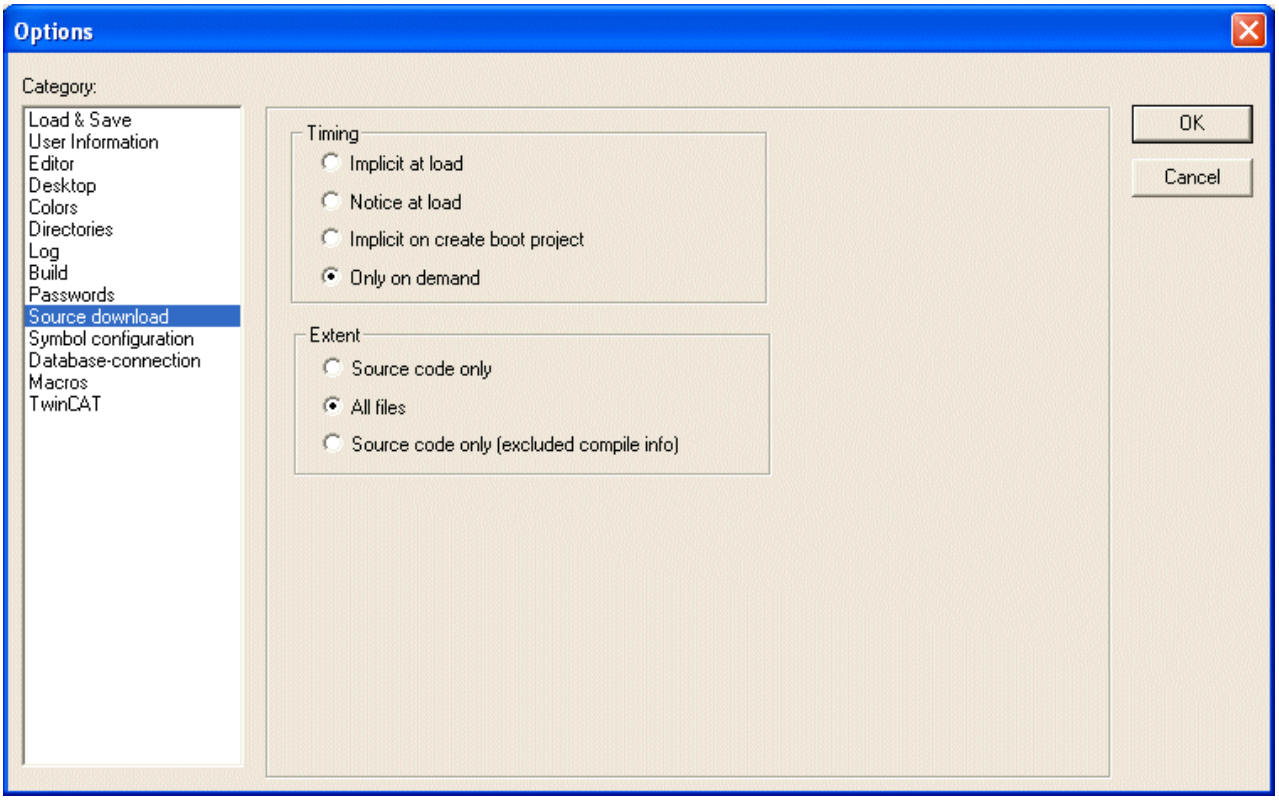
Options dialog box of the category Passwords

To protect your files from unauthorized access TwinCAT PLC Control offers the option of using a password to protect against your files being opened or changed.
Enter the desired password in the field **Password**. For each typed character an asterisk (*) appears in the field. You must repeat the same word in the field **Confirm Password**. Close the dialog box with OK. If you get the message:"The password does not agree with the confirmation", then you made a typing error during one of the two entries. In this case repeat both entries until the dialog box closes without a message. If you now save the file and then reopen it, then you get a dialog box in which you are requested to enter the password. The project can then only be opened if you enter the correct password. Otherwise TwinCAT PLC Control reports: "The password is not correct."

Along with the opening of the file, you can also use a password to protect against the file being changed. For this you must enter a password in the field **Write Protection Password** and confirm this entry in the field underneath. A write-protected project can be opened without a password. For this simply press the button Cancel, if TwinCAT PLC Control tells you to enter the write-protection password when opening a file. Now you can compile the project, load it into the PLC, simulate, etc., but you cannot change it.
Of course it is important that you memorize both passwords. The passwords are saved with the project. In order to create differentiated access rights you can define user groups ("Project" "Object access rights" and "Passwords for user groups").

Source download

If you choose this category, then you get the following dialog box:



You can choose to which **Timing** and what **Extent** the project is loaded into the controller system.

Extent:

- The option **Source code only** exclusively involves just the project file (extension .pro).
- The option **All files** also includes files such as the associated library files, visualization bitmaps, configuration files, etc.
- The option **Source code only** (excluded compile info) involves like above the pure Source code, but without Compile info. Thus this function is a pure source filling. An Online change after the source upload from the PLC is not possible.
  This option is interesting for the BX platform, because of the less data range.

Timing:

Using the option **Implicit at load** allows the selected file range to be automatically loaded into the controller system on the command ‘Online‘ ‘Download'.

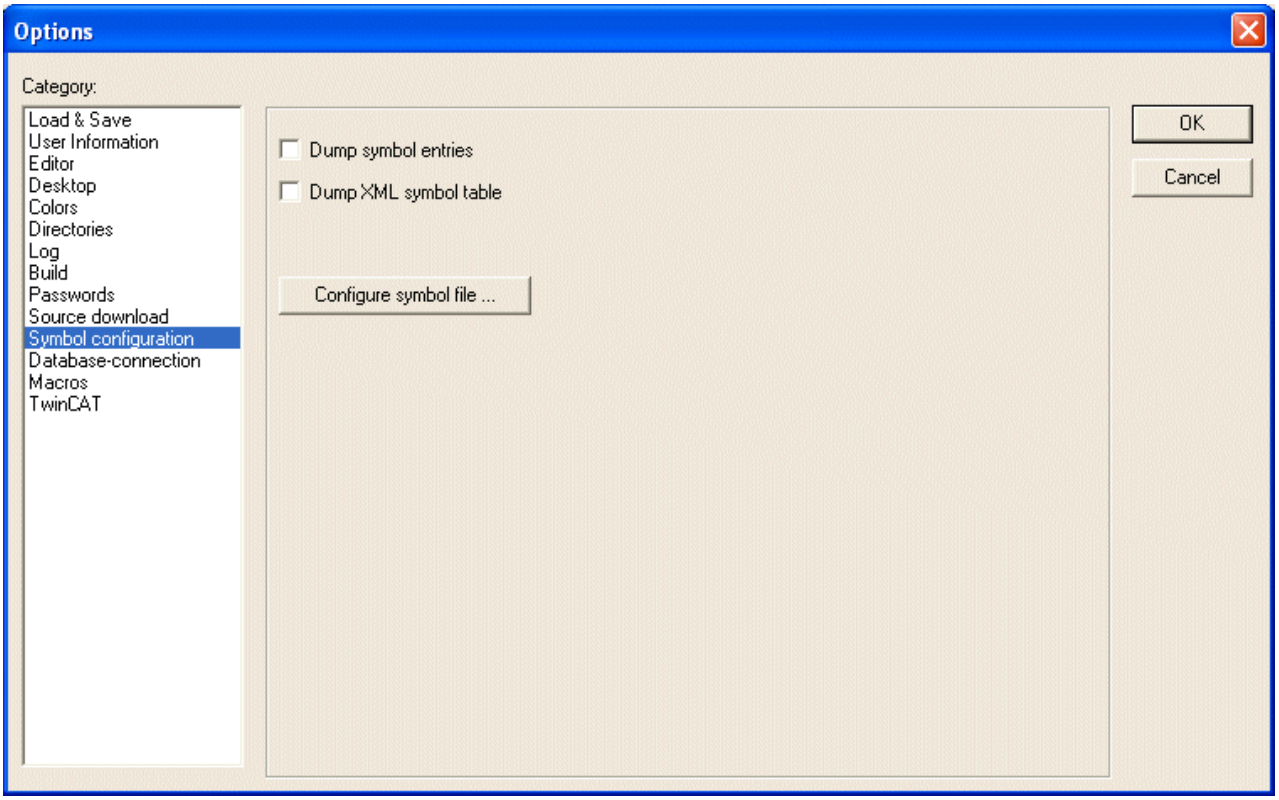
Using the option **Notice at load** offers a dialog, when the command ‘Online‘ ‘Download' is given, with the question ”Do you want to write the source code into the controller system?”. Pressing ‘Yes’ will automatically load the selected range of files into the controller system, or you can alternatively finish with ‘No’.

Using the option **Implicit on create boot project** allows the selected file range to be automatically loaded into the controller system on the command 'Online' 'Create boot project'.

When using the option **Only on demand** the selected range of files must be expressly loaded into the controller system by giving the command ‘Online‘ ‘Sourcecode download'.
The project which is stored in the controller system can be retrieved by using ‘File‘ ‘Open' with Open project from PLC. The files will be unpacked in the process. See Chapter 'File' 'Open' for details!

Symbol configuration

This form of symbol management exists only because of compatibility reasons und should only be used for the TwinCAT OPC Server. For all other cases exists a (at each compile) new created XML file with the name <projectname>.tpy

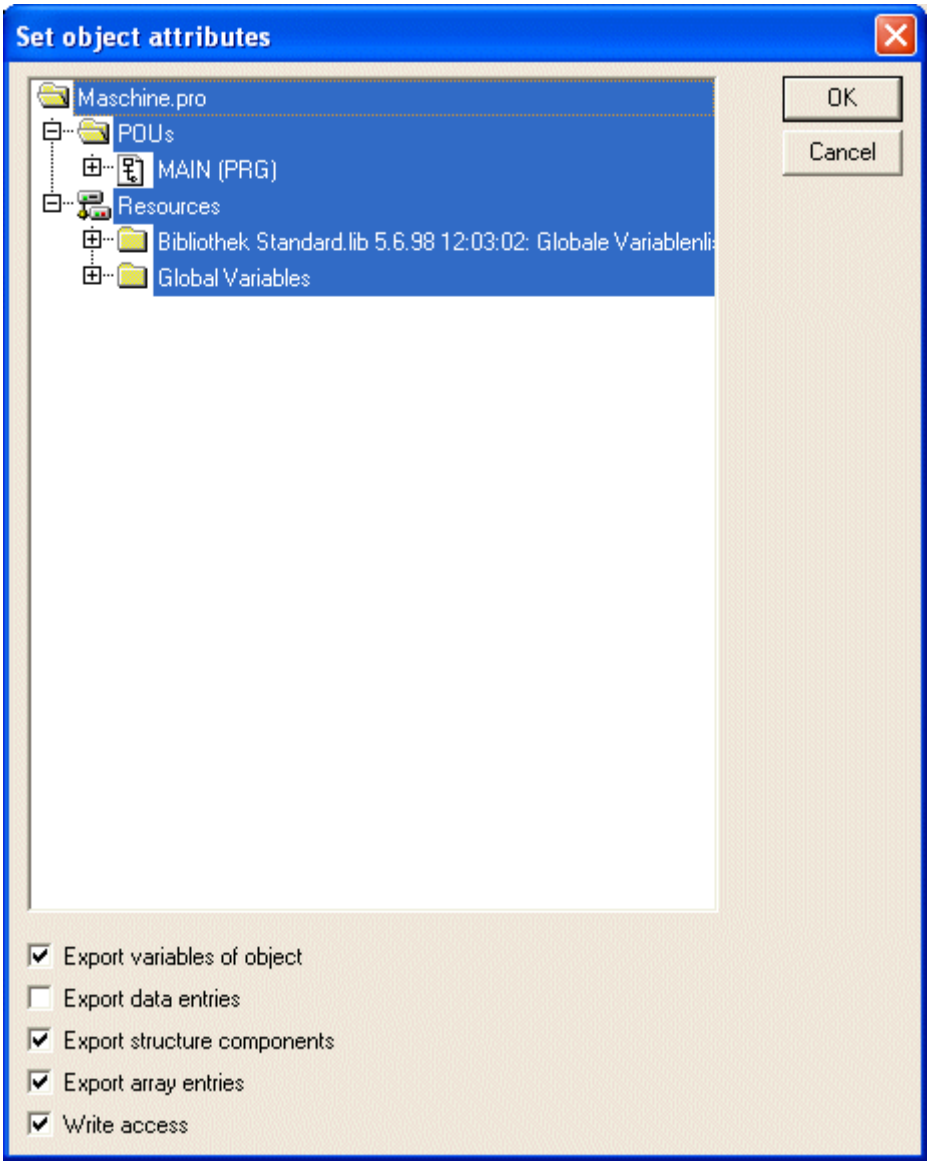


The dialog presented here is used for configuring the symbol file. This will be created as a text file <project name>.sym resp. a binary file <project name>.sdb in the project directory.

If the option **Dump symbol entries** is selected, then symbol entries for the project variables will be automatically created in a symbol file the file at each compilation of the project.

If additionally the option **Dump XML symbol table** is activated, then also an XML file containing the symbol information will be created in the project directory. It will be named <project name>.SYM_XML.

If you chose **Configure symbol file**, the following dialog opens:



Use the tree-structured selection editor to mark the variables which should be entered in the symbol file. For this purpose you can select a POUs entry (e.g. Global Variables) which automatically will mark all variables belonging to this POU, or you can select single variables which you find listed for each POU in the tree. Then set the desired options in the lower part of the dialog box by clicking the mouse on the corresponding small boxes. Activated options are checked. The following options can be set:
Export variables of object: The variables of the selected object are exported in the symbol file.

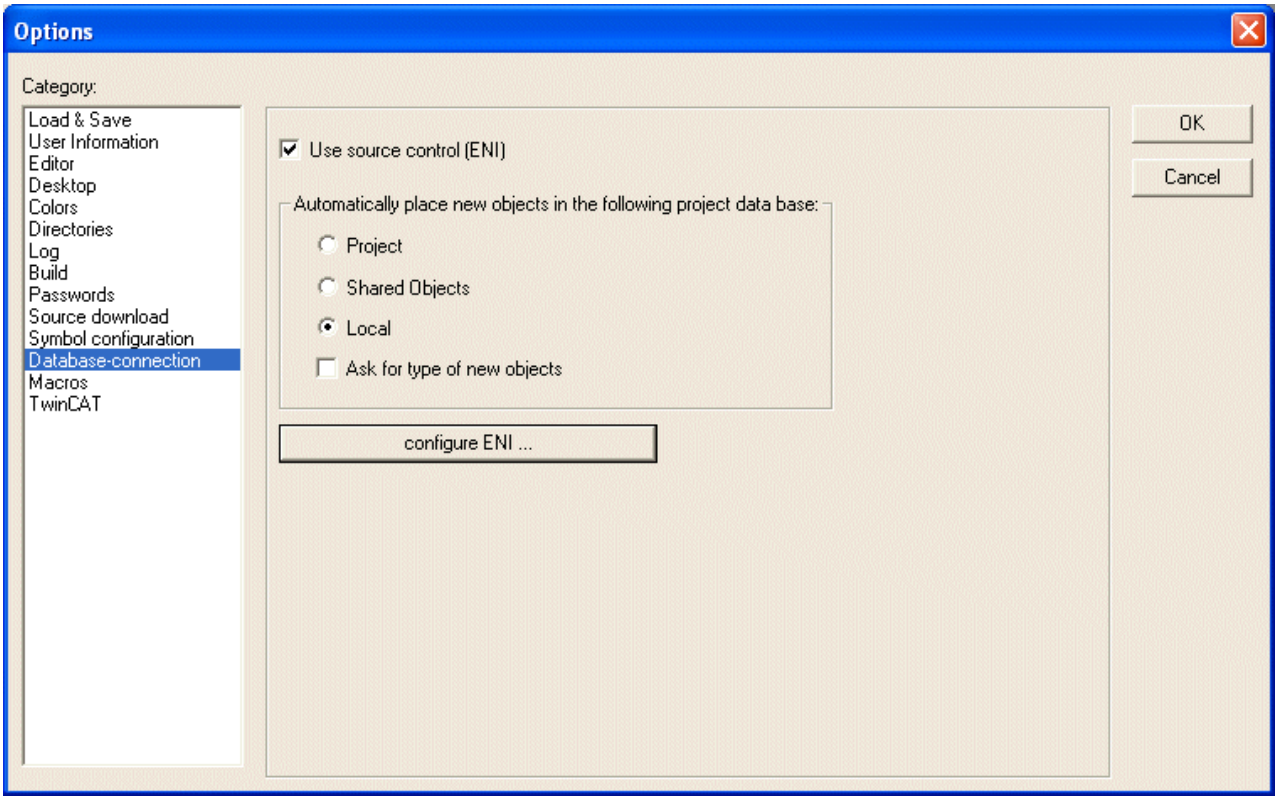
The following options can take effect only if the **Export variables of object** option is activated:

Export data entries: Entries for access to the global variables are created for object's structures and arrays.
Export structure components: An individual entry is created for each variable component of object's structures.
Export array entries: An individual entry is created for each variable component of object's arrays.
Write Access: Object's variables may be changed by the OPC server.

Once the option settings for the currently selected POU are complete, other POUs can be also be selected and can be given an option configuration. This can be carried out for any desired number of POU selections, one after the other. When the dialog box is closed by selecting **OK**, all configurations carried out since the dialog box was opened are applied.

Database Connection

This dialog is used to define whether the project should be managed in a project data base and to configure the ENI interface correspondingly.



Use source control (ENI): Activate this option, if you want to access a project data base via the ENI Server in order to administer all or a selection of POUs of the project in this data base.

Preconditions: ENI Server and data base must be installed and you must be registered as an user in the database.

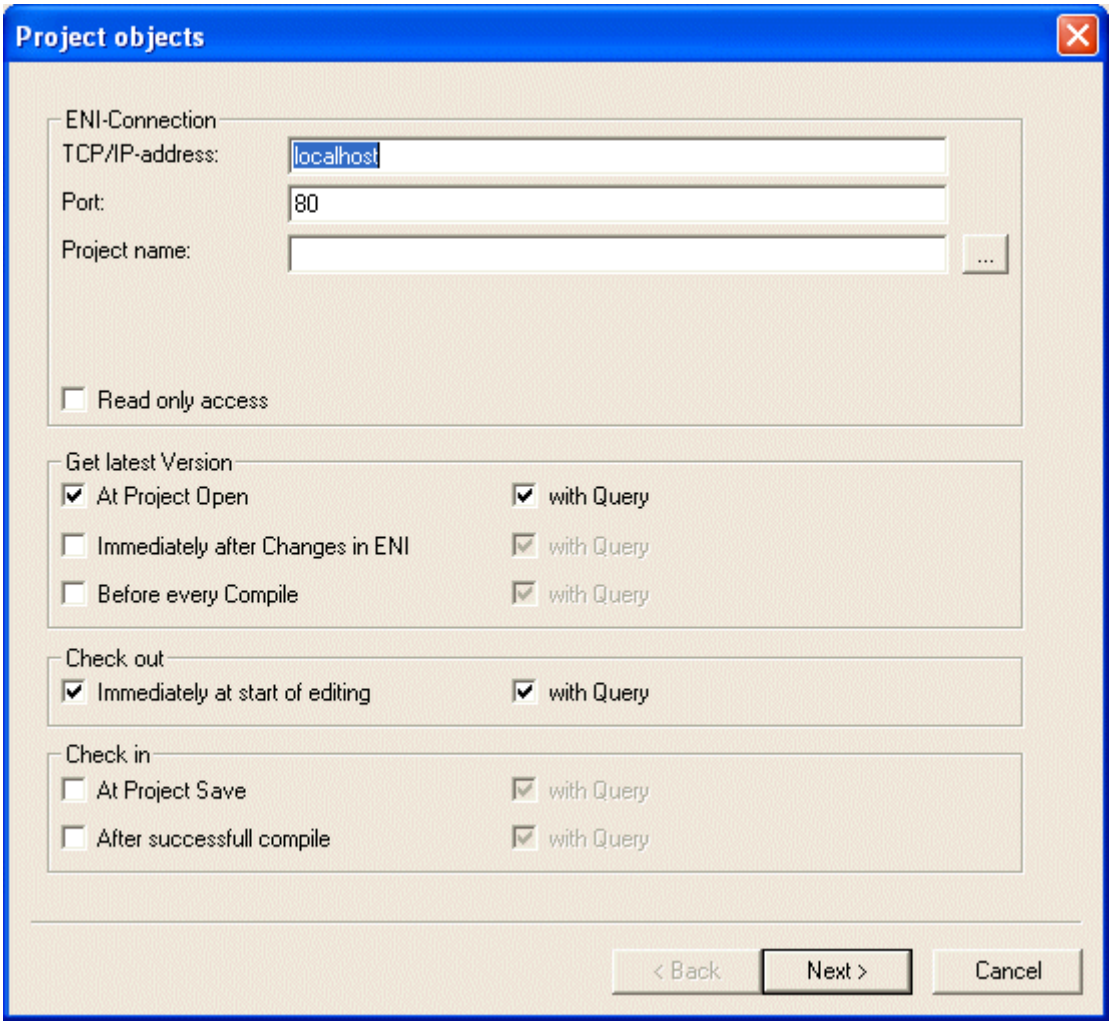
If the option is activated, then the data base functions (Check in, Get last version etc.) will be available for handling the project POUs. Then some of the data base functions will run automatically like defined in the options dialogs, and in the menu 'Project' 'Data Base Link' you will find the commands for calling the functions explicitly. Besides that a tab 'Data base-connection' will be added in the dialog Properties, where you can assign a POU to a particular data base category.

Connect new Objects with the following data base:

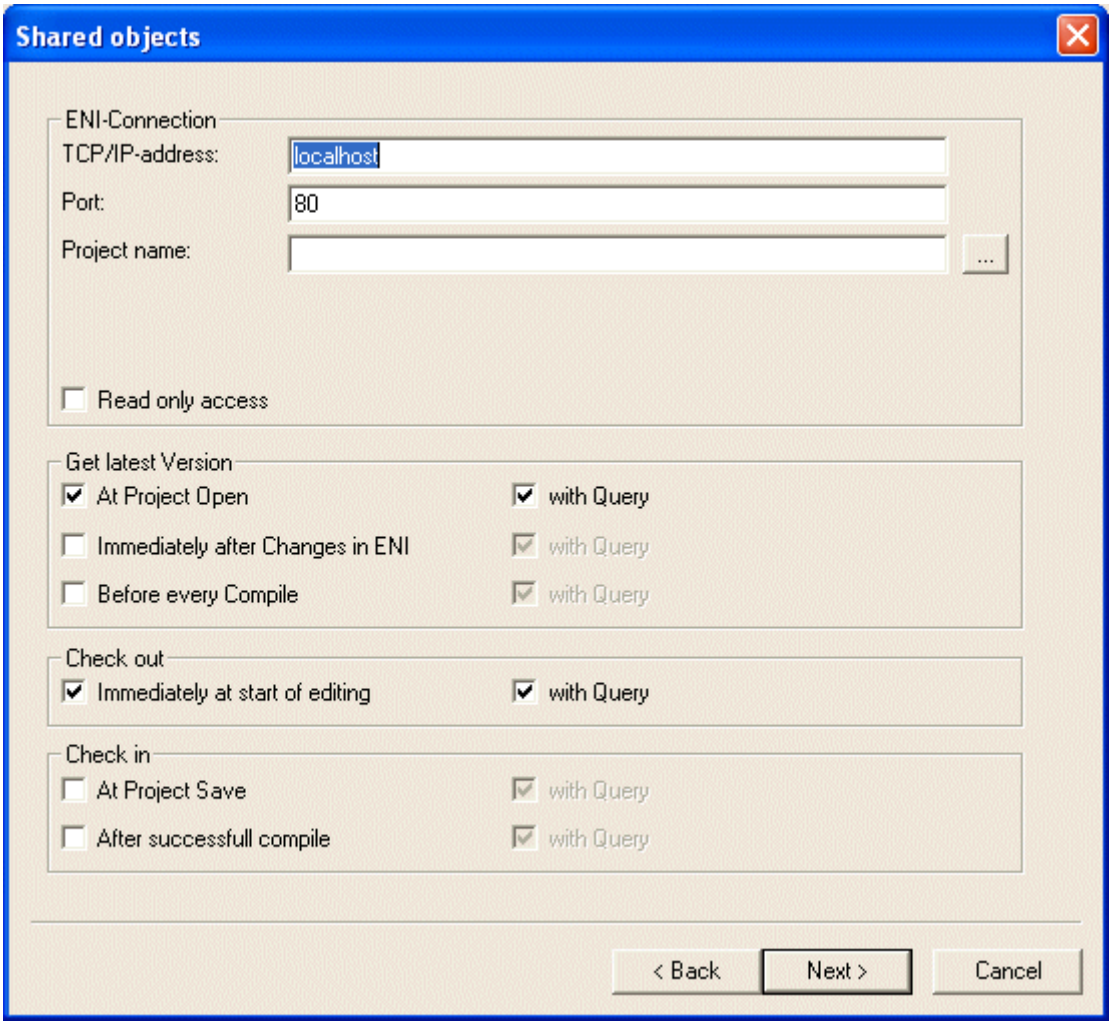
Here you set a default: If a new object is inserted in the project ('Project' 'Object' 'Add'), then it will automatically get assigned to that object category which is defined here. This assignment will be displayed in the object properties dialog ('Project' 'Object' 'Properties') and can be modified there later. The possible assignments:

Project: The POU will be stored in that data base folder which is defined in the dialog ENI configuration/ Project objects in the field 'Project name'.
Shared Objects: The POU will be stored in that data base folder which is defined in the dialog ENI configuration/Shared objects in the field 'Project name'.
Local: The POU will not be managed in a ENI data base, but only will be stored locally in the project. Besides 'Project objects' and 'Shared objects' there is a third data base category 'Compile files' for such objects which are not created until the project has been compiled. Therefore this category is not relevant for the current settings.
Ask for type of new objects: If this option is activated, then whenever a new object is added to the project, the dialog 'Object' 'Properties' will open, where you can choose to which of the three object categories mentioned above the POU should be assigned. By doing so the standard setting can be overwritten.

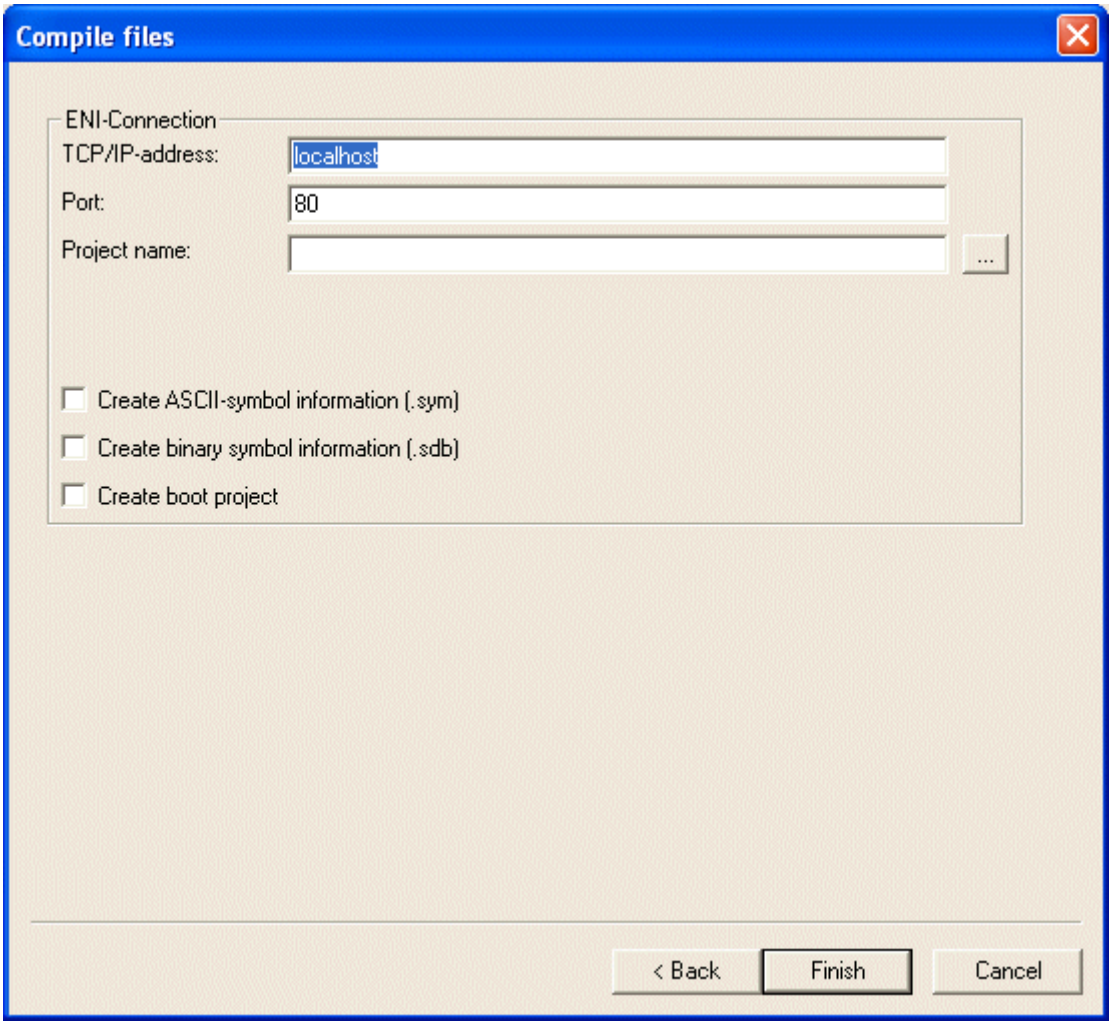
configure ENI: This button opens the first of three ENI configuration dialogs:
Each object of a project, which is determined to get managed in the ENI data base, can be assigned to one of the following data base categories: 'Project objects', 'Shared objects' or 'Compile files'. For each of these categories a separate dialog is available to define in which data base folder it should be stored and which presettings should be effective for certain data base functions:



Dialog ENI configuration / Project objects



Dialog ENI configuration / Shared objects



Dialog ENI configuration / Compile files

Each object will be stored also locally (with project) in any case.

The dialog will open one after the other if you are doing a primary configuration. In this case a **Wizard** (Button Next) will guide you and the settings entered in the first dialog will be automatically copied to the other ones, so that you just have to modify them if you need different parameter values.
If you want to modify an existing configuration, then the three dialogs are combined in one window (three tabs).
If you have not yet logged in successfully to the data base before, then the Login dialog will be opened automatically.

Options for project objects and shared objects regarding the project data base

These dialogs are part of the configuration of the project data base options ('Project' 'Options' 'Project source control'). Here you define the access parameters for the data base categories 'Project objects' und 'Shared objects'. All dialogs contain the same input items.

ENI-Connection

TCP/IP-Address: Address of the computer where the ENI-Server is running

Port: Default: 80; must be the same as set in the configuration parameters of the ENI Server

Project name: Name of the data base folder where the objects of this category should be stored. Press button ... to open a folder tree of the already existing data base projects. If the desired folder already exists, you can select it in this tree and its name will be entered in the 'Project name' edit field. If you had not logged in to the ENI Server until you try to open the folder tree by button ..., then you will first get the Login dialog where you must enter '**User name**' and '**Password**' as defined in your ENI user account to get access to the three data base categories.

Read only If this option is activated, then only read access is possible to the above defined data base folder.

Options for project objects, shared objects:

Get latest Version:

The data base function 'Get latest Version' (Menu 'Project' 'Data Base Link') copies the latest version of POUs from the above defined data base folder to the currently opened project, whereby the local version of objects will be overwritten. This will be done automatically for all objects, for which the version found in the data base differs from that in the project, as soon as one of the set timing conditions will meet. Activate the desired time options by setting a check mark:

At Project Open: As soon as the project is opened in TwinCAT PLC Control.

Immediately after Changes in ENI: As soon as a newer version of the POU is checked in to the data base (e.g. by another user); then the POU will be updated in the current project immediately and an appropriate message will pop up.

Before any Compile: Before any compile process in TwinCAT PLC Control.

Check out:

The data base function 'Check out' means that the POU will be marked as 'in the works' and will be locked for other users until it will be de-blocked again by a 'Check in' or 'Undo check out' command.
If the option **Immediately at start of editing** is activated, then an object will be checked out automatically as soon as you start to edit it. If the object is currently already checked out by another user (indicated by a red cross before the object name in the TwinCAT PLC Control. object organizer), then a message will pop up.

Check in:

The data base function 'Check in' means, that a new version of the object will be created in the data base. The older versions will be kept anyway.
You can activate one or both of the following options to define the time of automatic Checking in:

At Project Save: as soon as the project is saved

After successfull compile: as soon as the project has been compiled without errors

For each of the options 'Get last version', 'Check out' and 'Check in' additionally the option **with Query** can be activated. In this case, before the corresponding action is carried out, a dialog opens where you still can decide to cancel the action or otherwise confirm it.

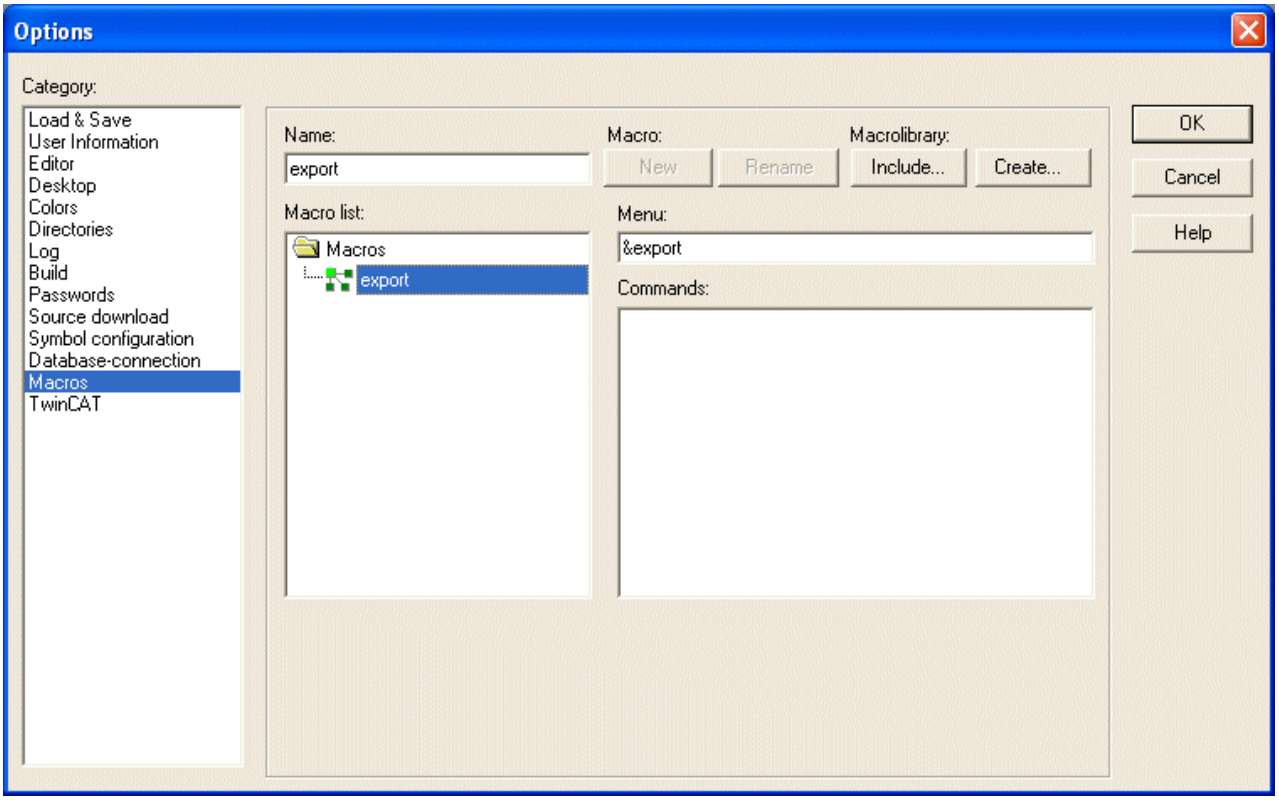
Only for dialog compile file:

Create ASCII-symbol information (.sym) **Create Binary-symbol information (.sdb)**	If this option is activated, then whenever a symbol file *.sym (text format) resp. *.sdb (binary format) will be created, this file will be written to the data base automatically. The entries in the symbol file are created like defined in the Project options category 'Symbol configuration'.
Create boot project	If this option is activated, then whenever a boot project will be created, this file will be written to the data base automatically .

Cancel will close the dialog without saving the done modifications in the currently opened dialog (the settings made in the previous dialogs will be kept anyway). You return to the main dialog 'Options' 'Project source control'.

Macros

If you choose this category, then you get the following dialog box:



In this dialog, macros can be defined using the commands of the batch mechanism, which can then be called up in the 'Edit' 'Macros' menu.
Perform the following steps to define a new macro:

1. In the input field **Name**, you enter a name for the macro to be created. After the **New** button is pressed, this name is transferred into the **Macrolist** field and marked as selected there. The macro list is represented in a tree structure. The locally defined macros are positioned one below the other. If macro libraries (see below) are integrated, then the library names will be listed and by a mouse-click on the plus- resp. minus-signs in front of those entries you can open or close a list of the library elements.
2. The **Menu** field is used to define the menu entry with which the macro will appear in the 'Edit' 'Macros' menu. In order to be able to use a single letter as a short-cut, the letter must be preceded by the symbol '&'. Example: the name "Ma&cro 1" generates the menu entry "Macro 1". Example: the name "Ma&cro 1" will create a menu item "Macro 1".

3.In the editor field **Commands** you define and/or edit the commands that are to constitute the newly created or selected macro. All the commands of the batch mechanism [▶ 334] and all keywords which are valid for those are allowed. You can obtain a list by pressing the Help button. A new command line is started by pressing <Ctrl><Enter>. The context menu with the common text editor functions is obtained by pressing the right mouse button. Command components that belong together can be grouped using quotation marks.

4. If you want to create further macros, perform steps 1-3 again, before you close the dialog by pressing the OK-button.

If you want to delete a macro, select it in the macro list and press button <Del>.

If you want to rename a macro, select it in the macro list, insert a new name in the edit field 'Name' and then press button **Rename**.

To **edit** an existing macro, select it in the macro list and edit the fields 'Menu' and/or 'Commands'. The modifications will be saved when pressing the OK-button.

As soon as the dialog is closed by pressing the **OK**-button the actual description of all macros will be saved in the project.

The macro menu entries in the 'Edit' 'Macros' menu are displayed in the order in which they were defined. The macros are not tested until a menu selection is made.

Macro libraries

Macros can be saved in external macro libraries. These libraries can be included in other projects.

· Creating a macro library containing the macros of the currently opened project:

Press button **Create**. You get the dialog '**Merge project**', where all available macros are listed. Select the desired entries and confirm with OK. The selection dialog will close and dialog '**Save Macrolibrary**' will open. Insert here a name and path for the new library and press button Save. The library will be created named as **<library name>.mac** and the dialog will be closed.

· Including a macro library <library name>.mac in the currently opened project:

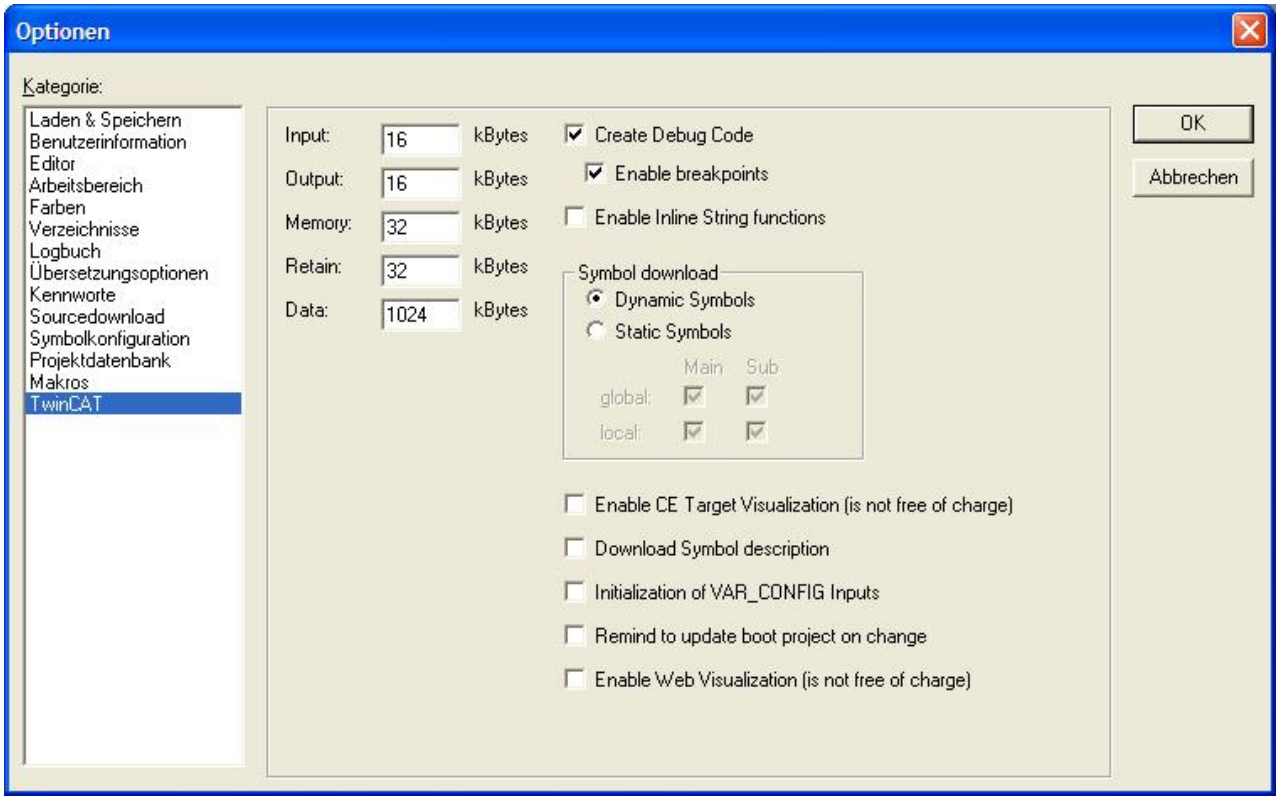
Press button **Include**. The dialog **'Open Macrolibrary'** will open, which shows files with extension *.mac. Select the desired library and press button Open. The dialog will be closed and the library will be added to the tree of the Macrolist.



The macros of a project also can be exported ('Project' 'Export').

TwinCAT

If you choose this category, then you get the following dialog box:



Options dialog box of the category TwinCAT

This menu can be used to set the size of the addressable memory for all input, output and flag variables. Any non-allocated variables are located outside this memory area. The checkbox on the right can be used to decide whether to allow break points or not. If variables from the PLC are to be read by name by a visualization system, all required variable names must be known in the runtime system of the PLC.

Enable Inline String functions:

String functions are not "thread safe": When using tasks, string functions may only be used in a single task. If the same function is used in different tasks, there is a danger of overwriting.

By selecting this functionality the behaviour is changed and the compiler dissolves these string functions directly in the PLC code.

Symbol download:

The use of dynamic symbols is generally recommended. With this type of symbol management, a request for a certain symbol causes the symbol to be assembled dynamically in the runtime system. The advantages are reduced memory requirement compared with static symbol management and shorter compile and download times. The number of handles is limited to 8192. Handles that are not required have to be released again.

If static symbol management is to be used for compatibility reasons, the amount of the variables stored in the runtime system of the PLC can be specified in this menu. Local and global management of variable names can be specified in the rows of a matrix. "Main" only generates structure or field names. "Sub" also generates the variable names for all structure or field components.

Sample:

abc : ARRAY[1..2] OF BOOL;

If you select only Main you get symbols only for 'abc'. If you select Sub you get also symbols for the components 'abc[1]' and 'abc[2]'.

Enable CE Target Visualization:
If this option is selected, the Target-Visualization is active. The library **SysLibTargetVisu.lib i**s included automatically in the library management.

Download symbol description:
If this option is selected, the symbol description is downloaded to the target system. The .tpy file describes the PLC project and is e.g. the input of TwinCAT System Manager or TwinCAT OPC. The file is stored in the boot directory and is named accordingly to the run time system CurrentPlc_1.tpy ... CurrentPlc_4.tpy.

Initialization of VAR_CONFIG inputs:
If this option is active, the input variables VAR_CONFIG are initialised with their initialization values. If the variables VAR_CONFIG are not linked, it is shown via the PLC diagnosis blocks.



The init value remains exist also if the VAR_CONFIG variables are not linked.

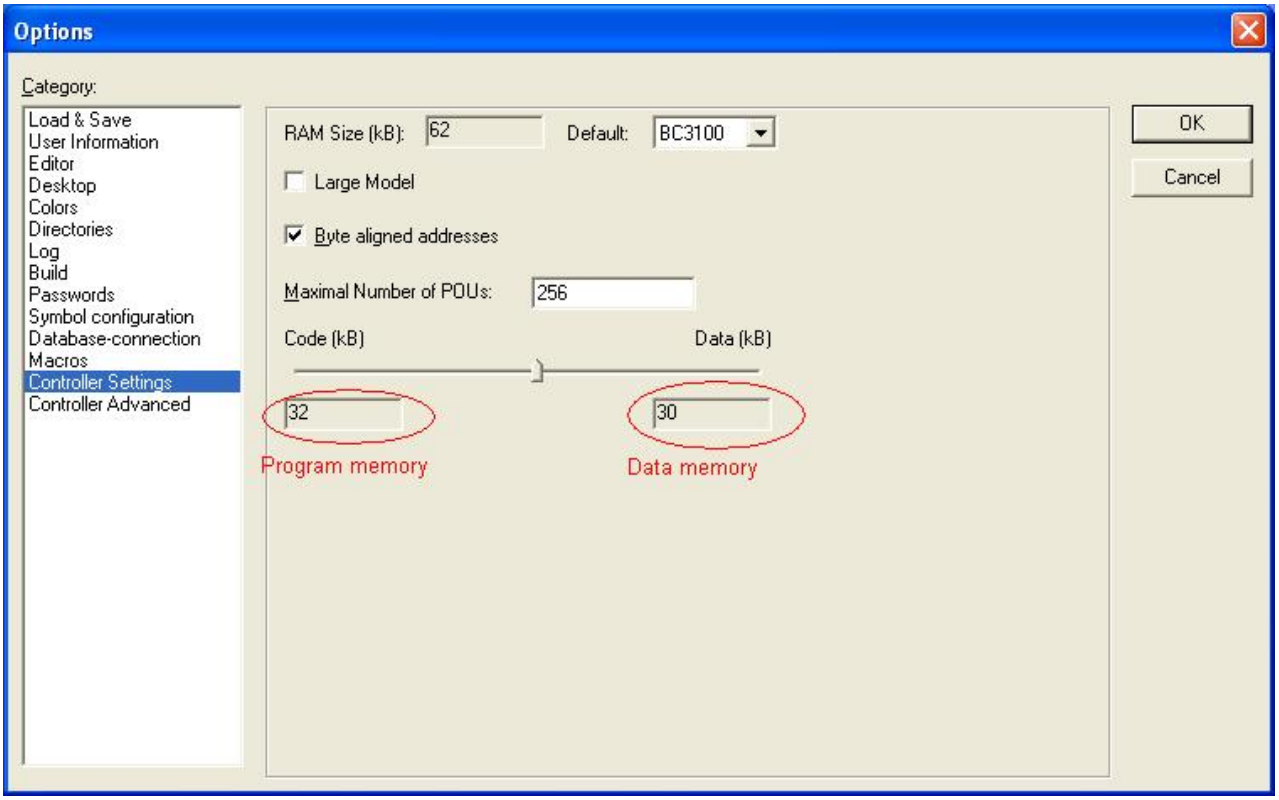
Remind to update boot project on change:
If this option is activated, you are asked at every download of your PLC project, whether a new boot project shall be produced directly.

Enable Web Visualization:
If this option is selected, the Web-Visualization is active.

Controller Settings

This category will only appear if the runtime system is a bus controller (BCxxxx BXxxxx).

If you choose this category, then you get the following dialog box:



The settings for the program memory and the data memory are adjustable here:

- The **data memory can be increased** by pushing the slider to the **left**. The adjusted data memory is shown in the right data field.
- The **program memory can be increased** by pushing the slider to the **right**. The adjusted program memory is shown in the left data field
- It is possible to increase the program or the data memory generally, by activating the option "Large Model"

4.3 Managing Projects

The commands which refer to entire project are found under the menu items "File" and "Project". Some of the commands under "Project" deal with objects and are therefore described in the chapter Objects [▶ 87].

Entries under the menu point File:

'File' 'New'

With this command you create an empty project with the name "Untitled". This name must be changed when saving

'File' 'New from pattern'

With this command you open any project that shuold be used as "pattern". A dialoge appears for selecting a project file, that can be openend with the name " Untitled".

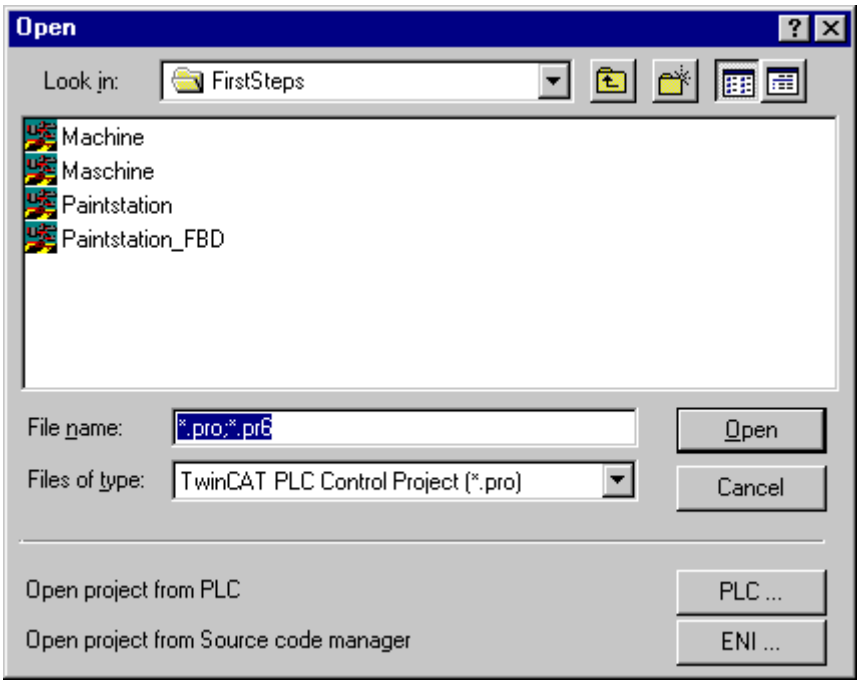
'File' 'Open'

With this command you open an already existing project. If a project has already been opened and changed, then **TwinCAT PLC Control** asks whether this project should be saved or not.
The dialog box for opening a file appears, and a project file with the extension "*.pro" or a library file with the extension "*.lib" must be chosen. This file must already exist. It is not possible to create a project with the command **"Open"**.
To upload a project file from the PLC, press **PLC** at **Open project from PLC**. You will obtain, as next, the

dialog Communication parameters (see menu ‘Online‘ ‘Communication parameters‘) for setting the transmission parameters when no connection exists yet to the PLC. Once an on-line connection has been created, the system checks whether the same named project files already exist in the directory on your computer hard disc. When this is the case you receive the dialogue **Load the project from the controller** where you can decide whether the local files should be replaced by those being used by the controller. (This sequence is the reverse of the sequence of 'Online' 'Load source code', with which the project source file is stored in the controller. Do not confuse with 'Create Boot project'!)

Note that after uploading a project, it is still without a name. You must save it under a new name.

If no project was loaded on the controller yet, you will receive a corresponding error message.



Standard dialog for opening a file in TwinCAT PLC Control

The option **Open project from Source code manager** is used to open a project that is managed in an ENI project database. The requirement is that you have access to an ENI server that serves the database. Click the ENI... button to get the 'Project Objects' dialog for establishing the connection to the server.

Enter the appropriate access data here (TCP/IP address, port, user name, password, read only access) and the directory of the database (project name) from which the objects are to be retrieved from the database and confirm with **Next.** Then the dialog closes and the corresponding one for the category '**Shared objects**' opens. Enter your access data here as well. Finish closes this dialog and automatically retrieves and opens the objects of the set directories. Now you can make the desired settings in the project options, which should apply to the further editing of the project. If you want to continue managing the project in the database, parameterize accordingly in the dialogs of the Project database category.

In the File menu, below the '**Exit**' menu item, the last opened projects are listed. When you select one of them, this project will be opened.

If passwords or user groups have been defined for the project, a dialog for entering the password appears.

'File' 'Close'

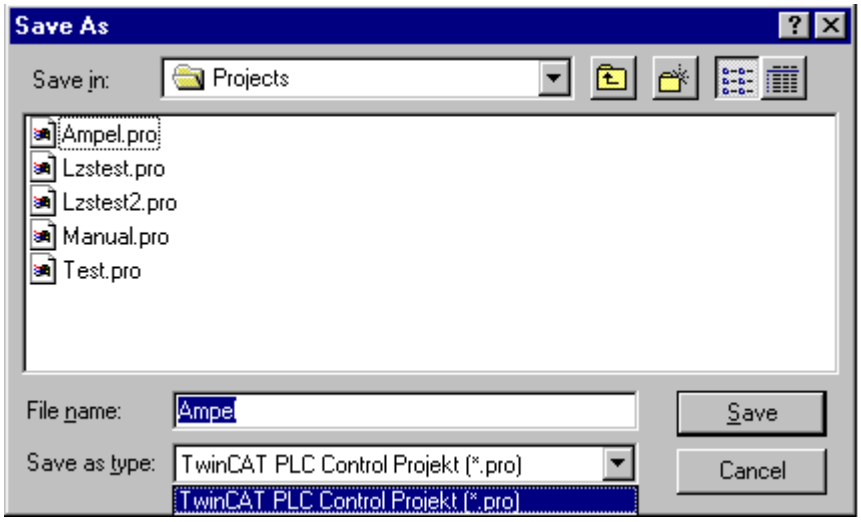
With this command you close the currently-open project. If the project has been changed, then TwinCAT PLC Control asks if these changes are to be saved or not. If the project to be saved carries the name "Untitled", then a name must be given to it (see "File" "Save as").

'File' 'Save' Shortcut: <Ctrl>+<S>

With this command you save any changes in the project. If the project to be saved is called "Untitled", then you must give it a name (see "File" "Save as").

'File' 'Save as'

With this command the current project can be saved in another file or as a library. This does not change the original project file. After the command has been chosen the Save dialog box appears. Choose either an existing File name or enter a new file name and choose the desired file type.

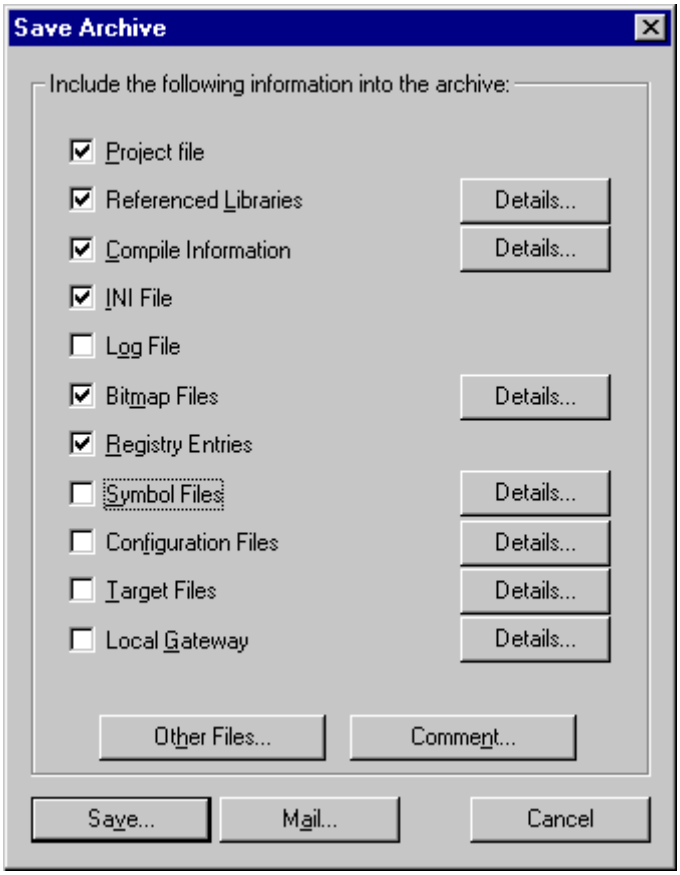


Dialog box for Save as

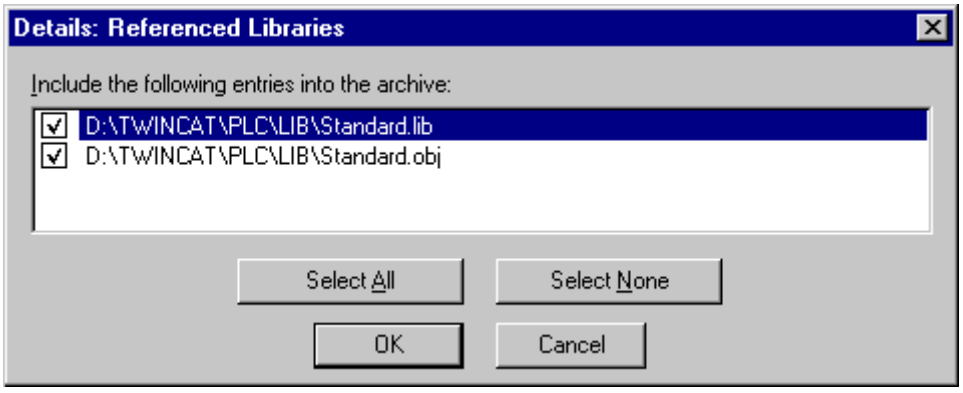
You can also save the current project as a library in order to use it in other projects. Choose the file type **Internal library (*.lib)** if you have programmed your POUs in TwinCAT PLC Control.
Choose the file type **External library (*.lib)** if you want to integrate the POUs which have been implemented in other languages (e.g. C). This means that another file is also saved which receives the file name of the library, but with the extension "*.h". This file is constructed as a C header file with the declarations of all POUs, data types, and global variables. Then click **OK.**
The current project is saved in the indicated file. If the new file name already exists, then you are asked if you want to overwrite this file.
When 'saving as a library', the entire project is compiled. If an error occurs thereby, then you are told that a correct project is necessary in order to create a library. The project is then not saved as a library.

'File' 'Save/Mail Archive'

This command is used to set up and create a project archive file. All files which are referenced by and used with a TwinCAT PLC project can be packed in a compressed zip file. The zip file can be stored or can be directly sent in an email. This is useful if you want to give forward a set of all project relevant files. When the command is executed, the dialog box 'Save Archive' opens:



Here you can define which file categories should be added to the archive zip file: Select or deselect a category by activating/deactivating the corresponding checkbox. Do this by a single mouseclick in the checkbox or by a doubleclick on the category name. If a category is marked with an arrow, all files of this category will be added to the zip file, if it is marked without an arrow, none of the files will be added. To select single files of a category press the corresponding button Details.



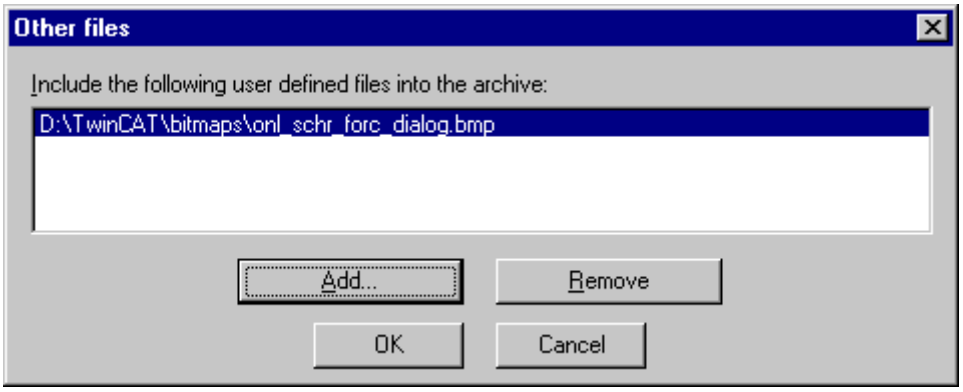
The dialog **'Details'** will open with a list of available files. In this dialog select/deselect the desired files: Use the button **Select All** or **Select None** to affect the complete list. A single file can be selected/deselected by a mouseclick in the checkbox, also by a doubleclick on the list entry or by pressing the spacebar when the list entry is marked. Close the Details dialog with Save to store the new settings.

In the main dialog **Save Archive** the checkbox of categories, for which not all files are selected, will appear with a grey background color.

The following file categories are available, the right column of the table shows which files can be added to the zip file:

Category	Associated Files
Project File	<projectname>.pro (the TwinCAT PLC Control project file)
Referenced Libraries	*.lib, *.obj, *.hex (libraries and if available the corresponding object and hex-files)
Compile Information	*.ci (compile information), *.ri (download/reference information) <temp>.* (temporary compile and download files) also for simulation
Log	*.log (project log file)
INI File	TwinCAT PLC Ctrl.ini
Registry Entries	
Symbol Files	*.sdb, *.sym (symbol information created from the project)
Bitmap Files	*.bmp (bitmaps for project POUs and visualizations)

To add any other files to the zip, press the button **Other Files**. The dialog 'Other files' will open where you can set up a list of desired files.



Press the button **Add** to open the standard dialog for opening a file, where you can browse for a file. Choose one and confirm with **Open**. The file will be added to the list in the '**Other files**' dialog. Repeat this for each file you want to add. To delete entries from the list, press the button Remove. When the list of selected files is ok, close the dialog with OK.
To add a Readme file to the archive zip, press the button **Comment**. A text editor will open, where you can enter any text. If you close the dialog with OK, during creation of the zip file a **readme.txt** file will be added.

Additionally to the entered comments it will contain information about the build date.

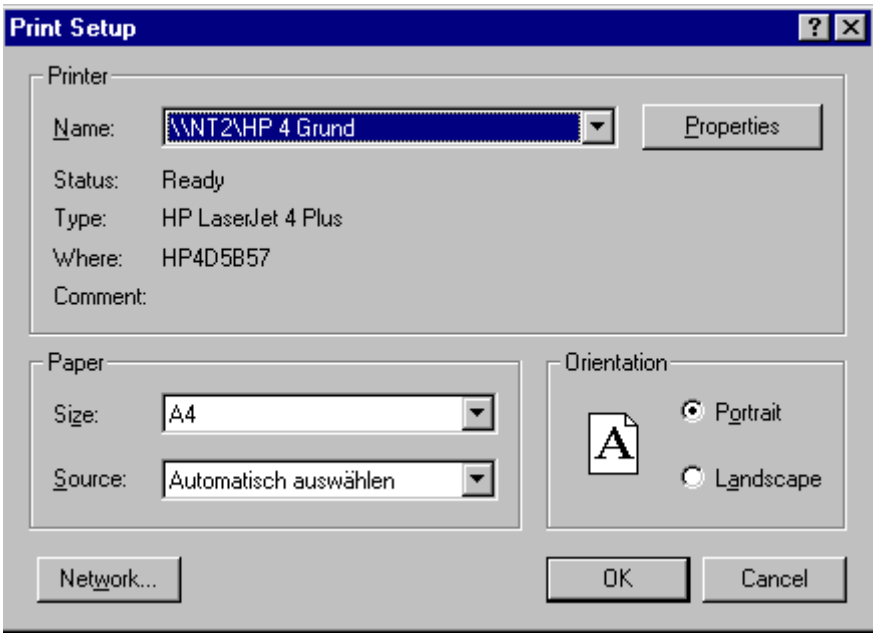
Generate zip archive:

If all desired selections have been made, in the main dialog press

- **Save...** to create and save the archive zip file: The standard dialog for saving a file will open and you can enter the path, where the zip should be stored. The zip file per default is named <projectname>.zip. Confirm with Save to start building it. During creation the current progress status is displayed and the subsequent steps are listed in the message window.
- **Mail…** to create a temporary archive zip and to automatically generate an empty email which contains the zip as an attachment. This feature only works if the MAPI (Messaging Application Programming Interface) has been installed correctly on the system, otherwise an error message is generated. During setup of the email the progressing status is displayed and the steps of the action are listed in the message window. The temporary zip file will be removed automatically after the action has been finished.
- **Cancel** to cancel the action; no zip file will be generated.

'File' 'Print' Shortcut: <Ctrl>+<P>

With this command the content of the active window is printed. After the command has been chosen, then the Print dialog box appears. Choose the desired option or configure the printer and then click OK. The active window is printed. Color output is available from all editors.



Dialog box for printing

You can determine the **number of the copie**s and print the version to a file. With the button **Properties** you open the dialog box to set up the printer. You can determine the layout of your printout with the command **"File" "Document settings"**. During printing the dialog box shows you the number of pages already printed. When you close this dialog box, then the printing stops after the next page. In order to document your entire project, use the command **"Project" "Document"**. If you want to create a document frame for your project, then open a global variables list and use the command **"Extras" "Make Docuframe file"**.

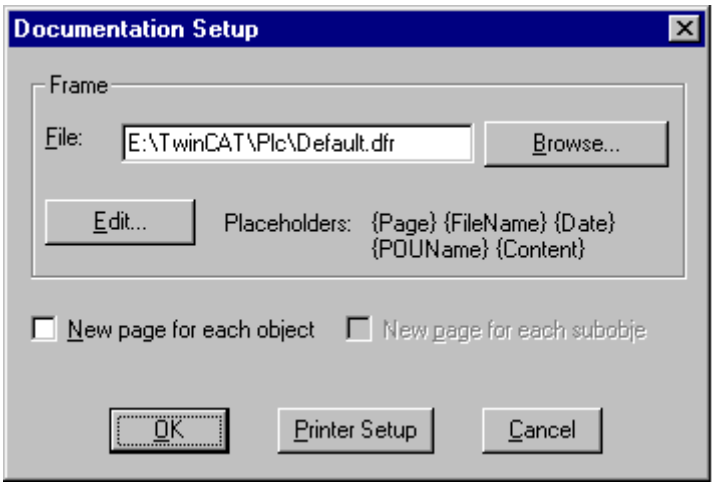
If the focus is in the message window, then all contents are printed out, represented line by line, as in the window displayed.

Possible Contents:

- build output
- cross reference list
- search result
- comparison result
- batch protocol

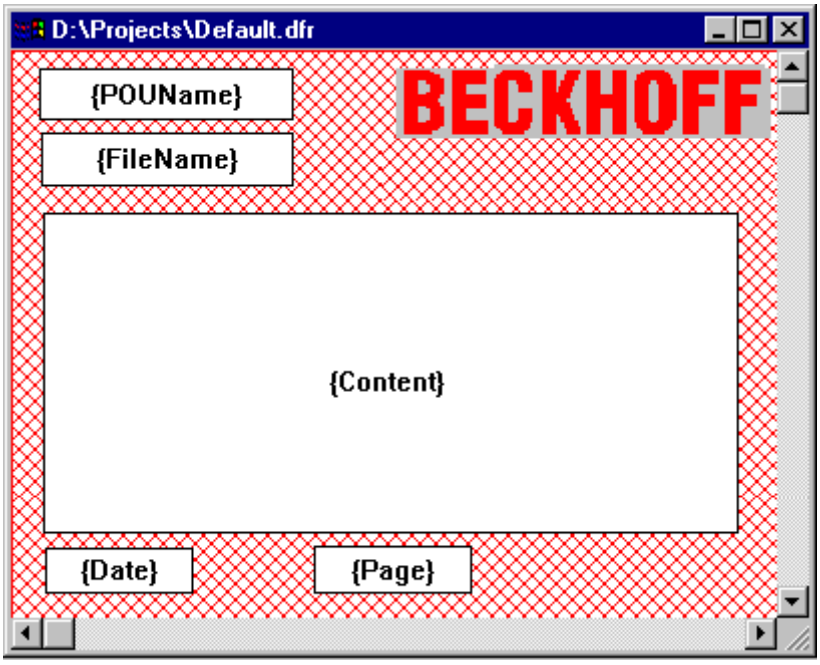
'File' 'Printer setup'

With this command you can determine the layout of the printed pages. The following dialog box is now opened:



Page Layout Dialog Box

In the field **File** you can enter the name of the file with the extension ".dfr" in which the page layout should be saved. The default destination for the settings is the file DEFAULT.DFR.
If you would like to change an existing layout, then browse through the directory tree to find the desired file with the button **Browse**.
You can also choose whether to begin a **new page** for each object and for each **subobject.**
Use the Printer **Setup** button to open the printer configuration. If you click on the Edit button, then the frame for setting up the page layout appears. Here you can determine the page numbers, date, filename and POU name, and also place graphics on the page and the text area in which the documentation should be printed.



Window for pasting the placeholders on the page layout

With the menu item **"Insert" "Placeholder**" and subsequent selection among the five placeholders **(Page, POU name, File name, Date, and Content)**, insert into the layout a so-called placeholder by dragging a rectangle on the layout while pressing the left mouse button. In the printout they are replaced as follows:

Command	Placeholder	Description
Page	{Page}	Here the current page number appears in the printout.
POU name	{POUName}	Here the current name of the POU appears.
File name	{FileName}	Here the name of the project appears.
Date	{Date}	Here the current date appears.
Contents	{Content}	Here the contents of the POU appears.

In addition, with "Insert" "Bitmap" you can insert a bitmap graphic (e.g. a company logo) in the page. After selecting the graphic, a rectangle should also be drawn here on the layout using the mouse. If the template was changed, then TwinCAT PLC Control asks when the window is closed if these changes should be saved or not.

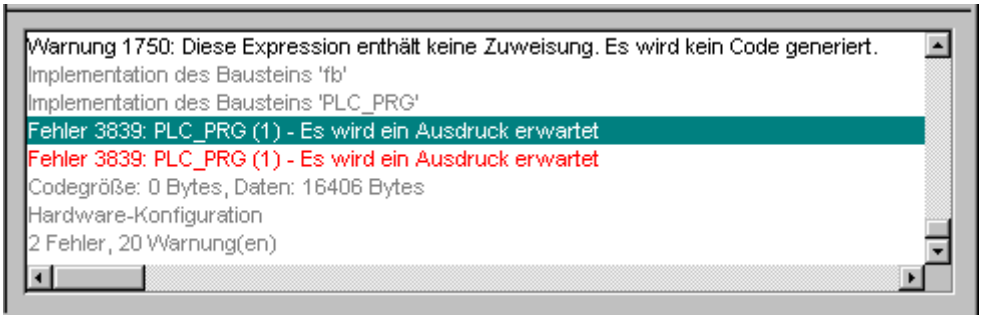
'File' 'Exit' Shortcut: <Alt>+<F4>

With this command you exit from TwinCAT PLC Control. If a project is opened, then it is closed as described in "File" "Save".

Possible Actions with Projects:

'Project' 'Build' <Ctrl>+<F8>

The project is compiled using **'Project' 'Build'**. The compilation process is basically incremental, that is only changed POUs are recompiled. A non-incremental compilation can also be obtained if the command **'Project' 'Clear all'** is first executed.
For target systems that support Online Change, all POUs that will be loaded into the controller on the next download are marked with a blue arrow in the Object Organizer after compilation.
The compilation process that is carried out with **'Project' 'Build'** occurs automatically if the controller is logged-in via **'Online' 'Log-in'.**
During compilation a message window is opened which shows the progress of the compilation process and any errors and warnings which may occur during compilation. Errors and warnings are marked with numbers. Using F1 you get more information about the currently selected error. A list of all error messages [▶ 223] is to be found in the appendix.



If the option **Save before compilation** is selected in the options dialog of the Load & Save category, the project is stored before compilation.

The cross-references are created during compilation and are stored in the build information. In order to apply the commands **Browse Call Tree**, **Show Cross Reference** and the commands **Unused variables**, **Concurrent access** and **Multiple write access on output** of the **'Project' 'Check'** menu, the project must be rebuilt after a change.

'Project' 'Rebuild all'

With **'Project' 'Rebuild all'** unlike incremental compilation (**'Project' 'Build'**) the project is completely recompiled. However, the download information will not be discarded, as is the case with the **'Clean all'** command.

'Project' 'Clean All'

This command deletes the information of the last download and the last compilation process.
After selecting the command, a dialog box appears indicating that login is no longer possible without a new download. Here the command can be canceled or confirmed.

After **'Clean all'** a login on the PLC project is only possible if the *.ri file with the project information from the last download was first explicitly saved outside the project directory (see 'Load Download-Information') and can now be reloaded prior to logging-in.

Project' 'Load Download- Information'

With this command the Download-Information belonging to the project can get reloaded, if it was saved to a directory different from that where the project is. After choosing the command the standard dialogue **'File Open'** opens.

The Download-Information is saved automatically at each download to a file, which is named **<project name><target identifier>.ri** and which is put to the project directory. This file is loaded, when the project is opened and at login it is used to check whether the PLC project is fitting to the currently opened TwinCAT PLC Control project (Id-check). Furthermore it is used to check, in which POUs the code has been changed. In systems which support the online change functionality, then only these POUs will be loaded to the PLC

during online change procedure. But: If the *.ri-file in the project directory gets deleted by the command **'Project' 'Clean all'**, you only can reload the Download-Information, if you had stored the *.ri-file in another directory too.

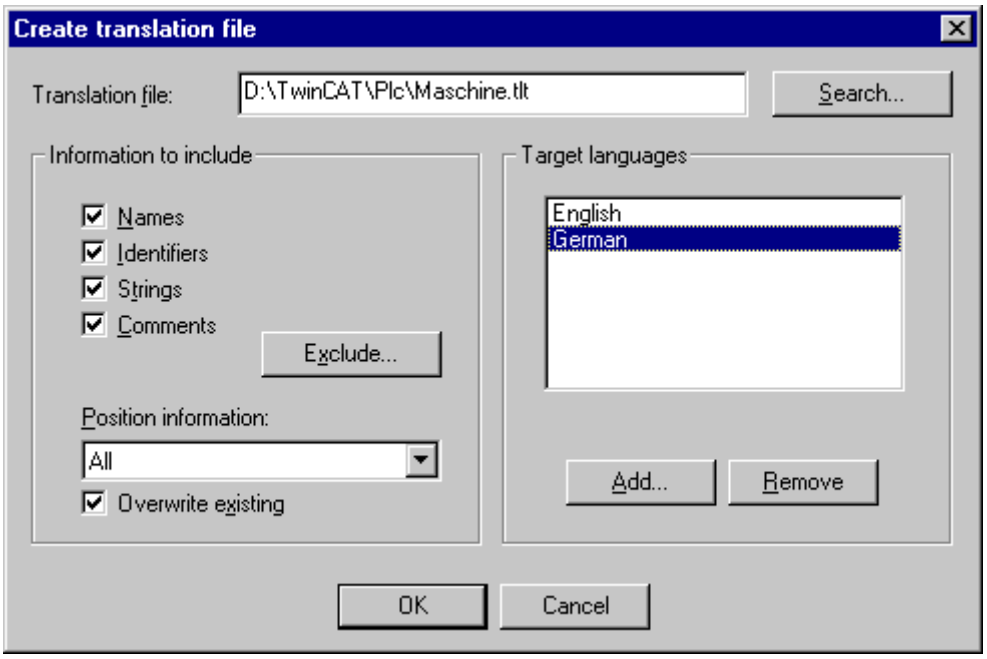
‘Project’ 'Translate into another language‘

This menu item is used for translating the current project file into another language. This is carried out by reading in a translation file that was generated from the project and externally enhanced in the desired national language with the help of a text editor.
Two menu sub-items are present:

- Create translation file
- Translate project

Create translation file

This command in the **'Project' "** menu leads to the **'Create translation file'** dialog:



In the **Translation file** field, enter a path that shows where the file is to be stored. The default file extension is ***.tlt**; this is a text file. Also possible is the using of the extension ***.txt.** This is recommendable if the file should be used with EXCEL or WORD.
If there already exists a translation file which you want to process, give the path of this file or use the Search button to reach the standard Windows file selection dialog.

The following information from the project can optionally be passed to the translation file that is being modified or created, so that they will be available for translation: **Names** (names, e.g. the title 'POUs' in Object Organizer), **Identifiers**, **Strings**, **Comments**. In addition, **Position information** for these project elements can be transferred.

If the corresponding options are checked, the information from the current project will be exported as language symbols into a newly created translation file or added to an already existing one. If the respective option is not selected, information belonging to the pertinent category, regardless of which project it came from, will be deleted from the translation file.

Position information: It describes with the specifications file path, POU and line the position of the language symbol, made available for translation. Three options are available for selection:

- **None:** No project information is generated
- **First occurrence:** The position on which the element first appears is added to the translation file.
- **All:** All positions on which the corresponding element appears are specified.

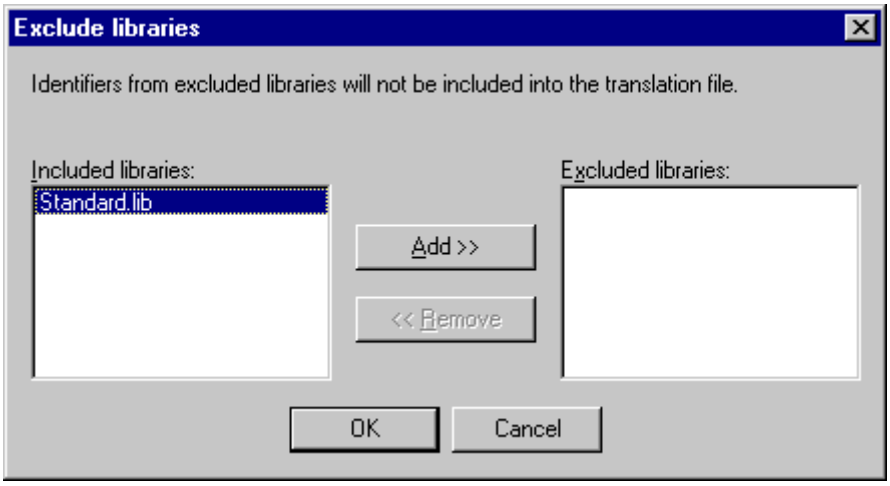
If a translation file created earlier is to be edited which already contains more position information than that currently selected, it will be correspondingly truncated or deleted, regardless of which project it was generated from.



A maximum of 64 position specifications will be generated per element (language symbol), even if the user has selected “All” under “Position Information” in the ‚Create Translation File' dialog.

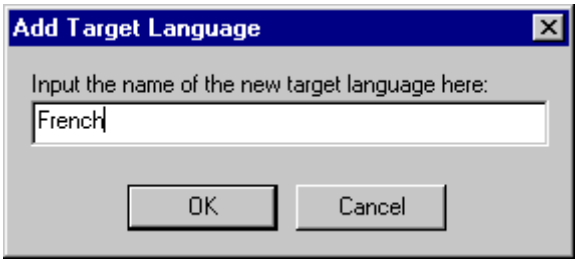
Overwrite existing: Existing position information in the translation file, that is currently being processed, will be overwritten, regardless of which project generated it.

Target languages: This list contains identifiers for all languages which are contained in the translation file, as well as those to be added upon completion of the '**Create translation file**' dialog.
The **Exclude** button opens the 'Exclude libraries' dialog.



Here, libraries included to the project can be selected, whose identifier information is not to be transferred to the translation file. To accomplish this, the corresponding entry in the table **Included libraries** on the left is selected with the mouse and placed in the Excluded libraries table to the right using the **Add** button. Likewise, entries already placed there can be removed using the **Remove** button. **OK** confirms the setting and closes the dialog.

The **Add** button opens the dialog '**Add Target Language**':



A language identifier must be entered into the editor field; it may not have a space at either the beginning or the end.

OK closes the '**Add Target Language**' dialog and the new target language appears in the target language list.
The **Remove** button removes a selected entry from the list.
You may also confirm the “**Create translation file**” dialog via **OK,** in order to generate a translation file.

If a translation file of the same name already exists you will get the following confirmation message to be answered Yes or No:
" The specified translation file already exists. It will now be altered and a backup copy of the existing file will be created. Do you want to continue?"

No returns you without action to the 'Create translation file' dialog.

If **Yes** is selected, a copy of the existing translation file with the filename “Backup_of_<translation file>.xlt” will be created in the same directory and the corresponding translation file will be modified in accordance with the options that have been entered.

The following takes place when a translation file is generated:

- For each new target language, a placeholder (”##TODO”) is generated for each language symbol to be displayed.
- If an existing translation file is processed, file entries of languages that appear in the translation file, but not in the target language list, are deleted, regardless of the project from which they were generated.

Editing of the translation file

The translation file must be opened and saved as a text file. The signs ## mark keywords. The ##TODO-placeholders in the file can be replaced by the valid translation. For each language symbol a paragraph is generated which starts with a ##NAME_ITEM and ends with a ##END_NAME_ITEM. (For comments correspondingly ##COMMENT_ITEM etc.).

See in the following an example of a translation file paragraph which handles the name of one of the POUs of the project. ST_Visu. The target languages shall be English(USA) and French. In this example the position information of the project element which should be translated has been added:

before translation:

```
##NAME_ITEM

[D:\projects\Bspdt_22.pro::ST_Visualisierung::0]

ST_Visualisierung

##English :: ##TODO

##French :: ##TODO

##END_NAME_ITEM
```

after translation:

```
##NAME_ITEM

[D:\projects\Bspdt_22.pro::ST_Visualisierung::0]

ST_Visualisierung

##English :: ST_Visualization

##French :: ST_Visu

##END_NAME_ITEM
```

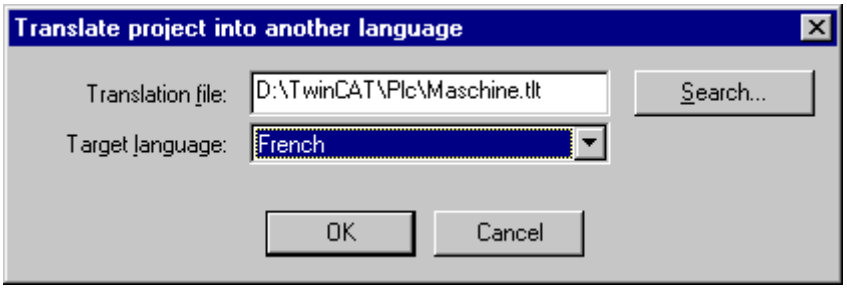
Please check that the translated Identifier and Names remain valid concerning the standard and that strings and comments are in correct brackets.



The following parts of a translation file should not be modified without detailed knowledge: Language block, Flag block, Position information, Original texts.

Translate Project (into another Language)

This command in the **'Project' 'Translate into Another Language'** menu opens the 'Translate Project into Another Language' dialog.



The current project can be translated into another language if an appropriate translation file is used.

If you want to save the version of the project in the language in which it was originally created, save a copy of the project prior to translation under a different name. The translation process cannot be undone.

In the field **Translation file** ,provide the path to the translation file to be used. By pressing **Search** you may access the standard Windows file selection dialog.

The field **Target language** contains a list of the language identifiers entered in the translation file, from which you can select the desired target language.

OK starts the translation of the current project into the chosen target language with the help of the specified translation file. During translation, a progress dialog is displayed, as well as error messages, if any. After translation, the dialog box and all open editor windows of the project are closed.

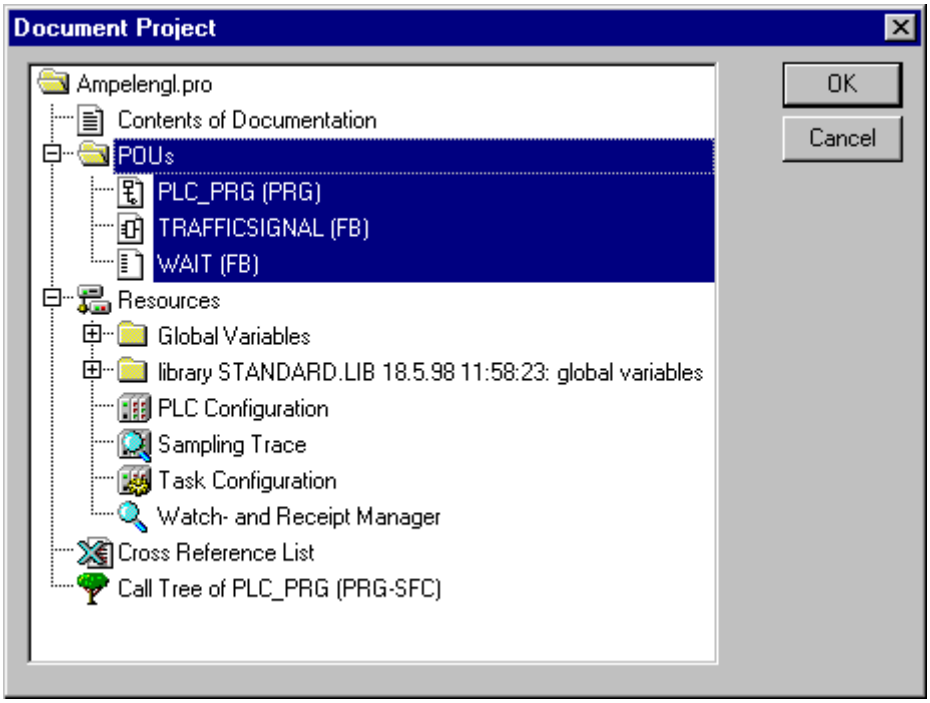
Cancel closes the dialog box without modification to the current project. If the translation file contains erroneous entries, an error message is displayed after OK is pressed, giving the file path and the erroneous line, e.g.: “[C:\Programs\projects\visu.tlt (78)]; Translation text expected.

'Project' 'Document'

This command lets you print the documentation of your entire project. The elements of a complete documentation are:

- The POUs,
- the contents of the documentation,
- the data types,
- the ressources (global variables, the sampling trace, the PLC configuration, the task configuration, the watch and recipe manager)
- the call trees of POUs and data types, as well as
- the cross reference list.

For the last two items the project must have been built without errors.



Dialog box for project documentation

Only those areas in the dialog box are printed which are highlighted in blue.

If you want to select the entire project, then select the name of your project in the first line. If, on the other hand, you only want to select a single object, then click on the corresponding object or move the dotted rectangle onto the desired object with the arrow key. Objects which have a plus sign in front of their symbols are organization objects which contain other objects. With a click on a plus sign organization object is expanded, and with a click on the resulting minus sign it can be closed up again. When you select an organization object, then all relevant objects are also selected. By pressing the <Shift> key you can select a group of objects, and by pressing the <Ctrl> key you can select several individual objects.

Once you have made your selection, then click on OK. The Print dialog box appears. You can determine the layout of the pages to be printed with **"File" "Printer setup"**.

'Project' 'Export'

With TwinCAT PLC Control projects can be exported or imported. That allows you to exchange programs between different IEC programming systems.
There is a standardized exchange format for POUs in IL, ST, and SFC (the Common Elements format of IEC 61131-3).
For the POUs in LD and FBD and the other objects TwinCAT PLC Control has its own filing format since there is no text format for this in IEC 61131-3.

The selected objects are written to an ASCII file.

POUs, data types, and the ressources can be exported. In addition, entries in the library manager, that is the linking information to the libraries, can be exported (not the libraries themselves!).



The re-import of an exported FBD or LD function block fails if a comment in the graphical editor contains a single quotation mark ('), because this is interpreted as string beginning.

When you have made your selection in the dialog box (the selection is made as described at **'Project' 'Document'**), you can still decide whether you want to export the selection to a file or have a separate export file generated for each object. To do this, turn off or on the **One file for each object** option accordingly, and then click **OK**. The dialog for saving files appears. Specify a file name with extension ".exp" or a directory for the individual object export files, which are then created there under "Object name.exp".

'Project' 'Import'

In the dialog that appears for opening files, select the desired export file.
The data will be imported into the current project. If an object with the same name already exists in the project, then a dialog box appears asking "Do you want to replace it?"If you answer with **Yes**, the object in the project will be replaced by the object from the import file, if you answer with **No**, the name of the new object will be supplemented by an underscore and a number ("_0", "_1", ..). With **Yes, all** or **No, all** this is done for all objects.

If the information to link to a library is imported, the library is loaded and added to the end of the list in the Library Manager. If the library has already been loaded in the project, it will not be loaded again. However, if a different save time for the library is specified in the export file that is imported, the library name is marked with an "*" in the Library Manager (e.g. standard.lib*30.3.99 11:30:14), analogous to loading a project. If the library cannot be found, the information dialog : "Cannot find library {<path>\}<name> <date> <time>" appears, analogous to loading a project.
In the message window the import is logged.

'Project' 'Merge'

This command allows you to copy objects (function blocks, data types and resources) and links to libraries from other projects into your project. If the command was given, then first opens the standard dialog for opening files. If you have selected a file there, then opens a dialog in which you can select the desired objects. The selection is made as described at **'Project' 'Document'**. If an object with the same name already exists in the project, then the name of the new object receives an underscore and a number ("_1", "_2" ...) as the last character.

'Project' 'Compare'

This command is used to compare two projects, or the current version of the opened project with the one that was saved last.

Overview:

In the following, the designation **'current project'** is used for the project that is currently being processed, and **'comparison project'** for the one that is called for comparison. After selecting the command, the project is displayed in the **comparison mode**. In the following, **'unit'** refers to the smallest comparison unit, which can consist of a line (declaration editor, ST, IL), a network (FBD, LD) or an element/function block (CFC, SFC).

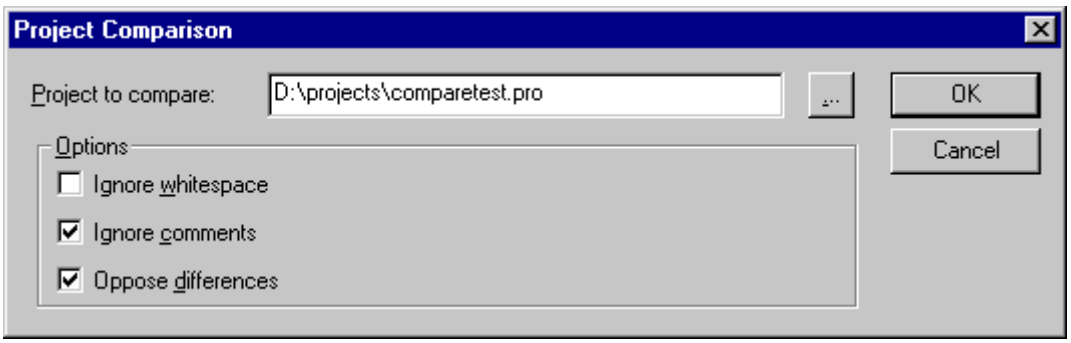
In comparison mode, the current and comparison projects are compared in a divided window, and the blocks found to be different are color-coded. In the case of editor function blocks, there is also direct comparison for the contents. Before the comparison run, filters regarding the consideration of blanks and comments can be activated. Furthermore, it can be selected whether changes within existing units are displayed as such in the comparison mode or whether all different units are marked as 'newly inserted' or 'no longer present'. The version of the comparison project can be transferred to the current project for different units or for a whole block of equally marked ones.



In compare mode (see status bar: COMPARE) the project cannot get edited !

Execute comparison:

After executing the command 'Project' 'Compare' the dialog 'Project Comparison' opens:



Insert the path of the reference project at **Project to compare**. Press button if you want to use the standard dialog for opening a project. If you insert the name of the actual project, the current version of the project will be compared with the version which was saved last.

If the project is under source control in an ENI data base, then the local version can be compared with the actual version found in the data base. For this activate option Compare with ENI-Project.

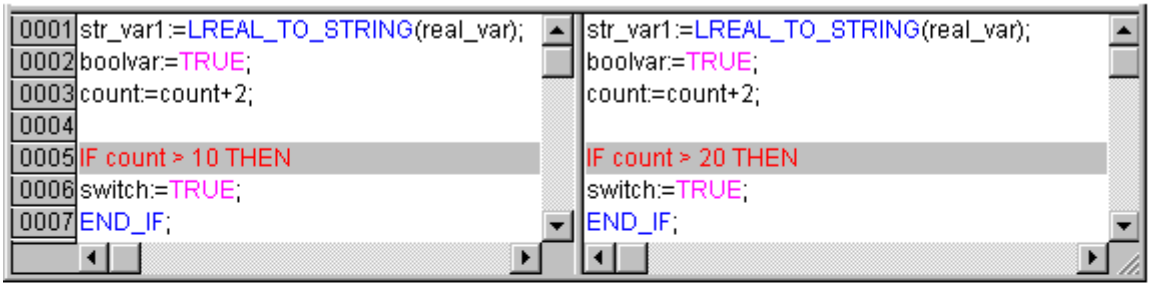
The following **options** concerning the comparison can be activated/deactivated:

- **Ignore whitespaces:** There will be detected no differences which consist in a different number of whitespaces.
- **Ignore comments:** There will be detected no differences in comments.
- **Oppose differences:** If a line, a network or an element within a POU has been modified, in compare mode it will be displayed in the bipartited window directly opposite to the version of the other project (marked red, see below). If the option is deactivated, the corresponding line will be displayed in the reference project as 'deleted' and in the actual project as 'inserted' (blue/green, see below). This means it will not be displayed directly opposite to the same line in the other project.

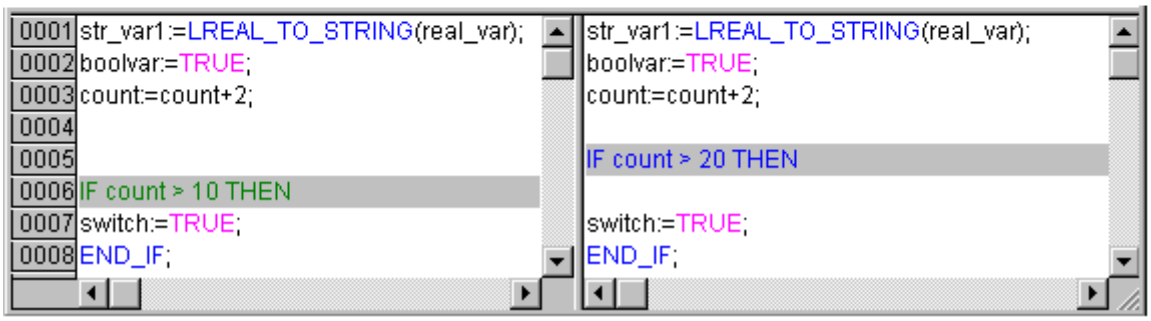
Example:

Line 0005 has been modified in actual project (left side).

Option 'Oppose differences' activated:



Option 'Oppose differences' not activated:

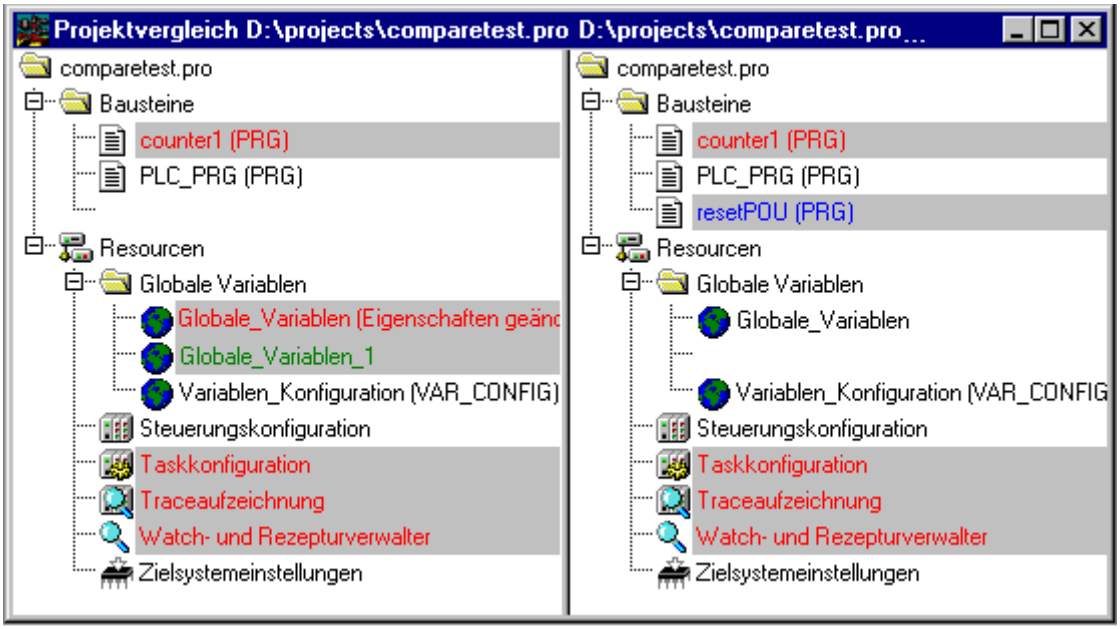


When the dialog Project Comparison is closed by pressing **OK**, the comparison will be executed according to the settings.

Representation of the comparison result:

1. Project overview in compare mode:

After the project compare has been executed, a bipartited window opens which shows the project in compare mode. In the title bar you find the project paths: "Project comparison <path of actual project> - <path of reference project>". The actual project is represented In the left half of the window, the reference project in the right one. Each structure tree shows the projects' name at the uppermost position, apart from that it corresponds to the object organizer structure.



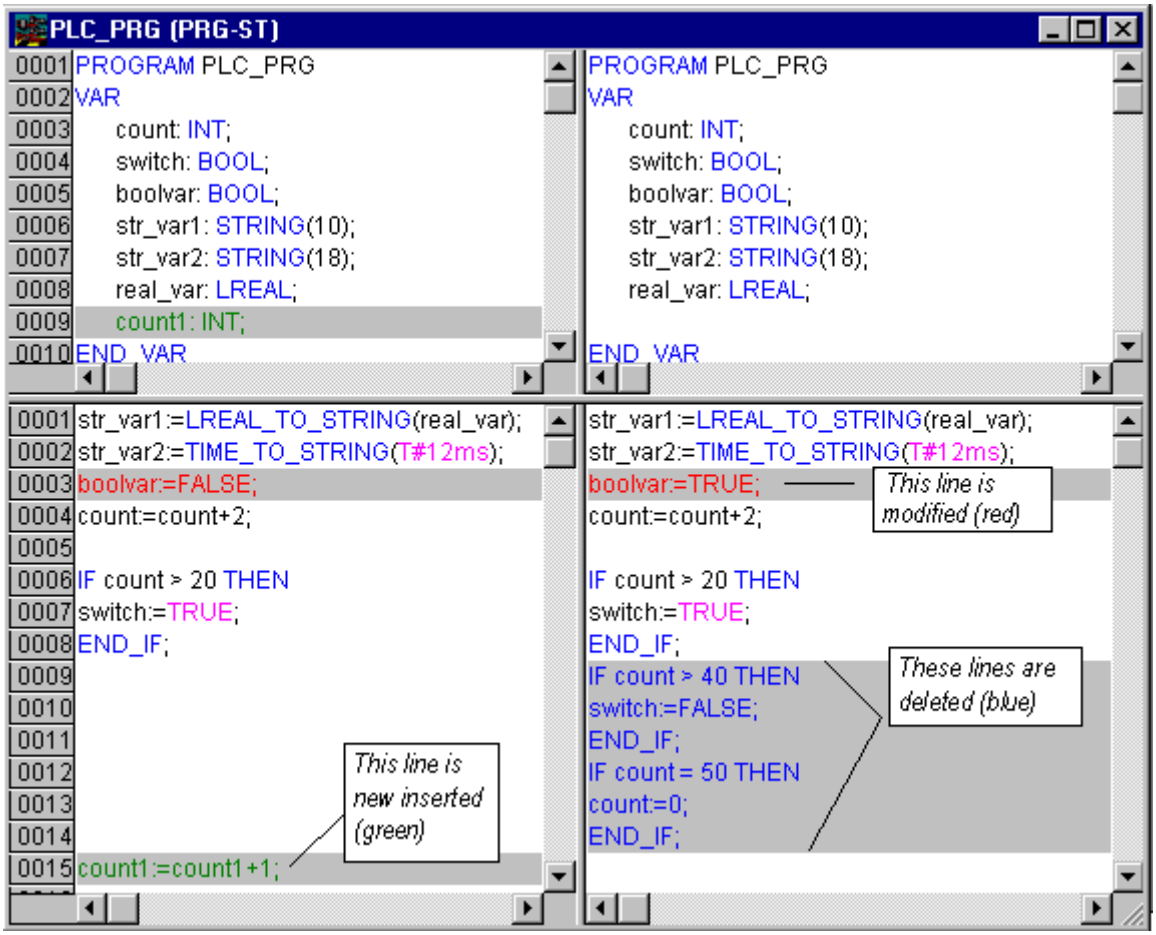
POUs which are different, are marked in the structure tree by a shadow, a specific color and eventually by an additional text :

- **Red:** Unit has been modified; is displayed with red colored letters in both partitions of the window.
- **Blue:** Unit only available in compare project; a gap will be inserted at the corresponding place in the structure overview of the actual project.
- **Green:** Unit only available in actual project; a gap will be inserted at the corresponding place in the structure overview of the actual project.
- **Black:** Unit for which no differences have been detected.
- **"(Properties changed)":** This text is attached to the POU name in the project structure tree, if differences in the properties of the POU have been detected.
- **"(Access rights changed)":** This text is attached to the POU name in the project structure tree, if differences in the access rights of the POU have been detected.

2. POU contents in compare mode:

By a double click on a line in the structure overview, which is marked red because of a modification, the **POU is opened**.

If it is a text or graphic editor POU, it will be opened in a bipartited window. The content of the reference project (right side) is set opposite to that of the actual project (left side). The smallest unit which will be regarded during comparison, is a line (declaration editor, ST, IL), a network (FBD, LD) or an element (CFC, SFC). The same coloring will be used as described above for the project overview.
Example:



If it is not a editor POU, but the task configuration, the target settings etc., then the POU version of the actual and the reference project can be opened in separate windows by a double click on the respective line in the project structure. For those project POUs no further details of differences will be displayed.

Working in the compare mode (Menu 'Extras', Context menu):

If the cursor is placed on a line in the bipartited window, which indicates a difference, the menu 'Extras' resp. The context menu offers a selection of the following commands, depending on whether working in the project overview or in a POU.

'Next difference'(<F7>): The cursor jumps to the next unit, where a difference is indicated. (line in project overview, line/network/element in POU)

'Previous difference'(<Shift><F7>): The cursor jumps to the previous unit, where a difference is indicated . (line in project overview, line/network/element in POU)

'Accept change'(<Space bar>): For all coherent (e.g. subsequent lines) units, which have the same sort of difference marking, the version of the reference project will be accepted for the actual project. The corresponding units will be shown (with the corresponding coloring) in the left side of the window. If it is an unit, which is marked red (just modification), then the acceptance will be recognizable by yellow coloring of the text in the actual project.

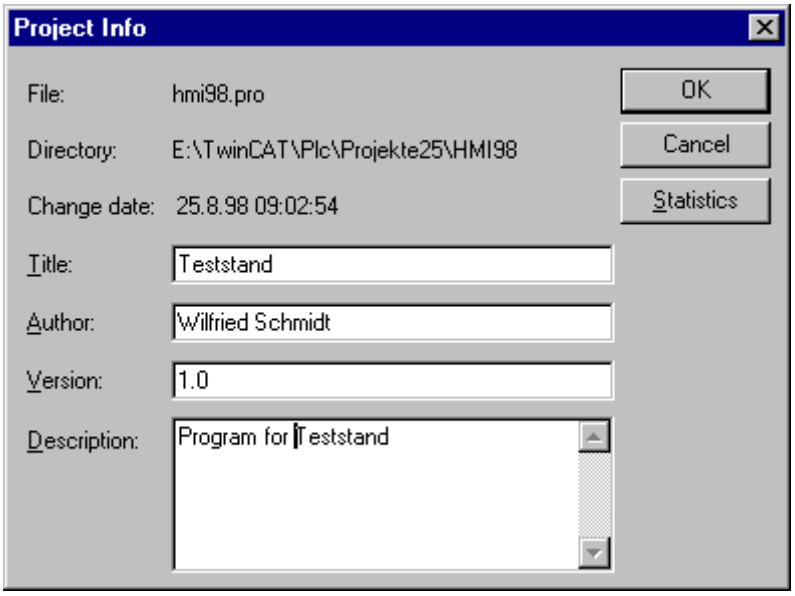
'Accept changed item'(<Strg><Space bar>): Only the single unit (line, network, element) where the cursor is currently placed, will be accepted for the actual version. The corresponding units will be shown (with the corresponding coloring) in the left side of the window. If it is an unit, which is marked red (just modification), then the acceptance will be recognizable by yellow coloring of the text in the actual project.

'Accept properties'(only in project overview): The object properties for the POU, where the cursor is currently placed, will be accepted for the actual project as they are set in the reference version.

'Accept access rights' (only in project overview): The object access rights for the POU, where the cursor is currently placed, will be accepted for the actual project as they are set in the reference version .

'Project' 'Project Info'

Under this menu item the information about your project can be saved. When the command has been given, then the following dialog box opens:



Dialog box for entering project information

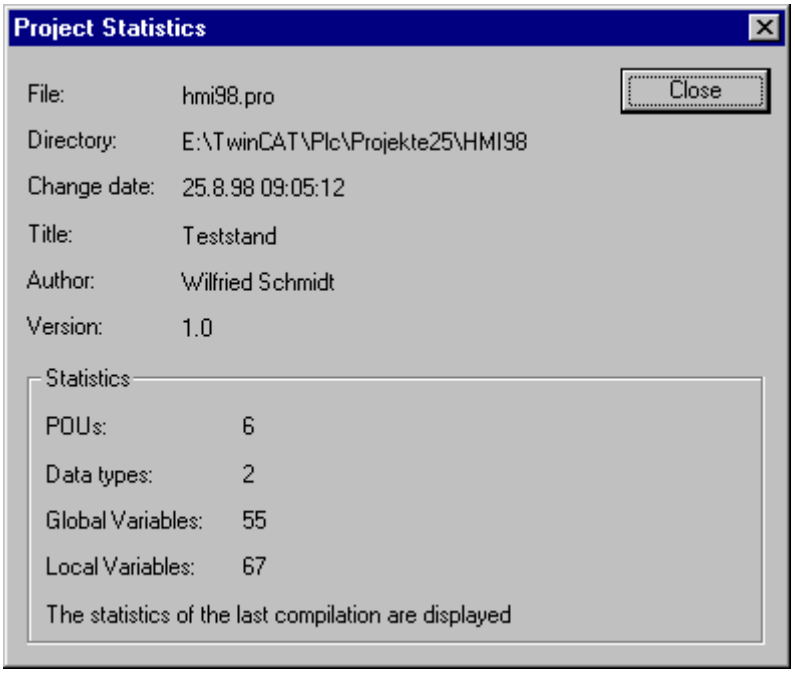
The following project information is displayed:

- File name
- Directory path
- The time of the most recent change (Change date)

This information can not be changed. In addition, you can can add the following information:

- A Title of the project,
- the name of the Author,
- the Version number, and
- a Description of the project.

This information is optional. When you press the button Statistics you receive statistical information about the project. It contains information such as the number of the POUs, data types, and the local and global variables as they were traced at the last compilation.



Example of project statistics

If you choose the option **Ask for project info** in the category **Load & Save** in the options dialog box, then while saving a new project, or while saving a project under a new name, the project info is called automatically.

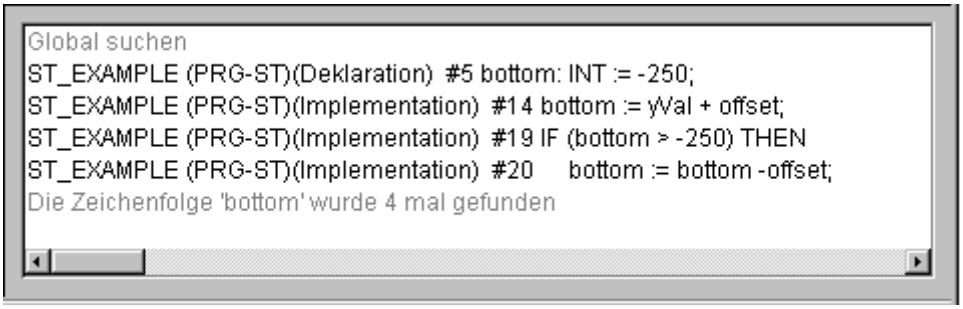
'Project' 'Global Search'

With this command you can search for the location of a text in POUs, data types, or in the objects of the global variables, in the PLC configuration, the task configuration and in the declaration part of the libraries. When the command is entered, a dialog box opens in which you can choose the desired object. The selection is made as in the "Project" "Document" description.
When you have confirmed the selection with **OK**, then the search dialog box appears.

This appears immediately when the command '**Global Search'** is invoked via the symbol in the menu bar; the search is then automatically carried out in all searchable parts of the project. The most recently entered search strings can be selected through the combo box of the **Search for** field. If a text string is found in an object, the object is loaded into the corresponding editor or in the library manager and the location where the string was found is displayed. The display of the text that is found, as well as the search and find next functions behave similarly to the command '**Edit' 'Search'.**

If you select the **In message window** button, all locations where the series of symbols searched for appears in the selected object will be listed line by line in tabular form in the message window. Afterward, the number of locations found will be displayed.
If the report window was not opened, it will be displayed. For each location that is found, the following will be displayed:

- Object name
- Location of the find in the Declaration (Decl) or in the Implementation (Impl) portion of a POU Line and network number if any
- The full line in the text editors
- Complete text element in the graphic editors



If you double-click the mouse on a line in the message window or press <Enter>, the editor opens with the object loaded. The line concerned in the object is marked. You can jump rapidly between display lines using the function keys <F4> and <Shift>+<F4>.

'Project' 'Global Replace'

With this command you can search for the location of a text in POUs, data types, or the objects of the global variables and replace this text by another. This is executed in the same way as with **"Project" "Global Search"** or **"Edit" "Replace"**. The libraries, however, are not offered for selection and no display in the message window is possible.

‘Project‘ ‘Check‘

A submenu listing the following commands will open:

- Unused Variables
- Overlapping memory areas
- Access conflict
- Multiple writes to output

Results are displayed in the message window.
Each of these functions tests the state of the most recent compilation. The project must therefore have been compiled error-free at least once, before the test can be carried out; if not, the menu items are “greyed out".
Results are displayed in the message window.

Unused Variables:
This function searches for variables that have been declared but not used in the program. They are outputted by POU name and line, e.g.: PLC_PRG (4) – var1. Variables in libraries are not examined.
Results are displayed in the message window.

Overlapping memory areas:
This function tests whether in allocation of variables via the “AT” declaration overlaps have arisen at specific memory areas. For example, an overlap occurs when allocating the variables “var1 AT %QB21: INT” and “var2 AT %QD5: DWORD” because they both use byte 21.
The output then appears as follows:

```
%QB21 is referenced by the following variables:
PLC_PRG (3): var1 AT %QB21
PLC_PRG (7): var2 AT %QD5
```

Results are displayed in the message window.

Access conflict:
This function in the 'Project' 'Check' menu searches for memory areas which are referenced in more than one task. No distinction is made here between read and write access. The output is for example:

```
%MB28 is referenced in the following tasks :
Task1 - PLC_PRG (6): %MB28 [read-only access]
Task2 - POU1.ACTION (1) %MB28 [write access]
```

Results are displayed in the message window.

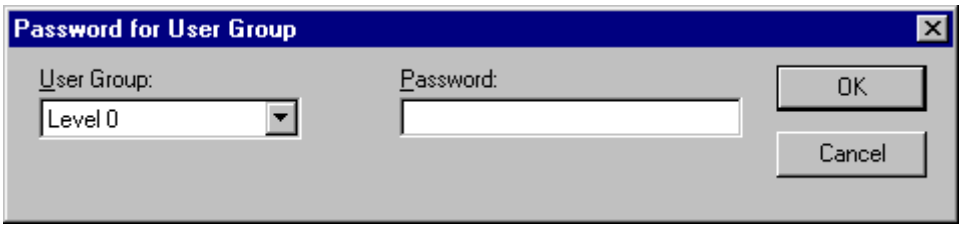
Multiple writes to output:
This function of the 'Project' 'Check' menu searches for memory areas to which a single project gains write access at more than one place. The output then appears as follows:

```
%QB24 is written to at the following locations:
PLC_PRG (3): %QB24
PLC_PRG.POU1 (8): %QB24
```

Results are displayed in the message window.

User groups

In TwinCAT PLC Control up to eight user groups with different access rights to the POUs, data types, and ressources can be set up. Access rights for single objects or all of them can be established. Only a member of a certain user group can open a project. A member of such a user group must identify himself by means of a password.
The user groups are numbered from 0 to 7, whereby the Group 0 has the administrator rights, i.e. only members of group 0 may determine passwords and access rights for all groups and/or objects.
When a new project is launched, then all passwords are initially empty. Until a password has been set for the 0 group, one enters the project automatically as a member of the 0 group.
If a password for the user group 0 is existing while the project is loaded, then a password will be demanded for all groups when the project is opened. For this the following dialog box appears:



Dialog box for password entry

In the combobox **User group** on the left side of the dialog box, enter the group to which you belong and enter on the right side the relevant **password.** Press **OK**. If the password does not agree with the saved password, then the message appears:
"The password is not correct."
Only when you have entered the correct password can the project be opened.

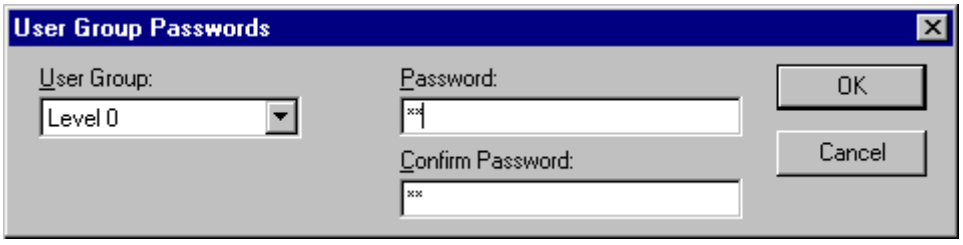


If passwords are not assigned for all user groups, it is possible to open a project via a user group for which none has been assigned!

With the command **'Passwords for user group'** you can assign passwords, with the command 'Object' 'Access rights' the rights for individual or all objects.

'Project' 'Passwords for user groups'

This command opens the password assignment dialog for user groups. This command can be executed only by members of group 0. When the command is executed, the following dialog opens:



Dialog for password assignment

In the left User Group combo box you can select the group. Enter the desired password in the Password field for this group. An asterisk (*) appears in the field for each typed letter. You must repeat the same word in the Confirm Password field. Close the dialog with OK after each password entry. If the message comes:" The password and its confirmation do not match."you have mistyped one of the two entries. Therefore, it is best

to repeat both entries until the dialog closes without a message. If necessary, only assign a password for the next group afterwards by calling the command again.
With the command 'Object' 'Access rights' you can assign the rights for individual or all objects

4.4 Objects

Now we shall explain how to work with objects and what help is available to keep track of a project (Folders, Call tree, Cross reference list,..).

Objects

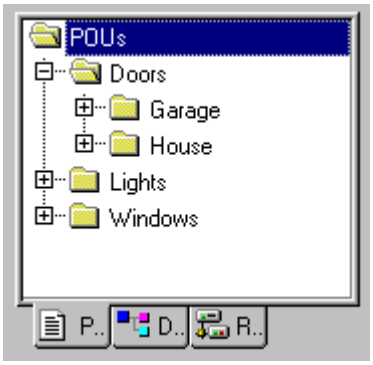
POUs, data types, and the ressources (global variables, the sampling trace, the PLC configuration, the task configuration, and the Watch and Recipe Manager) are all defined as "objects". The folders inserted for structuring the project are partially involved. All objects of a project are in the Object Organizer.
If you hold the mouse pointer for a short time on a POU in the Object Organizer, then the type of the POU (Program, Function or Function block) is shown in a Tooltip.For the global variables the tooltip shows the keyword (VAR_GLOBAL, VAR_CONFIG).

With drag & drop you can shift objects (and also folders, see 'Folder') within an object type. For this, select the object and shift it to the desired spot by holding down the left mouse button. If the shift results in a name collision, the newly introduced element will be uniquely identified by an appended, serial number (e.g. “Object_1”).

Folder

In order to keep track of larger projects you should group your POUs, data types and global variables systematically in folders.
You can set up as many levels of folders as you want. If a plus sign is in front of a closed folder symbol, then this folder contains objects and/or additional folders. With a click on the plus sign the folder is opened and the subordinated objects appear. With a click on the minus (which has replaced the plus sign) the folder can be closed again.
In the context menu you find the commands "**Expand nodes**" and "**Collapse nodes**" with the same functions.
With Drag&Drop you can move the objects as well as the folders within their object type. For this select the object and drag it with pressed left mouse button to the desired position.

Folders have no influence on the program, but rather serve only to structure your project clearly.



Example of folders in the Object Organizer

'New Folder'

With this command a new folder is inserted as a structural object. If a folder has been selected, then the new one is created underneath it. Otherwise it is created on the same level. If an action is selected, the new folder will be inserted at the level of the POU to which the action belongs.The context menu of the Object Organizer which contains this command appears when an object or the object type has been selected and you have pressed the right mouse button or <Shift>+<F10>.
The newly inserted folder initially has the designation 'New Folder'. Observe the following naming convention for folders:

- Folders at the same level in the hierarchy must have distinct names. Folders on different levels can have the same name.
- A folder can not have the same name as an object located on the same level.

If there is already a folder with the name “New Folder” on the same level, each additional one with this name automatically receives an appended, serial number (e.g. “New Folder 1”). Renaming to a name that is already in use is not possible.

'Expand nodes' 'Collapse nodes'

With the command Expand the objects are visibly unfolded which are located in the selected object. With Collapse the subordinated objects are no longer shown. With folders you can open or close them with a double mouse click or by pressing <Enter>.The context menu of the Object Organizer which contains this command appears when an object or the object type has been selected and you have pressed the right mouse button or <Shift>+<F10>.

'Project' 'Object Delete' Shortcut: <Del>

With this command the currently selected object (a POU, a data type or global variables), or a folder with the subordinated objects is removed from the Object Organizer and is thus deleted from the project. For safety you are asked once more for confirmation. If the editor window of the object was open, then it is automatically closed. If you delete with the command "Edit" "Cut", then the object is parked on the clipboard.

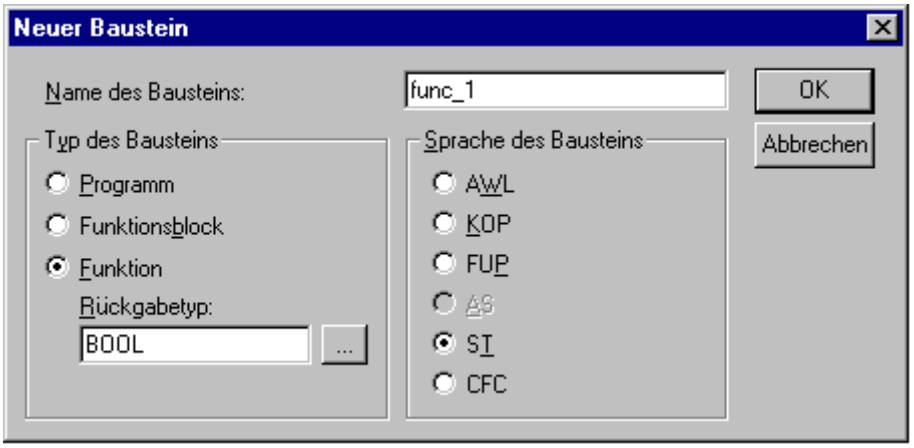
'Project' 'Object Insert' Shortcut: <Ins>

With this command you create a new object. The type of the object (POU, data type or global variables) depends upon the selected register card in the Object Organizer. Enter the **name** of the new object in the dialog box which appears. Remember that the name of the object may not have already been used.

Take note of the following restrictions:

- The name of a POU can not include any spaces
- A POU can not have the same name as another POU, or a data type.
- A data type can not receive the same name as another data type or a POU.
- A global variable list can not have the same name as another global variable list.
- An action can not have the same name as another action in the same POU.
- A visualization can not have the same name as another visualization.

In all other cases, identical naming is allowed. Thus for example actions belonging to different POUs can have the same name, and a visualization may have the same as a POU. In the case of a POU, the POU type (program, function or function block) and the language in which it is programmed must also be selected. 'Program' is the default value of **Type of the POU**, while that of **Language of the POU** is that of most recently created POU. If a POU of the function type is created, the desired data type must be entered in the Return Type text input field. Here all elementary and defined data types (arrays, structures, enumerations, aliases) are allowed. Input assistance (e.g. via <**F2**>) can be used.

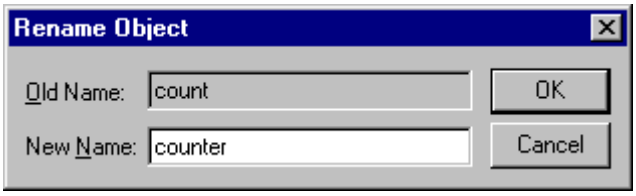


After pressing **OK**, which is only possible if there is no conflict with the naming conventions described above, the new object is set up in the Object Organizer and the appropriate input window appears.
If the command '**Edit' 'Insert**' is used, the object currently in the clipboard is inserted and no dialog appears. If the name of the inserted object conflicts with the naming conventions (see above), it is made unique by the addition of a serial number appended with a leading underline character (e.g. “Rightturnsig_1”).

If the project is under source control in an **ENI** data base, it may be (depends on the settings in the Project options dialog for 'Project source control') that you will be automatically asked in which data base category you want to handle the new object. In this case the dialog Properties will open where you can assign the object to one of the data base object categories.

'Project' 'Object Rename' Shortcut: <Spacebar>

With this command you give a new name to the currently-selected object or folder. Remember that the name of the object may not have already been used. If the editing window of the object is open, then its title is changed automatically when the name is changed.

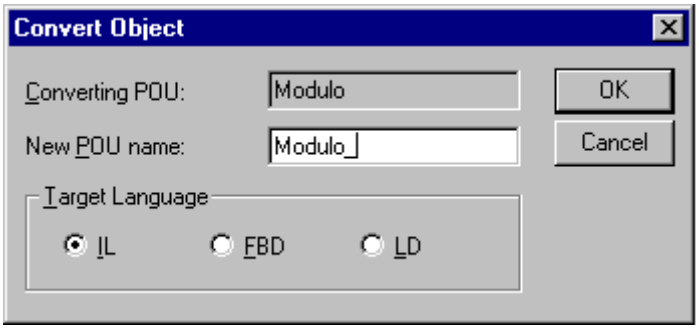


Dialog box for renaming a POU

'Project' 'Object Convert'

This command can only be used with POUs. You can convert POUs from the languages ST, FBD, LD, and IL into one of the three languages IL, FBD, and LD. For this the project must be compiled. Choose the language into which you want to convert and give the POU a new name. Remember that the name of the POU may not have already been used. Then press OK, and the new POU is added to your POU list.

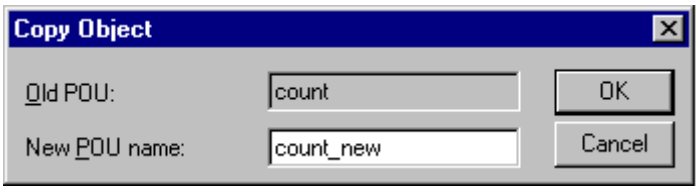
The type of processing that occurs during conversion corresponds to that which applies to compilation.



Dialog box for converting a POU

'Project' 'Object Copy'

With this command a selected object is copied and saved under a new name. Enter the name of the new object in the resulting dialog box. Remember that the name of the object may not have already been used. If, on the other hand, you used the command **"Edit" "Copy"**, then the object is parked on the clipboard, and no dialog box appears.



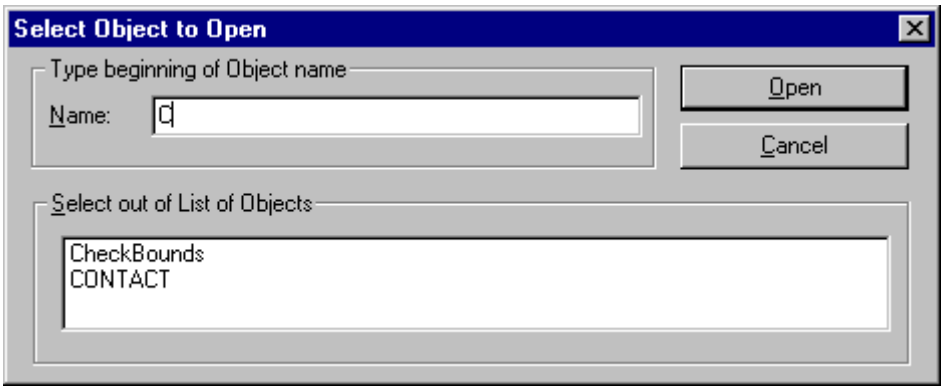
Dialog box for copying a POU

'Project' 'Object Open' Shortcut: <Enter>

With the command you load a selected object within the Object Organizer into the respective editor. If a window with this object is already open, then it gets a focus, is moved into the foreground and can now be edited. There are two other ways of opening an object:

- Doubleclick with the mouse on the desired object or
- type in the Object Organizer the first letter of the object name. Then a dialog box opens in which all objects of the available object types with this initial letter are shown. Select the desired object and click on the button Open in order to load the object in its edit window. Thereby the object is marked in the Object Organizer and all hierarchical in the object path above the object lying folders and objects were expanded. This option is supported with the object type Ressources only for global variables.

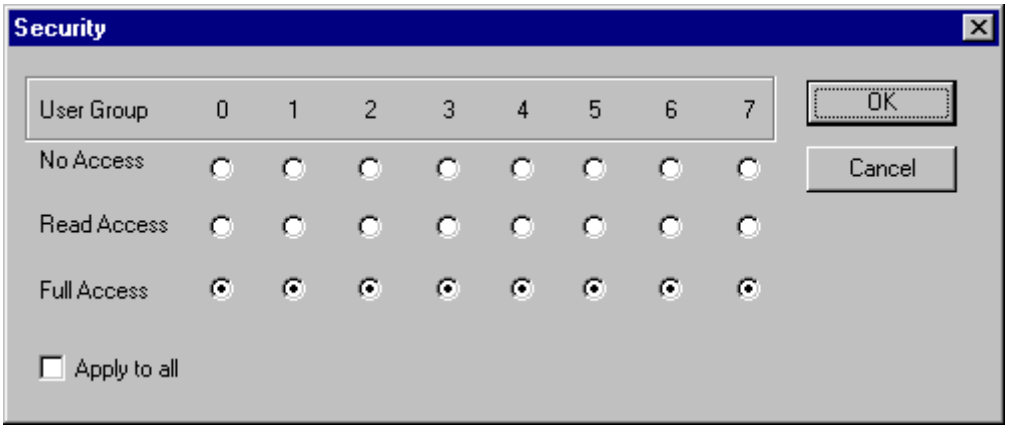
This last possibility is especially useful in projects with many objects.



Dialog box for choosing the object to be opened

'Project' 'Object Access rights'

With this command you open the dialog box for assigning access rights to the different user groups. The following dialog box appears:



Dialog box for assigning access rights

Members of the user group 0 can now assign individual access rights for each user group. There are three possible settings:

- **No Access**: the object may not be opened by a member of the user group.
- **Read Access**: the object can be opened for reading by a member of the user group but not changed.
- **Full Access**: the object may be opened and changed by a member of the user group.

The settings refer either to the currently-selected object in the Object Organizer or, if the option Apply to all is chosen, to all POUs, data types and ressources of the project. The assignment to a user group takes place when opening the project through a password request if a password was assigned to the user group 0.

'Project' 'Object properties '

This command will open the dialog 'Properties' for that object which is currently marked in the Object organizer. On the tab **Access rights** you find the same dialog as you get when executing the command 'Project' 'Object Access Rights'. It depends on the object and the project settings, whether there are additional tabs available where you can define object properties:

- If a global variable list is currently selected in the Object Organizer, then a tab '**Global variable list**' will be available where the parameters concerning the actualization of the list and concerning the data exchange of network variables are defined. The entries can be modified here. This dialog also will be opened if you create a new global variable list by selecting one of the entries in section 'Global Variables' in the Object Organizer and executing the command 'Add Object'.
- If the project is connected to an **data base** then a tab 'Database-connection' will be available. Here you can display and modify the current assignment of the object to one of the data base categories resp. to the category 'Local'.

'Project' 'Data base link'

This menu item is only available if you have activated the option '**Use source control (ENI)**' in the project options dialog for category '**Project source control**. A submenu is attached where you find the following commands for handling the object resp. the project in the currently connected ENI data base:

- Login (The user logs in to the ENI Server)

If an object is marked in the Object Organizer and the command Data Base Link is executed (from the context menu, right mouse button), then the following commands will be available for executing the corresponding data base actions:

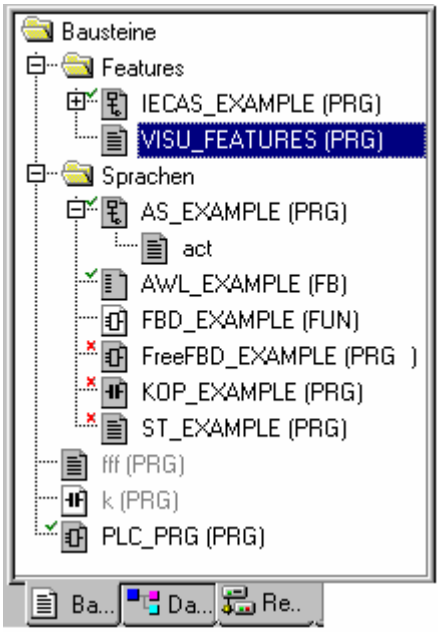
- Define
- Get Latest Version
- Check Out
- Check In
- Undo Check Out
- Show differences

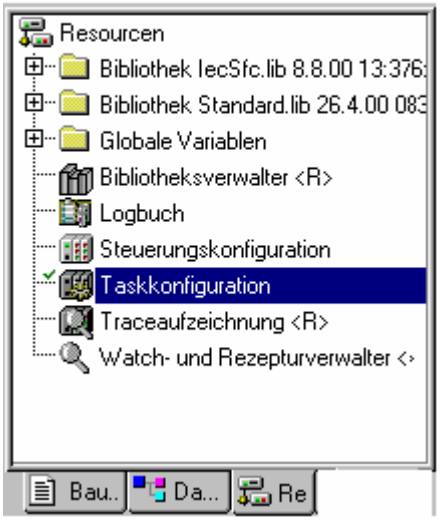
- Show Version History

If the command '**Data Base Link**' in the '**Project**' menu is activated, then additional menu items will be available, which concern all objects of the project:

- Multiple Define
- Get All Latest Versions
- Multiple Check Out
- Multiple Check In
- Multiple Undo Check Out
- Project Version History
- Label Version
- Add Shared Objects
- Refresh Status

See in the following the description of the commands.





Grey shaded icon:

Object is kept in the data base

Green check in front of the object name:

Object is checked out in the local project.

Red cross in front of the object name:

Object is currently checked out by another user.

<R> behind object name:

The object can only be read, but not edited.



Some objects (Task configuration, Sampling Trace, PLC Configuration, Target Settings, Watch- and Recipe Manager) are per default assigned with a <R> as long as they are not checked out. This means that you will not automatically be asked whether the object should be checked out, as soon as you start to edit the object; it not necessarily means that you cannot edit the object. If there is actually no write access then the command 'Check out' will not be available.

'Project' 'Data base command' 'Define'

Use this command of the menu 'Project' 'Data Base Link' to define whether the object, which is currently marked in the Object organizer, should be kept in the data base or just locally in the project. A dialog will open, where you can choose one of the two data base categories 'Project' or 'Shared objects', or the category 'Local'.
The icons of all objects which are managed in the data base will be displayed grey-shaded in the Object organizer.

'Project' 'Data base command' 'Get latest version'

Use this command of the menu 'Project' 'Data Base Link' to copy the current version of the object, which is marked in the Object organizer, from the data base to your project. This will overwrite the local version. In contrast to the 'Check Out' action the object will not be locked for other users in the data base.

'Project' 'Data base command' 'Check Out'

Use this command of the menu 'Project' 'Data Base Link' to check out the object, which is marked in the Object organizer, from the data base and by that to lock it for other users.

When executing the command the user will get a dialog '**Check out object**'. A comment can be added there which will be stored in the version history of the object in the data base.

After the dialog has been closed with OK the checked-out object will be marked with a green check in the object organizer of the local project. For other users it will be appear marked with a red cross and will not be editable by them.

'Project' 'Data base command' 'Check In'

Use this command of the menu 'Project' 'Data Base Link' to check in the object, which is marked in the Object organizer, to the data base. Thereby a new version of the object will be created in the data base. The old versions will be kept anyway.

When executing the command the user will get a dialog 'Check in object'. There a comment can be added which will be stored in the version history of the object in the data base.

After the dialog has been closed with OK the green check in front of the object name in the Object organizer will be removed.

'Project' 'Data base command' 'Undo Check Out'

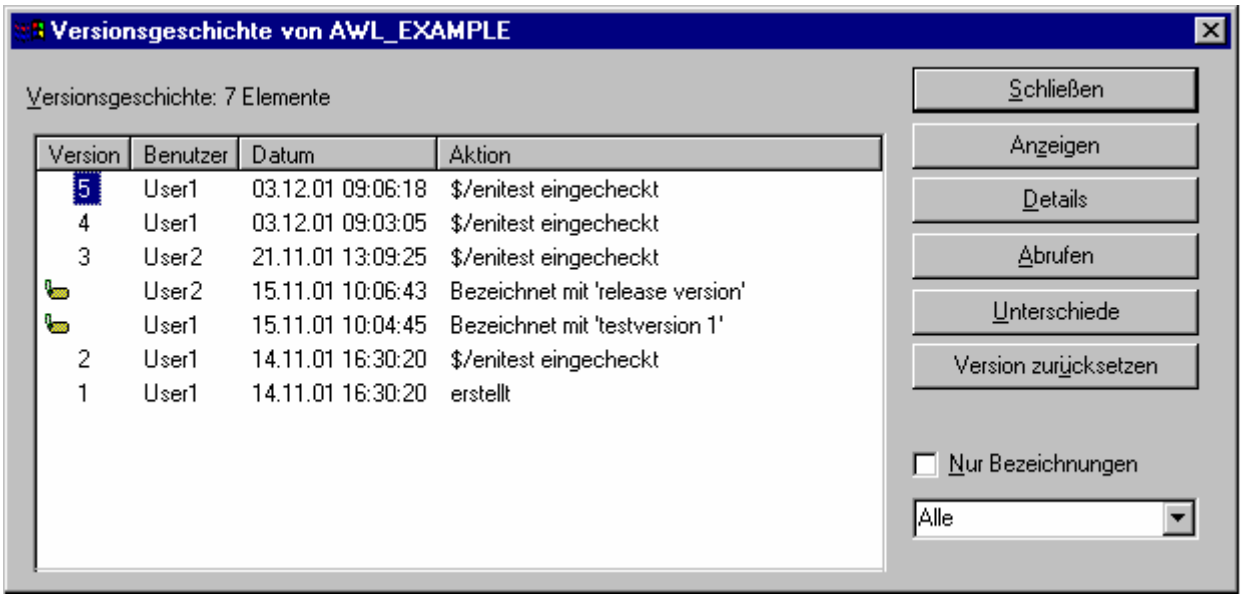
Use this command of the menu 'Project' 'Data Base Link' to cancel the Checking out of the object which is currently marked in the Object organizer. Thereby also the modifications of the object which have been made locally, will be canceled. No dialog will appear. The unchanged last version of the object will be kept in the data base and it will be accessible again for other users. The red cross in front of the object name in the Object organizer will disappear.

'Project' 'Data base command' 'Show Differences'

Use this command of the menu 'Project' 'Data Base Link' to display the object, which is currently opened by the user, in a window which is divided up in two parts. The differences of the versions will be marked like described for the project comparison.

'Project' 'Data base command' 'Show Version History'

Use this command of the menu 'Project' 'Data Base Link' to open a dialog 'Version history of <object name> for the object which is currently marked in the Object Organizer. There all versions of the object are listed which have been checked in to the data base or which have been labeled there:



Version: Data base specific numbering of the versions of the object which have been checked in one after the other. Labeled versions get no version number but are marked by a label-icon.

User: Name of the user, who has executed the check-in or labeling action

Date: Date and time stamp of the action

Action: Type of the action which has been executed. Possible types: 'created' (the object has been checked in to the data base for the first time), 'checked in' (all check-in's of the object excluding the first one) and 'labeled with <label>' (a label has been assigned to this version of the object)

The buttons:

Close: The dialog will be closed.

Display: The version which is currently marked in the table will be opened in a window. The title bar shows: "ENI: <name of the project in the data base>/<object name>

Details: The dialog '**Details of Version History**' will open:

File (name of the project and the object in the data base), **Version** (see above), **Date** (see above), **User** (see above),
Comment (Comment which has been inserted when the object has been checked in resp. has been labeled). Use the buttons **Next** resp. **Previous** to jump to the details window of the next or previous entry in the table in dialog 'Version history of ..'.

Get latest version: The version which is marked in the table will be loaded in TwinCAT PLC Control and there will overwrite the local version.

Differences: If in the table only one version of an object is marked, then this command will cause a comparison of this version with the latest (actual) data base version. If two versions are marked, then those will be compared. The differences are displayed in a bipartited window like it is done at the project comparison.

Reset version: The version which is marked in the table will be set as latest version. All versions which have been checked in later will be deleted ! This can be useful to restore an earlier status of an object.

Labels only: If this option is activated, then only those versions of the object will be displayed in the table, which are marked by a label.

Selection box below the option 'Labels only': Here you find the names of all users which have executed any data base actions for objects of the current project. Select 'All' or one of the names if you want to get the version history concerning all users or just for a certain one.

'Project' 'Data base command' 'Multiple Define'

Use this command of the menu 'Project' 'Data Base Link' to assign several objects at a single blow to a certain data base category. The dialog **'Properties**' will open like described for command **'Define'.** Choose the desired category and close the dialog with OK. After that the dialog **'ENI-Selection**' will open, listing all POUs of the project which are considered for the chosen category (Example: if you choose category 'shared objects' then the selection window will only offer the POUs of the Ressources tab). The POUs are presented in a tree structure complying to that of the Object Organizer. Select the desired POUs and confirm with OK.

'Project' 'Data base command' 'Get All Latest Versions'

Use this command of the menu 'Project' 'Data Base Link' to call the latest version of each object of the currently opened project, which is kept under source control, from the data base. Consider the following:
- If in the meantime additional objects have been stored to the data base project folder, then those will now be added to the local project in PLC Control.
- If objects have been deleted in the data base in the meantime, those will not be deleted in the local project, but they will automatically get assigned to category 'Local'.
- The latest version of objects of category 'Shared Objects' will only be called, if these objects are already available in the local project. For further information see command 'Get latest version'.

'Project' 'Data base command' 'Multiple Check Out'

Use this command of the menu 'Project' 'Data Base Link' to check out several objects at a single blow. For this the dialog '**ENI-Selection'** will open, listing all POUs of the project. Select those which should be checked out and confirm with OK. For further information see command 'Check Out'.

'Project' 'Data base command' 'Multiple Check In'

Use this command of the menu 'Project' 'Data Base Link' to check in several objects at a single blow. For this the dialog **'ENI-Selection**' will open, listing all POUs of the project. Select those which should be checked in and confirm with OK. For further information see command 'Check In'.

'Project' 'Data base command' 'Project Version History'

Use this command of the menu 'Project' 'Data Base Link' to view the version history for the currently opened project. But this will only work, if the chosen data base system supports that functionality. The dialog 'History of <data base project name>' will open. It shows the actions (create, check in, label) which have been performed for the particular objects of the project in a chronological order. The total number of objects is displayed behind Version history. The dialog can be handled like described for command **'Show Version History**', but regard the following:
- The command 'Reset Version' is only available for single objects.
- The command 'Get latest version' will call all objects of the version of the currently marked entry to the local project ! That means, that the objects in TwinCAT PLC Control will be overwritten with the older version. But: Local objects, which were not yet part of the project in that older version, will not be removed from the local project !

'Project' 'Data base command' Procect Label Version'

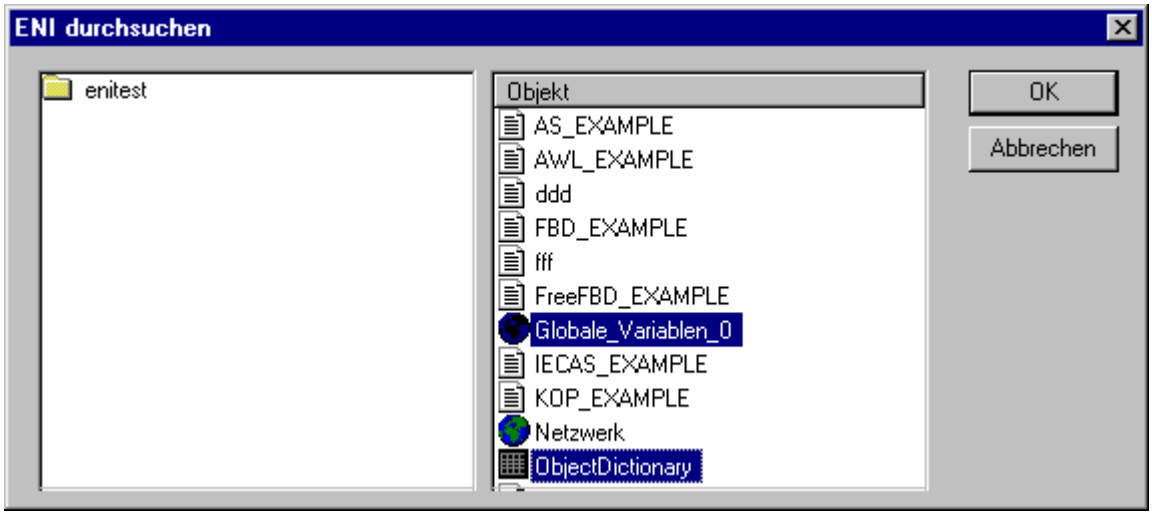
Use this command of the menu 'Project' 'Data Base Link' to put a "label" on the actual version of each object of a project, so that exactly this project version can be recalled later. A dialog 'Label <data base project name>' will open. Insert a **Label** name and optionally a **Comment.** When you confirm with OK, the dialog will close and the label and the action "labeled with <label name>" will appear in the table of the version history,

as well in the history for a single object as in the history of the project. A labeled version of the project does not get a version number, but is just marked with a label icon in the column 'Version'. If the option 'Labels only' is activated in the Version History dialog, then only labeled versions will be listed.



'Project' 'Data base command' 'Add Shared Objects'

Use this command of the menu 'Project' 'Data Base Link' if you explicitely want to add new objects of data base category 'Shared Objects' to the locally opened project. For objects of category 'Project Objects' this is not necessary, because the command 'Get (all) latest version(s)' automatically calls all objects which are found in the data base project folder, even if there are some which not yet available in the local project. But for objects of category 'Shared Objects' in this case just those objects will be called which are already available in the local project.
So execute the command 'Add Shared Objects' to open the dialog **'Browse ENI'**. A list in the right part of the window shows all objects which are available in the data base folder which is currently selected in the list on the left side. Choose the desired object and press OK or do a doubleclick on the entry to insert the object to the currently opened project.



'Project' 'Data base command' 'Refresh Status'

Use this command of the menu 'Project' 'Data Base Link' to update the display in the Object Organizer, so that you can see the actual status of the objects concerning the source control of the project.

'Project' 'Data base command' 'Login'

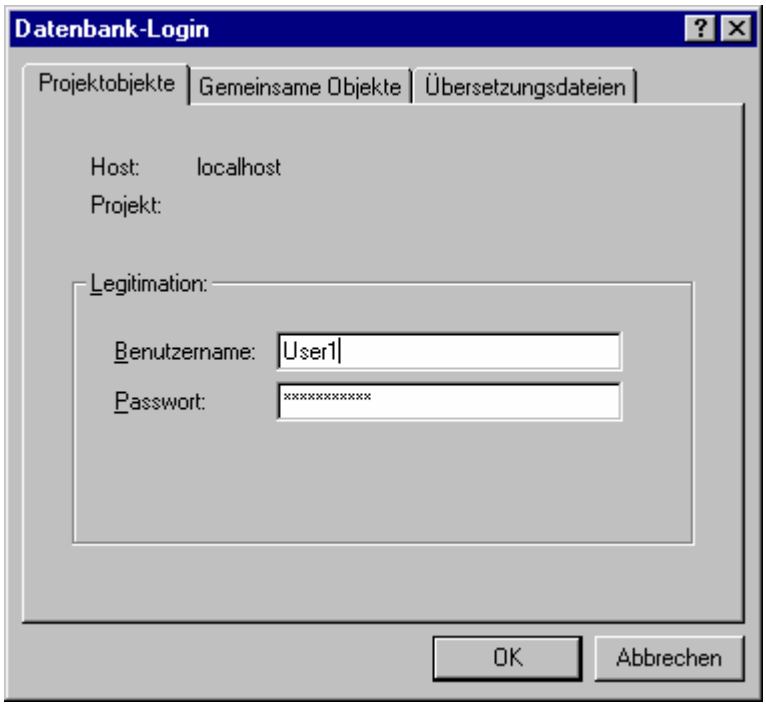
Use this command of the menu 'Project' 'Data Base Link' to open the dialog 'Login' where you can enter the access data for the ENI data base via the ENI Server. The access data also have to be defined in the ENI Server (ENI Admin, User Management) and - depending on the currently used data base - also in the user management of the data base. After the command has been executed, first the Login dialog for category 'Project objects' will open.
The following items are displayed:

Data base: project objects

Host: address of the computer where the ENI Server is running (must match with the entry in field 'TCP/IP address' in the project options dialog for 'Project source control').

Project: Name of the data base project (must match with the entry in field 'Project name' in the project options dialog for 'Project source control'/category 'Project Objects').

Credentials: Insert **User name** and **Password**.



Press OK to confirm the settings. The dialog will be closed and automatically the Login dialog for 'Shared objects' will open

Enter the access data in the same way as described for the 'Project objects' and confirm with OK.

'Project' 'Add action '

This command is used to generate an action allocated to a selected block in the Object Organizer. One selects the name of the action in the dialog which appears and also the language in which the action should be implemented.
The new action is placed under your block in the Object Organiser. A plus sign appears in front of the block. A simple mouse click on the plus sign causes the action objects to appear and a minus sign appears in front of the block. Renewed clicking on the minus sign causes the actions to disappear and the plus sign appears again. This can also be achieved over the context menu commands ‘**Expand Node**’ and ‘**Collapse Node**’.

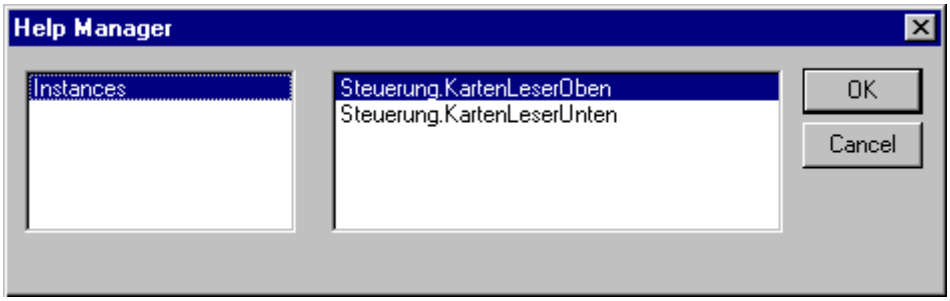
With a double click at the action or pressing <Return>, an action will be loaded in your editor.

'Project' 'Open Instance'

With this command it is possible to open and display the instance of the function block which is selected in the Object Organizer. In the same manner, a double click on the function block in the Object Organizer gives access to a selection dialog in which the instances of the function block as well as the implementation are listed. Select here the desired instance or the implementation and confirm using **OK**. The desired item is then displayed in a window.



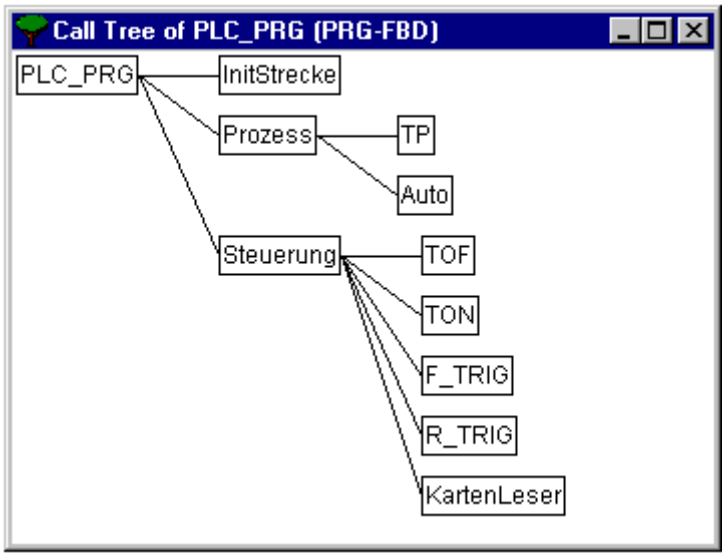
If you want to view instances, you first have to log in ! (The project has been compiled with no errors and downloaded to the PLC with ‘Online’ ’Login’).



Dialog box for opening an instance

'Project' 'Show call tree'

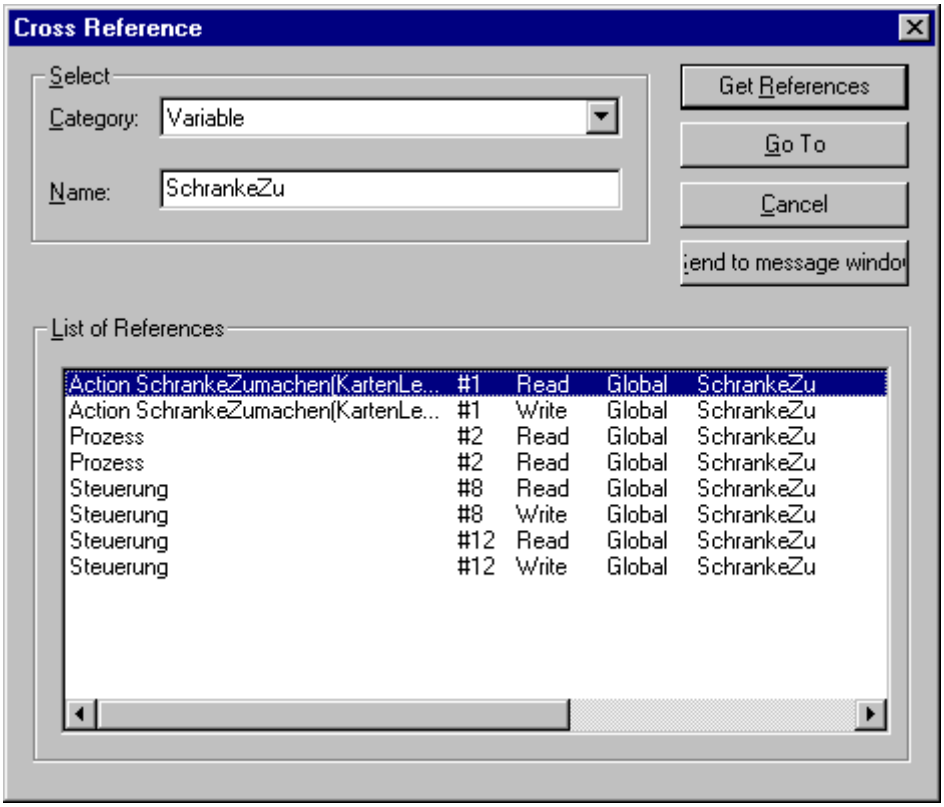
With this command you open a window which shows the call tree of the object chosen in the Object Organizer. For this the project must be compiled (see "Rebuild all"). The call tree contains both calls for POUs and references to data types.



Example of a call tree

'Project' 'Show cross reference list'

With this command you open a dialog box which makes possible the output of all application points for a variable, address, or a POU. For this the project must be compiled (see "Rebuild all"). Choose first the category Variable, Address, or POU and then enter the name of the desired element. To obtain all elements of the entered category enter a “*” in Name.
By clicking on the button **Get References** you get the list of all application points. Along with the POU and the line or network number, the variable name and the address binding, if any, are specified. The Domain space shows whether this is a local or a global variable; the Access column shows whether the variable is to be accessed for ‚reading’ or ‚writing’ at the current location.
When you select a line of the cross reference list and press the button **Go To** or doubleclick on the line, then the POU is shown in its editor at the corresponding point. In this way you can jump to all application points without a time-consuming search.
In order to make processing easier, you can use the **Send to message window** button to bring the current cross reference list into the message window and from there change to the respective POU.



Dialog box and example of a cross reference list

4.5 Editing Functions

You can use the following commands in all editors and some of them in the Object Organizer. All of the commands are located under the menu item "Edit".

'Edit' 'Undo' Shortcut: <Ctrl>+<Z>

This command undoes the action which was most recently executed in the currently-open editor window or in the Object Organizer. By repeatedly selecting this command, all actions can be undone back to the point at which the window was opened. This applies to all actions in the editors for POUs, data types and global variables and in the Object Organizer.
With "**Edit" "Redo**" you can restore an action which you have undone.

'Edit' 'Redo' Shortcut: <Ctrl>+<Y>

With the command in the currently-open editor window or in the Object Organizer you can restore an action you have undone ("Edit" "Undo") .As often as you have previously executed the command "Undo" , you can also carry out the command "Redo". The commands "Undo" and "Redo" apply to the current window.

The commands "Undo" and "Redo" apply to the current window. Each window carries its own action list. If you want to undo actions in several windows, then you must activate the corresponding window. When undoing or redoing in the Object Organizer the focus must lie here.

'Edit' 'Cut' Shortcut: <Ctrl>+<X> or <Shift>+<Del>

This command transfers the current selection from the editor to the clipboard. The selection is removed from the editor. In the Object Organizer this similarly applies to the selected object, whereby not all objects can be deleted, e.g. the PLC configuration. Remember that not all editors support the cut command, and that its use can be limited in some editors. The form of the selection depends upon the respective editor: In the text editors (IL, ST, and declarations) the selection is a list of characters. In the FBD and LD editors the choice is a number of networks which are indicated by a dotted rectangle in the network number field or a box with all preceding lines, boxes, and operands. In the SFC editor the selection is a part of a series of steps surrounded by a dotted rectangle.

In order to paste the content of the clipboard you use the command "**Edit" "Paste**". In the SFC editor you can also use the commands "**Extras" "Insert parallel branch (right)**" or "**Extras" "Paste after**".
In order to copy a selection onto the clipboard without deleting it, use the command **"Edit" "Copy"**.
In order to remove a selected area without changing the clipboard, use the command **"Edit" "Delete"**.

'Edit' 'Copy' Shortcut: <Ctrl>+<C>

This command copies the current selection from the editor to the clipboard. This does not change the contents of the editor window. With the Object Organizer this similarly applies to the selected object, whereby not all objects can be copied, e.g. the PLC configuration. Remember that not all editors support copying and that it can be limited with some editors. For the type of selection the same rules apply as with **"Edit" "Cut**".

'Edit' 'Paste' Shortcut: <Ctrl>+<V>

Pastes the content of the clipboard onto the current position in the editor window. In the graphic editors the command can only be executed when a correct structure results from the insertion. With the Object Organizer the object is pasted from the clipboard. Remember that pasting is not supported by all editors and that its use can be limited in some editors. The current position can be defined differently according to the type of editor: With the text editors (IL, ST, Declarations) the current position is that of the blinking cursor (a vertical line) which you place by clicking with the mouse). In the FBD and LD editors the current position is the first network with a dotted rectangle in the network number area. The contents of the clipboard are inserted in front of this network. If a partial structure has been copied, then it is inserted in front of the selected element. In the SFC editor the current position is determined the selection which is surrounded by a dotted rectangle. Depending upon the selection and the contents of the clipboard, these contents are inserted either in front of the selection or into a new branch (parallel or alternative) to the left of the selection. In SFC the commands "**Extras" "Insert parallel branch (right)**" or "**Extras" "Paste after**" can be used in order to insert the contents of the clipboard.

'Edit' 'Delete' Shortcut: <Del>

Deletes the selected area from the editor window. This does not change the contents of the clipboard. In the Object Organizer this applies likewise to the selected object, whereby not all objects can be deleted, e.g. the PLC configuration. For the type of selection the same rules apply as with "**Edit" "Cut**". In the library manager the selection is the currently selected library name.

'Edit' 'Find'

With this command you search for a certain text passage in the current editor window. The Find dialog box opens. It remains opens until the button **Cancel** is pressed.
In the field **Find what** you can enter the series of characters you are looking for.
In addition, you can decide whether the text you are looking for **Match whole word** only or not, or also whether Match case is to be considered, and whether the search should proceed **Up or Down starting** from the cur-rent cursor position.
The button **Find next** starts the search which begins at the selected position and continues in the chosen search direction. If the text passages is found, then it is highlighted. If the passage is not found, then a message announces this. The search can be repeated several times in succession until the beginning or the end of the contents of the editor window has been reached.
Remember that the found text can be covered up by the Find dialog box.



Find dialog box

'Edit' 'Find Next' Shortcut: <F3>

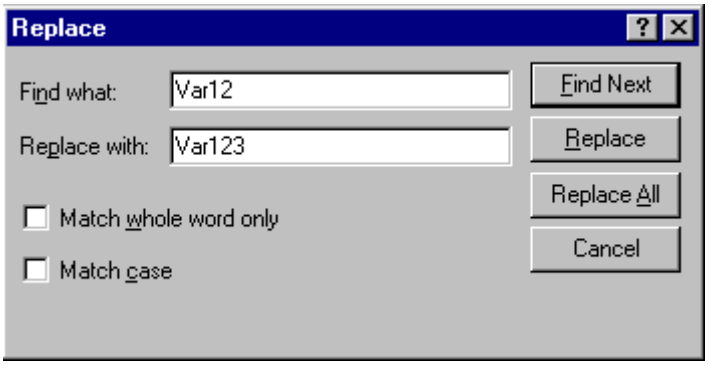
With this command you execute a search with the same parameters as with the most recent action **"Edit" "Find"**.

'Edit' 'Replace'

With this command you search for a certain passage just as with the command "**Edit" "Find**", and replace it with another. After you have chosen the command the dialog box for find and replace appears. This dialog box remains open until the button **Cancel** or **Close** is pressed.

The button **Replace** replaces the current selection with the text in the field **Replace with**. The button **Replace all** replaces every occurrence of the text in the field **Find next** after the current position with the text in the field **Replace with**. At the end of the procedure a message announces how many replacements were made.

At the end of the procedure a message announces how many replacements were made.



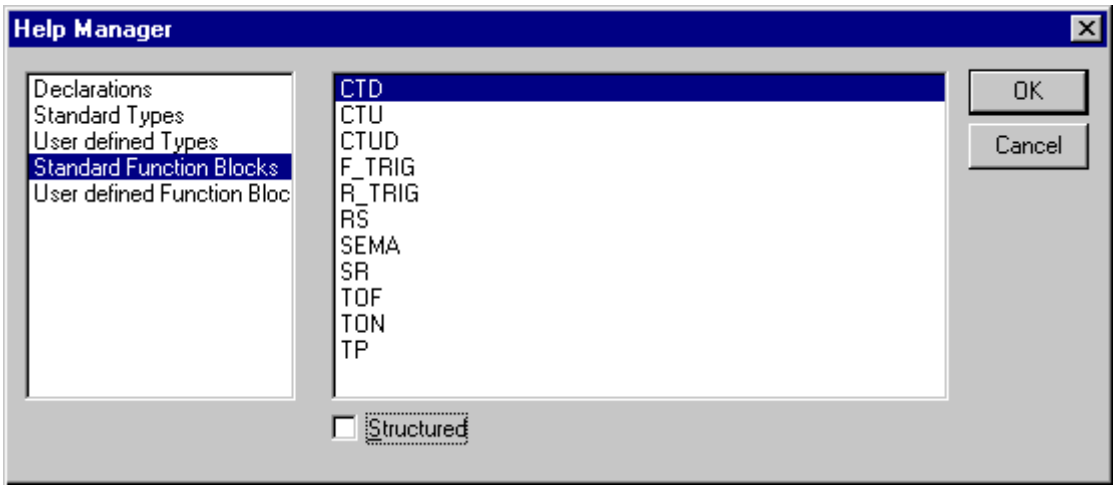
Dialog box for find and replace

'Edit' 'Input Assistant' Shortcut: <F2>

This command provides a dialog box for choosing possible inputs at the current cursor position in the editor window. In the left column choose the desired input category, select the desired entry in the right column, and confirm your choice with **OK.** This inserts your choice at this position.
The categories offered depend upon the current cursor position in the editor window, i.e. upon that which can be entered at this point (e.g. variables, operators, POUs, conversions, ...).

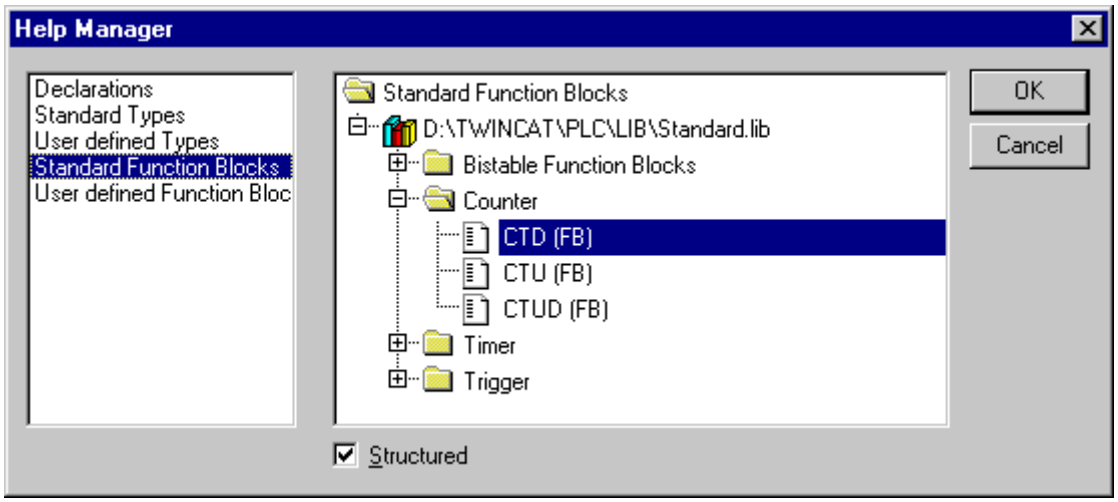
If the option **With arguments** is active, then when the selected element is inserted, the arguments to be transferred are specified with it, for example: function block fu1 selected, which defines the input variable var_in: fu1(var_in:=);
Insertion of function func1, which uses var1 and var2 as transfer parameters: func1(var1,var2)
It is basically possible to switch between structured and unstructured display of the available elements. This occurs through activation/deactivation of the Structured Display option.

- Unstructured Display:



The POUs, variables or data types in each category are simply sorted linearly in alphabetical order. At various places (e.g. in the Watch List), multi-stage variable names are required. In that event, the Input Assistant dialog displays a list of all POUs as well as a single point for the global variables. After each POU name there is a point. If a POU is selected by doubleclick or by pressing <**Enter**>, a list of the variables belonging to it opens. If instances and data types are present, it is possible to open further levels in the hierarchy display. OK transfers the selected variable.

- Structured Display:



If **Structured display** is selected, the POUs, variables or data types will be sorted hierarchically. This is possible for standard programs, standard functions, standard function blocks, defined programs, defined functions, defined function blocks, global variables, local variables, defined types, watch variables. The visual and hierarchical display corresponds to that of the Object Organizer; if elements in a library are referred to, these are inserted in alphabetical order at the very top and the pertinent hierarchy is displayed as in the Library Manager.

The in- and output variables of function blocks which are declared as local or global variables are listed in the category 'Local Variables' or 'Global Variables' under the instance name (e.g. Inst_TP ET, Inst_TP IN,...). To get there, select the instance name (e.g. Inst_TP) and confirm with **OK**.

If the instance of a function block is selected here, the option **With arguments** may be selected. In the text languages ST and IL as well as during task configuration, the instance name and the input parameters of the function block are then inserted.
For example, if Inst (DeklarationInst: TON;) is selected, the following is inserted:

```
Inst(IN:= ,PT:=)
```

If the option is not selected, only the instance name will be inserted. In the graphical languages or in the Watch window, only the instance name is generally inserted.
Components of structures are displayed in an analog fashion to function block instances.
For enumerations, the individual enumeration values are listed under the enumeration type. The order is: enumerations from libraries, enumerations from data types, local enumerations from POUs.

The general rule is that lines containing sub-objects are not selectable (except instances, see above), but can only have their hierarchy display expanded or contracted by one level, as for multi-stage variable names. If Input Assistant is invoked in the Watch and Recipe Manager or in the selection of trace variables in the trace configuration dialog, it is possible to make a multiple selection. When the <Shift> key is pressed, you can select a range of variables; when the <Ctrl> key is pressed you can select many individual variables. The selected variables are so marked. If, during range selection lines are selected that do not contain valid variables (e.g. POU names), these lines will not be included in the selection. When individual selections are made, such lines can not be marked.

In the watch window and in trace configuration it is possible to transfer structures, arrays or instances from the Input Assistant dialog. As a double click with the mouse button is associated with the extension or contraction of the element's hierarchy display, selection in these cases can only be confirmed by **OK**. Thereafter, the selected variables are inserted line by line in the watch window, that is each selected variable is written on a separate line. In the case of trace variables, each variable is inserted in a separate line of the trace variables list.
If the maximum number of trace variables, 20, is exceeded during insertion of the selected variables, the error message „A maximum of 20 variables is allowed“ appears. Further selected variables are then not inserted in the list.



Some entries (e.g. Global Variables) are only updated in the Input Assistant dialog after compilation.

'Edit' 'Declare Variable' Shortcut: <Shift>+<F2>

This command opens the dialog for the declaration of a variable. This dialog also opens automatically when the option 'Project' 'Options' 'Editor' 'Autodeclaration' is switched on and when a new undefined variable is used the declaration editor.

'Edit' 'Next Error' Shortcut: <F4>

After the incorrect compilation of a project this command can show the next error. The corresponding editor window is activated and the incorrect place is selected. At the same time in the message window the corresponding error message is shown.

'Edit' 'Next Error' Shortcut: <Shift>+<F4>

After the incorrect compilation of a project this command shows the previous error. The corresponding editor window is activated and the incorrect place is selected. At the same time in the message window the corresponding error message is shown.

'Edit' 'Macros'

This menu item leads to a list of all macros, which are defined for the project. (For info on generating macros see 'Project' 'Options' 'Macros'). When an executable macro is selected the dialog 'Process Macro'. The name of the macro and the currently active command line are displayed. The button **Cancel** can be used to stop the processing of the macro. In that event the processing of the current command will be finished anyway. Then an appropriate message is displayed in the message window and in the log during Online operation: "<Macro>: Execution interrupted by user“.

Macros can be executed offline and online, but in each case only those commandes are executed which are available in the respective mode.

4.6 Online Functions

The available online commands are assembled under the menu item "Online". The execution of some of the commands depends upon the active editor. The online commands become available only after logging in. Thanks to 'Online Change' functionality you have the possibility of making changes to programs on the running controller. See in this connection **'Online' 'Log-in'**

'Online' 'Login' Shortcut: <F11>

This command combines the programming system with the PLC (or starts the simulation program) and changes into the online mode.
If the current project has not been compiled since opening or since the last modification, then it is compiled now (as with "Project" "Build"). If errors occur during compilation, then TwinCAT PLC Control does not change into Online mode.
If the current project was changed on the controller since the last download, but not closed, and if the last download information was not deleted with the command '**Project' 'Clear all',** then after the command 'Login' a dialog opens with the question: „The program has been changed. Load changes? (Online Change)“. By answering **Yes** you confirm that, on log-in, the modified portions of the project are to be loaded onto the controller. **No** results in a log-in without the changes made since the last download being loaded onto the controller. **Cancel** cancels the command. <**Load all**> causes the entire project to be reloaded onto the controller.

After a successful login all online functions are available (if the corresponding settings in 'Project' '**Options**' category 'Build' have been entered).
Use the **"Online" "Logout"** command to change from online back to offline mode.

Error case

Error: "A connection to the PLC could not be established"

Check if your TwinCAT System runs (TwinCAT Icon in Taskbar is green). If not started, start the TwinCAT system by right mouse click on the icon then select system and start. The colour of the TwinCAT icon changes from red to green.

Error: "The program has been modified! Should the new program be loaded?"
The project which is open in the editor is incompatible with the program currently found in the PLC. Monitoring and debugging is therefore not possible. You can either choose "No," logout, and open the right project, or use "Yes" to load the current project in the PLC.

Message: „The program has been changed. Load changes? (ONLINE CHANGE)“.
The project is running on the controller. The target system supports 'Online Change' and the project has been altered on the controller with respect to the most recent download or the most recent Online Change. You may now decide whether these changes should be loaded with the controller program running or whether the command should be cancelled. You can also, however, load the entire compiled code by selecting the Load all button.

'Online' 'Logout' Shortcut: <F12>

The connection to the PLC is broken. Use the **"Online" "Login"** command to change to the online mode.

'Online' 'Download'

This command loads the compiled project in the PLC (download, not to mistake with **'Online''download source code'**!).
Download information is saved in a file called <projectname>0000000ar.ri , which is used during Online Change to compare the current program with the one most recently loaded onto the controller, so that only changed program components are reloaded. This file is erased by the command **'Project' 'Clear all'.**

'Online' 'Run' Shortcut: <F5>

This command starts the program in the PLC or in Simulation Mode. This command can be executed immediately after the **"Online" "Download"** command, or after the user program in the PLC has been ended with the **"Online" "Stop"** command, or when the user program is at a break point, or when the **'Online' 'Single Cycle'** command has been executed.

'Online' 'Stop' Shortcut: <Shift>+>F8>

Stops the execution of the program in the PLC or in Simulation Mode between two cycles. Use the **"Online" "Run"** command to restart the program.

'Online' 'Reset'

This command resets - with exception of the retain variables (VAR RETAIN) - all variables to that specific value, with which they have got initialized (also those variables which have been declared as VAR PERSISTENT !). If you have initialized the variables with a specific value, then this command will reset the variables to the initialized value. All other variables are set at a standard initialization (for example, integers at 0). As a precautionary measure, TwinCAT PLC Control asks you to confirm your decision before all of the variables are overwritten. The situation is that which occurs in the event of a power failure or by turning the controller off, then on (warm restart) while the program is running. Use the **'Online' 'Run'** command to restart the program.

‘Online’ ‘Reset All’

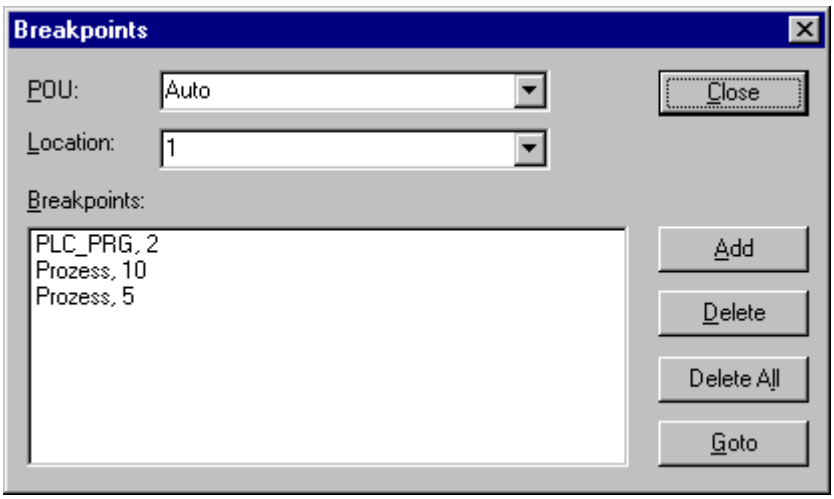
This command resets all variables including the persistent ones (PERSISTENT) to their initialization values and erases the user program on the controller. The controller is returned to its original state.

'Online' 'Toggle Breakpoint' Shortcut: <F9>

This command sets a breakpoint in the present position in the active window. If a breakpoint has already been set in the present position, that breakpoint will be removed. The position at which a breakpoint can be set depends on the language in which the POU in the active window is written. In the Text Editors (IL, ST), the breakpoint is set at the line where the cursor is located, if this line is a breakpoint position (recognizable by the dark-gray color of the line number field). You can also click on the line number field to set or remove a breakpoint in the text editors. In FBD and LD, the breakpoint is set at the currently selected network. In order to set or remove a breakpoint in the FBD or LD Editor, you can also click on the network number field. In SFC, the breakpoint is set at the currently selected step. In SFC you can also use <Shift> with a doubleclick to set or remove a breakpoint. If a breakpoint has been set, then the line number field or the network number field or the step will be displayed with a light-blue background color. If a breakpoint is reached while the program is running, the program will stop, and the corresponding field will be displayed in a red background color. In order to continue the program, use the **"Online" "Run"**, **"Online" "Step in"**, or **"Online" "Step Over"** commands.
You can also use the Breakpoint dialog box to set or remove breakpoints.

'Online' 'Breakpoint Dialogbox'

This command opens a dialog box to edit breakpoints throughout the entire project. The dialog box also displays all breakpoints presently set.
In order to set a breakpoint, choose a POU in the POU combobox and the line or the network in the Location combobox where you would like to set the breakpoint; then press the Add button. The breakpoint will be added to the list.
In order to delete a breakpoint, highlight the breakpoint to be deleted from the list of the set breakpoints and press the **Delete** button. The **Delete All** button can be used to delete all the breakpoints.
In order to go to the location in the editor where a certain breakpoint was set, highlight the respective breakpoint from the list of set breakpoints and press the **Go to** button.
To set or delete breakpoints, you can also use the **'Online' 'Toggle Breakpoint'** command.



Breakpoint Editing Dialog Box

'Online' 'Step Over' Shortcut: <F10>

This command causes a single step to execute. If a POU is called, the pro-gram stops after its execution. In SFC a complete action is executed. If the present instruction is the call-up of a function or of a function block, then the function or function block will be executed completely. Use the **"Online" "Step In"** command, in order to move to the first instruction of a called function or function block. If the last instruction has been reached, then the program will go on to the next instruction in the POU.

'Online' 'Step In' Shortcut: <F8>

A single step is executed. The program is stopped before the first instruction of a called POU. If necessary, there will be a changeover to an open POU. If the present position is a call-up of a function or of a function block , then the command will proceed on to the first instruction in the called POU. In all other situations, the command will function exactly as **"Online" "Step Over"**.

'Online' 'Single Cycle' Shortcut: <Ctrl>+<F5>

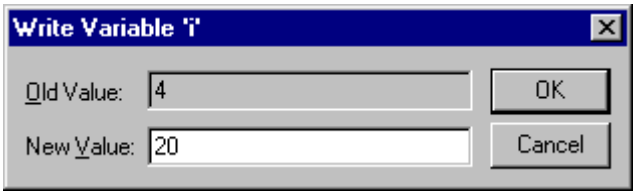
This command executes a single PLC Cycle and stops after this cycle. This command can be repeated continuously in order to proceed in single cycles. The Single Cycle ends when the **"Online" "Run"** command is executed.

'Online' 'Write Values' Shortcut: <Ctrl>+<F7>

With this command, one or more variables are set – one time only! – to user defined values at the beginning of a cycle.
The values of all single-element variables can be changed, so long as they are also visible in Monitoring.
Before the command 'Write values' can be executed, a variable value must be ready to be written.

1. Define the values

- For non-boolean variables a double mouse click in performed on the line in which a variable is declared, or the variable is marked and the <Enter> key is pressed. The dialog box 'Write variable <x>' then appears, in which the value to be written to the variable can be entered.



Dialog Box for Writing a New Variable Value.

- For boolean variables, the value is toggled (switched between TRUE and FALSE, with no other value allowed) by double-clicking on the line in which the variable is declared; no dialog appears.

The value set for Writing is displayed in brackets and in turquoise colour behind the former value of the variable. e.g. a=0 <:=34>.

Exception: In the FBD and LD Editor the value is shown turquoise without brackets next to the variable name.

Set the values for as many variables as you like.
The values entered to be written to variables can also be corrected or deleted in the same manner. This is likewise possible in the 'Online' 'Write/Force dialog' (see below).
The values to be written that were previously noticed are saved in a **writelist (Watchlist)**, where they remain until they are actually written, deleted or transferred to a forcelist by the command 'Force values'.

2. Write the values

The command to Write Values can be found at two places:

- Command 'Write Values' in the menu 'Online'.
- Button 'Write Values' in the dialog 'Editing the writelist and the forcelist'.

When the command 'Write values' is executed, all the values contained in the writelist are written, once only , to the appropriate variables in the controller at the beginning of the cycle, then deleted from the writelist. (If the command 'Force values' is executed, the variables in question are also deleted from the writelist, and transferred to the forcelist!)



In the sequential function chart language (SFC), the individual values from which a transition expression is assembled cannot be changed with 'Write values'. This is due to the fact that in monitoring the 'Total value' of the expression, not the values of the individual variables are displayed (e.g. "a AND b" is only displayed as TRUE if both variables actually have the value TRUE).
In FBD, on the other hand, only the first variable in an expression, used for example as input to a function block, is monitored. Thus a 'Write values' command is only possible for this variable.

'Online' 'Force values' Shortcut: <F7>

With this command, one or more variables are permanently set to user-defined values. The setting occurs in the run-time system, both at the beginning and at the end of the cycle.
The time sequence in one cycle: 1.Read inputs, 2. Force values 3. Process code, 4. Force values 5. Write outputs.
The function remains active until it is explicitly suspended by the user (command 'Online' 'Release force') or the programming system is logged-out.
For setting the new values, a **writelist** is first created, just as described under 'Online' 'Write values'. The variables contained in the writelist are accordingly marked in Monitoring. The writelist is transferred to a **forcelist** as soon as the command **'Online' 'Force values'** is executed. It is possible that an active forcelist already exists, in which case it is updated as required. The writelist is then emptied and the new values displayed in red as 'forced'. Modifications of the forcelist will be transferred to the program with the next 'Force values' command.

The forcelist is created at the first forcing of the variables contained in the writelist, while the writelist existed **prior** to the first writing of the variables that it contains.

The command for forcing a variable, which means that it will be entered into the forcelist can be found at the following places:

- Command 'Force Values' in the menu ‘Online'.
- Button 'Force Values' in the dialog 'Editing the writelist and the forcelist'.

In the sequential function chart language, the individual values from which a transition expression is assembled cannot be changed with 'Force values'. This is due to the fact that in monitoring the 'Total value' of the expression, not the values of the individual variables are displayed (e.g. "a AND b" is only displayed as TRUE if both variables actually have the value TRUE).
In FBD, on the other hand, only the first variable in an expression, used for example as input to a function block, is monitored. Thus a 'Force values' command is only possible for this variable.

'Online' 'Release Force' Shortcut: <Shift>+<F7>

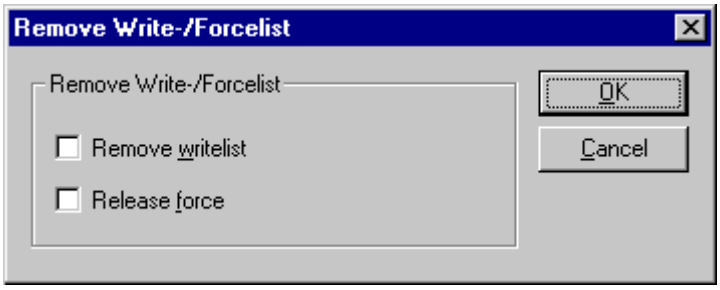
This command ends the forcing of variable values in the controller. The variable values change again in the normal way.
Forced variables can be recognized in Monitoring by the red color in which their values are displayed. You can delete the whole forcelist, but you can also mark single variables for which the forcing should be released.
To delete the whole forcelist, which means to release force for all variables, choose one of the following ways:

- Command 'Release Force' in menu 'Online'.
- Button 'Release Force' in dialog 'Editing the writelist and the forcelist'
- Delete the whole forcelist using the command ‘Release Force’ in the dialog ‘Remove Write-/Forcelist’. This dialog opens if you choose the command ‘Release Force’ while also a writelist exists.

To release force only **for single variables** you have to mark these variable first. Do this in one ways described in the following. After that the chosen variables are marked with an turquoise extension **<Release Force>**:

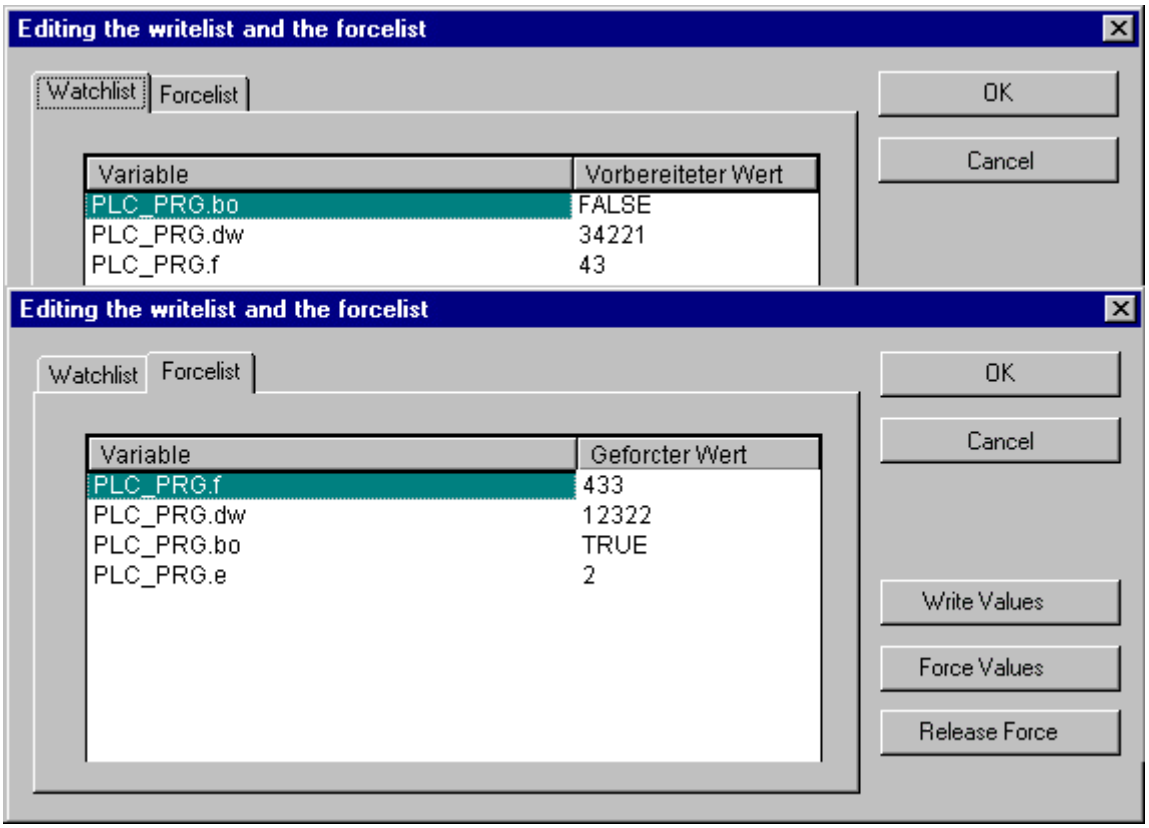
- A double mouse click on a line, in which a non boolean variable is declared, opens the dialog ‘Write variable <x>’. Press button <Release Force for this variable> .
- Repeat double mouse clicks on a line in which a boolean variable is declared to toggle to the display <Release Force> at the end of the line.
- In the menu ‘Online’ open the Write/Force-Dialog and delete the value in the edit field of the column ‘Forced value’.

When for all desired variables the setting “<Release Force>” is shown in the declaration window, choose the command **‘Force’** to transfer the modifications of the forcelist to the program.
If the current writelist (see 'Online' 'Write Values') is not empty while you execute the command ‘Release Force’, the dialog 'Remove Write-/Forcelist’ will be opened. There the user has to decide whether he just wants to **Release Force** or additionally wants to **Remove the writelist** or if he wants to remove **both** lists.



'Online'Write/Forcen Dialog'

This command leads to a dialog which displays in two registers the current writelist (Watchlist) and forcelist (Forcelist). Each variable name and the value to be written to it or forced on it are displayed in a table.



The variables reach the watchlist via the commands **'Online' 'Write Values'** and are transferred to the forcelist by the command **'Online' 'Force Values'.** The values can be edited here in the „Prepared Value“ or „Forced Value“ columns by clicking the mouse on an entry to open an editor field. If the entry is not type-consistent, an error message is displayed. If a value is deleted, it means that the entry is deleted from the writelist or the variable is noticed for suspension of forcing as soon as the dialog is closed with any other command than **Cancel**.
The following commands, corresponding to those in the Online menu, are available via buttons:

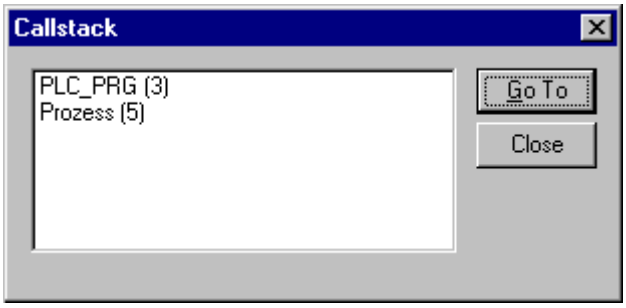
Force Values: All entries in the current writelist are transferred to the forcelist, that is the values of the variables in the controller are forced. All variables marked with 'Release Force' are no longer forced. The dialog is then closed.

Write Values: All entries in the current writelist are written once only to the corresponding variables in the controller. The dialog is then closed.

Release Force: All entries in the forcelist will be deleted or, if a writelist is present, the dialog “Delete write-/forcelist” comes up, in which the user must decide whether he only wants to **release forcing** or **discard the writelist**, or both. The dialog will close at that point, or after the selection dialog is closed as the case may be.

'Online' 'Show Call Stack'

You can run this command when the PLC stops at a breakpoint. You will be given a dialog box with a list of the POUs currently in the Call Stack



Example of a Call Stack

The first POU is always the in the appointed debug task called program, because this is where the executing begins.
The last POU is always the POU being executed. After you have selected a POU and have pressed the **Go to** button, the selected POU is loaded in its editor, and it will display the line or network being processed.

'Online' 'Flow Control'

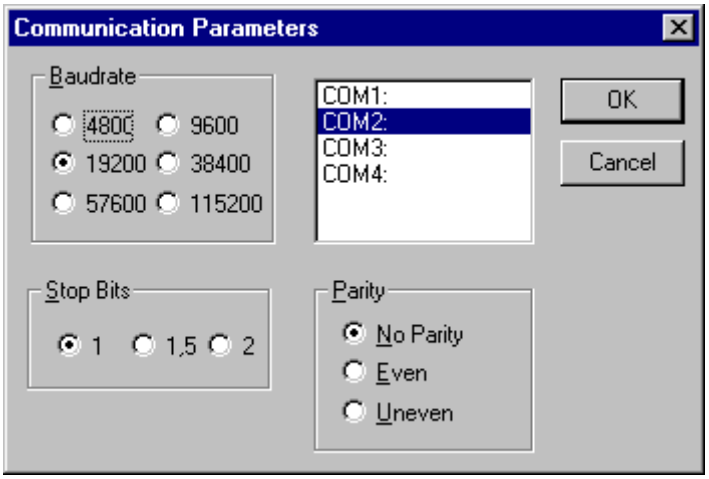
If you have selected the **flow control,** then a check will appear in front of the menu item. Following this, every line or every network will be marked which was executed in the last PLC Cycle. The line number field or the network number field of the lines or networks which just run will be displayed in green. An additional field is added in the IL-Editor in which the present contents of the accumulator are displayed. In the graphic editors for the Function Block Diagram and Ladder Diagram, an additional field will be inserted in all connecting lines not transporting any Boolean values. When these Out- and Inputs are verified, then the value that is transported over the connecting line will be shown in this field. Connecting lines that transport only Boolean values will be shaded blue when they transport TRUE. This enables constant monitoring of the information flow.

'Online' 'Simulation'

Simulation Mode is only accessibly for Buscontroller BC and not for TwinCAT PC. If Simulation Mode is chosen, then a check will appear in front of the menu item. In the simulation mode, the user program runs on the same PC under Windows. This mode is used to test the project. The communication between the PC and Simulation Mode uses the Windows Message mechanism. If the program is not in simulation mode, then the program will run on the PLC. The status of this flag is stored with the project.

'Online' 'Communication Parameters'

The parameters for transferring through the serial interface can be entered in a dialog box. It is important that these parameters agree with those entered in the PLC .

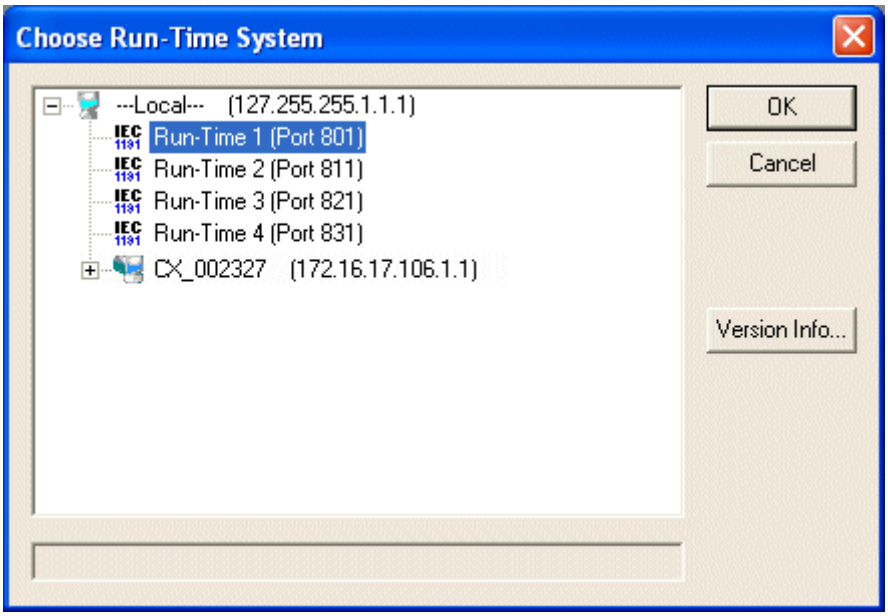


Dialog box for Entering Communication Parameters
Possible adjustments include: the baudrate; whether the transfer should be made with Even, Odd, or No Parity; the number of Stop Bits; and also the interface (COM1, COM2, etc.) via which the transfer is to occur. The selected parameters are stored with the project.

'Online' 'Choose Runtime System'

Only TwinCAT PC:

You can choose a runtime system. You see the local (max. four) and remote accessible runtime systems.



The runtime system information will be stored in the PLC project file after saving. Therefore, the selected port number is visible then in the TwinCAT System Manager configuration after a rescan.

Online' 'Sourcecode download'

This command loads the source code for the project into the controller system. This is not to be confused with the Code that is created when the project is compiled! You can enter the options that apply to Download (time, size) in the **'Project' 'Options' 'Sourcedownload'** dialog.

‘Online’ 'Create bootproject’

With this command (done online), the compiled project is set up on the controller in such a way that the controller can load it automatically when restarted. Storage of the boot project occurs differently depending on the target system. For example a file is created on the PC in the TwinCAT boot directory: TCPLC_P_x.wbp (x is the number of the run time system 1 up 4).

‘Online’ 'Create bootproject (offline)’

With this command, the compiled project is set up in the project folder together with the *.pro file. Storage of the boot project occurs differently depending on the target system. For example a file is created on a PC with WinXP the name will be TCPLC_P_x.wbp (x is the number of the run time system 1 up 4). The controller can load this boot project, if the user copies the file into the TwinCAT Boot folder manually.

‘Online’ 'Write file to controller’

This command is used for loading any desired file onto the controller. It opens the dialog for 'Write file to controller' in which you can select the desired file. After the dialog is closed using the 'Open' button, the file is loaded into the controller and stored there under the same name. The loading process is accompanied by a progress dialog.
With the command **'Online' 'Load file from controller'** you can retrieve a file previously loaded on the controller.

‘Online’ 'Load file from controller’

With this command, you can retrieve a file previously loaded into the controller using **'Online' 'Write file to controller'**. You receive the 'Load file from controller' dialog. Under **Filename**, provide the name of the desired file, and in the selection window enter the directory on your computer into which it is to be loaded as soon as the dialog is closed with the **„Save“** button.

The following menue entries are enabled, if the target system is BCxxxx:

'Online' 'coupler'

The following special commands are available:

K-Bus Reset: A reset of the Terminal bus is performed

Coupler Reset: The coupler will be restarted

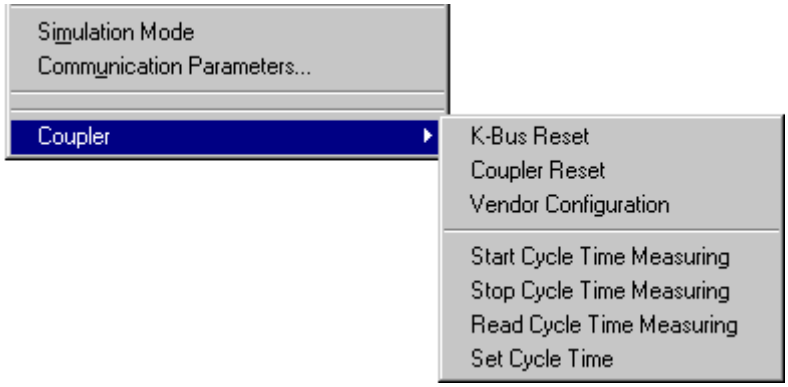
Vendor Configuration: The factory default settings will be loaded (Boot Project erased), the coupler must be subsequently restarted by initiating a coupler reset.

Start Cycle Time Measuring: The cycle-time measurement on the BCxxxx will be started.

Stop Cycle Time Measuring: The cycle-time measurement on the BCxxxx will be stopped

Read Cycle Time Measuring: he minimum cycle-time (since measurement start), maximum cycle-time, average cycle-time (taken from the last 200 cycles) , actual cycle-time and the number of PLC-cycles will be read.

Set Cycle Time: The target cycle-time and background task time can be set here. If the actual cycle-time exceeds the target cycle-time the next cycle-time will be correspondingly late (there is no longer a constant cycle). In general the target cycle-time is calculated as follows: 1.25 times the average cycle-time (from cycle measure read), background task time: 0.25 times the average cycle-time. The time can be set to a specified value to an accuracy of 1ms.



4.7 Windows

Under the "Window" menu item you will find all commands for managing the windows. There are commands both for the automatic set up of your window as well as for opening the library manager and for changing between open windows. At the end of the menu you will find a list of all open windows in the sequence they were opened. You can switch to the desired window by clicking the mouse on the relevant entry. A check will appear in front of the active window.

'Window' 'Tile Horizontal'

With this command you can arrange all the windows horizontally in the work area so that they do not overlap and will fill the entire work area.

'Window' 'Tile Vertical'

With this command you can arrange all the windows vertically in the work area so that they do not overlap and will fill the entire work area.

'Window' 'Cascade'

With this command you can arrange all the windows in the work area in a cascading fashion, one behind another.

'Window' 'Arrange Symbols'

With this command you can arrange all of the minimized windows in the work area in a row at the lower end of the work area.

'Window' 'Close all'

With this command you can close all open windows in the work area.

'Window' 'Messages' Shortcut: <Shift>+<Esc>

With this command you can open or close the message window with the messages from the last compiling, verifying, or comparing procedure. If the messages window is open, then a check (ü) will appear in front of the command.

'Window' 'Library Manager'

With this command you can open or close the library manager.

'Window' 'Log'

With this command you can open or close the Log window, where protocols of the online sessions can be displayed.

4.8 Help System

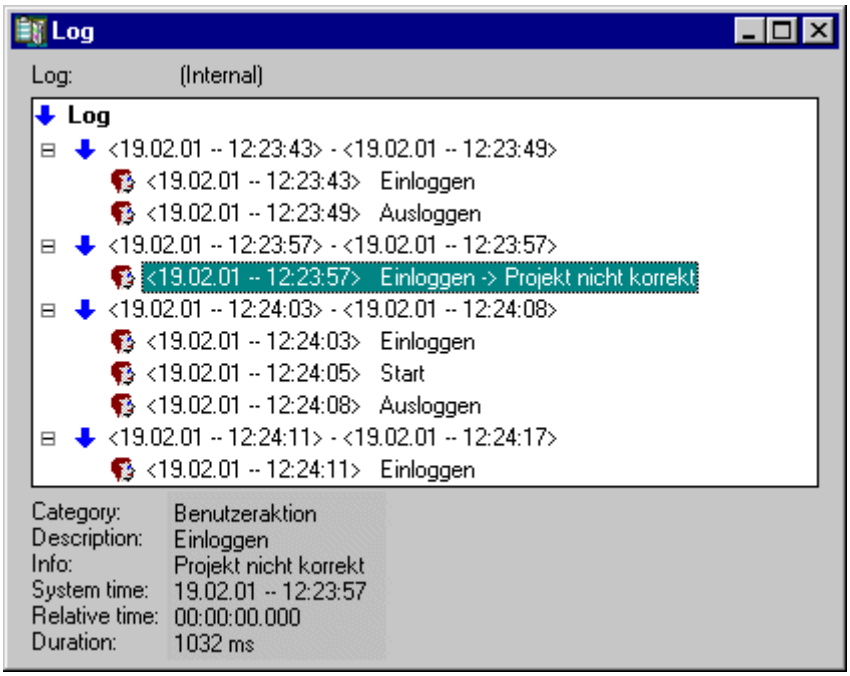
Should you encounter any problems with TwinCAT PLC Control during your work, online help is available to help to solve them. There you will find all the information that is also contained in this handbook. The help refers directly to Beckhoff Information System. The Beckhoff Information System has to be installed.

4.9 Log

The log stores in chronological order actions that occur during an Online session. For this purpose a binary log file (*.log) is set up. Afterward, the user can store excerpts from the appropriate project log in an external log.
The log window can be opened in either Offline or Online mode and can thus serve as a direct monitor online.

'Window''Log''

To open, select the menu item 'Window' 'Log'.



In the log window, the filename of the currently displayed log appears after **Log:**. If this is the log of the current project, the word "(Internal)" will be displayed.
Registered entries are displayed in the log window. The newest entry always appears at the bottom.
Only actions belonging to categories that have been activated in the 'Filter' field of the menu **'Project' 'Options' 'Log'** will be displayed. Available information concerning the currently selected entry is displayed below the log window:

Category: The category to which the particular log entry belongs. The following four categories are possible:

- **User action:** The user has carried out an Online action (typically from the Online menu).
- **Internal action:** An internal action has been executed in the Online layer (e.g. Delete Buffers or Init Debugging).
- **Status change:** The status of the runtime system has changed (e.g. from Running to Break, if a breakpoint is reached).
- **Exception:** An exception has occurred, e.g. a communication error.

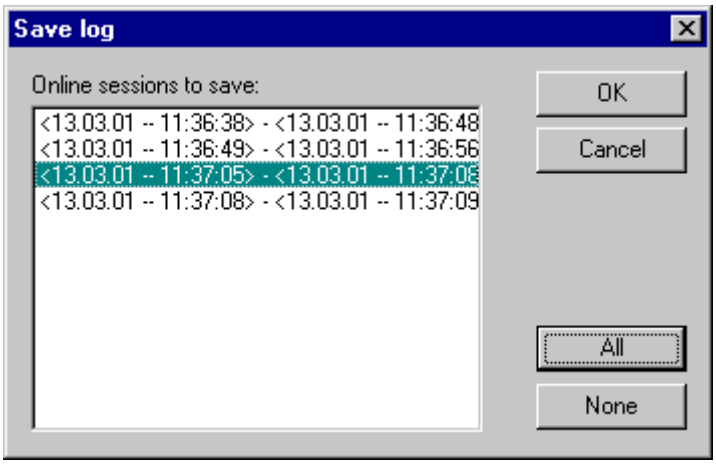
Description: The type of action. User actions have the same names as their corresponding menu commands; all other actions are in English and have the same name as the corresponding OnlineXXX() function.

Info: This field contains a description of an error that may have occurred during an action. The field is empty if no error has occurred. System time: The system time at which the action began, to the nearest second.

Relative time: The time measured from the beginning of the Online session, to the nearest millisecond.
Duration: Duration of the action in milliseconds.

Load...
An external log file *.log can be loaded and displayed using the standard file open dialog. The log that is present in the project will not be overwritten by the command. If the log window is closed and later opened again, or a new Online session is started then the version that is loaded will again be replaced by the project log.

Save…
This menu item can only be selected if the project log is currently displayed. It allows an excerpt of the project log to be stored in an external file. For that, the following dialog will be displayed, in which the Online sessions to be stored can be selected:



After successful selection, the standard dialog for storing a file opens ('Save Log').

Display Project Log
This command can only be selected if an external log is currently displayed. It switches the display back to the project log.

Storing the project log

Regardless of whether or not the log is stored in an external file (see above), the project log is automatically stored in a binary file entitled <projectname>.log. If a different path is not explicitly given in the 'Project' 'Options' 'Log' dialog, the file is stored in the same directory as that in which the project is stored.
The maximum number of Online sessions to be stored can be entered in the 'Project' 'Options' 'Log' dialog. If this number is exceeded during recording, the oldest session is deleted to make room for the newest one.

5 Editors

All editors for POUs (Program Organization Units) consist of a declaration part and a body. These are separated by a screen divider that can be dragged as required by clicking it with the mouse and moving it up or down. The body can consist of other a text or a graphic editor; the declaration portion is always a text editor.

Print margins

The vertical and horizontal margins that apply when the editor contents are printed, are shown by red dashed lines if the 'Show print range' option in the project options in the dialog **'Workspace'** was selected. The properties of the printer that was entered apply, as well as the size of the print layout selected in the **'File' 'Printer Setup'** menu. If no printer setup or no print layout is entered, a default configuration is used (Default.DFR and default printer). The horizontal margins are drawn as if the options 'New page for each object' or 'New page for each sub-object' were selected in '**Documentation settings**'. The lowest margin is not displayed.



An exact display of the print margins is only possible when a zoom factor of 100% is selected.

Comment

User comments must be enclosed in the special symbol sequences „(*“ and „*)“. Example: (*This is a comment.*)
Comments are allowed in all text editors, at any location desired, that is in all declarations, the IL and ST languages and in self-defined data types. If the Project is printed out using a **template**, the comment that was entered during variable declaration appears in text-based program components after each variable.
In the FBD and LD graphic editors, comments can be entered for each network. To do this, search for the network on which you wish to comment and activate 'Insert' 'Comment'. In CFC there are special comment POUs which can be placed at will.
In SFC, you can enter comments about a step in the dialog for editing step attributes.
Nested comments are also allowed if the appropriate option in the '**Project' 'Options' 'Build Options**' dialog is activated. In Online mode, if you rest the mouse cursor for a short time on a variable, the type and if applicable the address and comment of that variable are displayed in a tooltip.

Zoom to POU

With this command a selected POU is loaded into its editor. The command is available in the context menu (<F2>) or in the 'Extras' menu, if the cursor is positioned on the name of a POU in a text editor or if the POU box is selected in a graphic editor. If you are dealing with a POU from a library, then the library manager is called up, and the corresponding POU is displayed.

Open instance

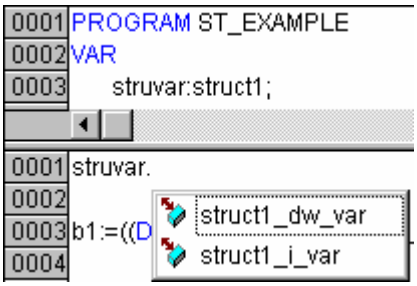
This command corresponds to the command 'Project' 'Open instance'. It is available in the context menu or in the 'Extras' menu, if the cursor is positioned on the name of a function block in a text editor or if the function block box is selected in a graphic editor.

Intellisense Function

If the option **'List components**' is activated in the project options dialog for category **'Editor'**, then the **"Intellisense" functionality** will be available in all editors, in the Watch- and Receiptmanager, in the Visualization and in the Sampling Trace:

- If you insert a dot "." instead of an identifier, a selection box will appear, listing all local and global variables of the project. You can choose one of these elements and press 'Return' to insert it behind the dot. You can also insert the element by a doubleclick on the list entry.
- If you enter a function block instance or a structure variable followed by a dot, then a selection box listing all input and output variables of the corresponding function block resp. listing the structure components will appear, where you can choose the desired element and enter it by pressing 'Return' or by a doubleclick.

Example:
Insert "struvar." -> the components of structure struct1 will be offered



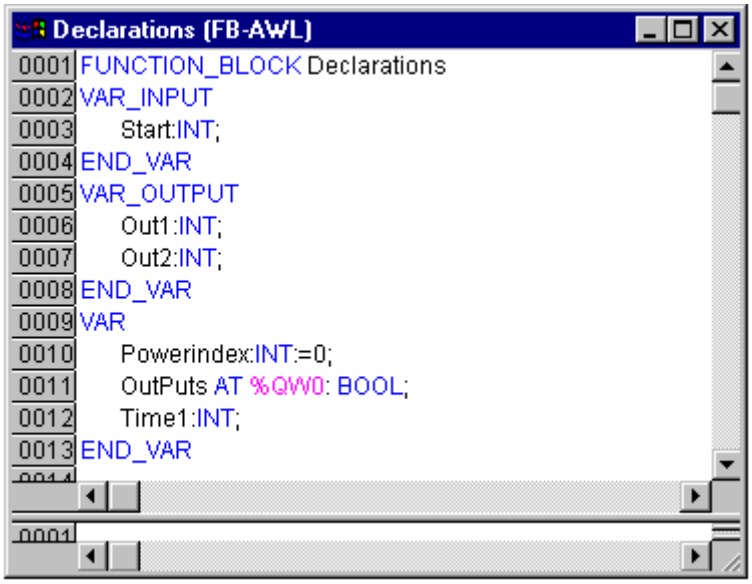
5.1 Declaration Editor

The declaration editor is used to declare variables of POUs and global variables, for data type declarations, and in the Watch and Recipe Manager. It gives access to the usual Windows functions, and even those of the IntelliMouse can be used if the corresponding driver is installed.
In Overwrite mode, 'OV' is shown in black on the status bar; switching between Overwrite and Insert modes can be accomplished with the <Ins> key. The declaration of variables is supported by syntax coloring.

The most important commands are found in the context menu (right mouse button).

Declaration Part

All variables to be used only in this POU are declared in the declaration part of the POU. These can include: input variables, output variables , in-put/output variables, local variables, retain variables, and constants. The declaration syntax is based on the IEC1131-3 standard. An example of a correct declaration of variables in TwinCAT PLC Control-Editor:



Declarationeditor

Input Variable

Between the key words VAR_INPUT and END_VAR, all variables are declared that serve as input variables for a POU. That means that at the call position, the value of the variables can be given along with a call.

Example:

```
VAR_INPUT
    in1:INT; (* 1. Inputvariable*)
END_VAR
```

Output Variable

Between the key words VAR_OUTPUT and END_VAR, all variables are declared that serve as output variables of a POU. That means that these values are carried back to the POU making the call. There they can be answered and used further.

Example:

```
VAR_OUTPUT
    out1:INT; (* 1. Outputvariable*)
END_VAR
```

Input and Output Variable

Between the key words VAR_IN_OUT and END_VAR, all variables are declared that serve as input and output variables for a POU.

With this variable, the value of the transferred variable is changed ("transferred as a pointer", Call-by-Reference). That means that the input value for such variables cannot be a constant. For this reason, even the VAR_IN_OUT variables of a function block can not be read or written directly from outside via <functionblockinstance><in/outputvariable>.

Example:

```
VAR_IN_OUT
    inout1:INT; (* 1. Input/Outputvariable *)
END_VAR
```

Local Variables

Between the keywords VAR and END_VAR, all of the local variables of a POU are declared. These have no external connection; in other words, they can not be manipulated from the outside.

Example:

```
VAR
    loc1:INT; (* 1. Local Variable*)
END_VAR
```

Remanent Variables

Remanent variables can retain their value throughout the usual program run period. These include Retain variables and Persistent variables.

- Retain variables are identified by the keyword **RETAIN**. These variables maintain their value even after an uncontrolled shutdown of the controller as well as after a normal switching off and on of the controller (resp. at the command 'Online' 'Reset'. When the program is run again, the stored values will be processed further. A concrete example would be an piece-counter in a production line, that recommences counting after a power failure. Retain-Variables are reinitialized at a new download of the program unlike persistent variables.

All other variables are newly initialized, either with their initialized values or with the standard initializations.

Example:

```
VAR RETAIN
    rem1:INT; (* Retain Variable*)
END_VAR
```



If a local variable in a program is declared as RETAIN, exactly this variable is stored in the retain area (like a global retain variable).

If a local variable in a function block is declared as RETAIN, the complete instance of this function block is stored in the retain area (all data of the function block), but only the declared retain variable is treated as such.

If a local variable in a function is declared as RETAIN, this has no effect. The variable is not stored in the retain area.

Persistent variables

In addition to the remanent variables specified with RETAIN, there is another class of remanent variables. These variables are saved with their instance path and symbol name. The generation of symbols must be selected for this. While the variables saved with RETAIN are no longer available after a "Rebuild all" of the

PLC program and subsequent cold start, persistent variables remain. With a reset of the PLC also RETAIN variables are reinitialized, persistent variables can only be initialized with a **overall reset**. The persistent variables are saved to a file when the TwinCAT system is shut down and are pre-initialized again with their saved values when the system is restarted.

Sample:

```
VAR PERSISTENT
    Rem2:INT; (* Persistente Variable*)
END_VAR
```

In order to check the correctness of the PERSISTENT and RETAIN variables, the system variables contain a byte, bootDataFlags, in the SYSTEMINFO [▶ 217] structure. This byte indicates the state of the boot data after loading. The upper four bits indicate the state of the persistent data, while the lower four bits indicate the state of the retain data. In order to be able to use this information, the "TcSystem.lib" library must be linked in.

Bit number	Description
0	RETAIN variables: LOADED (without error)
1	RETAIN variables: INVALID (the back-up copy was loaded, since no valid data was present)
2	RETAIN variables: REQUESTED (RETAIN variables should be loaded, a setting in TwinCAT System Control)
3	reserved
4	PERSISTENT variables: LOADED (without error)
5	PERSISTENT variables: INVALID (the back-up copy was loaded, since no valid data was present)
6	reserved
7	reserved

When shutting TwinCAT down the PERSISTENT and RETAIN data is written into two files on the hard disk. The path can be specified in TwinCAT System Control by means of the TwinCAT system properties (PLC tab). The standard setting is "<Drive>:\TwinCAT\Boot". The files all have a fixed name with fixed extensions:

File name	Description
TCPLC_P_x.wbp	Boot project (x = number of the run-time system)
TCPLC_R_x.wbp	RETAIN variables (x = number of the run-time system)
TCPLC_T_x.wbp	PERSISTENT variables (x = number of the run-time system)
TCPLC_R_x.wb~	Backup copy of the RETAIN variables (x = number of the run-time system)
TCPLC_T_x.wb~	Backup copy of the PERSISTENT variables (x = number of the run-time system)

If the file for the persistent and/or retain variables can not be written when shutting TwinCAT down, the standard reaction is for the backup file to be loaded. In that case bit 1 of the bootDataFlags (for the RETAIN variables) in the PLC and/or bit 5 (for the PERSISTENT variables) is set.

If the back-up file is not to be used under any conditions, a setting must be made in the NT registry. In the registry editor, under

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Beckhoff\TwinCAT\Plc]
"ClearInvalidRetainData"=dword:00000000
"ClearInvalidPersistentData"=dword:00000000
the value of "ClearInvalidRetainData" or of "ClearInvalidPersistentData" must be set to 1. The
default setting is 0.
```

Constants, Typed literals

Constants are identified by the key word **CONSTANT**. They can be declared locally or globally.

Syntax:

```
VAR CONSTANT
    <Identifier>:<Type> :=
<initialization>;
END_VAR
```

Example:

```
VAR CONSTANT
    con1:INT:=12; (* 1. Constant*)
END_VAR
```

You will find a listing of possible constants here in the appendix. See there also regarding the possibility of using typed constants (Typed Literals).

External Variables

Global variables which are to be imported into the POU are designated with the keyword **EXTERNAL**. They also appear in the Watch window of the declaration part in Online mode. If the VAR_EXTERNAL declaration does not match the global declaration in every respect, the following error message appears: "Declaration of '<var>' does not match global declaration!" If the global variable does not exist, the following error message appears: "Unkown global variable: '<var>'!"
Example:

```
VAR EXTERNAL
    var_ext1:INT:=12; (* 1st external variable *)
END_VAR
```

Keywords

Keywords are to be written in uppercase letters in all editors. Keywords may not be used as variables.

Variables declaration

A variables declaration has the following syntax:

```
<Identifier> {AT
<Address>}:<Type> {:=<initialization>};
```

The parts in the braces {} are optional.

Regarding the identifier, that is the name of a variable, it should be noted that it may not contain spaces or umlaut characters, it may not be declared in duplicate and may not be identical to any keyword. Upper/ lowercase writing of variables is ignored, in other words VAR1, Var1 and var1 are not different variables. Underlines in identifiers are meaningful, e.g. A_BCD and AB_CD are interpreted as different identifiers. Multiple consecutive underlines at the beginning of an identifier or within a identifier are not allowed.

The length of the identifier, as well as the meaningful part of it, are unlimited.

All declarations of variables and data type elements can include initialization. They are brought about by the ":=" operator. For variables of elementary types, these initializations are constants. The default-initialization is 0 for all declarations.

Example:

```
var1:INT:=12; (* Integer variable with initial
value of 12*)
```

If you wish to link a variable directly to a definite address, then you must declare the variable with the keyword **AT**. For faster input of the declarations, use the **shortcut mode**. In function blocks you can also specify variables with incomplete address statements. In order for such a variable to be used in a local instance, there must be an entry for it in the **variable configuration**.

Pay attention to the possibility of an automatic declaration!

AT Declaration

If you wish to link a variable directly to a definite address, then you must declare the variable with the keyword AT. The advantage of such a procedure is that you can assign a meaningful name to an address, and that any necessary changes of an incoming or outgoing signal will only have to be made in one place (e.g., in the declaration). Notice that variables requiring an input cannot be accessed by writing. A further restriction is that AT declarations can only be made for local and global variables, and not for input- and output variables from POUs. (Program Organization Units).

Examples:

```
counter_heat7 AT %QX0.0: BOOL;
lightcabinetimpulse AT %IX7.2: BOOL;
download AT %MX2.2: BOOL;
```



If boolean variables are assigned to a Byte, Word or DWORD address, they occupy one byte with TRUE or FALSE, not just the first bit after the offset!

'Insert' 'Declarations keywords'

You can use this command to open a list of all the keywords that can be used in the declaration part of a POU. After a keyword has been chosen and the choice has been confirmed, the word will be inserted at the present cursor position. You also receive the list, when you open the Input Assistant and choose the Declarations category.

'Insert' 'Types'

With this command you will receive a selection of the possible types for a declaration of variables. You also receive the list when you access the Input Assistant (<F2>).

The types are divided into these categories:

- Standard types: BOOL, BYTE, etc.
- Defined types: Structures, enumeration types, etc.
- Standard function blocks for instance declarations
- Defined function blocks for instance declarations

TwinCAT PLC Control supports all standard types of IEC61131-3. Examples for the use of the various types are found in the appendix.

Syntaxcoloring

In the text editors and in the declaration editor, you receive visual support in the implementation and declaration of variables. Errors are avoided, or discovered more quickly, because the text is displayed in color.
A comment left unclosed, thus annotating instructions, will be noticed immediately; keywords will not be accidentally misspelled, etc.

The following color highlighting will be used:

Blue	Keywords
Green	Comments
Pink	Boolean values (TRUE/FALSE)
Red	Input error (for example, invalid time constant, keyword, written in lower case,...)
Black	Variables, constants, assignment operators, ...

Shortcut Mode

The declaration editor for TwinCAT PLC Control allows you to use the shortcut mode. This mode is activated when you end a line with <Ctrl><Enter>

The following shortcuts are supported:

- All identifiers up to the last identifier of a line will become declaration variable identifiers
- The type of declaration is determined by the last identifier of the line.

In this context, the following will apply:

B or BOOL	gives the result BOOL
I or INT	gives the result INT
R or REAL	gives the result REAL
S or STRING	gives the result STRING

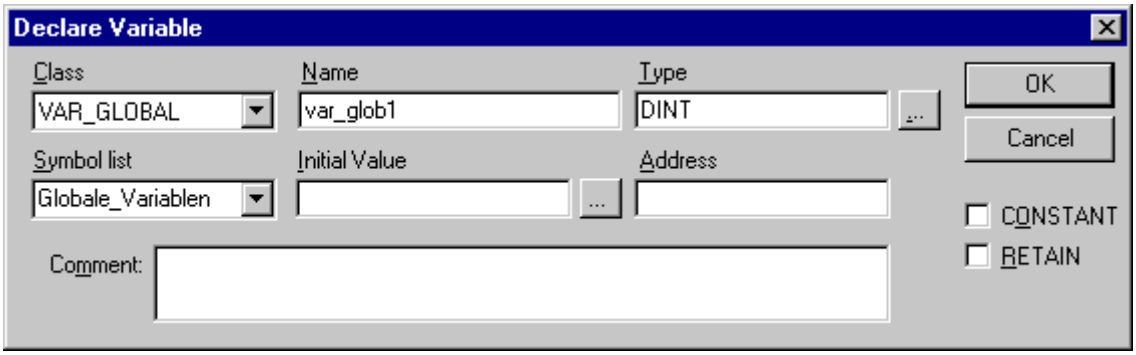
If no type has been established through these rules, then the type is BOOL and the last identifier will not be used as a type (Example 1.). Every constant, depending on the type of declaration, will turn into an initialization or a string (Examples 2. and 3.). An address (as in %MD12) is extended around the AT... attribute (Example 4.). A text after a semicolon (;) becomes a comment (Example 4.). All other characters in the line are ignored (e.g., the exclamation point in Example 5.).

Examples:

Short Description	Declaration
A	A: BOOL;
A B I 2	A, B: INT := 2;
ST S 2; A String	ST: STRING(2); (* A String *)
X %MD12 R 5; Real Number	X AT %MD12: REAL := 5.0;(* Real Number *)
B !	B: BOOL;

Autodeclaration

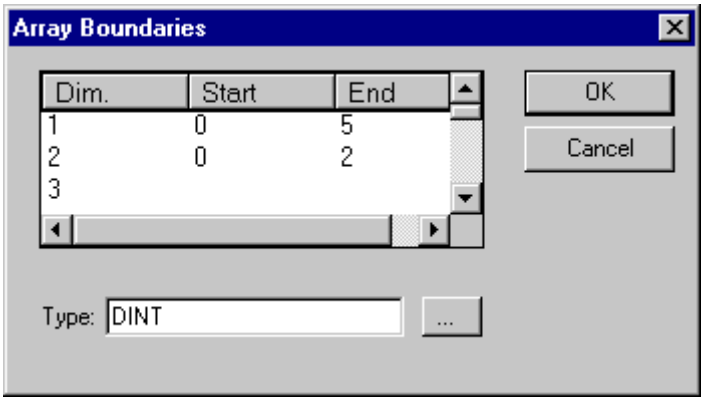
If the Autodeclaration of the options dialog box , then a dialog box will appear in all editors after the input of a variable that has not yet been declared. With the help of this dialog box, the variable can now be declared.



Dialog to declare variables

With the help of the **Class** combobox, select whether you are dealing with a local variable (**VAR**), input variable((**VAR_INPUT**), output variable (**VAR_OUTPUT**), input/output variable (**VAR_IN_OUT**), or a global variable (**VAR_GLOBAL**). With the **CONSTANT** and **RETAIN** options, you can define whether you are dealing with a constant or a retain variable.

The variable **name** you entered in the editor has been entered in the Name field, BOOL has been placed in the Type field.
The button... opens the Input Assistant dialog which allows you to select from all possible data types.
If ARRAY is chosen as the variable type, the dialog for entering array boundaries appears.



Declaration editor for arrays

For each of the three possible dimensions (**Dim**.), array boundaries can be entered under **Start** and **End** by clicking with the mouse on the corresponding field to open an editing space. The array data type is entered in the **Type** field. In doing this, the button... can be used to call up an input assistant dialog.

Upon leaving the array boundaries dialog via the **OK** button, variable declarations in IEC format are set up based on the entries in the Type field in the dialog. Example: ARRAY [1..5, 1..3] OF INT

In the field **Initial value**, you may enter the initial value of the variable being declared. If this is an array or a valid structure, you can open a special initialization dialog via the button or <F2>, or open the input assistant dialog for other variable types.
In the initialization dialog for an array you are presented a list of array elements; a mouse click on the space following „:=“opens an editing field for entering the initial value of an element.

In the initialization dialog for a structure, individual components are displayed in a tree structure. The type and default initial value appear in brackets after the variable name; each is followed by „:=“. A mouse click on the field following „:=“ opens an editing field in which you can enter the desired initial value. If the component is an array, then the display of individual fields in the array can be expanded by a mouse click on the plus sign before the array name and the fields can be edited with initial values.

After leaving the initialization dialog with OK, the initialization of the array or the structure appears in the field Initial value of the declaration dialog in IEC format. Example: x:=5,field:=2,3,struct2:=(a:=2,b:=3)

In the Address field, you can bind the variable being declared to an IEC address (AT declaration).
If applicable, enter a Comment. The comment can be formatted with line breaks by using the key combination <Ctrl> + <Enter>. By pressing OK, the declaration dialog is closed and the variable is entered in the corresponding declaration editor in accordance to the IEC syntax.



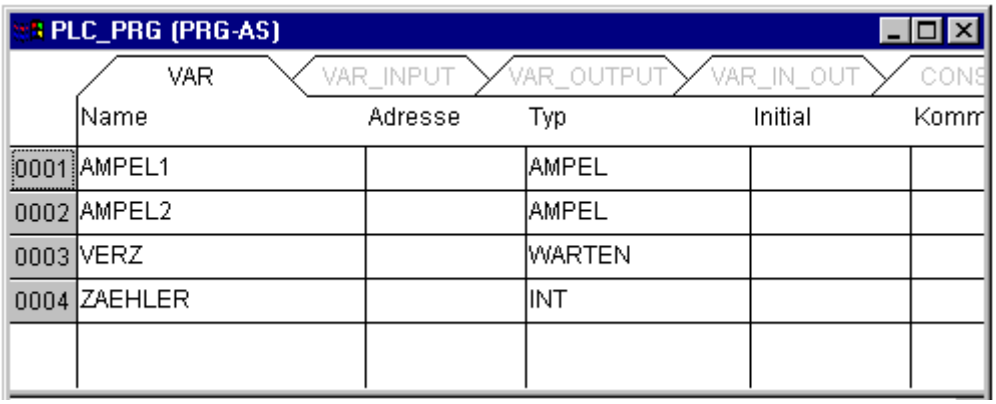
The dialog box for variable declaration you also get by the command 'Edit' 'Declare Variable' (see Chapter Editing Functions). If the cursor is resting on a variable in Online mode, the Autodeclare window can be opened with <Shift><F2> with the current variable-related settings displayed.

Line Numbers in the Declaration Editor

In offline mode, a simple click on a special line number will mark the entire text line. In the online mode, a single click on a specific line number will open up or close the variable in this line, in case a structural variable is involved.

Name	Input the identifier of the variable.
Address	Input the address of the variable (AT declaration)
Type	Input the type of the variable. (Input the function block when instantiating a function block)
Initial	Enter a possible initialization of the variable (corresponding to the ":= " assignment operator)
Comment	Enter a comment here.

Both of the display types of the declaration editor can be changed without causing any problems. In the online mode, there are no differences for the display.



Declaration Editor as a Table

'Insert' 'New Declaration '

With this command you bring a new variable into the declaration table of the declaration editor. If the present cursor position is located in an field of the table, then the new variable will be pasted in the preceding line; otherwise, the new variable is pasted at the end of the table. Moreover, you can paste a new declaration at the end of the table by using the right arrow key or the tab key in the last field of the table. You will receive a variable that has "**Name**" located in the Name field, and "**Bool**" located in the Type field, as its default setting. You should change these values to the desired values. Name and type are all that is necessary for a complete declaration of variables.

Pragma command

The pragma instruction is used to affect the properties of a variable concerning the compilation process. It can be used in with supplementary text in a program line of the declaration editor or in its own line.

The pragma instruction is enclosed in curly brackets, upper- and lower-case are ignored:
{ <Instruction text> }
If the compiler cannot meaningfully interpret the instruction text, the entire pragma is handled as a comment and read over. A warning is issued, however: „Ignore compiler directive ‚<Instruction text>'!"
Depending on the type and contents of pragma, the pragma either operates on the line in which it is located or on all subsequent lines until it is ended by an appropriate pragma, or the same pragma is executed with different parameters, or the end of the file is reached. By file we mean here: declaration part, implementation portion, global variable list, type declaration.

The opening bracket may immediately follow a variable name. Opening and closing brackets must be located on the same line.

```
The following pragma may currently be used:
```

```
{flag}: {flag [<flags>] [off|on]}
```

This pragma allows the properties of a variable declaration to be affected.
<flags> can be a combination of the following flags:

noinit	The variable will not be initialized.
nowatch	The variable can not be monitored.
noread	The variable is exported to the symbol file without read permission.
nowrite	The variable is exported to the symbol file without write permission.
noread, nowrite	The variable is not exported to the symbol file.

With the „**on**" modifier, the pragma operates on all subsequent variable declarations until it is ended by the pragma {flag off}, or until overwritten by another {flag <flags> on} pragma. Without the „on" or „off" modifier, the pragma operates only on the current variable declaration (that is the declaration that is closed by the next semicolon).
Examples:

The variable a will not be initialized and will not be monitored. The variable b will not be initialized:

```
VAR
    a : INT {flag noinit, nowatch};
    b : INT {flag noinit};
END_VAR
```

```
VAR
    {flag noint, nowatch on}
    a : INT {flag noinit on};
    b : INT;
    {flag off}
END_VAR
```

Neither variable will be initialized:

```
{flag noinit on}
VAR
    a : INT;
    b : INT;
END_VAR
{flag off}
```

```
VAR
    {flag noinit on}
    a : INT;
```

```
    b : INT;
    {flag off}
END_VAR
```

The flags „**noread**“ and „**nowrite**“ are used in a POU that has read and/or write permission to provide selected variables with restricted access rights. The default for the variable is the same as the setting for the POU in which the variable is declared. If a variable has neither read nor write permission, it will not be exported into the symbol file.
Examples:
If the POU has read and write permission, then with the following pragmas variable a can only be exported with write permission, while variable b can not be exported at all.

```
VAR
    a : INT {flag noread};
    b : INT {flag noread, nowrite};
END_VAR
```

Neither variable a nor b will be exported to the symbol file:

```
{flag noread, nowrite on}
VAR
    a : INT;
    b : INT;
END_VAR
{flag off}
```

The pragma operates additively on all subsequent variable declarations.
Example:
all POUs in use will be exported with read and write permission

```
a : afb;
...
FUNCTION_BLOCK afB
VAR
    b : bfb {flag nowrite};
    c : INT;
END_VAR
...
FUNCTION_BLOCK bfB
VAR
    d : INT {flag noread};
    e : INT {flag nowrite};
END_VAR
"a.b.d": Will not be exported.
"a.b.e": Will be exported only with read permission.
"a.c": Will be exported with read and write permission.
```

Pragmas for Controlling the Display of Library Declaration Parts

During creation of a library in TwinCAT PLC you can define via pragmas which parts of the declaration window should be visible resp. not visible in the Library Manager later when the library will be included in a project. The display of the implementation part of the library will not be affected by that. Thus comments or any variables declarations can be concealed from the user. The pragmas {library private} and {library public} each affect the rest of the same line resp. the subsequent lines, as long as they are not overwritten by the each other one.

Syntax:

{library public} The subsequent test will be displayed in the Library Manager.
{library private} :The subsequent test will be not displayed.

Example: See below the declaration part of a library, which is created in TwinCAT PLC. The comment "(* this is for all *)" should be displayed in the Library Manager after having included the library in a project., the comment "(* but this is not for all ")" however should not be displayed. The variables *local* and *in3* also should not be displayed:

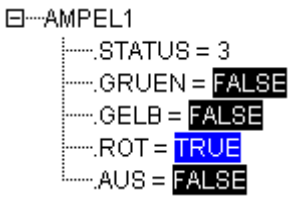
```
{library public}
    (*this is for all*)
{library private}
    (*this is not for all*)
{library public}
FUNCTION afun   : BOOL
VAR_INPUT
```

```
    in      : BOOL;
END_VAR
{library private}
VAR
    local   : BOOL;
END_VAR
{library public}
VAR_INPUT
    in2     : BOOL;
{library private}
    in3     : BOOL;
{library public}
END_VAR
```

Declaration Editors in Online Mode

In online mode, the declaration editor changes into a monitor window. In each line there is a variable followed by the equal sign (=) and the value of the variable. If the variable at this point is undefined, three question marks (???) will appear. For function blocks, values are displayed only for open instances (command: 'Project' 'Open instance').

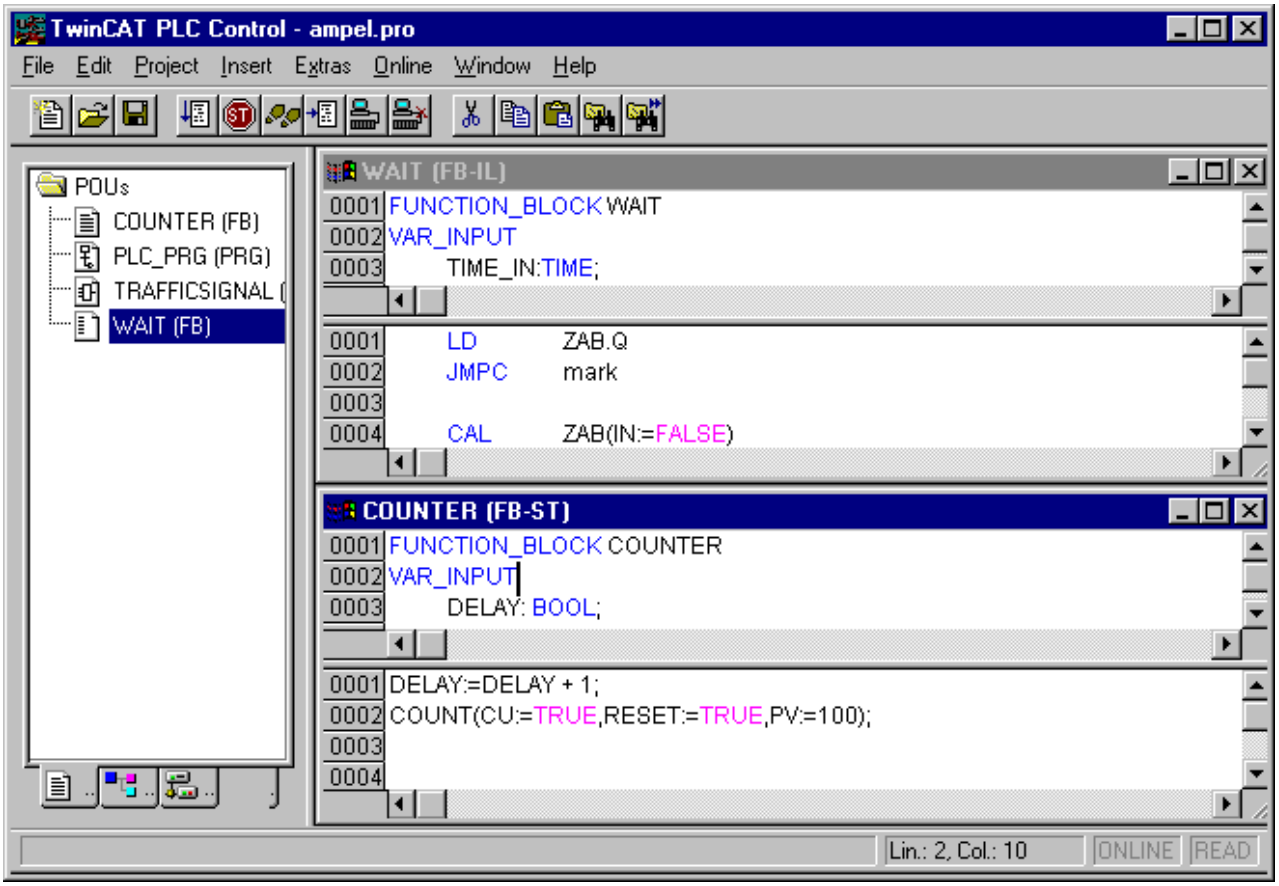
In front of every multi-element variable there is a plus sign. By pressing <Enter> or after doubleclicking on such a variable, the variable is opened up. In the example, the traffic signal structure would be opened up.



When a variable is open, all of its components are listed after it. A minus sign appears in front of the variable. If you doubleclick again or press <Enter>, the variable will be closed, and the plus sign will reappear. Pressing <Enter> or doubleclicking on a single-element variable will open the dialog box to write a variable. Here it is possible to change the present value of the variable. In the case of Boolean variables, no dialog box appears; these variables are toggled. The new value will turn red and will remain unchanged. If the "**Online" "Write values**" command is given, then all variables are placed in the selected list and are once again displayed in black. If the "Online" "**Force values**" command is given, then all variables will be set to the selected values, until the "Release force" command is given. In this event, the color of the force value changes to red.

5.2 Text Editors

The text editors (the instruction list editor [▶ 128] and the editor for Strucured text [▶ 129]) of TwinCAT PLC Control offer the usual capabilities of Windows text editors. The implementation in the text editors is supported by syntax coloring.



Text Editors for the Instruction List and Structured Text

The most important commands are found in the context menu (right mouse button or <Ctrl>+<F10>). The text editors use the following menu commands in special ways:

'Insert' 'Operator'

With this command all of the operators available in the current language are displayed in a dialog box. If one of the operators is selected and the list is closed with **OK**, then the highlighted operator will be inserted at the present cursor position.

'Insert' 'Operand'

With this command all variables in a dialog box are displayed. You can select whether you would like to display a list of the global, the local, or the system variables. If one of the operands is chosen, and the dialog box is closed with **OK**, then the highlighted operand will be inserted at the present cursor position.

'Insert' 'Function'

With this command all functions will be displayed in a dialog box. You can choose whether to have a list displaying user-defined or standard functions. If one of the functions is selected and the dialog box is closed with OK, then the highlighted function will be inserted at the current cursor position. If the With arguments option was selected in the dialog box, then the necessary input and output variables will also be inserted.

'Insert' 'Functionblock'

With this command all function blocks are displayed in a dialog box. You can choose whether to have a list displaying user-defined or standard function blocks. If one of the function blocks is selected and the dialog box is closed with OK, then the highlighted function block will be inserted at the current cursor position. If the With arguments option was selected in the dialog box, then the necessary input and output variables of the function block will also be inserted.

Calling POUs with output parameters

The output parameters of a called POU can be directly assigned upon being called in the text languages IL and ST. Example: Output parameter out1 of afbinst is assigned variable a.

```
IL: CAL afbinst(in1:=1, out1=>a)
ST: afbinst(in1:=1, out1=>a);
```

The text editors in Online mode

The online functions in the editors are set breakpoint and single step processing (steps). Together with the monitoring, the user thus has the debugging capability of a modern Windows standard language debugger.

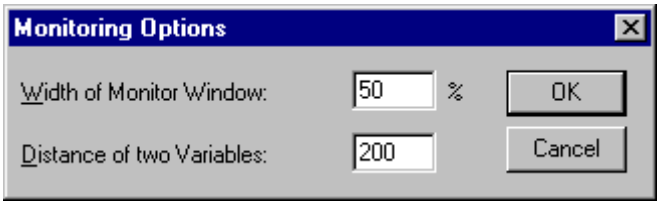
In Online mode, the text editor window is vertically divided in halves. On the left side of the window you will then find the normal program text; on the right side you will see a display of the variables whose values were changed in the respective lines.
The display is the same as in the declaration part. That means that when the PLC is running, the present values of the respective variables will be displayed.

The following should be noted when monitoring expressions or Bit-addressed variables: in the case of expressions, the value of the entire expression is always displayed. Example: a AND b is displayed in blue or with „:=TRUE“ if both a and b are TRUE. For Bit-addressed variables, the bit value that is addressed is always monitored (e.g. a.3 is displayed in blue or with „:=TRUE, if a has the value 4).

If you place the mouse pointer briefly above a variable, then the type and the comment about the variable will be displayed in a Tooltip.

'Extras' 'Monitoring Options'

With this command you can configure your monitoring window. In the text editors, the window is divided into two halves during monitoring. The pro-gram is located in the left half. In the right half, all variables that are located in the corresponding program line are monitored. You can specify the Monitor Window Width and which Distance two variables should have in a line. An distance declaration of 1 corresponds, in this case, to a line height in the selected font.



Monitoring Options Dialog Box

Breakpoint Positions

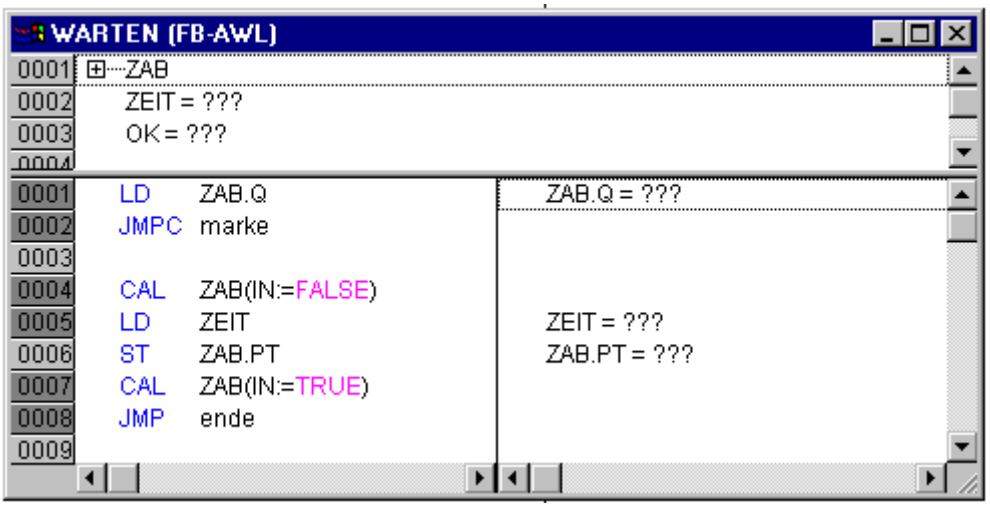
Breakpoint PositionsSince in TwinCAT PLC Control several IL lines are internally combined into a single C-code line, breakpoints can not be set in every line. Breakpoint positions include all positions in a program at which values of variables can change or where the program flow branches off. (Exception: function calls. If necessary, a breakpoint in the function must be set here.) At the positions lying inbetween, a breakpoint would not even make sense, since nothing has been able to change in the data since the preceding breakpoint position. This results in the following breakpoint positions in the IL:

- At the start of the POU
- At every LD, LDN (or, in case a LD is located at a label, then at the label)
- At every JMP, JMPC, JMPCN
- At every label· At every CAL, CALC, CALCN
- At every RET, RETC, RETCN
- At the end of the POU

Structured text accommodates the following breakpoint positions:·

- At every assignment
- At every RETURN and EXIT instruction
- in lines where conditions are being evaluated (WHILE, IF, REPEAT)
- At the end of the POU

Breakpoint positions are marked by the display of the line number field in a darker gray.



IL Editor with Possible Breakpoint Positions (darker number fields)
How do you set a breakpoint?

In order to set a breakpoint, click the line number field of the line where you want to set a breakpoint. If the selected field is a breakpoint position, then the color of the line numbers field will change from dark gray to light blue, and the breakpoint will be activated in the PLC.

Deleting Breakpoints

Correspondingly, in order to delete a breakpoint, click on the line number field of the line with the breakpoint to be deleted. Setting and deleting of breakpoints can also be selected via the menu ("**Online" "Toggle Breakpoint**"), via the function key <F9>, or via the symbol in the tool bar.

What happens at a breakpoint?

If a breakpoint is reached in the PLC, then the screen will display the break with the corresponding line. The line number field of the line where the PLC is positioned will appear in red.
The user program is stopped in the PLC. If the program is at a breakpoint, then the processing can be resumed with **"Online" "Run"**.
In addition, with **"Online" "Step over"** or **"Step in"** you can cause the program to run to the next breakpoint position. If the instruction where you are located is a CAL command, or, if there is a function call in the lines up to the next breakpoint position, then you can use "**Step over**" to bypass the function call. With "**Step in**", you will branch to the open POU.

Line Number of the Text Editor

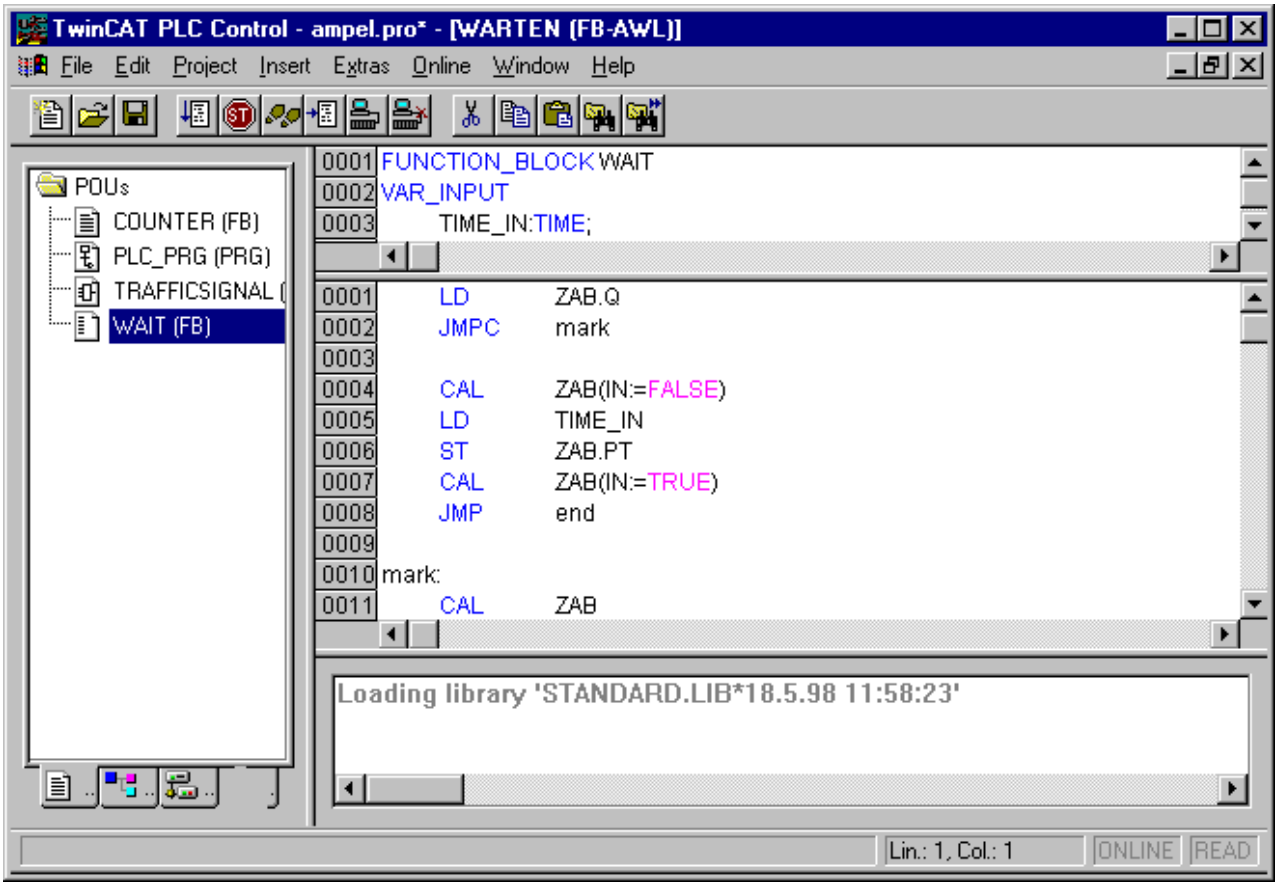
The line numbers of the text editor give the number of each text line of an implementation of a POU. In Off-line mode, a simple click on a special line number will mark the entire text line. In Online mode, the background color of the line number indicates the breakpoint status of every line:

- dark gray: This line is a possible position for a breakpoint.
- light blue: a breakpoint has been set in this line.
- red: The program has reached this point.

In Online mode, simply clicking the mouse will change the breakpoint status of this line.

5.3 Instruction List Editor

This is how a POU written in the IL looks under the corresponding TwinCAT PLC Control editor:



IL-Editor

All editors for POUs consist of a declaration part and a body. These are separated by a screen divider.

The instruction list editor is a text editor with the usual capabilities of Windows text editors. The most important commands are found in the context menu (right mouse button or <Ctrl>+<F10>).

Multiline POU calls are also possible:
Example:

```
CAL CTU_inst(
CU:=%IX10,
PV:=(
LD A
ADD 5
)
)
```

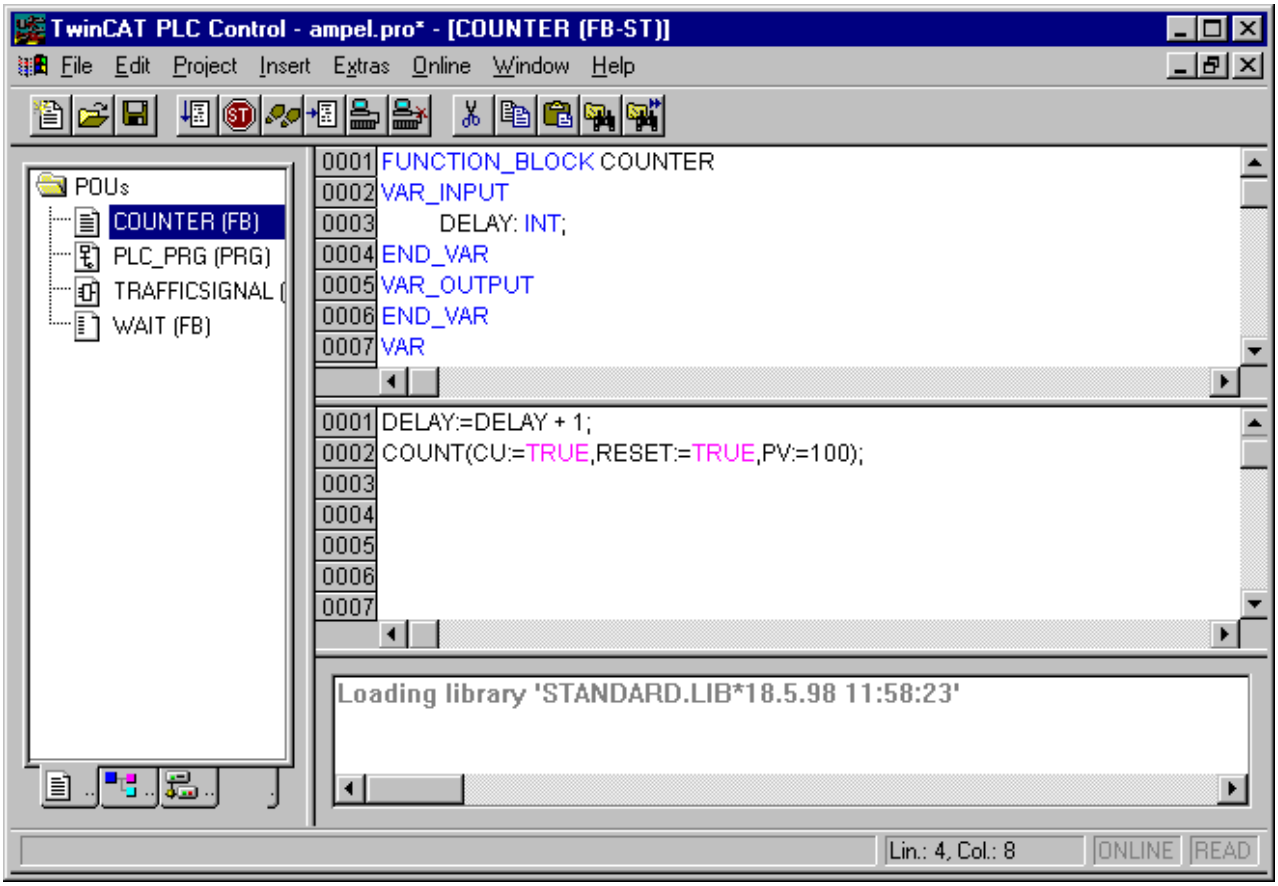
Flow Control

With the **"Online" "Flow control"** command, an additional field in which the accumulator contents is displayed is inserted in the IL editor on the left side of every line.

For information about the IL editor in Online mode, see 'Text Editors in Online Mode'.

5.4 Structured Text Editor

This is how a POU written in ST appears under the corresponding TwinCAT PLC Control editor:



Editor for Structured Text

All editors for POUs consist of a declaration part and a body. These are separated by a screen divider.

The editor for structured text is a text editor with the usual capabilities of Windows text editors. The most important commands are found in the con-text menu (right mouse button).

For information about the language, read the chapter Structured Text(ST).

5.5 Graphical Editors

The editors of both of the graphically oriented languages, sequential function chart SFC [▶ 152], LD [▶ 136] and FBD [▶ 132] of free graphic function block diagrams CFC [▶ 141] have many points in common. These points are summarized in the following chapters. The implementation in the graphics editors is supported by syntax coloring.

Zoom

Objects such as POUs, actions, transitions etc. in the languages SFC, LD, FBD, and CFC can be enlarged or reduced in size with a zoom function. All elements of the window contents of the implementation part are affected; the declaration part remains unchanged.

In standard form, every object is displayed with the zoom level 100%. The zoom level that is set is saved as an object property in the project.

The printing of project documentation always occurs at the 100% display level!
The zoom level can be set through a selection list in the toolbar. Values between 25% and 400% can be selected; individual values between 10% and 500% can be entered manually.

The selection of a zoom level is only available if the cursor rests on an object created in a graphical language or a visualization object.

Even with the object zoomed, cursor positions in the editors can be further selected and even reached with the arrow keys. Text size is governed by the zoom factor and the font size that is set.

The execution of all editor menu features (e.g. inserting a box) as a function of cursor position is available at all zoom levels, taking the same into account.

In Online mode, each object is displayed according to the zoom level that has been set; Online functionality is available without restriction.
When the IntelliMouse is used, an object can be enlarged/reduced by pressing the <CTRL> key and at the same time turning the wheel forward or backwards.

Network

In the LD and FBD editors, the program is arranged in a list of networks. Each network is designated on the left side by a serial network number and has a structure consisting of either a logical or an arithmetic expression, a program, function or function block call, and a jump or a return instruction.

Label

Each network has a label that can optionally be left empty. This label is edited by clicking the first line of the network, directly next to the network number. Now you can enter a label, followed by a colon.

Network comments

Every network can be supplied with a multi-lined comment. In **"Extras" "Options"**, you can enter the **maximum** number of lines to be made available for a **network comment**. This entry is made in the maximum comment size array. (The default value here is 4.) You can also enter the number of lines that generally should be reserved for comments (**minimum comment size**). If, for example, the number 2 is entered, then, at the start of each network there will be two empty lines after the label line. The default value here is 0, which has the advantage of allowing more networks to fit in the screen area.

If the minimal comment size is greater than 0, then in order to enter a comment you simply click in the comment line and then enter the comment. Otherwise you must next select the network to which a comment is to be entered, and use **"Insert" "Comment"** to insert a comment line. In contrast to the program text, comments are displayed in gray.

In the **Ladder editor** you can also assign an individual comment to each contact or coil. For this activate the option Comments per Contact and insert in the edit field Lines for variable comment the number of lines which should be reserved and displayed for the comment. If this setting is done, a comment field will be displayed in the editor above each contact and coil where you can insert text. If the option Comments per Contact is activated, then in the Ladder editor also the number of lines **(Lines for variable text :)** can be defined which should be used for the variable name of the contact resp. coil. This is used to display even long names completely by breaking them into several lines. In the Ladder editor it is possible to force linebreaks in the networks as soon as the network length would exceed the given window size and some of the elements would not be visible. For this activate the option Networks with Linebreaks.

'Insert' 'Network (after)' or 'Insert' 'Network (before)' Shortcut: <Shift>+<T> (Network after)

In order to insert a new network in the FBD or the LD editor, select the "Insert" "Network (after)" or the "Insert" "Network (before)" command, depending on whether you want to insert the new network before or after the present network. The present network can be changed by clicking the network number. You will recognize it in the dotted rectangle under the number. With the <Shift key> and a mouse click you can select from the entire area of networks, from the present one to the one clicked.

The network editors in the online mode

In the FBD and the LD editors you can only set breakpoints for networks. The network number field of a network for which a breakpoint has been set, is displayed in blue. The processing then stops in front of the network, where the breakpoint is located. In this case, the network number field is displayed in red. With single step processing (steps), you can jump from network to network.

All values are monitored upon entering and exiting network POUs (Program Organization Units).

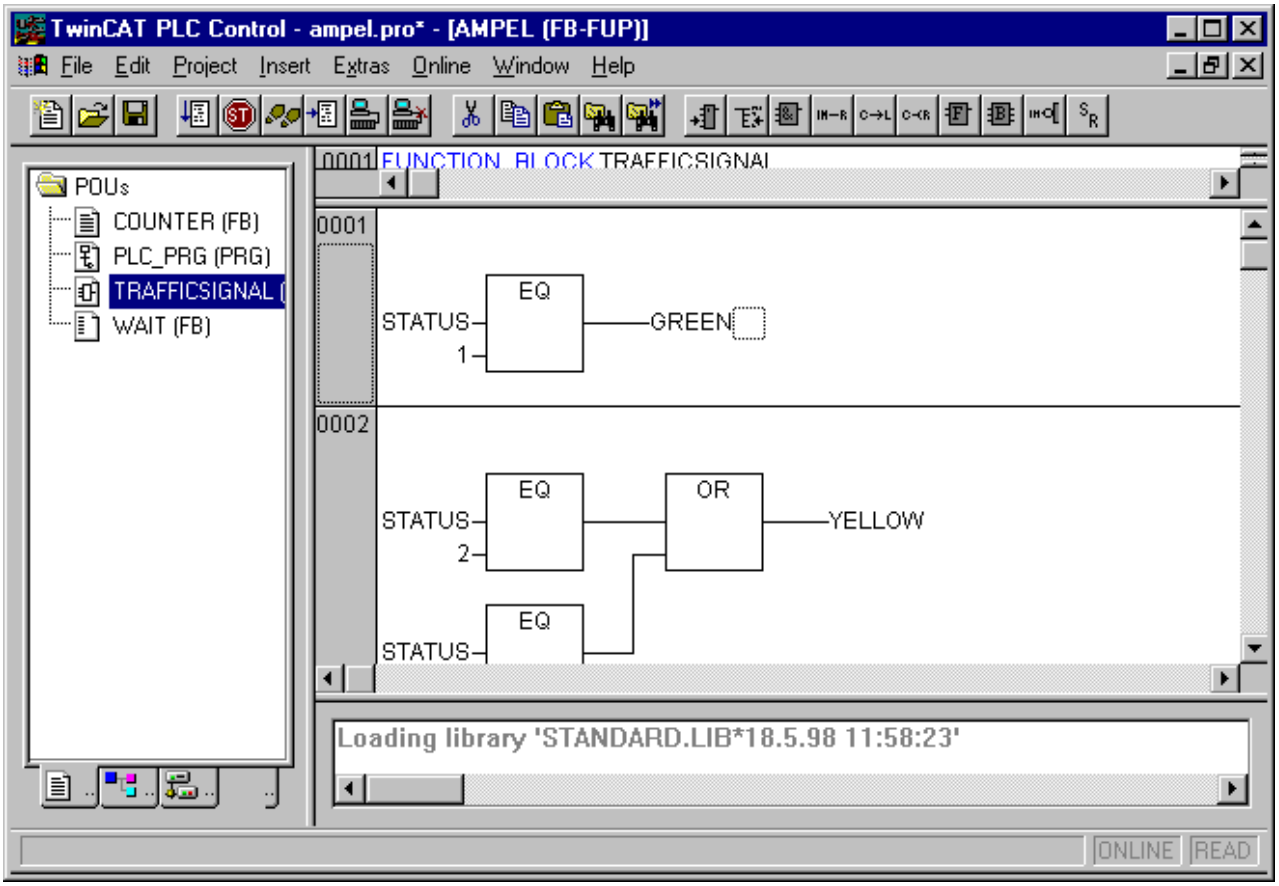
The following should be noted when monitoring expressions or Bit-addressed variables: In expressions, e.g. a AND b, used as transition condition or function block input, the value of the whole expression is always displayed (a AND b is shown in blue or as :=TRUE, if a and b are TRUE). For Bit-addressed variables, the bit value that is addressed is always monitored (e.g. a.3 is displayed in blue or with „:=TRUE, if a has the value 4).

The flow control is run with the **"Online" "Flow control"** command. Using the flow control, you can view the present values that are being carried in the networks over the connecting lines. If the connecting lines do not carry Boolean values, then the value will be displayed in a specially inserted field. The monitor fields for variables that are not used (e.g. in the function SEL) are displayed in a shade of grey. If the lines carry Boolean values, then they will be shaded blue, in the event that they carry TRUE. Therefore, you can accompany the flow of information while the PLC is running.

If you place the mouse pointer briefly above a variable, then the type and the comment about the variable will be displayed in a Tooltip.

5.6 Function Block Diagram Editor

This is how a POU written in the FBD under the corresponding TwinCAT PLC Control editor looks:



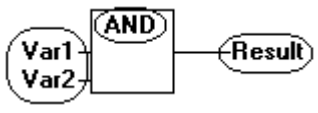
Editor for the Function Block Diagram

The Function Block Diagram editor is a graphic editor. It works with a list of networks, in which every network contains a structure that displays, respectively, a logical or an arithmetical expression, the calling up of a function block, a function, a program, a jump, or a return instruction.The most important commands are found in the context menu (right mouse button).

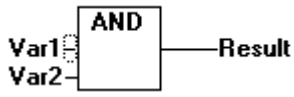
Cursor positions in FBD

Every text is a possible cursor position. The selected text is on a blue background and can now be changed. You can also recognize the present cursor position by a dotted rectangle. The following is a list of all possible cursor positions with an example:

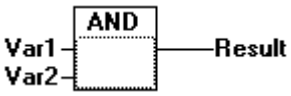
1) Every text field (possible cursor positions framed in black):



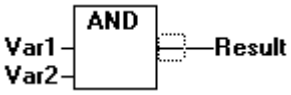
2) Every input:



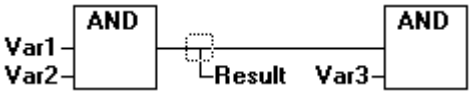
3) Every operator, function, or function block:



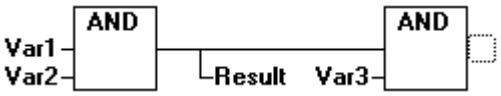
4) Outputs, if an assignment or a jump comes afterward:



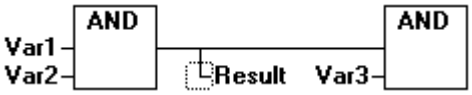
5) The lined cross above an assignment, a jump, or a return instruction:



6) Behind the outermost object on the right of every network ("last cursor position," the same cursor position that was used to select a network):



7) The lined cross directly in front of an assignment:



How to set the cursor

The cursor can be set at a certain position by clicking the mouse, or with the help of the keyboard. Using the arrow keys, you can jump to the nearest cursor position in the selected direction at any time. All cursor positions, including the text fields, can be accessed this way. If the last cursor position is selected, then the <up> or <down> arrow keys can be used to select the last cursor position of the previous or subsequent network. An empty network contains only three question marks "???". By clicking behind these, the last cursor position is selected.

'Insert' 'Assignment' Shortcut: <Ctrl>+<A>

This command inserts an assignment. Depending on the selected position, insertion takes place directly in front of the selected input (Cursor Position 2), directly after the selected output (Cursor Position 4), directly before the selected line cross (Cursor Position 5), or at the end of the network (Cursor Position 6). For an inserted assignment, a selection can be made accompanying the entered text "???", and the assignment can be replaced by the variable that is to be assigned. For this you can also use the Input Assistant. In order to insert an additional assignment to an existing assignment, use the "Insert" "Output" command.

'Insert' 'Jump' Shortcut: <Ctrl>+<L>

This command inserts a jump. Depending on the selected position, insertion takes place directly in front of the selected input (Cursor Position 2), directly after the selected output (Cursor Position 4), directly before the selected line cross (Cursor Position 5), or at the end of the network (Cursor Position 6). For an inserted

jump, a selection can be made accompanying the entered text "???", and the jump can be replaced by the label to which it is to be assigned.
'Insert' 'Return' Shortcut: <Ctrl>+<R>

This command inserts a RETURN instruction. Depending on the selected position, insertion takes place directly in front of the selected input (Cursor Position 2), directly after the selected output (Cursor Position 4), directly before the selected line cross (Cursor Position 5), or at the end of the network (Cursor Position 6).

'Insert' 'Box' Shortcut: <Ctrl>+<B>

With this command, operators, functions, function blocks and programs can be inserted. First of all, it is always inserted an “AND” operator. This can be converted by Selection and Overwrite of the type text („AND“) into every other operator, into every function, into every function block and every program. You can select the desired POU by using Input Assistant (<F2>). If the new selected block has another minimum number of inputs, these will be attached. If the new block has a smaller highest number of inputs, the last inputs will be deleted.

In functions and function blocks, the formal names of the in- and outputs are displayed.

In function blocks there exists an editable instance field above the box. If another function block that is not known is called by changing the type text, an operator box with two inputs and the given type is displayed. If the instance field is selected, Input Assistant can be obtained via <F2> with the categories for variable selection.

The newest POU is inserted at the selected position (see):

- If an input is selected (Cursor Position 2), then the POU is inserted in front of this input. The first input of this POU is linked to the branch on the left of the selected input. The output of the new POU is linked to the selected input.
- If an output is selected (Cursor Position 4), then the POU is inserted after this output. The first input of the POU is connected with the selected output. The output of the new POU is linked to the branch with which the selected output was linked.
- If a POU, a function, or a function block is selected (Cursor Position 3), then the old element will be replaced by the new POU.
- As far as possible, the branches will be connected the same way as they were before the replacement. If the old element had more inputs than the new one, then the unattachable branches will be deleted. The same holds true for the outputs.
- If a jump or a return is selected, then the POU will be inserted before this jump or return. The first input of the POU is connected with the branch to the left of the selected element. The output of the POU is linked to the branch to the right of the selected element.
- If the last cursor position of a network is selected (Cursor Position 6), then the POU will be inserted following the last element. The first input of the POU is linked to the branch to the left of the selected position.

All POU inputs that could not be linked will receive the text "???". This text must be clicked and changed into the desired constant or variable.

If there is a branch to the right of an inserted POU, then the branch will be assigned to the first POU output. Otherwise the outputs will be unassigned.

'Insert' 'Input' Shortcut: <Ctrl>+<U>

This command inserts an operator input. With many operators, the number of inputs may vary. (For example, ADD can have 2 or more inputs.) In order to extend such an operator by an input, you need to select the input in front of which you wish to insert an additional input (Cursor Position 1); or you must select the operator itself (Cursor Position 3), if a lowest input is to be inserted.

The inserted input is allocated with the text "???". This text must be clicked and changed into the desired constant or variable. For this you can also use the Input Assistant.
'Insert' 'Output'

This command inserts an additional assignment into an existing assignment. This capability serves the placement of so-called assignment combs; i.e., the assignment of the value presently located at the line to several variables.

If you select the lined cross above an assignment (Cursor Position 5) or the output directly in front of it (Cursor Position 4), then there will be another assignment inserted after the ones already there.

If the line cross directly in front of an assignment is selected (Cursor Position 4), then another assignment will be inserted in front of this one.

The inserted output is allocated with the text "???". This text must be clicked and changed into the desired variable. For this you can also use the Input Assistant

'Extras' 'Negation' Shortcut: <Ctrl>+<N>

With this command you can negate the inputs, outputs, jumps, or RETURN instructions. The symbol for the negation is a small circle at a connection.

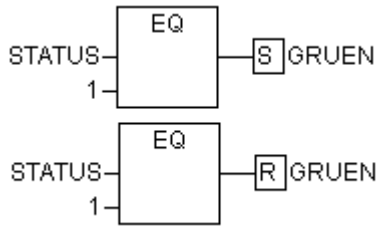
If an input is selected (Cursor Position 2), then this input will be negated.

If an output is selected (Cursor Position 4), then this output will be negated.

If a jump or a return is marked, then the input of this jump or return will be negated. A negation can be canceled through renewed negation.

'Extras' 'Set/Reset'

With this command you can define outputs as Set or Reset Outputs. A grid with Set Output is displayed with [S], and a grid with Reset Output is displayed with [R].



Set/Reset Outputs in FBD

An Output Set is set to TRUE, if the grid belonging to it returns TRUE. The output now maintains this value, even if the grid jumps back to FALSE. An Output Reset is set to FALSE, if the grid belonging to it returns FALSE. The output maintains its value, even if the grid jumps back to FALSE. With multiple executions of the command, the output will alternate between set, reset, and normal output.

'Extras' 'Zoom' Shortcut: <Alt>+<Enter>

With this command a selected POU is loaded into its editor (Cursor Position 3). If you are dealing with a POU from a library, then the library manager is called up, and the corresponding POU is displayed.

‘Extras’ ‘Open instance’

This command corresponds to the 'Project' 'Open instance' command.
Cutting, Copying, Pasting, and Deleting in FBD

The commands used to "Cut", "Copy", "Paste", and "Delete" are found under the "Edit" menu item.

If a line cross is selected (Cursor Position 5), then the assignments, jumps, or RETURNS located below the crossed line will be cut, deleted, or copied.

If an operator, a function, or a function block is selected (Cursor Position 3), then the selected object itself, will be cut, deleted, or copied, along with all of the branches dependent on the inputs, with the exception of the first branch. Otherwise, the entire branch located in front of the cursor position will be cut, deleted, or copied.

After copying or cutting, the deleted or copied part is located on the clipboard and can now be pasted, as desired. In order to do so, you must first select the pasting point.

Valid pasting points include inputs and outputs. If an operator, a function, or a function block has been loaded onto the clipboard (As a reminder: in this case all connected branches except the first are located together on the clipboard), the first input is connected with the branch before the pasting point. Otherwise, the entire branch located in front of the pasting point will be replaced by the contents of the clipboard. In each case, the last element pasted is connected to the branch located in front of the pasting point.

The following problem is solved by cutting and pasting: A new operator is pasted in the middle of a network. The branch located on the right of the operator is now connected with the first input, but must be connected with the second input. You can now select the first input and perform the command **"Edit" "Cut"**. Following this, you can select the second input and perform the command **"Edit" "Paste"**. This way, the branch is dependent on the second input.

The Function Block Diagram in the Online Mode

In the Function Block Diagram, breakpoints can only be set to networks. If a breakpoint has been set to a network, then the network numbers field will be displayed in blue. The processing then stops in front of the network where the breakpoint is located. In this case, the network numbers field will become red. Using stepping (single step), you can jump from network to network.

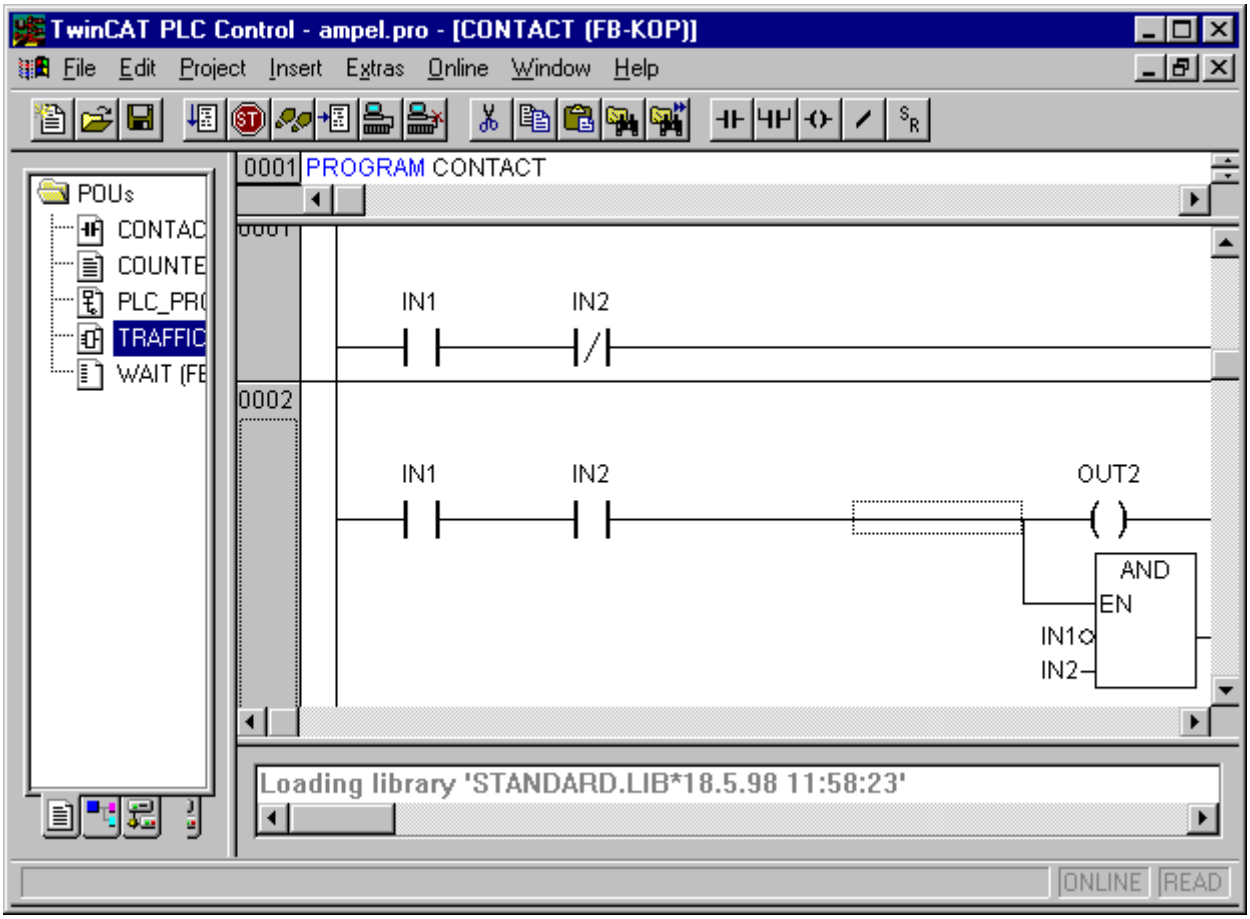
Exception: The current value is displayed for each variable.

Doubleclicking on a variable opens the dialog box for writing a variable. Here it is possible to change the present value of the variable. In the case of Boolean variables, no dialog box appears; these variables are toggled. The new value will turn red and will remain unchanged. If the **"Online" "Write values"** command is given, then all variables are placed in the selected list and are once again displayed in black. The flow control is run with the "**Online" "Flow control"** command. Using the flow control, you can view the present values that are being carried in the networks over the connecting lines. If the connecting lines do not carry Boolean values, then the value will be displayed in a specially inserted field. If the lines carry Boolean values, then they will be shaded blue in the event that they carry TRUE. By this means, you can accompany the flow of information while the PLC is running.

If you place the mouse pointer briefly above a variable, then the type and the comment about the variable will be displayed in a Tooltip.

5.7 Ladder Diagram Editor

This is how a POU written in the LD appears in the TwinCAT PLC Control editor:



POU in the Ladder Diagram

All editors for POUs consist of a declaration part and a body. These are separated by a screen divider.

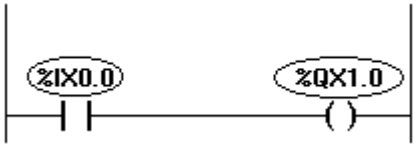
The LD editor is a graphic editor. The most important commands are found in the context menu (right mouse button or <Ctrl>+<F10>).

For information about the elements, see Ladder Diagram (LD).

Cursor Positions in the LD Editors

The following locations can be cursor positions, in which the function block and program accessing can be handled as contacts. POUs with EN inputs and other POUs connected to them are treated the same way as in the Function Block Diagram. Information about editing this network part can be found in the FBD Editor chapter.

1. Every text field (possible cursor positions framed in black)



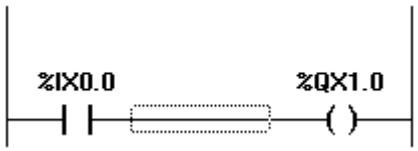
2. Every Contact or Function Block



3. Every Coil



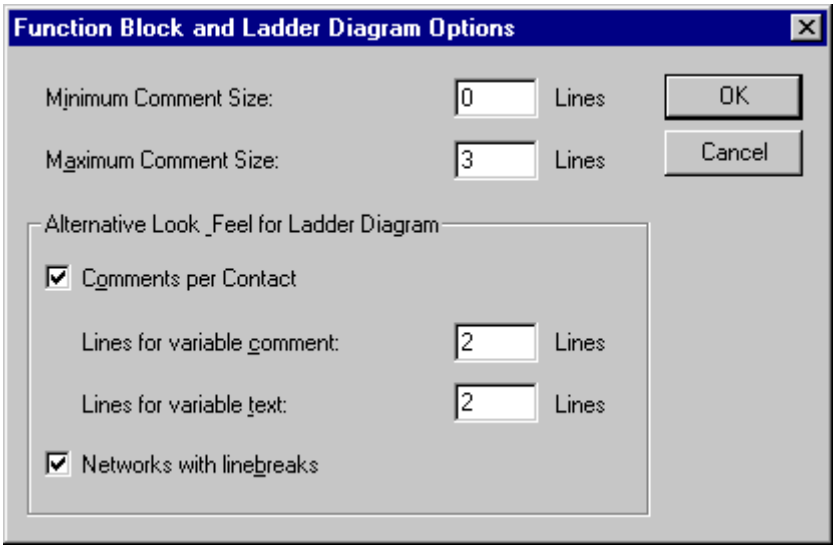
4. The Connecting Line between the Contacts and the Coils

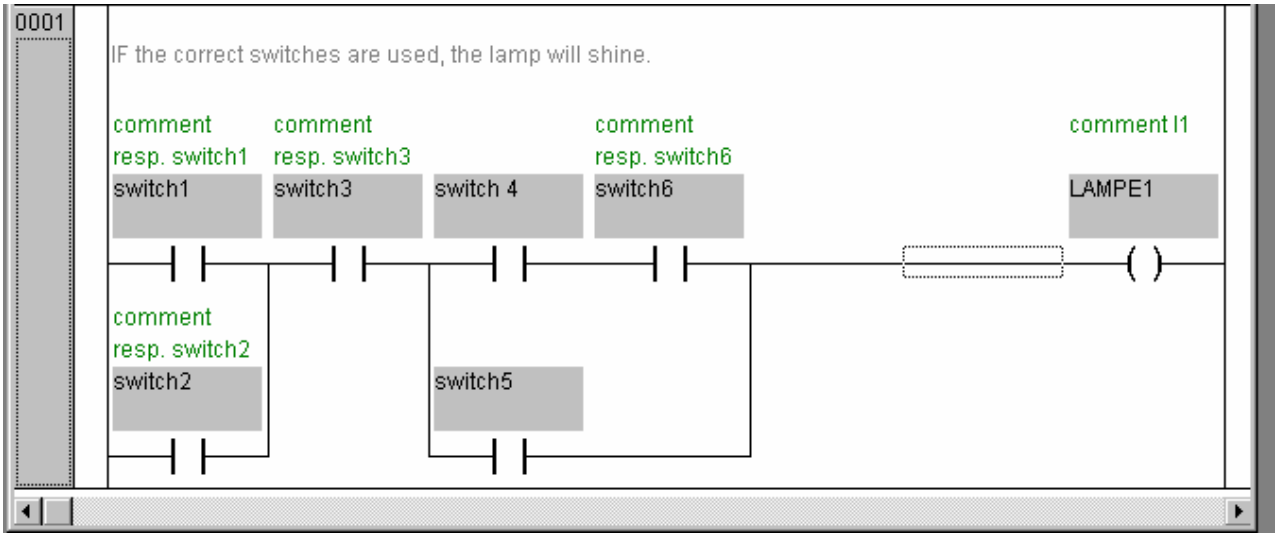


The Ladder Diagram uses the following menu commands in a special way:

'Insert' 'Contact' Shortcut: <Ctrl>+<O>

Use this command in the LD editor in order to insert a contact in front of the marked location in the network. If the marked location is a coil (Cursor Position 3) or the connecting line between the contacts and the coils (Cursor Position 4), then the new con-tact will be connected serially to the previous contact connection.
The contact is preset with the text "???". You can click on this text and change it to the desired variable or the desired constant. For this you can also use the Input Assistant.
You can activate the options Comments per Contact and Lines for variable comment in the dialog 'Function Block and Ladder Diagram Options' ('Extras' 'Options') to reserve a certain number of lines for the variable name. This might be useful, if long variable names are used, to keep the network short.
Also regard the option Networks with linebreaks, which you also can activate with the command **'Extras' 'Options'** in the dialog **'Function Block and Ladder Diagram Options'**.





'Insert' 'Parallel Contact' Shortcut: <Ctrl>+<R>

Use this command in the LD editor to insert a contact parallel to the marked position in the network. If the marked position is a coil (Cursor Position 3) or the connection between the contacts and the coils (Cursor Position 4), then the new contact will be connected in parallel to the entire previous contact connection. The contact is preset with the text "???". You can click on this text and change it to the desired variable or the desired constant. For this you can also use the Input Assistant.

'Insert' 'Function Block' Shortcut: <Ctrl>+<B>

Use this command in order to insert an operator, a function block, a function or a program as a POU. For this, the connection between the contacts and the coils (cursor position 4), or a coil (cursor position 3), must be marked. The new POU at first has the designation AND. If you wish, you can change this designation to another one. For this you can also use the Input Assistant. Both standard and selfdefined POUs are available.

The first input to the POU is placed on the input connection, the first output on the output connection; thus these variables must definitely be of type BOOL. All other in- and outputs of the POU are filled with the text „???“. These prior entries can be changed into other constants, variables or addresses. For this you can also use the Input Assistant.

'Insert' 'Coil' Shortcut: <Ctrl>+<L>

You can use this command in the LD editor to insert a coil in parallel to the previous coils. If the marked position is a connection between the contacts and the coils (Cursor Position 4), then the new coil will be inserted as the last. If the marked position is a coil (Cursor Position 3), then the new coil will be inserted directly above it. The coil is given the text "???" as a default setting. You can click on this text and change it to the desired variable. For this you can also use the Input Assistant.

POUs with EN Inputs

If you want to use your LD network as a PLC for calling up other POUs , then you must merge a POU with an EN input. Such a POU is connected in parallel to the coils. Beyond such a POU you can develop the network further, as in the Function Block Diagram. You can find the commands for insertion at an EN POU under the menu item **"Insert" "Insert at Blocks"**.

An operator, a function block, or a function with EN input performs the same way as the corresponding POU in the Function Block Diagram, except that its execution is controlled on the EN input. This input is annexed at the connecting line between coils and contacts. If this connection carries the information "On", then the POU will be evaluated.

If a POU has been created once already with EN input, then this POU can be used to create a network. This means that data from usual operators, functions, and function blocks can flow in an EN POU and an EN POU can carry data to such usual POUs.

If, therefore, you want to program a network in the LD editor, as in FBD, you only need first to insert an EN operator in a new network. Subsequently, from this POU, you can continue to construct from your network, as in the FBD editor. A network thus formed will perform like the corresponding network in FBD.

'Insert' 'Box with EN'

Use this command to insert a function block, an operator, a function or a program with EN input into a LD network. The marked position must be the connection between the contacts and the coils (Cursor Position 4) or a coil (Cursor Position 3).

The new function block is inserted in parallel to the coils and underneath them; it contains initially the designation AND. If you wish, you can change this designation to another one. From the Input Assistant dialog box that appears, you can select whether to insert a user-defined, or a standard (default) function block.

‘Insert’ ’Insert at blocks

With this command you can insert additional elements into a POU that has already been inserted (also a POU with EN input). The commands below this menu item can be executed at the same cursor positions as the corresponding commands in the Function Block Diagram (See Chapter 5.7).

With **Input** you can add a new input to the POU.

With **Output** you can add a new output to the POU.

With **POU**, you insert a new POU. The procedure is similar to that described under 'Insert' 'POU'.

With **Assign** you can insert an assignment to a variable. At first, this is shown by three question marks „???“, which you edit and replace with the desired variable. Input assistance is available for this purpose.

'Insert' 'Jump'

With this command you can insert a parallel jump in the selected LD editor, in parallel, at the end of the previous coils. If the incoming line delivers the value "On", then the jump will be executed to the indicated label.

The marked position must be the connection between the contacts and the coils (Cursor Position 4) or a coil (Cursor Position 3). The jump is present with the text "???". You can click on this text and make a change in the desired label.

'Insert' 'Return'

In the LD editor, you can use this command to insert a Return instruction in parallel at the end of the previous coils. If the incoming line delivers the value "On," then the processing of the POU in this network is broken off. The marked position must be the connection between the contacts and the coils(Cursor Position 4) or a coil (Cursor Position 3).

'Extras' 'Paste after'

Use this command in the LD editor to insert the contents of the clipboard as a serial contact after the marked position. This command is only possible if the contents of the clipboard and the marked position are networks comprised of contacts.

'Extras' 'Paste below' Shortcut <Ctrl>+<U>

Use this command in the LD editor to insert the contents of the clipboard as parallel contact below the marked position. This command is only possible if the contents of the clipboard and the marked position are networks comprised of contacts.

'Extras' 'Paste above'

Use this command in the LD editor to insert the contents of the clipboard as parallel contact above the marked position. This command is only possible if the contents of the clipboard and the marked position are networks comprised of contacts.

'Extras' 'Negate' Shortcut: <Ctrl>+<N>

Use this command to negate a contact, a coil, a jump or return instruction, or an input or output of EN POUs at the present cursor position (Cursor Position 2 and 3).

^Between the parentheses of the coil or between the straight lines of the contact, a slash will appear ((/) or |/|). If there are jumps, returns, or inputs or outputs of EN POUs , a small circle will appear at the connection, just as in the FBD editor.

The coil now writes the negated value of the input connection in the respective Boolean variable. Right at this moment, a negated contact switches the status of the input to the output, if the respective Boolean variable carries the value FALSE.

If a jump or a return is marked, then the input of this jump or return will be negated. A negation can be canceled through renewed negation.

'Extras' 'Set/Reset'

If you execute this command on a coil, then you will receive a Set Coil. Such a coil never overwrites the value TRUE in the respective Boolean variable. This means that once you have set the value of this variable to TRUE, it will always remain at TRUE. A Set Coil is designated with an "S" in the coil symbol.

If you execute this command once again, then you will be given a Reset Coil. Such a coil never overwrites the value FALSE in the respective Boolean variable. This means that once you have set the value of this variable to FALSE, it will always remain at FALSE. A Reset Coil is designated with an "R" in the coil symbol.

If you execute this command repeatedly, the coil will alternate between set, reset and normal coil.

‘Extras’ ‘Open instance’

This command corresponds to the 'Project' 'Open instance' command.

The Ladder Diagram in the Online Mode

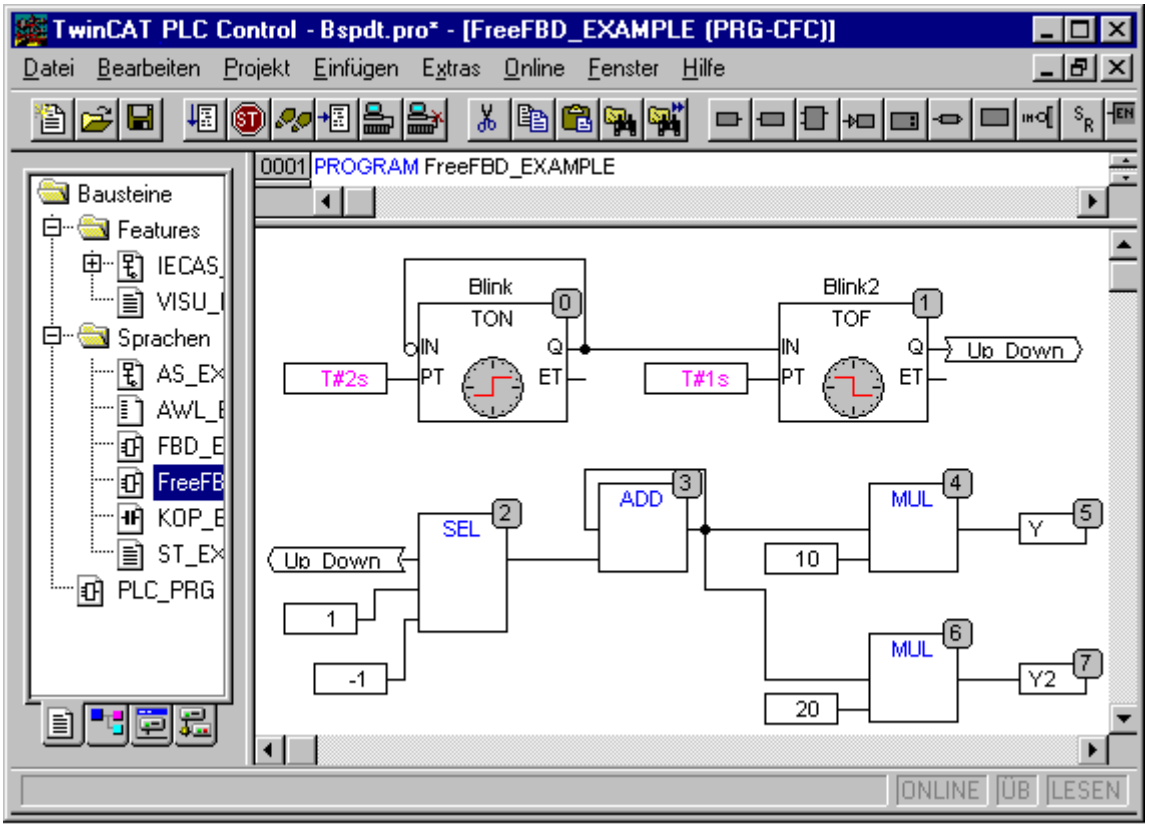
In Online mode, the contacts and coils in the Ladder Diagram that are in the "On" state are colored blue. Likewise, all lines over which the "On" is carried are also colored blue. At the inputs and outputs of function blocks, the values of the corresponding variables are indicated.

Breakpoints can only be set on networks; by using stepping, you can jump from network to network.

If you place the mouse pointer briefly above a variable, then the type and the comment about the variable will be displayed in a Tooltip.

5.8 Continuous Function Chart Editor (CFC)

It looks like a block which has been produced using the continuous function chart editor (CFC):

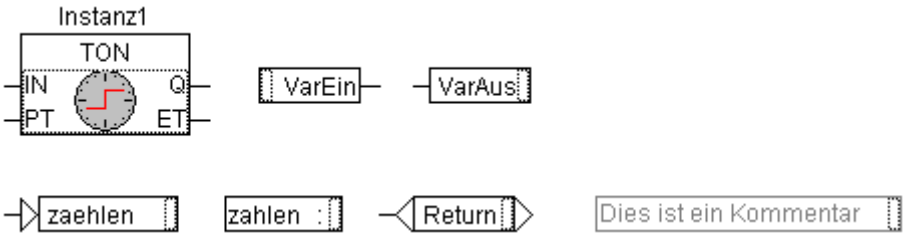


Editor for continuous function charts

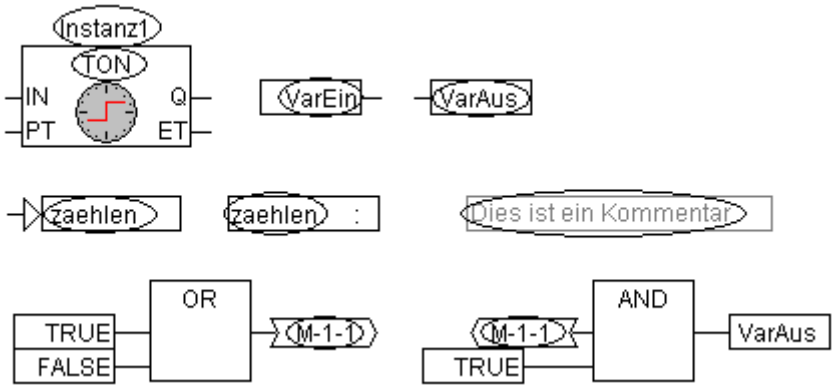
No snap grid is used for the continuous function chart editor so the elements can be placed anywhere. Elements of the sequential processing list include blocks, input, output, jump, label, return and comments. The inputs and outputs of these elements can be connected by dragging a connection with the mouse. The connecting line will be drawn automatically. The shortest possible connection line is drawn taking into account existing connections. The connecting lines are automatically adjusted when the elements are moved. If the case arises where a connecting line cannot be drawn simply because of lack of space, a red line will be shown between the input and the associated output instead. This line will be converted into a connecting line just as soon as space is available.
One advantage of the continuous function chart as opposed to the usual function block diagram FBD editor is the fact that feedback paths can be inserted directly.
The most important commands can be found in the context menu

Cursor positions in the continuous function chart editor CFC

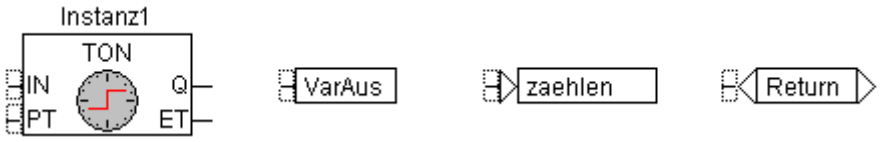
Each text is a possible cursor position. The selected text is shaded in blue and can be modified.
In all other cases the current cursor position is shown by a rectangle made up of points. The following is a list of all possible cursor positions with examples:
1. Trunks of the elements blocks, input, output, jump, label, return and comments.



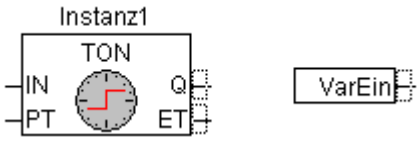
2. Text fields for the elements blocks, input, output, jump, label, return and comments as well as text fields for connection markers:



3. Inputs for the elements block, input, output, jump and return:



4. Outputs for the elements block and input:



'Insert' 'Block' Shortcut: <Ctrl>+<B>

This command can be used to paste in operators, functions, function blocks and programs. First of all, it is always pasted in as an "AND" operator. This can be converted by Selection and Overwrite of the text into every other operator, into every function, into every function block and every program. The input assistance serves to select the desired block from the list of supported blocks. If the new block has another minimum number of inputs, these will be attached. If the new block has a smaller highest number of inputs, the last inputs will be deleted.

'Insert' 'Input' Shortcut: <Ctrl>+<E>

This command is used to insert an input. The text offered "???" can be selected and replaced by a variable or constant. The input assistance can also be used here.

'Insert' 'Output' Shortcut: <Ctrl>+<A>

This command is used to insert an output. The text offered "???" can be selected and replaced by a variable. The input assistance can also be used here. The value which is associated with the input of the output is allocated to this variable.

'Insert' 'Jump' Shortcut: <Ctrl>+<J>

This command is used to insert a jump. The text offered "???" can be selected and replaced by the jump label to which the program should jump.
The jump label is inserted using the command 'Insert 'Label'

'Insert' 'Label' Shortcut: <Ctrl>+<L>

This command is used to insert a label. The text offered "???" can be selected and replaced by the jump label. In Online mode a RETURN label for marking the end of POU is automatically inserted.
The jump is inserted using the command '**Insert 'Jump'**

'Insert' 'Return' Shortcut: <Ctrl>+<R>

This command is used to insert a RETURN command.

Note that in Online mode a jump label with the name RETURN is automatically inserted in the first column and after the last element in the editor; in stepping, it is automatically jumped to before execution leaves the POU.

'Insert' 'Comment' Shortcut: <Ctrl>+<K>

This command is used to insert a comment.
You obtain a new line within the comment with <Ctrl> + <Enter>.

'Insert' 'Block Input' Shortcut: <Ctrl>+<U>

This command is used to insert a block input. The number of inputs is variable for many operators (e.g. ADD can have two or more inputs).
To increase the number of inputs for such an operator by one, the operator itself must be selected (Cursor position 1).

Insert' 'In-Pin', 'Insert' 'Out- Pin'

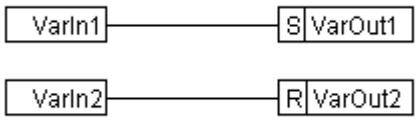
These commands are available as soon as a macro is opened for editing. They are used for inserting in- or out-pins as in- and outputs of the macro. They differ from the normal in- and outputs of POUs by the way they are displayed and in that they have no position index.

'Extras' 'Negate' Shortcut: <Ctrl>+<N>

This command is used to negate inputs, outputs, jumps or RETURN commands. The symbol for the negation is a small cross on the connection.
The input of the element block, output, jump or return is negated when it is selected (Cursor position 3)
The output of the element block or input is negated when it is selected (Cursor position 4).
A negation can be deleted by negating again.

'Extras' 'Set/Reset'

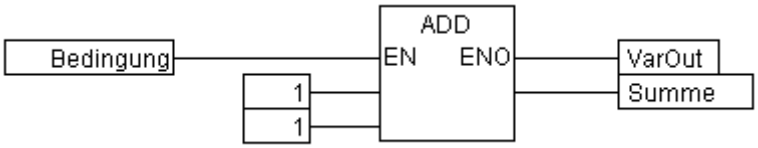
This command can only be used for selected inputs of the element output (Cursor position 3).
The symbol for Set is S and for Reset is R.



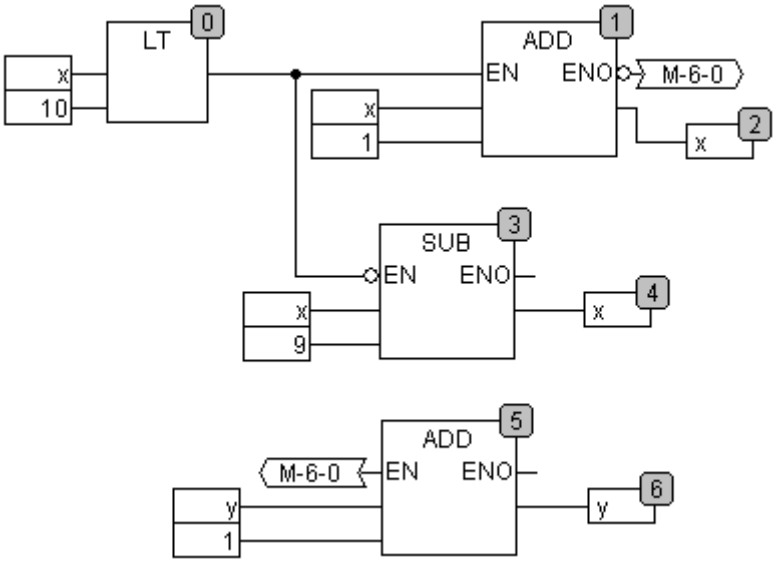
VarOut1 is set to TRUE, if VarIn1 delivers TRUE. VarOut1 retains this value, even when VarIn1 springs back to FALSE.
VarOut2 is set to FALSE, if VarIn2 delivers TRUE. VarOut2 retains this value, even when VarIn2 springs back to FALSE.
Multiple activation of this command causes the output to change between Set, Reset and the normal condition.

'Extras' 'EN/ENO' Shortcut: <Ctrl>+<O>

This command is used to give a selected block (Cursor position 3) an additional Boolean enable input EN (Enable In) and a Boolean output ENO (Enable Out).



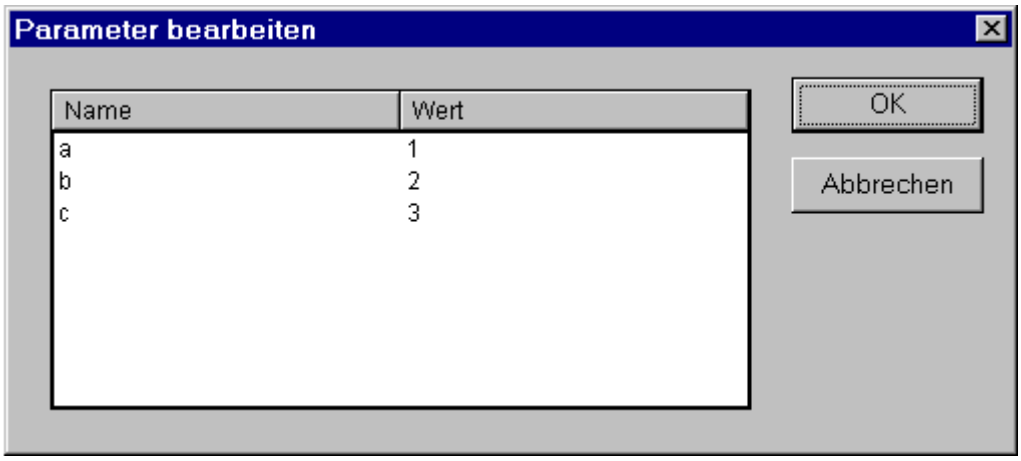
ADD is only executed in this example when the Boolean variable "Bedingung" (condition) is TRUE. VarOut is also set to TRUE after ADD has been executed. ADD will not be executed when the variable "Bedingung" (condition) is FALSE and VarOut retains its value FALSE. The example below shows how the value ENO can be used for further blocks.



x should initialized to 1 and y initialized to 0. The numbers in the right corner of the block indicate the order in which the commands are executed.
x will be increased by one until it reaches the value 10. This causes the output of the block LT(0) to deliver the value FALSE and SUB(3) and ADD(5) will be executed. x is set back to the value 1 and y is increased by 1. LT(0) is executed again as long as x is smaller than 10. y thus count the number of times x passes though the value range 1 to 10.

'Extras' 'Properties...'

Constant input parameters (VAR_INPUT CONSTANT) from functions and function blocks are not shown directly in the continuous function chart editor. These can be shown and their value can be changed when one selects the trunk of the block in question (Cursor position 1) and then selects the command ‚Extras' ‚Properties' or simply double clicks on the trunk. The dialog edit parameter opens:



The values of the constant input parameter (VAR_INPUT CONSTANT) can be changed. Here it is necessary to mark the parameter value in the column Value. Another mouse click or pressing on the space bar allows this to be edited. Confirmation of the change to the value is made by pressing the <Enter> key or pressing <Escape> rejects the changes. The button **OK** stores all of the changes which were made.

Selecting Elements

One clicks on the trunk of the element (Cursor position 1) to select it.
To mark more elements one presses the <Shift> key and clicks in the elements required, one after the other, or one drags the mouse with the left hand mousekey depressed over the elements to be marked.
The command **'Extras' 'Select all'** marks all elements at once.

Moving elements

One or more selected elements can be moved with the arrow keys as one is pressing on the <Shift> key. Another possibility is to move elements using a depressed left mousekey. These elements are placed by releasing the left mousekey in as far as they do not cover other elements or exceed the foreseen size of the editor. The marked element jumps back to its initial position in such cases and a warning tone sounds.

Copying elements

One or more selected elements can be copied with the command **'Edit' 'Copy'** and inserted with the command **'Edit' 'Paste'**.

Creating connections

An input of an element can be precisely connected to the output of another element. An output of an element can be connected to the inputs of a number of other elements.
There are a number of possibilities to connect the input of an element E2 with the output of an element E1.



Place the mouse on the output of element E1 (Cursor position 4), click with the left mousekey, hold the left mousekey down and drag the mouse cursor onto the input of element E2 (Cursor position 3) and let the left mousekey go. A connection is made from the output of element E1 to the mouse cursor during this dragging operation with the mouse.
Place the mouse on the input of element E2, click with the left mousekey, hold the left mousekey down and drag the mouse cursor onto the output of element E1 and let the left mousekey go.
Move one of the elements E1 or E2 (Cursor position 1) and place it in such a way by letting go of the left mousekey that the output of element E2 and the input of element E1 touch.
Where element E2 is a block with a free input, a connection can also be made by dragging the mouse from an output from E1 to the trunk of E2. A connection with the free input at the highest position on E2 will be created when the mousekey is released. In the case where block E2 does not have a free input but is an operator which can have an input added to it, a new input will be automatically generated.
The output and input of a block can be connected together (feedback path) by using this method. To establish a connection between two pins, click with the left mouse button on one pin, hold the button down and thus drag the connection to the desired pin, where you then release the button. If during the dragging of the connection extends outside working area of the editor, scrolling occurs automatically. For simple data types, type testing is carried out during the connection. If the types of the two pins are not compatible, the cursor changes to „Forbidden“. For complex data types, no testing takes place.

Deleting connections

There are a number of possibilities for removing the connection between the output of an element E1 and the input of an element E2:
Select the output of element E1 (Cursor position 4) and press the <Delete> key or execute the command **'Edit' 'Delete'.** Several connections will be removed at the same if the output of E1 is connected to more than one of inputs.
Select the input of element E2 (Cursor position 4) and press the <Delete> key or execute the command **'Edit' 'Delete'**.
Select the input of E2 with the mouse, hold the left mousekey depressed and drag the connection from the input to E2 away. The connection is removed when the left mousekey is released in a free area of the screen.

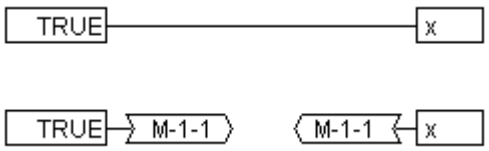
Changing connections

A connection between the output of an element E1 and the input of an element E2 can easily be changed into a connection between the output of element E1 and the input of element E3. The mouse is clicked on the input of E2 (Cursor position 3), the left mousekey is kept depressed, the mouse cursor is moved to the input of E3 and then released.

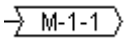
'Extras' 'Connection marker'

Connections can also be represented by a connector (connection marker) instead of a connecting line. Here the output and the associated input have a connector added to them which is given a unique name.
Where a connection already exists between the two elements which should now be represented by

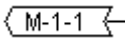
connectors, the output of the connecting line is marked (Cursor position 3) and the menu point ‚Extras' ‚Connection marker' is selected. The following diagram shows a connection before and after the selection of this menu point.



A unique name is given as standard by the program which begins with M, but which can be changed. The connector name is stored as an output parameter, but can be edited both at the input and at the output.

1. Edit the connectorname at the output: 

If the text in the connector is replaced, the new connector name is adopted by all associated connectors at the inputs. One cannot, however, select a name which already belongs to another connection marker since the uniqueness of the connector name would be violated.

2. Edit the connectorname at the input: 

If the text in the connector is replaced, the connection to the old connection marker will be deleted and a new connection will be created, or in other words only the name of another existing connection marker can be given.
Connections in connector representations can be converted to normal connections in that one marks the output of the connections (Cursor position 4) and again selects the menu point '**Extras' 'Connection marker'.**

Insert inputs/outputs “on the fly”

If exactly one input or output pin of an element is selected, then the corresponding input- or output- element can be directly inserted and its editor field filled with a string by entering the string at the keyboard.

Order of execution

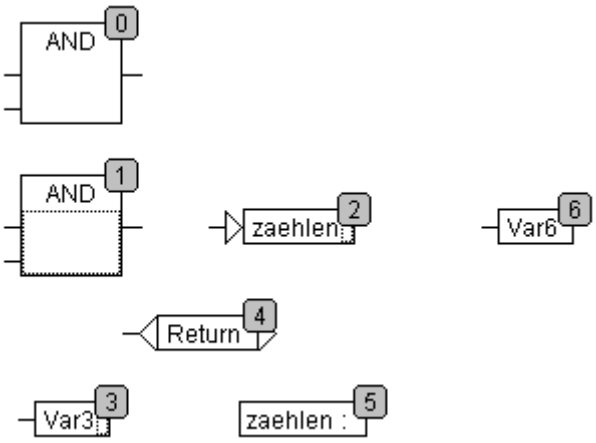
The elements block, output, jump, return and label each possess a number indicating the order in which they are executed. In this sequential order the individual elements are evaluated at run time.
When pasting in an element the number is automatically given according to the topological sequence (from left to right and from above to below). The new element receives the number of its topological successor if the sequence has already been changed and all higher numbers are increased by one.
The number of an element remains constant when it is moved.
The sequence influences the result and must be changed in certain cases.
If the sequence is displayed, the corresponding sequential execution number is shown in the upper right hand corner of the element.

'Extras' 'Order' 'Display'

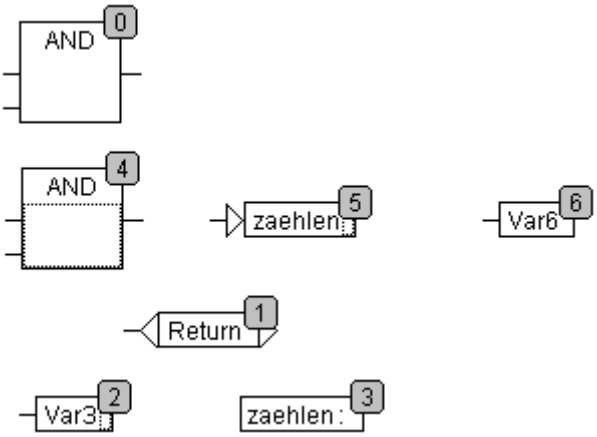
This command switches the display of the order of execution on and off. The default setting is to show it (recognized by a tick in front of the menu point).
The relevant order of execution number appears in the upper right hand corner for the elements block, output, jump, return and label.

'Extras' 'Order' 'Order topologically'

Elements are ordered in a topological sequence when the execution takes place from left to right and from above to below, that is the number increases from left to right and from above to below for topologically arranged elements. The connections are not relevant, only the location of the elements is important.
All **selected** elements are topologically arranged when the command **'Extras' 'Order' 'Order topologically'** is executed. All elements in the selection are taken out of the sequential processing list by this process. The elements are then entered into the remaining sequential processing list individually from bottom right through to upper left. Each marked element is entered into the sequential processing list before its topological successor, i.e. it is inserted before the element that in a topological sequencing would be executed after it, when all elements in the editor were sequenced according to a topological sequencing system. This will be clarified by an example.



The elements with numbers 1, 2 and 3 are selected. If the command **'Order topologically'** is selected the elements are first taken out of the sequential processing list. Var3, the jump and the AND-operator are then inserted again one after the other. Var3 is placed before the label and receives the number 2. The jump is then ordered and receives the number 4 at first but this then becomes 5 after the AND is inserted. The new order of execution which arises is:



When a newly generated block is introduced it will be placed by default in front of its topological successor in the sequential processing list.

'Extras' 'Order' 'One forwards'

With this command all selected elements with the exception of the element which is at the beginning of the sequential processing list are moved one place forwards in the sequential processing list.

'Extras' 'Order' 'One backwards'

With this command all selected elements with the exception of the element which is at the end of the sequential processing list are moved one place backwards in the sequential processing list.

'Extras' 'Order' 'To the beginning'

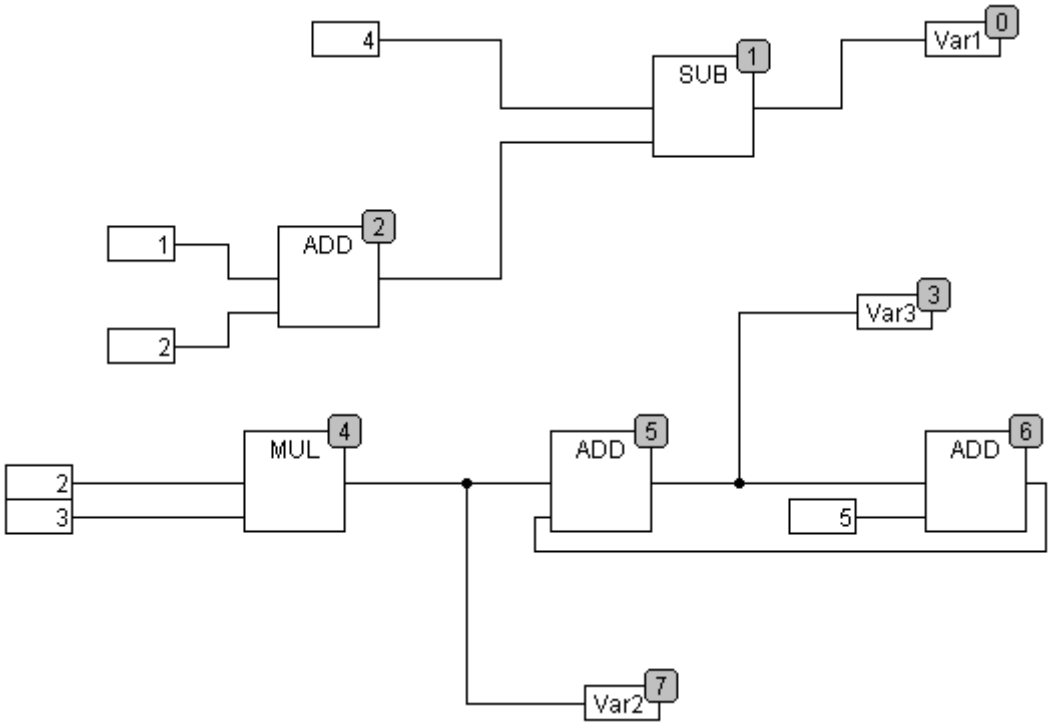
With this command all selected elements will be moved to the front of the sequential processing list whereby the order within the group of selected elements is maintained. The order within the group of unselected elements also remains the same.

'Extras' 'Order' 'To the end'

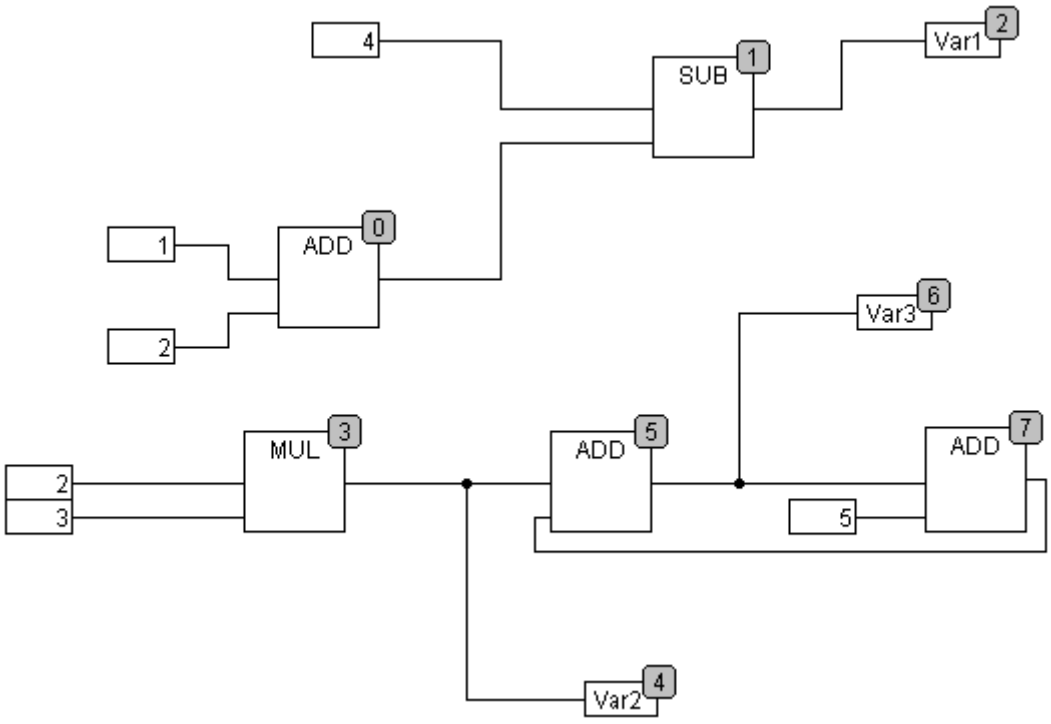
With this command all selected elements will be moved to the end of the sequential processing list whereby the order within the group of selected elements is maintained. The order within the group of unselected elements also remains the same.

'Extras' 'Order' 'Order everything according to data flow'

This command effects **all elements**. The order of execution is determined by the data flow of the elements and not by their position.
The diagram below shows elements which have been ordered topographically.



The following arrangement exists after selecting the command:



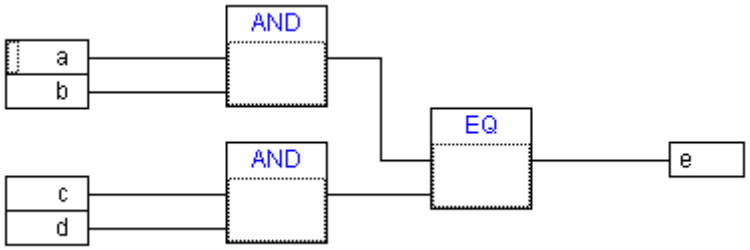
When this command is selected the first thing to happen is that the elements are ordered topographically. A new sequential processing list is then created. Based on the known values of the inputs, the computer calculates which of the as yet not numbered elements can be processed next. In the above "network" the block AND, for example, could be processed immediately since the values at its inputs (1 and 2) are known. Block SUB can only then be processed since the result from ADD must be known first, etc.
Feedback paths are inserted last.
The advantage of the data flow sequencing is that an output box which is connected to the output of a block

comes immediately after it in the data flow sequencing system which by topological ordering would not always be the case. The topological ordering can deliver another result in some cases than ordering by data flow, a point which one can recognise from the above example.

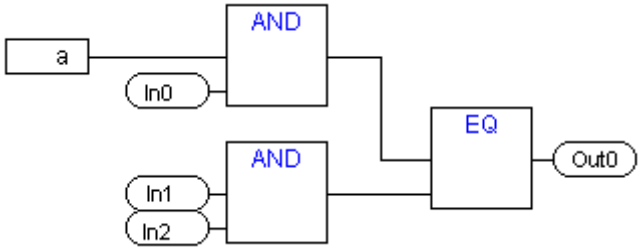
'Extras' 'Create macro'‘

With this command, several POUs that are selected at the same time can be assembled into a block, which can named as a macro. Macros only can be reproduced by Copy/Paste, whereby each copy becomes a separate macro whose name can be chosen independently. Macros are thus not references. All connections that are cut by the creation of a macro generate in- or out-pins on the macro. Connections to inputs generate an in-pin. The default name appears next to the pin in the form In<n>. For connections to outputs, Out<n> appears. Affected connections which had connection markers prior to the creation of the macro, retain the connection marker on the PIN of the macro.

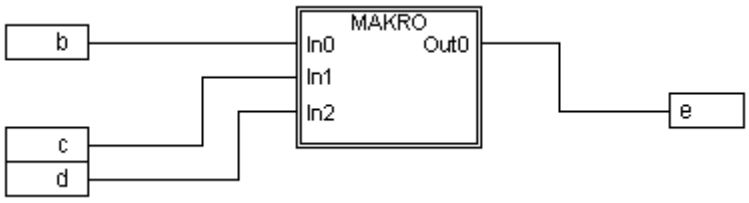
At first, a macro has the default name "MACRO". This can be changed in the Name field of the macro use. If the macro is edited, the name of the macro will be displayed in the title bar of the editor window appended to the POU name.
Example:
Selection:



Macro:



In the editor:



'Extras' 'Edit Macro’

By this command, or by double clicking on the body of the macro, the macro is opened for editing in the editor window of the associated POU. The name of the macro is displayed appended to the POU name in the title bar.
The pin boxes generated for the in- and outputs of the macro during creation can be handled like normal POU in- and outputs. They can also be moved, deleted, added, etc. They differ only in how they are displayed and have no position index. Pin boxes have rounded corners. The text in the pin-box matches the name of the pin in the macro display.
The order of the pins in the macro box follows the order of execution of the elements of the macro. A lower

order index before a higher one, higher pin before lower. The processing order within the macro is closed, in other words the macro is processed as a block, at the position of the macro in the primary POU. Commands for manipulating the order of execution therefore operate only within the macro.

'Extras' 'Expand macro'

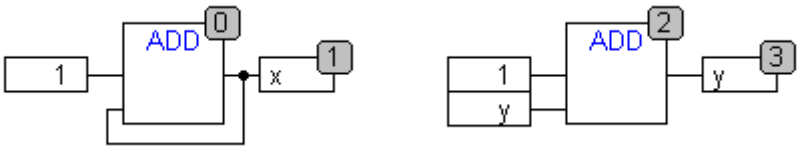
With this command, the selected macro is re-expanded and the elements contained in it are inserted in the POU at the macro's location. The connections to the pins of the macro are again displayed as connections to the in- or outputs of the elements. If the expansion of the macro can not occur at the location of the macro box for lack of space, the macro is displaced to the right and down until enough space is available.

'Extras' 'Back one macro level', 'Extras' 'Back all macro level'

These commands are also available in the toolbar, as soon as a macro is opened for editing. If macros are nested within one another, it is possible to switch to the next higher or to the highest display level.

Feedback paths

Feedback paths can only be displayed directly in the continuous function chart editor and not in the usual function block diagram editor. Here it should be observed that the output of a block always carries an internal intermediate variable. The data type of the intermediate variable results, for operators, from the largest data type of the inputs.
The data type of a constant is obtained from the smallest possible data type, that is the constant '1' adopts the data type SINT. If now an addition with feedback and the constant '1' is executed, the first input gives the data type SINT and the second is undefined because of the feedback. Thus the intermediate variable is also of the type SINT. The value of the intermediate variable is only then allocated to the output variable.
The diagram below shows an addition with feedback and an addition with a variable. The variables x and y should be of the type INT here.



There are differences between the two additions:
The variable y can be initialized with a value which is not equal to zero but this is not the case for intermediate variable for the left addition.
The intermediate variable for the left addition has the data type SINT while that on the right has the data type INT. The variables x and y have different values after the 129th call up. The variable x, although it is of the type INT, contains the value ?127 because the intermediate variable has gone into overflow. The variable y contains the value 129, on the other hand.

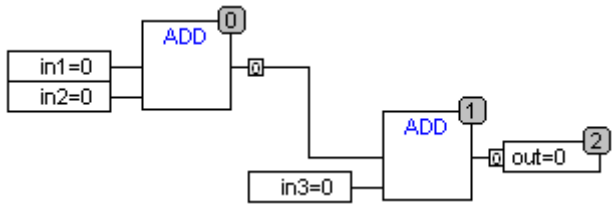
'Extras' 'Open instance'

This command corresponds to the **'Project' 'Open instance'** command.

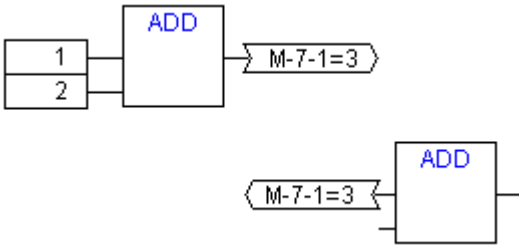
CFC in Online mode

Monitoring:

The values for inputs and outputs are displayed within the input or output boxes. Constants are not monitored. For non-boolean variables, the boxes are expanded to accommodated the values displayed. For boolean connections, the variable name as well as the connection are displayed in blue if the value is TRUE, otherwise they remain black. Internal boolean connections are also displayed Online in blue in the TRUE state, otherwise black. The value of internal non-boolean connections is displayed in a small box with rounded corners on the output pin of the connection.



PINs in macros are monitored like in- or output boxes.



Non-boolean connections with connection markers display their value within the connection marker. For boolean connections, the lines as well as the marker names are displayed in blue if the line is carrying the value TRUE, otherwise black.

Flow control:

When flow control is switched on, the connections that have been traversed are marked with the color selected in the project options.

Breakpoints:

Breakpoints can be set on all elements that also have a processing sequence order index. The processing of the program will be halted prior to execution of the respective element, that is for POUs and outputs before the assignment of inputs, for jump labels before execution of the element with the next index. The processing sequence index of the element is used as the breakpoint position in the Breakpoint dialog.

The setting of breakpoints on a selected element is accomplished with the F9 key or via the menu item 'Breakpoint on/off' in the 'Online' or 'Extras' menu or in the editor's context menu. If a breakpoint is set on an element, then this will be erased and reversed the next time the command 'Breakpoint on/off' is executed. In addition, the breakpoint on an element can be toggled by double-clicking on it.
Breakpoints are displayed in the colors entered in the project options.

RETURN label:

In Online mode, a jump label with the name „RETURN" is automatically generated in the first column and after the last element in the editor. This label marks the end of the POU and is jumped to when stepping just before execution leaves the POU. No RETURN marks are inserted in macros.

Stepping:

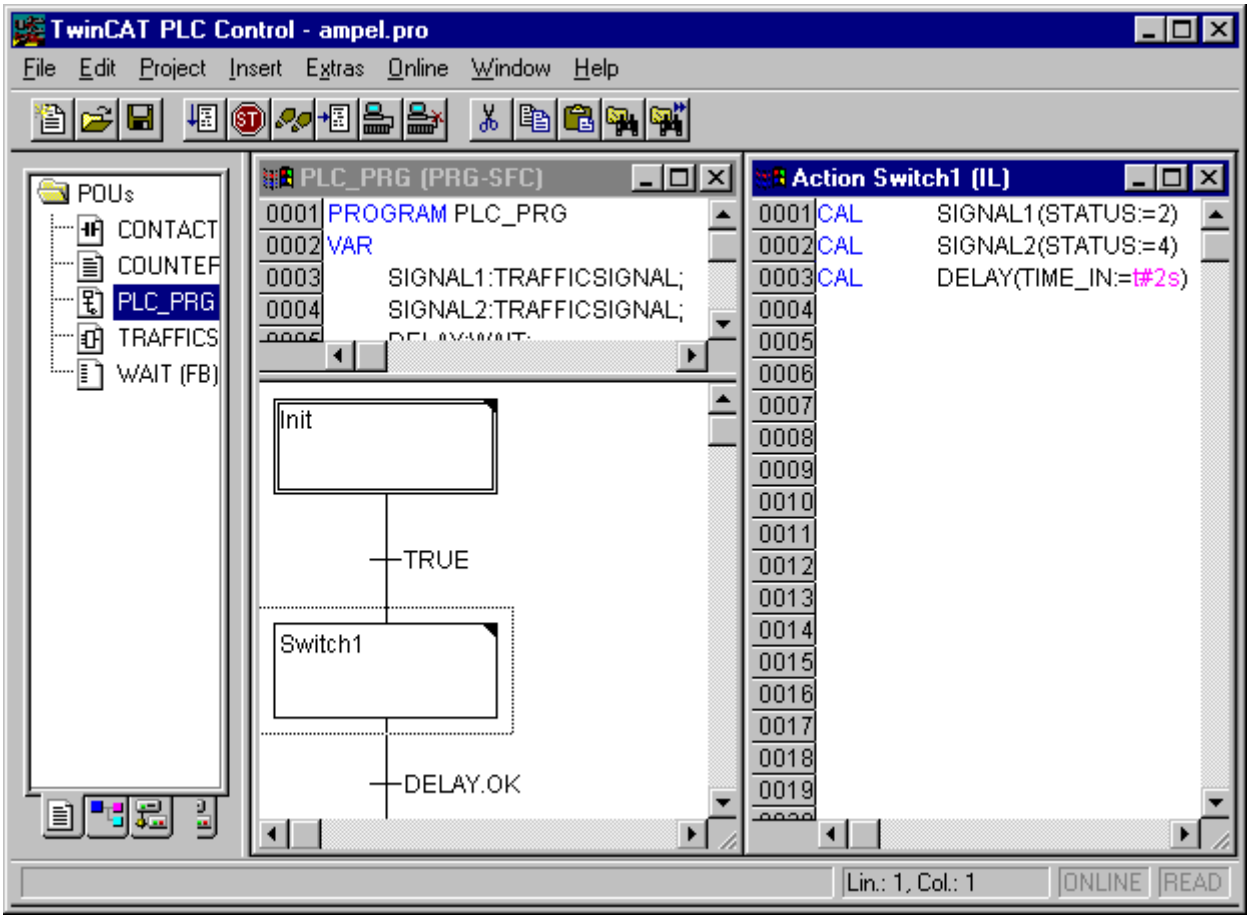
When using 'Step over' the element with the next-higher order index will always be jumped to. If the current element is a macro or a POU, then its implement branches when 'Step in' is in effect. If a 'Step over' is executed from there, the element whose order index follows that of the macro is jumped to.

'Extras' 'Zoom' Shortcut: <Alt> + <Enter>

With this command, the implementation of a POU can be opened if the POU is selected.

5.9 Sequential Function Chart Editor

This is how a POU written in the SFC appears in the TwinCAT PLC Control editor:



All editors for POUs consist of a declaration part and a body. These are separated by a screen divider. The sequential function chart editor is a graphic editor. The most important commands are found in the context menu (right mouse button). Tooltips show in Offline as well as in Online mode and in the zoomed state the full names or expressions of steps, transitions, jumps, jump labels, qualifiers or associated actions.

For information about the sequential function chart, see the chapter about Sequential Function Chart editor [▶ 152].

The editor for the sequential function chart must agree with the particulars of the SFC. In reference to these, the following menu items will be of service.

Marking Blocks in the SFC

A marked block is a bunch of SFC elements that are enclosed in a dotted rectangle. (In the example somewhat above, the step is marked Switch1.)
You can select an element (a step, a transition, or a jump) by pointing the mouse on this element and pressing the left mouse button, or you can use the arrow keys. In order to mark a group of several elements, press <Shift> for a block already marked, and select the element in the lower left or right corner of the group. The resulting selection is the smallest cohesive group of elements that includes both of these elements.

Observe that all commands can only be executed, if they do not contradict the conventions of the language.

'Insert' 'Step Transition (before)' <Ctrl>+<T>

This command inserts a step in the SFC editor followed by a transition in front of the marked block.

'Insert' 'Step Transition (after)' <Ctrl>+<E>

This command inserts a step in the SFC editor followed by a transition after the first transition in the marked block.

'Insert' 'Alternative Branch (right)' <Ctrl>+<A>

This command inserts an alternative branch in the SFC editor as a right branch of the marked block. For this the marked block must both begin and end with a transition. The new branch is then made up of one transition.

'Insert' 'Alternative Branch (left)'

This command inserts an alternative branch in the SFC editor as the left branch of the marked block. For this the marked block must both begin and end with a transition. The new branch is then made up of one transition.

'Insert' 'Parallel Branch (right)' <Ctrl>+<L>

This command inserts a parallel branch in the SFC editor as the right branch of the marked block. For this the marked block must both begin and end with a step. The new branch is then made up of one step. To allow jumps to the parallel branches that have been created, these must be provided with a jump label.

'Insert' 'Parallel Branch (left)'

This command inserts a parallel branch in the SFC editor as the left branch of the marked block. For this the marked block must both begin and end with a step. The new branch is then made up of one step. To allow jumps to the parallel branches that have been created, these must be provided with a jump label (see 'Extras' 'Add label to parallel branch').

'Insert' 'Jump' <Ctrl>+<U>

This command inserts a jump in the SFC editor at the end of the branch, to which the marked block belongs. For this the branch must be an alternative branch. The inserted text string 'Step' in the inserted jump can then be selected and replaced by the step name to be jumped to.

'Insert' 'Transition-Jump'

This command inserts a transition in the SFC editor, followed by a jump at the end of the selected branch. For this the branch must be a parallel branch. The inserted text string 'Step' in the inserted jump can then be selected and replaced by the step name or the jump label of a parallel branch to be jumped to.

'Insert' 'Add Entry-Action'

With this command you can add an entry-action to a step. An entry-action is only executed once, right after the step has become active. The entry-action can be implemented in a language of your choice.
A step with an entry-action is designated by an "E" in the bottom left corner. An entry-action cannot be defined to an IEC step.

'Insert' 'Add Exit-Action'

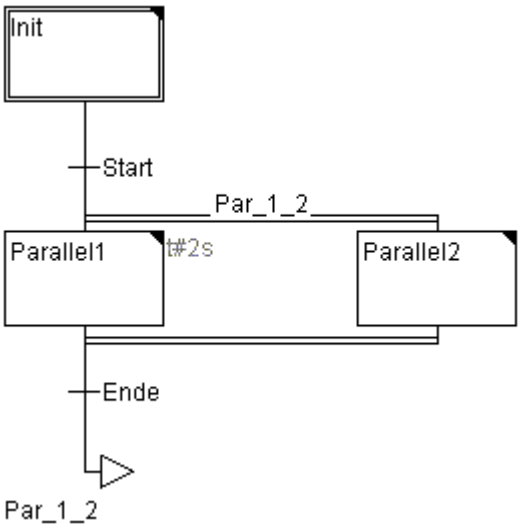
With this command you can add an exit-action to a step. An exit-action is only executed once, before the step is deactivated. The exit-action can be implemented in a language of your choice.
A step with an exit-action is designated by an "X" in the lower right corner. An exit-action can not be defined to an IEC step.

'Extras' 'Paste Parallel Branch (right)'

This command pastes the contents of the clipboard as a right parallel branch of the marked block. For this the marked block must both begin and end with a step. The contents of the clipboard must, likewise, be an SFC block that both begins and ends with a step.

'Extras' 'Add label to parallel branch'

In order to provide a newly inserted parallel branch with a jump label, the transition occurring before the parallel branching must be marked and the command 'Add label to parallel branch' must be executed. At that point, the parallel branch will be given a standard name consisting of „Parallel“ and an appended serial number, which can be edited according to the rules for identifier names. In the following example, "Parallel" was replaced by "Par_1_2" and the jump to the transition "End" was steered to this jump label.



Delete a label

A jump label can be deleted by deleting the label name.

'Extras' 'Paste After'

This command pastes the SFC block on the clipboard after the first step or the first transition of the marked block. (Normal copying pastes it in front of the marked block.) This will now be executed, if the resulting SFC structure is correct, according to the language norms.

'Extras' 'Zoom Action/Transition' Shortcut: <Alt>+<Enter>

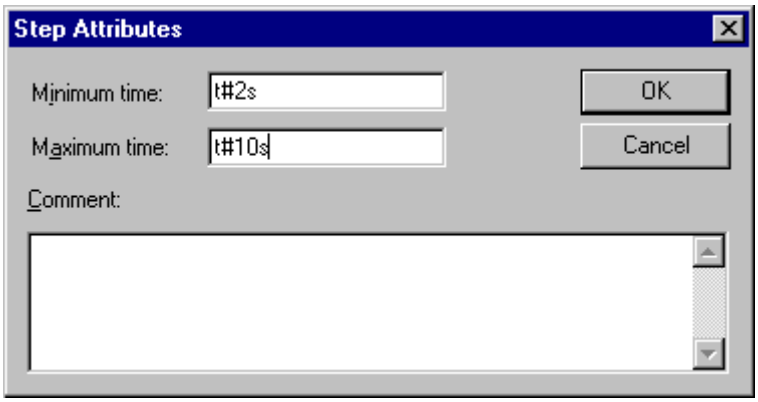
The action of the first step of the marked block or the transition body of the first transition of the market block is loaded into the editor in the respective language, in which it has been written. If the action or the transition body is empty, then the language must be selected, in which it has been written.

'Extras' 'Clear Action/Transition'

With this command you can delete the actions of the first step of the marked block or of the transitions body of the first transition.
If, during a step, you implement either only the action, the entry-action, or the exit-action, then the same will be deleted by the command. Otherwise a dialog box appears, and you can select which action or actions are to be deleted.
If the cursor is located in the action of an IEC step, then only this association will be deleted. If an IEC step with an associated action is selected, then this association will be deleted. During an IEC step with several actions, a selection dialog box will appear.

'Extras' 'Step Attributes'

With this command you can open a dialog box in which you can edit the attributes for the marked step.

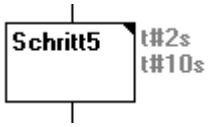


Dialog Box for Editing Step Attributes

You can take advantage of three different entries in the step attribute dialog box. Under **Minimum Time**, you enter the minimum length of time that the processing of this step should take. Under the **Maximum Time**, you enter the maximum length of time that the processing of this step should take. Note that the entries are of the **TIME** type, so you use a TIME constant (i.e. T#3s) or a variable of the TIME type.

Under **Comment** you can insert a comment to the step. In the 'Sequential function chart options' dialog which you open under **'Extras' 'Options'**, you can then enter whether comments or the time setting is displayed for the steps in the SFC editor. On the right, next to the step, either the comment or the time setting will appear.

If the maximum time is exceeded, SFC flags [▶ 156] are set which the user can query.



The example shows a step whose execution should last at least two, and at the most, ten seconds. In Online mode, there is, in addition to these two times, a display of how long the step has already been active.

SFC-Flags

If a step is active in SFC for longer than its attributes state, some special flags are set. There are also variables which can be set in order to control the program flow in the sequential function chart. To use the flags it is necessary, globally or locally, to declare them as output or input variables.

SFCEnableLimit

This variable is of the type BOOL. When it has the value TRUE, the timeouts of the steps will be registered in SFCError. Other timeouts will be ignored.

SFCInit

When this boolean variable has the value TRUE the sequential function chart is set back to the Init step. The other SFC flags are reset too (initialization). The Init step remains active, but is not executed, for as long as the variable has the value TRUE. It is only when SFCInit is again set to FALSE that the block can be processed normally.

SFCReset:

This variable, of type BOOL, behaves similarly to SFCInit. Unlike the latter, however, further processing takes place after the initialization of the Init step. Thus for example the SFCReset flag could be re-set to FALSE in the Init step.

SFCQuitError:

Execution of the SFC diagram is stopped for as long as this boolean variable has the value TRUE whereby a possible timeout in the variable SFCError is reset. All previous times in the active steps are reset when the variable again assumes the value FALSE.

SFCPause

Execution of the SFC diagram is stopped for as long as this boolean variable has the value TRUE.

SFCError:

This Boolean variable is TRUE when a timeout has occurred in a SFC diagram. If another timeout occurs in a program after the first one, it will not be registered unless the variable SFCError is reset first.

SFCTrans:

This boolean variable takes on the value TRUE when a transition is actuated.

SFCErrorStep:

This variable is of the type STRING. If SFCError registers a timeout, in this variable is stored the name of the step which has caused the timeout.

SFCErrorPOU:

This variable of the type STRING contains the name of the block in which a timeout has occurred.

SFCCurrentStep:

This variable is of the type STRING. The name of the step is stored in this variable which is active, independently of the time monitoring. In the case of simultaneous sequences the step is stored in the branch on the outer right.
No further timeout will be registered if a timeout occurs and the variable SFCError is not reset again.

SFCErrorAnalyzation:

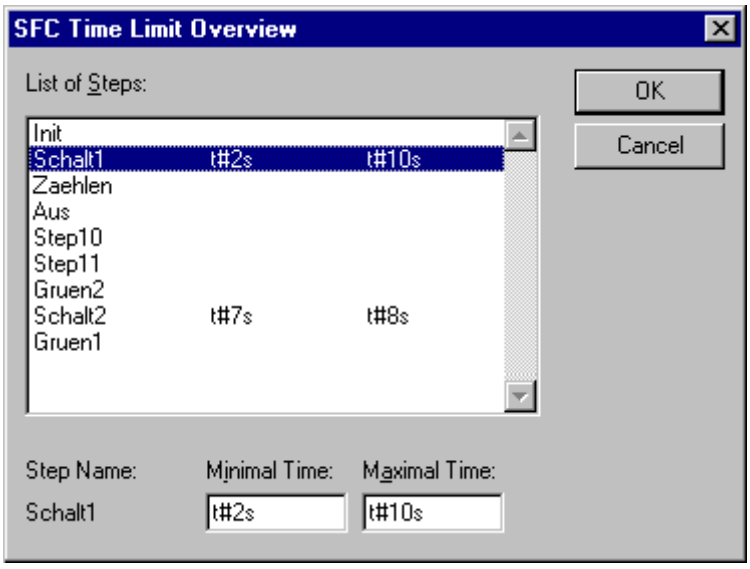
This variable, of type STRING, provides the transition expression as well as every variable in an assembled expression which gives a FALSE result for the transition and thus produces a timeout in the preceding step. A requirement for this is declaration of the SFCError flag, which registers the timeout. SFCErrorAnalyzation refers back to a function called AppedErrorString in the **TcSystem.Lib** library. The output string separates multiple components with the symbol “|”.

SFCTip, SFCTipMode:

This variables of type BOOL allow inching mode of the SFC. When this is switched on by SFCTipMode=TRUE, it is only possible to skip to the next step if SFCTip is set to TRUE. As long as SFCTipMode is set to FALSE, it is possible to skip even over transitions.

'Extras' 'Time Overview'

With this command you can open a window in which you can edit the time settings of your SFC steps:



Time Boundaries Overview for a SFC POU

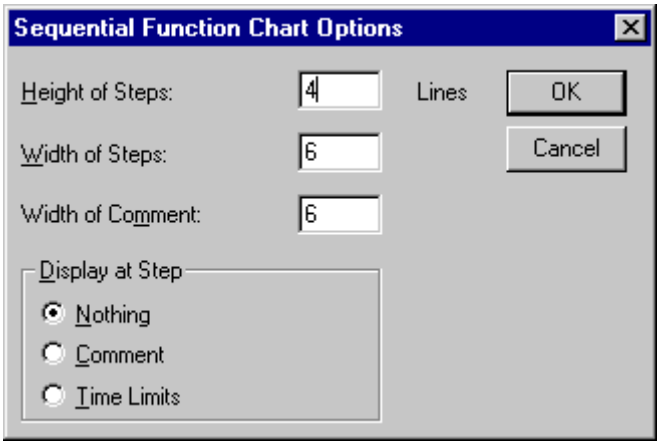
In the time boundaries overview, all steps of your SFC POU are displayed. If you have entered a time boundary for a step, then the time boundary is displayed to the right of the step (first, the lower limit, then the upper limit). You can also edit the time boundaries. To do so, click on the desired step in the overview. The **name of the step** is then shown below in the window. Go to the **Minimum Time** or **Maximum Time** field, and enter the desired time boundary there.

Note that the entries are of the **TIME** type, so you use a TIME constant (i.e. T#3s) or a variable of the TIME type. If you close the window with OK, then all of the changes will be stored.

In the example, steps 2 and 6 have a time boundary. Shift1 lasts at least two, and at most, ten seconds. Shift2 lasts at least seven, and at most, eight seconds.

'Extras' 'Options'

With this command you open a dialog box in which you can set different options for your SFC POU.



In the SFC Options dialog box you can undertake five entries. Under **Step Height**, you can enter how many lines high an SFC step can be in your SFC editor. 4 is the standard setting here. Under **Step Width**, you can enter how many columns wide a step should be. 6 is the standard setting here. The **Comment Width** defines the number of columns, which are shown, if you let the step shown with the comment.

You can also preset the **Display at Step**. With this, you have three possibilities: You can either have **Nothing** displayed, or the **Comment**, or the **Time Limits**. The last two are shown the way you entered them in **"Extras" "Step Attributes"**.

'Extras' 'Associate Action'

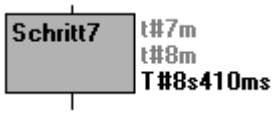
With this command actions and Boolean variables can be associated with IEC steps. To the right of, and next to the IEC step, an additional divided box is attached, for the association of an action. It is preset in the left field with the qualifier "N" and the name "Action." Both presets can be changed. For this you can use the Input Assistant.
New actions for IEC steps are created in the Object Organizer for an SFC POU with the **"Project" "Add Action"** command.

'Extras' 'Use IEC-Steps'

If this command is activated (denoted by a check in front of the menu item and a printed symbol in the Tool bar), then IEC steps will be inserted instead of the simplified steps upon insertion of step transitions and parallel branches.
If this option is switched on, the Init step is set as an IEC step when you create a new SFC POU.

The Sequential Function Chart in the Online Mode

With the sequential function chart editor in Online mode, the currently-active steps will be displayed as blue steps (black in the example). If you have set it under "Extras" "Options", then the time management is depicted next to the steps. Under the lower and upper bounds that you have set, a third time indicator will appear from which you can read how long the step has already been active.



In the picture above the step depicted has already been active 8 seconds and 410 milliseconds. The step must, however, be active for at least 7 minutes before the step will be left.

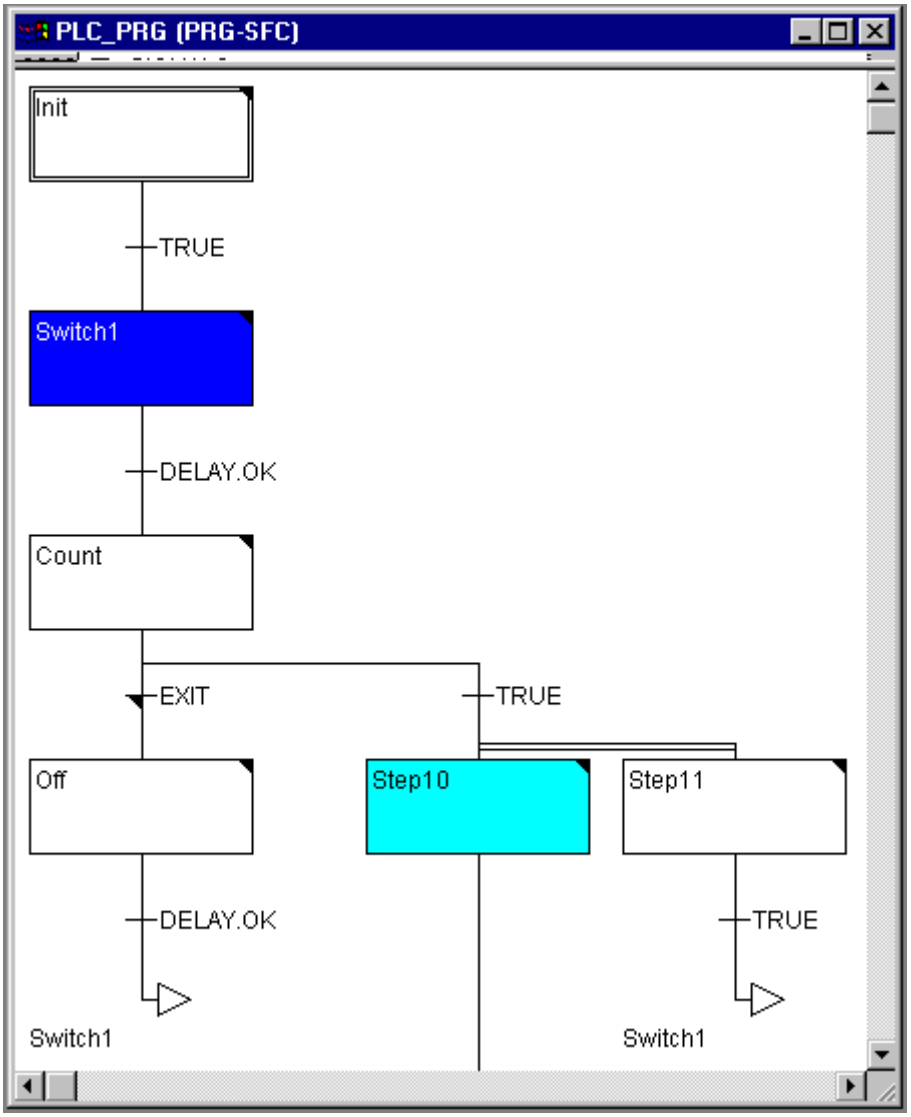
With **"Online" "Toggle Breakpoint"** , a breakpoint is set at a step, or in an action at the locations allowed by the language in use. Processing then stops prior to execution of this step or before the location of the action in the program. Steps or program locations where a breakpoint is set are marked in light blue.

If several steps are active in a parallel branch, then the active step whose action will be processed next is displayed in red.
If IEC steps have been used, then all active actions in Online mode will be displayed in blue.

With the command **'Online' 'Step over'** it is stepped always to the next step which action is executed.
If the current location is:

- a step in the linear processing of a POU or a step in the rightmost parallel branch of a POU, execution returns from the SFC POU to the caller. If the POU is the main program, the next cycle begins.
- a step in a parallel branch other than the rightmost, execution jumps to the active step in the next parallel branch.
- the last breakpoint location within a 3S action, execution jumps to the caller of the SFC.
- the last breakpoint location within an IEC action, execution jumps to the caller of the SFC.
- the last breakpoint position within an input action or output action, execution jumps to the next active step.

With **'Online' 'Step in'** even actions can be stepped into. If an input, output or IEC action is to be jumped into, a breakpoint must be set there. Within the actions, all the debugging functionality of the corresponding editor is available to the user.
If you rest the mouse cursor for a short time on a variable in the declaration editor, the type, the address and the comment of the variable will be displayed in a tooltip



Sequential Function Chart in the Online Mode with an Active Step (Shift1) and a Breakpoint (Step10)

⚠ CAUTION

Undefined state

If you rename a step and perform an Online Change while this step is active, the program will be stopped in undefined status.

Processing order of elements in a sequence:

1. First, all Action Control Block flags in the IEC actions that are used in this sequence are reset (not, however, the flags of IEC actions that are called within actions).
2. All steps are tested in the order which they assume in the sequence (top to bottom and left to right) to determine whether the requirement for execution of the output action is provided, and this is executed if that is the case.
3. All steps are tested in the order which they assume in the sequence to determine whether the requirement for the input action is provided, and this is executed if that is the case.
4. For all steps, the following is done in the order which they assume in the sequence:
   - If applicable, the elapsed time is copied into the corresponding step variable.
   - If applicable, any timeout is tested and the SFC error flags are serviced as required.
   - For non-IEC steps, the corresponding action is now executed.
5. IEC actions that are used in the sequence are executed in alphabetical order. This is done in two passes through the list of actions. In the first pass, all the IEC actions that are deactivated in the current cycle are executed. In the second pass, all the IEC actions that are active in the current cycle are executed.
6. Transitions are evaluated: If the step in the current cycle was active and the following transition returns TRUE (and if applicable the minimum active time has already elapsed), then the following step is activated.

The following must be noted concerning implementation of actions:

It can come about that an action is carried out several times in one cycle because it is associated with multiple sequences. (For example, an SFC could have two IEC actions A and B, which are both implemented in SFC, and which both call IEC action C; then in IEC actions A and B can both be active in the same cycle and furthermore in both actions IEC action C can be active; then C would be called twice).
If the same IEC action is used simultaneously in different levels of an SFC, this could lead to undesired effects due to the processing sequence described above. For this reason, an error message is issued in this case.
It can possibly arise during processing of projects created with older versions of TwinCAT PLC.



In monitoring expressions (e.g. A AND B) in transitions, only the „Total value“ of the transition is displayed.

6 Resources

In the Resources register card of the Object Organizer, there are objects for configuring and organizing your project and for keeping track of the values of the variables:

- Global Variables [▶ 161], that can be utilized in the entire project
- Alarm Configuration [▶ 164] for configuring of alarm classes and alarm groups which can be used for example in the visualization.
- Library Manager [▶ 252] for managing all libraries included in the project
- Log [▶ 113] for storing in chronological order actions that occur during an Online session.
- PLC Configuration [▶ 172] for configuring your hardware
- Sampling Trace [▶ 175] for graphic logging of variable values
- Task Configuration [▶ 173] for controlling your program control via tasks
- Watch- and Recipe Manager [▶ 179] for indicating and presetting variable values
- Workspace [▶ 48] Overview of all settings of project options

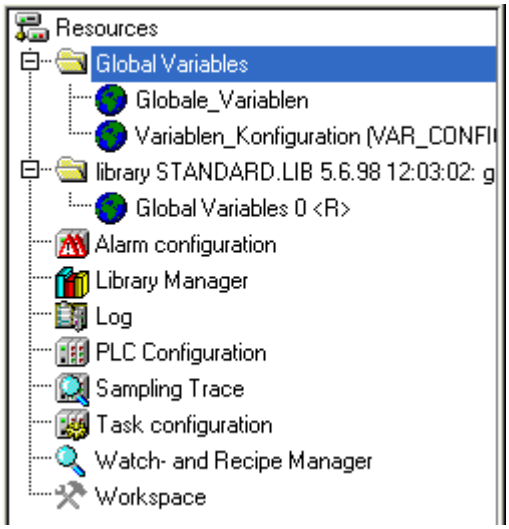


Fig. 1: Resources

6.1 Global Variables

In the Object Organizer, the Resources tab in the Global Variables folder contains three objects by default (the preassigned names of the objects are in brackets):

- Global Variable List
- Variable configuration

All variables defined in these objects are known throughout the project. If the Global Variables folder is not expanded (plus sign in front of the folder), open it by double-clicking or pressing <Enter> in the line. Select the corresponding object. The 'Edit Object' command opens a window with the global variables defined so far.

Several variable lists

If you have declared a large number of global variables and you want to better structure your global variable list, then you can create additional variable lists. Normal global variables (**VAR_GLOBAL**) and variable configurations (**VAR_CONFIG**) must be defined in separate objects. In the Object Organizer, select the Global Variables folder or one of the existing objects with global variables and execute the command **'Project' 'Insert Object'**. Give the object an appropriate name in the dialog box that appears. This creates another object with the keyword VAR_GLOBAL. If you want to have an object with variable configuration, change the keyword to VAR_CONFIG accordingly.

Global variables

Global variables can be declared as "normal" variables, constants or remanent variables that are known throughout the project.

Creating a global variable list

To create a new global variable list, select the entry '**Global variables**' in the Object Organizer at the **Resources** or a global variable list already created there. If you then select the command '**Project**' '**Object**' '**Insert**', the Global Variable List dialog opens.

By the way, this dialog opens to display an already made configuration of the global variable list marked in the Object Organizer also with the command'**Project**' '**Object**' '**Properties**'.

Editing the lists for remanent global variables

If it is supported by the runtime system, it is possible to work with remanent variables.

There are two types of remanent global variables

Persistent and retain variables are retained in the event of a controlled shutdown of the runtime system or an 'online' 'reset cold' or a download.

Persistent variables are not automatically retain variables!

Remanent variables additionally get the keyword RETAIN or PERSISTENT...

Syntax:

```
VAR_GLOBAL RETAIN
(* Variablendeklarationen *)
END_VAR
```

```
VAR_GLOBAL PERSISTENT
(* Variablendeklarationen *)
END_VAR
```

Global constants

Global constants are additionally assigned the keyword **CONSTANT**.

Syntax:

```
VAR_GLOBAL CONSTANT
(* Variablendeklarationen *)
END_VAR
```

Variable configuration

In function blocks, addresses for inputs and outputs can be specified for variables defined between the keywords VAR and END_VAR which are not completely defined. Addresses that are not completely defined are marked with an asterisk.

Sample:

```
FUNCTION_BLOCK locio
VAR
    loci AT %I*: BOOL := TRUE;
    loco AT %Q*: BOOL;
END_VAR
```

Two local I/O variables are defined here, a local-In (%I*) and a local-Out (%Q*).

If you want to configure local I/Os, the object **Variable_Configuration** is available by default for variable configuration in the Object Organizer in the **Resources** tab. The object can be renamed and additional objects can be created for the variable configuration.

The variable configuration editor works like the declaration editor. Variables for local I/O configuration must be placed between the keywords VAR_CONFIG and END_VAR. The name of such a variable consists of a complete instance path, where the individual function block and instance names are separated by dots. The

declaration must contain an address whose class (input/output) corresponds to that of the address not fully specified (%I*, %Q*) in the function block. The data type must also match the declaration in the function block.

Configuration variables whose instance path is not valid because the instance does not exist are marked as errors. Conversely, an error is also reported if no configuration exists for an instance variable. To get a complete list of all required configuration variables, the menu command '**All instance paths**' in the menu **'Insert'** can be used.

Sample:

In a program there is the following definition of function blocks:

```
PROGRAM PLC_PRG
VAR
    Hugo: locio;
    Otto: locio;
END_VAR
```

Then a correct variable configuration looks like this:

```
VAR_CONFIG
    PLC_PRG.Hugo.loci AT %IX1.0 : BOOL;
    PLC_PRG.Hugo.loco AT %QX0.0 : BOOL;
    PLC_PRG.Hugo.loci AT %IX1.0 : BOOL;
    PLC_PRG.Otto.loco AT %QX0.3 : BOOL;
END_VAR
```

Two local I/O variables are defined here, a local-In (%I*) and a local-Out (%Q*).



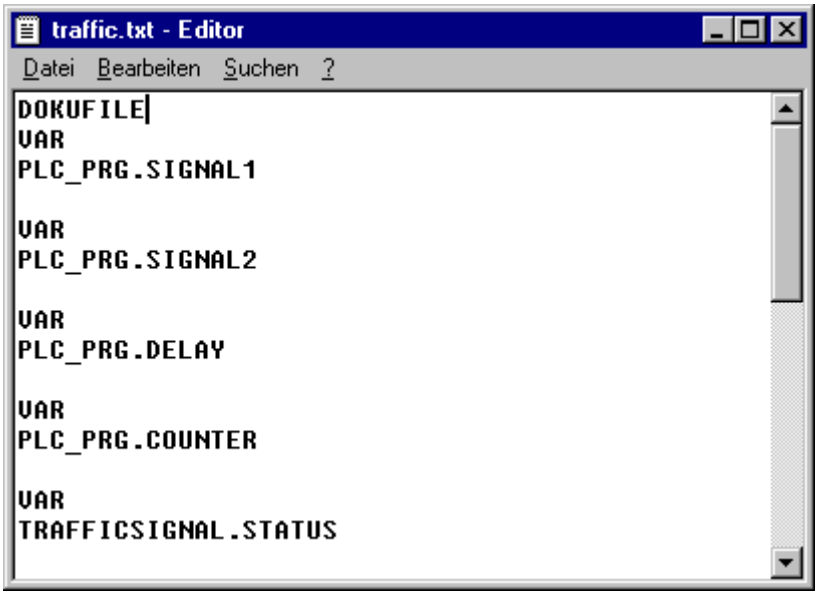
An output that is used in the variable configuration must not be described in the project directly or via a variable (AT declaration), since this is not observed.

'Insert''All Instance Paths'

This command creates a VAR_CONFIG - END_VAR function block containing all instance paths present in the project. Existing declarations are not re-inserted to preserve existing addresses. This menu item is available in the variable configuration window when the project is compiled ('**Project''Rebuild all**').

Document frame

If you need to document a project several times, for example with German and with English comments, or if you want to document several similar projects that use the same variable names, then you can save yourself a lot of work by creating a document frame with the command '**Extras' 'Make Docuframe file'.** You can load the created file into any text editor and edit it. The file begins with the line **DOKUFILE**, then follows a listing of the project variables, with three lines given for each variable, a line **VAR**, which indicates when a new variable is coming, then a line with the name of the variable, and finally an empty line. You can now replace this line with a comment about the variable. Simply delete variables from the text that you do not want to document. You can create as many document frames as you want for your project.



Windows Editor with document frame

To use a document frame, enter the command **'Extras' 'Link Docu File'**. If you now document the entire project, or print parts of your project, then the comment that you created in the document frame is inserted in the program text for all variables. This comment appears only in the printout!

'Extras' 'Make Docuframe File'

Use this command to create a document frame. The command is available when a Global Variable object is selected. The dialog for saving files under a new name opens. In the field for the file name the extension *.txt is already entered. Select any name. A text file is now created in which all variables of your project are listed.

'Extras' 'Link Docu File'

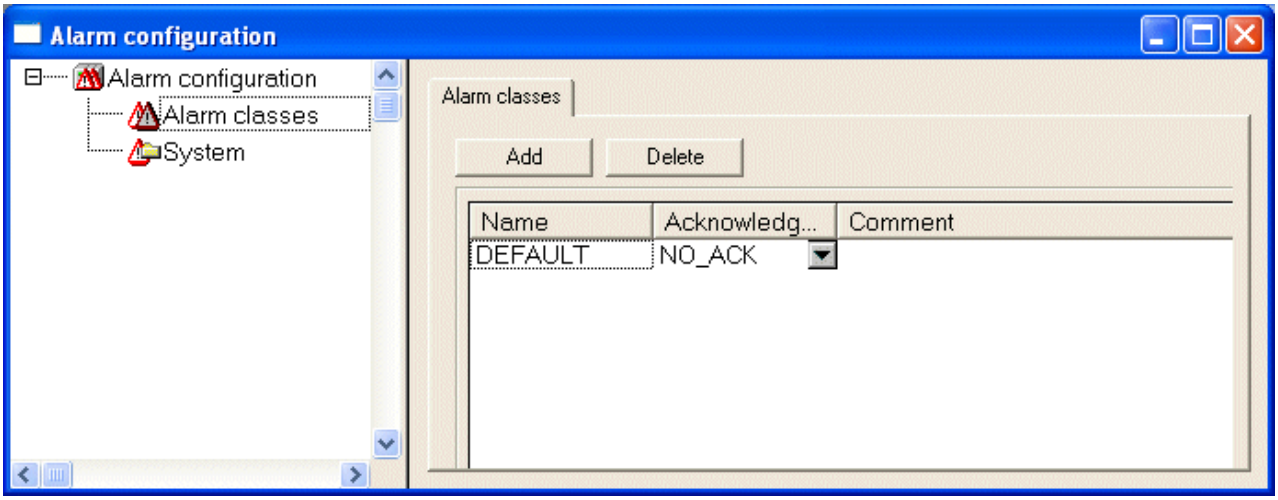
Use this command to select a document frame. The dialog for opening files opens. Select the desired document frame and press OK. If you now document the entire project, or print parts of your project, then the comment that you created in the document frame is inserted in the program text for all variables. This comment appears only in the printout! To create a document frame use the command **'Extras' 'Make Docuframe File'.**

6.2 Alarm Configuration

The alarm system integrated in TwinCAT PLC Control allows detecting critical process states, recording them and visualizing them for the user with the aid of a visualization element. he alarm handling can be done in TwinCAT PLC Control or alternatively in the PLC. For alarm handling in the PLC please see the target settings category 'Visualization'.
IThe dialogs for the 'Alarm configuration' are available in the 'Ressources' tab.
Here you define Alarm classes and Alarm groups. An alarm class serves for the typing of an alarm, that means it assigns certain parameters to the alarm. An alarm group serves for the concrete configuration of one or several alarms (which get assigned a certain class and further parameters) for the use in the project. Thus a class is useful for structuring the available alarms. The different alarm groups are defined by the user by inserting appropriate entries below the header 'System' in the configuration tree.
For the visualization of alarms the element 'Alarm table' [▶ 293] is available in the visualization. Using this table the user can watch and acknowledge alarms.
If a History, i.e. recording of Alarm-Events should be written to a log-file, such a file must be defined and for each alarm group the saving behaviour must be defined.

When you open the 'Alarm configuration' in the Ressources tab, the dialog 'Alarm configuration' opens with a bi-partite window, which concerning the mode of operation is similar to that of the PLC Configuration or Task configuration.
In the left part the configuration tree is displayed, in the right part the appropriate configuration dialog will be opened.

Open by a mouse-click on the plus sign at the entry 'Alarm configuration' the currently available configuration tree. If you are going to create a new configuration, this tree only will show the entries 'Alarm classes' and 'System'.



6.2.1 Alarm system, Terms

The usage of an alarm system in TwinCAT PLC Control obeys the following universal descriptions and definitions concerning alarms:

- **Alarm:** Generally an alarm is regarded as a special condition (expression value).
- **Priority:** The priority, also named "severity", of an alarm describes how important (severe) the alarm condition is. The highest priority is "0", the lowest valid priority value is "255".
- **Alarm state:** An expression/variable configured for the alarm control can have the following states: NORM (no alarm), INTO (alarm just has come), ACK (alarm has come and has been acknowledged by the user), OUTOF (alarm state has been terminated, alarm "has gone", but not yet acknowledged!)
- **Sub-State:** An alarm condition can have limits (Lo, Hi) and "extreme" limits (LoLo, HiHi). Example: The value of an expression ascends and first will transit the HI-limit, thus causing the coming of an HI-alarm. If the value continues ascending and exceeds also the HIHI-limit before the alarm gets acknowledged by the user, then the HI-alarm will get acknowledged automatically and just the HIHI-alarm remains in the alarm list (which is an internal list used for alarm administration). The HI-state in this case is named sub-state.
- **Acknowledgement of alarms:** The main purpose of alarms is to inform the user on alarm situations. In doing so it often is necessary to make sure that the user has noticed this information (see possible actions assigned to an alarm in the alarm class configuration). The user must acknowledge the alarm in order to get the alarm removed from the alarm list.
- **Alarm Event:** An alarm event must not be mixed up with an alarm condition. While an alarm condition can be valid for a longer period of time, an alarm event just describes the momentary occurrence of an change, e.g. a change from the normal state to the alarm state. In the TwinCAT PLC Control alarm configuration for the three types of events and the corresponding alarm states the same names are used (INTO, ACK, OUTOF).

In TwinCAT PLC Control the following features are supported:

- Deactivation of the alarm generation for single alarms as well as for alarm groups
- Selection of the alarms which should be displayed by defining alarm groups and priorities
- Saving of all alarm events in an alarm table

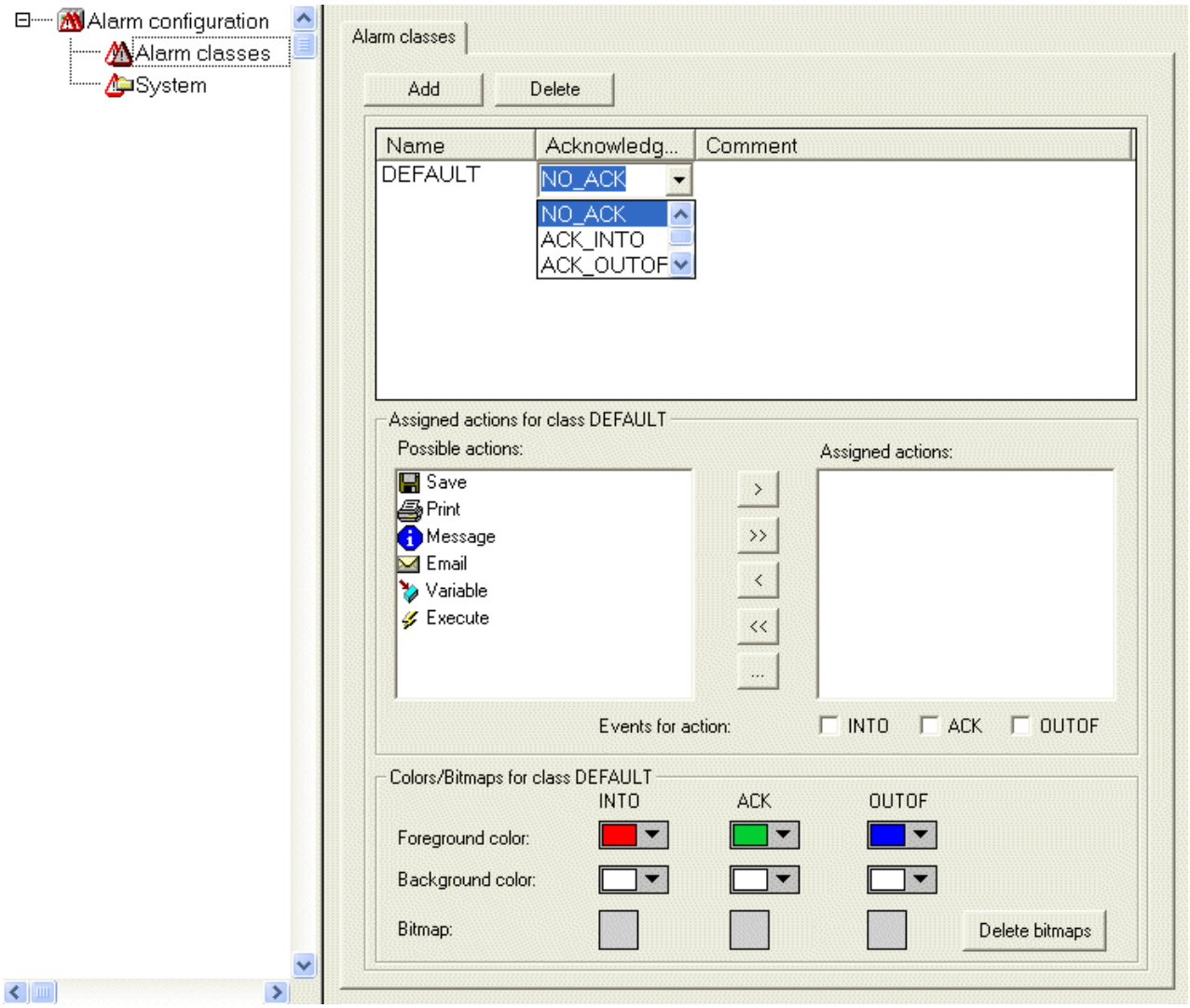
- Visualization element 'Alarm table' in the TwinCAT PLC Control Visualization

6.2.2 Alarm classes

Alarm classes are used for the general description of certain alarm criteria, such as how to handle acknowledgements (confirmation of an alarm by the user), which actions should automatically run as soon as a particular alarm state has been detected and which colors and bitmaps are to be used for a visualization of an Alarm table. Alarm classes are defined globally in the Alarm configuration and are then available as a base configuration when configuring alarm groups.

Configuration of alarm classes:

Select entry 'Alarm classes' in the alarm configuration tree. The configuration dialog 'Alarm classes' gets opened:



Configuration dialog 'Alarm classes'

Press button **Add** in order to create a new alarm class. Thereupon in the upper window a line will be inserted, primarily only with an entry "NOACK" (no acknowledgement) in the 'Acknowledgement' column. Define a name for the alarm class in the corresponding field in the **Name** column (open an edit frame by a mouse-click on the field) and if necessary modify the acknowledgement type in column **Acknowledgement**.

The following acknowledgements are available:

NO_ACK: No acknowledgement of the alarm by the user is required

ACK_INTO: A "come" alarm condition (status "INTO", alarm occurs) must be confirmed by the user.

ACK_OUTOF: A "gone alarm" (status "OUTOF", alarm terminated) must be confirmed by the user.

ACK_ALL: Gone and come alarm conditions must be confirmed by the user.

Additionally you can enter a **Comment**.

Entries for further alarm classes each will be added at the end of the list.
Use button **Delete** to remove the currently selected entry from the list.

Assigned actions for class <class name>:

Each alarm class defined in the upper window can get assigned a list of actions, which should be performed as soon as a Alarm Event occurs.
In the list of **Possible actions** select one and press button ">" to get it into the field **Assigned actions**. This field will finally contain the selection of actions assigned to the alarm class. Via button "**>>**" you can add all actions at a single blow. Via "**<**" resp.. "**<<**" you can remove one or all actions from the done existing selection.
If an action is marked in the 'Assigned actions' list, via "**...**" a corresponding dialog can be opened to define the desired e-mail settings, the printer settings, the process variable resp. the executable program and, if applicable, a message text.

The following action types (Possible actions) are supported (for a definition of a message text see below):

Action	Description	Settings to be done in the corresponding dialog:
Save:	The alarm event will be saved internally, in order to be given out e.g.in a log-file. **Please regard:** In this case the log-file must be defined in the configuration of the alarm group !	The settings are done in the Alarm group definition in the Alarm saving dialog
Print:	A message text is sent to a printer.	**Printer:** Select one of the printers defined on the local system; **Outputtext:** Message text (see below) which should be printed out
Message:	In the current visualization of the alarm a message window will be opened showing the defined text.	**Message:** Message text to be displayed in the message window
E-Mail:	An e-mail containing the defined message will be sent.	**From:** e-mail address of sender; **To:** e-mail address of recipient; **Subject:** any subject; **Message:** Message text (see below); **Server:** Name of the e-mail server
Variable:	A variable of the current program will get the alarm status resp. a message text string.	**Variable:** Variable name: You can select project variables via the input assistant (<F2>): A boolean variable will indicate the alarm states NORM =0 and INTO=1, an integer variable will indicate the alarm states NORM =0, INTO =1, ACK =2, OUTOF =4; a string variable will get the message text defined in field; **Message** (see below)
Execute:	An executable file will be started as soon as the alarm event occurs.	**Executable file:** name of the file to be executed (e.g. notepad.exe, you can use the "..." button to get the standard dialog for selecting a file; **Parameter:** appropriate parameter(s) which should be attached to the call of the exe-file

Definition of the message text:

For action types 'Message', 'Print', 'Email' or 'Variable' you can define a message text which should be output in case of an Alarm Event.
Line breaks at the text definitions in 'Message', 'Email' or 'Variable' can be inserted by <Ctrl>+<Enter>.
The following **placeholders** can be used when defining the alarm message:

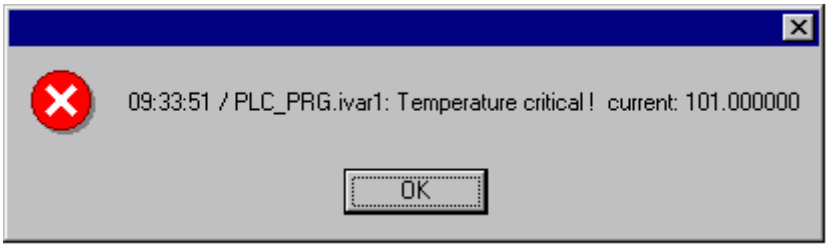
MESSAGE	The message text which is defined for the particular alarm in the configuration of the alarm group will be used.
DATE	Date, when the alarm status was reached (INTO).
TIME	Time of alarm entry.
EXPRESSION	Expression (defined in alarm group) which has caused the alarm.
PRIORITY	Priority of the alarm (defined for alarm group.)
VALUE	Current value of the expression (see above).
TYPE	Alarm type (defined in alarm group)
CLASS	Alarm class (defined in alarm group)
TARGETVALUE	Target value for alarm types DEV+ and DEV- (defined in alarm group)
DEADBAND	Tolerance of the alarm (defined in alarm group)
ALLDEFAULT	Any information on the alarm will be output, like described for the line entries in a log file (History).

Example of defining an alarm message:

For a definition of a message box enter the following in the message window:



Further on when defining the alarm in the alarm group enter in column 'Message' the following: "Temperature critical !". The output of the final alarm message will be like follows:



The message text will also be affected in case of a change of the project language if it is included in a *.vis-file or a translation file *.tlt. BUT: In this case - like texts referring to a visualization it has to be set between two "#"-characters (e.g. in the example shown above : "#Temperature critical !#" and "TIME /EXPRESSION: MESSAGE #current#: VALUE", in order to get the text entered in the translation file as ALARMTEXT_ITEMs.)

A l**og file** for action 'Save' is to be defined in the configuration of the alarm group.

Alarm Events for actions:

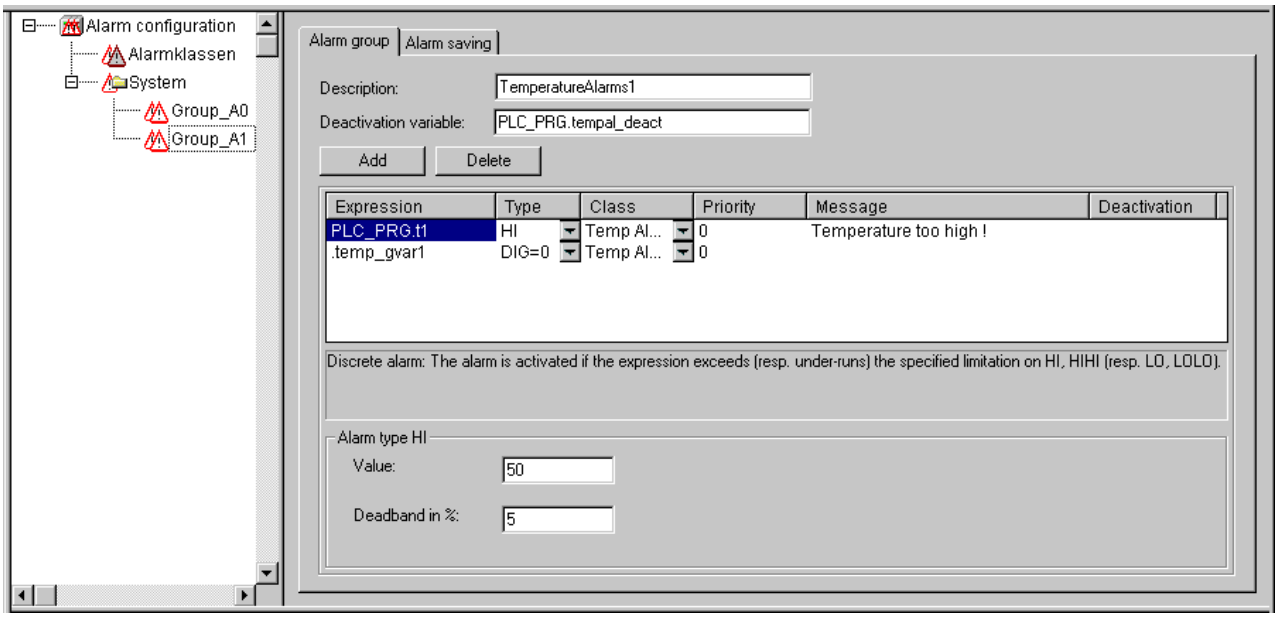
For each action you define, at which **alarm events** it should be started.
Activate the desired events:
INTO The alarm occurs. Status = INTO.
ACK Acknowledgement by the user has been done. Status = ACK.
OUTOF Alarm state terminated. Status = OUTOF.

Colors/Bitmaps for class <class name>

Each alarm class can get assigned own colors and bitmaps, which will be used for the differentiation of the alarms in the visualization element alarm table. Select a **Foreground color** and **Background color** for the possible events INTO, ACK and OUTOF (see above). The standard dialog for selecting a color will open as soon as you perform a mouse-click on the color symbol. For selecting a bitmap a mouse-click on the grey rectangle will open the standard dialog for selecting a file.

6.2.3 Alarm groups

Alarm groups are used for organizing the available alarms. Each alarm is definitely assigned to right one alarm group and is managed by this group. All alarms of a group can get assigned a common deactivation variable and common parameters regarding the alarm saving. Regard that even a single alarm must be configured within an alarm group.
A hierarchical structure of alarm groups can be defined via folder elements. When a alarm group is selected in the configuration tree, automatically the **dialog Alarm group** will be displayed:



Configuration dialog Alarm group

In the field **Description** you can enter a name for the alarm group.
As **Deactivation variable** a boolean project variable can be defined. At a rising edge on this variable the alarm creation for all alarms of the group will be deactivate, at a falling edge it will be re-activated.
Via button **Add** an alarm can be added to the group. A new line in the table window will be inserted and there the following parameters are to be set:

Expression: Enter here the project variable or an expression (e.g. "a + b") to which the alarm should refer. It is recommended to use the input assistant <F2> resp. the "Intellisense function" for an correct entry.

Type: The alarm types listed in the following can be used. For each type regard the appropriate comment resp. the definitions to be done in the area beyond the table !

- **DIG=0** Digital alarm, active as soon as the expression gets FALSE.

- **DIG=1** Digital alarm, active as soon as the expression gets TRUE.
- **LOLO** Analog alarm, active as soon as the value of the expression falls below the Value defined for Alarm type LOLO. You can define a tolerance (Deadband). As long as the expression value is within the dead band, no alarm will be activated, even if the LOLO-value has been falling below the limit.
- **LO** corresponding to LOLO
- **HI** Analog alarm, active as soon as the expression exceeds the Value defined for Alarm type HIHI. You can define a tolerance (Deadband). As long as the expression value is within the dead band, no alarm will be activated, even if the HI value has exceeded the limit.
- **HIHI** corresponding to HI
- **DEV-** Deviation from the target value; Alarm gets active as soon as the value of the expression falls below the value defined for Alarm type DEV- plus the percentage deviation. Percentage deviation = target value * (deviation in %) / 100.
- **DEV+** Deviation from the target value); Alarm gets active as soon as the value of the expression exceeds the value defined for Alarm type DEV+ plus the percentage deviation. Percentage deviation = target value * (deviation in %) / 100.
- **ROC** Rate of Change per time unit; Alarm gets active as soon as the expression deviates strongly from the previous value. The limit value for activating an alarm is defined by the number of value changes (Rate of changes) per second, minute or hour (units per).

Class: Choose the desired alarm class. The selection list will offer all classes which have been defined in the alarm class configuration before the last saving of the project.

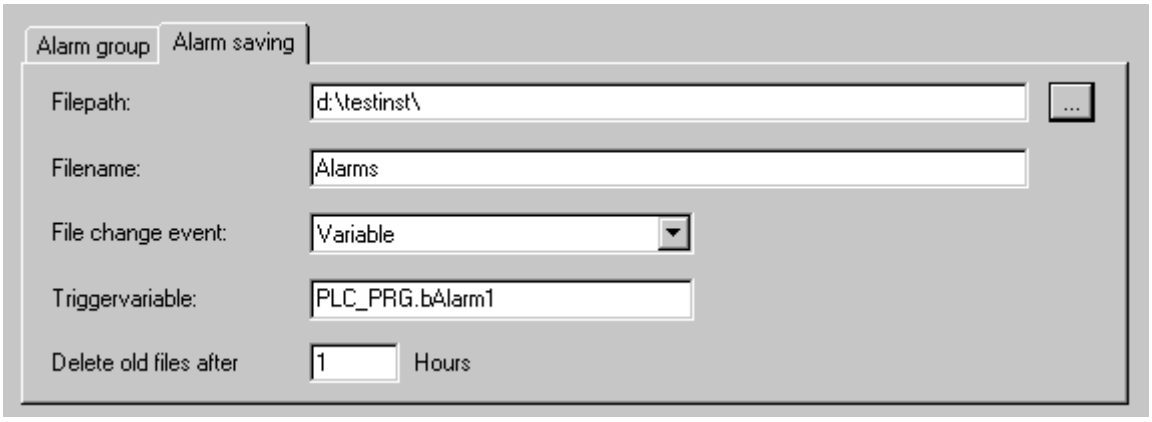
Priority: Here you can define a priority level 0-152. 0 is the highest priority. The priority will impinge on the sorting of the alarms within the alarm table.

Message: Define here the text for the message box, which will appear in case of an alarm. This box must be confirmed by the user with OK, but this OK will not automatically acknowledge the alarm ! For confirming (acknowledge) the alarm you must access the alarm table. This is possible via the visualization element alarm table or via the date of the alarm entry in the table. This date can be read from a log file which can be created optionally.

Deactivation: Here a project variable can be entered, which at a rising edge will deactivate any creation of the alarm. Regard however, that this setting will be overwritten by the entry which might be found in the field 'Deactivation variable' ! (see above).

6.2.4 Alarm saving

For each alarm group a file can be defined, in which the alarm events are stored, if (!) a 'Save' action has been assigned to the class in the alarm class configuration dialog.
Select the alarm group in the configuration tree and open the dialog tab 'Alarm saving':



Configuration dialog 'Alarm saving'

The following definitions are possible:

Filepath: Directories path of the file which is defined in Filename; via button "..." you get the standard dialog for selecting a directory.

Filename: Name of the file which should save the alarm events (e.g. "alarmlog"). Automatically a file will be created which gets the name defined here plus an attached digit and which has the extension ".alm". The digit indicates the version of the log-file. The first file gets a "0"; each further file, which will be created according to the defined File change event, will be numbered with 1, 2 etc. (Examples: "alarmlog0.alm", "alarmlog1.alm).

File change event: Define here the event which will cause the creation of a new file for alarm-saving. Possible entries: Never, after one Hour, after one Day, after one Week, after one Month, at a rising edge of the variable defined in field Triggervariable, when the number of records in the file as defined in Number of records gets exceeded.

Triggervariable resp. Number of records: see above, File change event.

Delete old files after .. Hours: Number of days since the day of creation, after which all alarm log-files except from the actual one should be deleted.

The log-file (History) contains the following entries:

(See the column types and exemplary entries for two alarms)

Date/Time in DWORD	Date	Time	Event	Expression	Alarm type	Limit	Tolerance	current value	class	Priority	Message
1046963332	6.3.03	16:08:52	INTO	PLC_PRG.b	LO	-30	5	-31	Alarm_high	0	Temperature !
1046963333	6.3.03	16:08:53	ACK	PLC_PRG.n	HIHI	35			Warnng	9	Rising Temp.!

EXAMPLE:

```
1046963332,6.3.03 16:08:52,INTO,PLC_PRG.ivar5,HIHI,,,,    9.00,a_class2,0,
1046963333,6.3.03 16:08:53,INTO,PLC_PRG.ivar4,ROC,2,,,    6.00,a_class2,2,
1046963333,6.3.03 16:08:53,INTO,PLC_PRG.ivar3,DEV-,,,,   -6.00,a_class2,5,
1046963334,6.3.03 16:08:54,INTO,PLC_PRG.ivar2,LOLO,-35,,3,  -47.00, warning, 10, warning: low temperature !
1046963334,6.3.03 16:08:54,INTO,PLC_PRG.ivar1,HI,20,,5,   47.00,a_class1,2,temperature to high ! Acknowledge !
```

6.2.5 'Extras' Menu: Settings

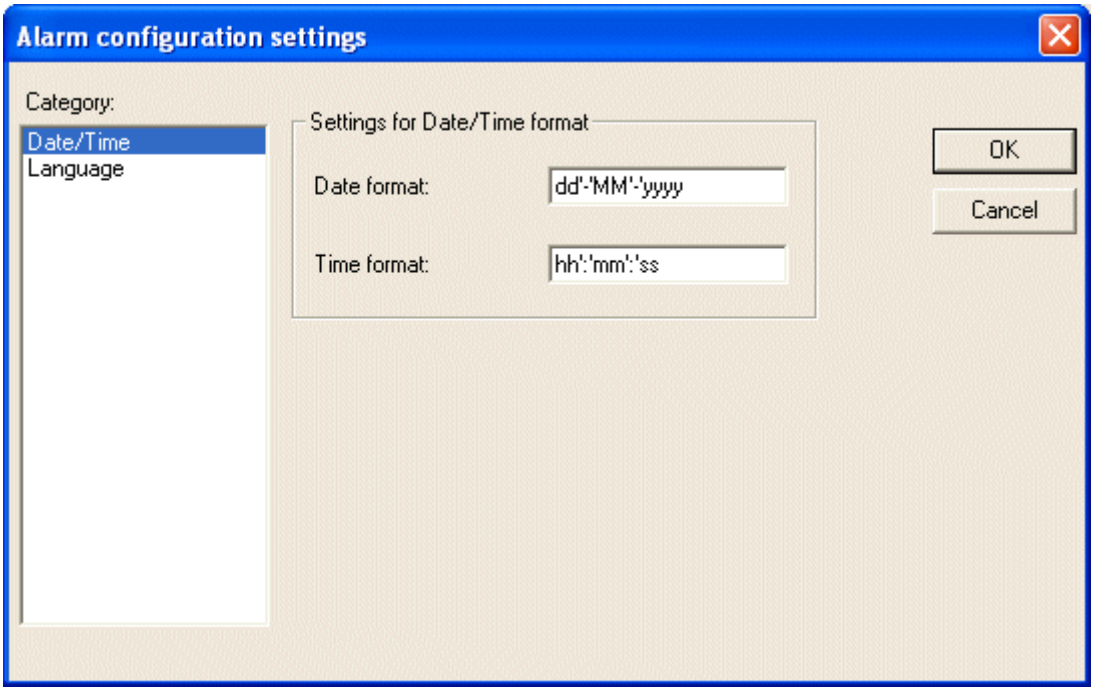
The dialog Alarm configuration settings opens on the command 'Extras' 'Settings' in the Alarm Configuration:

Category Date/Time:

Here you set the formatting for the representation of the alarms in the log-file. Define the format according to the following syntax. Dashes and colons are to be set in inverted commas:

for date: dd'-'MM'-'yyyy -> e.g. "12.Jan-1993"

for time: hh':'mm':'ss -> e.g. "11:10:34"

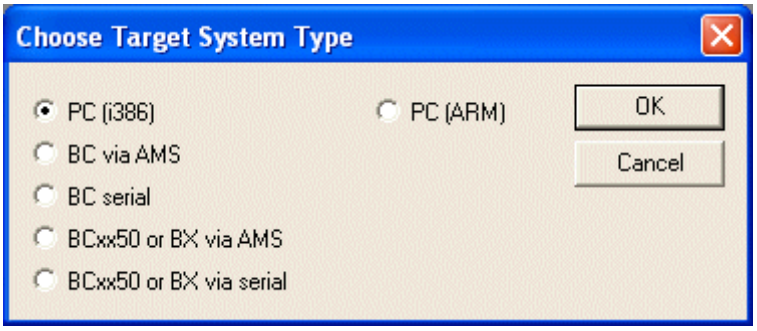


Language:

Choose here a language file which should be used when the language in TwinCAT PLC Control is changed. Regard that for this purpose the language file also must contain the translations for the text strings of the alarm configuration. In this context see the following descriptions:

- Visualization, Setting the language [▶ 309]
- Translate project into another language [▶ 66]

6.3 PLC Configuration



The PLC configuration is dependent on the respectively configured hardware. TwinCAT supports fourdifferent hardware platforms:

- PC (i386): the PLC program runs on the PC or CX10x0.
- PC (ARM): the PLC program runs on a CX9000.
- BCxxxx: the PLC program runs on a buscontroller.
- BCxx50 or BXxxxxx

If you want to run the PLC program on a BCxxxx/BCxx50 or BXxxxx Bus Terminal PLC you can choose whether you want load the program

- via AMS (dependent on the fieldbus)
- via serial interface.

6.4 Task Configuration

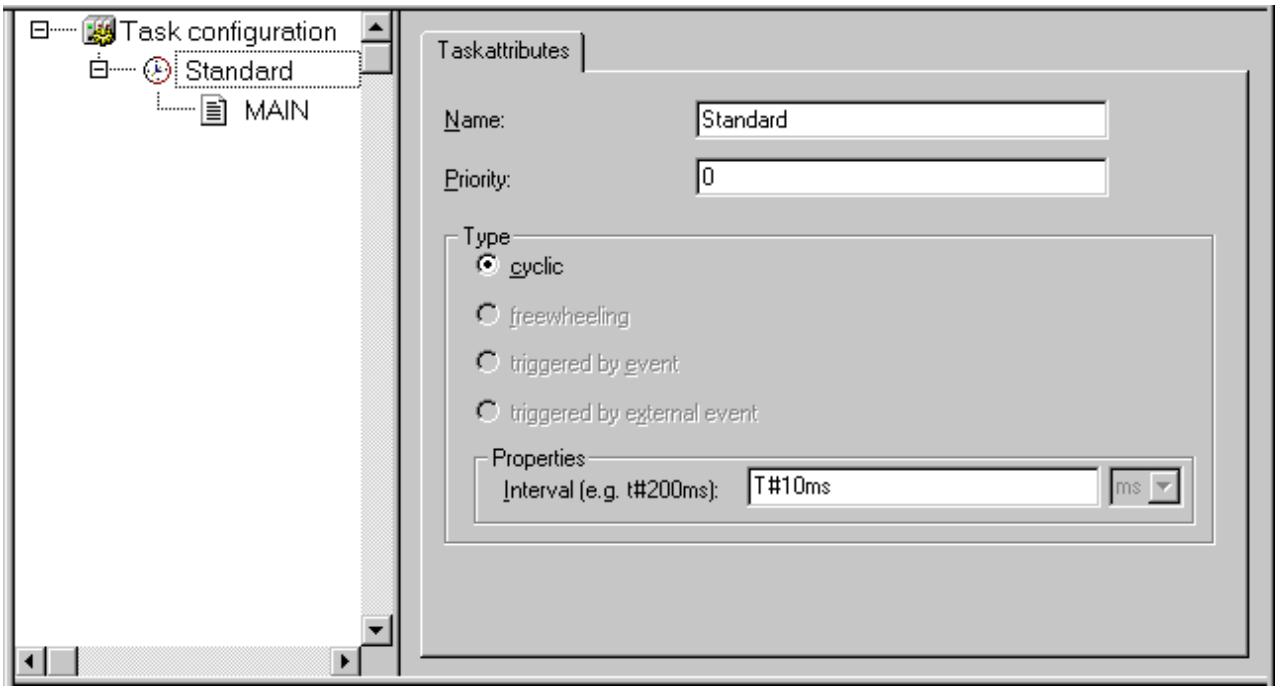
A **Task** is a time unit in the processing of an IEC program. It is defined by a name, a priority and by a type determining which condition will trigger the start of the task. This condition can be defined by a time only (cyclic).

For each task you can now specify a series of programs that will be started by the task.

The combination of priority and condition will determine in which choronological order the tasks will be executed.

In the Online Mode the task processing can be monitored in a diagram.

The Task Configuration is found as an object in the **Ressources** register card the Object Organizer. The Task editor is opened in a bipartited window.



In the left part of the window the tasks are represented in a configuration tree. At the topmost position you will always find the entry 'Taskconfiguration'. Below there are the entries for the particular tasks, represented by the task name. Below each task entry the assigned program calls are inserted.

In the right part of the window a dialog will be displayed which belongs to the currently marked entry in the configuration tree. Here you can configure the tasks and program calls.

NOTICE

Data loss

Please do not use the same string function in several tasks, because this may cause program faults by overwriting.

Working with the Task Configuration

The most important commands you find in the **context menu** (right mouse button).

At the heading of the task configuration are the words "Task Configuration." If a plus sign is located before the words, then the sequence list is closed. By doubleclicking on the list or pressing <Enter>, you can open the list. A minus sign now appears. By doubleclicking once more, you can close the list again.

- For every task, there is a list of program call-ups attached. Likewise, you can open and close this list the same way.
- With the **"Insert" "Insert Task"** command, you can insert a task.

- With the "**Insert" "Append Task"** command, you can insert a task at the end of the configuration tree.
- With the "**Insert" "Insert Program Call"**, a program call will be inserted.

Further on for each entry in the configuration tree an appropriate configuration dialog will appear in the right part of the window. There options can be activated/deactivated resp. inputs to editor fields can be made. Depending on which entry is selected in the configuration tree, there will be the dialog for defining the 'Taskattributes' or the dialog for defining a 'Program Call'. The settings made in the dialogs will be taken over to the configuration tree as soon as the focus is set to the tree again.

A task name or program name can also get edited in the configuration tree. For this perform a mouseclick on the name or select the entry and press the <Space> button to open an edit frame.

You can use the arrow keys to select the previous or next entry in the configuration tree.

'Insert' 'Insert Task' or 'Insert' 'Append Task'

With this command you can insert a new task into the task configuration. If a task is selected, then the **"Insert Task"** command will be at your disposal. The new task will be inserted after the selected one. If the words Task Configuration are selected, then the **"Append Task"** is available, and the new task will be appended to the end of the existing list. The dialog box will open for you to set the task attributes.

Insert the desired attributes:

- **Name**: a name for the task; with this name the task is represented in the configuration tree; the name can be edited there after a mouseclick on the entry or after pressing the <Space> key when the entry is selected.
- **Priority (0-3)**: (a number between 0 and 31; 0 is the highest priority, 31 is the lowest),
- **Type**:
  - **cyclic**: The task will be processed cyclic according to the time definition given in the field 'Interval'.

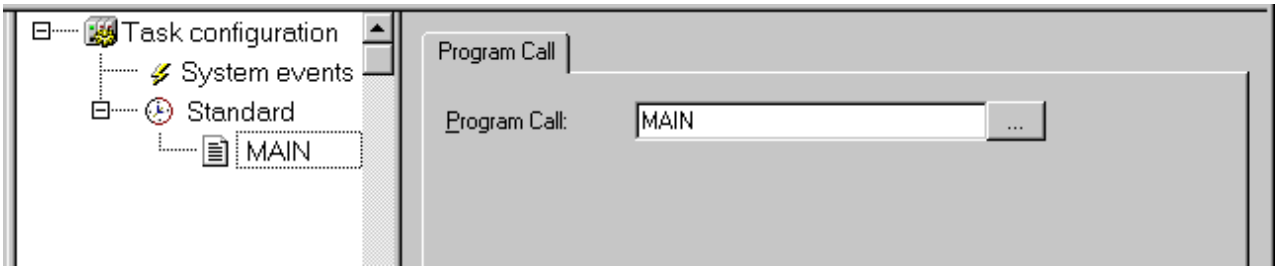
Freewheeling, triggered by event or triggered by external event: These task types are not supported!

Properties:

- **Interval** (for type 'cyclic'): the period of time, after which the task should be restarted. If you enter a number, then you can choose the desired unit in the selection box behind the edit field: milliseconds [ms] or microseconds [µs]. Inputs in [ms]-format will be shown in the TIME format (e.g. "t#200ms") as soon as the window gets repainted; but you also can directly enter the value in TIME format. Inputs in [ms] will always be displayed as a pure number (e.g. "300").

'Insert' 'Insert Program Call' or 'Insert' 'Append Program Call'

With these commands you will open the dialog box for entering a program call to a task in the task configuration. With **"Insert Program Call"**, the new program call is inserted in front of the cursor, and with **"Append Program Call"**, the program call is appended to the end of the existing list.



In the field, specify a valid program name for your project, or open the Input Assistant with the button... or with <F2> to select a valid program name. The program name can also get edited in the configuration tree. For this perform a mouseclick on the name or select the entry and press the <Space> button to open an edit frame. If the selected program requires input variables, then enter these in their usual form and of the declared type (for example, prg(invar:=17)).

'Extras' 'Set Debug Task'

With this command a debugging task can be set in Online mode in the task configuration. The text [DEBUG] will appear after the set task.
The debugging capabilities apply, then, only to this task. In other words, the program only stops at a breakpoint if the program is gone through by the set task. The setting of the Debug Task is stored in the project and will be set again automatically at log in / download.

Cancel the Debug Mode

To cancel the "Debug-Mode"

- choose "Task configuration"
- open the Context menu
- choose „Set Debug Task“

'Extras' 'Display Callstack'

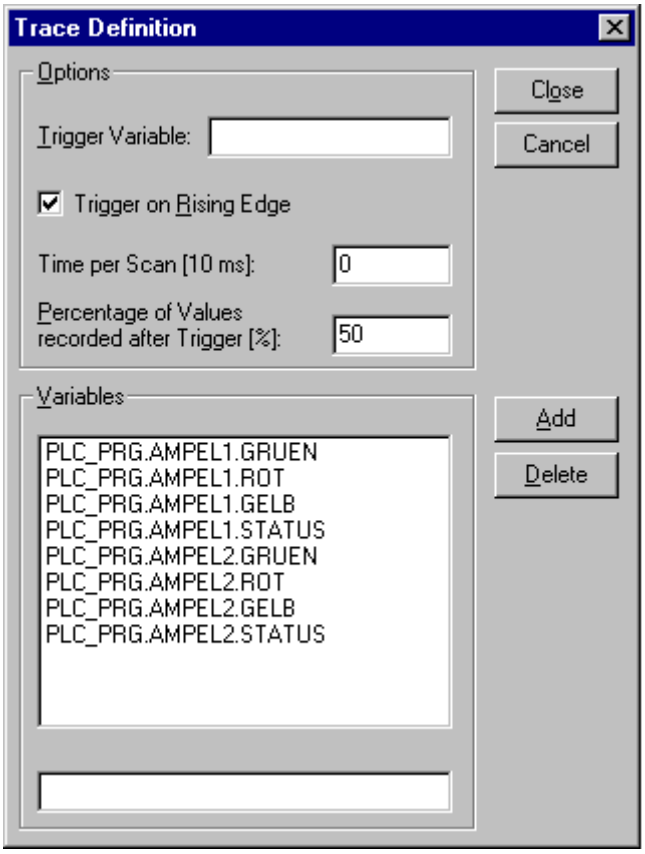
If the program is stopped at a breakpoint during debugging, then this command can be used to show the callstack of the corresponding POU. For this purpose the debug task must be selected in the task configuration tree. The window 'Callstack of task <task name>' will open. There you get the name of the POU and the breakpoint position (e.g. "prog_x (2)" for line 2 of POU prog_x) . Below the complete call stack is shown in backward order. If you press button 'Go To', the focus will jump to that position in the POU which is currently marked in the callstack.

6.5 Sampling Trace

Sample tracing means that the progression of values for variables is traced over a certain time frame. These values are written in a ring buffer (trace buffer). If the memory is full, then the "oldest" values from the start of the memory will be overwritten.
As a maximum, 20 variables can be traced at the same time. Since the size of the trace buffer in the PLC has a fixed value, in the event of very many or very wide variables (DWORD), fewer values can be traced.
In order to be able to perform a trace, open the object for a Sampling Trace in the Ressources register card in the Object Organizer . After this, you must enter the trace variables to be traced. (See **"Extras" "Trace Configuration"**.)
After you have sent the configuration with **"Define Trace"** to the PLC and have started the trace in the PLC (**"Start Trace"**), then the values of the variables will be traced. With **"Read Trace"**, the final traced values will be read out and displayed graphically as curves.

'Extras' 'Trace Configuration'

With this command you will be given the dialog box for entering the variables to be traced, as well as diverse trace parameters for the Sampling Trace.



Dialog Box for Trace Configuration

The list of the **Variables** to be traced is initially empty. In order to append a variable, then the variable must be entered in the field under the list. Following this, you can use the **Add** button or the <Enter> to append the variable to the list. You can also use the **Input Assistant**.
A variable is deleted from the list when you select the variable and then press the **Delete** button.

A Boolean or analogue variable can be entered into the field **Trigger Variable**. The input assistance can also be used here. The trigger variable describes the termination condition of the trace. In **Trigger Level** you enter the level of an analogue trigger variable at which the trigger event occurs. When **Trigger edge** positive is selected the trigger event occurs after an ascending edge of the Boolean trigger variable or when an analogue variable has passed through the trigger level from below to above. **Negative** causes triggering after a descending edge or when an analogue variable went from above to below. **Both** causes triggering for both descending and ascending edges or by a positive or negative pass, whereas none does not initiate a triggering event at all.
Trigger Position is used to set the percentage of the measured value which will be recorded before the trigger event occurs. If, for example, you enter 25 here then 25 % of the measured values are shown before the triggering event and 75% afterwards and then the trace is terminated. The field Sample Rate is

With the **Time per Scan** field, you can specify the time between two scans in milliseconds. The default setting "0" means one scanning procedure per cycle.
Select the mode for recalling the recorded values: With **Single** the **Number of the defined samples** are displayed one time. With **Continuous** the reading of the recording of the defined number of measured values is initiated anew each time. If, for example, you enter the number ‘35’ the first display contains the first measured values 1 to 35 and the recording of the next 35 measured values (36-70) will then be automatically read, etc.. **Manual** selection is used to read the trace recordings specifically with **’Extras‘ ‘Read trace'.**

The recall mode functions independently of whether a trigger variable is set or not. If no trigger variable is set the trace buffer will be filled with the defined number of measured values and the buffer contents will be read and displayed on recall.

The button **Save** is used to store the trace configuration which has been created in a file. The standard window ”File save as” is opened for this purpose.

Stored trace configurations can be retrieved using the button **Load**. The standard window ”File open” is opened for this purpose.

Please note that **Save** and **Load** in the configuration dialog only relates to the configuration, not to the values of a trace recording (in contrast to the menu commands ‘Extras’ ‘Save trace’ and ‘Extras’ ‘Load trace’).
If the field **Trigger Variable** is empty, the trace recording will run endlessly and can be stopped by **'Extras' 'Stop Trace'.**

'Extra' 'Start Trace'

With this command the trace configuration is transferred to the PLC and the trace sampling is started in the PLC.

'Extra' 'Read Trace'

With this command the present trace buffer is read from the PLC, and the values of the selected variables are displayed.

'Extra' 'Auto Read'

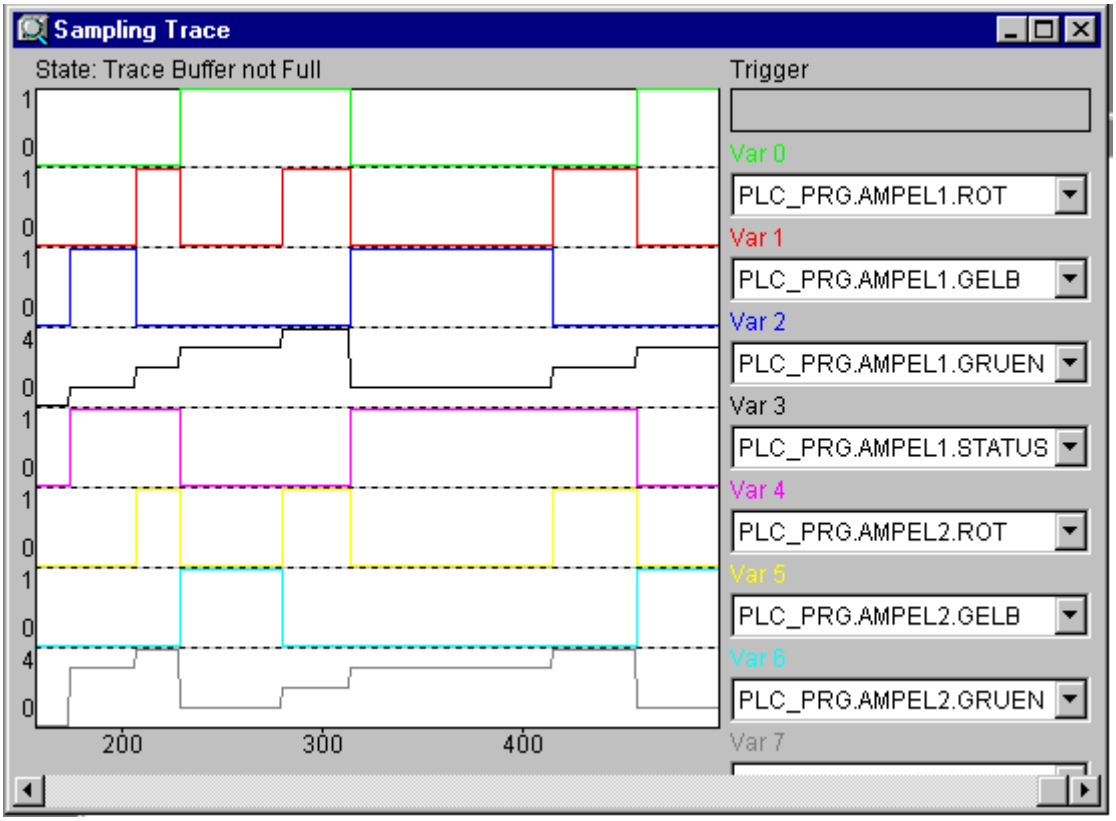
With this command the present trace buffer is read automatically from the PLC, and the values are continuously displayed. If the trace buffer is automatically read, then a check is located before the menu item.

'Extra' 'Stop Trace'

This command stops the Sampling Trace in the PLC. Before a new trace, the trace definition must be loaded, and the trace must be run again.

Selection of the Variables to be Displayed

The comboboxes to the right, next to the window for displaying curves trace variables defined in the trace configuration. If a variable is selected from the list, then after the trace buffer has been read the variable will be displayed in the corresponding color (Var 0 green, etc.). Variables can also be selected if curves are already displayed.
A maximum of up to eight variables can be observed simultaneously in the trace window.



Sampling Trace of Eight Different Variables without a Trigger

If a trace buffer is loaded, then the values of all variables to be displayed will be read out and displayed. If no scan frequency has been set, then the X axis will be inscribed with the continuous number of the traced value.
The status indicator of the trace window (first line) indicates whether the trace buffer is full and when the trace is completed. If a value for the scan frequency was specified, then the x axis will specify the time of the value. The time is assigned to the "oldest" traced value. In the example, e.g., the values for the last 25 seconds are indicated.
The Y axis is inscribed with integer values. The scaling is laid out in a way that allows the lowest and the highest value to fit in the viewing area. In the example, Var5 has taken on the lowest value of 6, and the highest value of 11: hence the setting of the scale at the left edge.
If the trigger requirement is met, then a vertical dotted line is displayed at the interface between the values before and after the appearance of the trigger requirement.
A memory that has been read will be preserved until you change the project or leave the system

'Extras' 'Cursor Mode'

The easiest way to set a cursor in the monitoring area is to click there with the left mouse button. A cursor appears and can be moved by the mouse. At the top of the monitoring window the current x-position of the cursor is displayed. In the fields next to 'Var 0', 'Var 1', ..., 'Var n' the value of the respective variable is shown.
Another way is the command 'Extras' 'Cursor mode'. With this command two vertical lines will appear in the Sampling Trace. First they are laying one on the other. One of the lines can be moved to the right or to the left by the arrow keys. By pressing <Ctrl>+<left> or <Ctrl>+<right> the speed of the movement can be increased.
If the mouse pointer is located in the graphics window and the left mouse button is pressed, then the cursor will likewise be displayed.

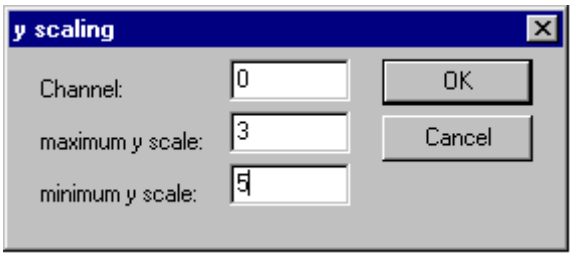
If additionally the <Shift> key is pressed, the second line can be moved, showing the difference to the first one.

'Extras' 'Multi Channel'

With this command you can alternate between single-channel and multi-channel display of the sampling trace. In the event of a multi-channel display, there is a check in front of the menu item. The multi-channel display has been preset. Here the display window is divided into as many as eight display curves. For each curve the maximum and the minimum value are displayed at the edge. In a single-channel display, all curves are displayed with the same scaling factor and are superimposed. This can be useful when displaying curve abnormalities.

'Extras' 'Y Scaling'

With this command you can change the preset Y scaling of a curve in the trace display. In the dialog box specify the number of the desired curve (**Channel**) and the new maximum (**maximum y scale**) and the new minimum value (**minimum y scale**) on the y axis. By doubleclicking on a curve you will also be given the dialog box. The channel and the former value are preset.



Dialog Box for Setting the Y Scale

'Extras' 'Stretch'

With this command you can stretch (zoom) the values of the sampling tracethat are shown. The beginning position is set with the horizontal picture adjustment bar. With repeated stretches that follow one-after-another, the trace section displayed in the window will increasingly shrink in size. This command is the counterpart to **"Extras" "Compress"**.

'Extras' 'Compress'

With this command the values shown for the sampling trace are compressed; i.e., after this command you can view the progression of the trace variables within a larger time frame. A multiple execution of the command is possible. This command is the counterpart to **"Extras" "Stretch".**

'Extras' 'Save Trace'

With this command you can save a sampling trace. The dialog box for saving a file is opened. The file name receives the extension "*.trc". The saved sampling trace can be loaded again with **"Extras" "Load Trace".**

'Extras' 'Load Trace'

With this command a saved sampling trace can be reloaded. The dialog box is opened for opening a file. Select the desired file with the "*.trc" extension. With **"Extras" "Save Trace"** you can save a sampling trace.

'Extras' 'Trace in ASCII-file'

With this command you can save a sampling trace in an ASCII-file. The dialog box is opened for saving a file. The file name receives the extension "*.txt". The values are deposited in the file according to the following scheme:
TwinCAT PLC Control Trace
D:\TWINCAT PLC CONTROL\PROJECTS\TRAFFICSIGNAL.PRO
Cycle PLC_PRG.COUNTER PLC_PRG.LIGHT1
0 2 1
1 2 1
2 2 1
.....

If no frequency scan was set in the trace configuration, then the cycle is located in the first column; that means one value per cycle has been re-corded at any given time. In the other respects, the entry here is for the point in time in ms at which the values of the variables have been saved since the sampling trace has been run.
In the subsequent columns, the corresponding values of the trace variables are saved. At any given time the values are separated from one another by a blank space.
The appertaining variable names are displayed next to one another in the third line, according to the sequence (PLC_PRG.COUNTER, PLC_PRG.LIGHT1).

6.6 Watch and Recipe Manager

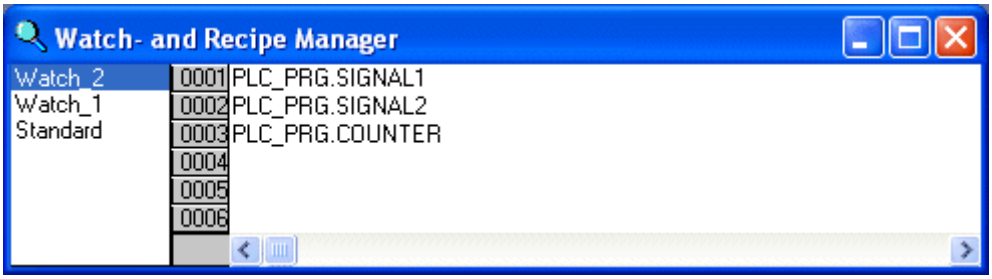
With the help of the Watch and Recipe Manager you can view the values of selected variables. The Watch and Recipe Manager also makes it possible to preset the variables with definite values and transfer them as a group to the PLC (**"Write Recipe"**). In the same way, current PLC values can be read into and stored in the Watch and Recipe Manager (**"Read Recipe"**). These functions are helpful, for example, for setting and entering of control parameters.
All watch lists created (**"Insert" "New Watch List"**) are indicated in the left column of the Watch and Recipe Manager. These lists can be selected with a mouse click or an arrow key. In the right area of the Watch and Recipe Manager the variables applicable at any given time are indicated.
In order to work with the Watch and Recipe Manager, open the object for the **Watch and Recipe Manager** in the **Ressources** register card in the Object Organizer.

In **Offline Mod**e, you can create several watch lists in the Watch and Recipe Manager using the **"Insert" "New Watch List"**.
For inputting the variables to be watched, you can call up a list of all variables with the Input Assistant, or you can enter the variables with the key-board, according to the following notation:
<POUName>.<Variable Name>.

With global variables, the POU Name is left out. You begin with a point. The variable name can, once again, contain multiple levels. Addresses can be entered directly.

Example of a multiple-level variable:
PLC_PRG.Instance1.Instance2.Structure.Componentname

Example of a global variable:
.global1.component1



Watch and Recipe Manager in the Offline Mode

The variables in the watch list can be preset with constant values. That means that in Online mode you can use the **"Extras" "Write Recipe"** command to write these values into the variables. To do you must use := to assign the constant value of the variable:

Example:
PLC_PRG.TIMER:=50
In the example, the PLC_PRG.COUNTER variable is preset with the value 6

'Insert' 'New Watch List'

With this command a new watch list can be inserted into the Watch and Recipe Manager. Enter the desired name for the watch list in the dialog box that appears.

'Extras' 'Rename Watch List'

With this command you can change the name of a watch list in the Watch and Recipe Manager. In the dialog box that appears, enter the new name of the watch list.

'Extras' 'Save Watch List'

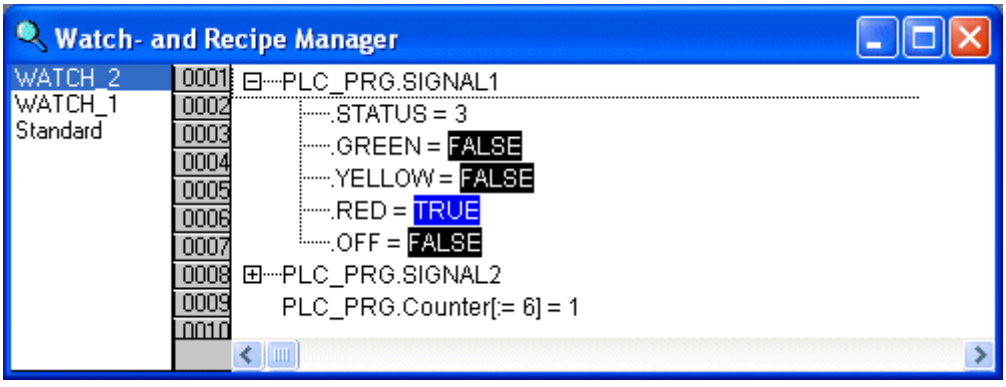
With this command you can save a watch list. The dialog box for saving a file is opened. The file name is preset with the name of the watch list and is given the extension "*.wtc". The saved watch list can be loaded again with "Extras" "Load Watch List".

'Extras' 'Load Watch List'

With this command you can reload a saved watch list. The dialog box is opened for opening a file. Select the desired file with the "*.wtc" extension. In the dialog box that appears, you can give the watch list a new name. The file name is preset without an extension. With "Extras" "Save Watch List", you can save a watch list.

Watch and Recipe Manager in the Online Mode

In Online mode, the values of the entered variables are indicated. Structured values (arrays, structures, or instances of function blocks) are marked by a plus sign in front of the identifier. By clicking the plus sign with the mouse or by pressing <Enter>, the variable is opened up or closed. In order to input new variables, you can turn off the display by using the **"Extra" "Active Monitoring"** command. After the variables have been entered, you can use the same command again to activate the display of the values.



Watch and Recipe Manager in the Online Mode
In the Offline Mode you can preset variables with constant values (through inputting := <value> after the variable). In the Online Mode, these values can now be written into the variables, using the "Extras" "Write Recipe" command. With the "Extras" "Read Recipe" command you can replace the presetting of the variable with the present value of the variable.



Only those values the watch list are loaded which was selected in the Watch and Recipe Manager!

'Extra' 'Monitoring Active'

With this command at the Watch and Recipe Manager in the Online mode, the display is turned on or off. If the display is active, a check will appear in front of the menu item. In order to enter new variables or to preset a value (see Offline Mode), the display must be turned off through the command. After the variables have been entered, you can use the same command again to activate the display of the values.

'Extras' 'Write Recipe'

With this command in the Online Mode of the Watch and Recipe Manager, you can write the preset values (see Offline Mode) into the variables.

'Extras' 'Read Recipe'

With the command, in the Online Mode of the Watch and Recipe Manager, you can replace the presetting of the variables (see Offline Mode) with the present value of the variables.

Example:

```
PLC_PRG.Counter [:= <present value>] =
<present value>
```

In the Watch and Recipe Manager, you can also "Force values" and "Write values". If you click on the respective variable value, then a dialog box opens, in which you can enter the new value of the variable. Changed variables appear in red in the Watch and Recipe Manager.

7 Reference programming

7.1 Data Types

The user can use standard data types and self-defined data types when programming. Each identifier is assigned a data type that determines how much memory is reserved and which values correspond to the memory contents.

NOTICE

Loss of data

The different data types cover different number ranges. It may happen that information is lost during type conversion from larger to smaller types.

Requirements

Standard data type	User-defined data type
BOOL [▶ 182]	ARRAY [▶ 186] (fields, arrays)
BYTE [▶ 183]	POINTER [▶ 187]
WORD [▶ 183]	ENUM [▶ 187] (enumeration type)
DWORD [▶ 183]	STRUCT [▶ 188] (structures)
SINT [▶ 184]	ALIAS [▶ 189] (references)
USINT [▶ 184]	Subrange types [▶ 189]
INT [▶ 184]	
UINT [▶ 184]	
DINT [▶ 184]	
UDINT [▶ 184]	
LINT (64-bit integer, currently not supported by TwinCAT)	
ULINT (Unsigned 64-bit integer, currently not supported by TwinCAT)	
REAL [▶ 184]	
LREAL [▶ 185]	
STRING [▶ 185]	
TIME [▶ 185]	
TIME_OF_DAY [▶ 185] (TOD)	
DATE [▶ 185]	
DATE_AND_TIME [▶ 186] (DT)	

7.1.1 Standard Data Types

7.1.1.1 BOOL

BOOL type variables may be given the values TRUE and FALSE.

Type	Memory use
BOOL	8 Bit

A BOOL type variable is true, if the least significant bit in the memory is set (e.g. 2#00000001). If no bit is set in the memory, the variable is FALSE (2#00000000). All other values can´t be interpeted accurately and be displayed (***INVALID: 16#xy *** in the Online View). Such problems may appear, if for example overlapped memory ranges are used in the PLC program.

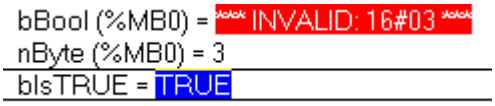
Example:

The boolean variable is in the same memory range as the byte variable.

```
PROGRAM MAIN
VAR
    bBool AT%MB0    : BOOL;
    nByte AT%MB0    : BYTE := 3;
    bIsTRUE         : BOOL;
END_VAR
```

```
IF bBool THEN
    bIsTRUE := TRUE;
ELSE
    bIsTRUE := FALSE;
END_IF
```

Online display after program start



7.1.1.2 BYTE

Integer data type.

Type	Lower bound	Upper bound	Memory use
BYTE	0	255	8 Bit

7.1.1.3 WORD

Integer data type.

Type	Lower bound	Upper bound	Memory use
WORD	0	65535	16 Bit

7.1.1.4 DWORD

Integer data type.

Type	Lower bound	Upper bound	Memory use
DWORD	0	4294967295	32 Bit

7.1.1.5 SINT

(Short) signed integer data type.

Type	Lower bound	Upper bound	Memory use
SINT	-128	127	8 Bit

7.1.1.6 USINT

Unsigned (short) integer data type.

Type	Lower bound	Upper bound	Memory use
USINT	0	255	8 Bit

7.1.1.7 INT

Signed integer data type.

Type	Lower bound	Upper bound	Memory use
INT	-32768	32767	16 Bit

7.1.1.8 UINT

Unsigned integer data type.

Type	Lower bound	Upper bound	Memory use
UINT	0	65535	16 Bit

7.1.1.9 DINT

Signed integer data type.

Type	Lower bound	Upper bound	Memory use
DINT	-2147483648	2147483647	32 Bit

7.1.1.10 UDINT

Unsigned integer data type.

Type	Lower bound	Upper bound	Memory use
UDINT	0	4294967295	32 Bit

7.1.1.11 REAL

32 Bit floating point data type. It is required to represent rational numbers.

Type	Lower bound	Upper bound	Memory use
REAL	~ -3.402823 x 10^{38}	~ 3.402823 x 10^{38}	32 Bit

7.1.1.12 LREAL

64 Bit floating point data type. It is required to represent rational numbers.

Type	Lower bound	Upper bound	Memory use
LREAL	~ -1.79769313486231E308	~ 1.79769313486232E308	64 Bit

7.1.1.13 STRING

A STRING type variable can contain any string of characters. The size entry in the declaration determines how much memory space should be reserved for the variable. It refers to the number of characters in the string and can be placed in parentheses or square brackets.

Example of a string declaration with 35 characters:

```
str:STRING(35):='This is a String';
```

Type	Memory use
STRING	• If no size specification is given, the default size of 80 characters will be used: **Memory use [Bytes] = 80 + 1 Byte for string terminated Null character;** • If string size specification is given: **Memory use [Bytes] = String Size + 1 Byte for string terminated Null character);**

7.1.1.14 TIME

Duration time. The least siginificant digit is one millisecond. The data type is handled internally like DWORD.

Type	Lower bound	Upper bound	Memory use
TIME	T#0ms	T#71582m47s295ms	32 Bit

7.1.1.15 TOD

Time of day. The least siginificant digit is one millisecond. The data type is handled internally like DWORD.

Type	Lower bound	Upper bound	Memory use
TIME_OF_DAY TOD	TOD#00:00	TOD#1193:02:47.295	32 Bit

7.1.1.16 DATE

Date. The least significant digit is one second. The data type is handled internally like DWORD.

Type	Lower bound	Upper bound	Memory use
DATE	D#1970-01-01	D#2106-02-06	32 Bit

7.1.1.17 DT

Date and time. The least siginificant digit is one second. The data type is handled internally like DWORD.

Type	Lower bound	Upper bound	Memory use
DATE_AND_TIME DT	DT#1970-01-01-00:00	DT#2106-02-06-06:28:15	32 Bit

7.1.2 User Data Types

7.1.2.1 Arrays

One-, two-, and three-dimensional fields (arrays) are supported as elementary data types. Arrays can be defined both in the declaration part of a POU and in the global variable lists.

Syntax:

```
<Field_Name>:ARRAY[<LowLim1>..<UpLim1>, <LowLim2>..<UpLim2>]OF <elem. Type>
```

LowLim1, *LowLim2* identify the lower limit of the field range; *UpLim1* and *UpLim2* identify the upper limit. The range values must be integers.

Example:

```
Card_game: ARRAY [1..13, 1..4] OF INT;
```

Initializing of Arrays

You can initialize either all of the elements in an array or none of them.

Example for initializing arrays:

arr1 : ARRAY [1..5] OF INT := 1,2,3,4,5;

arr2 : ARRAY [1..2,3..4] OF INT := 1,3(7); (* kurz für 1,7,7,7*)

arr3 : ARRAY [1..2,2..3,3..4] OF INT := 2(0),4(4),2,3; (* kurz für 0,0,4,4,4,4,2,3 *)

Example for the initialization of an array of a structure:

```
TYPE STRUCT1
STRUCT
p1:int;
p2:int;
p3:dword;
END_STRUCT
```

```
arr1 : ARRAY[1..3] OF STRUCT1 :=(p1:=1,p2:=10,p3:=4723),(p1:=2,p2=0,p3:=299),(p1:=14,p2:=5,p3:=112);
```

Example of the partial initialization of an Array:

arr1 : ARRAY [1..10] OF INT := 1,2;

Elements to which no value is pre-assigned are initialized with the default initial value of the basic type. In the example above, the elements *arr1*[3] to *arr1*[10] are therefore initialized with 0.

Array components are accessed in a two-dimensional array using the following syntax:

```
<Field_Name>[Index1,Index2]
```

Example:

```
Card_game[9,2]
```

If you define a function in your project with the name CheckBounds [▶ 218], you can automatically check for out-of-range errors in arrays ! The name of the function is fixed and can only have this designation.

7.1.2.2 Pointer

Variable or function block addresses are saved in pointers while a program is running. Pointer declarations have the following syntax:

```
<Identifier>: POINTER TO <Datatype/Functionblock>;
```

A pointer can point to any data type or function block even to user-defined types. The function of the Address Operator ADR [▶ 209] is to assign the address of a variable or function block to the pointer.

A pointer can be dereferenced by adding the content operator "^" after the pointer identifier. With the help of the SIZEOF [▶ 209] Operator, e.g. a pointer increment can be done.



A pointer is counted byte-wise! You can get it counted up like it is usual in the C-Compiler by using the instruction p=p+SIZEOF(p^);.

NOTICE

Adress transfer

After an Online Change there might be changes concerning the data on certain addresses. Please regard this in case of using pointers on addresses.

Example:

```
pt:POINTER TO INT;
var_int1:INT := 5;
var_int2:INT;
pt := ADR(var_int1);
var_int2:= pt^; (* var_int2 is now 5 *)
```

Example 2 (Pointer increment):

```
ptByCurrDataOffs : POINTER TO BYTE;
udiAddress       : UDINT;
(*--- pointer increment ---*)
udiAddress := ptByCurrDataOffs;
udiAddress := udiAddress + SIZEOF(ptByCurrDataOffs^);
ptByCurrDataOffs := udiAddress;
(* -- end of pointer increment ---*)
```

7.1.2.3 Enumeration (ENUM)

Enumeration is a user-defined data type that is made up of a number of string constants. These constants are referred to as enumeration values. Enumeration values are recognized in all areas of the project even if they were locally declared within aPOU. It is best to create your enumerations as objects in the Object Organizer under the register card Data types. They begin with the keyword TYPE and end with END_TYPE.

Syntax:

```
TYPE <Identifier>:(<Enum_0>
,<Enum_1>, ...,<Enum_n>);END_TYPE
```

The <Identifier> can take on one of the enumeration values and will be initialized with the first one. These values are compatible with whole numbers which means that you can perform operations with them just as you would with INT. You can assign a number x to the <Identifier>. If the enumeration values are not initialized, counting will begin with 0. When initializing, make certain the initial values are increasing. The validity of the number will be reviewed at the time it is run.

Example:

```
TRAFFIC_SIGNAL: (Red, Yellow, Green:=10);
(*The initial value for each of the colors is red 0, yellow 1, green 10 *)
TRAFFIC_SIGNAL:=0; (* The value of the traffic signal is red*)

FOR i:= Red TO Green DO
    i := i + 1;
END_FOR;
```

You may not use the same enumeration value more than once.

Example:

```
TRAFFIC_SIGNAL: (red, yellow, green);
COLOR: (blue, white, red);
```

Error: red may not be used for both TRAFFIC_SIGNAL and COLOR.

7.1.2.4 Structures (STRUCT)

Structures are created as objects in the Object Organizer under the register card Data types. They begin with the keyword TYPE and end with END_TYPE.The syntax for structure declarations is as follows:

```
TYPE <Structurename>:
STRUCT
    <Declaration of Variables 1>
    .
    .
    <Declaration of Variables n>
END_STRUCT
END_TYPE
```

<Structurename> is a type that is recognized throughout the project and can be used like a standard data type. Interlocking structures are allowed. The only restriction is that variables may not be placed at addresses (the AT declaration is not allowed!).

Example for a structure definition named Polygonline:

```
TYPE Polygonline:
STRUCT
    Start:ARRAY [1..2] OF INT;
    Point1:ARRAY [1..2] OF INT;
    Point2:ARRAY [1..2] OF INT;
    Point3:ARRAY [1..2] OF INT;
    Point4:ARRAY [1..2] OF INT;
    End:ARRAY [1..2] OF INT;
END_STRUCT
END_TYPE
```

You can gain access to structure components using the following syntax:

```
<Structure_Name>.<Componentname>
```

For example, if you have a structure named "Week" that contains a component named "Monday", you can get to it by doing the following: Week.Monday

NOTICE

Different structures

Due to different alignments, structures and arrays may have different configurations and sizes on different hardware platforms (e.g. CX1000 and CX90xx).
During data exchange the size and structure alignment must be identical!

Example for a structure definition with name ST_ALIGN_SAMPLE:

```
TYPE ST_ALIGN_SAMPLE:
    STRUCT
     _diField1   : DINT;
     _byField1   : BYTE;
     _iField     : INT;
     _byField2   : BYTE;
     _diField2   : DINT;
     _pField     : POINTER TO BYTE;
   END_STRUCT
END_TYPE
```

On CX90xx (RISC) platforms the member components of structure ST_ALIGN_SAMPLE have the following sizes and offsets:

```
_diField1 (DINT), Offset = 0 (16#0), Size = 4
_byField1 (BYTE), Offset = 4 (16#4), Size = 1
_iField (INT), Offset = 6 (16#6), Size = 2
_byField2 (BYTE), Offset = 8 (16#8), Size = 1
_diField2 (DINT), Offset = 12 (16#C), Size = 4
_pField (POINTER TO BYTE), Offset = 16 (16#10), Size = 4
```

Overall size through natural alignment with Pack(4) and so-called padding bytes: 20

On CX10xx platforms the member components of structure ST_ALIGN_SAMPLE have the following sizes and offsets:

```
_diField1 (DINT), Offset = 0 (16#0), Size = 4
_byField1 (BYTE), Offset = 4 (16#4), Size = 1
_iField (INT), Offset = 5 (16#5), Size = 2
_byField2 (BYTE), Offset = 7 (16#7), Size = 1
_diField2 (DINT), Offset = 8 (16#8), Size = 4
_pField (POINTER TO BYTE), Offset = 12 (16#C), Size = 4
```

Overall size: 16

Display of structure ST_ALIGN_SAMPLE for CX90xx platforms (RISC) **with representation of the padding bytes**:

```
TYPE ST_ALIGN_SAMPLE:
    STRUCT
     _diField1    : DINT;
     _byField1    : BYTE;
     _byPadding   : BYTE;
     _iField      : INT;
     _byField2    : BYTE;
     _a_byPadding : ARRAY[0..2] OF BYTE;
     _diField2    : DINT;
     _pField      : POINTER TO BYTE;
   END_STRUCT
END_TYPE
```

7.1.2.5 References (Alias types)

You can use the user-defined derived data type to create an alternative name for a variable, constant or function block. Create your references as objects in the Object Organizer under the register card Data types. They begin with the keyword TYPE and end with END_TYPE.

Syntax:

```
TYPE <Identifier>: <Assignment
term>;

END_TYPE
```

Example:

```
TYPE message:STRING[50];

END_TYPE;
```

7.1.2.6 Subrange types

A sub-range data type is a type whose range of values is only a subset of that of the basic type. The declaration can be carried out in the data types register, but a variable can also be directly declared with a subrange type:
Syntax for the declaration in the **'Data types'** register:

```
TYPE <Name> : <Inttype> (<ug>..<og>) END_TYPE;
```

Type	Description
<Name>	must be a valid IEC identifier
<Inttype>	is one of the data types SINT, USINT, INT, UINT, DINT, UDINT, BYTE, WORD, DWORD (LINT, ULINT, LWORD).
<ug>	Is a constant which must be compatible with the basic type and which sets the lower boundary of the range types. The lower boundary itself is included in this range.
<og>	Is a constant that must be compatible with the basic type, and sets the upper boundary of the range types. The upper boundary itself is included in this basic type.

Example:

```
TYPE
  SubInt : INT (-4095..4095);
END_TYPE
```

Direct declaration of a variable with a subrange type:

```
VAR
    i1 : INT (-4095..4095);
    i2: INT (5...10):=5;
    ui : UINT (0..10000);
END_VAR
```

If a constant is assigned to a subrange type (in the declaration or in the implementation) that does not apply to this range (e.g. 1:=5000), an error message is issued.

In order to check for observance of range boundaries at runtime, the functions CheckRangeSigned [▶ 221] or CheckRangeUnsigned [▶ 222] must be introduced.

7.2 Operators

TwinCAT PLC Control supports all IEC Operators. In contrast with the standard functions, these operators are recognized implicitly throughout the project. Operators are used like functions in POU implementation.

7.2.1 Overview IEC Operators

The table shown below shows the operators in ST and IL with the available modifiers in IL.

Take note that for the 'IL operator' column: Only the line in which the operator is used will be displayed. A prerequisite is that the (first) required operand have been successfully loaded in the preceding line (e.g. LD in). The 'Mod. IL' column shows the possible modifiers in IL:

C	The command is only executed if the result of the preceding expression is TRUE.
N	for JMPC, CALC, RETC: The command is only executed if the result of the preceding expression is FALSE.
N	otherwise: negation of the operand (not of the accumulator)
(	Operator enclosed in brackets: only after the closing bracket is reached will the operation preceding the brackets be carried out.

Please obtain a detailed description of usage from the Appendix.

Operator ST	Operator AWL	Mod. AWL	Signification
'			String delimiters (e.g. 'string1')
[..]			Size of Array range (e.g. ARRAY[0..3] OF INT)
:			Delimiter between Operand and Typ in a declaration (e.g. var1 : INT;)
;			Termination of instruction (e.g. a:=var1;)
^			Derefereced Pointer (e.g. pointer1^)
	LD var1	N	Load value of var1 in buffer
:=	ST var1	N	Store actual result to var1
	S boolvar		Set boolean operand boolvar exactly then to TRUE, when the actual result is TRUE
	R boolvar		Set boolean operand boolvar exactly then to FALSE, when the actual result is TRUE
	JMP marke	CN	Jump to label
<Programmname>	CAL prog1	CN	Call program prog1
<Instanzname>	CAL inst1	CN	Call function block instance inst1
<Fktname>(vx,vy,..)	<Fktname> vx,vy,..	CN	Call function fctname and transmit variables vx, vy
RETURN	RET	CN	Leave POU and go back to caller
	(		The value following the bracket is handled as operand, the operation before the bracket is not executed before the expression in the brackets.
	)		Now execute the operation which has been set back
AND	AND	N, (	Bitwise AND
OR	OR	N, (	Bitwise OR
XOR	XOR	N, (	Bitwise exclusive OR
NOT	NOT		Bitwise NOT
+	ADD	(	Addition
-	SUB	(	Subtraction
*	MUL	(	Multiplication
/	DIV	(	Division
>	GT	(	Greater than
>=	GE	(	Greater or equal
=	EQ	(	Equal
<	LT	(	Less than
<>	NE	(	Not equal

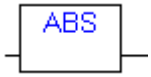
Operator ST	Operator AWL	Mod. AWL	Signification
<=	LE	(	Less or equal
in1 MOD in2	MOD		Modulo Division
INDEXOF(in)	INDEXOF		Internal index of POU in1; [INT]
SIZEOF(in)	SIZEOF		Number of bytes required for the given data type of in
SHL(in,K)	SHL		Bitwise left-shift of operator in by K
SHR(in,K)	SHR		Bitwise right-shift of operator in by K
ROL(in,K)	ROL		Bitwise rotation to the left of operator in by K
ROR(in,K)	ROR		Bitwise rotation to the right of operator in by K
SEL(G,in0,in1)	SEL		Binary selection between 2 operands in0 (G is FALSE) and in1 (G is TRUE)
MAX(in0,in1)	MAX		Returns the greater of 2 values
MIN(in0,in1)	MIN		Returns the lesser of 2 values in0 and in1
LIMIT(Min,in,Max)	LIMIT		Limits the value range (in is set back to MIN or MAX in case of exceeding the range)
MUX(K, in0,.. in_n)	MUX		Selects the Kth value out of a group of values (in0 to In_n)
ADR(in)	ADR		Address of the operand in [DWORD]
BOOL_TO_<type>(in)	BOOL_TO_<type>		Type conversion of the boolean operand
<type>_TO_BOOL(in)	<type>_TO_BOOL		Type conversion to BOOL
INT_TO_<type>(in)	INT_TO_<type>		Type conversion of an INT Operand to another elementary type
REAL_TO_<type>(in)	REAL_TO_<type>		Type conversion of an REAL operand to another elementary type
LREAL_TO_<type>(in)	LREAL_TO_<type>		Type conversion of a LREAL operand to another elementary type
TIME_TO_<type>(in)	TIME_TO_<type>		Type conversion of a TIME operand to another elementary type
TOD_TO_<type>(in)	TOD_TO_<type>		Type conversion of a TOD operand to another elementary type
DATE_TO_<type>(in)	DATE_TO_<type>		Type conversion of a DATE operand to another elementary type

Operator ST	Operator AWL	Mod. AWL	Signification
DT_TO_<type>(in)	DT_TO_<type>		Type conversion of a DT operand to another elementary type
STRING_TO_<type>(in)	STRING_TO_<type>		Type conversion of a STRING operand to another elementary type
TRUNC(in)	TRUNC		Conversion from REAL to INT
ABS(in)	ABS		Absolut value of operand in
SQRT(in)	SQRT		Square root of operand in
LN(in)	LN		Natural logarithm of operand in
LOG(in)	LOG		Logarithm of operand in, base 10
EXP(in)	EXP		Exponential function of operand in
SIN(in)	SIN		Sine of operand in
COS(IN)	COS		Cosine of operand in
TAN(in)	TAN		Tangent of operand in
ASIN(in)	ASIN		Arc sine of operand in
ACOS(in)	ACOS		Arc cosine of operand in
ATAN(in)	ATAN		Arc tangent of operand in
EXPT(in,expt)	EXPT expt		Exponentation of operand in with expt
LEN(in)	LEN		String length of operand in
LEFT(str, size)	LEFT		Left inital string of given size of string str standard.lib
RIGHT(str, size)	RIGHT		Right initial string of given size of string str standard.lib
MID(str, len, pos)	MID		Partial string of str of given length
CONCAT(str1, str2)	CONCAT		Concatenation of two subsequent strings standard.lib
INSERT(str1, str2, pos)	INSERT		Insert string str1 in String str2 at position pos standard.lib
DELETE(str1, len, pos)	DELETE		Delete partial string (length len), start at position pos of str1 standard.lib
REPLACE(str1, str2, len, pos)	REPLACE		Replace partial string of lenght len by str2, start at position pos of str1
FIND(str1, str2)	FIND		Search for partial string str2 in str1
SR	SR		Bistable FB is set dominant
RS	RS		Bistable FB is set back
SEMA	SEMA		FB: Software Semaphor (interruptable)

Operator ST	Operator AWL	Mod. AWL	Signification
R_TRIG	R_TRIG		FB: rising edge is detected
F_TRIG	F_TRIG		FB: falling edge is detected
CTU	CTU		FB: Counts up
CTD	CTD		FB: Counts down
CTUD	CTUD		FB: Counts up and down
TP	TP		FB: trigger
TON	TON		FB: on-delay timer
TOF	TOF		FB: off-delay timer

7.2.2 Numeric Operators

7.2.2.1 ABS



Returns the absolute value of a number. ABS(-2) equals 2.
The following type combinations for input and output variables are possible.

IN	OUT
INT	INT, REAL, WORD, DWORD, DINT
REAL	REAL
BYTE	INT, REAL, BYTE, WORD, DWORD, DINT
WORD	INT, REAL, WORD, DWORD, DINT
DWORD	REAL, DWORD, DINT
SINT	REAL
USINT	REAL
UINT	INT, REAL, WORD, DWORD, DINT, UDINT, UINT
DINT	REAL, DWORD, DINT
UDINT	REAL, DWORD, DINT, UDINT

7.2.2.2 ACOS



Returns the arc cosine (inverse function of cosine) of a number.
IN can be type BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT must be type REAL.

7.2.2.3 ASIN



Returns the arc sine (inverse function of sine) of a number.
IN can be type BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT must be type REAL.

7.2.2.4 ATAN



Returns the arc tangent (inverse function of tangent) of a number.
IN can be type BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT must be type REAL.

7.2.2.5 COS



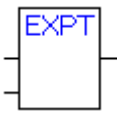
Returns the cosine of number.
IN can be type BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT must be type REAL.

7.2.2.6 EXP



Returns the exponential function.
IN can be type BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT must be type REAL.

7.2.2.7 EXPT



Exponentiation of a variable with another variable: OUT = $IN1^{IN2}$.
IN1 and IN2 can be type BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT must be type REAL.

Example in IL:

```
LD 7

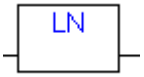
EXPT 2

ST var1 (* Result is 49 *)
```

Example in ST:

```
var1 := EXPT(7,2);
```

7.2.2.8 LN



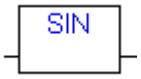
Returns the natural logarithm of a number. IN can be type BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT must be type REAL.

7.2.2.9 LOG



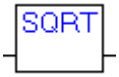
Returns the logarithm of a number in base 10. IN can be type BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT must be type REAL.

7.2.2.10 SIN



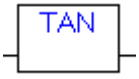
Returns the sine of a number. IN can be type BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT must be type REAL.

7.2.2.11 SQRT



Returns the square root of a number. IN can be type BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT must be type REAL.

7.2.2.12 TAN



Returns the tangent of a number. IN can be type BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT must be type REAL.

7.2.3 Arithmetic Operators

7.2.3.1 ADD

Addition of variables of the types: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL and LREAL.Two TIME variables can also be added together resulting in another time (e.g., t#45s + t#50s = t#1m35s).

Example in IL:

```
LD 7

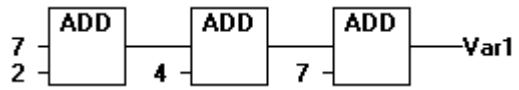
ADD 2,4,7

ST var1
```

Example in ST:

```
var1 := 7+2+4+7;
```

Example in FBD:



7.2.3.2 MUL

Multiplication of variables of the types: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL and LREAL.

Example in IL:

```
LD 7

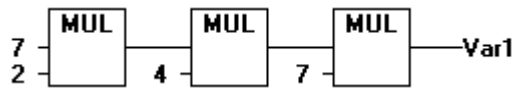
MUL 2,4,7

ST var1
```

Example in ST:

```
var1 := 7*2*4*7;
```

Example in FBD:



7.2.3.3 SUB

Subtraction of one variable from another of the types: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL and LREAL. A TIME variable may also be subtracted from another TIME variable resulting in third TIME type variable. Note that negative TIME values are undefined.

Example in IL:

```
LD 7

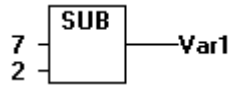
SUB 8

ST var1
```

Example in ST:

```
var1 := 7-2;
```

Example in FBD:



7.2.3.4 DIV

Division of one variable by another of the types: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL and LREAL.

Example in IL:

```
LD 8

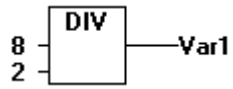
DIV 2

ST var1
```

Example in ST:

```
var1 := 8/2;
```

Example in FBD:



If you define functions in your project with the names CheckDivByte [▶ 219], CheckDivWord [▶ 220], CheckDivDWord [▶ 220]and CheckDivReal [▶ 219], you can use them to check the value of the divisor if you use the operator DIV, for example to avoid a division by 0. The functions must have the above listed names.

7.2.3.5 MOD

Modulo Division of one variable by another of the types: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT and UDINT. The result of this function will be the remainder of the division. This result will be a whole number.

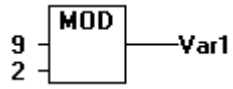
Example in IL:

```
LD 9
MOD 2
ST var1 (* Result is 1 *)
```

Example in ST:

```
var1 := 9 MOD 2;
```

Example in FBD:



Similar Functions

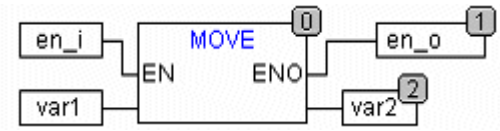
LMOD

7.2.3.6 MOVE

Assignment of a variable to another variable of an appropriate type. As MOVE is available as a box in the graphic editors LD, CFC, there the (unlocking) EN/EN0 functionality can also be applied on a variable assignment. In the FBD editor this is not possible however.

Example in CFC in conjunction with the EN/EN0 function:

Only if en_i is TRUE, var1 will be assigned to var2.



Example in IL:

```
LD ivar1
MOVE
ST ivar2 (* Ergebnis: var2 erhält Wert von var1 *)
```

(! you get the same result with:

```
LD ivar1
ST ivar2
```

)

Example in ST:

```
ivar2 := MOVE(ivar1);
```

(! you get the same result with:

```
ivar2 := ivar1;
```

)

7.2.4 Bitstring Operators

7.2.4.1 AND

Bitwise AND of bit operands. The operands should be of the type BOOL, BYTE, WORD or DWORD.

Example in IL:

```
var1 :BYTE;

LD 2#1001_0011

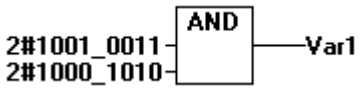
AND 2#1000_1010

ST var1 (* Result is 2#1000_0010 *)
```

Example in ST:

```
var1 := 2#1001_0011 AND 2#1000_1010
```

Example in FBD:



7.2.4.2 OR

Bitwise OR of bit operands. The operands should be of the type BOOL, BYTE, WORD or DWORD.

Example in IL:

```
var1 :BYTE;

LD 2#1001_0011

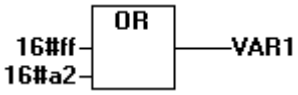
OR 2#1000_1010

ST var1 (* Result is 2#1001_1011 *)
```

Example in ST:

```
Var1 := 2#1001_0011 OR 2#1000_1010
```

Example in FBD:



7.2.4.3 XOR

Bitwise XOR of bit operands. The operands should be of the type BOOL, BYTE, WORD or DWORD.

Example in IL:

```
Var1 :BYTE;

LD 2#1001_0011

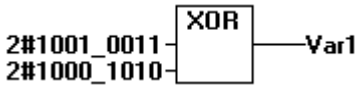
XOR 2#1000_1010

ST Var1 (* Result is 2#0001_1001 *)
```

Example in ST:

```
Var1 := 2#1001_0011 XOR 2#1000_1010
```

Example in FBD:



Please note, that the behaviour of the XOR POU in the extended form (more than 2 inputs) is not standard conformal implemented. The inputs are checked in piars, and then the respective results are compared against each other.

7.2.4.4 NOT

Bitwise NOT of a bit operand. The operand should be of the type BOOL, BYTE, WORD or DWORD.

Example in IL:

```
Var1 :BYTE;

LD 2#1001_0011

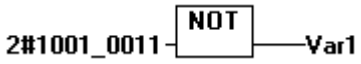
NOT

ST Var1 (* Result is 2#0110_1100 *)
```

Example in ST:

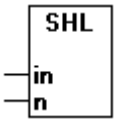
```
Var1 := NOT 2#1001_0011;
```

Example in FBD:



7.2.5 Bitshift Operators

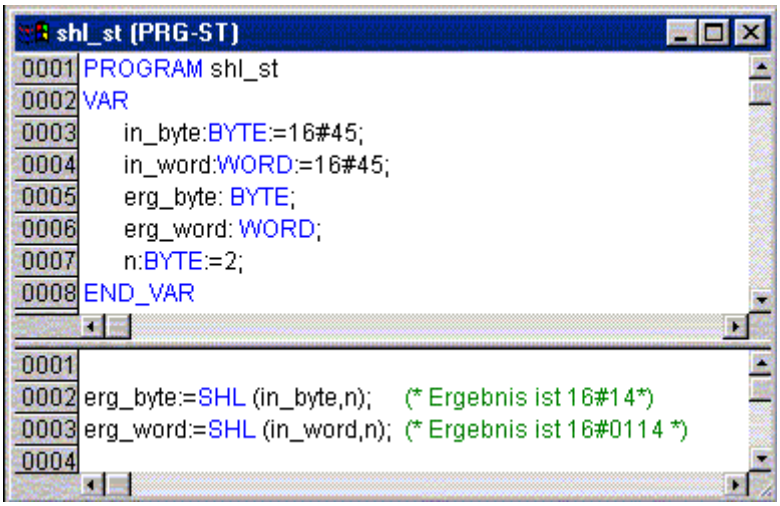
7.2.5.1 SHL



Bitwise left-shift of an operand : A:= SHL (IN, N)A, IN and N should be of the type BYTE, WORD, or DWORD. IN will be shifted to the left by N bits and filled with zeros on the right.

Please note, that the amount of bits, which is regarded for the arithmetic operation, is pretended by the data type of the input variable! If the input variable is a constant the smallest possible data type is regarded. The data type of the output variable has no effect at all on the arithmetic operation.

See in the following example in hexadecimal notation that you get different results for erg_byte and erg_word depending on the data type of the input variable (BYTE or WORD), although the values of the input variables in_byte and in_word are the same.
Example in ST:



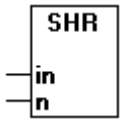
Example in IL:

```
LD 1

SHL 1

ST Var1 (* Result is 2 *)
```

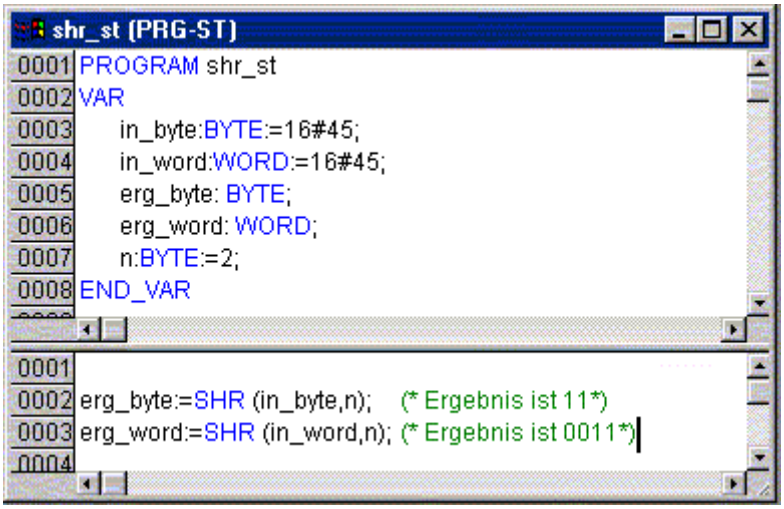
7.2.5.2 SHR



Bitwise right-shift of an operand: A:= SHR (IN, N)A, IN and N should be of the typeBYTE, WORD or DWORD. IN will be shifted to the right by N bits and filled with zeros on the left.

Please note, that the amount of bits, which is regarded for the arithmetic operation, is pretended by the data type of the input variable! If the input variable is a constant the smallest possible data type is regarded. The data type of the output variable has no effect at all on the arithmetic operation.

Example in ST:



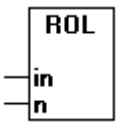
Example In IL:

```
LD 32

SHR 2

ST Var1 (* Result is 8 *)
```

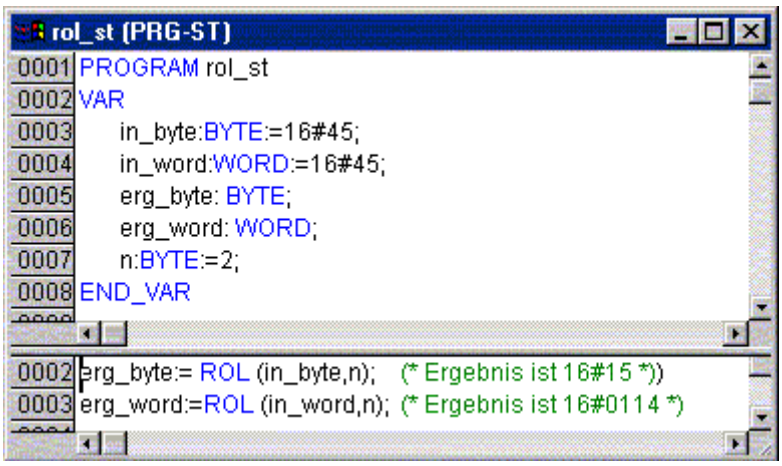
7.2.5.3 ROL



Bitwise rotation of an operand to the left: A:= ROL (IN, N)A, IN and N should be of the typeBYTE, WORD or DWORD. IN will be shifted one bit position to the left N times while the bit that is furthest to the left will be reinserted from the right.

Please note, that the amount of bits, which is regarded for the arithmetic operation, is pretended by the data type of the input variable! If the input variable is a constant the smallest possible data type is regarded. The data type of the output variable has no effect at all on the arithmetic operation.

Example in ST:



Example in IL:

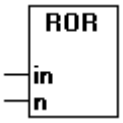
```
Var1 :BYTE;

LD 2#1001_0011

ROL 3

ST Var1 (* Result is 2#1001_1100 *)
```

7.2.5.4 ROR

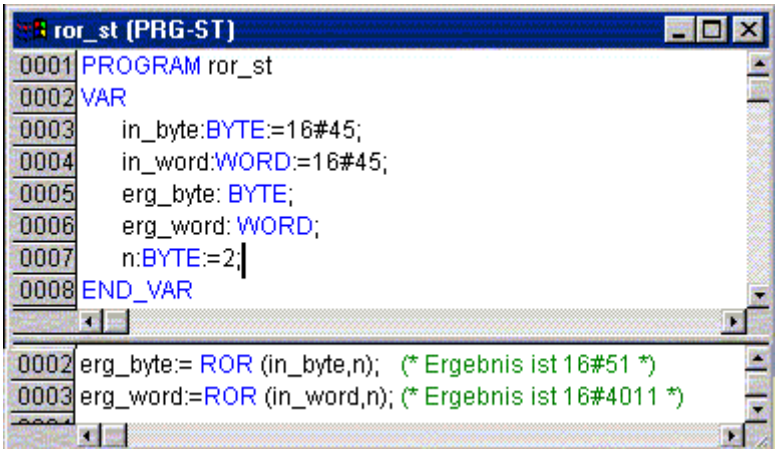


Bitwise rotation of an operand to the right: A:= ROR (IN, N)A, IN and N should be of the typeBYTE, WORD or DWORD. IN will be shifted one bit position to the right N times while the bit that is furthest to the right will be reinserted from the left.

Please note, that the amount of bits, which is regarded for the arithmetic operation, is pretended by the data type of the input variable! If the input variable is a constant the smallest possible data type is regarded. The data type of the output variable has no effect at all on the arithmetic operation.

See in the following example in hexadecimal notation that you get different results for erg_byte and erg_word depending on the data type of the input variable (BYTE or WORD), although the values of the input variables in_byte and in_word are the same.

Example in ST:



Example in IL:

```
Var1 :BYTE;

LD 2#1001_0011

ROR 3

ST Var1 (* Result is 2#0111_0010 *)
```

7.2.6 Selection Operators

7.2.6.1 SEL

Binary Selection.

```
OUT := SEL(G, IN0, IN1)
```

means:

```
OUT := IN0 if G=FALSE;OUT := IN1 if G=TRUE.
```

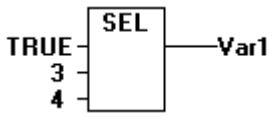
IN0, IN1 and OUT can be any type of variable, G must be BOOL. The result of the selection is IN0 if G is FALSE, IN1 if G is TRUE.

Example in IL:

```
LD TRUE
SEL 3,4
ST Var1 (* Result is 4 *)

LD FALSE
SEL 3,4
ST Var1 (* Result is 3 *)
```

Example in FBD:



For the purpose of the run time optimization, one processes as follows: An expression, which is upstream IN0, is calculated only if G is FALSE. An expression of the IN1 is upstream, is calculated only if G is TRUE!

7.2.6.2 MAX

Maximum function. Returns the greater of the two values.

```
OUT := MAX(IN0, IN1)
```

IN0, IN1 and OUT can be any type of variable.

Example in IL:

```
LD 90

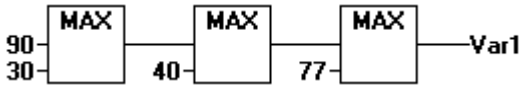
MAX 30

MAX 40

MAX 77

ST Var1 (* Result is 90 *)
```

Example in FBD:



7.2.6.3 MIN

Minimum function. Returns the lesser of the two values.

```
OUT := MIN(IN0, IN1)
```

IN0, IN1 and OUT can be any type of variable.

Example in IL:

```
LD 90

MIN 30

MIN 40

MIN 77

ST Var1 (* Result is 30 *)
```

Example in FBD:



7.2.6.4 LIMIT

Limiting

```
OUT := LIMIT(Min, IN, Max)
```

means:

```
OUT := MIN (MAX (IN, Min), Max)
```

Max is the upper and Min the lower limit for the result. Should the value IN exceed the upper limit Max, LIMIT will return Max. Should IN fall below Min, the result will be Min.

IN and OUT can be any type of variable.

Example in IL:

```
LD 90

LIMIT 30,80

ST Var1 (* Result is 80 *)
```

7.2.6.5 MUX

Multiplexer

```
OUT := MUX(K, IN0,...,INn)
```

means:

```
OUT := INK.
```

IN0, ...,INn and OUT can be any type of variable. K must be BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT or UDINT. MUX selects the Kth value from among a group of values.

Example in IL:

```
LD 0

MUX 30,40,50,60,70,80

ST Var1 (* Result is 30 *)
```

Note that an expression occurring ahead of an input other than INK will not be processed to save run time!

7.2.7 Comparison Operators

7.2.7.1 GT

Greater than

A Boolean operator which returns the value TRUE when the value of the first operand is greater than that of the second. The operands can be BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME and STRING.

Example in IL:

```
LD 20

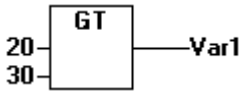
GT 30

ST Var1 (* Result is FALSE *)
```

Example in ST:

```
VAR1 := 20 > 30 > 40 > 50 > 60
> 70;
```

Example in FBD:



7.2.7.2 LT

Less than

A Boolean operator that returns the value TRUE when the value of the first operand is less than that of the second. The operands can be BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME and STRING.

Example in IL:

```
LD 20

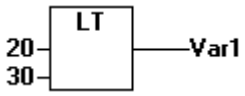
LT 30

ST Var1 (* Result is TRUE *)
```

Example in ST:

VAR1 := 20 < 30;

Example in FBD:



7.2.7.3 LE

Less than or equal to

A Boolean operator that returns the value TRUE when the value of the first operand is less than or equal to that of the second. The operands can be BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME and STRING.

Example in IL:

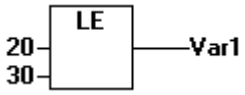
```
LD 20
LE
30

ST Var1 (* Result is TRUE *)
```

Example in ST:

```
VAR1 := 20 <= 30;
```

Example in FBD



7.2.7.4 GE

Greater than or equal to

A Boolean operator that returns the value TRUE when the value of the first operand is greater than or equal to that of the second. The operands can be BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME and STRING.

Example in IL:

```
LD 60

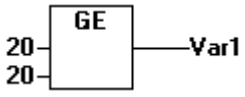
GE 40

ST Var1 (* Result is TRUE *)
```

Example in ST:

```
VAR1 := 60 >= 40;
```

Example in FBD:



7.2.7.5 EQ

Equal to.

A Boolean operator that returns the value TRUE when the operands are equal. The operands can be BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME and STRING.

Example in IL:

```
LD 40

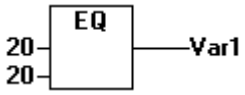
EQ 40

ST Var1 (* Result is TRUE *)
```

Example in ST:

```
VAR1 := 40 = 40;
```

Example in FBD:



7.2.7.6 NE

Not equal to

A Boolean operator that returns that value TRUE when the operands are not equal. The operands can be BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME and STRING.

Example in IL:

```
LD 40

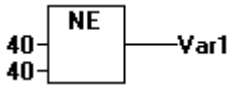
NE 40

ST Var1 (* Result is FALSE *)
```

Example in ST:

```
VAR1 := 40 <> 40;
```

Example in FBD:



7.2.8 Selection of different Operators

7.2.8.1 INDEXOF

Perform this function to find the internal index for a POU.

Example in ST:

```
var1 := INDEXOF(POU2);
```

7.2.8.2 SIZEOF

Perform this function to determine the number of bytes required by the given data type.

Example in IL:

```
arr1:ARRAY[0..4] OF INT;

var1:INT;

LD arr1

SIZEOF

ST var1 (* Result is 10 *)
```

7.2.8.3 ADR (Address Operator)

Address Function

ADR returns the address of its argument in a DWORD. This address can be sent to manufacturing functions to be treated as a pointer or it can be assigned to a pointer within the project.

NOTICE

Address data

After an Online Change there might be changes concerning the data on certain addresses. Please regard this in case of using pointers on addresses.

Example in ST:

```
dwVar := ADR(bVar);
```

Example in IL:

```
LD var1

ADR

ST var2
```

7.2.8.4 ADRINST(Instance address operator)

The ADRINST operator delivers the address of a program instance or function block instance (ADRINST is about equivalent to C++ **this** operator).

7.2.8.5 ^ (Content Operator)

A pointer can be dereferenced by adding the content operator "^" after the pointer identifier.
Example in ST:

```
pt:POINTER TO INT;

var_int1:INT;

var_int2:INT;

pt := ADR(var_int1);

var_int2:=pt^;
```

7.2.8.6 CAL (Calling Operator)

Calling a Function Block

Use CAL in IL to call up a function block instance. The variables that will serve as the input variables are placed in parentheses right after the name of the function block instance.

Example: Calling up the instance Inst from a function block where input variables Par1 and Par2 are 0 and TRUE respectively.

```
CAL INST(PAR1 := 0, PAR2 := TRUE)
```

7.2.8.7 BITADR

BITADR returns the bit address of allocated variable.

Example in ST:

```
bOFF AT%QX10.1 : BOOL;
iBitAdr : BYTE;
iBitAdr := BITADR( bOFF ); (* returns 81*)
```

7.2.9 Type Converting Operators

7.2.9.1 BOOL_TO conversions

Converting from the BOOL type variable to a different type: For number type variables the result is 1 when the operand is TRUE and 0 when the operand is FALSE. The result is "TRUE" or "FALSE" respectively for STRING type variables.

Examples in ST:

```
i:=BOOL_TO_INT(TRUE); (* Result is 1 *)

str:=BOOL_TO_STRING(TRUE); (* Result is 'TRUE' *)

t:=BOOL_TO_TIME(TRUE); (* Result is T#1ms *)

tof:=BOOL_TO_TOD(TRUE); (* Result is TOD#00:00:00.001 *)

dat:=BOOL_TO_DATE(FALSE); (* Result is D#1970-01-01 *)

dandt:=BOOL_TO_DT(TRUE); (* Result is DT#1970-01-01-00:00:01 *)
```

7.2.9.2 TO_BOOL conversions

Conversion from another variable type to BOOL:The result is TRUE when the operand is not equal to 0. The result is FALSE when the operand is equal to 0. The result is true for STRING type variables when the operand is "TRUE", otherwise the result is FALSE.

Examples in ST:

```
b := BYTE_TO_BOOL(2#11010101); (* Result is
TRUE *)

b := INT_TO_BOOL(0); (* Result is FALSE *)

b := TIME_TO_BOOL(T#5ms); (* Result is TRUE *)

b := STRING_TO_BOOL('TRUE'); (* Result is TRUE *)
```

7.2.9.3 STRING_TO conversions

Converting from the variable type STRING to a different type: The operand from the STRING type variable must contain a value that is valid in the target variable type, otherwise the result will be 0.

Examples in ST:

```
b :=STRING_TO_BOOL('TRUE'); (* Result
is TRUE *)

w :=STRING_TO_WORD('abc34'); (* Result is 0 *)

t :=STRING_TO_TIME('T#127ms'); (* Result is T#127ms *)
```

7.2.9.4 TO_STRING conversions

Conversions from different type to STRING.

Examples in ST:

```
str:=BOOL_TO_STRING(TRUE); (* Result is
'TRUE' *)

str :=TIME_TO_STRING(T#12ms); (* Result is 'T#12ms'
*)

str :=DATE_TO_STRING(D#2002-08-18); (* Result is
'D#2002-08-18' *)

str:=TOD_TO_STRING(TOD#14:01:05.123); (* Result is
'TOD#14:01:05.123' *)

str:=DT_TO_STRING(DT#1998-02-13-14:20); (* Result is
'DT#1998-02-13-14:20' *)

k := LREAL_TO_STRING(); (* Result is '1.4' *)
```

7.2.9.5 TIME_TO conversions

Converting from the variable type TIME to a different type: The time will be stored internally in a DWORD in milliseconds. This value will then be converted. When you perform a type conversion from a larger to a smaller type, you risk losing some information. For the STRING type variable, the result is a time constant.

Examples in ST:

```
str :=TIME_TO_STRING(T#12ms); (* Result is
'T#12ms' *)

dw:=TIME_TO_DWORD(T#5m); (* Result is 300000 *)
```

7.2.9.6 DATE_TO conversions

Converting from the variable type DATE to a different type: The date will be stored internally in a DWORD in seconds since Jan. 1, 1970. This value will then be converted. When you perform a type conversion from a larger to a smaller type, you risk losing some information . For STRING type variables, the result is the date constant.

Examples in ST:

```
b :=DATE_TO_BOOL(D#1970-01-01); (* Result is
FALSE *)

i :=DATE_TO_INT(D#1970-01-15); (* Result is 29952 *)

str :=DATE_TO_STRING(D#2002-08-18); (* Result is
'D#2002-08-18' *)

vdt:=DATE_TO_DT(D#2002-08-18); (* Result is DT#2002-08-18-00:00
*)

udw:=DATE_TO_DWORD(D#2002-08-18); (* Result is 16#3D5EE380 *)
```

7.2.9.7 TOD_TO conversions

Converting from the variable type TIME_OF_DAY to a different type: The time will be stored internally in a DWORD in milliseconds (beginning with 12:00 A.M. for the TIME_OF_DAY variable). This value will then be converted. When you perform a type conversion from a larger to a smaller type, you risk losing some information. For the STRING type variable, the result is a time constant.

Examples in ST:

```
si:=TOD_TO_SINT(TOD#00:00:00.012); (* Result
is 12 *)

str:=TOD_TO_STRING(TOD#14:01:05.123); (* Result is
'TOD#14:01:05.123' *)
```

```
tm:= TOD_TO_TIME(TOD#14:01:05.123); (* Result is T#841m5s123ms
*)

udi:= TOD_TO_UDINT(TOD#14:01:05.123); (* Result is 16#03020963
*)
```

7.2.9.8 DT_TO conversions

Converting from the variable type DATE_AND_TIME to a different type: The type will be stored internally in a DWORD in seconds since Jan. 1, 1970. This value will then be converted. When you perform a type conversion from a larger to a smaller type, you risk losing some information . For STRING type variables, the result is the DATE_AND_TIME constant.

Examples in ST:

```
byt :=DT_TO_BYTE(DT#1970-01-15-05:05:05); (*
Result is 129 *)

str:=DT_TO_STRING(DT#1998-02-13-14:20); (* Result is
'DT#1998-02-13-14:20' *)

vtod:=DT_TO_TOD(DT#1998-02-13-14:20); (* Result is TOD#14:20
*)

vdate:=DT_TO_DATE(DT#1998-02-13-14:20); (* Result is D#1998-02-13
*)

vdw:=DT_TO_DWORD(DT#1998-02-13-14:20); (* Result is 16#34E45690
*)
```

7.2.9.9 REAL_TO-/LREAL_TO conversions

Converting from the variable type REAL or LREAL to a different type:
The value will be rounded up or down to the nearest whole number and converted into the new variable type. Exceptions to this are the variable types STRING, BOOL, REAL and LREAL. When you perform a type conversion from a larger to a smaller type, you risk losing some information.

Please regard at a conversion to type STRING that the total number of digits is limited to 16. If the (L)REAL-number has more digits, then the sixteenth will be rounded. If the length of the STRING is defined to short, it will be cut beginning from the right end.

Example in ST:

```
i := REAL_TO_INT(1.5); (* Result is 2 *)

j := REAL_TO_INT(1.4); (* Result is 1 *)

k := LREAL_TO_STRING(); (* Result is '1.4' *)
```

Example in IL:

```
LD 2.7

REAL_TO_INT

GE %MW8
```

7.2.9.10 Conversion between integral number types

Conversion from an integral number type to another number type: When you perform a type conversion from a larger to a smaller type, you risk losing some information. If the number you are converting exceeds the range limit, the first bytes for the number will be ignored.

Example in ST:

```
si := INT_TO_SINT(4223); (* Result is 127
*)
```

If you save the integer 4223 (16#107f in hexadecimals represented) as a SINT variable, it will appear as 127 (16#7f in hexadecimals represented).

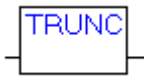
Example in IL:

```
LD 2

INT_TO_REAL

MUL 3.5
```

7.2.9.11 TRUNC



Converting from REAL to INT. The whole number portion of the value will be used. When you perform a type conversion from a larger to a smaller type, you risk losing some information.

Examples in ST:

```
i:=TRUNC(1.9); (* Result is 1 *).
```

```
i:=TRUNC(-1.4); (* Result is -1 *).
```

Example in IL:

```
LD 2.7
TRUNC
GE %MW8
```

Similar Functions

LTRUNC

FLOOR

7.3 Operands

7.3.1 Constants

7.3.1.1 BOOL constants

BOOL-Constants are the logical values TRUE and FALSE.

7.3.1.2 TIME constants

TIME constants can be declared in TwinCAT PLC Control. These are generally used to operate the timer in the standard library. A TIME constant is always made up of an initial "t" or "T" (or "time" or "TIME" spelled out) and a number sign "#".This is followed by the actual time declaration which can include days (identified by "d"), hours (identified by "h"), minutes (identified by "m"), seconds (identified by "s") and milliseconds (identified by "ms"). Please note that the time entries must be given in this order according to length (d before h before m before s before m before ms) but you are not required to include all time increments.

Examples of correct TIME constants in a ST assignment:

```
TIME1 := T#14ms;

TIME1 := T#100S12ms; (*The highest component may be allowed to
exceed its limit*)

TIME1 := t#12h34m15s;
```

would be not correct:

```
TIME1 := t#5m68s; (*limit exceeded in a lower
component*)
```

```
TIME1 := 15ms; (* T# is missing *)

TIME1 := t#4ms13d; (*Incorrect order of entries*)
```

7.3.1.3 DATE constants

These constants can be used to enter dates. A DATE constant is declared beginning with a "d", "D", "DATE" or "date" followed by "#". You can then enter any date with format Year-Month-Day.

Examples:

```
DATE#1996-05-06

d#1972-03-29
```

7.3.1.4 TIME_OF_DAY constants

Use this type of constant to store times of the day. A TIME_OF_DAY declaration begins with "tod#", "TOD#", "TIME_OF_DAY#" or "time_of_day#" followed by a time with the format: Hour:Minute:Second. You can enter seconds as real numbers or you can enter fractions of a second.

Examples:

```
TIME_OF_DAY#15:36:30.123

tod#00:00:00
```

7.3.1.5 DATE_AND_TIME constants

Date constants and the time of day can also be combined to form so-called DATE_AND_TIME constants. DATE_AND_TIME constants begin with "dt#", "DT#", "DATE_AND_TIME#" or "date_and_time#". Place a hyphen after the date followed by the time.

Examples:

```
DATE_AND_TIME#1996-05-06-15:36:30

dt#1972-03-29-00:00:00
```

7.3.1.6 Number constants

Number values can appear as binary numbers, octal numbers, decimal numbers and hexadecimal numbers. If an integer value is not a decimal number, you must write its base followed by the number sign (#) in front of the integer constant. The values for the numbers 10-15 in hexadecimal numbers will be represented as always by the letters A-F. You may include the underscore character within the number.

Examples:

```
14 (Decimal Number)

2#1001_0011 (Binary Number)

8#67 (Octal Number)

16#A (Hexadecimal Number)
```

These number values can be from the variable types BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL or LREAL. Implicit conversions from "larger" to "smaller" variable types are not permitted. This means that a DINT variable cannot simply be used as an INT variable. You must use the type conversion functions from to be able to do this.

7.3.1.7 REAL/LREAL constants

REAL and LREAL constants can be given as decimal fractions and represented exponentially. Use the standard American format with the decimal point to do this.

Example:

```
7.4 instead of 7,4

1.64e+009 instead of 1,64e+009
```

7.3.1.8 STRING constants

A string is a sequence of characters. STRING constants are preceded and followed by single quotation marks. You may also enter blank spaces and special characters (umlauts for instance). They will be treated just like all other characters. In character sequences, the combination of the dollar sign ($) followed by two hexadecimal numbers is interpreted as a hexadecimal representation of the eight bit character code. In addition, the combination of two characters that begin with the dollar sign are interpreted as shown below when they appear in a character sequence:

Characters	Description
$$	Dollar signs
$'	Single quotation mark
$L or $l	Line feed
$N or $n	New line
$P or $p	Page feed
$R or $r	Line break
$T or $t	Tab

Examples:

```
'w1Wüß?'

'Susi und Claus'

':-)'
```

7.3.1.9 Typed literals

Basically, in using IEC constants, the smallest possible data type will be used. If another data type must be used, this can be achieved with the help of typed literals without the necessity of explicitly declaring the constants. For this, the constant will be provided with a prefix which determines the type.

This is written as follows: <Type>#<Literal>
<Type> specifies the desired data type; possible entries are: BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, LREAL. The type must be written in uppercase letters.
<Literal> specifies the constant. The data entered must fit within the data type specified under <Type>.
Example:

```
var1:=DINT#34;
```

If the constant can not be converted to the target type without data loss, an error message is issued:
Typed literals can be used wherever normal constants can be used.

7.3.2 Variables

Variables can be declared either locally in the declaration part of a POU or in the global variable list.
The variable identifier may not contain any blank spaces or special characters, may not be declared more than once and cannot be the same as any of the keywords. Capitalization is not recognized which means that VAR1, Var1, and var1 are all the same variable. The underscore character is recognized in identifiers (e.g., "A_BCD" and "AB_CD" are considered two different identifiers). An identifier may not have more than one underscore character in a row. The first 32 characters are significant. Variables can be used anywhere the declared type allows for them. You can access available variables through the Input Assistant.

7.3.2.1 Addresses

The direct display of individual memory locations is done using special character sequences. These sequences are a concatenation of the percent sign "%", a range prefix, a prefix for the size and one or more natural numbers separated by blank spaces. The following range prefixes are supported:

Prefix	Description
I	Input
Q	Output
M	Memory location

The following size prefixes are supported:

Prefix	Description
X	Single bit
B	Byte (8 Bit)
W	Word (16 Bit)
D	Double word (32 Bit)
*	Config variables (VAR_CONFIG)

```
Examples:
%QX75.1  (*Bit 1 of output byte 75*)
%IW215 (*Input word 215*)
%QB7 (*Output byte 7*)
%MD48 (*Double word in memory position 48 in the memory location*)
```



Boolean values will be allocated bytewise, if no explicit single-bit address is specified

Example: A change in the value of varbool1 AT %QW0 affects the range from QX0.0 to QX0.7.

Memory location

You can use any supported size to access the memory location, however the addressing is always bytewise. For example, the address %MD48 would address bytes numbers 48, 49, 50, and 51 in the memory location area because the size of a DWORD is 4 bytes. You can access words, bytes, and bits in the same way: the address %MX5.0 allows you to access the first bit in the fifth byte.

NOTICE

Using other platforms

For a CX9xxx and CP with ARM platform, the 4-byte alignment is obligatory to be complained.

Example:

```
rMyVar1      AT %MW0       : REAL;
rMyVar2      AT %MW4       : REAL;
```

7.3.2.2 Accessing variables for arrays, structures and POUs

Two-dimensional array components can be accessed using the following syntax:

```
<Fieldname>[Index1, Index2]
```

Structure variables can be accessed using the following syntax:

```
<Structurename>.<Variablenname>
```

Function block and program variables can be accessed using the following syntax:

```
<Functionblockname>.<Variablename>
```

7.3.2.3 Addressing bits in variables

In integer variables, individual bits can be accessed. For this, the index of the bit to be addressed is appended to the variable, separated by a dot. The bit-index can be given by any constant. Indexing is 0-based.
Example:

```
a : INT;
b : BOOL;
 ...
a.2 := b;
```

The third bit of the variable a will be set to the value of the variable b.

If the index is greater than the bit width of the variable, the following error message is issued: Index '<n>' outside the valid range for variable '<var>'!

Bit addressing is possible with the following variable types: SINT, INT, DINT, USINT, UINT, UDINT, BYTE, WORD, DWORD.

If the variable type does not allow it, the following error message is issued: „Invalid data type '<type>' for direct indexing“

A bit access must not be assigned to a VAR_IN_OUT variable!

7.3.2.4 Functions

In ST a function call can also appear as an operand.

Example:

```
Result := Fct(7) + 3;
```

7.3.2.5 System Flags

7.3.2.5.1 System flags

System flags are implicitly declared variables that are different on each specific PLC. To find out which system flags are available in your system, use the command "Insert" "Operand". An Input Assistant dialog box pops up, select the category System Variable.

7.3.2.5.2 SYSTEMINFO

```
VAR_GLOBAL
    SystemInfo     AT%MB32768(*The real address may differ!*) : SYSTEMINFOTYPE;
END_VAR
```

System flags are implicitly declared variables that are different on each specific PLC. The type SYSTEMINFOTYPE is declared in the system library, so this library has to be included in the library manager.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	PLCSystem.Lib
TwinCAT v2.8.0	PC (i386)	TcSystem.Lib

7.3.2.5.3 SYSTEMTASKINFOARR

```
VAR_GLOBAL
    SystemTaskInfoArr       AT%MB32832(*The real address may differ!
*) : ARRAY[1..4] OF SYSTEMTASKINFOTYPE;
END_VAR
```

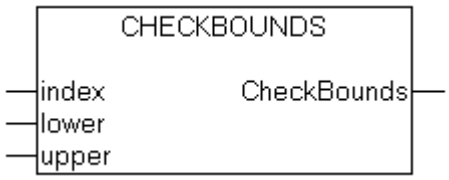
System flags are implicitly declared variables that are different on each specific PLC. The type SYTEMTASKINFOTYPE is declared in the system library, so this library has to be included in the library manager. The index in this array is the task-Id. You can get the task-Id when you call the functionblock GETCURTASKINDEX from your task.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	PLCSystem.Lib
TwinCAT v2.8.0	PC (i386)	TcSystem.Lib

7.4 System Functions

7.4.1 CheckBounds



If you define a function in your project with the name CheckBounds you can use it to check for range overflows in your project! The name of the function is defined and may have only this identifier. An example of how this function is implemented is shown below:

The function can cause a significant increase in system load, so it should only be used for testing purposes.

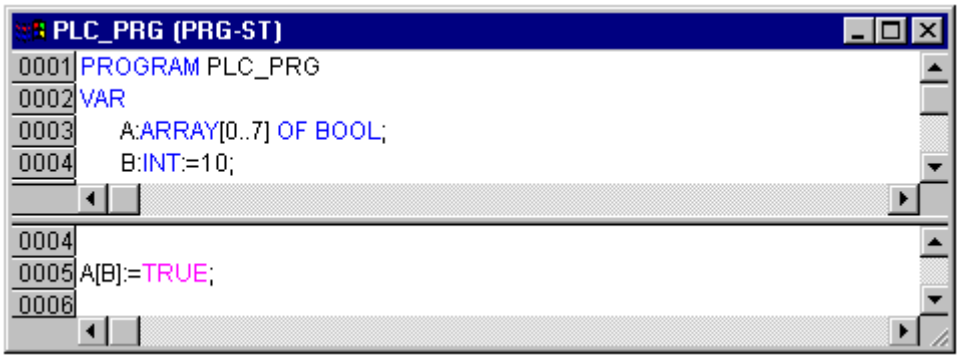
FUNCTION CheckBounds : DINT

```
VAR_INPUT
    index, lower, upper : DINT;
END_VAR
```

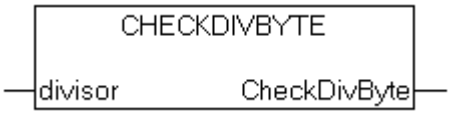
Example of CheckBounds function implementation:

```
IF index<lower THEN
    CheckBounds := lower;
ELSIF index>upper THEN
    CheckBounds := upper;
ELSE
    CheckBounds := index;
END_IF
```

The following typical program for testing the CheckBounds function goes beyond the boundaries of a defined array. The CheckBounds functions makes sure that the value TRUE is not assigned to the position A[10], but rather to the upper area boundary A[7] which is still valid. Therefore, the CheckBounds function can be used to correct extensions beyond array boundaries.



7.4.2 CheckDivByte : BYTE



If you define functions in your project with the name CheckDivByte, you can use them to check the value of the divisor if you use the operator DIV [▶ 198], for example to avoid a division by 0. The name of the function is defined and may have only this identifier.

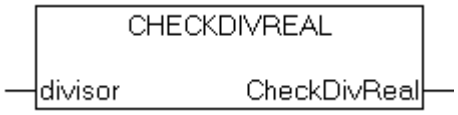
The function can cause a significant increase in system load, so it should only be used for testing purposes.

```
VAR_INPUT
    divisor : BYTE;
END_VAR
```

Example of CheckDivByte implementation:

```
IF divisor = 0 THEN
    CheckDivByte := 1;(**)
ELSE
    CheckDivByte := divisor;
END_IF
```

7.4.3 CheckDivReal : REAL



If you define functions in your project with the name CheckDivReal, you can use them to check the value of the divisor if you use the operator DIV [▶ 198], for example to avoid a division by 0. The name of the function is defined and may have only this identifier.

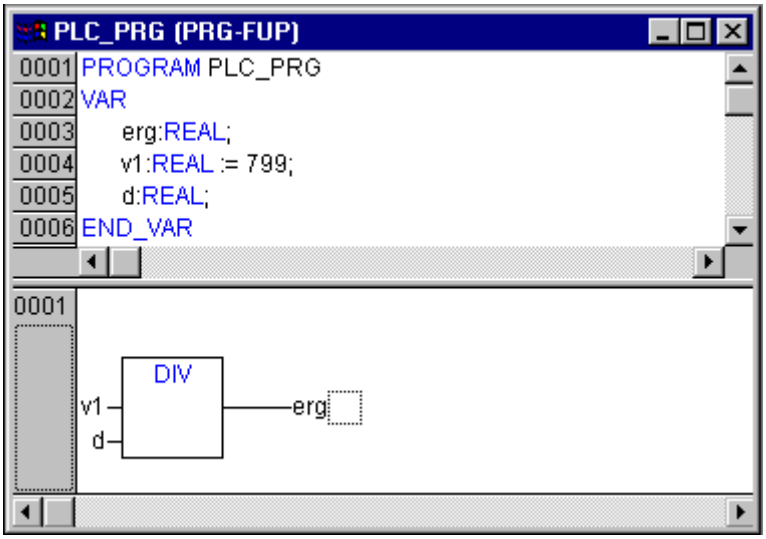
The function can cause a significant increase in system load, so it should only be used for testing purposes.

```
VAR_INPUT
    divisor : REAL;
END_VAR
```

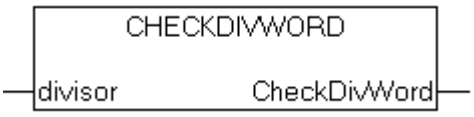
Example of CheckDivReal implementation:

```
IF divisor = 0 THEN
    CheckDivReal := 1;
ELSE
    CheckDivReal := divisor;
END_IF
```

Operator DIV uses the output of function CheckDivReal as divisor. In a program like shown in the following example this avoids a division by 0, the divisor (d) is set from 0 to 1. So the result of the division is 799.



7.4.4 CheckDivWord : WORD



If you define functions in your project with the name CheckDivWord, you can use them to check the value of the divisor if you use the operator DIV [▶ 198], for example to avoid a division by 0. The name of the function is defined and may have only this identifier.

The function can cause a significant increase in system load, so it should only be used for testing purposes.

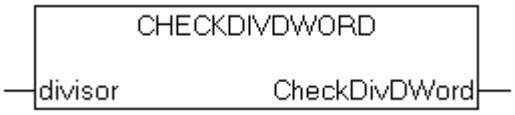
CheckDivWord function may cause increased system load. Please use CheckDivWord only for test purposes.

```
VAR_INPUT
    divisor : WORD;
END_VAR
```

Example of CheckDivWord implementation:

```
IF divisor = 0 THEN
    CheckDivWord := 1;
ELSE
    CheckDivWord := divisor;
END_IF
```

7.4.5 CheckDivDWord : DWORD



If you define functions in your project with the name CheckDivDWord, you can use them to check the value of the divisor if you use the operator DIV [▶ 198], for example to avoid a division by 0. The name of the function is defined and may have only this identifier.

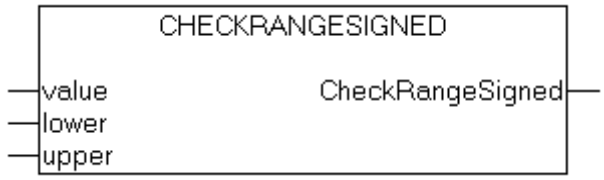
The function can cause a significant increase in system load, so it should only be used for testing purposes.

```
VAR_INPUT
    divisor : DWORD;
END_VAR
```

Example of CheckDivDWord implementation:

```
IF divisor = 0 THEN
    CheckDivDWord := 1;
ELSE
    CheckDivDWord := divisor;
END_IF
```

7.4.6 CheckRangeSigned : DINT



To check for observance of range boundaries at runtime, the functions CheckRangeSigned must be introduced. In these, boundary violations can be captured by the appropriate method and means (e.g. the value can be cut out or an error flag can be set.). They are implicitly called as soon as a variable is written as belonging to a subrange type constructed from a signed type.

```
VAR_INPUT
    value, lower, upper: DINT;
END_VAR
```

Example:

In the case of a variable belonging to a signed subrange type, the function CheckRangeSigned is called; it could be programmed as follows to trim a value to the permissible range:

```
FUNCTION CheckRangeSigned : DINT
VAR_INPUT
    value, lower, upper: DINT;
END_VAR

IF (value < lower) THEN
    CheckRangeSigned := lower;
ELSIF(value > upper) THEN
    CheckRangeSigned := upper;
ELSE
    CheckRangeSigned := value;
END_IF
```

In calling up the function automatically, the function name CheckRangeSigned is obligatory, as is the interface specification: return value and three parameters of type DINT
When called, the function is parameterized as follows:

value	the value to be assigned to the range type
lower	the lower boundary of the range
upper	the upper boundary of the range
Return value	the value that is assigned to the range type

An assignment i:=10*y implicitly produces the following in this example:
i := CheckRangeSigned(10*y, -4095, 4095);
Even if y for example has the value 1000, then i still has only the value 4095 after this assignment.

NOTICE

Value of variable/continuous loop

If neither of the functions CheckRangeSigned or CheckRangeUnsigned [▶ 222] is present, no type checking of subrange types occurs during runtime! The variable i could then take on any value between –32768 and 32767 at any time!
If a function CheckRangeSigned or CheckRangeUnsigned is is implemented as shown above, a continuous loop can develop with the use subrange types in a FOR loop. This occurs exactly if the area indicated for the **FOR loop** is just as large or larger than that subrange types!

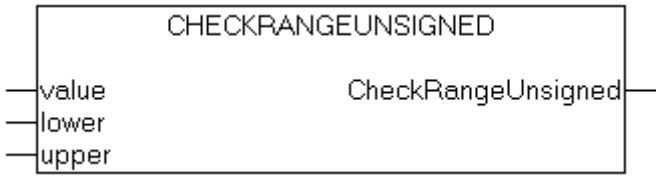
Example:

```
VAR
    ui : UINT (0..10000);
END_VAR

FOR ui:=0 TO 10000 DO
...
END_FOR
```

The FOR loop is not left, because ui cannot became larger than 10000. Likewise, the contents of the CheckRange functions are to be considered when using increment values in the FOR loop!

7.4.7 CheckRangeUnsigned : UDINT



To check for observance of range boundaries at runtime, the functions CheckRangeUnsigned must be introduced. In these, boundary violations can be captured by the appropriate method and means (e.g. the value can be cut out or an error flag can be set.). They are implicitly called as soon as a variable is written as belonging to a subrange type constructed from a unsigned type.

```
VAR_INPUT
    value, lower, upper: UDINT;
END_VAR
```

Example:

IIn the case of a variable belonging to a unsigned subrange type, the function CheckRangeUnsigned is called; it could be programmed as follows to trim a value to the permissible range:

```
FUNCTION CheckRangeUnsigned : UDINT
VAR_INPUT
    value, lower, upper: UDINT;
END_VAR

IF (value < lower) THEN
    CheckRangeUnsigned := lower;
ELSIF(value > upper) THEN
    CheckRangeUnsigned := upper;
ELSE
    CheckRangeUnsigned := value;
END_IF
```

In calling up the function automatically, the function name CheckRangeUnsigned is obligatory, as is the interface specification: return value and three parameters of type UDINT
When called, the function is parameterized as follows:

value	the value to be assigned to the range type
lower	the lower boundary of the range
upper	the upper boundary of the range
Return value	the value that is assigned to the range type

An assignment `i:=10*y` implicitly produces the following in this example:
`i := CheckRangeSigned(10*y, 0, 4095);`
Even if y for example has the value 1000, then i still has only the value 4095 after this assignment.

NOTICE

Value of variable/continuous loop

If neither of the functions CheckRangeSigned [▶ 221] or CheckRangeUnsigned is present, no type checking of subrange types occurs during runtime! The variable i could then take on any value between –32768 and 32767 at any time!
If a function CheckRangeSigned or CheckRangeUnsigned is is implemented as shown above, a continuous loop can develop with the use subrange types in a FOR loop. This occurs exactly if the area indicated for the **FOR loop** is just as large or larger than that subrange types!

Example:

```
VAR
    ui : UINT (0..10000);
END_VAR

FOR ui:=0 TO 10000 DO
...
END_FOR
```

The FOR loop is not left, because ui cannot became larger than 10000. Likewise, the contents of the CheckRange functions are to be considered when using increment values in the FOR loop!

7.5 Compiler Errors

Here you will find the error messages that the parser displays (italics) and possible causes.

Warnings

Number	Error Message	Possible Cause
1100	Unknown function '<name>' in library.	An external library is used. Please check, whether all functions, which are defined in the .hex file, are also defined in the .lib file.
1101	Unresolved symbol '<Symbol>'.	The code generator expects a POU with the name <Symbol>. It is not defined in the project. Define a function/program with this name.
1102	Invalid interface for symbol '<Symbol>'.	The code generator expects a function with the name <Symbol> and exactly one scalar input, or a program with the name <Symbol> and no input or output.
1103	The constant '<name>' at code address '<address>' overwrites a 16K page boundary!	A string constant exceeds the 16K page boundary. The system cannot handle this. It depends on the runtime system whether the problem could be avoided by an entry in the target file. Please contact the PLC manufacturer.
1200	Task '<name>', call of '<name>' Access variables in the parameter list are not updated	Variables, which are only used at a function block call in the task configuration, will not be listed in the cross reference list.
1300	File not found '<name>'	The file, to which the global variable object is pointing, does not exist. Please check the path.
1301	Analyze-Library not found! Code for analyzation will not be generated.	The analyze function is used, but the library analyzation.lib is missing. Add the library in the library manager.
1302	New externally referenced functions inserted. Online Change is therefore no longer possible!	Since the last download you have linked a library containing functions which are not yet referenced in the runtime system. For this reason you have to download the complete project.
1400	Unknown Pragma '<name>' is ignored!	This pragma is not supported by the compiler. See keyword ‘pragma’ for supported directives.
1401	The struct '<name>' does not contain any elements.	The structure with name <name> does not contain any elements. But Variables of this type will use 1 Byte of memory.
1410	'RETAIN' and 'PERSISTENT' do not have any effect in functions	Remanent variables which are defined locally in functions are handled like normal local variables.

Number	Error Message	Possible Cause
1411	Variable '<name>' in the variable configuration isn't updated in any task	The top level instance of the variable is not referenced by a call in any task. Thus it will not be copied from the process image. Example: Variable Configuration: VAR_CONFIG plc_prg.aprg.ainst.in AT %IB0 : INT; END_VAR plc_prg: index := INDEXOF(aprg); The program aprg is referenced but not called. Thus plc_prg.aprg.ainst.in never will get the actual value of %IB0.
1412	Unexpected token '<name>' in pragma {pragma name}	You are using a pragma which is not written correctly resp. which cannot be used at this location. See keyword 'pragma' in the this Online Help for getting help for a correction.
1413	'<name>' is not a valid key for list '<name>'. The key will be ignored	In the pragma a nonexistent parameter list is specified. Check the list name resp. have a look in the Parameter Manager for the currently available lists.
	Expression contains no assignment. No code was generated.	The result of this expression is not used. For this reason there is no code generated for the whole expression.
1501	String constant passed as 'VAR_IN_OUT': '<name>' must not be overwritten!	The constant may not be written within the POU, because there no size check is possible.
1502	Variable '<name>' has the same name as a POU. The POU will not be called!	A variable is used, which has the same name like a POU Example: PROGRAM a ... VAR_GLOBAL a: INT; END_VAR ... a; (* Not POU a is called but variable a is loaded.*)
1503	The POU ‘<name>’ has no outputs. Box result is set to 'TRUE'.	The Output pin of a POU which has no outputs, is connected in FBD or KOP. The assignment automatically gets the value TRUE.

Number	Error Message	Possible Cause
1504	<name>’ (‘<number>’): Statement may not be executed due to the evaluation of the logical expression	Eventually not all branches of the logic expression will be executed. Example: IF a AND funct(TRUE) THEN .... If a has is FALSE then funct will not be called.
1505	Side effect in '<name>'! Branch is probably not executed !	The first input of the POU is FALSE, for this reason the side branch, which may come in at the second input, will not be executed.
1506	Variable '%s' has the same name as a local action. The action will not be called!	Rename the variable or the action.
1507	Instance '<name>' has the same name as a function. The instance will not be called.	You call in ST an instance which has the same name like a function. The function will be called ! Use different names.
1550	Multiple calls of the POU '<name>' in one network may lead to undesired side effects	Check, whether the multiple call of this POU is really necessary. By a multiple call unwanted value overstrikes may occur.
1700	Input box without assignment.	An input box is used in CFC which has no assignment. For this no code will be generated.
1750	Step '<name>': the minimal time is greater than the maximal time!	Open dialog 'Step attributes' for this step and correct the time definitions.
	<name>(element #<element number>): Invalid watchexpression '%s'	The visualization element contains an expression which cannot be monitored. Check variable name and placeholder replacements.
1801	'<name> (number): No Input on Expression '<name>' possible	In the configuration of the visualization object at field input a composed expression is used. Replace this by a single variable.
1802	<Visualization object>(Element number): Bitmap '<name>' was not found	Make sure, that an external bitmap-file is available in that path which is defined in the visualization configuration dialog.
1803	'<name>'('<number>'): "The print action would not supported for web- and target visualization.	A print action is assigned to an alarm configured in the visualization. This will not be regarded in the Web- or Target-Visualization.
1804	'<name>'('<number>'): The font '<name>' is not supported by the target.	In the visualization you are using a font, which is not supported by the target system. See in the target settings, category 'Visualization' for the supported fonts.
1805	'<name>'('<number>'): 'Store trend data in PLC' should be set.	You are using a visualization element for storing trend data. This however will not be regarded on the PLC, because option 'Store trend data' is not activated in the target settings, category Visualization.

Number	Error Message	Possible Cause
1806	'<name>'('<number>'): The target setting 'Alarm handling in the PLC' should be set.	You are using an element for alarm visualization. This however will not be regarded on the PLC because option 'Alarm handling in the PLC' is not activated in the target settings, category 'Visualization'.
1807	<name> (<number>): No message window for alarms for target visualization	Regard that action "message" is not supported for the Target-Visualization!
1850	Input variable at %IB<number> is used in task '<name>' but updated in another task	Please check which tasks are using this variable and whether the current programming is not causing undesirable effects. The update of the variable value usually is done in the task with the highest priority.
1851	Output variable at %IQ<number> is used in task '<name>' but updated in another task	Please check which tasks are using this variable and whether the current programming is not causing undesirable effects. The update of the variable value usually is done in the task with the highest priority.
1852	CanOpenMaster might not be called cyclically in event task '<name>'! Set modul parameter UpdateTask!	Currently the CanOpen Master is called by the named event task. If you want to get it called cyclically, specify an appropriate task via parameter UpdateTask in the PLC Configuration in dialog 'Module parameters'.
1853	A PDO (index: '<number>') might not be updated cyclically in event task '<name>'	Currently the named PDO is controlled via the named event task. But if you want to get it called cyclically, you must assign an appropriate task to the PDO by shifting IO-references to this task.
	POU '<name>' (main routine) is not available in the library	The Start-POU (z.B. PLC_PRG) will not be available, when the project is used as library.
1901	Access Variables and Variable Configurations are not saved in a library!	Access variables and variable configuration are not stored in the library.
1902	'<name>': is no Library for the current machine type!	The .obj file of the lib was generated for another device.
1903	<name>: is no valid Library	The file does not have the format requested for the actual target.
1904	The constant '<name>' hides a constant of the same name in a library	In your project you have defined a constant which has the same name like one which is defined in a linked library. The library variable will be overwritten !
1970	Parameter manager: List '<name>' , Column '<name>', Value '<name>' could not be imported!	Check the Import-file *.prm for entries which do not match the current configuration (standard values resp. XML-description file) of the Parameter Manager.

Number	Error Message	Possible Cause
1980	Global network variables '<name>' '<name>': simultaneous reading and writing may result in loss of data!	In the configuration of the network variables list (Select list in the Resources tab and open dialog 'Global variables list' via command 'Properties' in the context menu) options 'Read' and 'Write' are activated. Regard that this might result in data losses during communication.
1990	No 'VAR_CONFIG' for '<name>'	For this variable there is no address configuration available in the Variable_Configuration (VAR_CONFIG). Open window Variable_Configuration in the Ressources tab and there insert the appropriate configuration (Command 'Insert 'All instance paths').
	Task '<task name>': no cycle time specified for cyclic task	In the Task configuration a cyclic task has been created, for which no cycle time has been defined. Enter an appropriate time span in dialog 'Taskattributes' at "Interval".

Compiler Errors

Number	Error Message	Possible Cause
	Code too large. Maximum size: '<number>' Byte (<number>K)	The maximum program size is exceeded. Reduce project size.
3101	Total data too large. Maximum size: '<number>' Byte (<number>K)	Memory is exceeded. Reduce data usage of the application.
3110	Error in library file '<name>'.	The .hex file is not in INTEL Hex format.
3111	Library '<name>' is too large. Maximum size: 64K	The .hex file exceeds the set maximum size.
3112	Nonrelocatable instruction in library.	The .hex file contains a nonrelocatable instruction. The library code cannot be linked.
3113	Library code overwrites function tables.	The ranges for code and function tables are overlapping.
3114	Library uses more than one segment.	The tables and the code in the .hex file use more than one segment.
3115	Unable to assign constant to VAR_IN_OUT. Incompatible data types.	The internal pointer format for string constants cannot get converted to the internal pointer format of VAR_IN_OUT, because the data are set "near" but the string constants are set " huge" or "far". If possible change these target settings.
3116	Function tables overwrite library code or a segment boundary.	Code 166x: The external library cannot be used with the current target settings. These must be adapted resp. the library must be rebuilt with appropriate settings.
3117	<Name> (<Zahl>): Expression too complex. No more registers available	The named expression is to complex to be handled by the available registers. Please try to reduce the expression by using interim variables.
3120	Current code-segment exceeds 64K.	The currently generated code is bigger than 64K. Eventually to much initializing code is created.
3121	POU too large." A POU may not exceed the size of 64K.	A POU may not exceed the size of 64K.
3122	Initialisation too large. Maximum size: 64K	The initialization code for a function or a structure POU may not exceed 64K.
3123	Data segment too large: segment '<Number>%s', size <size> bytes (maximum <number> bytes)	Please contact your manufacturer.
3124	String-constant too large: <Number> characters (Maximum 253 characters)	The given constant must be reduced in number of characters.
3130	User-Stack too small: '<number>' DWORD needed, '<number>' DWORD available.	The nesting depth of the POU calls is to big. Enter a higher stack size in the target settings or compile build project without option ‚Debug' (can be set in dialog 'Project' 'Options' 'Build').
3131	User-Stack too small: '<number>' WORD needed, '<number>' WORD available.	Please contact the PLC manufacturer.
3132	System-Stack too small: '<number>' WORD needed, '<number>' WORD available.	Please contact the PLC manufacturer.
3150	Parameter <number> of function '<name>': Cannot pass the result of a IEC-function as string parameter to a C-function.	Use a intermediate variable, to which the result of the IEC function is assigned.
3160	Can't open library file '<name>'.	A library <name> is included in the library manager for this project, but the library file does not exist at the given path.
3161	Library '<name>' contains no codesegment	A .obj file of a library at least must contain one C function. Insert a dummy function in the .obj file, which is not defined in the .lib file.

Number	Error Message	Possible Cause
3162	Could not resolve reference in Library '<name>'(Symbol '<name>', Class '<name>', Type '<name>')	The .obj file contains a not resolvable reference to another symbol. Please check the settings of the C-Compiler.
3163	Unknown reference type in Library '<name>' (Symbol '<name>' , Class '<name>' , Type '<name>')	The .obj file contains a reference type, which is not resolvable by the code generator. Please check -the settings of the C-Compiler.
	"%s (%d): Boolean expression to complex	The temporary memory of the target system is insufficient for the size of the expression. Divide up the expression into several partial expressions thereby using assignments to intermediate variables.
3201	<name> (<network>): A network must not result in more than 512 bytes of code	Internal jumps can not be resolved. Activate option "Use 16 bit Sprungoffsets" in the 68k target settings.
3202	Stack overrun with nested string/array/structure function calls	A nested function call CONCAT(x, f(i)) is used. This can lead to data loss. Divide up the call into two expressions.
3203	Expression too complex (too many used address registers).	Divide up the assignment in several expressions.
3204	A jump exceeds 32k Bytes	Jump distances may not be bigger than 32767 bytes.
3205	Internal Error: Too many constant strings" In a POU there at the most 3000 string constants may be used.	In a POU there at the most 3000 string constants may be used.
3206	Function block data exceeds maximal size	A function block may produce maximum 32767 Bytes of code.
3207	Array optimization	The optimization of the array accesses failed because during index calculation a function has been called.
3208	Conversion not implemented yet	A conversion function is used, which is not implemented for the actual code generator.
3209	Operator not implemented	A operator is used, which is not implemented for this data type and the actual code generator. MIN(string1,string2).
3210	Function '<name>' not found	A function is called, which is not available in the project.
3211	Max string usage exceeded	A variable of type string can be used in one expression 10 times at the most.
3212	Wrong library order at POU <POU name>	The order of libraries for this POU does not match with that in the cslib.hex file. Correct the order accordingly. (only for 68K targets, if the checking option is activated in the target file.)
3250	Real not supported for 8 Bit Controller	The target is currently not supported.
3251	date of day types are not supported for 8 Bit Controller	The target is currently not supported.
3252	size of stack exceeds <number> bytes	The target is currently not supported.
3253	Could not find hex file: '<name>'	The target is currently not supported.
3254	Call to external library function could not be resolved.	The target is currently not supported.
	An error occurred during import of Access variables	The .exp file contains an incorrect access variables section.
3401	An error occurred during import of variable configuration	The .exp file contains an incorrect configuration variables section.
3402	An error occurred during import of global variables	The .exp file contains an incorrect global variables section.

Number	Error Message	Possible Cause
3403	Could not import <name>	The section for object <name> in the .exp file is not correct.
3404	An error occurred during import of task configuration	The section for the task configuration the .exp file is not correct.
3405	An error occurred during import of PLC configuration	The section for the PLC configuration in the .exp file is not correct.
3406	Two steps with the name '<name'. Second step not imported.	The section for the SFC POU in the .exp file contains two steps with equal names. Rename one of the steps in the export file.
3407	Predecessor step '<name>' not found	The step <name> is missing in the .exp file.
3408	Successor step '<name>' not found	The step <name> is missing in the .exp file.
3409	No successional transition for step '<´name>'	In the .exp file a transition is missing, which requires step <name> as preceeding step.
3410	No successional step for transition '<name>'	In the .exp file a step is missing which requires the transition <name> as preceeding condition.
3411	Step '<name>' not reachable from initial step	In the .exp file the connection between step <name> and the initial step is missing.
3412	Macro '<name>' not imported	Check the export file.
3413	Error during import of the CAMs.	You have imported an export file (*.exp) which contains erroneous information on a CAM. Check the export file.
3414	Error during import of the CNC program list	You have imported an export file (*.exp) which contains erroneous information on a CNC program. Check the export file.
3415	Error during import of the Alarm configuration	You have imported an export file (*.exp) which contains erroneous information on the Alarm Configuration. Check the export file.
3450	PDO'<PDO-name>': Missing COB-Id!	Click on the button ‚Properties' in the PLC configuration dialog for the module and enter a COB ID for the PDO <PDO Name>.
3451	Error during load: EDS-File '<name>' could not be found, but is referenced in hardware configuration!	Eventually the device file needed for the CAN configuration is not in the correct directory. Check the directory setting for configuration files in ‚Project' 'Options' 'Directories'.
3452	The module '<name>' couldn't be created!	The device file for module <name> does not fit to the current configuration. Eventually it has been modified since the configuration has been set up or it is corrupted.
3453	The channel '<name>' couldn't be created!	The device file for channel <name> does not fit to the current configuration. Eventually it has been modified since the configuration has been set up or it is corrupted.
3454	The address '<name>' points to an used memory!	Option 'Check for overlapping addresses' is activated in the dialog ‚Settings' of the PLC configuration and an overlap has been detected. Regard, that the area check is based on the size which results of the data types of the modules, the size which is given by the entry ‚size' in the configuration file.
3455	Error during load: GSD-File '<name>' could not be found, but is referenced in hardware configuration!	Eventually the device file required by the Profibus configuration is not in the correct directory. . Check the directory setting for configuration files in ‚Project' 'Options' 'Directories'.

Number	Error Message	Possible Cause
3456	The profibus device '<name>' couldn't be created!	The device file for module <name> does not fit to the current configuration. Eventually it has been modified since the configuration has been set up or it is corrupted.
3457	Error in module description!	Please check the device file of this module.
3458	The PLC-Configuration couldn't be created! Check the configuration files.	Check if all required configuration and device files are available in the correct path (see defined compile directory in 'Project' 'Options' / Directories)
3460	3S_CanDrv.lib has the wrong version.	Make sure, that the 3S_CanDrv.lib which is included in the project is up to date.
3461	3S_CanOpenMaster.lib has the wrong version.	Make sure, that the 3S_CanOpenMaster.lib which is included in the project is up to date.
3462	3S_CanOpenDevice.lib has the wrong version.	Make sure, that the 3S_CanOpenDevice.lib which is included in the project is up to date.
3463	3S_CanOpenManager.lib has the wrong version.	Make sure, that the 3S_CanOpenManager.lib which is included in the project is up to date.
3464	3S_CanNetVar.lib has the wrong version.	Make sure, that the 3S_CanNetVar.lib which is included in the project, is up to date.
3465	CanDevice: Sub indices have to be numerated sequentially	In parameter lists used by the CanDevice the subindices must be numbered sequentially and without interruption. Check the corresponding list in the Parameter Manager.
3466	CAN network variables: No CAN controller found in the PLC configuration	There are network variables configured for a CAN network (Ressources, Global Variables), but in the PLC Configuration there is no CAN Controller available.
3468	CanDevice: Update task not available in the task configuration.	The update task (used for calling the CANdevice), which is defined in the Base Settings dialog of the CANdevice in the PLC Configuration, must be configured in the Task Configuration of the project.
3469	The CanOpenMaster can not be called. Please assign a task manually.	Assign a task, which should call the Master, via parameter UpdateTask in the Module parameters dialog in the PLC Configuration.
3470	Invalid name in parameter UpdateTask	Open the CanMasters Module parameter dialog in the PLC Configuration. Check parameter UpdateTask. The specified task must be available in the project. If you cannot set an appropriate task here, the device file must be checked for the corresponding value definitions for UpdateTask.
	No 'VAR_CONFIG' for '<name>'	Insert a declaration for this variable in the global variable list which contains the 'Variable_Configuration'.
3501	No address in 'VAR_CONFIG' for '<name>'.	Assign an address to this variable in the global variable list which contains the 'Variable_Configuration'.
3502	Wrong data type for '<name>' in 'VAR_CONFIG	In the global variables list which contains the ‚Variable_Configuration' the variable is declared with a different data type than in the POU.
3503	Wrong data type for '<name>' in 'VAR_CONFIG'	In the global variables list which contains the ‚Variable_Configuration' the variable is declared with a different address than in the POU.

Number	Error Message	Possible Cause
3504	Initial values are not supported for 'VAR_CONFIG	A variable of the ‚Variable_Configuration' is declared with address and initial value. But an initial value can only be defined for input variables without address assignment.
3505	'<name>'is no valid instance path	The Variable_Configuration contains a nonexisting variable.
3506	Access path expected	In the global variable list for Access Variables the access path for a variable is not correct. Correct: <Identifier>:'<Access path>':<Type> <Access mode>
3507	No address specification for 'VAR_ACCESS'-variables	The global variable list for Access Variables contains an address assignment for a variable. This is not allowed. Valid variable definition: <Identifier>:'<Access path>':<Type> <Access mode>
3550	Duplicate definition of identifier '<name>'	There are two tasks are defined with an identic same name. Rename one of them.
3551	The task '<name>' must contain at least one program call	Insert a program call or delete task.
3552	Event variable '<name>' in task '%s' not defined	There is an event variable set in the ‘Single’ field of the task properties dialog which is not declared globally in the project. Use another variable or define the variable globally.
3553	"Event variable '<name>' in task '%s' must be of type 'BOOL'	Use a variable of type BOOL as event variable in the ‘Single’ field of the task properties dialog.
3554	Task entry '<name>' must be a program or global function block instance	In the field ‚Program call' a function or a not defined POU is entered. Enter a valid program name.
3555	The task entry '<name>' contains invalid parameters	In the field ‚Append program call' there are parameters used which do not comply with the declaration of the program POU.
3556	Tasks are not supported by the currently selected target	The currently defined task configuration cannot be used for the currently set target system. Change target or modify the task configuration correspondingly.
3557	Maximum number of Tasks ('<number>') exceeded	The currently defined number of tasks exceeds the maximum number allowed for the currently set target system. Change target or modify the task configuration correspondingly. Attention: Do not edit the XML description file of the task configuration!
3558	Priority of task '<name>' is out of valid range between '<lower limit>' and '<upper limit>'	The currently defined priority for the task is not valid for the currently set target system. Change target or modify the task configuration correspondingly.
3559	Task '<name>': Interval-Tasks are not supported by the current target	The current task configuration contains an interval task. This is not allowed by the currently set target system. Change target or modify the task configuration correspondingly.
3560	Task '<name>': free wheeling tasks are not supported by the current target	The current task configuration contains an free wheeling task. This is not allowed by the currently set target system. Change target or modify the task configuration correspondingly.
3561	Task '<name>': event tasks are not supported by the current target	The current task configuration contains event tasks which are not supported by the currently set target system. Change target or modify the task configuration correspondingly.

Number	Error Message	Possible Cause
3562	Task '<name>': external event tasks are not supported by the current target	The current task configuration contains external event tasks which are not supported by the currently set target system. Change target or modify the task configuration correspondingly.
3563	The interval of task '<name>' is out of valid range between '<lower limit>' and '<upper limit>'	Change the interval value in the configuration dialog for the task.
3564	The external event '<name>' of task '<name>' is not supported by the current target	The currently set target system does not support the external event which is defined in the task configuration for this task. Change target or modify the task configuration correspondingly.
3565	Maximum number of event tasks ('<number>') exceeded	The currently set target system does not allow as many event tasks as are defined at the moment. Change target or modify the task configuration correspondingly.
3566	Maximum number of interval tasks ('<number>') exceeded	The currently set target system does not allow as many interval tasks as defined at the moment. Change target or modify the configuration correspondingly.
3567	Maximum number of free wheeling tasks ('<number>') exceeded	The currently set target system does not allow as many free wheeling tasks as defined at the moment. Change target or modify the configuration correspondingly.
3568	Maximum number of external interval tasks ('<number>') exceeded	The currently set target system does not allow as many external interval tasks as defined at the moment. Change target or modify the configuration correspondingly.
3569	POU '<name>' for system event '<name>' not defined	The POU which should be called by the named system event, as defined in the task configuration, is not available in the project. Modify the task configuration correspondingly or make sure that the POU is available in the project.
3570	The tasks '<name>' and '<name>' share the same priority	Modify the task configuration so that each task has a different priority.
3571	The library 'SysLibCallback' is not included in the project! System events can not be generated.	In order to create event tasks, the SysLibCallback.lib is needed. Link this library to the project in the library manager or modify the task configuration (task attributes) in that way that there is no task triggered by an event.
3575	Task <'name>': the cycle time has to be a multiple of <number> µs.	Correct the cycle time accordingly in the Taskattributes dialog for this task. The target system defines a base time and prescribes that the cycle time must be equal to or be a multiple of this base time.
	Implicit variables not found!	Use command ‚Rebuild all'. If nevertheless you get the error message again please contact the PLC manufacturer.
3601	<name> is a reserved variable name	The given variable is declared in the project, although it is reserved for the codegenerator. Rename the variable.
3610	'<name>' not supported	The given feature is not supported by the current version of the programming system.
3611	The given compile directory '<name>' is invalid	There is an invalid directory given in the ‚Project' ‚Options' ‚Directories' for the Compile files.

Number	Error Message	Possible Cause
3612	Maximum number of POUs (<number>) exceeded! Compile is aborted.	Too many POUs and data types are used in the project. Modify the maximum number of POUs in the Target Settings / Memory Layout.
3613	Build canceled	The compile process was cancelled by the user.
3614	Project must contain a POU named '<name>' (main routine) or a task configuration	Create an init POU of type Program (e.g. PLC_PRG) or set up a task configuration.
3615	<name> (main routine) must be of type program	A init POU (e.g. PLC_PRG) is used in the project which is not of type Program.
3616	Programs musn't be implemented in external libraries	The project which should be saved as an external library contains a program. This will not be available, when the library will be used.
3617	Out of memory	Increase the virtual memory capacity of your computer.
3618	BitAccess not supported in current code generator!	The code generator for the currently set target system does not support bit access on variables.
3619	Object file '<name>' and library '<name>' have different versions!	Make sure that for the library there are available matching versions of *.lib and *.obj resp. *.hex files. These files must have the very same time stamp.
3620	The POU '<name>' must not be present inside a library	You want to save the project as a library of version 2.1. In this version a library may not contain a PLC_PRG object. Use a different POU name.
3621	"Cannot write compile file '<name>'"	Probably in the path which is specified for the compile file there is already a file of the same name, which is "read only". Remove that file resp. change the access rights.
3622	"The symbol file '<name>' could not be created"	Probably in the path which is specified for the symbol file (usually project directory) there is already a file of the same name, which is "read only". Remove that file resp. change the access rights.
3623	"Cannot write boot project file '<name>'"	Probably in the path which is specified for the symbol file (target specific) there is already a file of the same name, which is "read only". Remove that file resp. change the access rights.
3624	"Target setting <targetsetting1>=<set value> not compatible with <targetsetting2>=<set value>"	Check and correct these settings in the Targetsettings dialogs (Ressources tab). If the settings are not visible resp. editable there, please contact the PLC Manufacturer.
	POU with name ‘<name>' is already in library '<name>'	A POU name is used in the project, which is already used for a library POU. Rename the POU.
3701	Name used in interface is not identical with POU Name	Use command ‘Project’ ‘Rename object’ to rename the POU in the object organizer, or change the name of the POU in the declaration window. There the POU name has to be placed next to one of the keywords PROGRAM, FUNCTION or FUNCTIONBLOCK.
3702	Overflow of identifier list	Maximum 100 identifiers can be entered in one variable declaration.
3703	Duplicate definition of identifier '<name>'	Take care that there is only one identifier with the given name in the declaration part of the POU.
3704	data recursion: <POU 0> -> <POU 1> -> .. -> <POU 0>	A FB instance was used, which needs itself.

Number	Error Message	Possible Cause
3705	<Name>: VAR_IN_OUT in Top-Level-POU not allowed, if there is no Task-Configuration	Create a task configuration or make sure that there are no VAR_IN_OUT variables used in PLC_PRG.
3720	Address expected after 'AT'	Add a valid address after the keyword AT or modify the keyword.
3721	Only 'VAR' and 'VAR_GLOBAL' can be located to addresses	Put the declaration to a VAR or VAR_GLOBAL declaration area.
3722	Only 'BOOL' variables allowed on bit addresses	Modify the address or modify the type of the variable to which the address is assigned.
3726	Constants can not be laid on direct addresses	Modify the address assignment correspondingly.
3727	No array declaration allowed on this address	Modify the address assignment correspondingly.
3728	Invalid address: '<address>'	This address is not supported by the PLC configuration. Check PLC configuration resp. modify address.
3729	Invalid type '<name>' at address: '<name>'	The type of this variable cannot be placed on the given address. Example: For a target system working with ‘alignment 2’ the following declaration is not valid: var1 AT %IB1:WORD;
3740	Invalid type: '<name>'	An invalid data type is used in a variable declaration.
3741	Expecting type specification	A keyword or an operator is used instead of a valid type identifier.
3742	Enumeration value expected	In the definition of the enumeration type an identifier is missing after the opening bracket or after a comma between the brackets.
3743	Integer number expected	Enumerations can only be initialized with numbers of type INT.
3744	Enum constant '<name>' already defined.	Check if you have followed the following rules for the definition of enumeration values: - Within one enum definition all values have to be unique. - Within all global enum definitions all values have to be unique. - Within all local enum definitions all values have to be unique
3745	Subranges are only allowed on Integers!	Subrange types can only be defined resting on integer data types.
3746	Subrange '<name>' is not compatible with Type '<name>'	One of the limits set for the range of the subrange type is out of the range which is valid for the base type.
3747	unknown string length: '<name>'	There is a not valid constant used for the definition of the string length.
3748	More than three dimensions are not allowed for arrays	More than the allowed three dimensions are given in the definition of an array. If applicable use an ARRAY OF ARRAY.
3749	lower bound '<name>' not defined	There is a not defined constant used to define the lower limit for a subrange or array type.
3750	upper bound '<name>' not defined	There is a not defined constant used to define the upper limit for a subrange or array type.
3751	Invalid string length '<number of characters>'	The here defined string length exceeds the maximum value which is defined for the currently set target system.

Number	Error Message	Possible Cause
3752	More than 9 dimensions are not allowed for nested arrays	An array can be 1- 2- or 3-dimensional. The number of dimensions reached by nesting of arrays (e.g "arr: ARRAY [0..2,0..2,0..2] OF ARRAY [0..2,0..2,0..2] OF ARRAY [0..2,0..2,0..2, 0..2] OF DINT" maximum may be 9 and is exceeded in the current error case. Reduce appropriately to max. 9 dimensions.
3760	Error in inital value	Use an initial value which corresponds to the type definition. To change the declaration you can use the declaration dialog for variables (Shift/F2 or 'Edit"Autodeclare').
3761	'VAR_IN_OUT' variables must not have an inital value.	Remove the initialization at the declaration of the VAR_IN_OUT variable.
3780	VAR, VAR_INPUT, VAR_OUTPUT or VAR_IN_OUT expected	The first line following the name of a POU must contain one of these keywords.
3781	"'END_VAR' or identifier expected	Enter a valid identifier of a END_VAR at the beginning of the given line in the declaration window.
3782	Unexpected end	In the declaration editor: Add keyword END_VAR at the end of the declaration part. In the texteditor of the programming part: Add an instruction which terminates the last instruction sequence (e.g. END_IF).
3783	END_STRUCT' or identifier expected	Ensure that the type declaration is terminated correctly.
3784	The current target doesn't support attribute <attribute name>	The target system does not support this type of variables (e.g. RETAIN, PERSISTENT)
	The global variables need too much memory. Increase the available memory in the project options.	Increase the number of segments given in the settings in dialog ‚Project' ‚Options' ‚Build'.
3801	The variable '<name>' is too large (<Size> Byte)	The variable uses a type which is bigger than 1 data segment. The segment size is a target specific parameter and can be modified in the target settings/memory layout. If you do not find this in the current target settings, please contact your PLC manufacturer.
3802	Out of retain memory. Variable '<name>', <number> bytes.	The memory space available for Retain variables is exhausted. The size of the memory area can be set target-specific in the target settings /memory layout. If you do not find the settings field in the dialog, please contact your PLC manufacturer. (Please regard: If retain variables are used in an function block instance, the complete instance POU will be stored in the retain memory area !)
3803	Out of global data memory. Variable '<name>', ‚<number>' bytes.	The memory space available for global variables is exhausted. The size of the memory area can be set target-specific in the target settings /memory layout. If you do not find the settings field in the dialog, please contact your PLC manufacturer.
3820	"'VAR_OUTPUT' and 'VAR_IN_OUT' not allowed in functions	In a function no output or in_output variables may be defined.
3821	At least one input required for functions	Add at least on input parameter for the function.
3840	Unknown global variable '<name>'!	In the POU a VAR_EXTERNAL variable is used, for which no global variable declared.

Number	Error Message	Possible Cause
3841	Declaration of '<name>' do not match global declaration!	The type given in the declaration of the VAR_EXTERNAL variable is not the same as that in the global declaration.
3850	Declaration of an unpacked struct '<name>' inside a packed struct '<name>' is not allowed!	This structure definition leads to misalignment in the memory. Modify the definition appropriately.
	Multiple underlines in indentifier	Remove multiple underlines in the identifier name.
3901	At most 4 numerical fields allowed in addresses	There is a direct assignment to an address which has more than four levels. (e.g. %QB0.1.1.0.1).
3902	Keywords must be uppercase	Use capital letters for the keyword or activate option ‚Autoformat' in ‚Project' ‚Options'.
3903	Invalid duration constant	The notation of the constant does not comply with the IEC61131-3 format.
3904	Overflow in duration constant.	The value used for the time constant cannot be represented in the internal format. The maximum value which is representable is t#49d17h2m47s295ms
3905	Invalid date constant	The notation of the constant dies not comply with the IEC61131-3 format.
3906	Invalid time of day constant	The notation of the constant dies not comply with the IEC61131-3 format.
3907	Invalid date and time constant	The notation of the constant dies not comply with the IEC61131-3 format.
3908	Invalid string constant	The string constant contains an invalid character.
	Identifier expected	Enter a valid identifier at this position.
4001	Variable '<name>' not declared	Declare variable local or global.
4010	Type mismatch: Cannot convert '<name>' to '<name>'.	Check what data type the operator expects (Browse Online Help for name of operator) and change the type of the variable which has caused the error, or select another variable.
4011	Type mismatch in parameter '<name>' of '<name>': Cannot convert '<name>' to '<name>'.	The data type of the actual parameter cannot be automatically converted to that of the formal parameter. Use a type conversion or use another variable type.
4012	Type mismatch in parameter '<name>' of '<name>': Cannot convert '<name>' to '<name>'.	A value with the invalid type <Typ2> is assigned to the input variable '<name>'. Replace the variable or constant to one of type <Typ1> or use a type conversion respectively a constant with type-prefix.
4013	Type mismatch in output '<name>' of '<name>': Cannot convert '<name>' to '<name>'.	A value with the invalid type <Typ2> is assigned to the output variable '<name>'. Replace the variable or constant to one of type <Typ1> or use a type conversion respectively a constant with type-prefix.
4014	Typed literal: Cannot convert '<name>' to '<name>	The type of the constant is not compatible with the type of the prefix. Example: SINT#255
4015	Data type '<name>' illegal for direct bit access	Direct bit addressing is only allowed for Integer- and Bitstring data types. You are using a variable var1 of type REAL/LREAL or a constant in bit access <var1>.<bit>.
4016	Bit index '<number>' out of range for variable of type '<name>	You are trying to access a bit which is not defined for the data type of the variable.
4017	'MOD' is not defined for 'REAL	The operator MOD can only be used for integer and bitstring data types.

Number	Error Message	Possible Cause
4020	Variable with write access or direct address required for 'ST', 'STN', 'S', 'R'	Replace the first operand by a variable with write access.
4021	No write access to variable '%s' allowed	Replace the variable by a variable with write access.
4022	Operand expected	Add an operand behind the command.
4023	Number expected after '+' or '-'	Enter a digit.
4024	<operator 0> or <operator 1> or ... expected before '<name>'	Enter a valid operand at the named position.
4025	':=' or '=>' expected before '<name>'	Enter one of the both operators at the named position.
4026	'BITADR' expects a bit address or a variable on a bit address	Use a valid bit address (e.g. %IX0.1).
4027	Integer number or symbolic constant expected	Enter a integer number or the identifier of a valid constant.
4028	'INI' operator needs function block instance or data unit type instance	Check the data type of the variable, for which the INI operator is used.
4029	Nested calls of the same function are not possible.	At not reentrant target systems and in simulation mode a function call may not contain a call of itself as a parameter. Example: fun1(a,fun1(b,c,d),e); Use a intermediate table.
4030	Expressions and constants are not allowed as operands of 'ADR'	Replace the constant or the expression by a variable or a direct address.
4031	'ADR' is not allowed on bits! Use 'BITADR' instead.	Use BITADR. Please note: The BITADR function does not return a physical memory address
4032	'<number>' operands are too few for '<name>'. At least '<number>' are needed	Check how many operands the named operator requires and add the missing operands.
4033	'<number>' operands are too many for '<name>'. At least '<number>' are needed	Check how many operands the named operator requires and remove the surplus operands.
4034	Division by 0	You are using a division by 0 in a constant expression. If you want to provocate a runtime error, use – if applicable - a variable with the value 0.
4035	ADR must not be applied on 'VAR CONSTANT' if 'replaced constants' is activated	An address access on constants for which the direct values are used, is not possible. If applicable, deactivate the option ‚Replace Constants' in ‚Project' ‚Options' ‚Build'.
4040	Label '<name>' is not defined	Define a label with the name <LabelName> or change the name <LabelName> to that of a defined label.
4041	Duplicate definition of label '<name>'	The label '<name>' is multiple defined in the POU. Rename the label or remove one of the definitions.
4042	No more than %d labels in sequence are allowed	The number of jump labels is limited to '<Anzahl>'. Insert a dummy instruction.
4043	Format of label invalid. A label must be a name optionally followed by a colon.	The label name is not valid or the colon is missing in the definition.
4050	POU '%s' is not defined	Define a POU with the name '<name>' using the command 'Project' 'Add Object' or change '<name>' to the name of a defined POU.
4051	'%s' is no function	Use instead of <name> a function name which is defined in the project or in the libraries.
4052	'%s' must be a declared instance of FB '%s'	Use an instance of data type '<name>' which is defined in the project or change the type of <Instance name> to '<name>' .

Number	Error Message	Possible Cause
4053	'%s' is no valid box or operator	Replace '<name>' by the name of a POU or an operator defined in the project.
4054	POU name expected as parameter of 'INDEXOF'	The given parameter is not a valid POU name.
4060	'VAR_IN_OUT' parameter '%s' of '%s' needs variable with write access as input.	To VAR_IN_OUT parameters variables with write access have to be handed over, because a VAR_IN_OUT can be modified within the POU.
4061	'VAR_IN_OUT' parameter '%s' of '%s' must be used.	A VAR_IN_OUT parameter must get handed over a variable with write access, because a VAR_IN_OUT can be modified within the POU.
4062	No external access to 'VAR_IN_OUT' parameter '%s' of '%s'.	VAR_IN_OUT Parameter only may be written or read within the POU, because they are handed over by reference.
4063	'VAR_IN_OUT' parameter '%s' of '%s' must not be used with bit addresses.	A bit address is not a valid physical address. Hand over a variable or a direct non-bit address.
4064	'VAR_IN_OUT' must not be overwritten in local action call!	Delete the parameters set for the VAR_IN_OUT variable in the local action call.
4070	The POU contains a too complex expression	Decrease nesting depth by dividing up the expression into several expressions. Use intermediate variables for this purpose.
4071	Network too complex	Divide up the network into several networks.
4072	Inconsistent use of an action identifier in FB type ('<name>') and instance ('<name>').	You have defined two actions of a function block fb: e.g. a1 and a2, but in the call of one of the actions in the FBD you are using a type (string within the box, e.g. fb.a1 different to that used in the instancename (e.g. inst.a2, above box). Correct the name correspondingly into the name of the desired action.
	'^' needs a pointer type	You are trying to dereference a variable which is not declared as a pointer.
4110	'[<index>]' needs array variable	[<index>] is used for a variable which is not declared as an array with ARRAY OF.
4111	Index expression of an array must be of type 'INT'	Use an expression of the correct type or a type conversion.
4112	Too many indexes for array	Check the number of indices (1, 2, oder 3), for which the array is declared and remove the surplus.
4113	Too few indexes for array	Check the number of indices (1, 2, oder 3), for which the array is declared and add the missing ones.
4114	One of the constant indizes is not within the array range	Make sure that the used indices are within the bounds of the array.
4120	'.' needs structure variable"	The identifier on the left hand of the dot must be a variable of type STRUCT or FUNCTION_BLOCK or the name of a FUNCTION or a PROGRAM.
4121	'<name>' is not a component of <object name>	The component '<name>' is not included in the definition of the object <object name>.
4122	'%s' is not an input variable of the called function block	Check the input variables of the called function block and change '<name>' to one of these.
	'LD' expected	Insert at least one LD instruction after the jump label in the IL editor.
4201	IL Operator expected	Each IL instruction must start with an operator or a jump label.

Number	Error Message	Possible Cause
4202	Unexpected end of text in brackets	Insert a closing bracket after the text.
4203	<name> in brackets not allowed	The operator <name> is not valid in a IL bracket expression. (not valid are: 'JMP', 'RET', 'CAL', 'LDN', 'LD', 'TIME')
4204	Closing bracket with no corresponding opening bracket	Insert an opening bracket or remove the closing one.
4205	No comma allowed after ')'	Remove comma after closing bracket.
4206	Label in brackets not allowed	Shift jump label so that it is outside of the brackets.
4207	'N' modifier requires operand of type 'BOOL', 'BYTE', 'WORD' or 'DWORD'	The N modifier requires a data type, for which a boolean negation can be executed.
4208	Conditional Operator requires type 'BOOL'	Make sure that the expression gives out a boolean result or use a type conversion.
4209	Function name not allowed here	Replace the function call by a variable or a constant.
4210	'CAL', 'CALC' and 'CALN' require a function block instance as operand	Declare an instance of the function block which you want to call.
4211	Comments are only allowed at the end of line in IL	Shift the comment to the end of the line or to an extra line.
4212	Accumulator is invalid before conditional statement	The accu is not defined. This happens if an instruction is preceeding which does not submit a result (e.g. 'CAL').
4213	'S' and 'R' require 'BOOL' operand	Use a boolean variable at this place.
4250	Another 'ST' statement or end of POU expected	The line does not start with a valid ST instruction.
4251	Too many parameters in function '%s'	There are more parameters given than are declared in the definition of the function.
4252	Too few parameters in function '%s'	There are less parameters given than are declared in the definition of the function.
4253	'IF' or 'ELSIF' require 'BOOL' expression as condition	Make sure that the condition for IF or ELSIF is a boolean expression.
4254	'WHILE' requires 'BOOL' expression as condition	Make sure that the condition following the ‘WHILE’ is a boolean expression.
4255	'UNTIL' requires 'BOOL' expression as condition	Make sure that the condition following the ‘UNTIL’ is a boolean expression.
4256	'NOT' requires 'BOOL' operand	Make sure that the condition following the ‘NOT’ is a boolean expression.
4257	Variable of 'FOR' statement must be of type 'INT'	Make sure that the counter variable is of an integer or bitstring data type (e.g. DINT, DWORD).
4258	Expression in 'FOR' statement is no variable with write access	Replace the counter variable by a variable with write access.
4259	Start value in 'FOR' statement is no variable with write access	The start value in the ‚FOR' instruction must be compatible to the type of the counter variable.
4260	End value of 'FOR' statement must be of type 'INT'	The end value in the ‚FOR' instruction must be compatible to the type of the counter variable.
4261	Increment value of 'FOR' statement must be of type 'INT'	The incremental value in the ‚FOR' instruction must be compatible to the type of the counter variable.
4262	'EXIT' outside a loop	Use 'EXIT' only within 'FOR', 'WHILE' or 'UNTIL' instructions.
4263	Expecting Number, 'ELSE' or 'END_CASE'	Within a ‘CASE' expression you only can use a number or a 'ELSE' instruction or the ending instruction 'END_CASE'.

Number	Error Message	Possible Cause
4264	'CASE' requires selector of an integer type	Make sure that the selector is of an integer or bitstring data type (e.g. DINT, DWORD).
4265	Number expected after ','	In the enumeration of the CASE selectors there must be inserted a further selector after a comma.
4266	At least one statement is required	Insert an instruction, at least a semicolon.
4267	Function block call requires function block instance	The identifier in the functionblock call is no instance. Declare an instance of the desired functionblock or use the name of an already defined instance.
4268	Expression expected	Insert an impression here.
4269	'END_CASE' expected after 'ELSE'-branch	Terminate the 'CASE' instruction after the 'ELSE' part with an 'END_CASE'.
4270	'CASE' constant '%ld' already used	A 'CASE' selector may only be used once within a ‘CASE' instruction.
4271	The lower border of the range is greater than the upper border.	Modify the area bounds for the selectors so that the lower border is not highte than the upper border.
4272	Exptecting parameter '%s' at place %d in call of '%s'!	You can edit a function call in that way, that also the parameter names are contained, not only the parameter values. But nevertheless the position (sequence) of the parameters must be the same as in the function definition.
4273	Parts of the 'CASE'-Range '%ld..%ld' already used in Range '%ld..%ld'	Make sure that the areas for the selectors which are used in the CASE instruction, don’t overlap.
4274	Multiple 'ELSE' branch in 'CASE' statement	A CASE instruction may not contain more than one ‚ELSE' instruction.
	Jump requires 'BOOL' as input type	Make sure that the input for the jump respectively the RETURN instruction is a boolean expression.
4301	POU '%s' need exactly %d inputs	The number of inputs does not correspond to the number of VAR_INPUT and VAR_IN_OUT variables which is given in the POU definition.
4302	POU '%s' need exactly %d outputs	The number of outputs does not correspond to the number of VAR_OUTPUT variables which is given in the POU definition.
4303	'%s' is no operator	Replace '<name>' by a valid operator.
4320	Non-boolean expression '<name>' used with contact	The switch signal for a contact must be a boolean expression.
4321	Non-boolean expression '<name>' used with coil	The output variable of a coil must be of type BOOL.
4330	Expression expected at input 'EN' of the box '<name>'	Assign an input or an expression to the input EN of POU '<name>’.
4331	Expression expected at input '<number>' of the box '<name>'	The input <number> of the operator POU is not assigned.
4332	Expression expected at input '<name>' of the box '<name>'	The input of the POU is of type VAR_IN_OUT and is not assigned.
4333	Identifier in jump expected	The given jump mark is not a valid identifier.
4334	Expression expected at the input of jump	Assign a boolean expression to the input of the jump. If this is TRUE, the jump will be executed.
4335	Expression expected at the input of the return	Assign a boolean expression to the input of the RETURN instruction. If this is TRUE, the jump will be executed.
4336	Expression expected at the input of the output	Assign a suitable expression to the output box.

Number	Error Message	Possible Cause
4337	Identifier for input expected	Insert a valid expression or identifier in the input box.
4338	Box '%s' has no inputs	To none of the inputs of the operator POU '<name>' a valid expression is assigned.
4339	Typemismatch at output: Cannot convert '<name>' to '<name>'.	The type of the expression in the output box is not compatible to that of the expression which should be assigned to it.
4340	Jump requires 'BOOL' as input type	Make sure that the input for the jump is a boolean expression.
4341	Return requires 'BOOL' as input type	Make sure that the input for the RETURN instruction is a boolean expression.
4342	Expression expected at input 'EN' of the box '<name>'	Assign a valid boolean expression to the EN input of the box.
4343	Values of Constants: ‘<name>’	Input '<name>' of box '<name>' is declared as VAR_INPUT CONSTANT. But to this POU box an expression has been assigned in the dialog 'Edit Parameters' which is not type compatible.
4344	'S' and 'R' require 'BOOL' operand	Insert a valid boolean expression after the Set resp. Reset instruction.
4345	Invalid Type for parameter '<name>' of '<name>': Cannot convert '<type>' to '<type>'.	An expression is assigned to input '<name>' of POU box '<name>' which is not type compatible.
4346	Not allowed to use a constant as an output"	You can only assign an output to a variable or a direct address with write access.
4347	'VAR_IN_OUT' parameter needs variable with write access as input.	To VAR_IN_OUT parameters only variables with write access can be handed over, because these can be modified within the POU.
4348	Invalid program name '<name>'. A variable with the same name exists already.	You have inserted a program box in the CFC editor, which has the same name as a (global) variable already existing in your project. You must rename accordingly.
4349	Input or output in POU <name> has been deleted: Check all connections to the box. This errormessage disappears only after the CFC was edited	Correct the POU appropriately.
4350	An SFC-Action can not be accessed from outside!	SFC actions only can be called within the SFC POU in which they are defined.
4351	Step name is no identifier: '<name>'	Rename the step or choose a valid identifier as step name.
4352	Extra characters following valid step name:'<name>'	Remove the not valid characters in the step name.
4353	Step name duplicated: '<name>'	Rename one of the steps.
4354	Jump to undefined Step: '<name>'	Choose an existent step name as aim of the jump resp. insert a step with name ‚<name>’.
4355	A transition must not have any side effects (Assignments, FB-Calls etc.)	A transition must be a boolean expression.
4356	Jump without valid Step Name: '<name>'	Use a valid identifier as aim (mark) of the jump.
4357	IEC-Library not found	Check whether the library iecsfc.lib is inserted in the library manager and whether the library paths defined in ‘Project’ ‘Options’ ‘Paths’are correct.
4358	Action not declared: '%s'.	Make sure that in the object organizer the action of the IEC step is inserted below the SFC POU and that in the editor the action name is inserted in the box on the right hand of the qualifier.

Number	Error Message	Possible Cause
4359	Invalid Qualifier: '%s'	In the box on the left hand of the action name enter a qualifier for the IEC action.
4360	Time Constant expected after qualifier '%s'	Enter next to the box on the left hand of the action name a time constant behind the qualifier.
4361	'%s' is not the name of an action	Enter next to the box on the right hand of the qualifier the name of an action or the name of a variable which is defined in the project.
4362	Nonboolean expression used in action: '%s'	Insert a boolean variable or a valid action name.
4363	IEC-Step name already used for variable: '<name>'	Please rename the step or the variable.
4364	A transition must be a boolean expression	The result of the transition expression must be of type BOOL.
4365	Time Constant expected after qualifier '<name>'	Open dialog ‚step attributes' for the step '<name>' and enter a valid time variable or time constant
4366	The label of the parallel branch is no valid identifier: '<name>'	Enter a valid identifier next to the triangle which marks the jump label.
4367	The label '<name>' is already used	There is already a jump label or a step with this name. Please rename correspondingly.
4368	Action '<name>' is used in multiple step chains, where one is containing the other!	The action '<name>' is used in the POU as well as in one or several actions of the POU.
4369	Exactly one network required for a transition	There are used several FBD resp. LD networks for a transition. Please reduce to 1 network.
4370	Additional lines found after correct IL-transition	Remove the not needed lines at the end of the transition.
4371	Invalid characters following valid expression: '<name>	Remove the not needed characters at the end of the transition.
4372	Step '<name>': Time limit needs type 'TIME'	Define the time limits of the step in the step attributes by using a variable of type TIME or by a time definition in correct format (e.g. "t#200ms").
4373	IEC-actions are only allowed with SFC-POUs	There is an action assigned to a non-SFC-POU (see in the Object Organizer), which is programmed in SFC and which contains IEC actions. Replace this action by one which contains no IEC actions.
4374	Step expected instead of transition '<name>'	The SFC POU is corrupt, possibly due to any export-import actions.
4375	Transition expected instead of step '<name>'	The SFC POU is corrupt, possibly due to any export-import actions.
4376	Step expected after transition '<name>'	The SFC POU is corrupt, possibly due to any export-import actions.
4377	Transition expected after step '<name>'	The SFC POU is corrupt, possibly due to any export-import actions.
	Unrecognized variable or address	The watch variable is not declared within the project. By pressing <F2> you get the input assistant which lists the declared variables.
4501	Extra characters following valid watch expression	Remove the surplus signs.
4520	Error in Pragma: Flag expected before '<name>'!	The pragma is not correct. Check whether ‘<name>' is a valid flag.
4521	Error in Pragma: Unexpected element '<name>'!	Check whether pragma is composed correctly.

Number	Error Message	Possible Cause
4522	"flag off' pragma expected!	The switch off of the pragma is missing, add a 'flag off' instruction.
4550	Index out of defined range : Variable OD "number>, Line <line number>.	Ensure that the index is within the area which is defined in the target settings/ networkfunctionality.
4551	Subindex out of defined range : Variable OD "number>, Line <line number>.	Ensure that the subindex is within the area which is defined in the target settings / networkfunctionality.
4552	Index out of defined range : Parameter OD "number>, Line <line number>.	Ensure that the index is within the area which is defined in the target settings / networkfunctionality
4553	Subindex out of defined range : Parameter OD "number>, Line <line number>.	Ensure that the subindex is within the area which is defined in the target settings / networkfunctionality.
4554	Variablename invalid: Variable OD <number>, Line <line number>.	Enter a valid project variable in the filed ‚variable'. Use the syntax <POU name>.<variable name> resp. for global variables .<variable name>
4555	Empty table-entry, input not optional: Parameter OD <number>, Line <line number>	You must make an entry in this field.
4556	Empty table-entry, input not optional: Variable OD <number>, Line <number>	You must make an entry in this field.
4557	The required parameter memory is too large	The maximum size of data which can be loaded via parameter lists of type Parameters to the controller has been exceeded. This size is defined by the target system. Information on the data size is displayed in the message window at compilation. Reduce the parameter lists size.
4558	The required variable memory is too large	The maximum size of data which can be loaded via parameter lists of type Variables to the controller has been exceeded. This size is defined by the target system. Information on the data size is displayed in the message window at compilation. Reduce the parameter lists size.
4560	Invalid value: Dictionary '<Name>', column '<Name>', line '<line number>'	Check this entry. It depends on the currently used column (attribute) definition which entries are valid for this field. This definition is given by the target-specific XML description file of the Parameter Manager resp. by the standard settings which will be used if there is no description file.
4561	Column not defined: '<Name>'	Entries in a column of the parameter list refer to another column, which is not defined however. The column definitions are given by the description file (XML) of the Parameter Manager for the current target. If a description file is not available, standard settings are used.
4562	Index/subindex used already: Dictionary '<Name>', line '<Line Number>'	The Index/Subindex-combination must be unique throughout all parameter lists, because it can be used for the parameter access. Correct the indices correspondingly.
4563	Identifier '<Name>' used already: Dictionary '<Name>', line '<Line Number>'	The name must be unique throughout all parameter lists, because it can be used for parameter access.

Number	Error Message	Possible Cause
4564	Index '<Name>' is out of range: Dictionary '<Name>', line '<Line Number>'	Enter an index which is within the range defined in the target settings, category network functionality in field 'Index range...' for the respective list types (Variables, Parameters, Mappings).
4565	Subindex '<Name>' is out of range: Dictionary '<Name>', line '<Line Number>'	Enter an subindex which is within the range defined in the target settings, category network functionality in field 'SubIndex range'.
4566	An error occurred during import of the parameter manager	You have imported an export file which contains erroneous information on the Parameter Manager. Check the *.exp-file.
	Networkvariables: '<name>' expression is not from type bool!	Make sure that the variable defined in the properties dialog of the network variables list at option 'Transmit on event', is of type BOOL.
4601	Network variables '<name>': No cyclic or freewheeling task for network variable exchange found	There is no cyclic or free-wheeling task resp. PLC_PRG in the project where the network variables of type CAN or UDP of the given list are used (only declaration is not sufficient!). You must take care that the variables are used in an appropriate task or in PLC_PRG. If you want to use them in several tasks, regard, that at data exchange the task with the highest priority will be regarded.
4602	'<name of network variables list>': The object uses UDP port '<port number>' instead of '<port number>'	In the Settings of the named network variables list a port number is used which is not the same as that which is used in the first network variables list found in the global variables folder. Take care that all network variables lists are using the same port!
4620	Unused variables have been found in the project.	Unused variables have been found in the project.
4621	There are overlaps at the assignment of variables to memory areas via the „AT"-declaration.	There are overlaps at the assignment of variables to memory areas via the „AT"-declaration.
4622	IEC addresses assigned to the same memory area are referenced in more than one task.	IEC addresses assigned to the same memory area are referenced in more than one task.
4623	The project gains write access to the same memory area at more than one place.	The project gains write access to the same memory area at more than one place.
4650	AxisGroup '<Name>': Task '<Name>' does not exist.	In the PLC Configuration in the definition of the axis group (dialog 'Module parameters', column 'Value') there is a name defined for the task which is controlling the data transfer of this axis group, which is not known in the Task Configuration. Correct Task Configuration resp. PLC Configuration correspondingly.
4651	AxisGroup '<Name>': Cycletime (dwCycle) not set.	In dialog 'Module parameters' of the axis group enter a value for the cycle time (dwCycle).
4670	CNC program '<Name>': Global variable '<Name>' not found.	In the CNC program a global variable is used (e.g. $glob_var$), which is not defined in the project. Add the appropriate declaration resp. correct the assignment to the variable in the CNC program.
4671	CNC program '<Name>': Variable '<Name>' has an incompatible type.	There is a variable assigned in a instruction of the CNC program , which is declared of a data type which is not valid in this place. Use another variable resp. correct the type specification.

Number	Error Message	Possible Cause
4685	CAM '<Name>': CAM table type unknown.	Check the data type which is specified in the CAM Editor dialog "Compile options.." for the equidistant resp. element optimized point table.
4686	CAM '<Name>': CAM point exceeds datatype range.	In this CAM points are used, which are out of the data range specified for the point table. For the current range definition see dialog 'Compile options..' in the CAM-Editor.
	'<Number>' ('<Name>'): Watch expression '<Name>' is not a numeric variable.	In the configuration of the visualization a variable is used which is not a number, as required in this place (e.g. at the configuration of XOffset or Angle values etc.).
4701	'<Name>' ('<Number>'): Watch expression '<name>' is not of type BOOL.	In the configuration of the visualization a variable is used which is not of type BOOL, as required in this place.
4702	'<Name>' ('<Number>'): Watch expression '<name>' is not of type STRING.	The visualization contains a variable which is not of type STRING although this is required in this place (e.g. at the tooltip configuration).
4703	'<Name>' ('<Number>'): Invalid watch expression '<Name>'	The visualization contains an invalid variable.
4704	'<Name>'('<Number>'): Invalid initial value in watchlist '<Name>'.	In this watchlist, used in a visualization (INTERN command in category Input), there is a erroneous initial value. Check the used list.
4705	'<name>' ('<number>'): No valid alarm group assigned to alarm table.	Enter a valid alarm group In the configuration dialog of the alarm table (category Alarm table).
	Invalid type for conversion	You are using a type conversion which is not supported by the currently chosen codegenerator.
4901	Internal error: Overflow in array access!	The array bounds are to large for a 32-bit-variable. Reduce the array index range.

7.6 Programming conventions for creating the IEC61131-3

The following programming conventions are used for naming objects, variables and instances of objects and variables in a standardised fashion. This will make the code and particularly the interfaces of function blocks and functions (e.g. in the PLC libraries) easily legible and comprehensible for us and for others. Code maintenance becomes easier.

It goes without saying that this document only refers to the creation of new PLC libraries. For compatibility reasons, existing libraries are, of course, not changed!!! For TwinCAT users and PLC programmers, this document should be seen as a guideline since these programming conventions are not specified in the IEC61131-3 standard.

Naming of the PLC libraries

All libraries that are supplied with TwinCAT or can be installed from the Supplement CD start with the prefix **Tc**.

Example: **TcUtilities.Lib**

The libraries for the PC runtime system have the extension *.**Lib**, the Bus Controller libraries(BCs)have the extension *.**Lb6** and additionally the letters **BC** at the end of the library name.

Example:

TcTempCtrl.Lib

TcTempCtrlBC.Lb6

Identifier

The identifier should be named with a uniform prefix, so that the object type can easily be recognised. The recommended conventions are listed below:

Definition of object names

Object	Prefix	Description	Example
FUNCTION_BLOCK	FB_	Function Block	FB_GetData
STRUCT	ST_	Structure	ST_BufferEntry
ENUM	E_	Type of Enumeration	E_SignalStates
TYPE	T_	Reference	T_Nibble
PROGRAM	P_	Program	P_Axis
FUNCTION	F_	Function	F_Convert

Generating instances of objects

Object	Prefix	Description	Instance	Call example
function block	fb	Instance name from function block	fbGetData	fbGetData();
struct	st	Instance name of the structure	stBufferEntry	stBufferEntry.nCounter := 5;
enum	e	Enum instance	eSignalState	eSignalState := E_STOP;
alias type	none	Instance name of a reference type	Nibble	Nibble := 1;

Generating instances of variables

Type	Prefix	Description	Example
SINT, USINT, …, DINT, UDINT, BYTE, WORD, DWORD, LWORD....	n, i	numeric / integer	nCount, iError
BOOL	b	bit	bSwitch
REAL, LREAL	f	float	fValue
STRING	s	string	sName
TIME	t	time	tDelay
DATE	d	date	dMonday
DATE_AND_TIME	dt	date and time	dtNewYear
ARRAY[...] OF ...	arr	arrays	arrMessages
p	p	pointer	pData

Other prefixes that may also be used

Prefix	Description	Example
cb	count of bytes	cbLength
cw	count of words	cwRead

Upper / lower case

The object name prefix will always be upper case. Underscore (_) is used as a separator between prefix and object name. For variable and instance names, the prefix will always be lower case. The first letter of an identifier will always be upper case. If an identifier consists of several words, the first letter of a word will always be upper case. Separators (e.g. _) between the two words should not be used (see examples above).

Valid characters

The identifiers should only contain the following letters, digits and special characters:

0...9, A...Z, a...z,

Remarks

All identifiers should be specified in English.

Version check

Each library must contain a function in which the following information is stored:

- Version number
- TwinCAT version under which the library was generated
- Date (American format)
- Programmer
- Comments

The version number must be readable during program run-time. The version number consists of three numeric values. Each value can be read individually (see example). In order to make the function name unambiguous, it will contain the name of the library. Example from the library TcBABasic.lib:

```
FUNCTION F_GetVersionTcBABasic : UINT
VAR_INPUT
      nVersionElement       :       INT;
END_VAR
VAR
END_VAR
(* Version history:
Date      | Version    | created under      | Author      | Remark
-----------------------------------------------------------------------------------------
10/01/2001    | 1.0.0      | V2.7.0 (Build 402) | S. Mustermann   | first release
*)

CASE nVersionElement OF
1: (* major number *)
      F_GetVersionTcBABasic := 1;
2: (* minor number *)
      F_GetVersionTcBABasic := 0;
3: (* revision number *)
      F_GetVersionTcBABasic := 0;
ELSE
   F_GetVersionTcBABasic := 16#FFFF;
END_CASE
```

8 Library Management

The library manager shows all libraries that relate to the present project. The POUs, data types, and global variables of the libraries can be used the same way as user-defined POUs, data types, and global variables.

The library manager is opened with the **"Window" "Library Manager"** command. Information concerning included libraries is stored with the project.

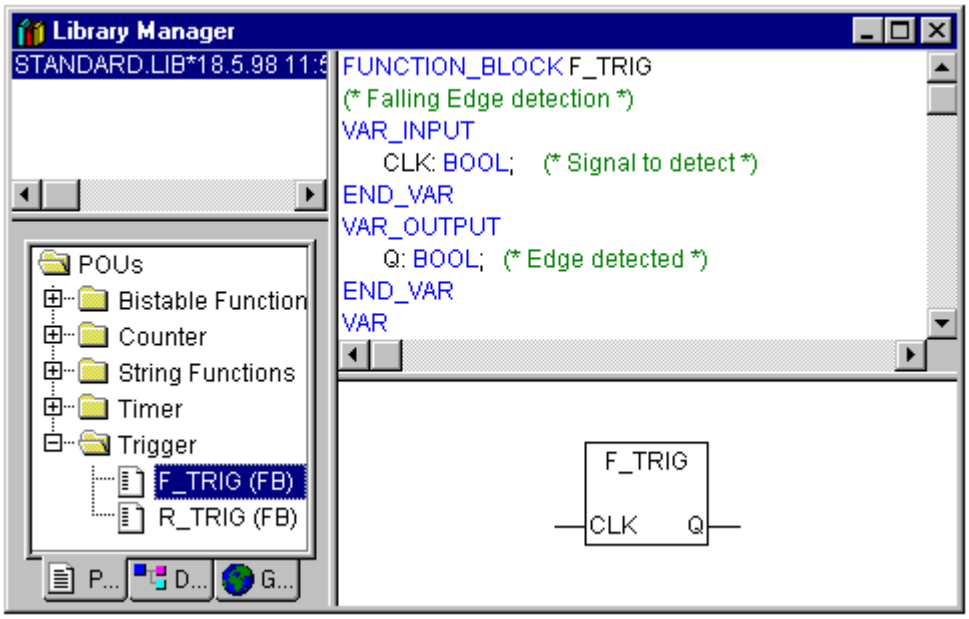


Fig. 2: Library Manager

Using the Library Manager

The window of the library manager is divided into three or four areas by screen dividers. The libraries attached to the project are listed in the upper left area. In the area below that, depending on which register card has been selected, there is a listing of the POUs , Data types, or Global variables of the library selected in the upper area. Folders are opened and closed by double clicking the line or pressing <Enter>. There is a plus sign in front of closed folders, and a minus sign in front of opened folders. If a POU is selected by clicking the mouse or selecting with the arrow keys, then the declaration of the POU will appear in the upper right area of the library manager; and in the lower right is the graphic display in the form of a black box with inputs and outputs. With data types and global variables, the declaration is displayed in the right area of the library manager.

Standard Library

The library 'standard.lib' is always available. It contains all functions and function blocks that are required by the IEC61131-3 as standard POUs for an IEC programming system. The difference between a standard function and an operator is that the operator is implicitly recognized by the programming system, while the standard POUs must be tied to the project (standard.lib).

User-defined Libraries

If a project is to be compiled in its entity and without errors, then it can be saved in a library with the "Save as" command in the "File" menu. The project itself will remain unchanged. Subsequently, you can gain access to the project under the entered name, just as with the standard library.

'Insert' 'Additional Library'

With this command you can attach an additional library to your project. In the dialog box for opening a file, choose the desired library with the "*.lib" extension. The library is now listed in the library manager, and you can use the objects in the library as user-defined objects.

Remove Library

With the "Edit" "Delete" command you can remove a library from a project and from the library manager.

Features

This command will open the dialog 'Information about internal (resp. external) library'. For internal libraries you will find there all data, which have been inserted in the Project Info when the library had been created. For external libraries the library name and library path will be displayed.

9 Engineering Interface (ENI)

The ENI interface ('Engineering Interface') allows to connect the Programming system to an external data base. There the data which are needed during creation of an automation project can be stored. The usage of an external data base guarantees the consistency of the data, which then can be shared by several users, projects and programs. Also it extends the functionality by making possible the following items:

- **Revision control** for projects and associated ressources (shared objects): If a object has been checked out from the data base, modified and checked in again, then in the data base a new version of the object will be created, but the older versions will be kept also and can be called again on demand. For each object and for a whole project the version history will be logged. Two versions can be checked for differences. (This does not apply to the using of a local data system as data base.)
- **Multi-User Operation:** The latest version of a sample of objects, e.g. of POUs of a project, can be made accessible for a group of users. Then the objects which are currently checked out by one of the users will be marked as "in the works" and not editable by the other users. Thus several users can work in parallel on the same project without risking to overwrite versions mutually.
- **Access by external tools:** Besides the TwinCAT PLC Control also other tools, which have an ENI too, can access the common data base. These might be e.g. external visualizations, ECAD systems etc., which need the data created in TwinCAT PLC Control or which also produce data.

The ENI is composed of a client and a server part. So it is possible to hold the data base on a remote computer, which is required for multi-user operation. The TwinCAT PLC Control is a client of the independent ENI Server Process as well as another application, which needs access to the data base (Please see the separate documentation on the ENI Server).

Currently the ENI supports the data base systems **'Visual SourceSafe 5.0' and 'Visual SourceSafe 6.0'** and a local file system. Objects can be stored there in different **'folders'** (data base categories with different access properties). The objects can be checked out for editing and thereby will get locked for other users. The latest version of an object can be called from the data base. Furtheron in parallel you can store any objects just locally in the project as usual for projects without source control. The *.pro file is the local working copy of an project managed by the data base.

Preconditions for Working with an ENI project data base

If you want to use the ENI in TwinCAT PLC Control to manage the project objects in an external data base, the below mentioned preconditions must be fulfilled:



For a guide concerning installation and usage of the ENI Server please see the separate server documentation TS1600 | TwinCAT ENI Server. There you will also find a quick start guide. Also consider the possibility of using the ENI Explorer which allows to perform data base actions independently from the currently used data base system.

- the communication between TwinCAT PLC Control and the ENI Server requires TCP/IP because the ENI Server uses the HTTP protocol.
- an ENI Server (ENI Server Suite)) must be installed and started locally or on a remote computer. A license is required to run it with the standard database drivers which are installed with the server. Just the driver for a local file system can be used with a non-licensed ENI Server version.
- in the ENI Server administration tool (ENI Admin) the following must be configured:
  - the user must be registered and have access rights (User Management)
  - the access rights concerning the data base folders must be set correctly (Access Rights)
  - **Recommendation**: the administrator password for the access to ENIAdmin.exe and ENIControl.exe should be defined immediately after the installation
- in the ENI Server service control tool (ENI Control) the connection to the desired data base must be configured correctly (Data base). You will automatically be asked to do this during installation, but you can modify the settings later in ENI Control.
- a project data base for which an ENI-supported driver is available, must be installed. It is reasonable to do this on the same computer, where the ENI Server is running. Alternatively, a local file system can be used, for which a driver will be provided by default.

- in the data base administration possibly the user (Client) as well as the ENI Server must be registered as valid users with access rights. Anyway, this is required for the 'Visual SourceSafe', if you use another data base system, please see the corresponding documentation for information on the user configuration.
- for the current project the ENI interface must be activated (to be done in the TwinCAT PLC Control dialog **'Project' 'Options' 'Project data base'**).
- for the current project the connection to the data base must be configured correctly; this is to be done in the TwinCAT PLC Control dialogs **'Project' 'Options' 'Project data base'.**
- in the current project the user must log in to the ENI Server with username and password; this is to be done in the Login dialog, which can be opened explicitly by the command **'Project' 'Data Base Link' 'Login'** resp. which will be opened automatically in case you try to access the data base without having logged in before.

Working with the ENI project data base

The **data base commands** (Get Latest Version,Check Out, Check In, Version History, Label Version etc.) which are used for managing the project objects in the ENI project data base, will be available in the current project as soon as the connection to the data base has been activated and configured correctly. See for this Preconditions for Working with an ENI project data base. The commands then are disposable in the submenu 'Data Base Link' of the context menu or of the 'Project' menu and refer to the object which is currently marked in the Object Organizer. The current assignment of an object to a data base category is shown in the Object Properties and can be modified there. The properties of the data base categories (communication parameters, access rights, check in/check out behaviour) can be modified in the option dialogs of the project data base ('Project' 'Options' 'Project Source Control').

Object categories concerning the project data base

There are four categories of objects of a TwinCAT PLC Control project concerning the project source control:

- The ENI distinguishes three categories ("ENI object categories") of objects which are managed in the project data base: Project objects, Shared objects, Compile files.
- If an object should not be stored in the data base, it will be assigned to category 'Local', which means that it will be handled as it is known for projects without any source control.

Thus in the programming system an object can be assigned to one of the categories 'Project objects', 'Shared objects' or 'Local'; the 'Compile files' do not yet exist as objects within the project.
Assigning an object to one of the categories is done automatically when the object is created, because this is defined in the project options dialog 'Project source control', but it can be modified anytime in the 'Object Properties' dialog.
Each ENI object category will be configured separately in the settings for the **'ENI settings' ('Project' 'Options', category project data base)**. That means that each category gets defined own parameters for the communication with the data base (directory, port, access right, user access data etc.) and concerning the behaviour at calling the latest version, checking out and checking in. These settings then will be valid for all objects belonging to the category. As a result you will have to log in (username, password) to each data base category separately; this will be done via the 'Login' dialog (**'Project' 'data base agssignment' 'Login**').

It is advisable to create a separate folder for each object category in the data base, but it is also possible to store all objects in the same folder. (The 'category' is a property of an object, not of a folder.)

See in the following the three ENI object categories:

Requirements

	Description
Project	For objects representing project-specific source information, e.g. shared function blocks within a project, important for multi-user operation. The 'Get All' command automatically fetches all objects of this category from the project directory of the database into the local project, even those that were not yet created there.
Shared objects	For general-purpose, project-independent objects, such as block libraries, which are normally used by several users in different projects. ***Notice*** **The 'Get All' command copies only those objects of this category from the project directory of the database to the local project that have already been created there.**
Compilation files	For the project-specific compile information (e.g. symbol files) automatically generated by TwinCAT PLC Control, which is also required by other tools. For example, a visualization requires the variables of a programming system including the addresses, which, however, are only assigned during compilation.



Optionally, project function blocks can also be excluded from management in the project database and saved exclusively locally, i.e. only with the project, as is traditionally the case.

10 Visualization

For visualizing, that means watching and operating the data of a controller which has is programmed with TwinCAT, no additional tool is required. The programming system contains an integrated visualization editor, allowing the user to create visualization objects to the parallel to the development of the application in the same user interface.

The integration offers a lot of benefits:

The visualization integrated in TwinCAT PLC Control does not need a tag list and it can directly access the variables in the controller. Any OPC- or DDE-layer, which often is difficult to configure, is also not necessary, because the communication is done by the same mechanism which is used for the programming system. Thus the engineering effort for the realization of visualizations will be clearly reduced. Besides that the connection of controller and visualization allows running modes, in which the conventional display on a PC with a monitor is not needed any longer.

Modes:

1. Directly in the programming system

Directly in the programming system [▶ 258]

For testing the created visualization masks, but also for service or diagnosis purposes in direct connection with the controller you do not need a further tool: In online mode you immediately get a display of the visualizations within the programming system.

2. Target-Visualization

Target-Visualization [▶ 320]

For controllers with integrated display the visualization data can be loaded from the programming system together with the application to the target system. There they get displayed automatically. This solution can be portated with low effort to any devices which are programmable with TwinCAT PLC Control.

Overview on the functions:

The different visualization variants of the TwinCAT system have differences in their functional scope. These differences are shown in the following table [▶ 321].

• **Elements**
- Rectangle, Ellipse, Rounded rectangle
- Line, Polygon, Polyline, Curve
- Bitmap, WMF-file
- Button, Table, Histogram, Bar Display, Meter
- Reference to another visualization

• **Animations (depending on element type)**
- Text display
- Color changes
- Visible/Invisible
- Shift
- Rotation
- Scaling
- Offset on the particular edges of an object (for Bar Display)
- Button activ/inactiv
- Current line (only text display)

• **Input possibilities**
- Toggle/tap Boolean value
- Text input
- Change of visualization
- Special actions (Leave visualization, Read/Write receipts, Switch language, call external EXE etc.)
- Choose line (only text display)

• **Further properties**
- Switching language
- Tooltips for all elements
- ASCII Import/Export
- Background Bitmap
- Automatic Scaling
- Drawing operations: Alignment, Order, Grouping
- Placeholder concept for creating objects with complex graphic elements
- Programmed visualization expressions

10.1 Visualization Editor

A visualization is a **graphical representation** of the project variables which allows inputs to the PLC program in online mode via mouse and keypad. The TwinCAT PLC Control **visualization editor**, which is part of the programming system provides graphic elements which can be arranged as desired and can relate to project variables.

Thereupon in **online mode** the look of the graphical elements will change depending on the variable's values.

Simple example

To represent a fill level, which is calculated by the PLC program, draw a bar, and connect it to the corresponding project variable, so that the length and color of the bar will show the current fill level value. Add a text field which will display the current value in a text string and a button for starting and stopping the program.

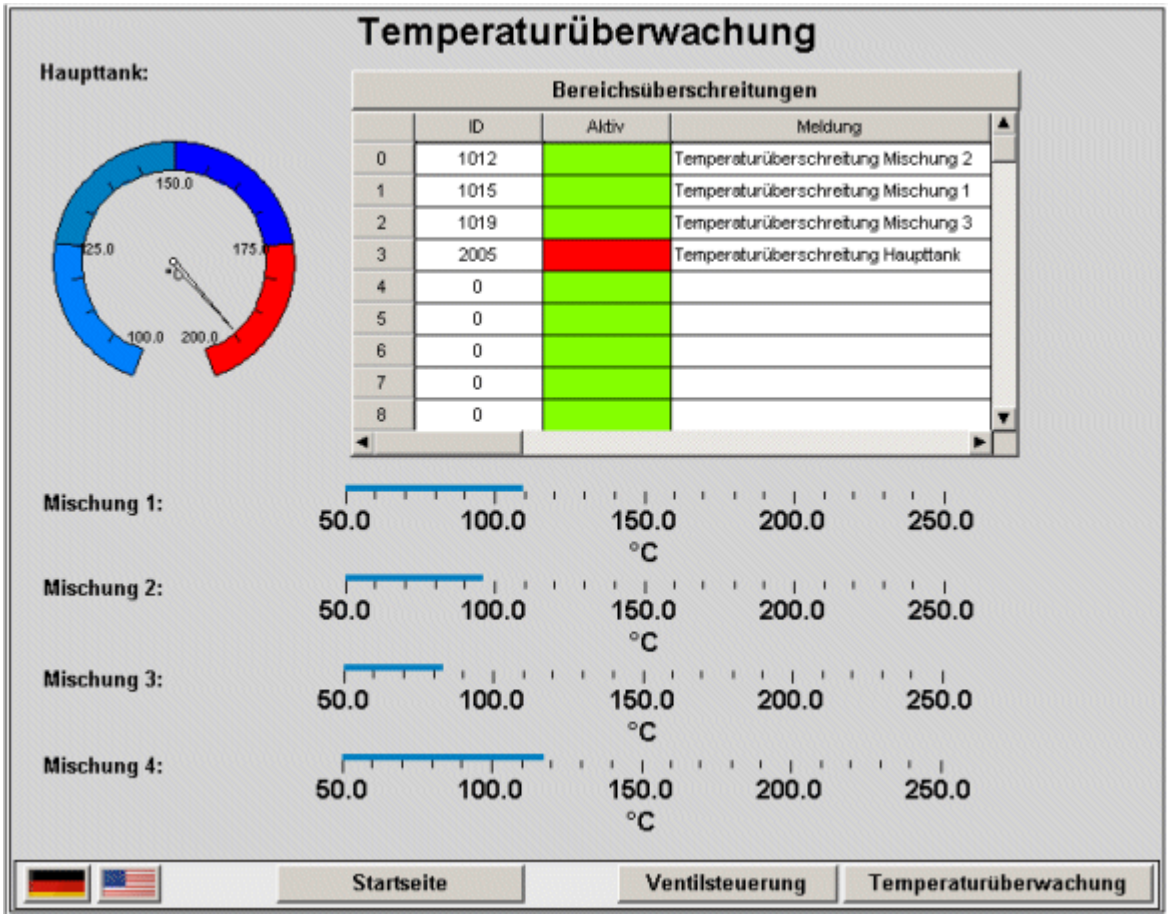


Fig. 3: Example of visualization

The properties of a single visualization element as well as of the whole visualization object will be defined in appropriate **configuration** dialogs. There it is possible to set basic parameters by activating options as well as to define a dynamic parameterizing by entering project variables.

Additional special possibilities for configuring are given by the programmability of element properties via structure variables.

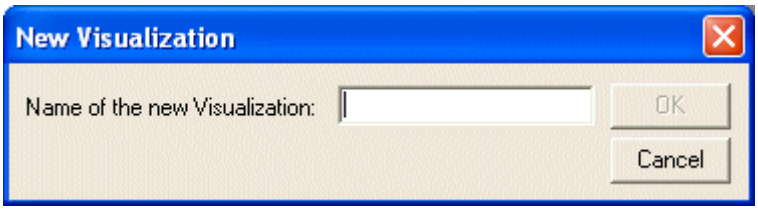
Using placeholders in the configuration dialogs may save a lot of effort in case you want to use the same visualization object several times with different configurations.

The visualization which is created in the programming system will in many cases be used as the only user interface available for controlling and watching the associated PLC program in online mode. For this purpose, it must be possible to give inputs to the program solely by activating visualization elements. To reach this you can use special input possibilities during the configuration of the visualization, and you have the option to define special hotkeys for each visualization.

10.1.1 Create a new Visualization

A visualization object is a TwinCAT PLC Control object which is managed in the 'Visualization' register of the Object Organizer. It contains an arrangement of visualization elements and can get certain object properties. One or several visualization objects can be created in a TwinCAT PLC Control project and might be linked with each other.
In order to create a visualization object in the Object Organizer, you must select the register card for

 **Visualization** in the Object Organizer.
Using the **'Project' 'Object Add'** command, you can create a new visualization object.



Open the 'New visualization' dialog, in which you can enter the name of the new visualization. Once a valid entry is made, that is not a name that is already in use and no special characters used, you can close the dialog with **OK**. A window opens, in which you can edit the new visualization.

NOTICE

Name of the vizualisation

A vizualisation should not have the same name as another block in the project. For vizualisation changes this can lead to problems.

10.1.2 Insert Visualization element

A visualization element is a graphical element, which is used to fill a visualization object. The available elements are offered in the TwinCAT PLC Control menu bar. Each element gets a separate configuration.

You can insert various geometric forms, as well as bitmaps, metafiles, buttons and existing visualizations, into your visualization.
Geometric forms at your disposal include: rectangles, rounded rectangles, ellipses/circles, and polygons.

Go to the **'Insert'** menu item and select freely from the following commands: **'Rectangle'**, **'RoundedRectangle'**, **'Ellipse'**, **'Polygon'**, **'Polyline'**, **'Curve'**, **'Pie'**, **'Bitmap'**, **'Visualization'**, **'Button'**, **'Table'**, **'Meter'**, **'BarDisplay'**, **'Histogram'**, **'Alarmtable'**, **'Trend'**, **'WMFfile'**.

A check appears in front of the selected command. You can also use the tool bar. The selected element appears pushed down(e.g. ).
If you now go to the editor window with the mouse, you will see that the mouse pointer is identified with the corresponding symbol (e.g. ).

Click on the desired starting point of your element and move the pointer with pressed left mouse key until the element has the desired dimensions.

If you want to create a polygon or a line, first click with the mouse on the position of the first corner of the polygon resp. on the starting point of the line, and then click on the further desired corner points. By doubleclicking on the last corner point you will close the polygon and it will be completely drawn respectively the line will be completed. If you want to create a curve (Bezier curves) determine the initial and two other points with mouse clicks to define the circumscribing rectangle. An arc is drawn after the third mouse click. You can then change the position of the end point of the arc by moving the mouse and can then end the process with a double click or add another arc with additional mouse clicks.



Furthermore pay attention, to the status bar and the change from select and insert modes.

'Insert' 'Rectangle'

Symbol: 

With the command you can insert a rectangle as an element into your present visualization.

'Insert' 'Rounded Rectangle'

Symbol: 

With the command you can insert a rectangle with rounded corners as an element in your present visualization.

'Insert' 'Ellipse'

Symbol: 

With the command you can insert a circle or an ellipse as an element in your present visualization.

Insert' 'Polygon'

Symbol: 

With the command you can insert a polygon as an element in your present visualization.

Insert' '´Polyline'

Symbol: 

With the command you can insert a polyline as an element in your present visualization.

Insert' 'Curve'

Symbol: 

With the command you can insert a Bezier curve as an element into your current visualization.

Insert' 'Pie'

Symbol: 

Use this command to insert a Pie Segment as an element into your current visualization.
While pressing the left mouse button, bring up an area in the desired size. An oval element including a line marking the radius at the 0° position will be displayed. As long as keeping the mouse button pressed you can immediately change size and position of the element by moving the mouse. A little black square is attended to the the element, indicating the corner of a virtual rectangle surronding the element.
In order to define the start and end angles of a Pie, select the end point of the radius line on the circular arc by a mouse-click. As soon as you, keeping the mouse button pressed, move the cursor, two small rectangles will be displayed, indicating the two angle positions. As from now those can be selected and moved seperately. If you want the angle values get defined dynamically by variables, open the configuration dialog category 'Angle' and enter the desired variable names.
You can resize or reshape the element later by either clicking on the centre point, the cursor getting displayed as diagonally crossed arrows, and moving the mouse while keeping the mouse button pressed (or using the arrow keys). Alternatively you can select and move the corner indicating little square outside of the element. In order to move the element to another position, click inside the element to get the cursor beeing displayed as vertically crossed arrows and then move the cursor.

Insert' 'Bitmap'

Symbol: 

With the command you can insert a bitmap as an element in your present visualization.
While pressing the left mouse button, bring up an area in the desired size. The dialog box is opened for opening a file. Once you have selected the desired bitmap, it will be inserted into the area brought up. You can define in the bitmap configuration dialog, whether a just link to the bitmap file should be stored or the bitmap should be inserted as an element..

Insert' 'Visualization'

Symbol: 

With the command you can insert an existing visualization as an element in your present visualization.
While pressing the left mouse button, bring up an area in the desired size. A selection list of existing visualizations opens. After you have selected the desired visualization, it will be inserted in the defined area.
An inserted visualization will also be named as a reference.

Insert' 'Button'

Symbol: 

This command is used to insert a button into your current visualization.

Drag the element to the desired size with the left mouse button held down.
If a toggle variable is configured for the button it displays the state of this variable by visually displaying whether it is pressed or not pressed. Conversely, the variable is toggled by „pressing" the button.

Insert' 'WMF-File'

Symbol: 

This command is used to insert a Windows Metafile. The standard dialog for opening a file will appear, where you can select a file (extension *.wmf). After having closed the dialog with OK the file will be inserted as an element in the visualization. Please regard, that no link to a file will be saved, like it is done when you insert a bitmap, but the elements of the metafile will be inserted as a group.

Insert' 'Table'

Symbol: 

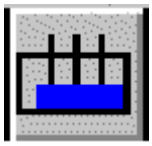
Use this command to insert a Table element as an element into your current visualization. It is used to display the current values of the elements of an array.
While pressing the left mouse button, bring up an area in the desired size. Before the element gets displayed the configuration dialog 'Configure table'' will be opened. Here you will find additionally to the standard categories Tooltip and Security the categories 'Table', 'Columns', 'Rows' and 'Selection' where you can define contents and appearance of the table.

Insert' 'Meter'

Symbol: 

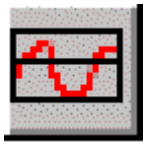
Use this command to insert a Meter as an element into your current visualization. It provides a scale which is defined as a sector of a circular arc, and a pointer element.
While pressing the left mouse button, bring up an area in the desired size. Before the element gets displayed the configuration dialog Configure Meter will be opened. Here you can define various parameters concerning the display of the element and a preview is available to check the configuration before really inserting the element by confirming the dialog.

Insert' 'Bar Display'

Symbol: 

Use this command to insert a Bar Display element into your current visualization. It is used to visualize the value of the assigned variable by a bar indicating the value by its lenght along a horizontal scale.
While pressing the left mouse button, bring up an area in the desired size. Before the element gets displayed the configuration dialog 'Configure bar display' will be opened. Here you can define various parameters concerning the display of the element and a preview is available to check the configuration before really inserting the element by confirming the dialog.

Insert' 'Histogram'

Symbol: 

Use this command to insert a Histogram element into your current visualization. It is used to visualize the elements of an array by bars which are placed side by side each indicating the value of the element by its length.
While pressing the left mouse button, bring up an area in the desired size. Before the element gets displayed the configuration dialog 'Configure Histogram' will be opened. Here you can define various parameters concerning the display of the element and a preview is available to check the configuration before really inserting the element by confirming the dialog.

Insert' 'Alarm table'

Symbol: 

Use this command to insert an alarm table into your current visualization object.
While pressing the left mouse button, bring up an area in the desired size. Before the element gets displayed the configuration dialog 'Configure Alarm table' will open. Here you will find additionally to the standard categories Tooltip and Security the categories 'Alarmtable', 'Settings for sorting', 'Columns' and Settings for

alarmtable' where you can define contents and appearance of the table.
An alarm table can be used to visualize the alarms, which have been defined in the Alarm configuration of the project.

Insert' 'Trend'

Symbol: 

Use this command to insert a trend element into your current visualization object.. While pressing the left mouse button, bring up an area in the desired size. The configuration (axes, variables, history) is done in the configuration dialog 'Trend'.
The trend element, also named oszilloscope element, is used to display variable values within a certain time period. It stores the data in a file on the client and displays them as a graph. As soon as a value changes, a new entry will be made in the file, showing date/time and the new value.
The trend element is drawn transparently. So you can assign any desired background (bitmap, color).

10.1.3 Positioning of Visualization Elements

Selecting Visualization Elements

The selection mode is activated by default. In order to select an element, click with the mouse on the element. You can also select the first element of the elements list by pressing the <Tab> key and jump to the next by each further keystroke. If you press the <Tab> key while pressing the <Shift> key, you jump backwards in the order of the elements list.
In order to select elements, which are placed one upon the other, first select the top level element by a mouse-click. Then do further mouse-clicks while the <Ctrl> button is pressed, to reach the elements in the underlying levels .
In order to mark multiple elements, press and hold the <Shift> key and click the corresponding elements, one after another; or, while holding down the left mouse button, pull a window over the elements to be selected.
In order to select all the elements, use the 'Extras' 'Select All' command.
If you are in the element list (called by '**Extras**' '**Element list**'), you can select the concerned element in the visualization by selecting a line.

Changing the Selection and Insert Mode

Changing the Selection and Insert Mode
After the insertion of a visualization element, there is an automatic change back into the selection mode. If you want to insert an additional element the same way, you can once again select the corresponding

command in the menu or the symbol  in the tool bar.
You can also quickly change between the selection mode and the insert mode by pressing the <Ctrl>-key and the right mouse button simultaneously.
In the insert mode, the corresponding symbol will also appear at the mouse pointer, and the name will also be indicated in black in the status bar.

'Extras' 'Select'

This command is used to switch the selection mode on or off. This can also be achieved using the symbol

 or by pressing the right mouse-key while holding down the key at the same time.

Copying Visual Elements

One or more selected elements can be inserted with the '**Edit**' '**Copy**' command, the <Ctrl>+<C> key combination, or the corresponding copy symbol, and with '**Edit**' '**Paste**'.
A further possibility is to select the elements and to again click in one of these elements with the <**Ctrl**> key held down. If you now hold the left mouse button down, you can separate the elements thus copied from the original.

Modifying Visualization Elements

You can select an element which has already been inserted by a mouse click on the element or by pressing the <tab> key. A small black square will appear at each corner of each of the elements, (with ellipses at the corners of the surrounding rectangle). Except in the case of polygons, lines or curves further squares appear in the middle of the element edges between the corner points.



With a selected element, the turning point (balance point) is also displayed at the same time. You can then rotate the element around this point with a set motion/angle. The turning point is displayed as a small black circle with a white cross ( ). You can drag the turning point with a pressed left mouse button.
You can change the size of the element by clicking on one of the black squares and, while keeping the left mouse button pressed, controlling the new outline.
With the selection of a polygon, you can drag each individual corner using the same technique. While doing this, if you press the <Ctrl>-key then an additional corner point will be inserted at the corner point, an additional corner point will be inserted, which can be dragged by moving the mouse. By pressing the <Shift>+<Ctrl>-key, you can remove a corner point.

Dragging Visualization Elements

One or more selected elements can be dragged by pressing the left mouse button or the arrow key.

Grouping Elements

Elements can be grouped by selecting all desired elements and performing the command 'Extras' 'Group'. The group will behave like a single element:

- the grouped elements get a collective frame; when dragging the frame all elements will be stretched or compressed; only the group can be moved to another position.
- the grouped elements get collective properties: inputs only can effect the group and not a single element. Thus the elements also get one collective configuration dialog (category 'group'). The property 'Change color' can not be configured for a group !

To redefine a single element of a group, the grouping must be redone by the command 'Extras' 'Break up group'. The configuration of the group will be lost in this case.

.

'Extras' 'Send to Front'

Use this command to bring selected visualization elements to the front.

'Extras' 'Send to Back'

Use this command to send selected visualization elements to the back.

'Extras' 'Align'

Use this command to align selected visualization elements.
The following alignment options are available:

- **Left**: the left edge of each of the elements will be aligned to the element that is furthest to the left
- the same is true for **Right / Top / Bottom**
- **Horizontal Center**: each of the elements will be aligned to the average horizontal center of all elements
- **Vertical Center**: each of the elements will be aligned to the average vertical center of all elements

'Extras' 'Elementlist'

This command opens a dialog box containing a list of all visualization elements including their number, type, and position. The position is given according to the x and y position of the upper left (x1, y1) and the lower right (x2, y2) corner of the element.

When one or more items have been selected, the corresponding elements in the visualization are marked for visual control and if necessary, the display will scroll to that section of the visualization that contains the elements.

Use the To front button to bring selected visualization elements to the front. Use the To behind button to move them to the back.

Below the elements list there you find – depending on which element is currently selected - one of the following combinations of edit fields where you can modify size and position of the element:

- If a rectangle, rounded rectangle, ellipse, bitmap, visualization, button, or a meta file is currently selected, then next to the text "Rectangle (x1, y1, x2, y2)" there are four edit fields, where the actual x/y positions are shown and can be modified.
- If a line, polygon, or a curve is currently selected, a table will be available showing the actual X-Position and Y-Position of each of the black squares which mark the shape of the element, as soon as it is selected. These values can be edited here.

To set the modified position values in the elements list and in the visualization, press button Set rectangle (in case 1.) resp. Set polygon (in case 2.).

Use the Delete button to remove selected visualization elements.

Use the Undo and Redo buttons to undo or restore changes that have been made just as you would do with the commands 'Edit' 'Undo' and 'Edit' 'Redo' . In the dialog box, you can observe the changes that are being made.

Click on **OK** to close the dialog box and confirm the changes.

Use **Configure** to get the configuration dialog for the element

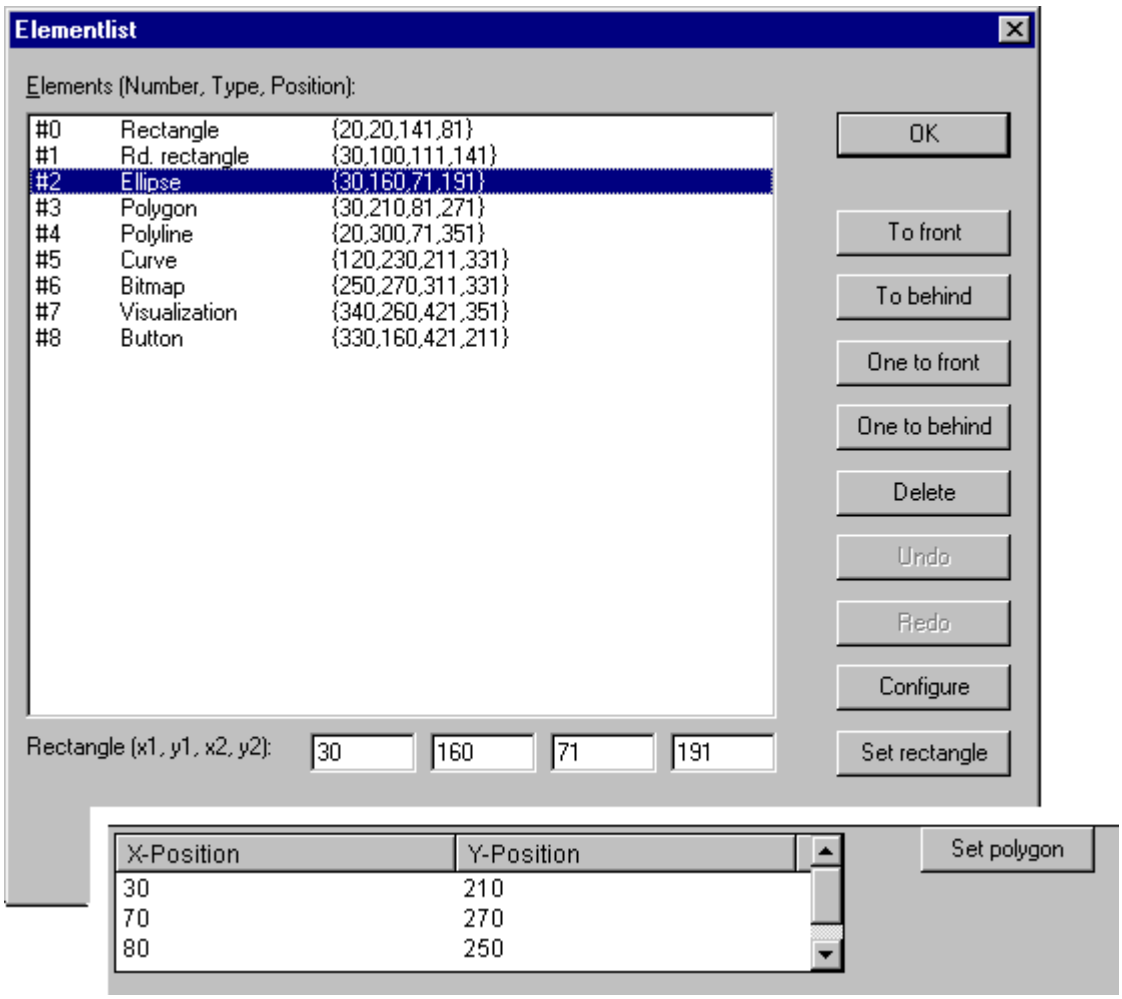


Fig. 4: Dialog Elementlist

Status Bar in the Visualization

If a visualization has the focus, the current X and Y position of the mouse cursor in pixels relative to the upper left corner of the image is displayed in the status bar. If the mouse pointer is located on an Element, or if the element is being processed, then the number of the element will be displayed. If you have selected an element to insert, then this element will also appear (for example, Rectangle).

10.1.4 Configure Visualization

When configuring a visualization you have to distinguish between the configuration a particular graphic element and the visualization object as a whole. Correspondingly a different selection of configuration dialogs will be available, which can be opened by the command '**Configure**' from Menu '**Extras**' resp. from the context menu.
In this dialogs the properties of an element or object are set either by activating options or dynamically by inserting project variables. Besides that the properties can be programmed via the components of a structure variable, which can be defined for each visualization element.

Regard the order of analysis, which will be followed in online mode:
- The values which are given dynamically, i.e. via project variables, will overwrite the fix parameters defined for the same property.
- If an element property is defined by a "normal" project variable as well as by the component of a structure variable, then in online mode primarily the value of the project variable will be regarded.

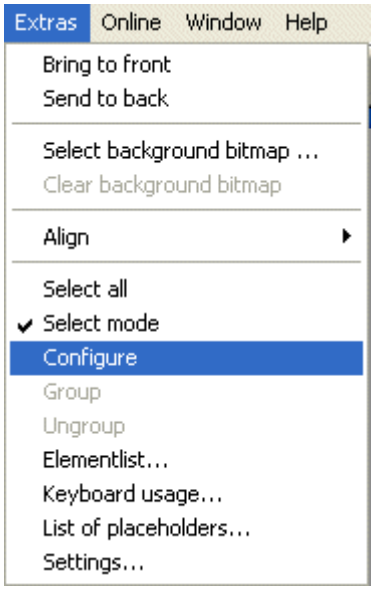
The possibility of using placeholders should also be noted, as well as the special input options with regard to the use of a visualization with the TwinCAT PLC Control visualization, i.e. as an exclusive operating interface for a control program.

Placeholders

At each location in the configuration dialog at which variables or text are entered, a placeholder can be set in place of the respective variable or text. This makes sense if the visualization object is not to be used directly in the program, but is created to be inserted in other visualization objects as an "instance". When configuring such an Instance, the placeholders can be replaced with variable names or with text.

See for Placeholder [▶ 316] concept.

10.1.4.1 Configure Visualization Elements



'Extras' 'Configure'

With this command, the 'Configure element' dialog opens for configuring the selected visualization element (see **Select visualization element**). You are given the dialog box when you double-click on the element. Select a category in the left area of the dialog box (available categories depending on element type) and fill out the requested information in the right area. This has to be done by activating options resp. by inserting the name of valid variables, whose values should define the property of the element.

There are also configuration dialogs available for a group of elements. Regard that the settings will be valid for the "element" group. If you want to configure the particular elements of the group, you have to resolve the group.

If you have defined an element property by a "static" setting as well as dynamically by a variable, then in online mode the variable will overwrite the static value (Example: "Alarm color Inside" can be defined statically in category 'Color' and additionally dynamically in category 'Colorvariables' by a variable). If the setting is controlled by a "normal" project variable as well as by a structure variable, then the value of structure variable also will be overwritten by the "normal" project variable.



Meter, Bar Display and Histogram must be re-grouped before!

At locations in the element configuration where variables are operative, the following Entries are possible:

- Variable names, for which input assistant is available
- Expressions which are assembled from component accesses, field accesses with constant index, variables, and direct addresses.
- Operators and constants, which can be combined at will with the expressions.
- Placeholders instead of variable names or text strings.

Examples of permissible expressions:

```
 x + y
100*PLC_PRG.a
TRUE
NOT PLC_PRG.b
9*sin(x + 100)+cos(y+100)
```

Function calls are not possible. Invalid expressions result in an error message on login („Invalid Watch expression...").

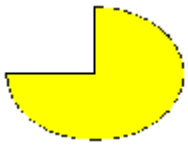
Examples of invalid expressions:

```
fun(88)
a := 9
RETURN.
```

There are two possible ways in the configuration dialogs to write global variables: „.globvar" and „globvar" are equivalent. The style with a dot (which is that used in the Watch- and Recipe Manager) is not allowed within an assembled expression, however.
Regard also the possibility of using Placeholders.

Angle

In the configuration dialog 'Configure Pie' in the Angle category you can each enter a value or a variable defining the **start angle** and the **end angle** of the sector element in degrees. The sector will be drawn clockwise from the start angle position to the end angle position.
Example:
Enter start angle: "90", end angle: "180"



Dialog for Configuration of angles:

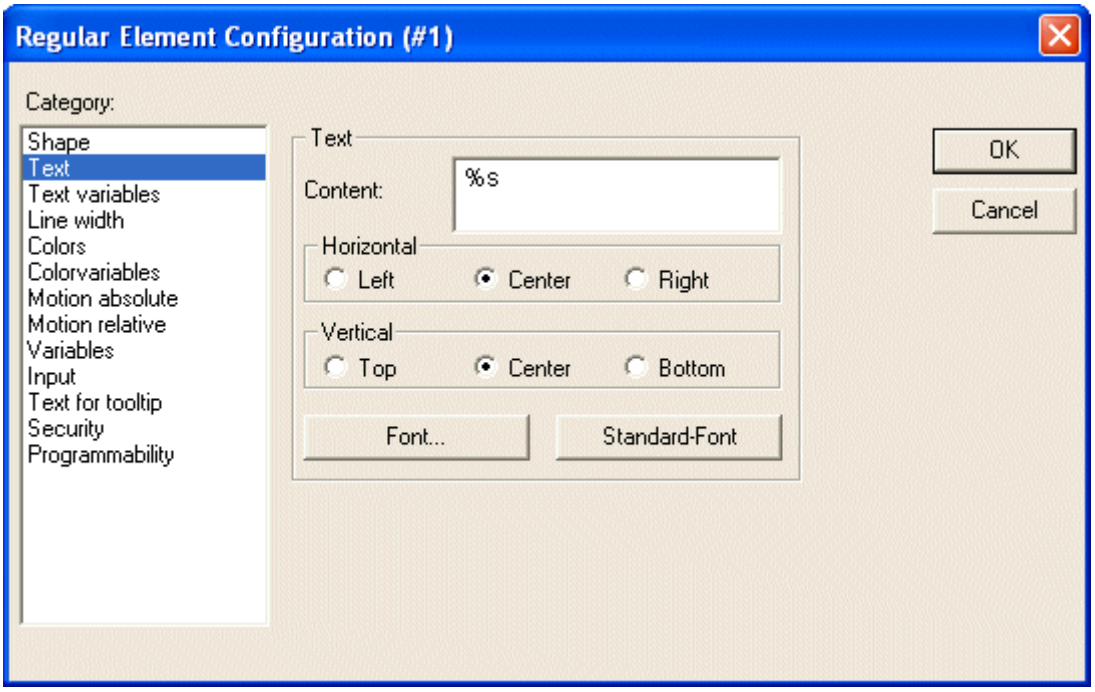
Shape

In the visualization element configuration dialog box, you can select in the **Shape** category from among **Rectangle**, **RoundedRectangle**, **Line** and **Ellipse** respectively **Polygon**, **Line** and **Curve**.

The form will change into the size already set.

Text

Dialog for configuration of elements



In the dialog for configuring visualization elements, you can specify a text for the element in the Text category. This can be entered directly or/and a variable can be defined which will determine the text string. The usage of placeholders is possible. Also the default settings for font and alignment are done here.

As soon as text parameters are additionally provided dynamically, which means by a system or structure variable (see also below, category 'Text variables' resp. 'Programmability'), the static definitions which are done in the currently opened dialog , will be overwritten !

In case of multiple definition of an element property consider the specific order of precedence concerning according to which a value might be overwritten in online mode by another.

Enter the text in the **Content** field. With the key combination <Crtl>+<Enter> you can insert line breaks, with <Crtl>+<Tab>, tab stops. Besides the input of a pure text string you can use the following formatting sequences:

- If you include "%s "into the text, then this location, in Online mode, will be replaced by the value of the variable from the Text Output field of the Variables category. You also can use a formatting string, which conforms with the standard C-library function 'sprintf':

Character	Argument / Output as
d,i	Decimal number
o	Unsigned octal number (without leading zerol)
x	Unsigned hexadecimal number (without leading 0x)
u	Unsigned decimal number
c	Single character
s	String
f	REAL-values [-]m.dddddd, whereby the accuracy defines the number of d's (Default is 6). The plus sign resp. minus sign defines right aligned (default) or left aligned , m is the number of all decimal (position before decimal point, decimal point, position after decimal point) d defines number of decimal after point.

The value of the variable will be displayed correspondingly in online mode. You can enter any IEC-conforming format strings, which fit to the type of the used variable.

It is not checked whether the type which is used in the formatting string matches with the type of the variable which is defined in the 'Text Output' field !

Example:
Input in the 'Content' field: Fill level %2.5f mm
Input in the 'Text Output' field e.g.: fvar1 (REAL variable)
→Output in online mode e.g.: Fill level 32.89997 mm

- If you enter "%t ", followed by a certain sequence of special placeholders, then this location will be replaced in Online mode by the system time. The placeholders define the display format, see the following table.

Do not insert any other characters before %t in the 'Content' field (in contrast this is allowed for e.g. "%s", see above)

Placeholder	Format
%a	Abbreviated weekday name. I.e. "Wed"
%A	Full weekday name. I.e. "Wednesday"
%b	Abbreviated month name. I.e. "Feb"
%B	Full month name. I.e. "February"
%c	Date and time representation appropriate for locale <Month>/<Day>/<Year> <Hour>:<Minutes>:<Seconds>, i.e. "08/28/02 16:58:45"
%d	Day of month as decimal number (01-31), i.e. "24"
%H	Hour in 24-hour format (00 – 23), i.,e. "16"
%I	Hour in 12-hour format (01 – 12), i.e. "05"
%j	Day of year as decimal number (001 – 366), i.e. "241
%m	Month as decimal number (01 – 12) i.e. "3" for March
%M	Minute as decimal number (00 – 59) i.e. "13"
%p	Current locale's A.M./P.M. indicator for 12-hour clock.
%S	Second as decimal number (00 – 59)
%U	Week of year as decimal number, with Sunday as first day of week (00 – 53)
%w	Weekday as decimal number (0 – 6; Sunday is 0)
%W	Week of year as decimal number, with Monday as first day of week (00 – 53)
%x	Date representation for current locale <Month>/<Day>/<Year>, i.e. "08/28/02"
%X	Time representation for current locale <hours>:<Minutes>:<Seconds>, i,e. "16:58:45"
%y	Year without century, as decimal number (00 – 99), i,e. "02"
%Y	Year with century, as decimal number i,e. "2002"
%z, %Z	Time-zone name or abbreviation; no characters if time zone is unknown
%%	Percentsign

Examples:

%t%a %b %d.%m.%y %H:%M:%S
-> Display in online mode: Wed Aug 28.08.02 16:32:45

Between the placeholders you can insert any text strings:
%Today is %d.%m.%y
-> Display in online mode: Today is 28.08.02

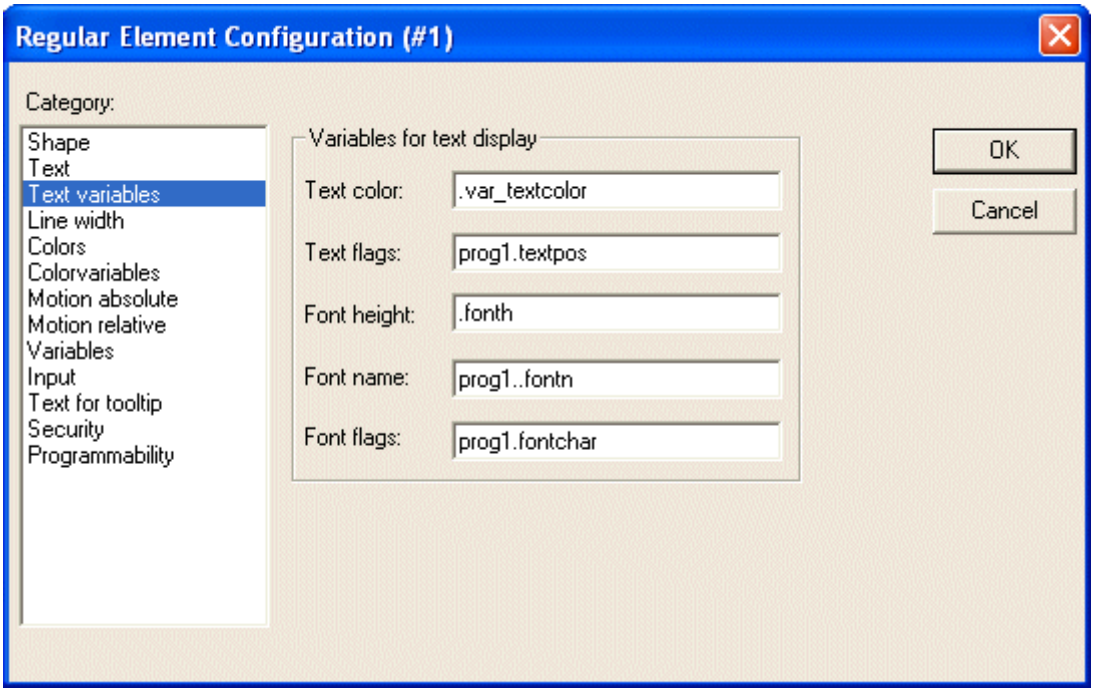
If a text string is to be transferred into a translation file, which will then be used in Online mode to enable switching into another national language, it must be delimited at the beginning and end by #.
Examples: "#Pump 1#" or else even "#Pump# 1"
The second case might for example, in the event of multiple occurrences of the text Pump (Pump 1, Pump 2, etc.), prevent multiple appearances in the translation

- If you include" %<PREFIX>" into the text, you can enter instead of "PREFIX" a certain string, which will serve as an identifier concerning the use of dynamic texts. The prefix will be used together with an ID number, which is to be defined in the 'Variables' category of the configuration dialog in field 'Textdisplay'. The combination references to a certain text, which is contained in a xml-file listing all possible dynamic texts.Thus at run time the text which is indicated by the current ID-Prefix-combination will be displayed. For further information see also the description of 'Settings', category Language

The configured text will appear online in the prescribed alignment within the element: **horizontallyleft**, **center** or **right** and **verticallytop**, **center** or **bottom**.
If you use the **Font** button, a dialog box for selection of the font will appear. Select the desired font and confirm the dialog with **OK**. With the **Standard-Font** button you can set the font that is selected in the project options ('Project' 'Options' 'Editor'). If the font is changed there, then this font will be displayed in all elements except in those elements for which another font has explicitly been selected by using the **Font** button

Textvariables

Dialog Box for Configuring Visualization Elements (Text Category)



variable which should dynamically set color and font of that string which is defined in category 'Text'. At best enter the variable name with the aid of the input assistant (<F2>).
You can also use components of the structure VisualObjectType to set the text properties. For this see the description of category 'Programability'; there you will find the possible values of the particular structure components and their effect.

If there are corresponding static definitions in category 'Text', these will be overwritten by the dynamic parameter values



In case of multiple definition of an element property consider the specific order of precedence concerning according to which a value might be overwritten in online mode by another.

The parameters of the dialog:

Parameter	Meaning:	Example entry of project variable:	Example Usage of variable in program:	corresponding component of structure VisualObjectType:
Textcolor:	Textcolor	"plc_prg.var_textcolor"	var_textcolor=16#FF00FF -> color	dwTextColor
Textflags:	Alignment (right, left, centered...)	"plc_prg.textpos"	textpos:=2 --> Text right justified	dwTextFlags
Fontheight:	Font height in Pixel	".fonth"	fonth:=16; -> Font height 16 pt	ntFontHeight
Fontname:	Font-name	"vis1.fontn"	fontn:=arial; -> Arial is used	stFontName
Fontflags:	Font display (bold, underlined, italic...)	"plc_prg.fontchar"	fontchar:=2 -> Text will be displayed bold	dwFontFlags

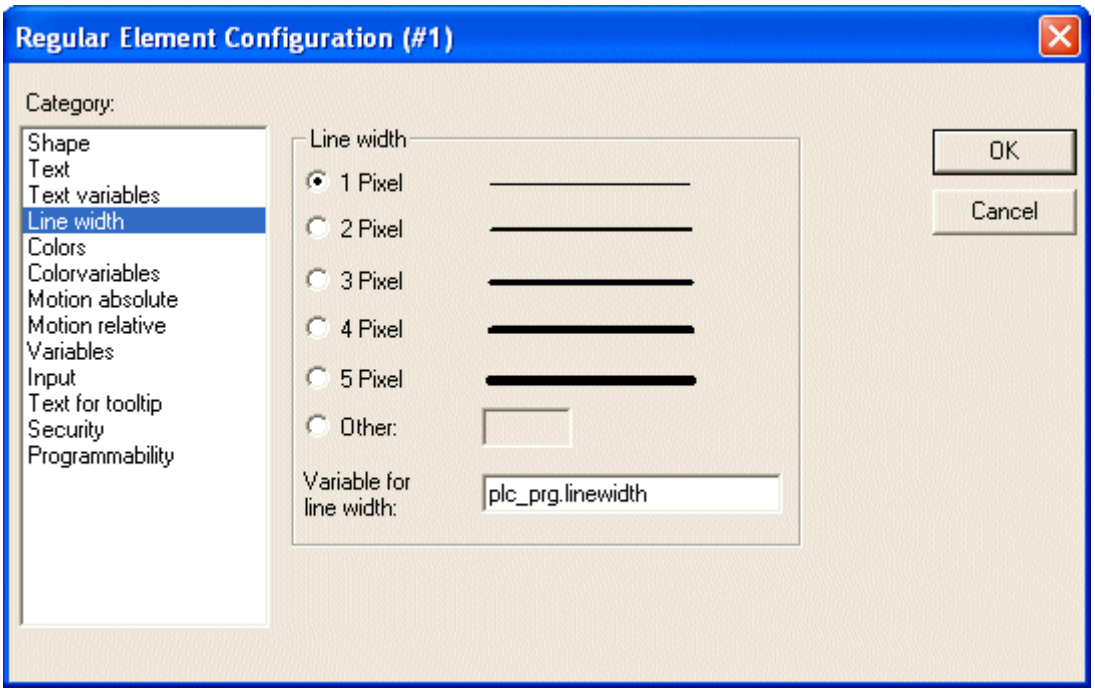
Line width

In the dialog for configuring visualization elements, you can choose the line width for an element. As predefined options you find width settings from 1 to 5 pixel, additionally an other value can be entered manually (Other:), or a project variable (Variable for line width:) can be inserted. For the latter the input assistance ((<F2>) can be used.

As soon as the parameter is additionally defined dynamically, i.e. by a structure variable (see below, category 'Programmability'), the static setting will be overwritten in online mode.

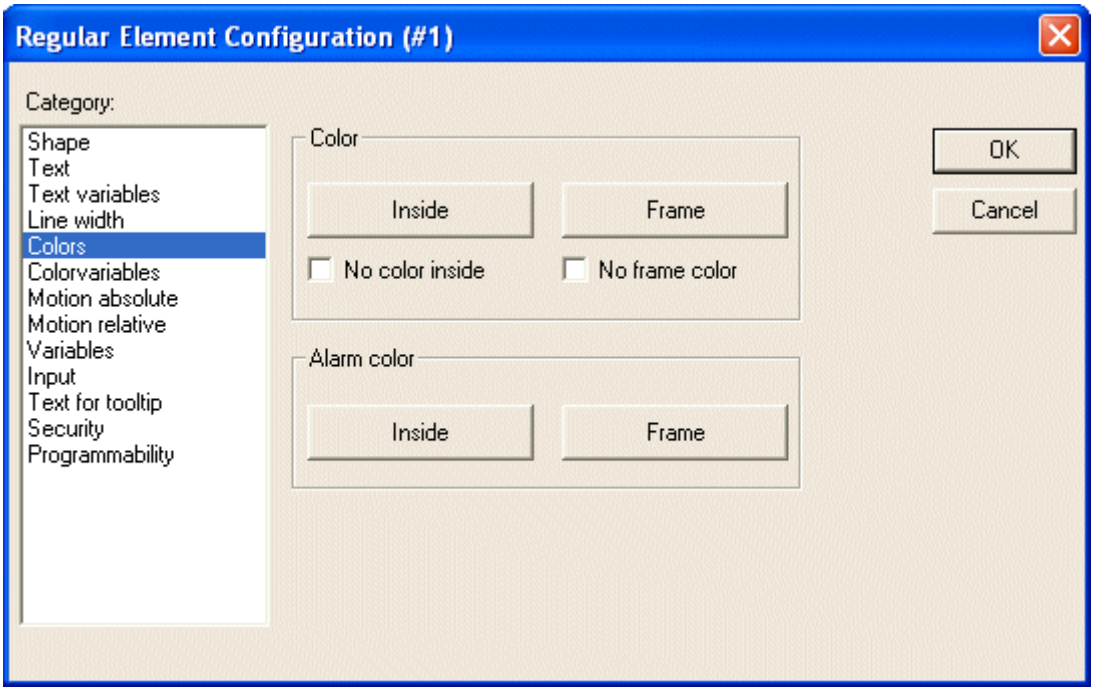


In case of multiple definition of an element property consider the specific order of precedence concerning according to which a value might be overwritten in online mode by another.



Dialog for configuring visualization elements (category Textvariables)

Colors



Dialog Box for Configuring Visualization Elements (Line width category)

In the visualization element configuration dialog box, in the Color category you can select primary colors and alarm colors for the inside area and for the frame of your element. Chosing the options no color inside and no frame color you can create transparent elements

As soon as the parameter is additionally defined dynamically by a variable, the static setting will be overwritten in online mode



In case of multiple definition of an element property consider the specific order of precedence concerning according to which a value might be overwritten in online mode by another.

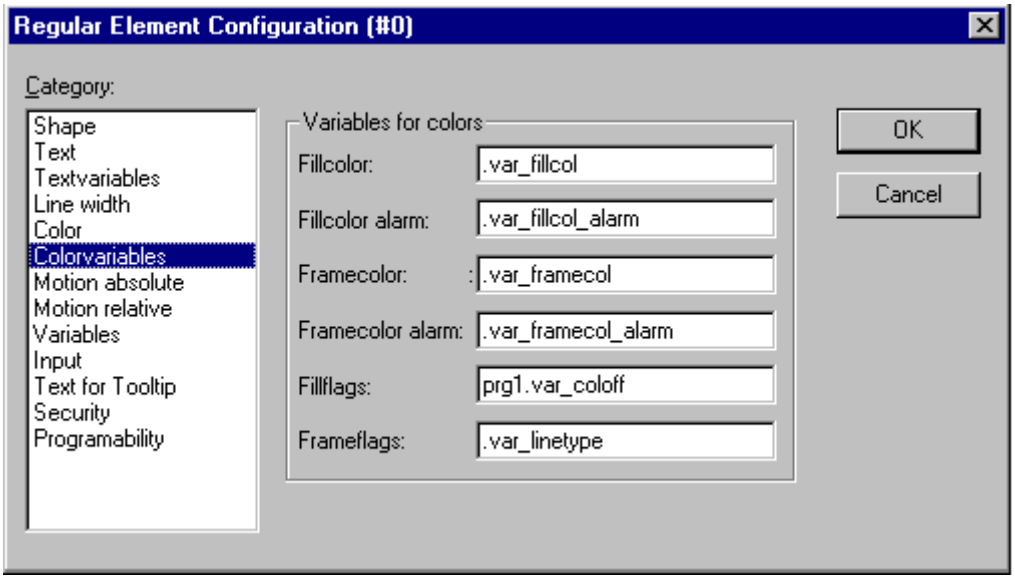
Dialog Box for Configuring Visualization Elements (Color Category)

If you now enter a Boolean variable in the **Variables** category in the **Change Color** field, then the element will be displayed in the Color set, as long as the variable is FALSE. If the variable is TRUE, then the element will be displayed in its **Alarm Color.**

The change color function only becomes active, if the PLC is in Online Mode!

If you want to change the color of the frame, then press the **Frame** button, instead of the **Inside** button. In either case, the dialog box will open for selection of the color.
Here can to choose the desired hue from the primary colors and the user-defined colors. By pressing the Define Colors you can change the user-defined colors

Color Variables



Dialog Box for Configuring Visualization Elements (Color Category)

Here you can enter project variables (e.g. PLC_PRG.color_inside), which should determine the particular property in online mode: These property definitions also or additionally can be programmed with the aid of components of the structure VisualObjectType. Therefore see the description on the "Programability" of a visualization element. There you will find a list of the possible values and their effects.

The variables, entered in the Color Variables dialog, in online mode will overwrite the static values given in the 'Color' category as well as corresponding values given by a structure variable.



In case of multiple definition of an element property consider the specific order of precedence concerning according to which a value might be overwritten in online mode by another.

The parameters of the dialog:

Parameter	Description:	Example of an entry:	Example for using the variable in the program:	corresponding component of structure VisualObjectType
Fillcolor:	fill color	"plc_prg.var_fillcol"	var_fillcol:= 16#FF00FF -> Fill color Pink	dwFillColor
Fillcolor alarm:	fill color if the 'Change color' variable is TRUE	"plc_prg.var_fillcol_a"	var_fillcol_a:= 16#FF00FF -> Alarm fill color Pink	dwFillColorAlarm
Frame color:	frame color	"plc_prg.var_framecol"	var_framecol:= 16#FF00FF -> frame color Pink	dwFrameColor
Framecolor Alarm:	frame color if the 'Change color' variable is TRUE	"plc_prg.var_framecol"	var_framecol:= 16#FF00FF -> alarm frame color Pink	dwFrameColorAlarm
FillFlags:	The current inside color configuration can be activated (FALSE) resp. deactivated (TRUE).	"plc_prg.var_col_off"	var_col_off:=1 □ the color definition for the fill color will not be regarded, that for the frame remains valid	dwFillFlags
FrameFlags:	Display of the frame (solid, dotted etc.)	"plc_prg.var_linetype"	var_linetype:=2; -> frame will be displayed as dotted line	dwFrameFlags

Motion absolute

In the visualization element configuration dialog box, in the Motion absolute category, X- or Y-Offset fields variables can be entered. These variables can shift the element in the X or the Y direction, depending on the respective variable value. A variable in the Scale field will change the size of the element linear to its current value. This value, which is used as scaling factor, will be divided by 1000 implicitely, so that it is not necessary to use REAL-variables in order to get a reduction of the element. The scaling always will refer to the balance point.

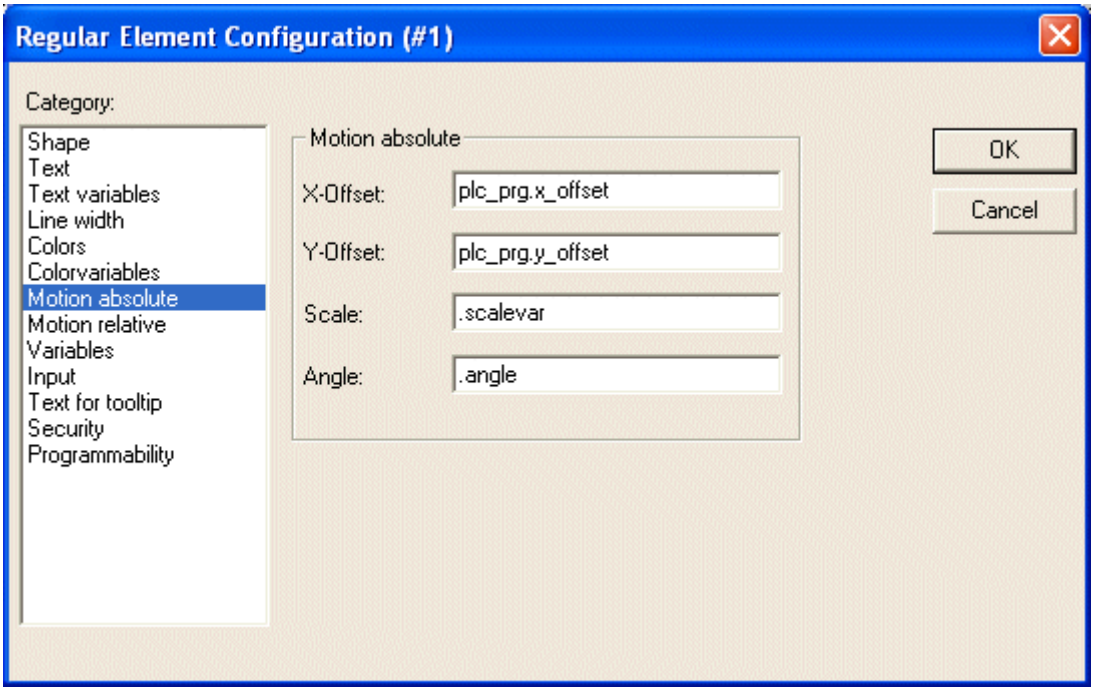
A variable in the Angle field causes the element to turn on its turning point, depending on the value of the variable. (Positive Value = Mathematic Positive = Clockwise). The value is evaluated in degrees. With polygons, every point rotates; in other words, the polygon turns. With all other elements, the element rotates, in such a way, that the upper edge always remains on top.

The turning point appears after a single click on the element, and is displayed as a small black circle with a white cross. You can drag the turning point with a pressed left mouse button.

In online mode the variables which are set in the 'Motion absolute' dialog will override the values of structure components which additionally might be used to define the same property ('Programability').

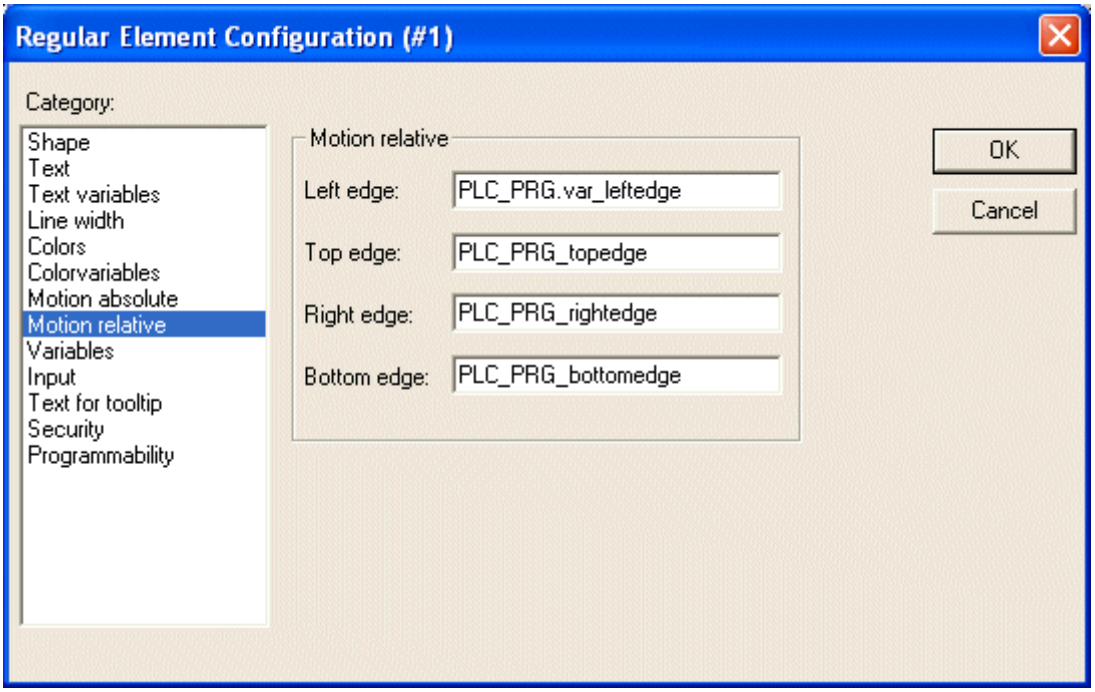


In case of multiple definition of an element property consider the specific order of precedence concerning according to which a value might be overwritten in online mode by another.



Visualization Element Configuration Dialog Box (Motion Absolute Category)

Motion relative



Dialog Box for Configuration of Visualization Elements (Motion Relative Category)

In the dialog for configuring visualization elements in the Motion Relative category, you can assign variables to the individual element edges. Depending on the values of the variables, the corresponding element edges are then moved. The easiest way to enter variables into the fields is to use the Input Assistant (<F2>).

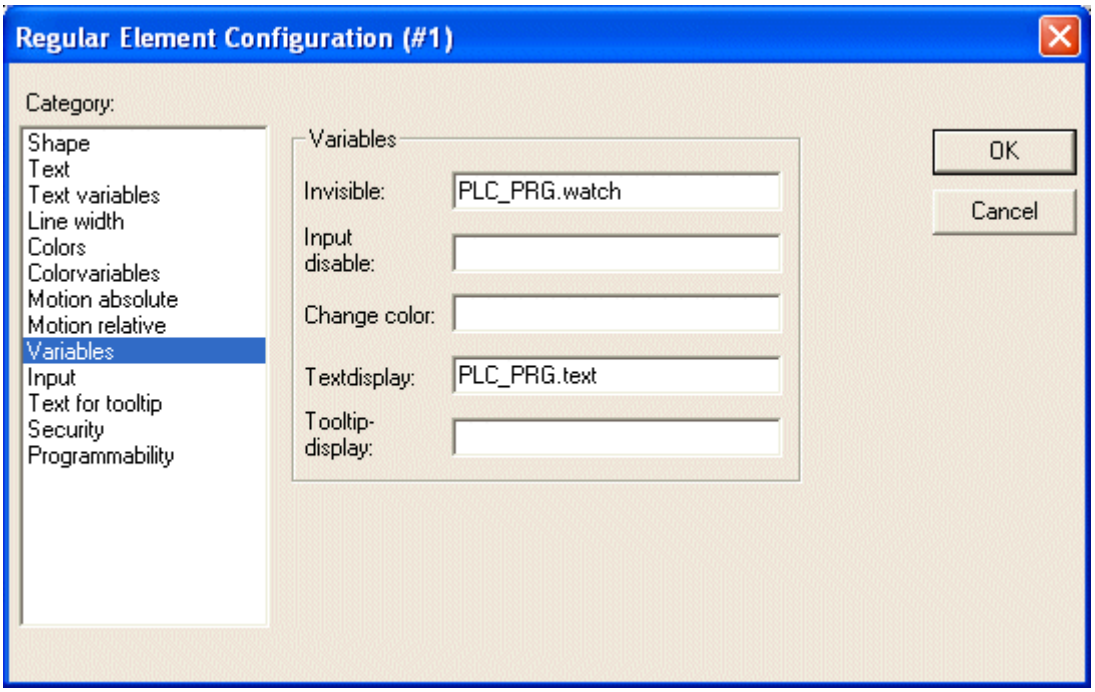
The four entries indicate the four sides of your element. The base position of the corners is always at zero. A new value in the variables, in the corresponding column, shifts the boundary in pixels around this value. Therefore, the variables that are entered ought to be INT variables.

Positive values shift the horizontal edges downward, or, the vertical edges, to the right!

In online mode the variables which are set in the 'Motion absolute' dialog will override the values of structure components which additionally might be used to define the same property ('Programmability').

In case of multiple definition of an element property consider the specific order of precedence concerning according to which a value might be overwritten in online mode by another.

Variables



You can enter the variables that describe the status of the visualization elements in the Variable category within the dialog box for configuring visualization elements. The simplest way to enter variables in the fields is to use the Input Assistant.

In online mode the variables which are set in the 'Motion absolute' dialog will override the values of structure components which additionally might be used to define the same property ('Programability').

In case of multiple definition of an element property consider the specific order of precedence concerning according to which a value might be overwritten in online mode by another.

You can enter Boolean variables in the Invisible and Change color fields. The values in the fields determine their actions. If the variable of the Invisible field contains the value FALSE, the visualization element will be visible. If the variable contains the value TRUE, the element will be invisible.

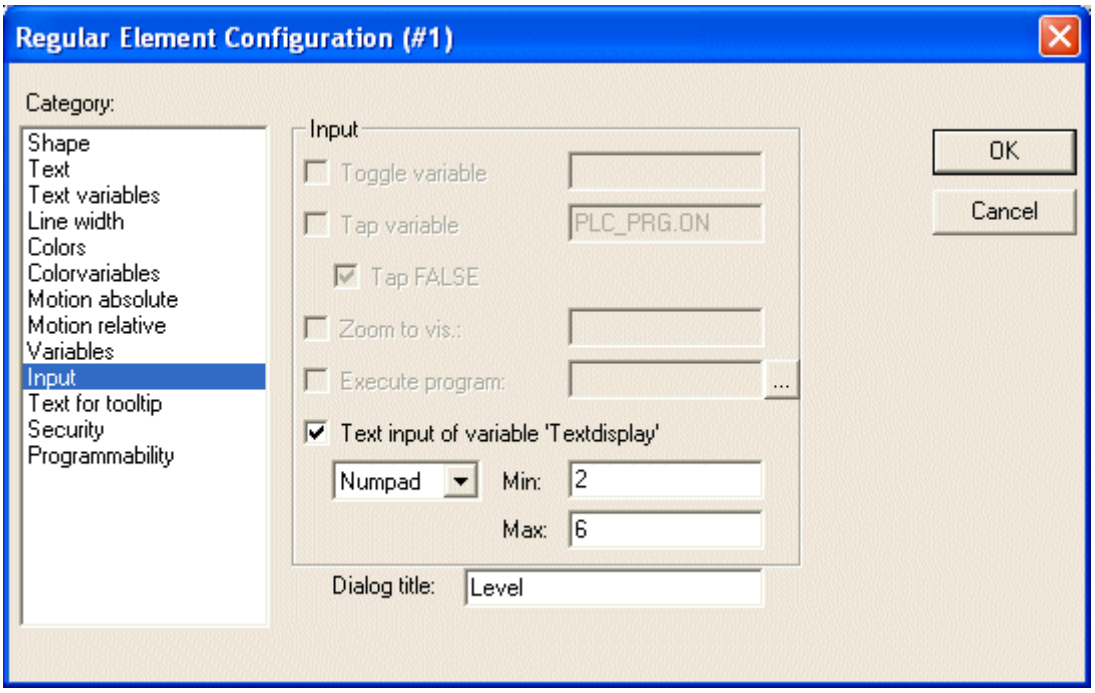
Disable input: If the variable entered here is TRUE, all settings of category 'Input' will be ignored.
Change color: If the variable which is defined in this field, has the value FALSE, the visualization element will be displayed in its default color. If the variable is TRUE, the element will be displayed in its alarm color.

Textdisplay:

- If you have inserted a "%s" in the Content field of the Text category or if you have included "%s" in the textstring, then the value of the variable which is defined in 'Textdisplay' will be displayed in online mode in the visualization object. "%s" will be replaced by the value.
- If you have inserted resp. included a "%<PREFIX>" in the Content field of the Text category ("PREFIX" must be a sequence of letters), then the variable resp. the numeric value which is entered here in 'Textdisplay' will be interpreted as an ID, which in combination with the prefix serves as a reference on a text, which is described in a XML-file. This text will be displayed in online mode instead of "%<PREFIX>" in the visualization object. Thus a dynamic modification of the text display is possible. See further information in the description of the dialog 'Settings', category Language.
- If you want to edit the value of the variable in Online mode using the keyboard, you can do this via the 'Text input of variable' 'Textdisplay' in the Input category.

Tooltip-display: Enter here a variable of type STRING whose value should be displayed in a tooltip for the element in online mode.

Input



Dialog for configuring the visualization elements (Category Input)

Toggle variable: If this option is activated, in online mode you will toggle the value of the variables which are located in the input field by each mouse click on the visualization element. You can obtain input assistance for data entry via <F2>. The value of the Boolean variable changes with each mouse click from TRUE to FALSE and then back to TRUE again at the next mouse click, etc.

Tap Variable: If this option is activated, in online mode you can switch the value of the Boolean variable which is located in the input field, between TRUE and FALSE. Place the mouse cursor on the element, press the mouse-key and hold it depressed. If option Tap FALSE is activated, the value is set to FALSE as soon as the mouse key is pressed, otherwise it is set to TRUE at this moment. The variable changes back to its initial value as soon as you release the mouse key.

Zoom to Vis...: If this option is activated, you can enter in the edit field the name of a visualization object of the same project to which you want to jump by a mouse-click on the element in online mode. In this case always first the window of the target visualization will be opened before that of the current one will be closed.

The following entries are allowed:

- The name of a visualization object of the current project (see Object Organizer)
- If a visualization reference that contains placeholders is to be jumped to, the placeholders can be directly replaced by variable names or text when called up. For this purpose, conform to the following syntax:

<Visuname>(<Placeholder1>:=<Text1>, <Placeholder2>:=<Text2>,..., <Placeholder n>:=<Textn>).
During compilation of the visualization it will be checked, whether the text matches with one of the defined replacement values defined in the placeholder list, if it does not, a warning will be output.
Example:
Calling the visualization visu1, whereby the placeholders var_ref1 and var_ref2 used in visu1 are replaced by the variables PLC_PRG.var1 and PROG.var1 respectively:
visu1(var_ref1:=PLC_PRG.var1, var_ref2:=PROG.var1

- If a program variable of the type STRING (e.g. PLC_PRG.xxx) has been entered instead of a visualization object, then this variable can be used to define the name of the visualization object (e.g. ‚visu1') which the system should change to when a mouse click occurs (e.g. xxx:= ‚visu1).
- If you issue the command „ZOOMTOCALLER" in the Zoom to vis. field, a backward jump into the calling visualization is achieved in Online mode by a mouse click on the element, if such a constellation was configured.

The use of a program variable is not permitted for the PLC HMI CE.

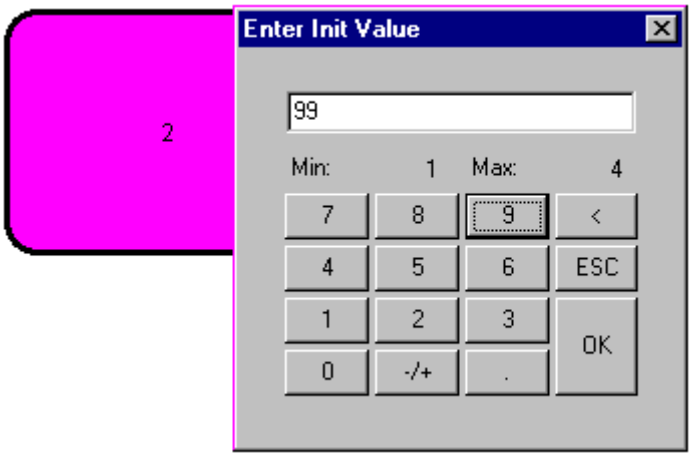
The implicit variable CurrentVisu (type STRING, for implicit (system) variables see here) describes the name of the currently opened visualization object. For example it can be used in the application to control which visualization should be opened resp. to see which is the currently opened. But this will only work if the names of the visualization objects are defined in capital letters (see 'Create a visualization object [▶ 259]').
Example: CurrentVisu:='TC_VISU';

Execute program: If this option is activated you can enter ASSIGN- or special "INTERN"-commands in the input field, which will be executed in online mode as soon as you perform a mouse-click on the element. Press button "..." to get the dialog **Configure programs** where you can select the desired commands (Add) and arrange them in the desired order (Before, After).

This feature especially is important if the visualization will be the only operating interface of a system (pure operating version). See a description of the commands: Special input possibilities for the TwinCAT PLC Control operating version.

If you select the Text input of variable 'Textdisplay', then in Online mode you will get the possibility to enter an value in this visualization element which will upon pressing <Enter> be written to the variable that appears in the Textdisplay field of the Variables category.

Select in the scroll box which kind of input should be possible later in online mode. **Text**: An edit field will open, where you can enter the value. **Numpad** resp. **Keypad**: A window will open showing an image of the numeric resp. alphabetic keypad, where you can enter a value by activating the appropriate key elements. This might be useful if the visualization must be operatable via a touch screen. The range of valid input values can be restricted by defining a minimum and a maximum value in the edit fields **Min**: and **Max**:



In case of multiple definition of an element property consider the specific order of precedence concerning according to which a value might be overwritten in online mode by another.

Tooltip

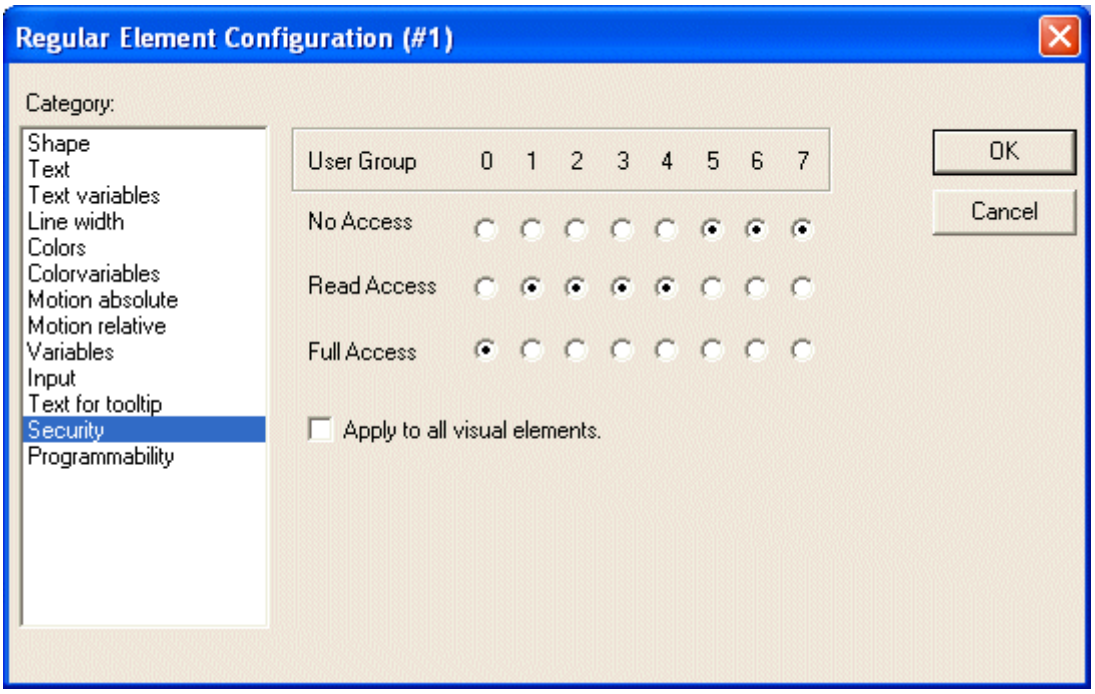
The dialog Text for Tooltip offers an input field for text which appears in a text field as soon as the mouse cursor is passed over the object in online mode. The text can be formatted with line breaks by using the key combination <Ctrl> + <Enter>.

In case of multiple definition of an element property consider the specific order of precedence concerning according to which a value might be overwritten in online mode by another.

Security

It might be useful that different user groups get different operating possibilities and display of a visualization. This can be reached by assigning different access rights concerning particular visualization elements. You can do this for the eight user groups which are available in TwinCAT PLC Control (see also 'Project' 'Object' 'Properties' resp. 'Project' 'User Group Passwords'). The access rights can be assigned by activating the appropriate option in the configuration dialog 'Access rights' for a visualization element:

Visualization Element Configuration Dialog Box (Category Security)



The access rights for a visualization element and their effect in online mode:

Access rights	Description:
No Access	Element will not be visible
Read Access	Element will be visible but not operatable (no inputs allowed)
Full Access	Element is not visible and not operatable

If you want to assign the access rights also to all other elements of the visualization object, activate option Apply to all visual elements.

Please regard, that the access rights which are set for the visualization object in the 'Project' 'Object' 'Properties' dialog, are independent on those of the particular visualization elements !

Programability

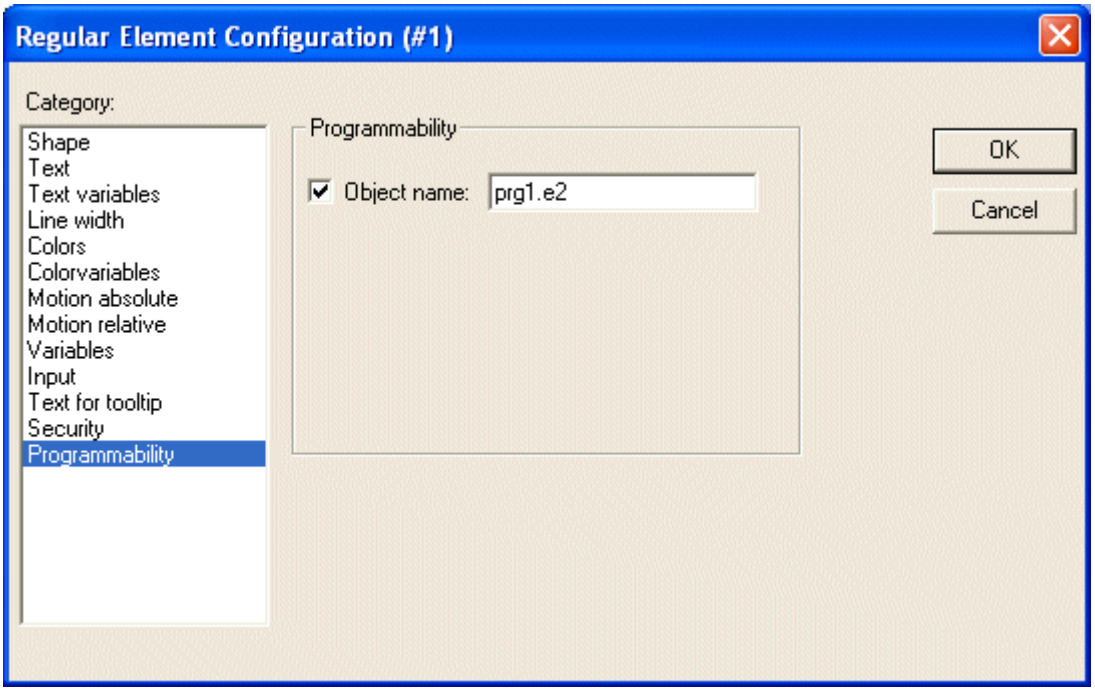
The properties of an visualization element can not only be defined by a static setting or by a "normal" project variable, but also by the components of a structure variable, which is exclusively used for programming visualization elements.
For this purpose the structure **VisualObjectType** is available in the library **SysLibVisu.lib**. Its components can be used to define most of the element properties.

> Note the evaluation sequence that applies later in online mode:
> - The values supplied dynamically, i.e. via normal project variables or via the structure variables, overwrite the fixed settings of the element configurations.
> - If an element property is addressed both by a project variable entered directly in the configuration dialog and via the component of a structure variable, the value of the project variable is evaluated first in online mode.

To configure the element properties via a structure variable, proceed as follows:

In the configuration dialog of the programmability category, enter a new, **unique** (!) variable name in the **Object name:** field in the project. To do this, you need to activate the option by clicking in the check box. The variable automatically receives the type of the structure **VisualObjectType**, which is part of a library. The declaration is implicit and not directly visible in the project.

After the next compilation run, the structure variable assigned to the visualization element can be used in the project. (Hint: activate the Intellisense function 'List components' in the project options, category Editor, to get the structure elements offered in a selection list after entering the variable name followed by a dot). For example, if you have entered an object name "visu1_line" for a visualization element, you can use visu1_line.nLIneWidth:=4 in the program to set the line thickness for this element.



Dialog for configuring the programmability of a visualization element (category Programmability)

The structure VisualObjectType:

The following table shows all elements available in structure VisualObjectType and the corresponding configuration option in the dialogs of the different categories:

The data type is integrated at the beginning of the component name. Where:
n INT
dw DWORD
b BOOL
st STRING

Component (+data type)	Meaning	Sample (the object name "vis1" was defined for the element.)	corresponding setting option in the configuration dialog:
nXOffset:INT;	Move the element in X direction	vis1.nXOffset:=val2; (element is set to position X=val2)	- Cat. Absolute movement: X-Offset
nYOffset:INT;	Move the element in Y direction	vis1.nYOffset:=22; (element is set to position Y=val2)	- Cat. Absolute movement: Y-Offset
nScale:INT;	Resize	vis1.nScale:=plc_prg.scale_var; (size of the element changes linearly with the change of variable plc_prg.scale_var)	- Cat. Movement absolute: Scaling
nAngle:INT;	Rotating the element around its pivot point	vis1.anglevar:=15; (element rotates 15 degrees clockwise)	- Cat. Absolute movement: Angle
Component (+data type)	Meaning	Sample (the object name "vis1" was defined for the element.)	corresponding setting option in the configuration dialog:
nLeft:INT;	Move the left element edge in X direction.	vis1.nLeft:=val2; (element edge moves to position X=val2)	- Cat. Absolute movement: X-Offset
nTop:INT;	Shifting the upper element edge in Y-direction. (pos.® move down)	vis1.nTop:=val2; (element edge moves to position Y=val2)	- Cat. Absolute movement: X-Offset
nRight:INT;	Move the right element edge in X direction.	vis1.nRight:=val2; (element edge moves to position X=val2)	- Cat. Absolute movement: X-Offset
nBottom:INT;	Move the bottom element edge to Y-direction. (pos.® move down)	vis1.nBottom:=val2; (element moves to position X=val2)	- Cat. Absolute movement: X-Offset
bInvisible:BOOL;	causes change between visible and invisible by changing TRUE and FALSE	vis1.visible:=TRUE; (element is invisible)	- Cat. Colors: no color inside + No Frame color - Cat. Color variables: FillFlags + FrameFlags
stTextDisplay: STRING;	Text that appears in the element	vis1.TextDisplay:='ON / OFF'; Element is labeled with this text	- Cat. Text: Entry at 'Content
bToggleColor: BOOL;	causes color change by changing TRUE and FALSE	vis1.bToggleColor:=alarm_var; (If alarm indicator alarm_var becomes TRUE, the element changes to the color it gets via the components dwFillColorAlarm, dwFrameColorAlarm or by the settings in the Conf.dialog Cat. Colors.	- Cat. Input: Variable toggle - Cat. Variables: Color change

Component (+data type)	Meaning	Sample (the object name "vis1" was defined for the element.)	corresponding setting option in the configuration dialog:
bInputDisabled: BOOL;	Inputs from category Input are considered with TRUE, not with FALSE	vis1.bInputDisabled:=FALSE; (no input possible on this element)	- Cat. Variables: disable input
stTooltipDisplay:STRING;	Tooltip text	vis1.stTooltipDisplay:='Switch for .....';	- Cat. Text for tooltip: Entry for 'Content'
Component (+data type)	Meaning	Sample (the object name "vis1" was defined for the element.)	corresponding setting option in the configuration dialog:
dwTextFlags: DWORD;	Text position: 1 left aligned 2 right aligned 4 horizontally centered 8 top 16 bottom 32 vertically centered **Both horizontal and vertical positioning should always be set (addition of values)!**	vis1.dwTextFlags:=36; (text is placed in the middle of the element (4 + 32)	- Cat. Text: Options from Horizontal and Vertical - Cat. Text variables: Text flags
dwTextColor : DWORD;	Text color (to enter the color values see after the table)	vis1.dwTextColor := 16#00FF0000; (text is displayed in blue)	- Cat. Text: Font | Color - Cat. Text variables: Text color
nFontHeight : INT;	Font height in pixels. Should be in the range 10-96.	vis1.nFontHeight:=16; (text height is 16 pt)	- Cat. Text: Font | Degree' - Cat. Text variables: Font height
dwFontFlags : DWORD;	Font representation. The following flags can be set: 1 italic 2 bold 4 underlined 8 strikethrough + Combinations by adding the values	vis1.dwFontFlags:=10; (text is displayed bold and strikethrough)	- Cat. Text: Font | Font style - Cat. Text variables: Fontflags
stFontName : STRING;	Font name	vis1.stFontName:='Arial'; (Arial as font for the text)	- Cat. Text: font | Font - cat. Text variables: Font name
nLineWidth : INT;	Line width of the element frame (number of pixels)	vis1.nLWidth:=3; (frame is 3 pixels thick)	-Cat. Line width
dwFillColor : DWORD;	Fill color. (to enter the color values see after the table)	vis1.dwFillColor":=16#00FF0000; (element is blue in "normal" state)	- Cat. Colors: Color | Interior - Cat. Color variables: Color Inside

Component (+data type)	Meaning	Sample (the object name "vis1" was defined for the element.)	corresponding setting option in the configuration dialog:
Component (+data type)	Meaning	Sample (the object name "vis1" was defined for the element.)	corresponding setting option in the configuration dialog:
dwFillColorAlarm : DWORD;	Fill color in case of alarm (by TRUE of component bToggleColor, see above) (for input of color values see after the table)	vis1.dwFillColorAlarm:= 16#00808080; (if variable togglevar is set to TRUE, the element becomes gray)	- Cat. Colors: Alarm color | Inside - Cat. Color variables: Alarm color inside
dwFrameColor: DWORD;	Frame color (for the input of the color values see after the table)	vis1.dwFrameColor:= 16#00FF0000; (frame is blue in "normal" state)	- Cat. Colors: Color | Frame - Cat. Color variables: Color for frame
dwFrameColorAlarm: DWORD;	Frame color in case of alarm (by TRUE of component bToggleColor, see above) (for input of color values see after the table)	vis1.dwFrameColorAlarm: =16#00808080; (if variable vis1.bToggleColor is set to TRUE, the element frame becomes gray)	- Cat. Colors: Alarm color Frame - Cat. Color variables: Alarm color for frame
dwFillFlags: DWORD;	Color, as defined with the color variables, can be turned on and off. 0 = switched on, >0 = switched off	vis1.dwFillFlags:=1; (the element becomes invisible)	- Cat. Colors: No color inside+ None Frame color - Cat. Color variables: FillFlags
dwFrameFlags: DWORD;	Frame presentation 0 full line 1 dashed (---) 2 dotted (...) 3 dash-dot (_._) 4 dash-dot-dot (_..) 8 hide line	vis1.FrameFlags:=1; (the frame is displayed dashed)	- Cat. Color variables: FrameFlags

The input of color values:

Sample: e1.dwFillColor := 16#00FF00FF;

A color is specified as a hexadecimal number resulting from the blue/green/red (RGB) components. The first two zeros after "16#" should be set to fill the DWORD size. 256 (0-255) colors are available for each color value

FF Blue part
00 Green part
FF Red part

Example for a blinking visualization element:

Define a global variable 'blink1' of type VisualObjectType in the configuration of a rectangle. In a program of function block the value of a component of the structure can be modified.

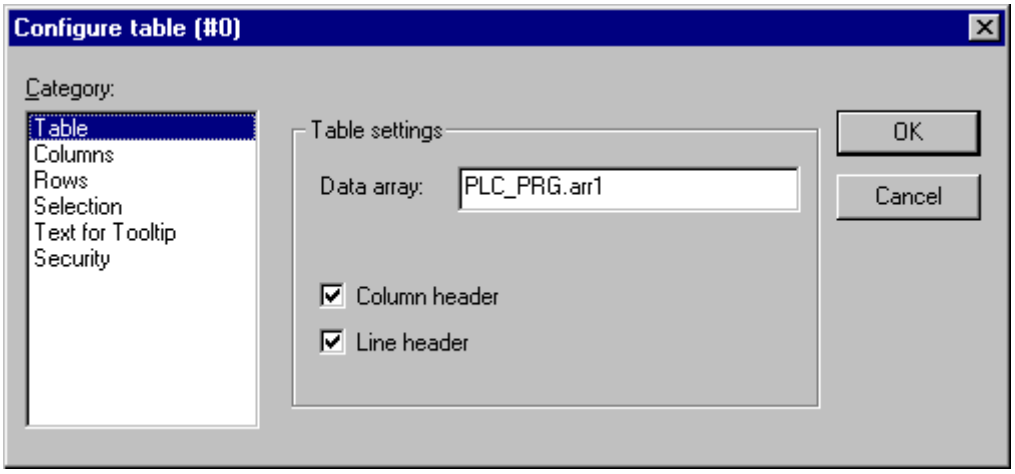
```
PROGRAM PLC_PRG
VAR
n:INT:=0;
bMod:BOOL:=TRUE;
END_VAR
(* Blinking Element *)
n:=n+1;
bMod:= (n MOD 20) > 10;
IF bMod THEN
```

```
    blinker.nFillColor := 16#00808080; (* grey *)
ELSE
    blinker.nFillColor := 16#00FF0000; (* blue *)
END_IF
```

10.1.4.2 Configure Table

As soon as a table is inserted for the purpose of visualization of an array, the dialog Configure Table will be openend. Besides the categories 'Tooltip' and 'Security' which are also available for other visualization elements, the following categories will be available for configuring display and contents of the table:

Category Table:



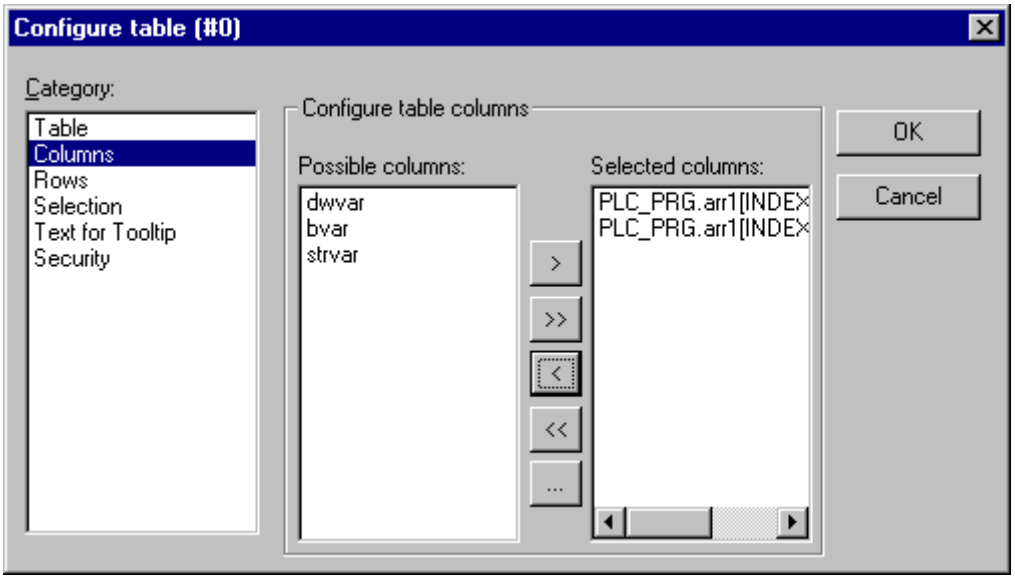
Dialog for the configuration of a Table, Category Table

Do the following table settings:

Data array: Insert the name of an array which should be visualized in the table. It is recommended to use the input assistant (<F2>) resp. the Intellisense function.

Column header, Line header: Activate these options if you want to get displayed the titles in the table. The line title reflects the array index (first column of the table), the column title can be defined in category 'Columns'.

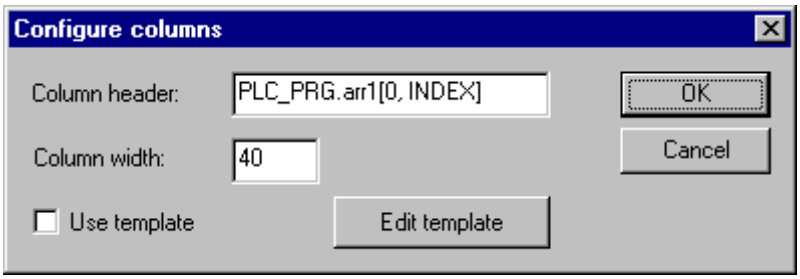
Category Columns:



Dialog for the configuration of a Table, Category Columns

Here you define the table elements. In the left window you get al list of all elements, which are handled in the array per index. In case of an array of a structure these would be the structure components.

Using the arrow button > you can transfer a selected component from the left window to the right window where you define the set of elements to be displayed in the table. Pressing button >> all elements will be transferred at a single blow. In the same manner you can remove elements from an already defined set (<, <<). In order to modify the default settings concerning the display of the table column for one of the elements, perform a double-click on the desired entry in the right part of the window, or press button '...' to open the dialog 'Configure columns':



Dialog for Configuring a Table, Category Columns, Column Properties

Editing the column header and the column width:

Initially the edit field **Column header** will contain an automatically created title (e.g. "PLC_PRG.arr1[INDEX].iNo" in case of an array of structure. for the column representing the structure component "iNo"). which you can change Further on the **Column width** (number of characters) can be set.

Editing configuration parameters for all elements of a column:

By default the table fields are displayed as simple **rectangles** and the entries are not editable. If you however activate the option **Use template**, a pre-defined set of parameters can be used, e.g. for the line width, the text input possibilities etc. To edit these parameters the common configuration dialogs as know for rectangles, bitmaps, buttons etc. are available, which you can access by a mouse-click on the button **Edit template**.

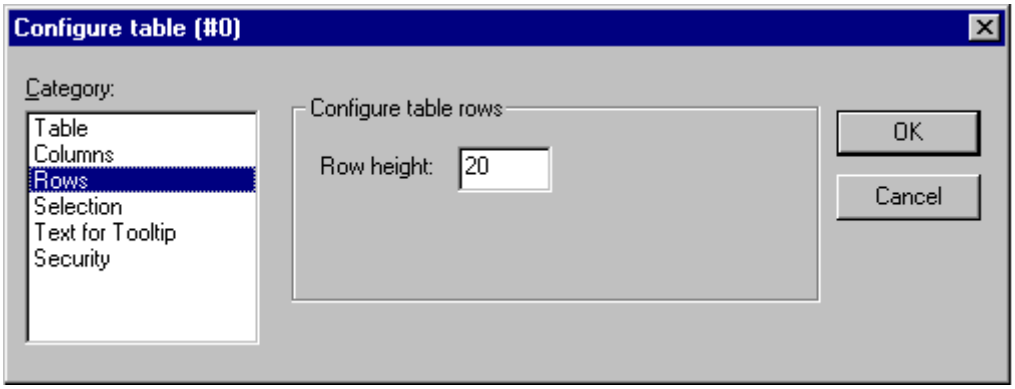
In order to define online actions for all entries of a column, i.e. for all array-elements of one array dimension, use the **placeholder [INDEX]** when entering the array name in the configuration dialogs (e.g. PLC_PRG.arr1[INDEX] like used in the default column header).

Example:

You want to visualize an array "arr1 [0..2] of BOOL" (table with 1 column) and you want, that in online mode by a mouse-click on a table cell the cell gets red-colored and the corresponding array element will be toggled and vice versa.
To reach this activate 'Use template' in the configuration dialog for the column and define the template as follows: Category 'Input', Action 'Toggle variable': "PLC_PRG.arr1[INDEX]. Category 'Colors': Alarm color red.
Category 'Variables', Action 'Change color': "PLC_PRG.arr1[INDEX].

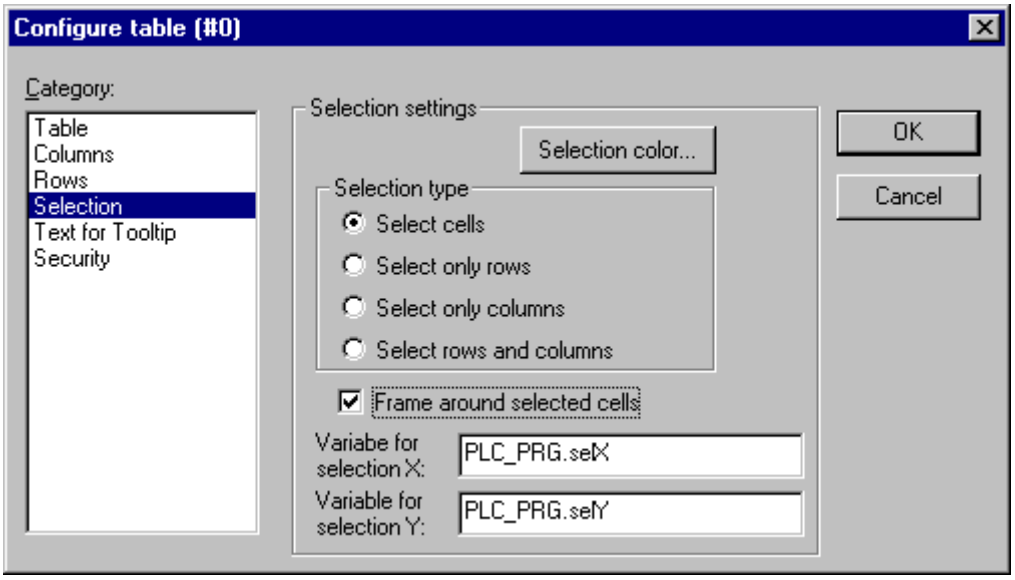
Category Rows:

Row height: Insert the desired height in number of pixels



Dialog for configuring a Table, Category Rows

Category Selection:



Dialog for configuring a Table, Category Selection

Here you can set the following parameters concerning the selection behaviour within the table:

Selection color: Press this button to get the standard color dialog for choosing a color for selected cells.

Selection type: Define which part of the table will be selected when you perform a mouse-click on one of the table fields in online mode:

Select single cells: Only the cell will be selected.

Select only rows: The whole line will be selected.

Select only columns: The whole column will be selected.

Select rows and columns: The whole column and line will be selected.

Frame around selected cells: A selected cell gets surrounded by a frame.

Variable for selection X, Variable for selection Y: Here you each can enter a project variable, which will indicate the X- resp. Y-Index of the selected table cell.

Example:

Create a table element visualizing the array of a structure:

Define the following structure

```
TYPE strucTab :
STRUCT
  iNo: INT;
  bDigi : BOOL;
  sText:STRING;
  byDummy: BYTE;
END_STRUCT
END_TYPE
```

In PLC_PRG define the following array:

```
arr1:ARRAY [1..5] OF strucTab;
```

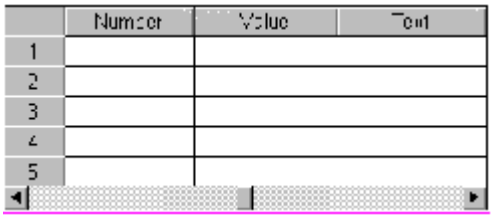
and the following variables:

```
selX:INT;
selY:INT;
```

Create a visualization object and insert a table element. Configure like follows:

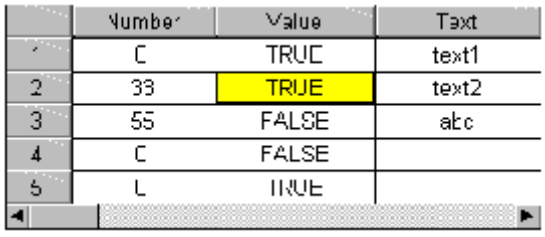
```
Cat. table: data array: PLC_PRG.arr1
```

Cat. Columns: (Close the dialog which will open with YES) – Transfer the components iNo, bDigi, sText to the right window - In the right window perform a double-click on the first entry (PLC_PRG.arr1[INDEX].iNo) and in the dialog which will open, replace the default title by "Number". Confirm with OK and also define new column titles for the other two entries (e.g. "Value" and "Text"). – In category 'Spec.Table' enter at 'Variable Selection X': "PLC_PRG.selX" and at Variable Selection "Y: PLC_PRG.selY". Activate option 'Frame around selected cells'. Press button 'Selection color' and choose color 'yellow'. Close the configuration dialog with OK. The table element now should be displayed as shown in the following:

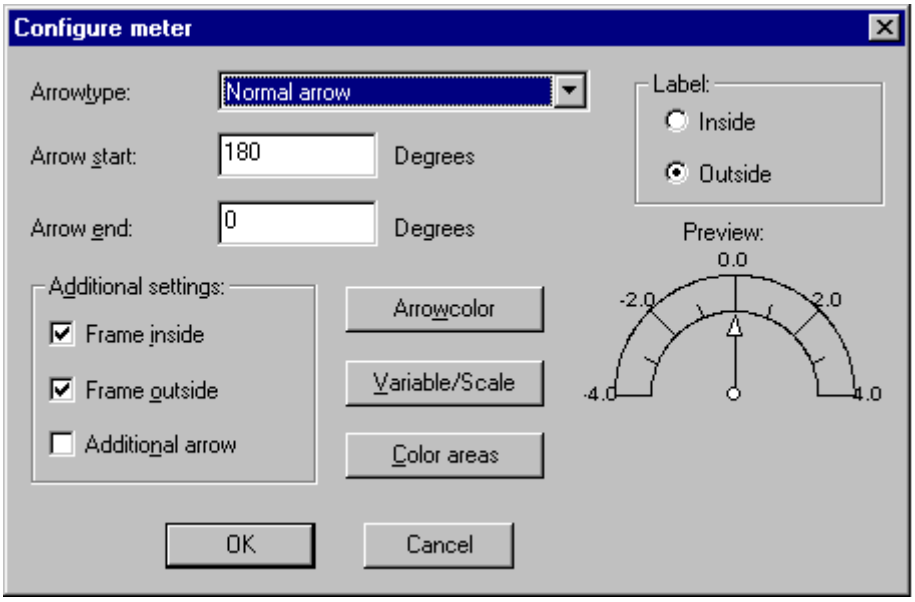


At the left border the numbers of the array index, at the top the titles of the selected structure components. You can modify the column widths by placing the cursor on the separator between two columns and moving the mouse as soon as the cursor appears as a horizontal double-arrow. .
In online mode the current values of the array elements will be displayed in the table cells. As soon as you select a table cell by a mouse-click, it will get yellow-colored and surrounded by a frame.

Example:



Meter



Dialog for the configuration of a Meter element

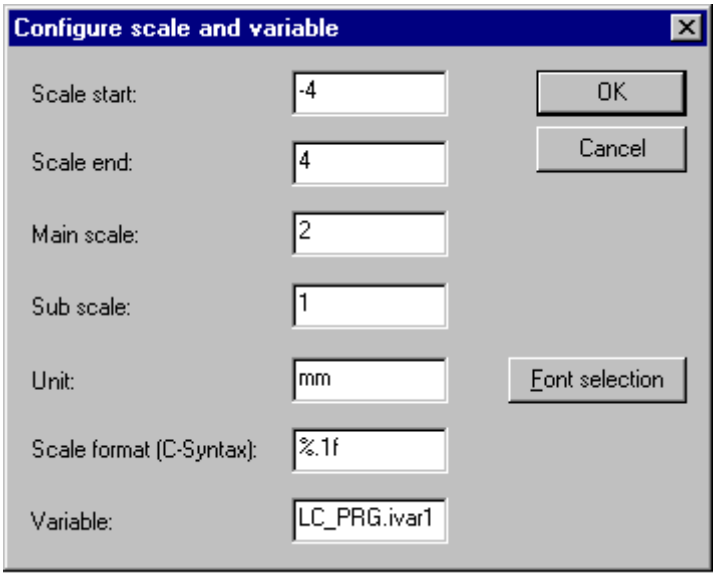
This dialog will open automatically as soon as you insert a Meter into a visualization object. A Preview is part of the dialog, immediately showing how the element will look as a result of the currently set parameters:

Arrowtype: Define the type of the arrow which will point at the current value on the Meter. Possible types: Normal arrow, thin arrow, Wide arrow, Thin needle.

Arrow start, Arrow end: Here you define the start and the end positions of the scale on a virtual circular arc in ° Degrees (angle). (Example: a Start angle of 180° and an End angle of 0° will define a upturned semicircle).

Arrow color: This button opens the standard dialog for choosing a color. Define the color of the pointer.

Variable/Scale: This button opens the dialog Configure scale and variable:



Dialog for Configuring the Scale and Variable for a Meter element

Scale start, Scale end: lowest and highest value on the scale, e.g. "-4" and "4".

Main scale: Define which intervals on the scale should be marked "with all", that means which should get a scale pitch and a label. If you insert e.g. "2", each second integer value will be indicated.

Sub scale: In addition to the main scale (Label + long pitch lines) here you can define a sub-scale which will be displayed as short pitch lines without any labels.

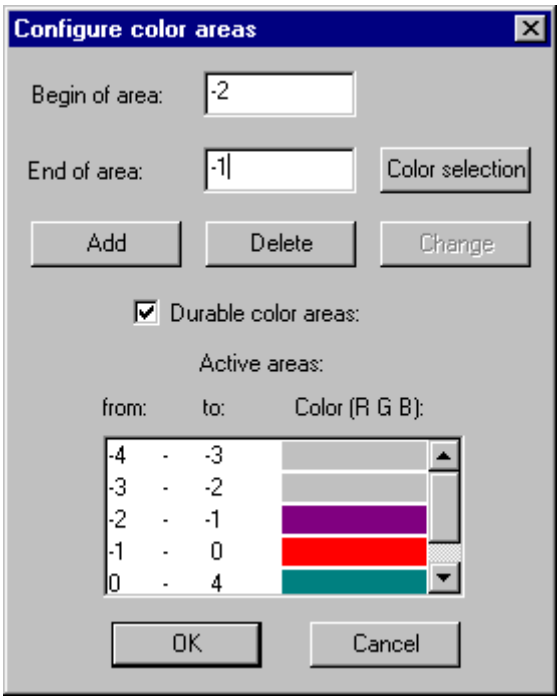
Unit: Define here the scale unit, e.g. "cm" or "sec". The unit is indicated by a label at the origin of the pointer.

Scale format (C-Syntax): According to the C-syntax you can define the display format of the scale labels; see the description concerning Category 'Text'.. **Example**: If you insert "%1.1f" the scale values will be indicated by a floating-point number with one decimal place before and one after the comma (e.g. "12.0")

Variable: Here you can define a variable which is assigned to the pointer position. (e.g. "PLC_PRG.posvar")

Font selection: This button will open the standard dialog for defining the font used in the Meter element.

Color areas: This button opens the dialog Configure color areas: Here you can define a separate color for each partition of the scale.



Dialog for the configuration of color areas for a Meter

Begin of area, End of area: Insert here the start and end values of the scale partition which should get the color defined in the following:

Color selection: This button opens the standard dialog for choosing a color. Confirm your selection with OK, which will close the dialog, and press button **Add**, whereupon the color and the assigned partition of the scale will be added to the window 'Active areas'. In order to remove an already defined area, select the entry and press **Delete**.

If the option **Durable color areas** is activated, the defined color ranges will be displayed permanently, otherwise in online mode just that partition of the scale will be colored which contains the current value of the respective value.

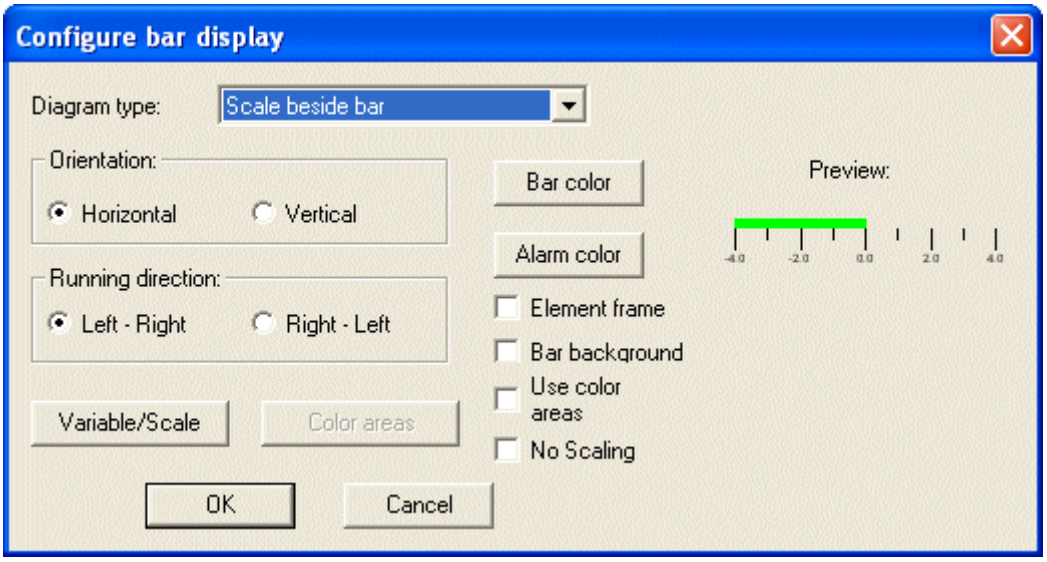
Label: Depending on which of the options is activated (**inside** or **outside**), the scale labels are placed at the inside or the outside of the circular arc of the scale.

Additional settings:

Frame inside, Frame outside: If one or both options is/are activated, an inner or outer frame will be added to the scale arc.

Additional arrow: In addition to the main pointer a little arrow will indicate the current value directly on the scale.

Bar Display



Dialog for the configuration of a Bar Display element

This dialog will be opened as soon as you insert a Meter element into a visualization object. A **Preview** is part of the dialog, immediately showing how the element will look as a result of the currently set parameters:

Diagram type: Choose one of the options: 'Scale beside bar', 'Scale inside bar' und 'Bar inside scale'.

Orientation: Define one of the options: Horizontal or Vertical bar.

Running direction: Choose whether the bar should be elongated corresponding to a growing value of the assigned variable in Left – Right or in Right – Left direction.

Bar color: This button opens the standard dialog for choosing a color. Define a color for the bar in normal state (no alarm). If option 'Use color areas' (see below) is activated, no entries are possible.

Alarm color: This button opens the dialog **Configure alarm**, where you define at which value the bar will be displayed in alarm color and which is the alarm color: Insert the desired limit value in the edit field and activate one of the **Conditions greater than** or l**ower than**, to define whether values higher or lower than the limit value should set off an alarm. Press button **Alarm color** to open the standard color dialog for choosing an alarm color. Close both dialogs with OK to confirm the settings and to return to the main dialog for configuring the bar display. If the option 'Use color ranges' (see below) is activated, no entries are possible.

Variable/Scale: This button opens the dialog **Configure scale and variable**, which corresponds to that used for the Meter element.

Element frame: If this option is activated a frame will enclose the bar display.

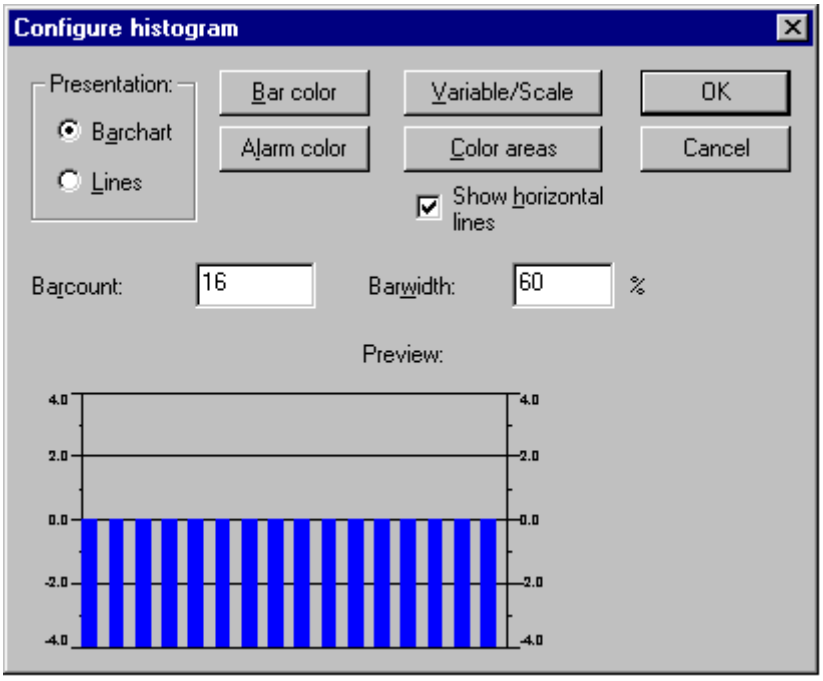
Bar background: If this option is activated, the whole display range will be indicated by a black bar in the background of the current values' bar, otherwise only the current values' bar will be displayed.

Use color areas: If this option is activated, any settings defined in the dialogs for 'Bar color' and 'Alarm color' (see above) will not be valid. In this case the color area definitions will be used, which have been made in the dialog 'Configure color areas'. This dialog can be opened by pressing button 'Color areas' (see below)

Color areas: This button opens the dialog Configure color areas where you can define a separate color for each partition of the scale. These definitions will only be valid if the option 'Use color areas' (see above) is activated. Use the dialog as described for the Meter element.

Histogram

A histogram element can be used to visualize an array. The values of the array elements will be represented by bars or lines side by side, indicating the current values of the element by their height.



Dialog for the configuration of a Histogram

The configuration dialog will be opened as soon as you insert a histogram element into a visualization object. A **Preview** is part of the dialog, immediately showing how the element will look as a result of the currently set parameters:

Presentation: Activate one of the options **Barchart** or **Lines**.

Show horizontal lines: If this option is activated, horizontal lines spanning the diagram will additionally display the scale gradation.

Alarm color: This button opens the dialog **Configure alarm**, where you define at which value the bar will be displayed in alarm color and which is the alarm color: Insert the desired threshold value in the edit field and activate one of the **Conditions greater than** or **less than**, to define whether values higher or lower than the limit value should set off an alarm. Press button **Alarm color** to open the standard color dialog for choosing an alarm color. Close both dialogs with OK to confirm the settings and to return to the main dialog for configuring the histogram.

Variable/Scale: This button opens the dialog **Configure scale and variable**, which can be filled like described for the Meter element.

Color areas: This button opens the dialog **Configure color areas**: Here you can define a separate color for each partition of the scale. See the description of the Meter where the same dialog is available.

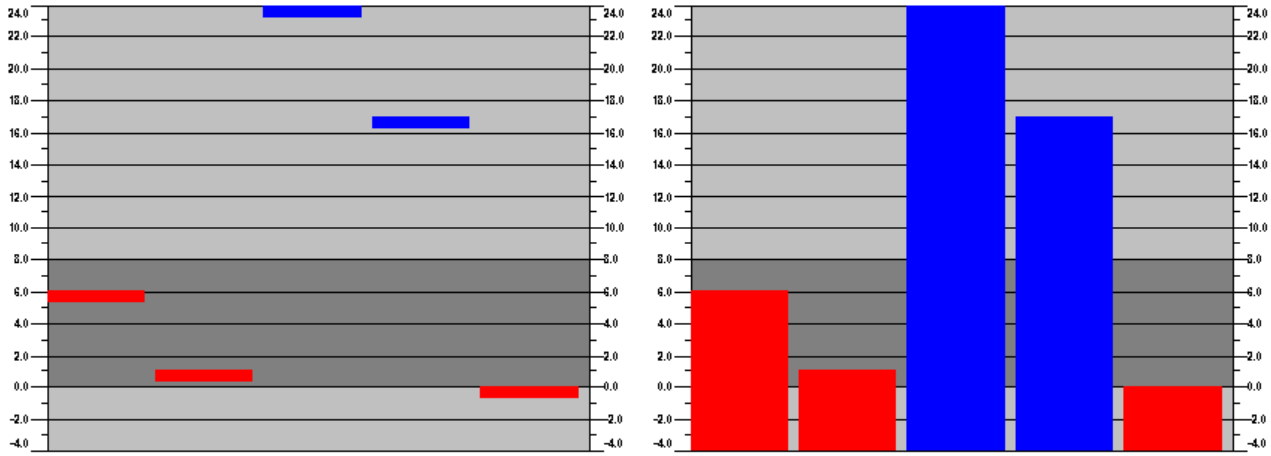
Bar color: This button opens the standard dialog for choosing a color. Define a color for the bar in normal state (no alarm).

Barcount: Insert the desired number of bars (e.g. corresponding to the number of elements of the array).

Barwidth: Define the width of the bars in percent by the total width available for one bar.

Example:

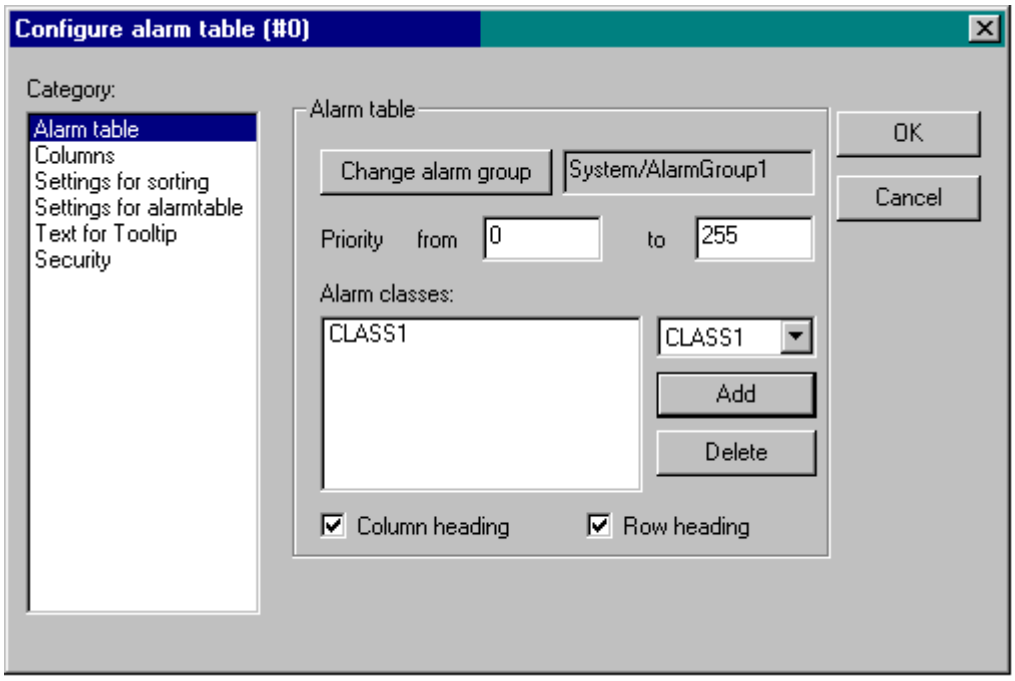
See in the following picture an example of the online display (bars resp. lines) of a histogram which represents an array arr1 [0..4] of INT. The number of bars appropriately was set to "5", the scale start to "–4", the scale end to "24", the main gradation was set to "2", the sub-gradation to "1" and the scale range 0 – 8 has got assigned another color (dark grey) than the rest of the scale range. Furtheron the bars should be displayed alarm-colored (blue) as soon as the value of the corresponding array element exceeds "8". You see the array elements arr1[2] and arr1[3] currently beeing in alarm state:



10.1.4.3 Configure Alarm table

The element 'Alarm table' is used to visualize alarms, which must be defined before in the TwinCAT PLC Control Alarm configuration.
As soon as the element gets inserted in the visualization object, the dialog 'Configure alarm table'. Besides the known categories for configuration of tooltip and security the following settings concerning display and selection in the table can be made:

Category Alarm table



Dialog for configuration of a alarm table, category Alarm table

Define what you want to get displayed in the alarm table:

Change alarm group: Press this button to get the selection tree of the alarm configuration, which offers all alarm groups currently defined. Choose the desired group (which even may contain just one alarm).

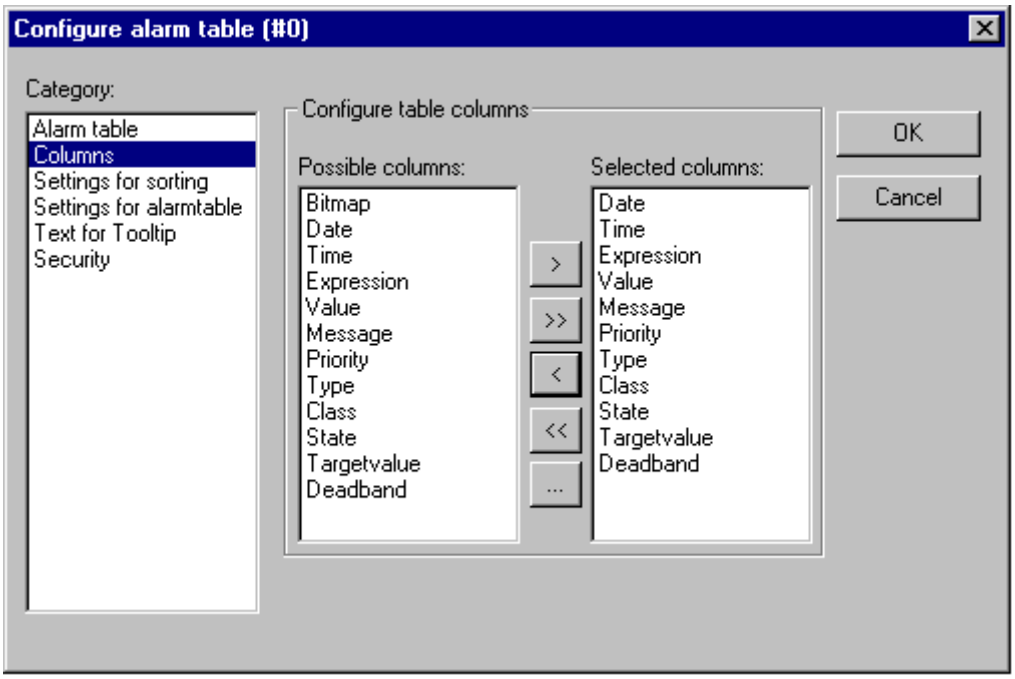
Priority: Define the priority for which you want to get displayed all alarms. Permissible range: 0 to 255.

Alarm classes: Mark a class which you want to get displayed and press button **Add** to add the class to the list in the window 'Alarm classes'. Do this for all required classes. In order to remove a marked entry from the alarm classes window press button **Delete**.

Activate options **Column heading** resp. **Row heading**, if the headings should be displayed in the alarm table.

Category Columns:

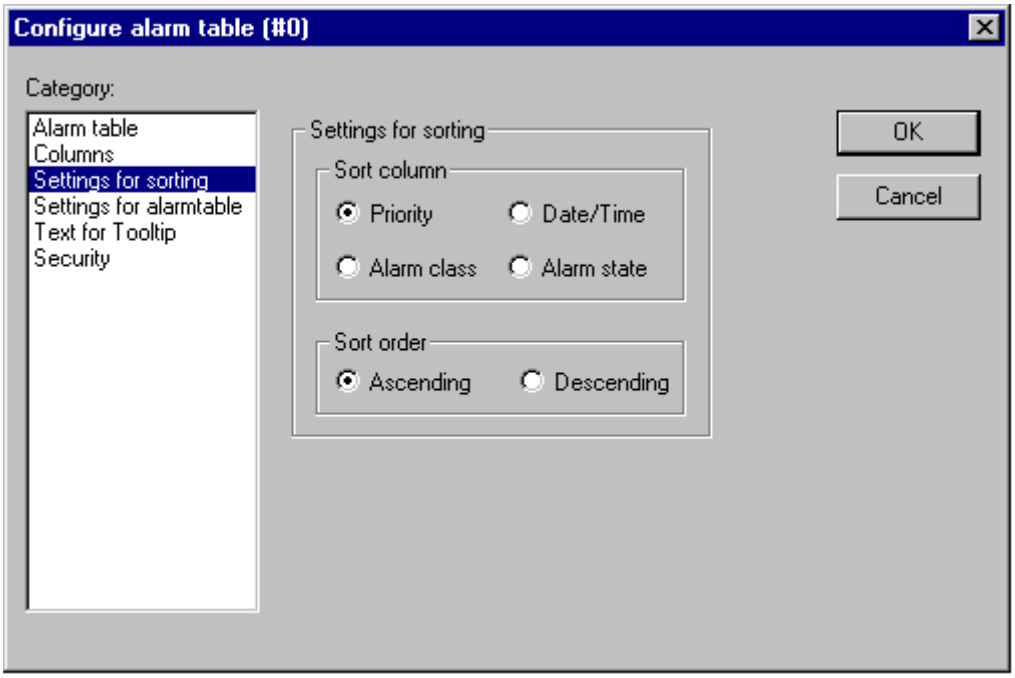
Define here, which of the columns (alarm parameters) should be displayed in the alarm table: The parameters are defined -- except for date and time (alarm is coming) and alarm state in the Configuration of the alarm groups: Bitmap, Date, Time, Expression, Value, Message, Priority, Type, Class, State, Target value (for alarm types DEV+ and DEV-), Deadband.



Dialog for configuration of an alarm table, category 'Columns'

Using the buttons ">", ">>", you can take single resp. all parameters from the left to the right window. The selection defined in the right window will be displayed in the alarm table. Using the buttons "<" resp. "<<" entries can be removed from the selection.
For each column you can open the dialog 'Configure columns' by a double-click on the entry in the right window. In this dialog Column header and Column width can be defined.

Category Settings for sorting:



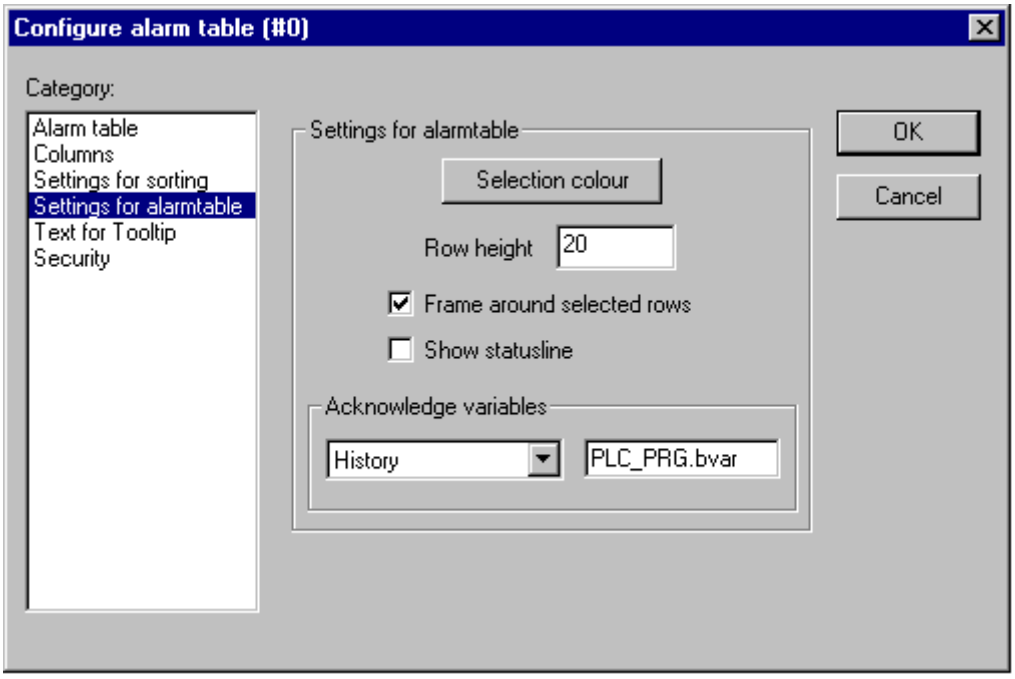
Dialog for configuration of an alarm table, category 'Settings for sorting'

Define here according to which criteria the alarm table should be sorted:

Sort column: Sorting according to Priority, Alarm class, Date/Time or Alarm state

Sort order: Ascending or Descending; Example: Ascending according to priority means that the table will start with alarms of priority 0 (if available), followed by higher numbered priorities.

Category Selection settings for alarm table:



Dialog for configuration of an alarm table, category 'Settings for alarmtable'

Define here some settings for the display for the chosen table fields:

Selection color: This button opens the standard dialog for choosing a color. Define the color in which selected fields should be displayed.

Row height: Height of the table rows in Pixel.

Slider size: Height of the slider (Pixel) at the bottom of the table.

Frame around selected rows: If this option is activated, selected table rows will get a frame.

Show statusline: If this option is activated, below the alarm table a status bar will be displayed providing the following buttons for the operation in online mode:

Acknowledge: All alarm entries marked in the alarm table get acknowledged.

Acknowledge all: All alarm entries listed in the alarm table get acknowledged.

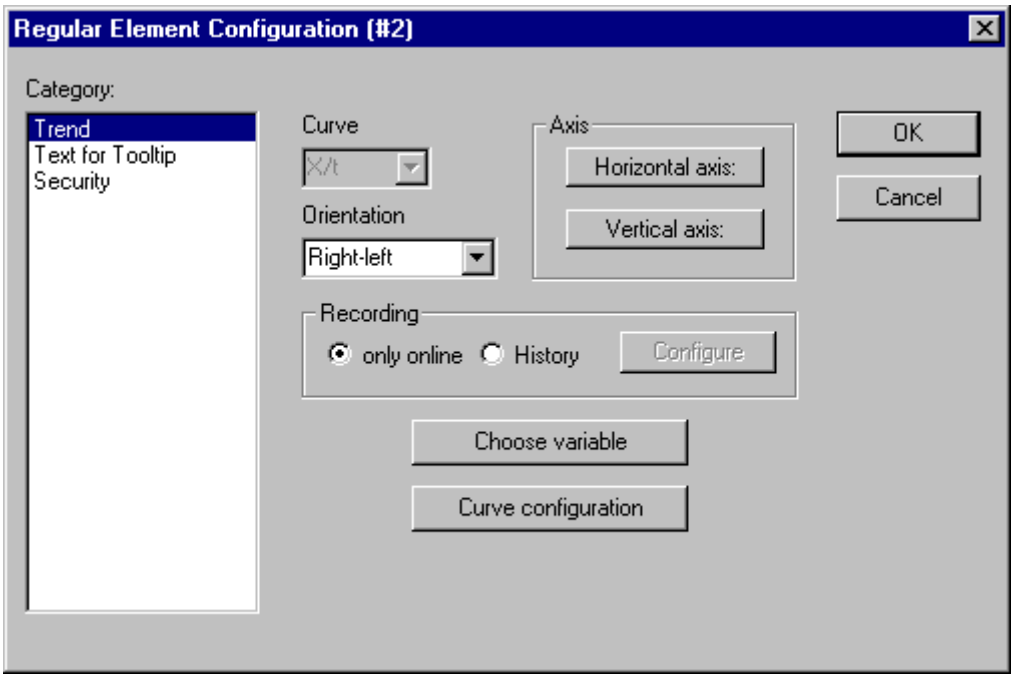
History: If this button is pressed, instead of the current status of the alarms the table will show a complete list of all events which have occured up to now (all transitions between any alarm stati). In this list no acknowledgement is possible ! Any new events will be added currently.
If you have defined a record file, also there you will find this history for all alarm classes, for which the action 'Save' has been activated.

Start: cancels **Stop** (see below)

Stop: The current update of the list with newly occurring events will be stopped until it is restarted by pressing button 'Start'.

Acknowledge variables: This option is only available as long as you have not chosen option 'Show statusline' (see above). If it is activated, the functions described above for the status string buttons can get controlled by variables. In order to define these variables, choose a function from the selection list and enter a project variable in the assigned edit field. Thus for example the acknowledgement of all alarms in online mode can be done by a rising edge of the assigned variable.

Trend



Dialog for configuration of a Trend element

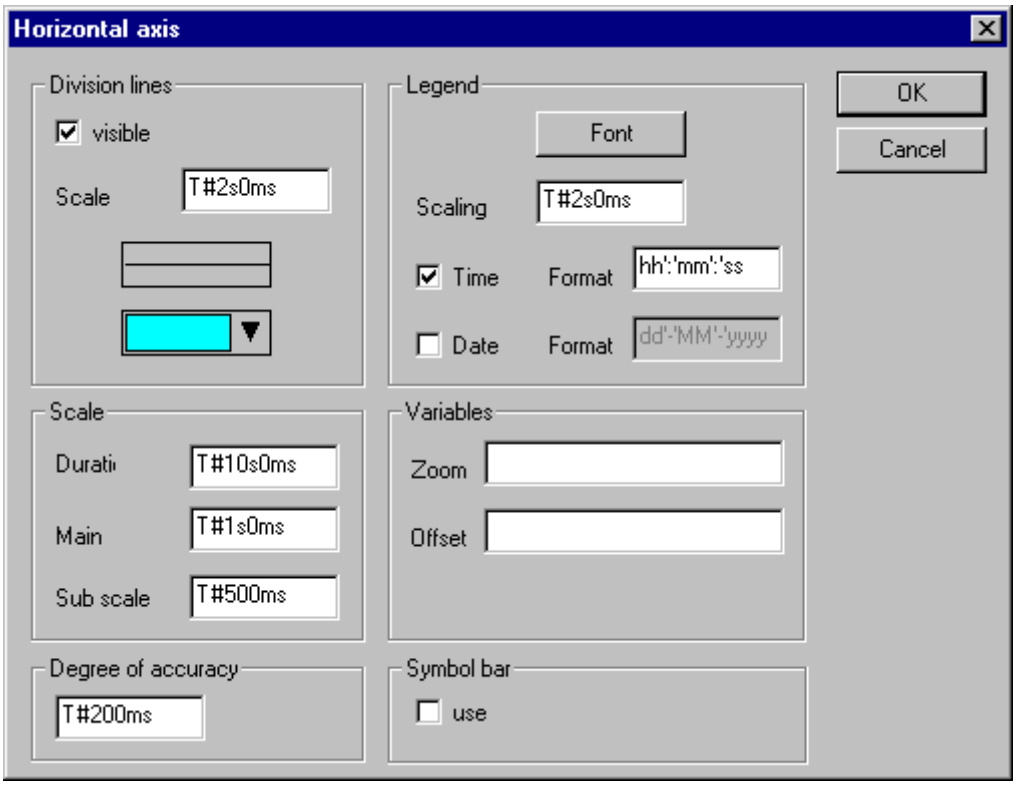
The Trend element can be used to log the time dependent behaviour of variable values in the online mode. It can be compared with the trace functionality..The online presentation is done in a diagram, in case of logging to a text file each of the values is written to a separate line.
In the dialog for configuring visualization elements in category 'Trend' you can do the following settings :

Curve: X/t, horizontal axis = time axis, vertical axis = scale of values

Orientation: Left-right or Right.-left: The latest value will be displayed on the left/right side

Axis:

Horizontal axis:



Dialog for configuration of the horizontal axis in the trend element

Division lines: Activate option 'visible', if vertical division lines should be displayed which are elongating the scale marks. In this case define the 'Scale': The given number defines the interval between the division lines on the horizontal axis. Type (normal ___, dashed _ _ _, dotted ....., dashdotted _ . _ .) and color of the lines can be defined in dialogs which will open when you perform a mouse-click on the corresponding rectangle showing the currently set line type resp. color.

Scale: The shown range of the scale is determined by the entry for Duration. If here e.g. " T#20s0ms" is defined, the scale will display a period of 20 seconds. The Main division and the Sub scale division, which will be displayed by the means of long and short marks are to be defined according to the same syntax.

Degree of accuracy: Define here (in the standard format for dates, e.g. T#50ms) the interval for displaying the current values of the variables.

Legend: Here you define the display of the legend. Via button **Font** the standard dialog for setting the font will be opened. At **Scaling** define the distances between the particular letterings on the scale (e.g. T#4ms, if the scale markings should get a lettering each 4 milliseconds. The lettering will contain the time and/or date, depending on which options are activated. The desired format each can be defined in the field 'format'.

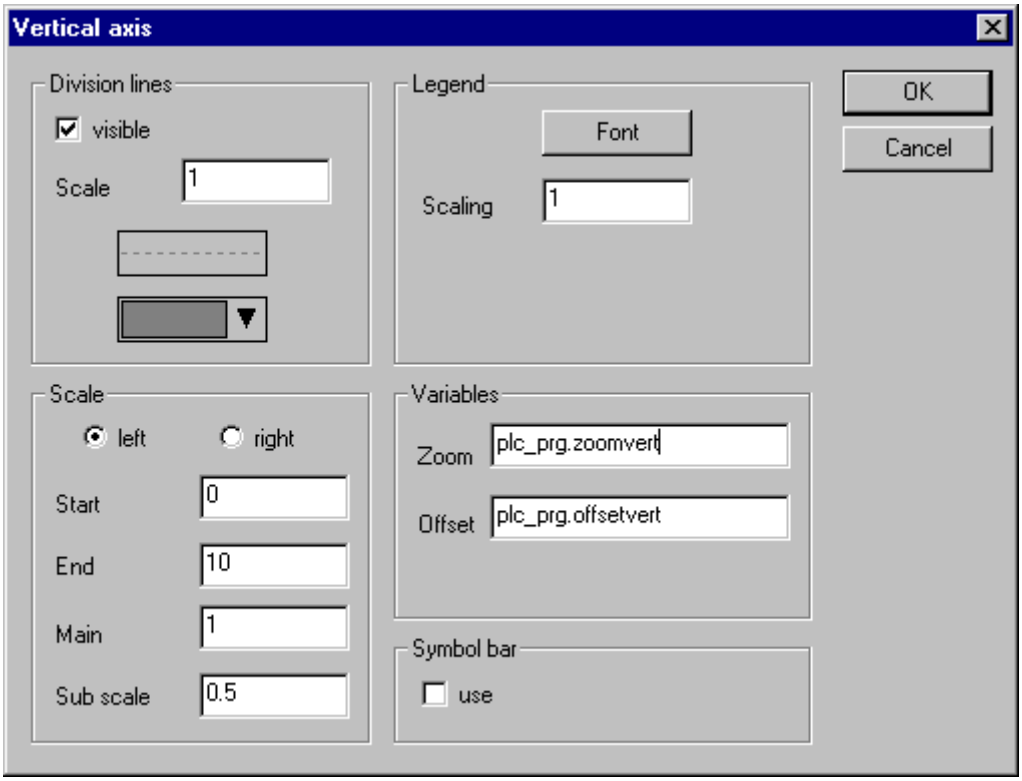
Variables: Here you can define project variables, which contain the zoom values resp. offset values for the horizontal scale. For example the offset of the display range of the horizontal axis will be set to "10" as soon as the variable assigned here gets value 10.

Symbol bar: If option use is activated, at the bottom of the element a horizontal symbol bar will be added, providing buttons for scrolling and zooming in online mode.



The simple arrow buttons will move the displayed range along the time axis step by step, the double arrow buttons will shift it to the end resp. start of the record. The zoom buttons allow a zooming of the horizontal scale step by step. To get a possibility to restore the original settings concerning zoom and offset, define the vertical symbol bar to get the 'home' symbol.

Vertical Axis



Dialog for configuration of the vertical axis in the trend element

Division lines: corresponding to the horizontal axis (see above)

Scale: Define whether the scale should be displayed at the left or right border of the trend diagram. Choose the Start value (lower end) and End value (upper end) of the scale as well as the Main and Sub scale divisions (longer and shorter markings will be displayed in the here defined distances).

Legend: Font and divisions; see above, horizontal axis

Variables: see above, horizontal axis

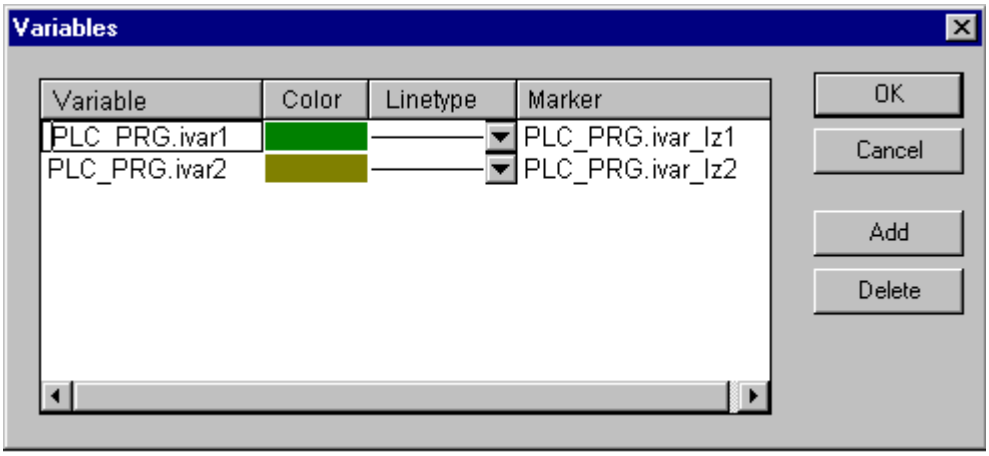
Symbol bar: see above, horizontal axis, additionally there is a "home" button  for restoring the standard settings concerning zooming and offset of the axes.

Recording: Define here whether the trend should be recorded 'only online', i.e. the time dependent behaviour of the variable values will be displayed using the chosen range of the scale, or whether the record should be saved to history file, which can be configured after pressing the button 'History'. The dialog corresponds to that which is used for the configuration of the alarm log file. In the log file for each time of measurement a separate line is written which contains the name and the values of all regarded variables. Each line starts with a unique identifier in DWORD format, which is built from the date of measuring.

Choose variable: Press this button to get the dialog Variables, where you can configure the variables for which the trend record should be done and how they should be displayed.

Enter a project variable in column **Variable** (mouse-click on the field will open an edit frame). It is recommended to use the input assistant <F2> or Intellisense function.

Color and **Line type** for the display of the variable in the record you can define by a mouse-click on the corresponding field in column Color (standard dialog for choosing a color) resp. by selecting a line type in the corresponding field of column Line type (normal ___, dashed _ _ _, dotted ....., dashdotted _ . _ .)

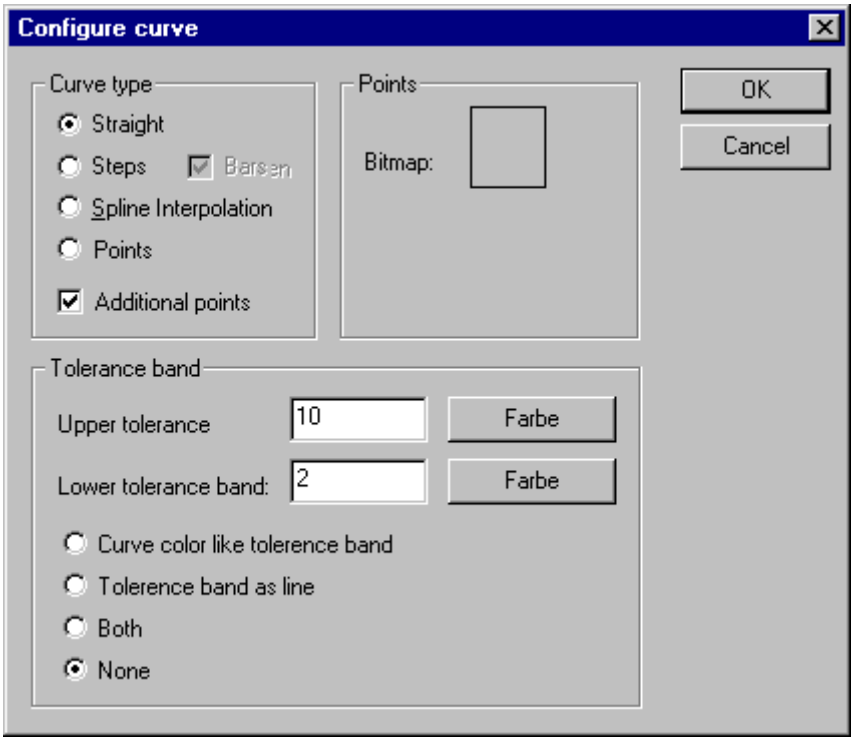


Dialog for doing the curve configuration for the trend element

In column **Marker** you can define a variable, which will provide the currently recorded value when you use the marker function in online mode. The marker will be displayed as a little grey triangle in the upper left corner of the diagram. If you click on the triangle and keep the mouse-button pressed, you can shift a vertical marker line along the horizontal time axis. The variable defined as 'marker' then will read the corresponding value from the record curve of the associated project variable.

Do the settings for all variables you want to record. Via button **Add** a further line will be added at the end of the list. A line can be deleted by button **Delete**.

Curve configuration: This button opens the dialog Curve configuration. Here some settings concerning the trend curves can be done:



Dialog for doing the curve configuration for the trend element

Curve type: Select one of the options Straight line, Steps or Points. For the first two types the display of **Additional points** can be defined. For displaying a point a bitmap can be defined, otherwise a filled rectangle (same color as curve) will be used as point symbol. Press the rectangle next to **Bitmap** to get the standard dialog for selecting a bitmap file. Via **Delete** the currently set bitmap can be removed from the configuration.

Tolerance band: You can define an upper and lower limit value on the vertical axis to be displayed as a tolerance band. For each band a color (Press the color rectangle to get the selection dialog) can be defined. If the bands should be displayed in online mode, activate option T**olerance band as line**. If you want the curve to get displayed in the color defined for the respective band as soon as exceeding the tolerance value, activate Curve color outside tolerance like tolerance band. Activate **Both** or **None** if you want to activate both or none of the above described display options at a time.

Example:

Display of a trend element in online mode:

1. Declaration in program PLC_PRG:

```
VAR
    n: INT;
    rSinus:REAL;
    rValue:REAL;
    rSlider1:REAL; (*für Lesezeichenfunktion*)
    rSlider2:REAL; (*für Lesezeichenfunktion*)
END_VAR
```

2. Program part of PLC_PRG:

```
n:=n+1;
rValue := rValue + 0.01;
rSinus:=SIN(rValue)*50 + 50;
IF n>100 THEN
n:=0;
END_IF
```

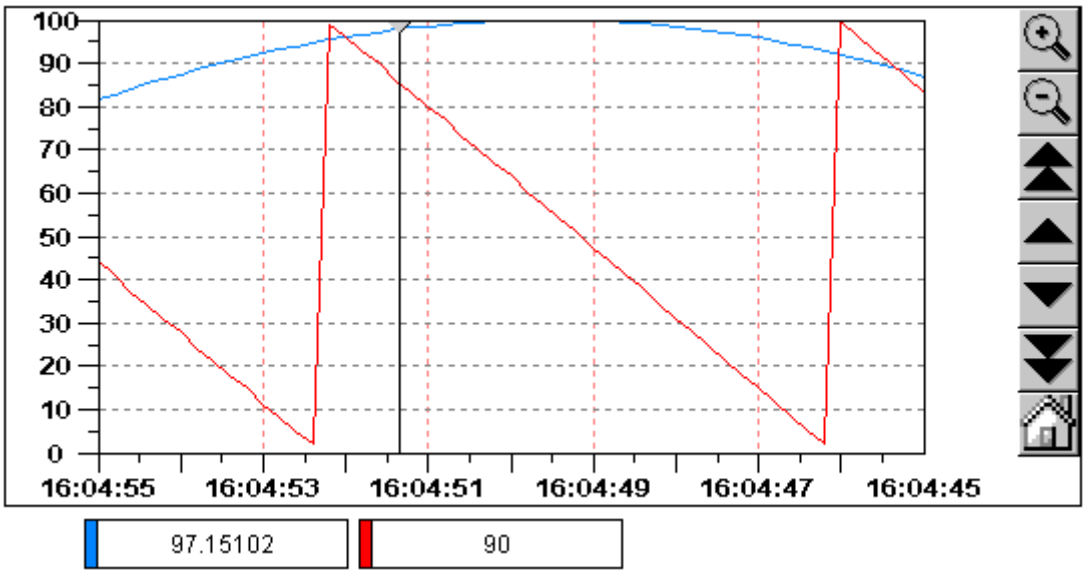
3. Configuration of a trend element in a visualization

Orientation Left-Right, History activated
Horizontal axis: Division lines: T#2s, Duration: T#10s, Main: T#1s, Sub scale: T#500ms, Degree if accuracy: T#200ms, Legend: Time Format ('hh':'mm':'ss'), Scaling T#2s. Symbol bar activated.
Vertical axis: Division lines visible, Scale: 10, dotted, grey; Scale left, Start: 0, End: 100, Main: 10, Sub scale: 5; Legend: 10; Symbol bar activated.
Variables:
1. Variable PLC_PRG.rsinus, blue line, Marker: PLC_PRG_TRD.rSlider1;
2. Variable PLC_PRG.n, red line, Marker: PLC_PRG_TRD.rSlider2
Curve configuration: Straight line, no tolerance band

Configuration of two display fields for the current record values provided by the marker variables:

Rectangle element 1: Category Text: insert "%s" in the Content field; Category Variables: insert in field Textdisplay: PLC_PRG.rSlider1
Rectangle element 2: Category Text: insert "%s" in the Content field; Category Variables: insert in field Textdisplay: PLC_PRG.rSlider2
(additionally insert a rectangle element at the left border of the rectangle elements 1 and 2, showing the curve color of the corresponding record variable)

Result in online mode after login and start of the program:



The record is running from the left to the right; the latest value is shown on the leftmost position; every 200 milliseconds the current value will be added to the display. The arrow buttons in the symbol bars allow shifting the displayed time range. Using the simple arrow buttons you can shift step by step, using the double arrows you get to the end resp. start of the record. For example: if you go to the start of the record by pressing the double arrows pointing to the left, you get a still display of the former values. If you then move the marker (grey triangle in the upper left corner) along the time axis, you can read the exact values of each of the both recorded variables for each time in the rectangle elements below the diagram.

Bitmap

You can enter the options for a bitmap in the **Bitmap** category within the visualization element configuration dialog box.

Enter the bitmap file and its path in the **Bitmap** field. You can use the ... button to open the standard Windows Browse dialog box from which you can select the desired bitmap.

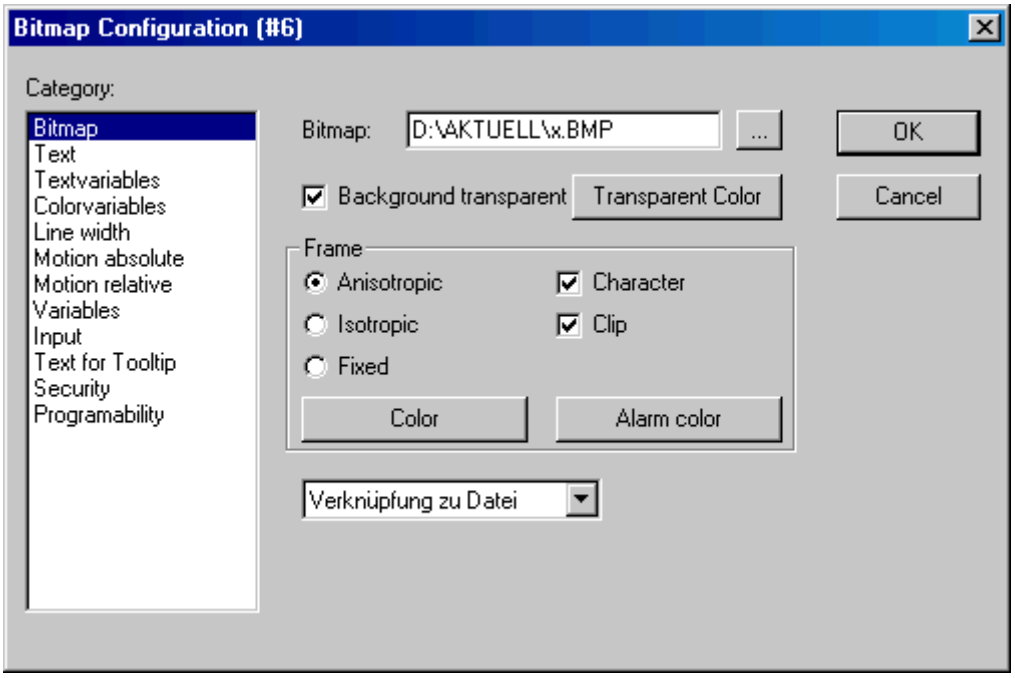
All other entries affect the **frame** of the bitmap.

By selecting **Anisotropic, Isotropic or Fixed** you specify how the bitmap should react to changes in the size of the frame. **Anisotropic** means that the bitmap remains the same size as the frame which allows you to change the height and width of the bitmap independently. **Isotropic** means that the bitmap retains the same proportions even if the overall size is changed (i.e., the relationship between height and width is maintained). If **Fixed** is selected, the original size of the bitmap will be maintained regardless of the size of the frame.

If the **Clip** option is selected together with the **Fixed** setting, only that portion of the bitmap that is contained within the frame will be displayed.

If you select the **Draw** option, the frame will be displayed in the color selected in the **Color** and **Alarm** color buttons in the **color** dialog boxes. The alarm color will only be used if the variable in the **Change Color** field in the **Variable** category is TRUE.

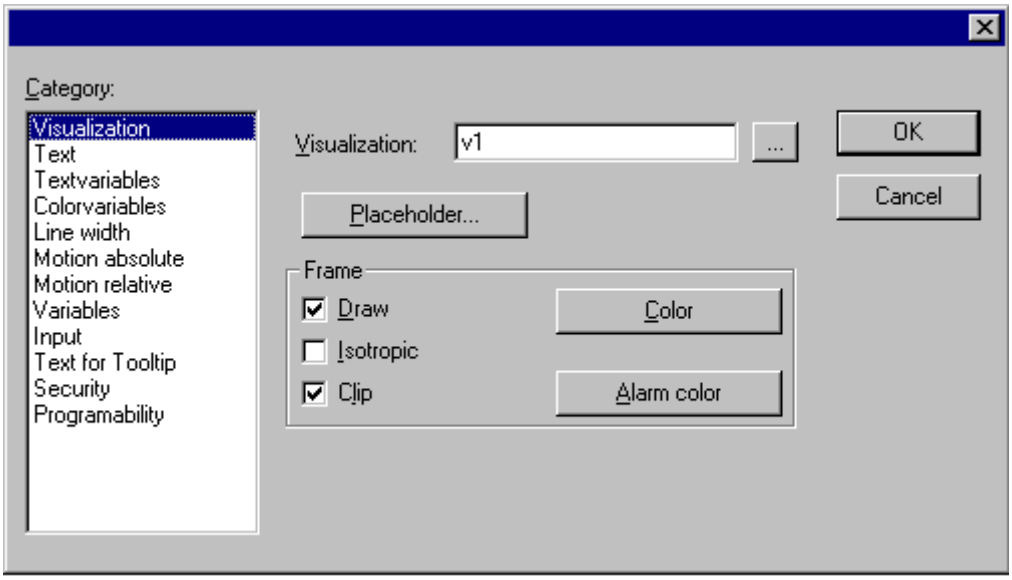
In the selection list in the lower part of the dialog you can define whether the bitmap should be inserted in the project (Insert) or whether just a link to an external bitmap-file (path as entered above in the 'Bitmap' field) should be created (**Link to file**). It is reasonable to keep the bitmap file in the project directory, because then you can enter a relative path. Otherwise you would enter an absolute path and this might cause problems in case you want to transfer the project to another working environment.



Visualization Element Configuration Dialog Box (Bitmap Category)

Visualization

When you insert a visualization as an element in another visualization, you are creating a "reference" of the visualization.
The configuration of this reference can be done in the Visualization category within the visualization element configuration dialog box.



Visualization Element Configuration Dialog Box (Visualization Category)

Enter the object name for the visualization, which should be inserted, in the Visualization field. Use the ... button to open a dialog box containing the visualizations available in this project. Any visualization may be used with the exception of the current one.

The following entries affect the visualization **frame**.

- If you select the **Draw** option, the frame will be displayed in the color selected in the Color and Alarm color buttons in the color dialog boxes. The **alarm color** will only be used if the variable in the **Change Color** field in the **Variables** category is TRUE.

- If **Isotropic** is selected, the proportions of the visualization will be maintained even if the size changes (i.e., the relationship between height and width will remain the same). Otherwise the proportions can be changed.
- If the **Clip** option is selected in Online mode, only the original portion of the visualization will be displayed. For example, if an object extends beyond the original display area, it will be clipped and may disappear from view completely in the visualization.

The **Placeholder** button leads to the 'Replace placeholder' dialog. It lists in the 'Placeholder' column all the placeholders which had been inserted in the configuration dialogs of the "mother"-visualization and offers in the 'Replacements' column the possibility of replacing these for the current reference with a definite value. Which replacements are possible in a given case depends on whether a value group was predefined in the 'Extras' 'Placeholder list' dialog in the "mother"-visualization. If this is the case, it will be displayed in a combo box for selection. If nothing was pre-defined, double clicking on the corresponding field in the Replacements column opens an editing field which can be filled in as desired.

A further possibility for replacing placeholders in references occurs directly when you define the call of a visualization by an entry into the **Zoom to vis.** option field in the configuration dialog ('Input' category).



No control of the chronological sequence of replacements is possible! Therefore no placeholders should be replaced with text that also contains placeholders!



When using placeholders it is no longer possible to check for invalid entries in the configuration of the visualization element immediately upon compilation of the project. Hence the appropriate error messages are first issued in Online mode (...Invalid Watch expression..).

NOTICE

Online behaviour of a visualization reference:

If you insert a visualization and then select and configure this reference, it will be regarded as a single object and in online mode will react to inputs correspondingly to its configuration. In contrast: if you do not configure the reference, then in online mode its particular visualization elements will react exactly like those of the original visualization

Example of an application of the placeholder concept:

Instances of a function block can easily be displayed with the help of references of the same visualization. For example, in configuring the visualization visu, which visualizes the variables of function block, one could begin each variable entry with the placeholder FUB (e.g. FUB.a). If a reference from visu is then used (by inserting visu in another visualization or by calling via 'Zoom to vis.'), then in the configuration of this reference the placeholder FUB can then be replaced with the name of the function block instance to be visualized.

This might look like shown in the following:

In the project define a function block containing the following declarations:

```
FUNCTION_BLOCK fu
VAR_INPUT
    changecol : BOOL; (* should cause a color change in the visualization *`)
END_VAR
```

In PLC_PRG define two instances of 'fu':

```
inst1_fu : fu;
inst2_fu : fu;
```

Create a visualization object 'visu'. Insert an element and open the configuration dialog, category 'Variables'. Enter in field 'Change color' the following: "FUB.changecol". Open category 'Input' and enter in field 'Tap Variable' "FUB.changecol". Open category 'Text' and enter"FUB - change color ".

Create another visualization object 'visu1'.

Insert visualization 'visu' twice in 'visu1' (two references of 'visu').
Mark the first reference of 'visu' and open the configuration dialog of category 'Visualization'. Press button 'Placeholder', so that the placeholder list will be displayed. There replace entry 'FUB' by 'PLC_PRG.inst_1'.

Now mark the second reference of 'visu' and (like described for the first one) replace 'FUB' by 'PLC_PRG.inst_2'.
Now in online mode the values of the variables which are used to configure the two instances of 'fu' will be visualized in the corresponding reference of 'visu'.

Of course the placeholder FUB can be used at all places in the configuration of 'visu' where variables or text strings are entered.

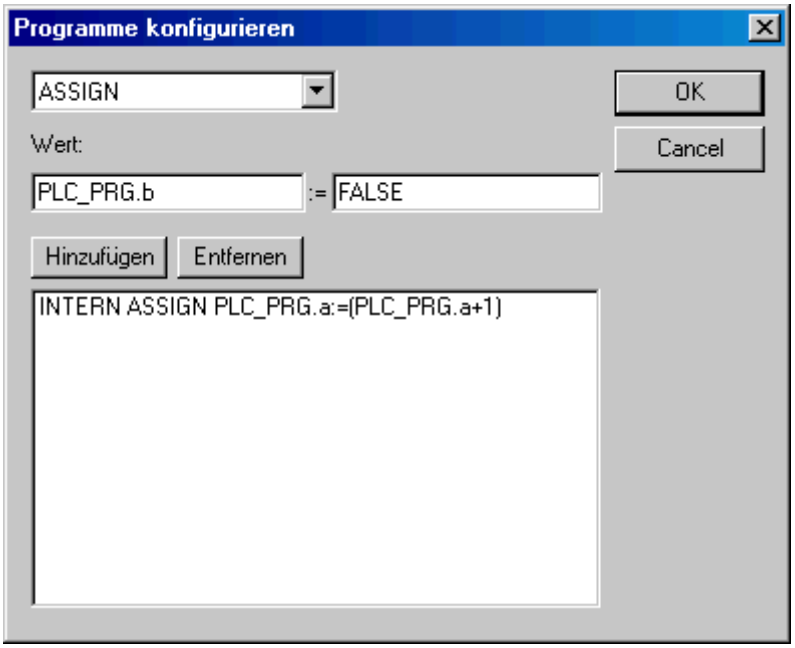
10.1.4.4 Special input possibilities for operating versions

The TwinCAT visualization can target specifically be used with TwinCAT Visualization or as Target-Visualization as a mere operating interface. Then no menus and status and tool bars will be available to the user and no possibility to modify the code.

Thus, when a visualization is created with TwinCAT for the purpose of being used as a 'operating version' the principal control and monitoring functions in a project must be assigned to visualization elements thus making them accessible via mouse click or keyboard in Online mode. See in the following some special input possibilities to configure visualization elements for the purpose of being used in TwinCAT Visualization. They are available in the configuration dialog for a visualization element:

Enter internal commands in the field **Execute program** in the category **Input** according to the following syntax

INTERN <COMMAND>[PARAMETER]*



The following table shows the available internal commands. Some of them expect to receive several parameters, which are then entered separated by spaces. Optional parameters are enclosed in square brackets. For those commands which require that a Watch list be specified, a placeholder can be used instead of the direct name. If you enter several commands for one element, these are separated by commas.

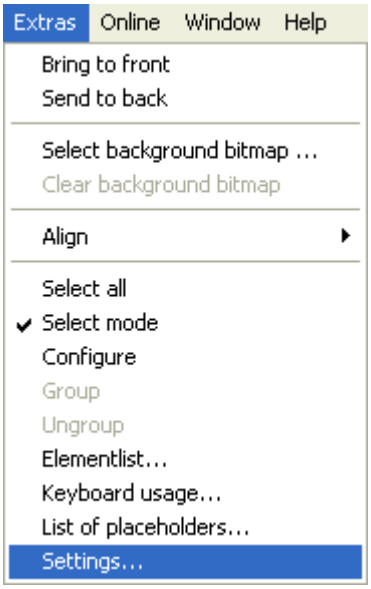
Requirements

INTERN Command	The equivalent in TwinCAT PLC Control	Explanation
ASSIGN <Variable>:=<Expression>	Assignment	A variable or expression gets assigned to another variable. Example: INTERN ASSIGN PLC_PRG.ivar1:=PROG1.ivar+12; INTERN ASSIGN PLC_PRG.bvar:=FALSE;
PROGRAM <Pfad ausführbares Programm> [Pfad aufzurufender Datei]	Program call	The program will be executed. Example: INTERN PROGRAM C:\programms\notepad.exe text.txt
LANGUAGEDIALOG	Visualization settings	The dialog for visualization settings which includes the category language gets opened.
LANGUAGE <Sprachenangabe, wie in eingestellter Sprachdatei *.xml, *.vis, *.tlt oder *.txt verwendet>	Visualization settings Language	The desired language is set without using the dialog for visualization settings See Language switching in the visualization [▶ 309]!
LANGUAGE DEFAULT	Visualization settings Language	For dynamic texts the default language will be used, which is defined in the currently included xml-file [▶ 314] .
CHANGEUSERLEVEL		A dialog for setting the user group level will open. The eight TwinCAT PLC Control user group levels are offered for selection.
CHANGEPASSWORD	cp. 'Project' 'User Group Passwords...'	A dialog for changing the user group password will appear.
TRACE[1]	Ressourcen, Traceaufzeichnung	The window for trace recording (Sampling Trace) will be opened. The menu commands Trace **Start, Read, Stop, Save, Load** which are available in the full version of **TwinCAT PLC Control** are available in this window.
SAVEPROJECT[1]	'File' 'Save'	The program will be saved.
EXITPROGRAM[1]	'File' 'Close'	The program will be finished.
PRINT[1]	'File' 'Print'	The current visualization will be printed out online.

[1] Not supported for Target-Visualization.

10.1.4.5 Configure Visualization Object

Besides the configuration of the individual visualization elements also the visualization object on the whole can get configured. This is possible concerning the settings for frame, language, grid, background etc. as well as the assignment of special hotkey definitions (keyboard usage), which should be valid for exactly one visualization object.



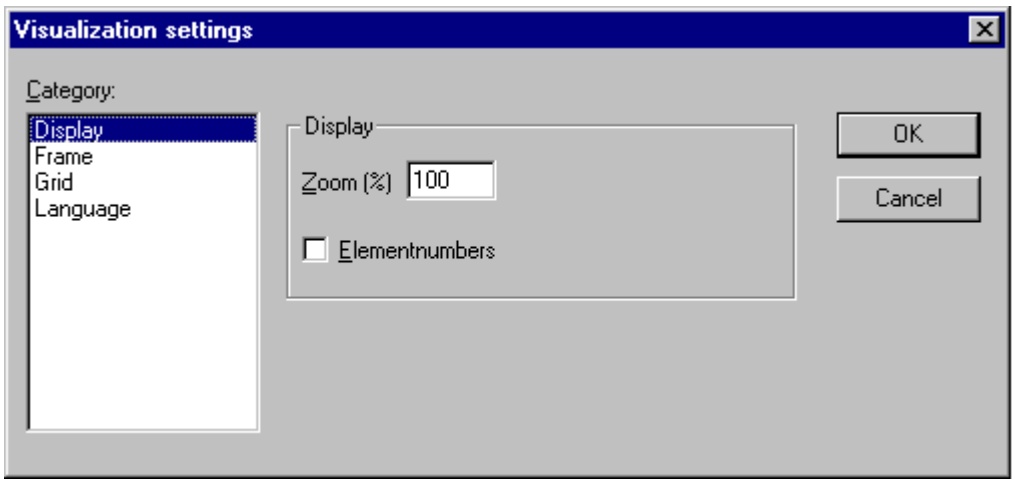
'Extras' 'Settings' / Display, Frame, Grid, Language

When this command is used, a dialog box will open in which you can make certain settings that affect the visualization.

The categories Display, Frame and Language also can be edited in the online mode.

Category Display :

Enter a zoom factor into the field Zoom of between 10 and 500 % in order to increase or decrease the size of the visualization display.



Setting dialog for visualizations (Category Display)

Category Frame:

If **Auto-scrolling** is selected, the visible portion of the visualization window will move automatically when you reach the edge while drawing or moving a visualization element. If **Best fit in Online mode** is selected, the entire visualization including all elements will be shown in the window in Online mode regardless of the size of the window. When **Include Background Bitmap** is selected, the background bitmap will be fitted into the window as well, otherwise only the elements will be considered.

Category Grid:

Define here whether the grid points are **visible** in the offline mode, whereby the spacing between the visible points is at least 10 even if the entered size is smaller than that. In this case the grid points only appear with a spacing which is a multiple of the entered size. Selecting **Active** causes the elements to be placed on the snap grid points when they are drawn and moved. The spacing of the grid points is set in the field Size.

Language category:

Here you can specify in which national language the text that you have assigned to an element in the Text and Text for tooltip options in the configuration dialog should be displayed. Statically, this can be done by using a **language file**. The function **'Dynamic texts'** alternatively offers the possibility to let the content of a text display change depending on a project variable. See chapter Language selection [▶ 309]



The text display changes only in Online mode!

'Extras' 'Select Background Bitmap'

Use this command to open the dialog box for selecting files. Select a file with the extension "*.bmp". The selected bitmap will then appear as the background in your visualization.
The bitmap can be removed with the command '**Extras**' '**Clear Background Bitmap**'.

'Extras' 'Clear Background Bitmap'

Use this command to remove the bitmap as the background for the current visualization.
You can use the command '**Extras' 'Select Background Bitmap**' to select a bitmap for the current visualization.

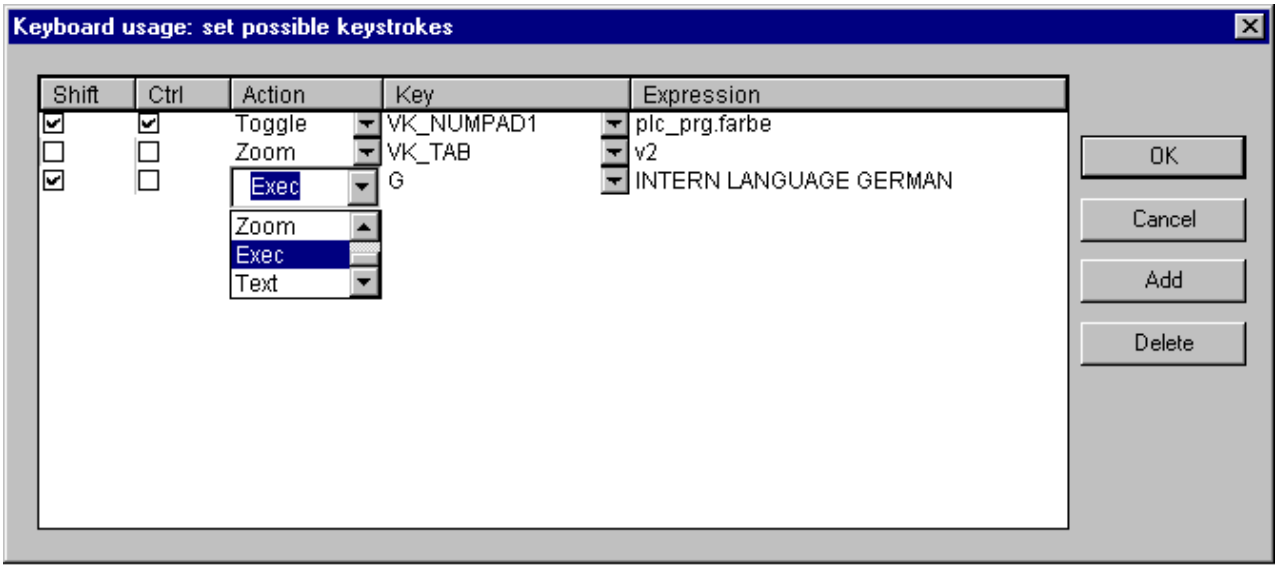
'Extras' 'Keyboard usage'

The use of hotkeys can optimize the pure keyboard operation of a visualization.

In the configuration of a visualization object you can define hotkeys which will cause actions like visualization elements do. For example you could define that – if visualization 'xy' is active – in online mode the hotkey <Strg><F2> will stop the program, which also will happen as soon as element 'z' of visu 'xy' gets an input (by mouse-click or via touch screen).

Anyway per default the keys <Tabulator> <Space> <Enter> will work in that way that in online mode each element of a visualization can be selected and activated.

The dialog 'Keyboard usage: set possible keystrokes' can be called in the menu 'Extras' or in the context menu:



Dialog 'Keyboard usage: set possible keystrokes'

In column Key a selection list offers the following keys to which an action can get assigned:

Shortcut	Meaning
VK_TAB	Tab-Key
VK_RETURN	Enter-Key
VK_SPACE	Space-Key
VK_ESCAPE	Esc-Key
VK_INSERT	Insert.-Key
VK_DELETE	Delete-Key
VK_HOME	Pos1-Key
VK_END	End-Key
VK_PRIOR	Picture () Button
VK_NEXT	Picture (¯) Button
VK_LEFT	Arrow-Key (¬)
VK_RIGHT	Arrow-Key (®)
VK_UP	Arrow-Key()
VK_DOWN	Arrow-Key(¯)
VK_F1-VK_F12	Function keys F1 to F12
0-9	Keys 0 to 9
A-Z	Keys A to Z
VK_NUMPAD0 - VK_NUMPAD9	Keys 0 to 9 of the numeric keypad
VK_MULTIPLY	Key* of the numeric keypad
VK_ADD	Key+ of the numeric keypad
VK_SUBTRACT	Key- of the numeric keypad
VK_DIVIDE	Key□ of the numeric keypad

In the columns **Shift** and **Ctrl** you can add the <Shift>- and/or the <Ctrl>-key to the already chosen key, so that a key combination will result.

In column **Action** you define what should happen as soon as the key (combination) will be pressed. Select the desired action from the list and insert an appropriate expression. See in the following the available actions and valid expressions, corresponding to those which can be set in the configuration dialog of category 'Input':

Action	Meaning	Expression
Toggle	Toggle variable	Variable, e.g. "plc_prg.tvar"
Tap true	Tap variable (set to TRUE)	Program variable, e.G. "plc_prg.svar"
Tap false	Tap variable (set to FALSE)	Program variable, e,G. "plc_prg.xvar"
Zoom	Zoom to Vis.	Name of the visualization object to which you want to jump, e.g. "Visu1"
Exec	Execute Program	Name of the executable file, e.g. "notepad C:\help.txt" (Notepad will start and open the file help.txt)
Text	Text input of variable 'Textdisplay'	Number of the element for which the text input is to be configured, e.g. "#2" (Display of element numbers can be switched on in 'Extras' 'Settings'; also see 'Element list...')

In column **Expression** you must enter – depending on the type of action – a variable name, a INTERN-command, a visualization name of a text string, exactly like you would do in the configuration dialog of category 'Input' for the corresponding visualization element.

Use button **Add** to add another empty line at the end of the table. Use the **Delete** button to remove the line where the cursor is positioned currently. **OK** resp. **Cancel** will save resp. not save the done settings and close the dialog.

The keyboard usage can be configured separately for each visualization object. Thus the same key (combination) can start different actions in different visualization.

Example:

The following key configurations have been done for the visualizations VIS_1 and VIS_2:

VIS_1:				
Shift	Ctrl	Action	Key	Expression
x		Toggle	A	PLC_PRG.automatic
	x	Zoom	Z	VIS_2

VIS_2:				
Shift	Ctrl	Action	Key	Expression
		Exec	E	INTERN LANGUAGE DEUTSCH
	x	Zoom	Z	TC_VISU

If you now go online and set the focus to VIS_1, then pressing <Shift><A> will cause that variable PLC_PRG.automatic will be toggled. <Ctrl><Z> will cause a jump from Visu1 to VIS_2.

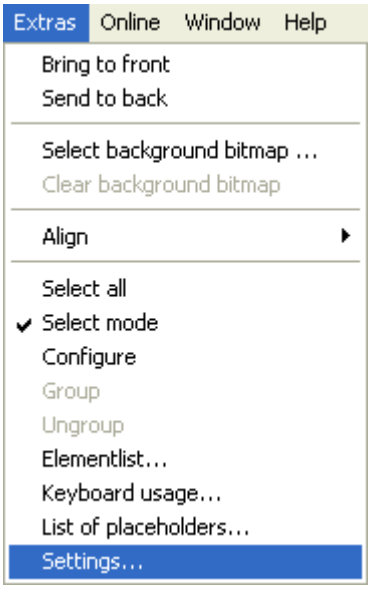
If VIS_2 is the active window, pressing key <E> will cause that the language within the visualization will switch to German. <Ctrl><Z> here will cause a jump to visualization TC_VISU.

10.2 Language switching in the Visualization

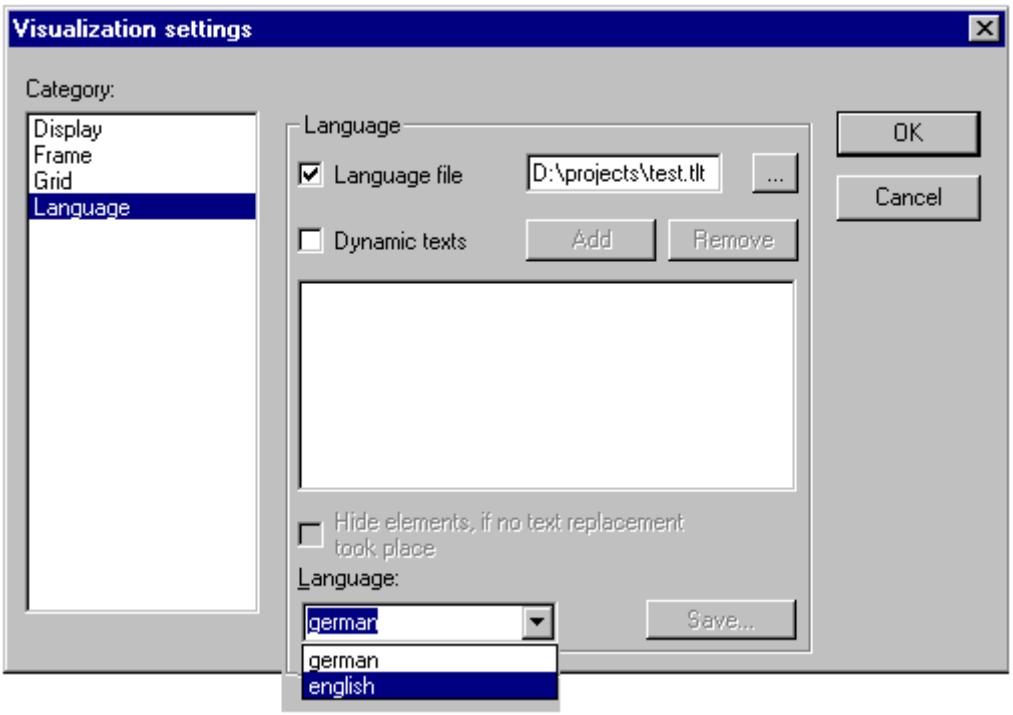
The language switching for texts in a visualization can be done via static or via dynamic texts, which must be provided by a file. Unicode format is only possible with dynamic texts.

How can a language switching be done:

In the configuration dialog ‘Visualization settings’



in the selection list below **Language** you can choose one of the languages defined in the currently used language file, which should be used as **start language** in online mode; for the example shown below: german and english.



A **language switch** in **online mode** is done via a visualization element. For this purpose the internal commands “INTERN LANGUAGE <language> and “INTERN LANGUAGEDIALOG” are available (see chapter “Special input possibilities for operating versions” [▶ 304]), which can be used in the configuration dialog in category ‘Input’.

Example :

You can insert a button element which can be used to switch the visualization texts to German. For this purpose label the element with ‘German’, in configuration category ‘Input’ activate option ‘Execute program’ and define a command “INTERN LANGUAGE <language>”. “language” is to be replaced by the language shortcut used in the language file, thus for the vis-file example: “INTERN LANGUAGE german”. If the button will be operated in online mode the visualization texts will be displayed according to the entries which are

available for “german” in the language file.

10.2.1 Static

10.2.1.1 Static language switching

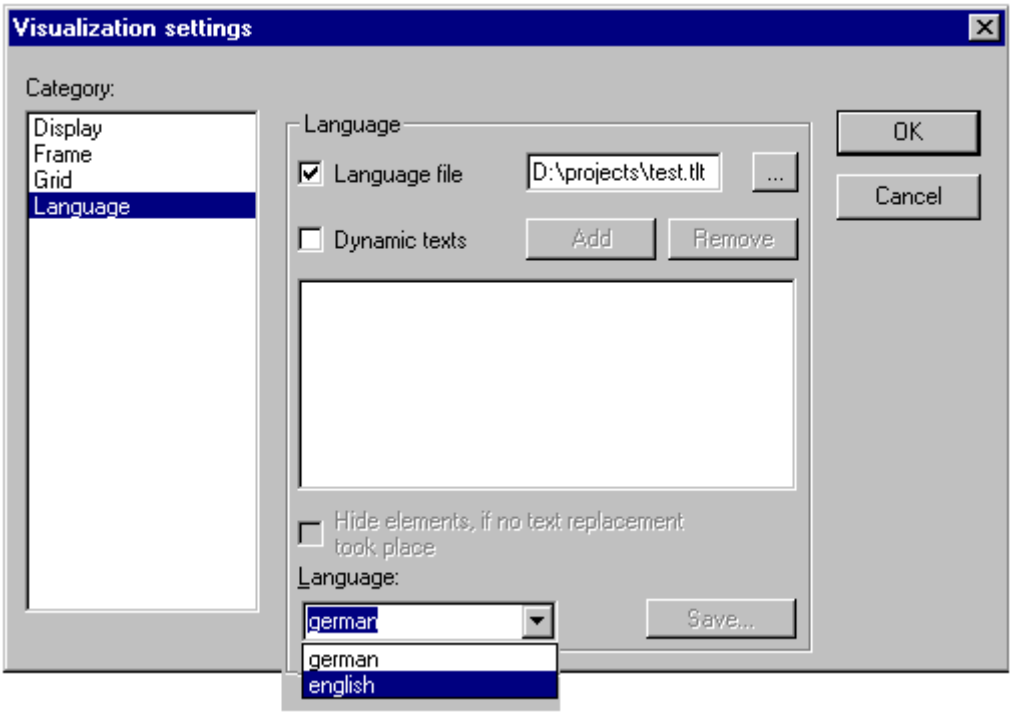
For a static switch of the language a language file (*.vis, *.tlt, *.txt) file can be used (for how to create see below). The difference to the dynamic language switching is that the language cannot be defined by a project variable during run time.

NOTICE

For visualizations it is recommended to use the *.**vis** language file, because *.tlt- resp. *.txt-translation files only work for visualizations in TwinCAT PLC Control and also in those not for the Meter, Bar Display and Histogram elements.

In dialog ‘Visualization settings’, you configure, which language file should be used with the project. In order to choose a translation (*.tlt, *.txt) or a pure visualization language file (*.vis), which contains the texts in the various languages, activate the **Language file** option and in the input field next to it enter the appropriate file path. Via the button  you get the standard dialog for opening a file.

Dialog for selection of a language file for a visualization



Regarding creating a **translation file** *.**tlt** or *.**txt,** please see 'Project' [▶ 66] 'Translate into other languages' [▶ 76].

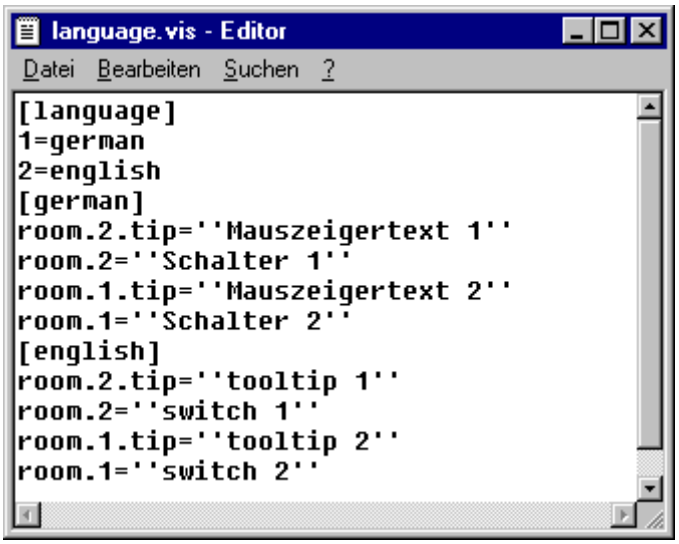
For creating a special **language file *.vis** see the following steps:

Open likewise the Visualization settings dialog, Language category. Choose option **Language file**. In the associate input field enter where you want to store the file. The extension is .vis. You also can use the dialog '**Open file**' by pressing the button . If a language file with the extension .vis is already present, it will be offered to you here.

In the input field next to **Language** you fill in a keyword for the language which is currently used in the visualization, i.e."german" (or "D") then press the button **Save**. A file with the extension .vis will be created, which now can be edited by a normal text editor.

For example you can open the file by NOTEPAD:

You get a list of the text variables for the language currently used in the visualization. It includes a reference to the title of this list, for example "1=german" as reference to the title [german]. You can extend the list by copying all lines, then replacing the German by English text and setting a new title [english]. Beyond the line 1=german you accordingly have to add 2=english.
Example of a language file for a visualisation (Category Language)



10.2.2 Dynamic

10.2.2.1 Dynamic Language switching

Dynamic texts allow switching between different, always language-assigned text versions for a visualization element. The difference to static texts is that the definite text selection also can be done via a variable used in the application.

In the configuration of the element a Prefix-ID-combination is entered, which is assigned to a text in a XML-file (also named “text list” in the following). The ID can be defined by a project variable.

Example of application: The ID represents an error number, as Prefix e.g. “Error” is used. The language file provides via the Prefix-ID-combination an appropriate error message, which – depending on the currently set language – will be displayed in this language.

Please regard:

- The language files for dynamic texts can be created in Unicode (UTF-16) or ANSI (ISO-8859-1).
- For the Target-Visualization the start language, the directory for the xml-files to be used and a list of xml-files can be defined by the target system. This allows to modify these parameters later without the need of creating a new boot project. Thus, in an easy way existing text lists can be modified (start language, texts) resp. new languages can be added. If the target system is providing such a configuration, the text lists which are defined for the visualization in TwinCAT PLC Control, will not be regarded in online mode! If no target-specific configuration is available for the language switching, then after a modification of the text lists defined in TwinCAT PLC Control a project download must be done.

10.2.2.2 Configuration

You can control dynamically which text will be displayed in a visualization element in online mode by using prefix-ID-combinations, each pointing to another text defined in a XML-file. Prefix and ID must be defined in the configuration dialogs of the visualization element, where the ID can be dynamically defined by using a

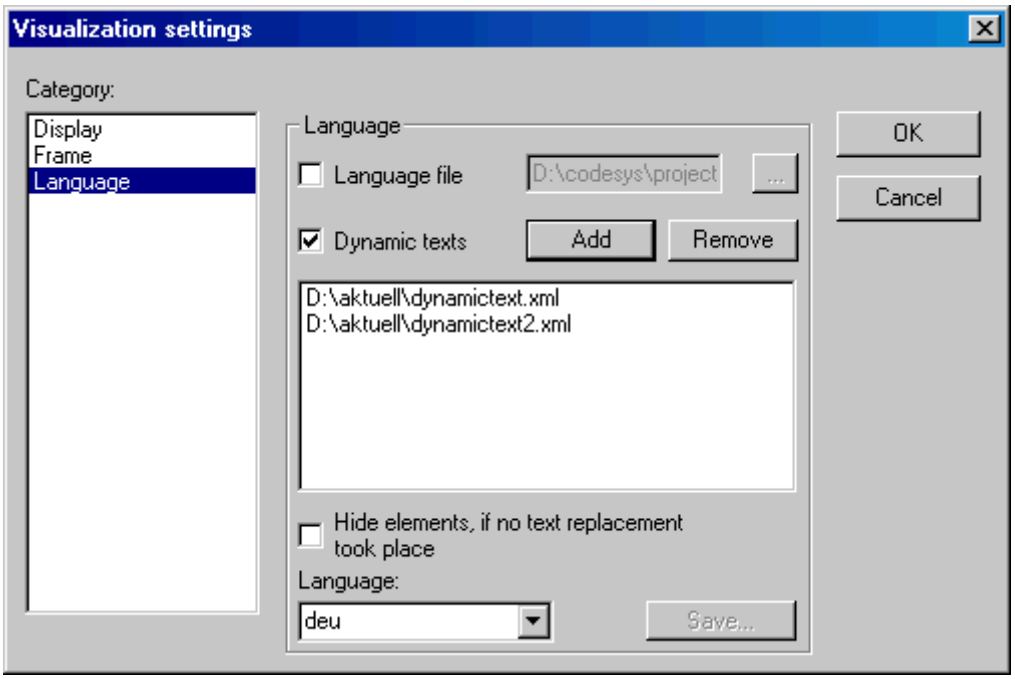
project variable. The XML-file describing the assignment to a text, must be linked to the project. ((see "XML file for dynamic Texts) [▶ 314]). This is done in the visualization settings dialog. The XML-file must have a certain format, the particular texts are listed below the according Prefix-ID-combination and a language code is added. Thus the text display not only can switch between different contents but also between different languages. The desired language can be selected as described above for the Language file in the 'Settings' dialog, category 'Language':

So in order to get a dynamic text display, the following entries have to be made in the different configuration dialogs of a visualization

Link the XML-file(s): Dialog 'Settings' category Language:

Activate option Dynamic texts and press button Add, in order to link one or several XML-files, which are available on your system, to the project. The selected files will be listed in the window below the button. Press button Delete if you want to remove a selected file from the list. If you want to get displayed just those visualization elements, for which a dynamic text replacement is done, then activate option Hide elements, if no text replacement took place.

Selecting one of the language identifiers offered in the selection list at field Language will cause the display of that text versions (for the corresponding prefix-ID-combination) which are marked with that language identifier in the XML.-file.



Configuration dialog Settings, category Language, for dynamic texts

Define the ID in configuration dialog 'Variables' in field 'Textdisplay':

Enter here a value (number) resp. a project variable which should define the ID of a text (as used in the XML-file).

Define the text format in configuration dialog 'Text':

In the Content field, insert a placeholder "%<PREFIX>" at that position of the text, where you want to get displayed a dynamic text in online mode. Instead of "PREFIX" you can enter any desired string. See the description for the 'Text' configuration dialog.

For each prefix-ID-combination, which is found in a linked XML-file, the assigned text will be displayed in the visualization element in online mode. If no appropriate entry is found, no replacement will be done.

10.2.2.3 XML File for dynamic Texts

For a description how to use dynamic texts in the visualization see 'Settings', category Language. The underlying file must be available in XML format (<file name>.xml). In this file texts are assigned to identifiers (which are a combination of a prefix and a ID). These prefix-ID combinations can be entered in the configuration of a visualization element.

In the header section of the file a default language and a default font assigned to a language can be defined.

The descriptions in the xml-file are enclosed by tag <dynamic-text> and <\dynamic_text> which have to be entered at the beginning resp. end of the file.

The language files for dynamic texts can be created in Unicode (UTF-16) or ANSI (ISO-8859-1). This is to be defined via the encoding syntax at the beginning of the xml-file.

See a file example below.



Primary formats of the xml-file, which do not use the <dynamic_text>\<\dynamic_text> tags or the header section, will be supported further on!

The **Target-Visualization** offers an interface for scanning the entries of dynamic textlists. Thus those can be used directly in the program.

The header section starts with **<header>** and is closed with **<\header>**. If you want to define a default language, use entry **<default-language>**. A default font which is assigned to a certain language, can be defined via entry <**default-font>**. These entries are optional. If they are missing, the dynamic text in the visualization will be displayed according to the local configuration settings of the visualization.

<header>	
<default-language> <language> </default-language>	Default language; that means that if there is no text entry available for the currently set language, that text will be used which is found within the same text entry for the default language. If also for the default language no text is found, "<PREFIX> < ID>“ will be displayed. f multiple XML-files are used, thus providing multiple headers, only that header section will be regarded, which is read at last. So it is reasonable to use only one header section! The language token must correspond which one of that used in the text entries (see below). Note: In online mode the default language can be set explicitly via a visualization element configured with command INTERN LANGUAGE DEFAULT in category Input, Execute program (see Chapter Special input possibilities for operating versions).
<default-font> <language><language></language>	Default font for <language>: The given font (e.g. "Arial" will automatically be used for all elements, which display dynamic texts in <language>. The language token must correspond which one of that used in the text entries (see below).
</default-font>	
<default-font> <language>... ... <default-font> ...	further default fonts for other languages
</header>	

The **list of assignments** Prefix-ID-text starts with **<text list>** and ends with **</text list>**.

The particular text entries each start with **<text prefix>** and end with **<\text>**. A text entry which is assigned to a Prefix-ID-combination must contain the following lines:

<text prefix>= "<PREFIX> id="<ID>"	"PREFIX" corresponds to the <PREFIX> used in the visualization element configuration (category Text); "ID" corresponds to the entry in category 'Variables', Textdisplay
<language> <!CDATA[<TEXT>]] </language>	Use any string as a 'language' identifier (e.g. "english"). This identifier then will be displayed in the 'Settings' dialog, category Language of the visualization element in the selection list at 'Language'; instead of "TEXT" insert any text which then will be displayed instead of the above defined ID-prefix-combination in the visualization element.
</text>	

For each prefix-ID-combination at least for one language a text entry must be available. E.g. see in the file example shown below: <deutsch> indicates the start of the german version of a text, </deutsch> terminates the text.
Dynamic texts on the one hand can serve to display texts in different languages, but of course on the other hand they also can be used to change the content of a text (same language) display dynamically.

Example:

You want to have two visualization elements, one for visualizing the current machine identification, the other for visualizing an error message according to a currently given error number:
(1) Define in PLC_PRG the following variables: ivar of type INT, defining the current machine identification; errnum of type INT defining the current error number.
(2) Configure a visualization element for displaying the current machine identification:
a. Enter in category ‚Text' in the text field: "%<Maschine>"
b. Enter in category ‚Variables' at Textdisplay“: „PLC_PRG.ivar”
(3) Configure another visualization element for displaying the error message for the currently occurred error:
a. Enter in category ‚Text' in the text field: "%<Error>"
b. Enter in category ‚Variables' at Textdisplay“: "PLC_PRG.errnum“
(4) Create a xml-file, e.g. with name dynamictextsample.xml, according to the syntax described above, which should look as follows for the current example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<dynamic-text>
<header>
    <default-language>deutsch</default-language>
    <default-font>
        <language>deutsch</language>
        <font-name> Arial </font-name>
        <font-color>0,0,0</font-color>
        <font-height>-13</font-height>
        <font-weight>700</font-weight>
        <font-italic>false</font-italic>
        <font-underline>false</font-underline>
        <font-strike-out>false</font-strike-out>
        <font-char-set>0</font-char-set>
    </default-font>
    <default-font>
        <language>english</language>
        <font-name> Arial </font-name>
        <font-color>0,0,0</font-color>
        <font-height>-13</font-height>
        <font-weight>700</font-weight>
        <font-italic>false</font-italic>
        <font-underline>false</font-underline>
        <font-strike-out>false</font-strike-out>
        <font-char-set>0</font-char-set>
    </default-font>
</header>
```

```
<text-list>
    <text prefix="ERROR" id="4711">
        <deutsch> Fehler an Position 4711 </deutsch>
        <english> Error at position 4711 </english>
    </text>
    <text prefix="ERROR" id="815">
        <deutsch> Fehler an Position 815 </deutsch>
        <english> Error at position 815 </english>
    </text>
    <text prefix="ERROR" id="2000">
        <deutsch> <![CDATA[Das ist ein Fehlertext über
        mehrere Zeilen]]> </deutsch>
        <english> <![CDATA[This is a error text over more than
        one line]]> </english>
    </text>
    <text prefix="MASCHINE" id="1">
        <deutsch> <![CDATA[Vorschub]]> </deutsch>
        <english> <![CDATA[Feed rate]]> </english>
    </text>
    <text prefix="MASCHINE" id="2">
        <deutsch> <![CDATA[Beschleunigung]]> </deutsch>
        <english> <![CDATA[Acceleration]]> </english>
    </text>
</text-list>
</dynamic-text>
```

(5) In the visualization open dialog ‚Settings', category Language: Activate option ‚Dynamic Texts'; Add file dynamictextsample.xml, now available on your computer, to the file list.
(6) Go online with the project.
(7) In the visualization settings set language to "deutsch". Set PLC_PRG.ivar to "1" and PLC_PRG.errnum to "4711". Now in the visualization elements the following texts should be displayed: "Vorschub" resp. "Fehler an Position 4711". The texts will be displayed in Arial 13.
(8) Set PLC_PRG.ivar to „2" and PLC_PRG.errnum to "2000". The texts will change to "Beschleunigung" and "Das ist ein Fehlertext über mehrere Zeilen".
(9) In the visualization settings change the language to "english". Now the following texts will be displayed: "Acceleration" and "This is a error text over more than one line".
(The change of language also could be managed by using the INTERN command 'LANGUAGE' by another visualization element.)

10.2.3 Language dependend Online Help

10.2.3.1 Calling up language-dependent Online Help via a visualization element

The calling of a different Help file with a visualization element can be tied in with the language currently entered for the visualization. For this purpose, the command INTERN HELP must be entered for this element in the 'Configure element' dialog at the location 'Execute program', and a [Visu-Helpfiles] section must be present in the TwinCAT PLC Control.ini file. Below this, the corresponding help files must be assigned to the languages available for selection in the visualization: e.g.:

```
[Visu-Helpfiles]
German=C:\PROGRAMME\HELP_D.HLP
English=C:\PROGRAMME\HELP_E.HLP
```

10.3 Placeholder Concept

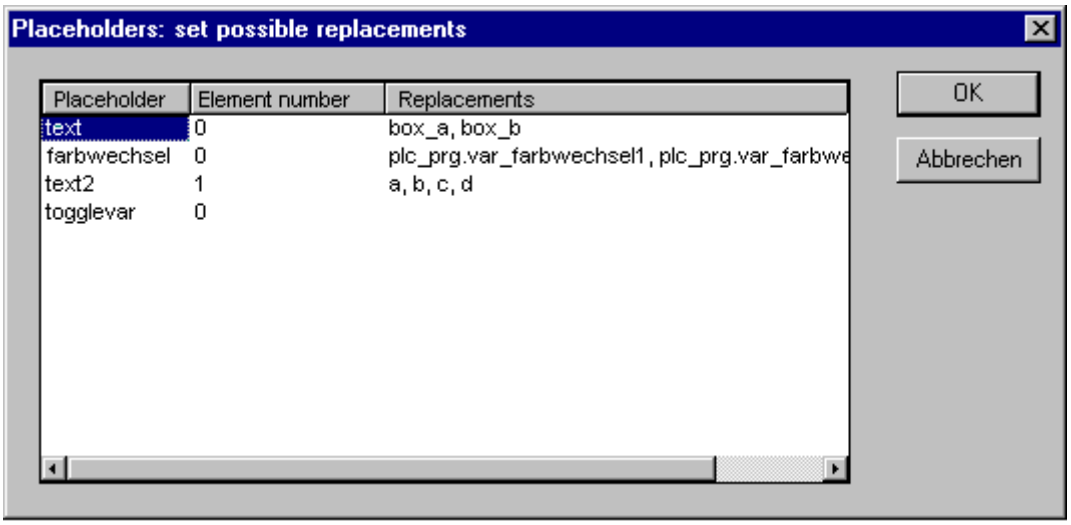
At each location in the configuration dialog at which variables or text are entered, a **placeholder** can be set in place of the respective variable or text. This makes sense if the visualization object is not to be used directly in the program, but is created to be inserted in other visualization objects as an "instance". When configuring such an **Instance**, the placeholders can be replaced with variable names or with text (see „Configuring an inserted visualization", there you also find an example for using placeholders).

Any string enclosed in two dollar signs ($) is a valid placeholder (e.g. $variable1$, variable$x$). For each placeholder a „value group" can be defined as an input specification in the 'Placeholder list' dialog (called from 'Extras' 'Placeholder list'). With one of these values you can replace the placeholder when configuring an instance of the visualization object. A placeholder list will be available in the instance to do this replacements

.

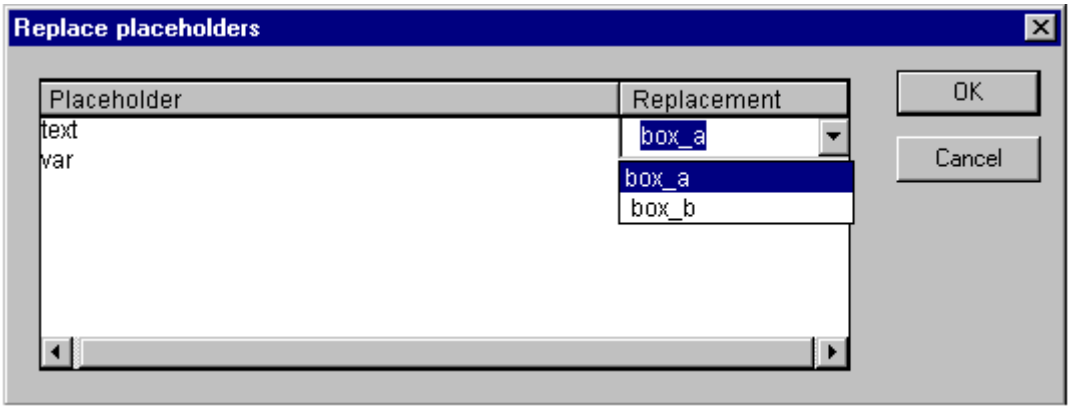
'Extras' 'List of Placeholders...'

- primarily you use the list when configuring a visualization object, which later should be inserted, which means instanced, in other visualization(s). For this reason you will use placeholders instead of or additionally to variables and strings in the configuration dialogs. You can open the dialog 'Placeholders' by the command 'List of Placeholders' in the 'Extras' menu or in the context menu. The list shows three columns:



Placeholder list for input of possible replacements for the placeholders

- Column **Placeholder** will list all placeholders, which are currently used in the configuration of the visualization object. Column **Element number** shows the elements which contain a placeholder. In column **Replacements** you can enter one or several strings (text, variable, expression) which you want to get available later when replacing a placeholder during the configuration of an instance of the visualization object. The elements of the selection must be entered separated by commas. If no or an impossible replacement string is specified, then the placeholder can be replaced with any desired text later during the configuration of the visualization's reference.
- later you use the list of placeholders when **configuring an instance** of the above mentioned visualization object, that means after this object has been inserted (as a 'reference') in another visualization by the command 'Insert' 'Visualization'. For this purpose do the following to open the dialog: Select the inserted visualization, execute command 'Extras' 'Configure' and press button 'Placeholders' in Category 'Visualization'. In this case the dialog will only contain two columns:



List of placeholders for replacing a placeholder in a visualization instance

- Column Placeholder – like described above – shows all placeholders which have been defined for the primary visualization object. If additionally a selection of possible replacements had been defined, this list will now be available in column 'Replacement'. Select one of the entries to replace the placeholder in the present instance. If no replacements have been pre-defined then you can manually enter an expression or variable. For this purpose perform a mouse-click on the field in column Replacement to open an editor field.

10.4 Online Mode

Regard the following items concerning a visualization in online mode:

Order of evaluation:

- Dynamically defined element properties (by variables) will overwrite the (static) base settings defined by options in the configuration dialogs.
- If an element property is defined by a "normal" project variable as well as by the component of a structure variable (Programability), then primarily the value of the project variable will be regarded.

- A visualization can be configured in that way that in online mode it can be operated solely by inputs via keyboard.
- The configuration settings for Display, Frame and Language can also be edited in online mode.
- As long as a visualization "reference" is not configured explicitly, the particular elements of the reference in online mode will react on inputs like those of the original visualization ("mother" of the references).
- When you switch the language ('Extras' 'Settings') this will only effect the display in online mode.
- A visualization can be printed in online mode.



For explanations on the online operation of certain visualization elements, such as trend and alarm table, please see the corresponding chapter on the configuration of the element.

Operation over the keyboard - in online mode

In order to get independent from the mouse or a touch screen, it is useful to configure a visualization in a way that allows pure keyboard operation:

Per default the following key (combinations) will work in online mode anyway (**no special configuration necessary**):

- Pressing the <Tabulator> key selects the first element in the element list for which an input is configured. Each subsequent pressing of the key moves one to the next element in the list. Pressing the key while keeping the <Shift> key depressed selects the previous element.

- The arrow keys can be used to change from a selected element to a neighbouring one in any direction.

- The <**Space bar**> is used to execute an activity on the selected visualization element. If the element is one which has a text output variable, a text input field will be opened which displays the text contents of the variable. Pressing the <**Enter**> key writes in this value.

Additional key (combinations) for the online operation can be defined in the configuration dialog 'Keyboard usage'. There also the keys <Tab>, <Space> and <Enter> can get assigned another functions than the above described standards.

The individual elements of references behave in Online mode identically to the corresponding ones in the visualization that is referenced. They will therefore react the same way as individual elements to inputs and operation by mouse and keyboard; the display of tooltips in references is also element-dependent. When processing the element list, as for instance when jumping from one input element to the next using the tabulator, the processing of all individual elements of a reference proceeds from the location of the reference in the element list before jumping to the next element of the list.

'File' 'Print' in online mode

'File' 'Print' is used to print out the contents of the visualization window in online mode. Visualizations which stretch over the border of the window can lead to inconsistencies particularly when there are moving elements in the visualization

10.5 Libraries

Visualizations can also be stored in libraries and thus be made available to projects in the form of library POUs. They can be inserted as references or they can be called up via the command „Zoom to vis." in the input configuration of another visualization which is part of the project..

NOTICE

Unique names

Visualizations used in a project must have unique names. It can be problematic if for instance a visualization from a library is called or referenced which has the same name as one present in the project. Because, in processing references or visualization calls in the program, first the visualizations in the project, and only thereafter the ones in the loaded libraries will be implemented.

10.6 TwinCAT PLC HMI Visualization

TwinCAT PLC HMI is a system needed for the execution of visualizations generated in TwinCAT PLC Control.

If a control program contains associated visualisations these are displayed in full screen mode when the **TwinCAT PLC HMI** is started. The user can operate the control and monitoring functions contained in the underlying project via the mouse or the keyboard. This is possible even if the TwinCAT PLC project file is read-protected. However, the user cannot edit the control program and no menus or toolbars are available, i.e. only pure 'operation' of the visualisation elements is available.
The main control and monitoring functions for a project intended for the operator version therefore have to be associated with the visualisation elements during project creation and operated in online mode. A special input facility is available for this purpose in the configuration dialog for a visualisation element.

By seamlessly integrating the development platform for the visualisation masks into the PLC programming system **TwinCAT PLC HMI** offers advantages other visualisations simply cannot come up to:

• A tag list of the variables to be used is not needed. It is possible to work directly with the TwinCAT PLC Control variables.

• Variable values in input fields can be modified by expressions (e.g. “Variable1+ Variable2 * 12 + 5”)

• A mighty place-holder concept allows the user to create object-oriented masks.

• The TwinCAT functions “Sampling Trace“ and “read/write recipe“ are also available in **TwinCAT PLC HMI**.

10.6.1 Installation, Start and Operating

Installation:

TwinCAT PLC HMI is available as supplement and can be installed with the setup. A licence is with costs, a time-limited demo version is not available.

Start:

TwinCAT PLC HMI (TCatPlcCtrlHmi.exe) is started by a connection or a command line:

In each case at least the desired TwinCAT PLC Control project has to be given. If no further parameters are set there, TwinCAT PLC HMI automatically will start with a visualization POU named **TC_VISU** (if existent in the project) and on that target mode, which was set when the project had been saved last.

Additionally as well the known command line and command file commands (see User Manual TwinCAT PLC Control) as the following special parameter can be used:

/visu <visualization POU>

If the project contains a visualization POU named TC_VISU, it will start automatically with this one. If another POU should be the entrance, it has to be set in the command line with "/visu <name of visualization POU>".

Example for a command line:

C:\TwinCAT\Plc\TCatPlcCtrlHmi.exe D:\PROJECTS\PROJECT.PRO /visu v_firstvisupage

The project project.pro will start with the visualization POU 'v_firstvisupage'. If the visualization should be started automatically with TwinCAT PLC Control, this can be done with a connection in TwinCAT StartUp (All Users) to the 'TCatPlcCtrlHmi.exe' and the named parameters.



Paths containing spaces must be bordered by quotation marks (").

Operating:

The project will start in **full screen mode** with the entrance POU.

TwinCAT PLC HMI can be operated corresponding to the functions of the visualization elements via keyboard and mouse.

If there is no visualization element configured with a corresponding function, TwinCAT PLC HMI at any time can be terminated by pressing **<Alt><F4>**.

10.7 Target Visualization

The Target-Visualization is one of the possible ways to use a TwinCAT Visualization. TwinCAT PLC Control can create ST code (Structured Text) for the visualization objects of a project. This code can be downloaded to the PLC together with the normal project code.

Thus, if it is supported by the target system and if an appropriate monitor is available, the visualization can be started directly on the PLC computer. It is not necessary any longer that the programming system is running in order to run the visualization. This means a considerable reduction of storage use.

Functions of the library **SysLibTargetVisu.lib** offer information about mouse-clicks performed by the user resp. entries of the dynamic textlist [▶ 328] and allow to use these information in the program.

10.7.1 Feature overview of visualisation with TwinCAT

It is shown in this table which possibilities the visualisation offers with TwinCAT.

Requirements

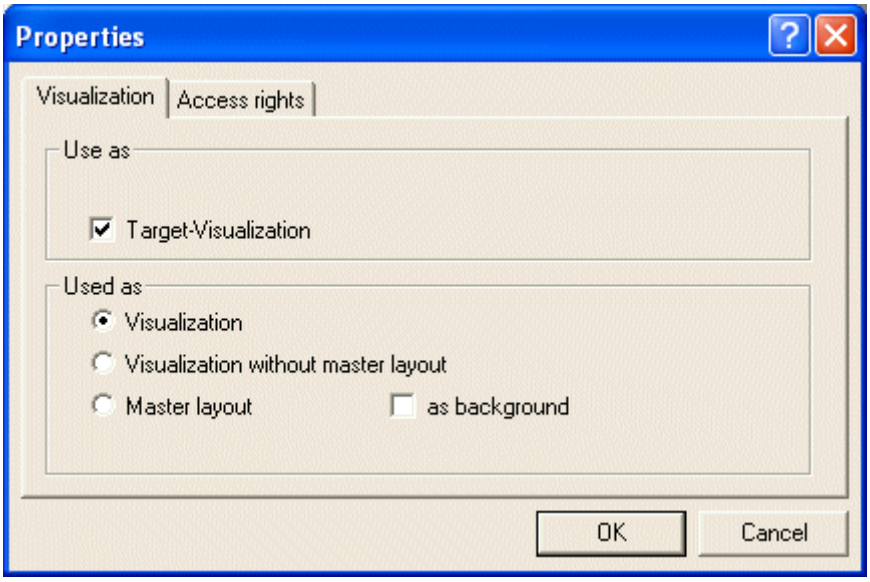
	PLC Control	PLC HMI	PLC HMI Web	PLC HMI CE	Comment
Current test version:	**Build 1010**	**2.10.0.900**	**1.0.0.7**	**1.0.9.10**	**From TwinCAT 2.10 Build 1334**
Rectangle	✓	✓	✓	✓	
Rounded rectangle	✓	✓	✓	✓	
Ellipse	✓	✓	✓	✓	
Polygon	✓	✓	✓	✓	
Curve	✓	✓	✓	✓	
Pie	✓	✓	✓	✓	
Bitmap	✓	✓	✓	✓	
Visaualisation	✓	✓	✓	✓	
Button	✓	✓	✓	✓	
WMF/JPG File	✓	✓	✓	✓	
Table	✓	✓	✓	✓	
ActiveX element	✓	✓	✗	✗	
Trend	✓	✓	✗	✓ / ✗	HMI CE: Only online trend is available
Alarm table	✓	✓	✗	✗	
Meter	✓	✓	✓	✓	
Bar display	✓	✓	✓	✓	
Histogram	✓	✓	✗	✗	
Invisible elements	✓	✓	✓	✓	
Change color	✓	✓	✓	✓	
Background bitmap	✓	✓	✓	✓	
Button background	✓	✓	✓	✓	
Tooltip	✓	✓	✓	✓	HMI CE: Elements not too near to the edge or tooltip is outside the window

	PLC Control	PLC HMI	PLC HMI Web	PLC HMI CE	Comment
Security	✓	✓	✓	✓	
Placeholder	✓	✓	✓ / ✗	✓ / ✗	Web/HMI CE: Zoom to visu with placeholders in the zoom command is not possible
Print function	✓	✓	✗	✗	
Password change	✓	✓	✓	✓	
Change user level	✓	✓	✓	✓	
Language dialog	✓	✓	✓	✓	
Language automatic change	✓	✓	✓	✓	
Exit	✓	✓	✗	✗	
Trace	✓	✓	✗	✗	
Text input 'Text'	✓	✓	✓	✓	Web: Please don't exceed the maximum string length of your variable HMI CE: The hidden function is not supported
Text input 'Numpad'	✓	✓	✓	✓	HMI CE: The hidden function is not supported
Text input 'Keypad'	✓	✓	✓	✓	Web: Please don't exceed the maximum string length of your variable HMI CE: The hidden function is not supported

See Restrictions [▶ 330]

10.7.2 Requirements

• The target system must support the functionality; that means that in the target settings (Project->Options->TwinCAT) the option 'Enable CE Target-Visualization' must be activated. If defined in the target file, this can be done by the user.



• The library **SysLibTargetVisu.lib** is required for implementing the visualization functions in the runtime system. It will be linked to the project as soon as the option 'Target-Visualization' is activated in the target settings. In the runtime system the SysLibTargetVisu.lib also must be implemented.

• If you will use the trend function under Windows CE, you have to insert the SysLibAlarmTrend.lib in your project manually. The trend function with ARM devices is not supported.

• Operating system of the PLC computer: Windows CE.

• The PLC computer needs devices for displaying and operating the visualization.

10.7.3 Creating a Target Visualization

1. Create a visualization in TwinCAT PLC.

In order to optimize the performance of the visualization, put as many elements, which are static (no movement, no dynamic texts, no dynamic color changes), **to the back**.
Hint: Using 'Extras' 'Elementlist' several or all elements can be "Sent to back" or "Sent to front" in a single blow. (Explanation: All static elements only once will be drawn to a background bitmap. Thus the cyclic repaint time will be reduced. This will be profitable especially for complex polygons or bitmaps.

If there is a visualization object **TC_VISU**, the Target-Visualization later will be started with this object. Otherwise it will start with that object which is the first one in the list of visualization objects in the Object Organizer.

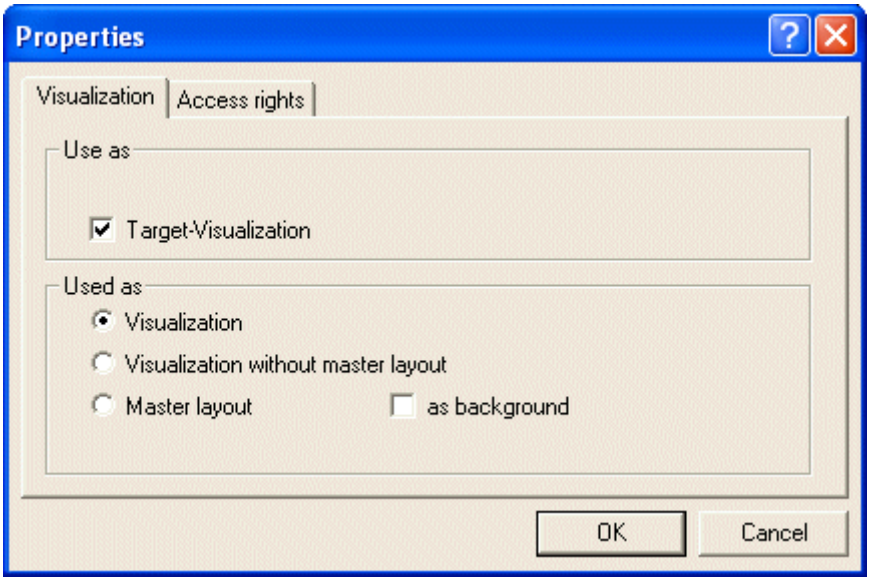
Consider whether implicit visualization variables in the current project should be handled as **remanent** variables and add the appropriate declarations in the Global Variables list [▶ 330].



The bitmaps of the visualization will be transferred as files.

2. Deactivate the option 'Target-Visualization'

For each visualization object, which should not be part of the Target-Visualization, deactivate the option 'Target-Visualization' in the Object Properties dialog ('Project' 'Object' 'Properties') in category 'Visualization



3. Configure...

In the Target Settings in tab 'Visualization' (it depends on the target system, whether the particular options are available in this dialog!) configure that the project should be prepared for use in the Target Visualization:
Activate option ‘Target visualization’

Additionally here you can define whether the user inputs and the re-painting of the visualization elements....
... should be controlled via VISU-tasks which are generated automatically or via individual programming:
Activate option '**Deactivate task generation**'.
... should be processed by one or by two POUs resp. tasks:
Activate option 'Use **VISU_INPUT_TASK**'. (Don't get irritated by the term "..._TASK" in this case; this option is also of effect if no VISU_INPUT_TASK is generated.)

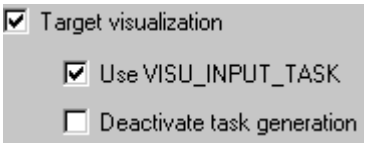
Thus, besides the possibility of deactivating the keyboard usage for table, the following configurations are possible:

The Target visualization is controlled by VISU-tasks
which are generated automatically and which call the POUs MAINTARGETVISU_PAINT_CODE and MAINTTARGETVISU_INPUT_CODE:

Case A:

Deactivate task generation is switched off:

Use **VISU_INPUT_TASK** is activated:



Automatically two tasks will be generated, each calling a program:
• **VISU_TASK** calls the implicitly available POU M**AINTARGETVISU_PAINT_CODE**, which does the repainting of the visualization elements.
• **VISU_INPUT_TASK** calls the implicitly available POU **MAINTARGETVISU_INPUT_CODE**, which does the processing of the user inputs.

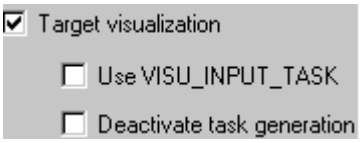
The default settings of the tasks:
- VISU_INPUT_TASK: cyclic, priority 2, interval t#50ms.
- VISU_TASK: cyclic, priority 3, interval t#200ms.

Of course the parameters can be modified. But: VISU_INPUT_TASK always should be processed before VISU_TASK in order to guarantee a useful interaction of user inputs and update of the visualization. The task calling the main program (e.g. PLC_PRG), should at least be processed as often as VISU_INPUT_TASK, ideally even with a higher priority, but it also could be added directly to VISU_INPUT_TASK

Case B:

Deactivate task generation is switched off:

Use **VISU_INPUT_TASK** is switched off:



Only task **VISU_TASK** will be generated automatically, but in this case will include the functionality of VISU_INPUT_TASK.
The implicit program POU **MAINTARGETVISU_ PAINT_CODE** in this case additionally will include the functionality of program MAINTTARGETVISU_INPUT_CODE.

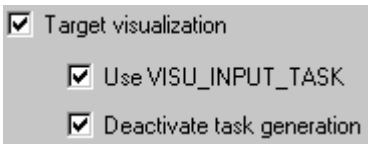
This configuration is intended for systems which do not allow multi-tasking. Disadvantageously no differentiated cycle times can be configured for the processing of user inputs and the repainting of the visualization elements, see above.

The Target visualization is not controlled by automatically created tasks;
The implicitly available POUs MAINTARGETVISU_PAINT_CODE and MAINTTARGETVISU_INPUT_CODE can be called by the application program resp. can be appended to any task:

Case C:

Deactivate task generation is activated:

Use VISU_INPUT_TASK is activated:



Both implicit POUs are available and can be called individually resp. can be appended to any task. (Referring to this regard the hints in (Case A).

Example for calls of the Target-Visulization POUs in the application program:

```
PROGRAM visu_control
VAR
    n: INT;
END_VAR

n:=n+1;
IF (n MOD 4) =0 THEN
   MAINTARGETVISU_PAINT_CODE();
END_IF;
MAINTARGETVISU_INPUT_CODE();
```

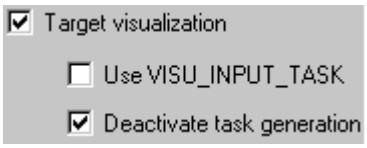
Here in program visu_control the POU which processes the user inputs is only called after each fourth call of the repainting POU - thus reducing the danger of getting disturbed the repainting by another user input.

When creating the application program, absolutely pay attention to reduce this danger of incorrect display !

Case D:

Deactivate task generation is activated:

Use **VISU_INPUT_TASK** is switched off:



Only the implicitly available program POU MAINTARGET VISU_PAINT_CODE can be used, but in this case it will additionally include the functionality of MAINTARGET VISU _INPUT_CODE.
This POU can be called in the application program resp. can be appended to any task.

4. Load the project

into the controller ('Online' 'Login').

10.7.4 Starting the Target Visualization

Start the loaded project on the PLC. Hereupon the visualization will start with the TC_VISU object resp. – if no TC_VISU is available – with that visualization object which is the first one in the list of visualization objects in the Object Organizer.

10.7.5 Scan of mouse-clicks and dynamic texts

The library SysLibTargetVisu.lib, which gets included automatically as soon as target setting 'Target-Visualization' is activated, offers the following functions for scanning user resp. entries of the dynamic textlist currently used for language dependent text display in the visualization.

Function GetText : BOOL

This function provides a language depending on text from the currently used dynamic textlist.

Parameters:

stResult: STRING(256);	Serves as an IN_OUT parameter and gets assigned the text found by prefix string „stPrefix“ and ID „dwID“.
nResultLength:INT;	Define here the maximum length of the string „stResult“, if this is <256 characters.
stPrefix: STRING;	Prefix of the text entry in the currently used dynamic textlist.
dwID: DWORD;	ID of the text entry in the currently used dynamic textlist.

Return value:
`FALSE` – No text matching „stPrefix“ and „dwID“ was found.
`TRUE` - A text matching „stPrefix“ and „dwID“ was found.

Function GetTextByStringId: BOOL

This function - like function GetText - can provide a language depending on text from the currently used dynamic textlist. The difference to GetTest is: The ID of the text entry is given as a string in parameter „stID“ instead of a numeric value. Thus it is also possible to use IDs which are defined as strings in the xml-file for the dynamic texts (e.g. „Text123“).

Parameters:

stResult: STRING(256);	Serves as an IN_OUT parameter and gets assigned the text found by prefix string „stPrefix“ and ID „dwID“.
nResultLength:INT;	Define here the maximum length of the string „stResult“, if this is <256 characters.
stPrefix: STRING;	Prefix of the text entry in the currently used dynamic textlist.
stID: STRING;	ID of the text entry in the currently used dynamic textlist.

Return value:
`FALSE` – No text matching „stPrefix“ and „stID“ was found.
`TRUE` - A text matching „stPrefix“ and „stID“ was found.

Additionally it is possible to scann user mouse events. For this do you have to insert the library TcMouseEvents.lib with the following functionallity:

Function GetLastLeftMouseDownEvent : BOOL resp. Funktion GetLastRightMouseDownEvent : BOOL

This function provides information on the last performed "Left" resp. "Right" MouseDown Event. It contains a pointer (pMouseEvent : POINTER TO MOUSEEVENT;) on structure MouseEvent consisting of the following parameters:

dwCounter : DWORD;	Number of MouseDown-Events since system start. With these parameters it can be evaluated whether no, one or multiple events have been prompted since the last request.
nXPos : INT;	Last mouse position in X/Y-coordinates.
nYPos : INT;	

Return value:
No return value.

Function GetLastMouseMoveEvent : BOOL

This function provides information on the last performed MouseMove-Event. It contains a pointer (pMouseEvent : POINTER TO MOUSEEVENT;) on structure MouseEvent, see above:function GetLastMouseDownEvent.

Return value:
No return value.

Function GetLastLeftMouseUpEvent : BOOL resp. Function GetLastRightMouseUpEvent : BOOL

This function provides information on the last performed "Left" resp. "Right" MouseUp-Event. It contains a pointer (pMouseEvent : POINTER TO MOUSEEVENT;) on structure MouseEvent, see above: function GetLastLeftMouseDownEvent resp. GetLastRightMouseDownEvent

Return value:
No return value.

10.7.6 Restrictions

Intern commands

PRINT
Printout of the current visualization is not possible.

EXITPROGRAM
This command is not supported.

TRACE
This command for opening the Sampling Trace window is not supported.

SAVEPROJECT
This command for saving the project is not usable for the Target-Visualization.

Grafic formats

In the Target-Visualization currently only simple bitmaps are supported. Not supported formats: .jpg, .tif, .ico
The format .jpg is supported from version 1.0.9 of the TargetVisu DLL.

Others

Slider in Table
The slider for scrolling is not displayed.

Texts
Texts exceeding the borders of an element currently do not get clipped.

Alarm handling
TheAlarming is not supported.

Trend
The Online Trend is supported from Version 1.0.8 (without history).

Place holder
The delivery of parameter to replace placeholder at the call is not supported.
Example:
<Visuname>(<Placeholder1>:=<Text1>, <Placeholder2>:=<Text2>,..., <Placeholder n>:=<Textn>)

VAR_IN_OUT
It is not possible to use VAR_IN_OUT variables in TwinCAT HMI CE.

Visu-Side
It is not possible to use the scroll bar in the HMI CE on your visualisation side.

10.8 System Variables

The following system variables can be used in a visualization program:

Requirements

Implicitely generated variable	Datatype	Function
CurrentVisu	String[40]	Name of the currently opened visualization. If the name gets changed, a change to another visualization will be done. Please regard, that the name string MUST be defined in **capital letters**.
CurrentCaller	String[40]	Name of the previously opened visualization. Is used for the ZOOMTOCALLER unctionality .
CurrentLanguage	String[40]	Currently set language, available in the language file. The language MUST be written in **capital letters.**
CurrentUserLevel	INT	Currently set user level 0..7
CurrentPasswords[0 .. 7]	ARRAY [0..7] OF STRING[20]	All passwords which are defined in TwinCAT PLC Control in "Usergroup passwords“.

11 Appendix

11.1 Using the Keyboard

If you would like to run TwinCAT PLC Control using only the keyboard, you will find it necessary to use a few commands that are not found in the menu.

- The function key <F6> allows you to toggle back and forth within the open POU between the Declaration and the Instruction parts.
- <Alt>+<F6> allows you to move from an open object to the Object Organizer and from there to the Message window if it is open. If a Search box is open, <Alt>+<F6> allows you to switch from Object Organizer to the Search box and from the there back to the object.
- Press <Tab> to move through the input fields and buttons in the dialog boxes.
- The arrow keys allow you to move through the register cards and objects within the Object Organizer and Library Manager.
- All other actions can be performed using the menu commands or with the shortcuts listed after the menu commands. <Shift>+<F10> opens the context menu which contains the commands most frequently used for the selected object or for the active editor.

Key Combinations

The following is an overview of all key combinations and function keys:

General Functions	
Move between the declaration part and the instruction part of a POU	<F6>
Move between the Object Organizer, the object and the Message window	<Alt>+<F6>
Context Menu	<Shift>+<F10>
Shortcut mode for declarations	<Ctrl>+<Enter>
Move from a message in the Message window back to the original position in the editor	<Enter>
Open and close multi-layered variables	<Enter>
Open and close folders	<Enter>
Switch register cards in the Object Organizer and the Library Manager	<Arrow keys>
Move to the next field within a dialog box	<Tab>
Context sensitive Help	<F1>

General Commands	
"File" "Save"	<Ctrl>+<S>
"File" "Print"	<Ctrl>+<P>
"File" "Exit"	<Alt>+<F4>
"Project" "Delete Object"	<Del>
"Project" "Add Object"	<Ins>
"Project" "Rename Object"	<Spacebar>
"Project" "Edit Object"	<Enter>
"Edit" "Undo"	<Ctrl>+<Z>
"Edit" "Redo"	<Ctrl>+<Y>
"Edit" "Cut"	<Ctrl>+<X> or <Shift>+<Del>
"Edit" "Copy"	<Ctrl>+<C>
"Edit" "Paste"	<Ctrl>+<V>
"Edit" "Delete"	<Del>
"Edit" "Find Next"	<F3>
"Edit" "Input Assistant"	<F2>
"Edit" "Next Error"	<F4>
"Edit" "Previous Error"	<Shift>+<F4>
'Online"Login'	<F11>
'Online"Logout'	<F12>
'Online"Run'	<F5>
'Online"Toggle Breakpoint'	<F9>
'Online"Step Over'	<F10>
'Online"Step In'	<F8>
'Online"Single Cycle'	<Ctrl>+<F5>
'Online"Write Values'	<Ctrl>+<F7>
'Online"Force Values'	<F7>
'Online"Release Force'	<Ctrl><Shift>+<F7>
'Online"Write/Force Dialog'	<Shift>+<F7>
‘Online’‘Display Flow Control’	<Ctrl>+<F11>
'Window"Messages'	<Shift>+<Esc>

FBD Editor Commands	
'Insert"Network (after)'	<Shift>+<T>
'Insert"Assign'	<Ctrl>+<A>
'Insert"Jump'	<Ctrl>+<L>
'Insert"Return'	<Ctrl>+<R>
'Insert"Operator'	<Ctrl>+<O>
'Insert"Function'	<Ctrl>+<F>
'Insert"Function Block'	<Ctrl>+<B>
'Insert"Input'	<Ctrl>+<U>
'Extras"Negate'	<Ctrl>+<N>
'Extras"Zoom'	<Alt>+<Enter>

CFC Editor Commands	
'Insert"POU'	<Ctrl>+<B>
'Insert"Input'	<Ctrl>+<E>
'Insert"Output'	<Ctrl>+<A>
'Insert"Jump'	<Ctrl>+<G>
'Insert"Label'	<Ctrl>+<L>
'Insert"Return'	<Ctrl>+<R>
'Insert"Comment'	<Ctrl>+<K>
'Insert"POU input'	<Ctrl>+<U>
'Extras"Negate'	<Ctrl>+<N>
'Extras"Set/Reset'	<Ctrl>+<T>
'Extras"Connections'	<Ctrl>+<M>
'Extras"EN/ENO'	<Ctrl>+<O>
'Extras"Zoom'	<Alt>+<Enter>

LD Editor Commands	
'Insert"Network (after)'	<Shift>+<T>
'Insert"Contact'	<Ctrl>+<O>
'Insert"Parallel Contact'	<Ctrl>+<R>
'Insert' 'Function Block'	<Ctrl>+<B>
'Insert"Coil'	<Ctrl>+<L>
'Extras"Paste Below'	<Ctrl>+<U>
'Extras"Negate'	<Ctrl>+<N>

SFC Editor Commands	
'Insert"Step-Transition (before)'	<Ctrl>+<T>
'Insert"Step-Transition (after)'	<Ctrl>+<E>
'Insert"Alternativ Branch (right)'	<Ctrl>+<A>
'Insert"Paralell Branch (right)'	<Ctrl>+<L>
'Insert"Jump' (AS)	<Ctrl>+<U>
'Extras"Zoom Action/Transition'	<Alt>+<Enter>
Move back to the editor from the SFC Overview	<Enter>

Work with the PLC Configuration	
Open and close organization elements	<Enter>
Place an edit control box around the name	<Spacebar>
‘Extras’ ‘Edit Entry’	<Enter>

Work with the Task configuration	
Place an edit control box around the task or program name	<Spacebar>

11.2 Command Line/Command File Commands

Command Line Commands

When TwinCAT PLC Control is started, you can add commands in the command line which will be asserted during execution of the program. These commands start with a "/". Capitalization/Use of small letters is not regarded. The commands will be executed sequentially from the left to the right.

Command	Description
/debug	
/online	
/run	
/show ...	Settings for the TwinCAT PLC Control frame window can be made.
/show hide	The window will not be displayed, it also will not be represented in the task menu.
/show icon	The window will be minimized in display.
/show max	The window will be maximized in display.
/show normal	The window will be displayed in the same status as it was during the last closing.
/out <outfile>	All messages are displayed in the message window and additionally are written in the file <outfile>.
/cmd <cmdfile>	After starting the commands of the <cmdfile> get executed.

The input of a command line is structured like this:

"<Path of the TwinCAT PLC Control-Exe file >""<Path of the project>"/<Command1>/<Command2>...

Example for a command line:

"D:\dir1 TwinCAT PLC Control" "C:\projects\ampel.pro" /show hide /cmd command.cmd

The project ampel.pro gets opened, but no window opens. The commands included in the command file command.cmd will be executed.

Command File (cmdfile) Commands

See the following table for a list of commands, which can be used in a command file (<cmdfile>). The command file you can call by a command line (see above). Capitalizing/Use of small letters is not regarded. The command line will be displayed as a message in the message window and can be given out in a message file (see below). Additionally to the command a "@" is prefixed. All signs after a semicolon (;) will be ignored (comment).

Commands of the online menu:

Commands	Description
online login	Login with the loaded project ('Online Login')
online logout	Logout ('Online' 'Logout')
online run	Start of the application program ('Online' 'Run')
online sim	Switch on of simulation mode 'Online' 'Simulation')
online sim off	Switch off of simulation mode ('Online' 'Simulation')

Commands of the file menu:

Commands	Description
file new	A new project is created ('File' 'New').
file open <projectfile>	The project <projectfile> will be loaded ('File' 'Open').
file close	The current project will be closed ('File' 'Close').
file save	The current project will be stored ('File' 'Save').
file saveas <projectfile>	The current project will be saved with the file name <projectfile> ('File' 'Save as').
file quit	TwinCAT PLC Control will be closed ('File' 'Exit').

Commands of the project menu:

Commands	Description
project compile	The current project will be compiled by "Rebuild all" ('Project' 'Rebuild all').
project check	The current project will be checked ('Project' 'Check').
project build	The current project will be built ('Project' 'Build').
project import <file1> ... <fileN>	The files <file1> ... <fileN> get imported into the current project ('Project' 'Import').
project export <expfile>	The current project will be exported in the file <expfile> ('Project' 'Export')
project expmul <expfile>	Each object of the current project will be exported in an own file, which gets the name of the object.

Commands for the control of the message file:

Commands	Description
out open <msgfile>	The file <msgfile> opens as message file. New messages will be appended.
out close	The currently shown message file will be closed.
out clear	All messages of the currently opened message file will be deleted.

Commands for the control of messages:

Commands	Description
echo on	The command lines will be displayed as messages.
echo off	The command lines will not be displayed as messages.
echo <text>	<text> will be displayed in the message window.

Commands for the control of replace of objects respectively for the control of files for import, export, replace:

Commands	Description
replace ok replace yes	Replace
replace no	Do not replace
replace noall	Replace none
replace yesall	Replace all

Commands for the control of the default parameters of dialogs:

Commands	Description
query on	Dialogs are displayed and need user input.
query off ok	All dialogs respond as if the user had clicked on the 'OK' button.
query off no	All dialogs respond as if the user had clicked on the 'No' button.
query off cancel	All dialogs respond as if the user had clicked on the 'Cancel' button.

Debug Command:

Commands	Description
debug	corresponds to the command "/debug" in the command line

Command for calling command files as subroutines:

Commands	Description
call <parameter1>... <parameter10>	Command files are called as subroutines. Up to ten parameters can be consigned. In the subroutine called you can access the parameters using $0 - $9.

Commands for setting the libraries:

Commands	Description
dir lib <libdir>	<libdir> is set as library directory
dir compile <compiledir>	<compiledir> is set as directory for compile files

Commands for setting a delay time concerning execution of the CMDFILEs:

Commands	Description
delay 5000	Waiting 5 Seconds.

Commands for control of the Watch and Recipe Manager:

Commands	Description
watchlist load <file>	The watch list saved in <file> will be loaded and the appropriate window will be opened ('Extras' 'Load Watch List').
watchlist save <file>	Saves the current watch list in <file> ('Extras' 'Save Watch List').
watchlist set <text>	A previous loaded watch list gets the name <text> ('Extras' 'Rename Watch List').
watchlist read	The values of the watch variables are updated ('Extras' 'Read Recipe').
watchlist write	The values of the watch list are written to the watch variables ('Extras' 'Write Recipe').

Commands for linking libraries:

Commands	Description
library add <library file1> <library file2> .. <library fileN>	Attaches the specified library file to the library list of the currently open project. If the file path is a relative path, the library directory entered in the project is used as the root of the path.
library delete [<library1> <library2> .. <libraryN>]	Deletes the specified library, or (if no library name is specified) all libraries from the library list of the currently open project.

Commands for copying objects:

Commands	Description
object copy <source project file> <source path> <target path>	Copies objects from the specified path of the source project file to the target path of the already opened project. If the source path is the name of an object, this will be copied. If it is a folder, all objects below this folder will be copied. In this case, the folder structure below the source folder will be duplicated. If the target path does not yet exist, it will be created.

Commands system calls:

Commands	Description
system <command>	Carries out the specified operating system command.

Commands for online modus:

Commands	Description
Reset	If you have initialized the variables with a specific value, then this command will reset the variables to the initialized value.
ResetAll	This command resets all variables including the persistent ones to their initialization values and erases the user program on the controller.
CreateBootproject	With this command the compiled project is set up on the controller.
ChooseRuntime <text>	A specific Runtime System can be chosen. For <text> use the Net-Id and seperated with ":" the port number. Example:ChooseRuntime 172.16.77.23.1.1:811 or ChooseRuntime 5.2.122.255.1.1:801

Example of a command file:

A command file like shown below will open the project file ampel.pro, will then load a watch list, which was stored as w.wtc, will then start the application program and write - after 1 second delay - the values of the variables into the watch list watch.wtc (which will be saved) and will finally close the project.

```
file open C:\work\projects\ampel.pro
query off ok
watchlist load c:\work\w.wtc
online login
online run
delay 1000
CreateBootproject
watchlist read
watchlist save c:\work\watch.wtc
online logout
file close
```

Trademark statements

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.

Third-party trademark statements

Intel, the Intel logo, Intel Core, Xeon, Intel Atom, Celeron and Pentium are trademarks of Intel Corporation or its subsidiaries.

Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

More Information:
www.beckhoff.com/tx1200

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com