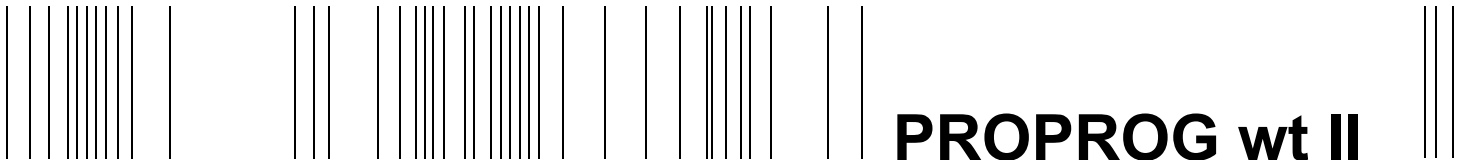


be in motion be in motion



PROPROG wt II

Manual

E

5.99003.06



Title	Manual
Product	PROPROG wt II
Last Revision:	May 09, 2005
Order number	340356
Copyright	<p>Owners may make as many copies as they like of this Manual exclusively for their own internal use. You are not allowed to copy or duplicate even extracts from this Manual for any other purposes.</p> <p>You are not permitted to exploit or communicate the contents of this Manual.</p> <p>Any other designations or company logos used in this Manual may be trademarks whose use by third parties for their own purposes may affect the rights of the owner of the trademark.</p>
Binding nature	<p>This Manual is a part of the unit/machine. This Manual must always be available to operators and be legible. If the unit/machine is sold, the owner must pass on this Manual together with the unit/machine.</p> <p>After selling the unit/machine you must pass on this original and all the copies that you made to the purchaser. After disposing of the machine in any way, you must destroy this original and all the copies that you made.</p> <p>When you pass on this Manual, all earlier revisions of the corresponding Manuals are invalidated.</p> <p>Note that all the data/numbers/information that are quoted are current values at the time of printing. This information is not legally binding for dimensioning, calculation and costing.</p> <p>Within the scope of further-development of our products, Baumüller Nürnberg GmbH reserve the right to change the technical data and handling.</p> <p>We cannot guarantee this Manual is completely error-free unless this is expressly indicated in our General Conditions of Business and Delivery.</p>
Manufacturer	<p>Baumüller Nürnberg GmbH Ostendstr. 80 - 90 90482 Nuremberg Germany Tel. +49 9 11 54 32 - 0 Fax: +49 9 11 54 32 - 1 30 www.baumueller.de</p>



Table of Contents

1	Introduction	7
1.1	PROPROG wt II - a highly efficient, powerful and complete tool	7
1.2	What kind of documentation do you get?	8
1.3	Symbols and textual conventions	8
2	PROPROG wt II and IEC 61131-3	11
2.1	What is IEC 61131-3?	11
2.2	Configuration elements	12
2.2.1	Configurations	12
2.2.2	Resources	12
2.2.3	Tasks	12
2.2.4	Configurations elements	13
2.3	POUs, programs, function blocks and functions	13
2.3.1	Program organization units - POUs	13
2.3.2	POUs	15
2.3.3	Instantiation	15
2.3.4	Declaration and instruction part of a POU	16
2.4	Variables and data types	17
2.5	Projects	18
2.6	Libraries	19
2.7	Programming languages and SFC	20
3	Getting started	21
3.1	System requirements	21
3.1.1	Hardware requirements	21
3.1.2	Software requirements	22
3.2	Installing the program	22
3.3	Calling the program	22
3.4	Using mouse and keyboard	23
3.5	User interface	24
3.5.1	Customizing the user interface	24
3.5.2	Menu	25
3.5.3	Toolbars	27
3.5.4	Defining keyboard shortcuts with the Shortcut Manager	29
3.5.5	Main screen and workspace	34
3.5.6	Message window	35
3.5.7	Cross reference window	36
3.5.8	Symbols in the Cross Reference Window	37
3.5.9	Status bar	40
3.6	Using help	41
3.6.1	Which kind of help files are available?	41
3.6.2	Using the general and the PLC specific help	41
3.6.3	Using help on functions and function blocks	41
3.7	Editors	42
3.7.1	The project manager - a powerful tool for program organization	42
3.7.2	Project tree	42
3.7.3	Instance tree	43
3.7.4	The graphic editor - easy programming in SFC, FBD and LD	43
3.7.5	The text editor - easy programming in IL and ST	44



TABLE OF CONTENTS

3.7.6	The page layout editor - creating page layouts for printing	45
3.8	The Edit Wizard	45
3.9	Overview window for graphical worksheets	48
3.10	Exiting worksheets	49
3.11	Exiting the program	50
4	Handling and editing projects	53
4.1	Creating a new project	53
4.1.1	Creating a new project using the Project Wizard	53
4.1.2	Creating a new project using a template	55
4.2	Saving an existing project as a template	56
4.3	Changing the properties of existing POUs	57
4.4	Inserting new POUs	59
4.5	Inserting worksheets	61
4.6	Announcing libraries	63
4.7	Deleting worksheets, POUs or libraries	65
4.8	Saving changes in worksheets while editing	65
4.9	Saving the existing project under a new name	66
4.10	Zipping the project files into an archive file	67
4.11	Translating the project language	70
4.11.1	Exporting a project translation file	70
4.11.2	Importing a project translation file	72
4.11.3	Switching the project language	74
4.12	Source conversion for IL, FBD and LD	75
5	Literals, data types and variables	77
5.1	Literals	77
5.1.1	Numeric literals	77
5.1.2	Character string literals	78
5.1.3	Duration literals	78
5.2	Introduction to the IEC data types	79
5.2.1	Elementary data types	79
5.2.2	Generic data types	80
5.2.3	User defined data types	81
5.3	Array data types	81
5.3.1	Declaring arrays	81
5.3.2	Programming example	82
5.3.3	Multi-dimensional arrays	83
5.3.4	Initializing arrays	84
5.4	Structured data types	84
5.4.1	Declaring structures	84
5.4.2	Programming example	85
5.4.3	Arrays of structures	85
5.4.4	Structures with arrays	86
5.4.5	Initializing structures	86
5.5	String data types	87
5.5.1	Declaring strings	87
5.5.2	Programming example	87
5.6	Calling the text editor with the data type worksheet	87
5.7	Editing type declarations using the Edit Wizard	88
5.8	Symbolic, located variables and directly represented variables	89
5.9	Global and local variables	90
5.10	Retentive variables	91
5.11	Initializing variables	91
5.12	Variable declaration keywords	93



5.13	Declaring variables	95
5.14	Instantiation	97
6	Editing in ST	99
6.1	Calling the text editor with a ST worksheet	99
6.2	Introduction to ST	100
6.3	Inserting and editing assignment statements	101
6.4	Inserting and editing further statements	102
6.5	Inserting statements using the Edit Wizard	103
6.6	Inserting variables	105
6.7	Calling functions or function blocks using the Edit Wizard	107
7	Editing in IL	111
7.1	Calling the text editor with an IL worksheet	111
7.2	Instructions, operators, modifiers and operands	112
7.3	Inserting instructions using the Edit Wizard	113
7.4	Inserting variables	115
7.5	Using jumps and labels	119
7.6	Calling functions or function blocks using the Edit Wizard	119
8	Editing in FBD	123
8.1	Calling the graphic editor with a FBD worksheet	123
8.2	Introduction to FBD	124
8.3	Inserting functions and function blocks using the Edit Wizard	125
8.4	Changing the properties of functions and function blocks	126
8.5	Replacing functions and function blocks	127
8.6	Inserting variables	127
8.7	Connecting objects	130
8.8	Negation of inputs and outputs	134
8.9	Duplicating inputs of functions	134
9	Editing in LD	137
9.1	Calling the graphic editor with a LD worksheet	137
9.2	LD networks, contacts, coils and power rails	138
9.3	Inserting contacts and coils	139
9.4	Inserting serial contacts and coils	140
9.5	Inserting parallel contacts or coils	141
9.6	Using the LD branch edit mode	141
9.7	Changing the properties of contacts and coils	142
9.8	Inserting variables	145
9.9	Calling functions or function blocks using the Edit Wizard	146
10	Editing in SFC	149
10.1	Calling the graphic editor with a SFC worksheet	149
10.2	Introduction to SFC	150
10.3	Inserting a first SFC network	150
10.4	Inserting more steps and transitions	151
10.5	Changing an initial step into a normal step or vice versa	153
10.6	Inserting alternative branches	155
10.7	Inserting simultaneous branches	156
10.8	Using the SFC branch edit mode	158
10.9	Inserting variables for actions	159
10.10	Inserting variables for transitions	162
10.11	Calling functions	166
10.12	Action and transition details	166



TABLE OF CONTENTS

11	Compiling, downloading and debugging	169
11.1	Inserting configurations, resources and tasks	169
11.1.1	Inserting a DEFAULT task	172
11.1.2	Insert a new task	173
11.1.3	Insert a system task	175
11.2	Associating programs to tasks	175
11.3	Compiling a project	176
11.4	Compiling a project using 'Make'	177
11.5	Patching POUs	179
11.6	Downloading the project	181
11.7	Calling worksheets in online mode	184
11.8	Adjusting the online layout and notation of online values	187
11.9	Switching between online and offline mode	188
11.10	Switching to address status and powerflow	189
11.11	Forcing and overwriting variables	190
11.12	Setting and resetting breakpoints	192
11.13	Debugging with set breakpoints	194
11.14	Using the watch window	196
11.15	Debugging user defined data types using the watch window	198
11.16	Using the Logic Analyzer	200
12	Printing your project with a customized page layout	207
12.1	Printing the project	207
12.1.1	Controlling the print process using the dialog 'Print Project'	207
12.1.2	Defining a page layout as default page layout	210
12.2	Using the page layout editor	211
12.2.1	Creating a new page layout	211
12.2.2	Defining the source area	212
12.2.3	Inserting elements in your page layout	214
12.2.4	Editing environment items	214
12.3	Using preview	215
13	System limits for PROPROG wt II	217
13.1	Project limits	217
13.2	Supported programming languages	218
13.3	Data types	219
13.4	User-defined data types depending on the implementation	219
13.5	Data type STRING depending on the implementation	220
13.6	ST depending on the implementation	221
13.7	VAR_IN_OUT depending on the implementation	221
13.8	Specials in PROPROG wt II	221
	List of Figures	223
	Index	227

1

INTRODUCTION

1.1 PROPROG wt II - a highly efficient, powerful and complete tool

PROPROG wt II is a standard programming system for IEC designed PLCs and traditional PLCs. It is based on the standard IEC 61131-3 and includes the full range of IEC features.

The programming system offers powerful features for the different developing phases of a PLC application:

- * Edit
- * Compile
- * Debug
- * Print

The programming system is based on a modern 32 bit windows technology, providing comfortable handling using zooming, scrolling, customizable toolbars, drag & drop operations, a shortcut manager and dockable windows.

The system allows especially handling of several configurations and resources within one project, including libraries and disposes of a powerful debug system. Projects are displayed and can be edited using a comfortable project tree editor in order to make the complexity of the IEC 61131-3 structure as simple and transparent as possible. Owing to the project tree editor easy inserting and editing of POU's, Data Types, Libraries and configuration elements is possible.

The programming system consists of a PLC independent kernel for programming in the various IEC programming languages, such as the textual languages ST and IL as well as the graphical languages FBD, LD and SFC. Each editor provides an Edit Wizard for fast and easy inserting pre-edited keywords, statements, operators, functions and function blocks. The Edit Wizard can also be used to declare variables and data types.

The independent kernel is completed with specific parts adapted to the different PLCs.

The new easy Online handling and the 32 bit simulation offers fast powerflow debug functionality and a real time multitasking test environment.

A comfortable tool for project documentation is implemented for printing the project documentation alternatively in a time-saving optimized way (using less paper) or with a stylish customized page layout.

1.2 What kind of documentation do you get?

1.2 What kind of documentation do you get?

The documentation is divided into several parts. For an understanding of all parts we are assuming knowledge about using MS-Windows.

The program manual provides all background information for a better understanding of the concepts of the PLC programming system and of the operations to be done. All steps from starting the program, editing worksheets up to exiting are described with several examples and figures. The manual should be used by users wishing to get a complete overview about how to realize a PLC program.

The context-sensitive Help which can be called by pressing F1 provides detailed and reference information for all program parts. The context-sensitive Help consists of several parts. A general part describes the general programming system features, which are not PLC-specific. The specific part describes all objects, dialogs and operations which differ from PLC to PLC. It should be used by experienced users having a concrete problem and searching for detailed information. Context-sensitive Help is also available for functions and function blocks, which can be inserted using the Edit Wizard.



Please refer also to your hardware documentation for PLC specific information.

1.3 Symbols and textual conventions

The following symbols are used in this manual:

- * is used for enumeration.
- is used for an operation which has to be done.
- ◇ is used for an operation which is optional.



is used for a sequence of operations to be done with the mouse.

In the procedures described in this manual the instructions 'click' and 'double click' relate to the left mouse button. If the right mouse button is meant (e.g. to open an object context menu) this is explicitly mentioned.



is used for a sequence of operations to be done with the keyboard.



NOTE

Notes are used to provide important information.



The book symbol is used to introduce references to other documents or chapters of this manual.



The Online help symbol is used to introduce references to the programming system help or PLC help.

The following textual conventions have been set up for this manual:

- ' commas are used for names of icons, menu items or proper names of objects e.g. menu item 'Cut'; function block 'Level'.
- <ALT> brackets are used for the name of keys on your keyboard and for words you have to enter.
- <ALT> + <F4> is used if you have to press two keys at the same time.
- editor name* Italic letters are used as place holders for proper names.

PROPROG WT II AND IEC 61131-3

2.1 What is IEC 61131-3?

The standard IEC 61131 has been established to standardize the multiple languages, sets of instructions and different concepts existing in the field of automation systems. The great variety of PLC concepts has led to an incompatibility between the different PLC platforms and manufacturers. The result was a great effort to be made for training, hard- and software investments.

IEC 61131 standardizes the programming languages, the interfaces between PLC and programming system, the sets of instructions and the handling and structuring of projects. The advantage of using IEC 61131 conform PLCs and programming systems is a portability of all platforms and the use of same concepts reducing costs for automation systems.

The standard consists of several parts and technical reports. The third part of the standard is dedicated to programming languages.

Obviously this standard has a great influence on the concept, structure, features and the handling of a programming system and the way to program the system.

The main changes that have come with IEC 61131-3 are:

- * Declaration of variables is similar to the variable declaration in higher programming languages.
- * Declaration of data types is possible.
- * Global and local data can be differentiated.
- * Programming means symbolic programming.

For a better understanding and an easier programming some IEC basics and their realization in your programming system are described in the following sections.

2.2 Configuration elements

An IEC 61131-3 conform PLC programming system reflects the hardware structure with the configuration elements.

These configuration elements are basically:

- * Configurations
- * Resources
- * Tasks

2.2.1 Configurations

A configuration can be compared to a programmable controller system, e.g. a rack. In a configuration one or several resources can be defined.

2.2.2 Resources

A resource can be compared to a CPU which can be inserted in the rack. In a resource global variables can be declared, which are only valid within this resource. In a resource one or several tasks can be executed.

2.2.3 Tasks

Tasks determine the time scheduling of the programs associated with them. This means that programs have to be associated to tasks. The settings of the task determine the time scheduling.

IEC 61131-3 describes different time scheduling which lead to three different task types:

- * **Cyclic tasks** are activated in a certain time interval and the program is executed periodically.
- * **Event or interrupt tasks** are activated if a certain event has happened.
- * **System tasks** will be activated if an error occurs when executing the PLC program. System tasks are associated with system programs.

Each task has a certain priority. In so called preemptive scheduling systems, an active task with low priority is interrupted immediately, when a task with higher priority becomes active due to a certain event. In systems with non-preemptive scheduling, task interruptions by tasks with higher priority are not possible.



NOTE

The supported task types depend on the used PLC.

2.2.4 Configurations elements

Configuration elements are represented graphically in the project tree. They are grouped together in the subtree 'Physical Hardware'.

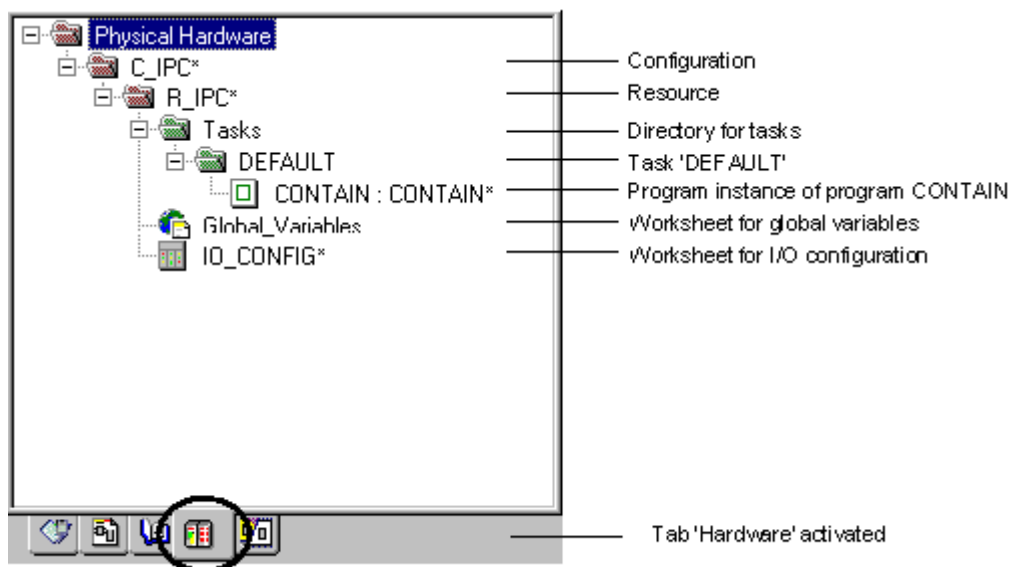


Figure 1: An example of configuration elements

The programming system reflects the structure of configuration elements in the subtree 'Physical Hardware' which may differ from PLC to PLC. In figure 2-1 the subtree 'Physical Hardware' with the configuration elements for simulation is shown.

In general one or several configurations can be used. In every configuration one or several resources can be declared. Several tasks with their associated programs can be used within one resource.

2.3 POU, programs, function blocks and functions

2.3.1 Program organization units - POUs

Program organization units or POUs are the language elements of a PLC program. They are small, independent software units containing the program code. The name of a POU should be unique within the project.

In IEC 61131-3 three types of POUs are distinguished referring to their different use:

- * Functions
- * Function blocks
- * Programs

Functions

Functions are POUs with multiple input parameters and exactly one output parameter. Calling a function with the same values returns always the same result. Return values can be single data types. Within a function it is possible to call another function but not a function block or a program. Recursive calls are not allowed.

IEC 61131-3 lists different types of standard functions:

- * Type conversion functions, such as INT_TO_REAL
- * Numerical functions, such as ABS and LOG
- * Standard arithmetic functions, such as ADD and MUL
- * Bit-string functions, such as AND and SHL
- * Selection and comparison functions, such as SEL and GE
- * Character string functions, such as RIGHT and INSERT
- * Functions of time data types, such as SUB with the data type TIME

Function blocks

Function blocks are POUs with multiple input/output parameters and internal memory. The value returned by a function block depends on the value of its internal memory. Within a function block it is possible to call another function block or functions. Recursive calls are not allowed.

IEC 61131-3 lists different types of standard function blocks:

- * Bistable elements, such as SR and RS
- * Edge detection function blocks, such as R_TRIG and F_TRIG
- * Counters, such as CTU and CTD
- * Timer function blocks, such as TON and TOF

Programs

Programs are POUs which contain a logical combination of functions and function blocks according to the needs of the controller process. The behavior and the use of programs are similar to function blocks. Programs can have an internal memory. Programs must be associated to tasks.

Within a program it is possible to call functions and function blocks. Recursive calls are not allowed.

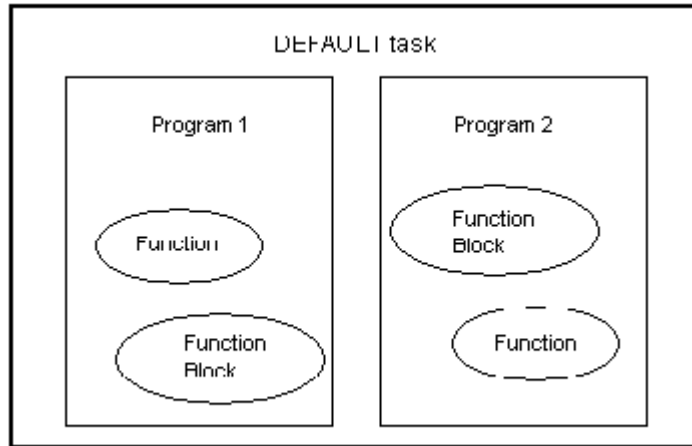


Figure 2: Diagram of a default task with two programs

2.3.2 POUs

Programs, function blocks and functions can be edited in the project tree. You can either display the complete project tree or only the subtree 'Data Types' and 'Logical POUs' by clicking on the tab 'POUs' as shown in the following figure.

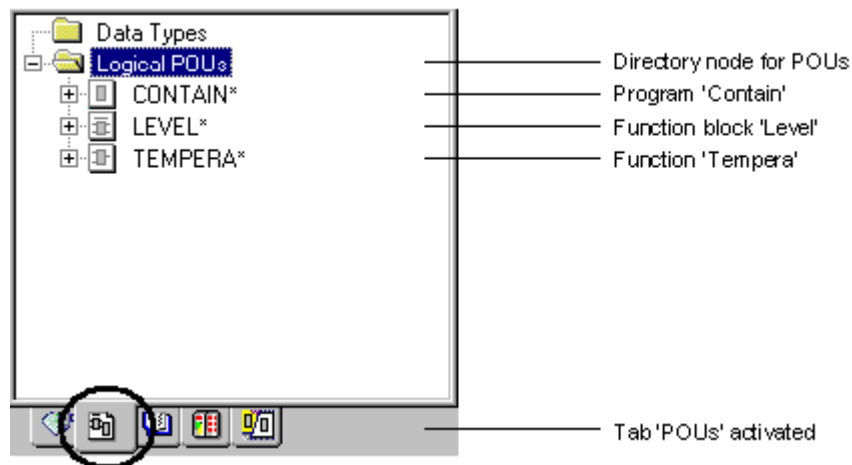


Figure 3: Subtree 'Logical POUs'

2.3.3 Instantiation

For reusing function block definitions IEC 61131-3 provides the possibility of instantiation. This means that the function block code body is defined once and that its internal memory is allocated to different instances, different memory regions. Each instance has an associated identifier (called instance name) and contains the input and output parameter and the internal memory of the function block. A function block can be instantiated in another function block or in a program. The instance name of a function block has to be declared

2.3 POU, programs, function blocks and functions

in the VAR declaration of the program or function block where it is going to be used. Programs can be instantiated within resources.

Instances are also displayed in the project tree window. The related subtree is made visible by clicking with the left mouse button on the tab 'Instances' as shown in the following figure.

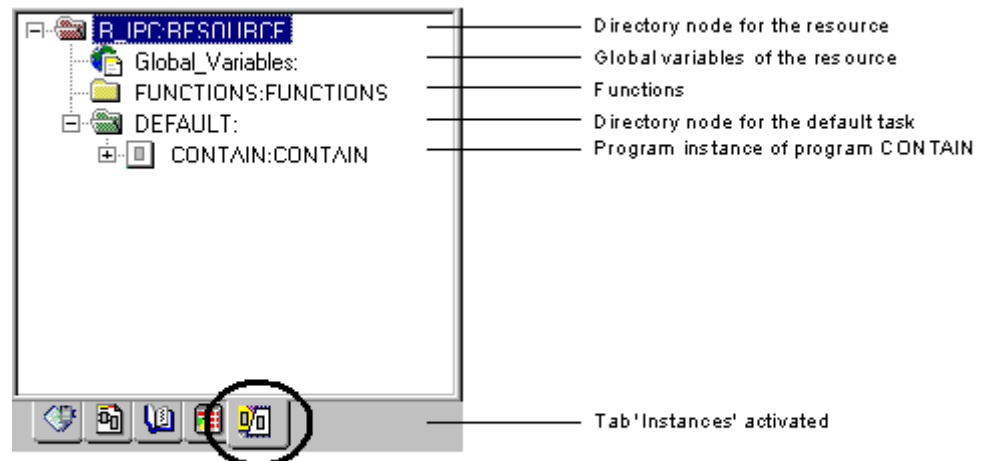


Figure 4: Project tree with the instances within the resource 'R_IPC'

2.3.4 Declaration and instruction part of a POU

Every POU consists of two different parts: The declaration part and the code body part.

In the **declaration part** all necessary variables are declared.

The **instruction** or **code body part** of a POU is the part in which the instructions are programmed in the desired programming language.

A POU consists of three types of worksheets. These three worksheets are represented graphically by icons:

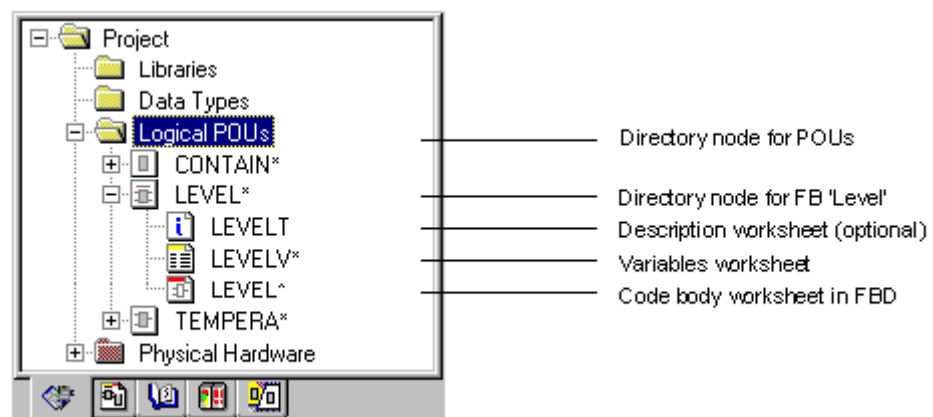


Figure 5: Worksheets of a function block in FBD

In the description worksheet annotations can be added for documentation purposes. In the variable worksheet all variables are going to be edited. The code body worksheet contains the instructions.

In the case of a SFC POU you have two more icons: the directory nodes for the action and transition worksheets.

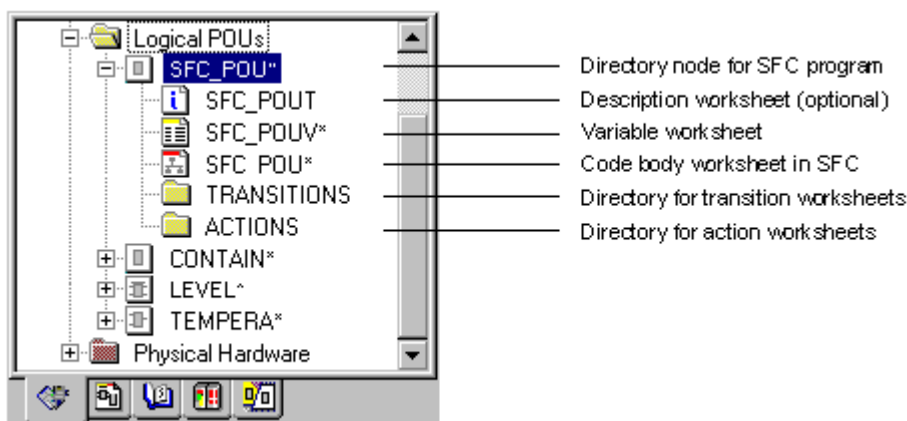


Figure 6: Icons of a SFC POU

2.4 Variables and data types

In IEC 61131-3 programming systems, **variables** are used instead of direct addressing of memory regions in former systems. Variables are assigned automatically to a memory region while compiling. IEC 61131-3 distinguishes different types of variable declarations e.g. VAR or VAR_INPUT. For PLC inputs and outputs direct addressing is possible using the keyword AT.

Variables with their properties are declared in the variable worksheet of the POU.

Data types determine what kind of value the variable can have. Data types define the initial value, range of possible values and the number of bits.

IEC 61131-3 distinguishes three kinds of data types:

- * Elementary data types
- * Generic data types
- * User defined data types

Elementary data types are data types whose range of possible values and number of bits is defined in IEC 61131-3. Elementary data types are e.g. BYTE, WORD or BOOL.

Generic data types are data types which include groups of elementary data types. They are called e.g. ANY_BIT or ANY_INT. ANY_INT includes e.g. the elementary data types INT, SINT, DINT, UINT, USINT and UDINT. Generic data types are necessary to define what kind of elementary data types can be connected to inputs or outputs of functions. If a function can be connected with ANY_INT it means that variables of the data types INT, SINT, DINT, UINT, USINT and UDINT can be connected.

User defined data types are data types which can be declared by the user. They have to be defined with a TYPE ... END_TYPE declaration. User defined data types can be structures or arrays.

User defined data types are declared in the data type worksheet in the subtree 'Data Types'. You can either display the complete project tree or only the subtree 'Data Types' and 'Logical POU's' by clicking on the tab 'POUs' as shown in the following figure.

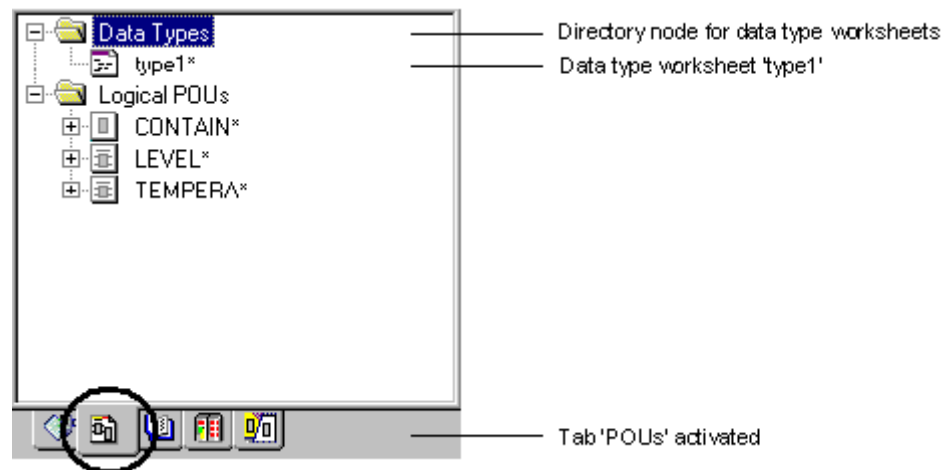


Figure 7: Subtree 'Data Types'



Variables and user defined data types and their declarations are described in the chapter 'Declaring variables and user defined data types' of this manual.

2.5 Projects

An IEC 61131-3 project contains all necessary elements of an automation system. It consists of libraries, data types, POU's and the set of configuration elements (such as resources, tasks and programs) which are represented in the subtree 'Physical Hardware'. A project is represented in the project tree.

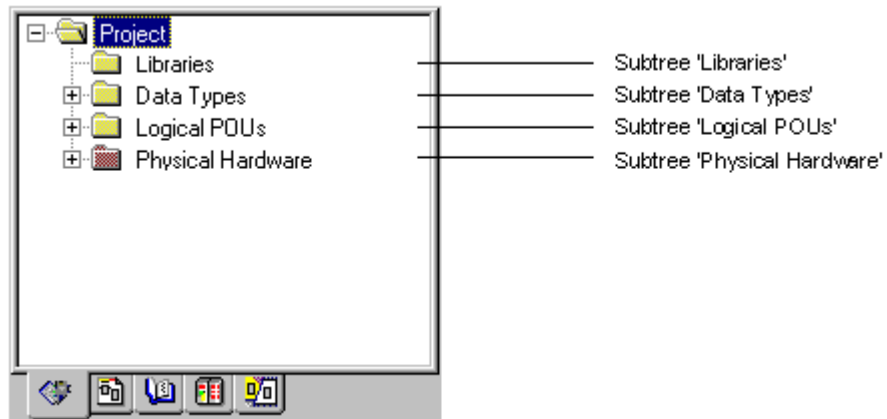


Figure 8: The project with its subtrees

2.6 Libraries

Libraries are projects which have been announced as libraries. You can reuse the programs, function blocks, functions and the user defined data types of the library in the project you are editing.

Firmware libraries are libraries containing POUs prepared by your PLC manufacturer. The file extension for firmware libraries is *.fwl.

User libraries are projects which you have created before and from which you want to reuse POUs. The file extension for user libraries is *.mwt.

Libraries have an own subtree in the project tree. You can either display the complete project tree or only the subtree 'Libraries' by clicking on the tab 'Libraries' as shown in the following figure.

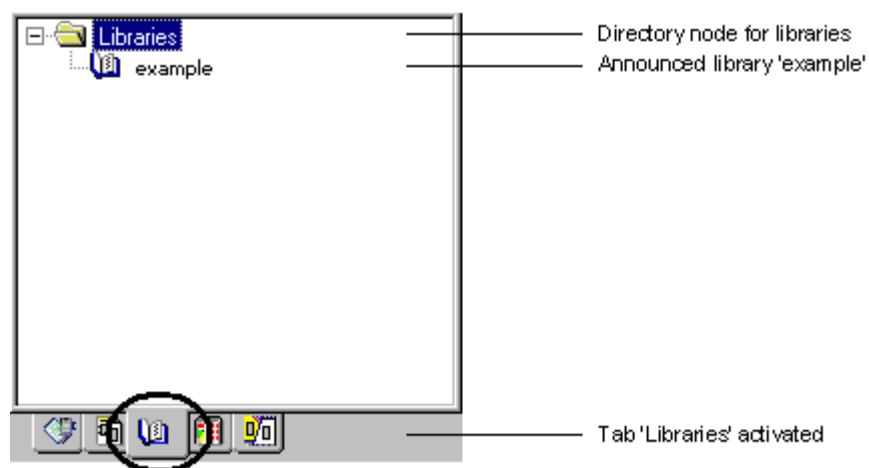


Figure 9: Subtree 'Library'

The subtree 'Library' consists of two or more icons. The first icon is a directory node. The icons within this directory node represent the announced libraries. In figure 2-9 you can see the announced user library 'example'.

2.7 Programming languages and SFC

IEC 61131-3 defines the syntax, the representation and the available language elements of 4 programming languages.

The programming languages can be differentiated into 2 textual languages and 2 graphical languages:

- * The **textual languages** are Structured Text (ST) and Instruction List (IL).
- * The **graphical languages** are Function Block Diagram (FBD) and Ladder Diagram (LD).

For structuring the internal organization of programs and function blocks **SFC** or **Sequential Function Chart** elements are defined in IEC 61131-3.



SFC and the 4 programming languages are described in the corresponding chapters of this manual.

3

GETTING STARTED

3.1 System requirements

3.1.1 Hardware requirements

To run the PLC programming system, the following workstation requirements must at least be fulfilled:

Device	Minimum	Recommended
IBM compatible PC with Pentium processor	133 MHz	350 MHz
System RAM	32 MB	64 MB
Hard disk	150 MB free memory space	
CD ROM drive	ja	
VGA Monitor Color settings Resolution	256 colors 800 x 600	True color 1024 x 768
RS232 interface	optional	
Mouse	recommended	
Ethernet	optional	

3.2 Installing the program

3.1.2 Software requirements

To run the PLC programming system, the following software requirements must at least be fulfilled:

- Microsoft Windows® 95, Microsoft Windows® 98, Microsoft Windows® NT, Microsoft Windows® 2000 or Microsoft Windows® XP
- Microsoft® Internet Explorer 4.0 or higher is required for HTML Help



NOTE

For installing as well as for executing the program administrator rights must be provided.

3.2 Installing the program

Using the installation program you can perform all necessary steps which are required to install the software. In order to start the installation program you have to perform the following steps:

- Insert the CD ROM disk into your CD ROM drive.
If AutoPlay is enabled, the installation program will start automatically. In this case you can skip the following two steps.
- If setup does not start automatically, open the Windows 'Start' menu and choose 'Run'. The 'Run' dialog appears.
- Type **d:\setup.exe** (where d is the appropriate CD ROM drive indicator) and press <↵>.
- The installation includes several dialogs where you have to do the corresponding entries.

After a successful installation, you will find the program group in the Windows® program menu. During installation you will be asked if you also want to install the program icons on your desktop.

3.3 Calling the program

To call the program open the Windows 'Start' menu, choose the menu 'Program' and select 'PROPROG wt II' or double click on the corresponding icon on your Windows desktop.

The program will be opened with the last project you have used. If you start the program for the first time it will be opened without any project.

To open an existing project, you have to perform the steps, which are described in the example of the following section 'Using mouse and keyboard'.

3.4 Using mouse and keyboard

The program supports full use of the mouse or the keyboard. For beginners it may be easier to start working with the mouse because it does not make necessary to learn the keyboard shortcuts. In rough industrial environments the keyboard may be more appropriate.

This manual explains both: the use of mouse and keyboard. In the next sections the general use of mouse and keyboard for the menu and toolbars is described.

The following is an example, how the usage of mouse and keyboard is described in this manual:



Opening an existing project using the mouse

- Click on the icon 'Open Project / Unzip Project' in the toolbar. The dialog 'Open project' appears.
- In the list box 'File type' select the desired file type.
- Browse to the corresponding project folder and locate the desired files (.mwt file for project file or .zwt file for zipped project files).
- Double click on the desired file name. The corresponding project is opened. If you choose a zipped project, the unzipping process is started automatically.



Opening an existing project using keyboard shortcuts

- Press <CTRL> + <O>.
- The dialog 'Open project' appears.
- In the list box 'File type' select the desired file type.
- Browse to the corresponding project folder by typing the folder names into the field 'file name' and pressing <↵>.
- Locate the desired files (.mwt file for project file or .zwt file for zipped project files).
- Open the desired project by typing the corresponding file name and pressing <↵>.

The usage of mouse and keyboard in the different editors is described in the following chapters.

3.5 User interface

The program user interface consists basically of six parts: Menu, toolbars, main screen, status bar, message window and cross reference window.

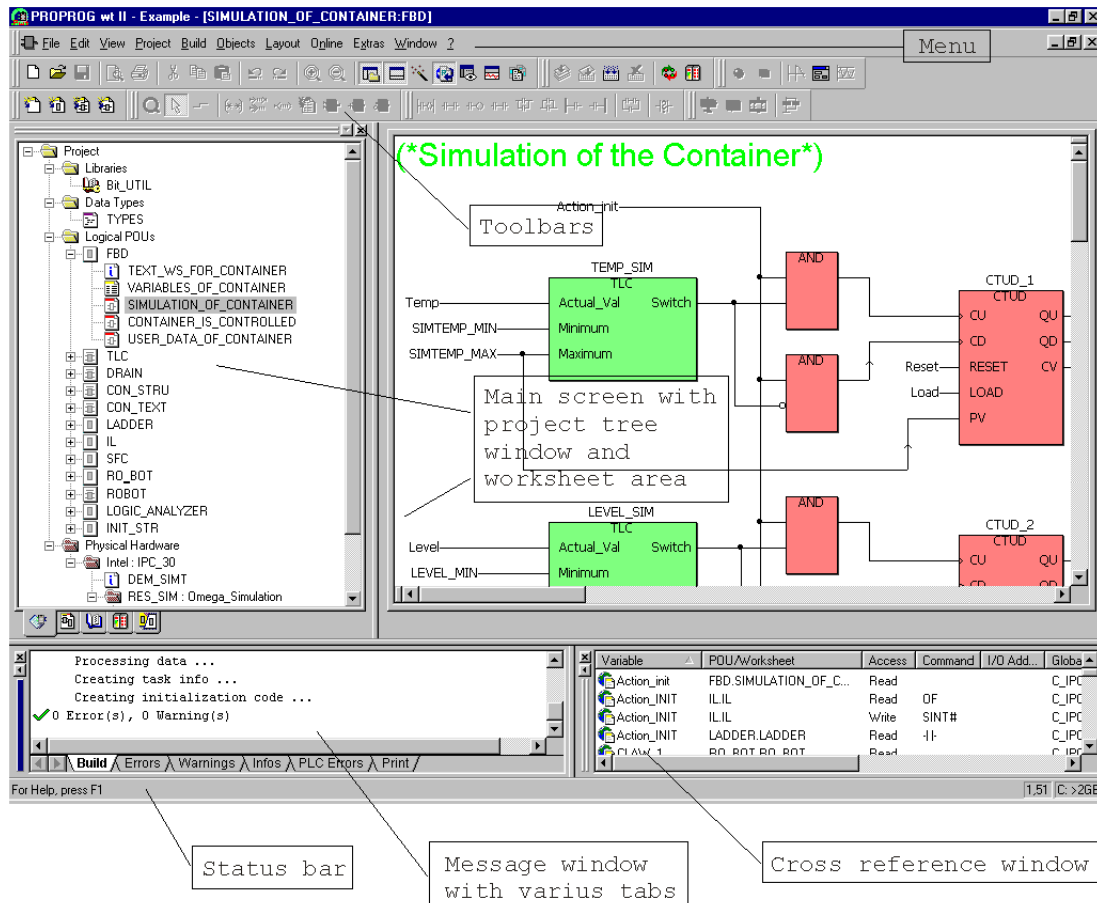


Figure 10: Program user interface with sample project 'example'

3.5.1 Customizing the user interface

You can customize the user interface to meet your personal requirements. For that purpose the dialog 'Options' is provided, containing several pages on which you can define the appearance and some preferences for the different parts of the system.

Using the 'Options' dialog it is possible to

- create new toolbars and populate them with the desired icons
- define general settings for the user interface (such as the appearance of the workspace)
- set compiler and debugger options
- set up the default paths for the project, for libraries and page layouts
- define default page layouts
- set up the automatic backup feature and the Autosave function

- set up the appearance of the text editor as well as the graphic editor (such as used colors, fonts, default worksheet size, etc.)

Detailed information about customizing the user interface



can be found in the general programming system help file. Please refer to the topic 'Customizing the user interface'.

3.5.2 Menu

The menu is located below the title bar. It contains several submenus.



NOTE

The menu items of these submenus change according to the program part or editor you are working with.

- * The submenu 'File' can be used to handle, save and zip/unzip projects. It also contains commands for printing, print setup and print preview. Additionally this submenu contains a menu item for exporting and importing. Using this menu items it is possible to export a translation file (*.csv) which contains all comments within the worksheets as well as the contents of the description worksheets. This csv-file can be translated into any language, e. g. by using a translation software. After (re)importing the translated file you can switch the project language by selecting the language in the project properties dialog.
- * The submenu 'Edit' contains all commands which are necessary for editing such as marking, choosing different working modes or cutting and pasting. Additionally it provides functionality for searching and replacing text strings used to edit textual worksheets (e. g. description worksheets, variables worksheets, ST worksheets or IL worksheets) and graphical worksheets. Depending on the active editor the menu contains also commands for handling objects (e. g. selecting, moving, enabling branch modes, etc.).
- * The submenu 'View' can be used to hide or show the different windows of the user interface (project tree window, message window, cross reference window, watch window, Edit Wizard, Logic Analyzer) and the status bar.
- * The submenu 'Project' can be used to insert new objects (such as data type worksheets and announced libraries), POU's and configurations.
- * The submenu 'Build' consists of different commands for starting the compilation after editing.
- * The submenu 'Objects' is available if you are using an editor. The menu item 'Variable' can be used to insert a new variable into the variable list of the current POU. When editing a graphical worksheet, the submenu provides additional menu items to insert and edit graphical objects, such as connectors, jumps, contacts, coils, etc. Depending on the graphical language you are using, some items may be grayed (i.e. inactive).
- * The submenu 'Layout' is available if you are using the graphic editor. It contains several designing utilities. You can display e.g. page borders (in the submenu 'borders') or a

grid for better organizing the content of your worksheets. Furthermore you can zoom into and out of the worksheet, modify the worksheet size, the autoscroll speed and the object size. This submenu also disposes of some features for the online layout.

- * The submenu 'Online' offers you commands for debugging a project, calling the Resource Control and activating the powerflow. In addition the command 'Online Layout' allows to set the appearance of graphical worksheets in online mode and a second submenu named 'Logic Analyzer' is available, containing the commands for controlling the Logic Analyzer.
- * The submenu 'Extras' can be used to call the dialogs 'Shortcut Keys' and 'Options' as well as other optional tools, such as the page layout editor. The dialog 'Shortcut Keys' (also known as Shortcut Manager) allows you to define your own keyboard shortcuts or customize the default shortcuts. The dialog 'Options' provides the facility to customize the menus, toolbars, text editors and text colors.
- * The submenu 'Window' can be used to arrange the windows and symbols on your screen and to close all open windows in one step.
- * The submenu 'Help' contains all commands for calling help.

The following procedures illustrate, how to call a menu item using the mouse and the keyboard.



Calling the menu item 'New Project...' with the mouse

- Click on the submenu 'File'. The submenu is opened and you can see the menu items.
- Select the menu item 'New Project...' with a left mouse click. The dialog 'New Project' appears.



Calling the menu item 'New Project...' using the keyboard

- Press <ALT> + <F>. The submenu is opened and you can see the menu items.
- Press <w> as it is the underlined character of the menu item 'New Project...'. The dialog 'New Project' appears.

NOTE



All submenus or menu items and dialog fields and boxes can be called pressing the underlined character of the corresponding word.

Using shortcuts is the easiest way of calling a menu item with the keyboard. For that reason the above mentioned method how to open a submenu and to choose an item is described only one time in this manual. In the following procedures the usage of shortcuts is described.



Calling the menu item 'New Project...' using the keyboard shortcut

- Press <CTRL> + <N>. This is the default shortcut for creating a new project. The dialog 'New Project' appears.

**NOTE**

Default shortcuts are already associated to the most important menu items. If not, you can open the dialog 'Shortcut Keys' (Shortcut Manager) and assign the corresponding menu item to your own shortcut.

3.5.3 Toolbars

The toolbars are located below the menu bar. By default all available toolbars are visible, providing two lines of different icons.

If you place the mouse cursor on any icon (without clicking it), a short description text, the so called tooltip appears. These tooltips display the name of the current icon. Additionally the status bar displays the function of the desired icon. The following figure shows an example:

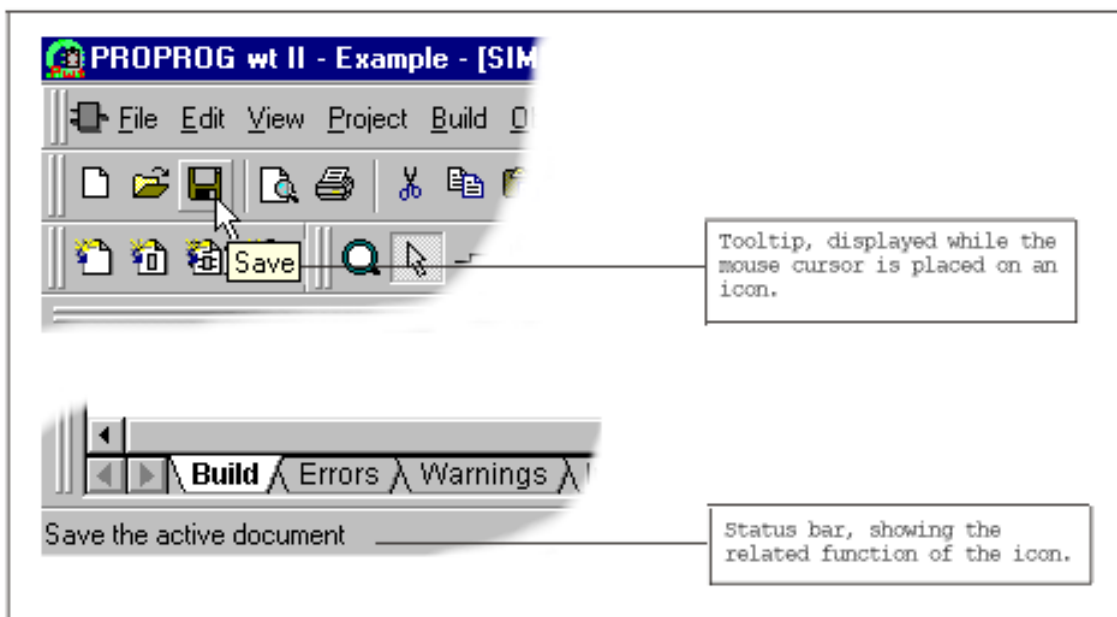


Figure 11: Sample tooltip (icon 'Save') and the corresponding description displayed in the status bar

**NOTE**

If the tooltips are not shown, open the dialog 'Options' by selecting the menu item 'Options' in the submenu 'Extras'. Activate the checkbox 'Show Tooltips' on the page 'Toolbars' and press 'OK' to confirm the dialog.

The toolbars have been implemented for realizing quickly often used operations with the mouse. In those cases one mouse click on a toolbar icon leads to the same result as doing several steps without the toolbar. You can adapt the toolbars corresponding to your needs. For this purpose you have to call the dialog 'Options' by selecting the menu item 'Options' in the submenu 'Extras'. In order to hide one certain toolbar, deselect the corresponding checkbox on the page 'Toolbars'.

Two different toolbar parts can be distinguished: General toolbars and specific toolbars.

- * General toolbars contain icons which are available everywhere in the program.
- * Specific toolbars contain icons which can be used only in specific editors. All icons are visible but the icons which cannot be used in a specific editor are grayed. You can hide each toolbar by using the dialog 'Options' as mentioned above.

NOTE



You can detach every toolbar from the other toolbar by double clicking on the gray toolbar background. The toolbar is then displayed in a window, which can be resized and moved to any position on your screen. To reinsert the toolbar window, just double click on the blue toolbar window title bar.

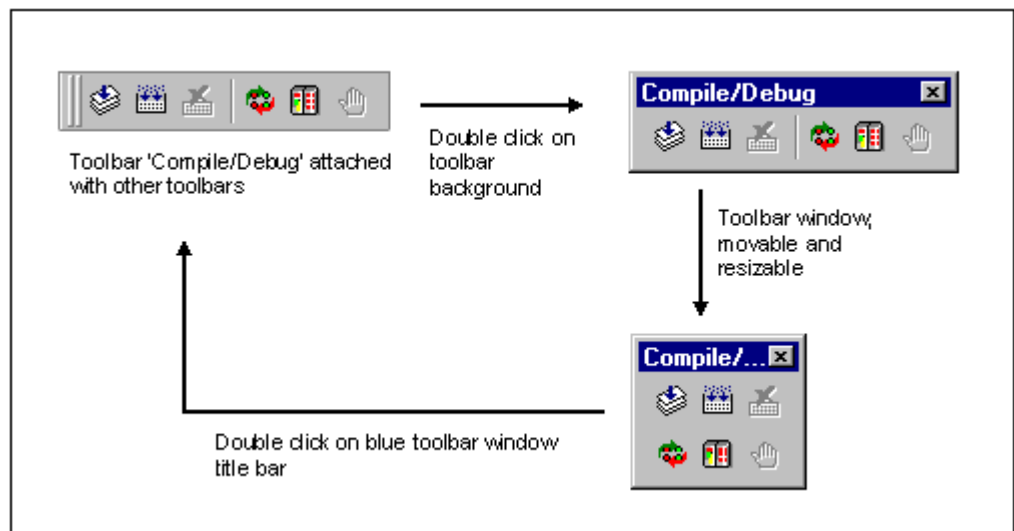


Figure 12: Example of a detached toolbar window



Using the toolbar for creating a new project

- Click on the icon 'New Project'.
The dialog 'New Project' appears, to select a template for the new project.



How to create a new toolbar



The procedure to create a new, customized toolbar and to populate it with the desired icons is described in detail in the general online help.

3.5.4 Defining keyboard shortcuts with the Shortcut Manager

As already mentioned in the section 'Menu' in this chapter, you can select certain menu items easier and faster by using keyboard shortcuts. A keyboard shortcut performs the same operation as the menu item to which it is assigned by simply pressing only one key or a key combination. In your PLC programming system several keyboard shortcuts can be used. The most important menu items (i. e. operations) are already associated to shortcuts by default. Assigned shortcuts are shown beneath the corresponding menu item in the submenus.

Using the Shortcut Manager, you can add new shortcuts for a specific menu item or modify existing shortcuts. For this purpose call the Shortcut Manager as follows.

NOTE



Many keyboard shortcuts are defined by default. Keep in mind, that this manual describes those default shortcuts. After modifying the default setting, some descriptions may not match your actual setting.



Calling the Shortcut Manager for adding/modifying shortcuts

- Click on the submenu 'Extras' to see the menu items.
- Select the menu item 'Shortcuts'. The dialog 'Shortcut Keys' appears.

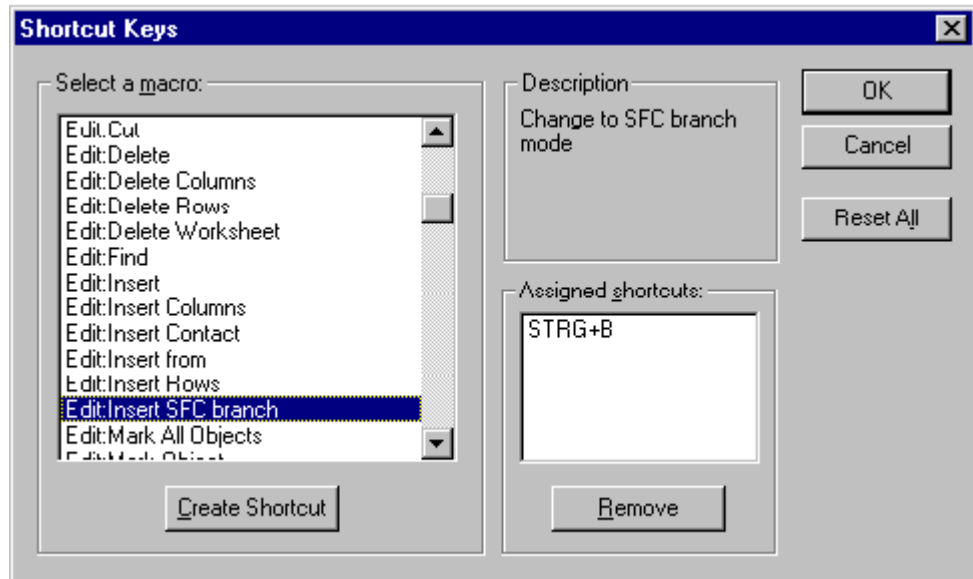


Figure 13: The 'Shortcut Manager' for adding/modifying keyboard shortcuts

The dialog shown in figure 3-4 is divided into three areas: The selection list 'Select a macro' contains a list of operations, which can be assigned to a shortcut. Most of these entries are available as menu items in the submenus. The field 'Description' displays a short text, describing the functionality of the selected macro (if available). If a shortcut is already assigned to the selected macro, this shortcut is displayed in the field 'Assigned shortcuts'.



Adding/modifying a keyboard shortcut

- Click on the desired operation/menu item in the list 'Select a macro'. The entry is highlighted. If available a short description is shown in the field 'Description'. If a shortcut is already assigned, it is shown in the field 'Assigned shortcuts'.
- To **modify** a shortcut, you have to remove the current shortcut and assign a new shortcut (refer to the steps below).
- To **remove** a certain shortcut, select it with a left mouse click and press 'Remove'.
- To **add** a new shortcut, perform the steps described in the following procedure. In this example, it is assumed, that you want to assign an additional shortcut to the menu item 'Stop Compile' in the submenu 'Build'.
- To **reset all** shortcuts to their default settings, click on the button 'Reset All'. All shortcuts you have created will be overwritten.



Assigning the shortcut <ALT> + <F6> to the macro 'Build:Stop Compile'

For the following description it is assumed, that the dialog 'Shortcut Keys' is already opened.

- Double click on the desired macro 'Build:Stop Compile' in the field 'Select a macro'. The dialog 'Assign Shortcut' appears.



Figure 14: Dialog 'Assign Shortcut' before pressing the desired shortcut keys

- Press the desired shortcut keys (in our example <ALT> and <F6>). They are shown in the text field. Below the text field the Shortcut Manager shows, whether the desired keys are already assigned to another macro.



Figure 15: Dialog 'Assign Shortcut' after pressing the desired shortcut keys

- Confirm the new shortcut by clicking on 'OK'.

NOTE



If the desired keys are already assigned to another macro and you confirm your current selection, the shortcut keys will be reassigned without a warning message.

The dialog 'Assign Shortcuts' is closed and the new assigned shortcut is shown in the dialog 'Shortcut Keys'.

The following list contains the default shortcuts in alphabetical order, assigned by the software manufacturer:

This operation/menu item is assigned to this default shortcut
Activate: Cross References	<ALT> + <4>
Activate: Edit Wizard	<ALT> + <3>
Activate: Editor Window	<ALT> + <0>
Activate: Logic Analyzer	<ALT> + <6>
Activate: Message Window	<ALT> + <2>
Activate: Overview Window	<ALT> + <9>
Activate: Project Tree Window	<ALT> + <1>
Activate: Watch Window	<ALT> + <5>
Build: Build Cross References	<F12>
Build: Compile Worksheet	<SHIFT> + <F9>
Build: Go to next Error	<CTRL> + <F12>
Build: Go to Previous Error	<SHIFT> + <F12>
Build: Make	<F9>
Build: Patch POU	<ALT> + <F9>
Build: Rebuild Project	<CTRL> + <F9>
Cross References: Next Reference	<ALT> + <F12>
Cross References: Previous Reference	<SHIFT> + <CTRL> + <F12>
Edit: Copy	<CTRL> + <C> <CTRL> + <INS>
Edit: Cut	<SHIFT> + <CTRL> + <X>
Edit: Delete	
Edit: Find Next (global)	F3
Edit: Find Next (local)	<CTRL> + <F3>
Edit: Find Previous (global)	<SHIFT> + <F3>
Edit: Find Previous (local)	<ALT> + <F3>
Edit: Insert	<INS>
Edit: Modes: Connect Objects	<CTRL> + <L>
Edit: Modes: Insert LD Branch	<CTRL> + <T>
Edit: Modes: Insert SFC branch	<CTRL> +
Edit: Modes: Mark Object	<CTRL> + <M>
Edit: Paste	<CTRL> + <V> <SHIFT> + <INS>
Edit: Select all	<CTRL> + <A>
Edit: Stretch/Compress: Delete Columns	<CTRL> + <4>
Edit: Stretch/Compress: Delete Rows	<CTRL> + <2>
Edit: Stretch/Compress: Insert Columns	<CTRL> + <3>
Edit: Stretch/Compress: Insert Rows	<CTRL> + <1>

This operation/menu item is assigned to this default shortcut
Edit: Undo	<CTRL> + <Z>
File: New Project	<CTRL> + <N>
File: Open Project / Unzip Project	<CTRL> + <O>
File: Print	<CTRL> + <P>
File: Save	<CTRL> + <S>
Help	<SHIFT> + <F1>
Help	<F1>
Layout: Grid	<CTRL> + <.>
Layout: Overview Window	<ALT> + <9>
Layout: Previous View	<CTRL> + <<>
Layout: Zoom in	<CTRL> + <+>
Layout: Zoom out	<CTRL> + <->
Objects: Action block	<ALT> + <F8>
Objects: Contact below	<CTRL> + <F7>
Objects: Contact left	<ALT> + <F7>
Objects: Contact network	<F6>
Objects: Contact right	<F7>
Objects: Duplicate FP	<CTRL> + <F5>
Objects: Function/Function block	<ALT> + <F5>
Objects: Simultaneous / Alternative Divergence	<CTRL> + <F8>
Objects: Step/Transition	<F8>
Objects: Toggle contact/coil properties	<SHIFT> + <F7>
Objects: Toggle FP negation	<SHIFT> + <F5>
Objects: Variable	<F5>
Online: Debug	<F10>
Online: Logic Analyzer: Insert/Delete Variable	<CTRL> + <F11>
Online: Logic Analyzer: Start Recording	<F11>
Online: Logic Analyzer: Trigger conditions	<SHIFT> + <F11>
Online: Resource Control	<CTRL> + <F10>
View: Cross References Window	<ALT> + <F2>
View: Edit Wizard	<SHIFT> + <F2>
View: Logic Analyzer	<ALT> + <F11>
View: Message Window	<CTRL> + <F2>
View: Project Tree Window	<F2>
View: Watch Window	<ALT> + <F10>
Window: Close All	F4

Figure 16: Default shortcuts in alphabetical order

3.5.5 Main screen and workspace

The main screen (see figure 3-1) is divided into two parts: The project tree window and the workspace. The workspace contains the opened worksheets. You can open a worksheet by double clicking on the corresponding worksheet icon in the project tree.

If several worksheets are opened only one worksheet is visible. A sheet tab is assigned to every opened worksheet as shown in the following figure. You can select (activate) a particular worksheet by clicking on the corresponding tab or by pressing <CTRL> + <TAB>.

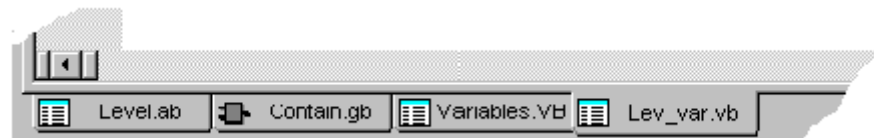


Figure 17: Worksheet tabs in the workspace



NOTE

If the worksheet tabs are not visible, open the dialog 'Options' by selecting the menu item 'Options' in the submenu 'Extras'. Click on the tab 'General' and activate the checkbox 'Workbook style'. Confirm your selection with 'OK'.

It is also possible to arrange several worksheet windows in a desired combination. For this purpose choose the menu items 'Cascade' and 'Tile' in the submenu 'Window'. This way you can easily prepare the workspace for different working phases.

You can maximize the workspace, which means, that the project tree is not displayed and the workspace is enlarged to the whole main screen width. This could be useful for displaying large networks in the graphical editor.



Displaying/Hiding the project tree window with the mouse

- Click on the icon 'Project Tree'.
Depending on the previous state, the project tree window is now visible or hidden.



Displaying/Hiding the project tree window with the keyboard

- Press <F2>.
Depending on the previous state, the project tree window is now visible or hidden.

3.5.6 Message window

The message window is located below the main screen. It contains several pages, which are activated by clicking on the corresponding tab.

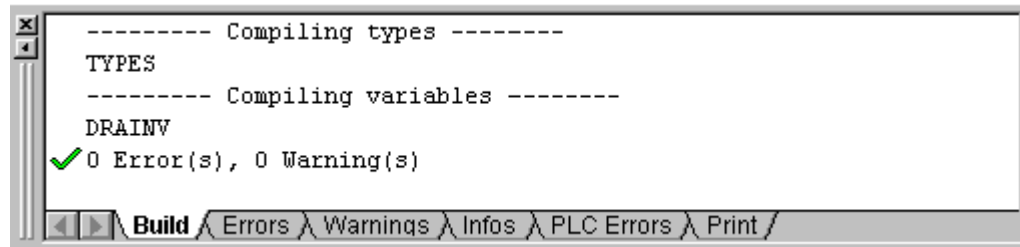


Figure 18: Message window with its different page tabs

The message window allows to display different steps during compiling, the compiler errors and warnings and some other information.

One of the main benefits of the message window is the possibility of direct accessing the worksheets in which the compiler detected errors. For this purpose you just have to double click on the corresponding error.

You can toggle between hidden and visible message window.



Displaying/Hiding the message window with the mouse

- Click on the icon 'Messages'.
Depending on the previous state, the message window is now visible or hidden.



Displaying/Hiding the message window with the keyboard

- Press <CTRL> + <F2>.
Depending on the previous state, the message window is now visible or hidden.

The message window is a dockable window. This means, that you can detach the window from the desktop by double clicking on the gray window border. It is then displayed as a usual window, i.e. you can change the size and move the window to any position on the screen. To reattach it into the desk, just double click into the blue window title bar. The handling is similar as for the toolbars.

How to use the message window



Detailed information about using the message window are described in the general programming system online help.

3.5.7 Cross reference window

The cross reference list contains all external variables, local variables and function blocks, which are used within the current project. It is a helpful tool for debugging and fault isolation.

NOTE



Every POU contains its own local data. That means if you open a particular worksheet, the local variables in the cross reference list are updated.

To use the cross reference list in an existing or a new project, you have to build the cross references in order to display the required information in the cross reference window.

In addition the program allows to toggle between hidden and visible cross reference window.

Building the cross reference list with the mouse



- If the cross reference window is not visible, click on the icon 'Cross References'.
The cross reference window is now visible.
- Place the mouse cursor in the cross reference window.
- Click with the right mouse button on the window background to open the context menu.
- Select the menu item 'Build Cross References'.
The cross reference list is created automatically as shown in the following figure.



NOTE



It is also possible to call the menu item 'Build Cross References' in the submenu 'Build' or in the editor context menu.

Displaying/Hiding the cross reference window with the keyboard



- Press <ALT> + <F2>.
Depending on the previous state, the cross reference window is now visible or hidden.






Variable	POU/Worksheet	Access	Command	I/O Ad...	Global Path	Type	Comment	Line/Po...
 Action_INIT	IL.LL	Write	ST		C_IPC.RES_...	BOOL	Action INIT is acti...	16
 Action_INIT	LD.LD	Read	- -		C_IPC.RES_...	BOOL	Action INIT is acti...	35/31
 CLAW_1	RD_BOT.RD_B...	Read			C_IPC.RES_...	INT		17/20
 CLAW_2	RD_BOT.RD_B...	Read			C_IPC.RES_...	INT		17/41
 Coil_out1_0	LD.LD	Write	{-}	%QX1.0	C_IPC.RES_...	BOOL	LD	122/9

Figure 19: Cross reference window

For each entry the following information are available:

- * A symbol at the beginning of each line allows a fast overview on the displayed variable types. The various symbols are described in the table shown below.
- * 'POU/Worksheet': POU name in which the variable/FB is declared and the particular worksheet name where it is implemented.
- * 'Access': The access of the variable is 'Read' (Load) or 'Write' (Stored).
- * 'Command': Command in which the variable is used (only applicable for IL and LD code body worksheets).
- * 'I/O Address': Physical PLC address.
- * 'Global Path': Instance path of global variables, consisting of the configuration name and the resource name, where the global variable is declared.
- * 'Type': The associated data type.
- * 'Comment': User-defined comment.
- * 'Line/Position(X/Y)': Line number in textual worksheets or element position in graphical worksheets.





NOTE



To open the corresponding worksheet in which a particular variable is used, just double click on the required line in the cross reference window. The worksheet is opened automatically and the variable is marked. Furthermore a variable is marked in the cross reference window, if you select the variable in an editor.

3.5.8 Symbols in the Cross Reference Window

In order to get a fast overview of the objects which are displayed, each item is shown with a symbol representing its type. The following table lists the different items and their corresponding symbol shown in the column 'Variable':

Symbol	Meaning	Symbol	Meaning
	local variable		step
	global variable		transition detail








Symbol	Meaning	Symbol	Meaning
	input variable (VAR_IN)		action detail
	output variable (VAR_OUT)		jump
	in-/output variable (VAR_IN_OUT)	LBL	label
	function block		connector

Figure 20: Symbols in the cross reference window



Filtering the cross reference list

You can filter the cross reference list to show only a particular subset of variables or function blocks.

- Move the mouse cursor into the cross reference window.
- Click with the right mouse button on the window background to open the context menu.
- Choose the menu item 'Filter...'.
The dialog 'Cross Reference Filter' appears.

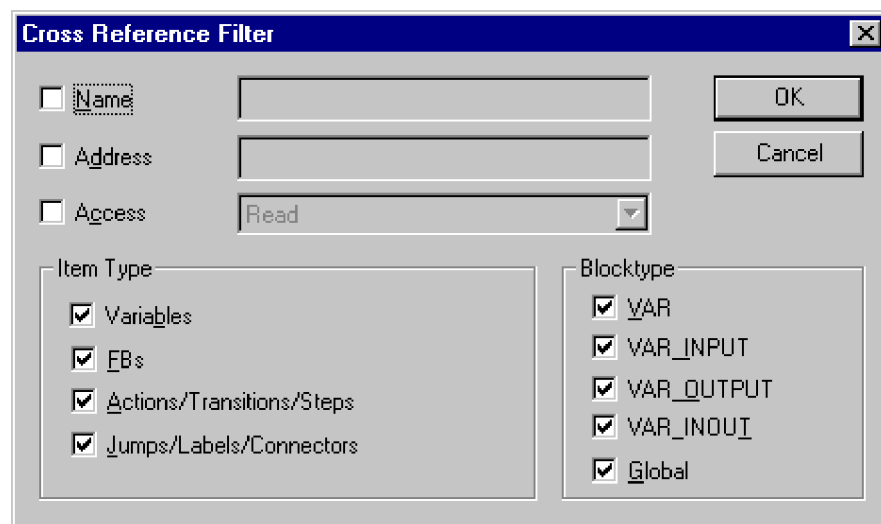


Figure 21: Dialog 'Cross Reference Filter', used to filter the displayed entries in the cross reference list

- Choose the desired filter settings by selecting one or several checkboxes.
- To filter the list for a specific name, address or access, mark the corresponding checkbox in order to activate the text field and enter the desired text. For that purpose you can use wildcards (refer to the note below). Only elements matching this name will be displayed.
- After confirming the dialog the cross reference window contains only elements which match the filter settings. The column 'Variable' indicates in its column head, that filter settings are applied.
- Using **wildcards** in the dialog 'Cross Reference Filter':
In the text fields you can use the wildcard symbol '*'. A wildcard serves as a placeholder for any character and is used in conjunction with other characters. You can use a wildcard at the beginning of an identifier as well as at the end.

Example: Let us assume, that you want to filter the cross reference list in a way, that it displays the variables var1, var2 up to var10. For this purpose enter the string 'var*' in the input field.

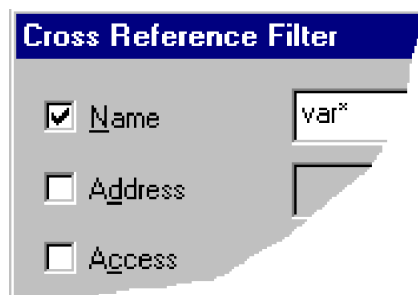


Figure 22: Using the wildcard symbol in the dialog 'Cross Reference Filter'



Sorting the cross reference list

It is possible to sort the list entries alphabetically. Each column can be used as sort criterion. The entries can be sorted in ascending and descending order. To sort the list, perform the following steps:

- Click on the column name, which is intended to be the sort criterion.
It is marked by an arrow, which indicates the sort order:

Variable ▲

Variable ▼

Ascending or descending sort order.

If you have specified a filter string this button would appear as follows:

Variable (Filter active) ▲

This figure corresponds to the described example where the checkboxes 'Local', 'External' and 'FBs' were activated and the string 'var1' was used.

- Click once more on the same column name to reverse the sort order.



Undocking/Docking the cross reference window

Like the output window the cross reference window is dockable. This means, that you can detach the window from the desktop by double clicking in the gray window border. It is then displayed as a usual window, i.e. you can change the size and move the window to any position on the screen. To reattach it into the desk, just double click into the blue window title bar. The handling is similar as for the toolbars.



Using the cross reference toolbar

It is possible to display a toolbar which allows fast access on the references shown in the cross reference list.

To display the cross reference toolbar, call the dialog 'Options' by selecting the menu item 'Options...' in the submenu 'Extras'.

Open the dialog page 'Toolbars', activate the checkbox 'Cross references' and confirm the dialog. The cross reference toolbar is now visible on the desktop:



From left to right the icons have the following meaning:

Go to the first/previous/next/last reference.

3.5.9 Status bar

The status bar displays different messages while you are working with the program.

The left part of the status bar provides information about operations you have done or displays system messages. If your mouse pointer is placed on an icon or a menu item (without selecting it) the status bar displays a short description concerning the icon or menu item under the cursor.

The fields on the right show the cursor position in the editor: When using the graphical editor x-y-coordinates are shown, in the text editor the fields display the current row and column. Beneath the cursor position the free hard disk space is displayed. If there is not enough disk space available, this field appears red.

The different states of the programming system are represented in the status bar by colors. The following colors are used for the different states:

- * gray: offline
- * green: online
- * red: timeout

It is possible to hide the status bar by selecting the menu item 'Status Bar' in the submenu 'View'.

3.6 Using help

3.6.1 Which kind of help files are available?

The context-sensitive help system provides topics for each program part. The help is separated into two or more parts:

- * A general help file describes the general programming system features, which are not PLC-specific.
- * One or more specific help files contain the description of all objects, dialogs and operations which differ from PLC to PLC.
- * Help on functions and function blocks. The software provides full help on all functions and function blocks. These help topics are available for IEC standard functions/FBs, PLC specific and firmware functions/FBs as well as for user defined functions and function blocks (please refer to the section below).

3.6.2 Using the general and the PLC specific help

The general and the specific help are arranged in a hierarchical structure using three types of topics:

- * Main topics: These help topics describe the general handling of the editors. Main topics are called by pressing F1 while an editor is opened.
- * Object topics: These help topics give background information about an object used in an editor. Object topics are called by marking the object in the worksheet and pressing <SHIFT> + <F1>.
- * Dialog topics: These help topics explain the meaning and the usage of the dialog fields. Dialog topics are called by clicking the button 'Help' in the dialog.

All these topics can be called context-sensitive. But it is also possible to call the help contents or help index for getting an overview and then choosing the desired topics. For this purpose you have to select the menu item 'Help Topics' in the submenu 'Help'.

3.6.3 Using help on functions and function blocks

In the Edit Wizard you can call specific help to each function and function block by performing the following steps:

Click with the right mouse button on the specific function or function block in the Edit Wizard selection area. The context menu appears. Select the menu item 'Help on FU/FB'. The related help topics will be displayed.

For standard, firmware and PLC specific functions and function blocks the context sensitive help topic of the file 'Help on FU/FB' is called, which is part of the programming system. If you call help for an user defined function or function block, either the contents of the description worksheet within this POU or the contents of an user edited HTML file is opened in a specific help window, which differs from the standard help window.

How to set up the desired help file for user defined functions and FBs



The procedures to set up the desired help file for user defined functions and function blocks are described in the general programming system help.



Searching a topic in the help system

- Select the menu item 'Help topics' in the submenu '?'. The general help contents appears displaying the contained topic titles with the hierarchical structure.
- If you are searching for an entire topic, click on the tab 'Index' and enter the desired topic title.
- Further information to a specific topic can be called by clicking on the hypertext jumps within the text or at the end of a topic. These jumps to other related topics appear as colored or underlined phrases.
- If you are searching for a specific information or a specific word within the help file, you can use the AnswerWorks feature as follows:
- In the help window click on the button 'Ask me...'. The AnswerWorks dialog appears. Type your request into the text field and click on 'Search'. The help system displays a list of topics, which contain an answer to your request. To display the desired help topic, click on the related jump in the result list.

3.7 Editors

3.7.1 The project manager - a powerful tool for program organization

The project manager is a comfortable and powerful tool for program organization and project management. It includes the project tree editor and the instance tree.

Every project is represented with its own project tree in the project manager. The project tree allows you to edit the structure of your project.



NOTE

It is recommended to leave the project manager open the whole time while you are working with the program, because it gives you a better orientation at what level you are working and what is left to do.

3.7.2 Project tree

You can edit your project within four subtrees. In the subtree 'Library' libraries can be announced. In the subtrees 'Data Types' and 'Logical POU's' new POU's and worksheets can be added for editing data type, variable and code body worksheets. In the subtree 'Physical Hardware' all configuration elements can be inserted, global variables can be declared and programs can be associated to tasks.

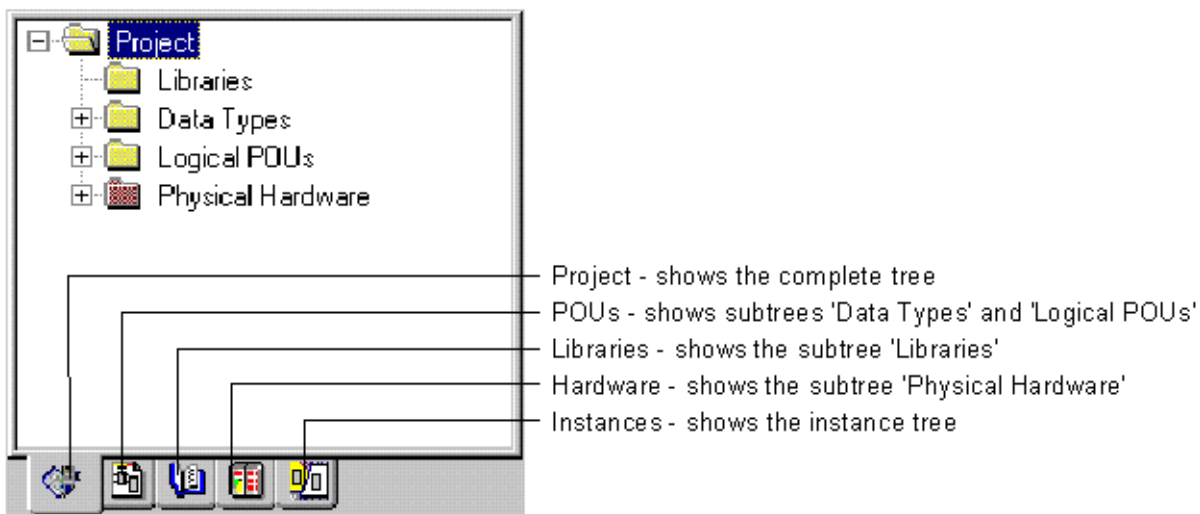


Figure 23: Project tree window with sheet tabs

The project tree allows to select several objects, i. e. you can select several icons and apply a command (such as copying, cutting or deleting icons, printing worksheets, global searching/replacing of text, etc.) on each selected object at the same time. For marking several objects in the project tree you can either keep the <SHIFT> key pressed while clicking on each desired icon (to select adjacent icons) or keep the <CTRL> key pressed (in order to mark icons which are not side by side in the tree).

Detailed information about the project tree functionality



can be found in the general programming system help. Please refer to the topic 'General features of the project tree'.

3.7.3 Instance tree

The instance tree displays the project structure in the PLC and is used to open the worksheets in online mode.

It is possible to display the complete project tree or only particular subtrees. For this purpose the project tree window provides several tabs at it's bottom. They can be used to switch between the various views.

3.7.4 The graphic editor - easy programming in SFC, FBD and LD

The graphic editor is one of the editors which have been implemented for programming graphical code body worksheets of the POUs. The system supports three graphical programming languages: SFC, FBD and LD.

The following features facilitate editing in graphical languages:

- * An Edit Wizard provides full edit functionality for inserting and replacing functions and function blocks.

- * Full drag & drop functionality is provided for moving and copying objects using the mouse.
- * All graphical editors provide simple keyboard operation for inserting and scrolling. The following modes are available:
 - o Object mode by pressing
Cursor keys or <CTRL> + Cursor keys
 - o Mouse cursor mode by pressing
<SHIFT> + Cursor keys or <SHIFT> + <CTRL> + Cursor keys.
- * Duplication of function inputs can be done directly via keyboard, toolbar and menu.
- * Negation of Inputs, Outputs, Contacts and Coils can be done directly via keyboard, toolbar and menu.
- * Easy auto routing for standard editing cases is possible.
- * Items can be inserted directly on a line or to the inputs or outputs of already existing items (only in FBD editor).
- * Splitter and overview windows are available.
- * Freestyle editing allows to arrange items smoothly wherever you want.
- * Grid for adjusting new inserted objects.
- * Jump into the code body worksheets of user defined POUs are possible by double clicking on the corresponding object in the graphical worksheet or by using the object context menu item 'Object Open'.

Detailed information about the graphic editor functionality



can be found in the general programming system help. Please refer to the topics 'Graphic editor' and 'General editing operations in the graphic editor'.

It is possible to mix the graphical programming languages. The system checks all user entries to detect and avoid invalid connections (e.g. connection between two outputs). While inserting new graphical elements the layout of the already existing network is adapted automatically.

3.7.5 The text editor - easy programming in IL and ST

The text editor is used for programming textual code body worksheets of the POUs. The system supports two textual programming languages: IL and ST.

The text editor is also used to edit data type worksheets and variable worksheets.

The handling of the text editor is similar to the handling of a normal ASCII-editor. The following features facilitate editing in IL and ST:

- * An Edit Wizard provides full edit functionality for inserting pre-edited Data types, Operands, Keywords, functions and function blocks.
- * Full drag and drop functionality is provided for moving and copying text elements using the mouse.
- * Optional line numbers can be displayed.
- * Syntax highlighting is possible.
- * Multiple Undo/Redo.
- * Multiple Zooms.
- * Different views via splitter window are possible.

- * Jump into the code body worksheets of user defined function blocks are possible using the context menu item 'Open FB' of the instance name in a textual worksheet.

Detailed information about the text editor functionality



can be found in the general programming system help. Please refer to the topics 'Text editor' and 'General editing operations in the text editor'.

3.7.6 The page layout editor - creating page layouts for printing

The page layout editor provides an easy way to create page layouts for printing the project documentation.

A page layout represents a template which is used to print the contents of a code body worksheet. It defines an area in which the content of the worksheet is going to be printed.

3.8 The Edit Wizard

The Edit Wizard is a useful tool, which facilitates the insertion and replacement of

- * keywords and statements (in ST),
- * operators (in IL),
- * function and function blocks (all languages).

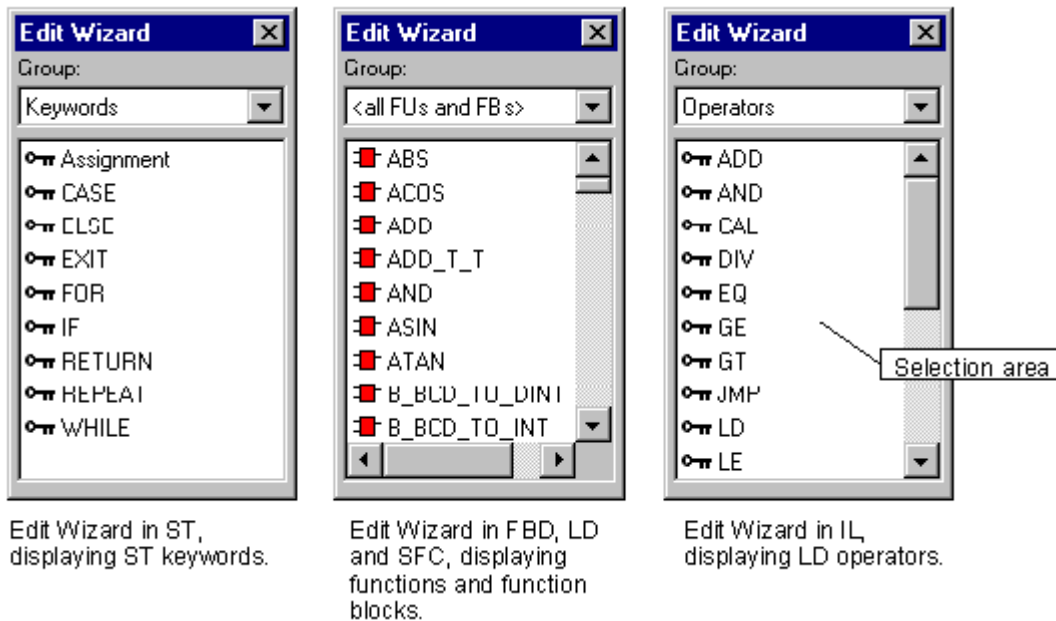
The Edit Wizard for the various languages and worksheets is shown in the following figure.

In general, the Edit Wizard is used as follows: Locate the code body position in your worksheet, where a new keyword, statement, function or function block has to be inserted and place the text cursor or insertion mark at this position. Then determine the desired keyword, statement, function or function block in the Edit Wizard selection area and insert it into the code body with a left mouse double click.

Especially in text editors the usage of the Edit Wizard provides the following advantages:

- * It prevents from entering syntactical faults, such as forgotten semicolons, selection or iteration statements without end statement, etc. This is done by inserting pre-edited statements, functions or function blocks, i. e. the statement structure is already prepared with place holders. The variables and values are inserted as comments, which the user simply has to overwrite.
- * It is not necessary, that the user knows the syntax of all different statement types, such as functions or function blocks.

3.8 The Edit Wizard



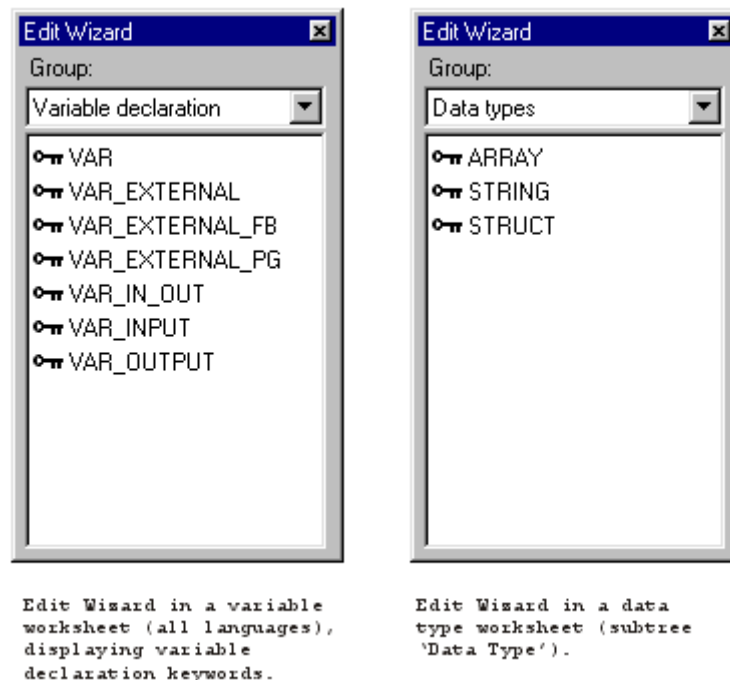
Edit Wizard in ST, displaying ST keywords.

Edit Wizard in FBD, LD and SFC, displaying functions and function blocks.

Edit Wizard in IL, displaying LD operators.

Figure 24: Edit Wizard for the different languages

The Edit Wizard can also be used to insert variable definitions in the various variable worksheets as well as data types in a data type worksheet.



Edit Wizard in a variable worksheet (all languages), displaying variable declaration keywords.





Edit Wizard in a data type worksheet (subtree 'Data Type').

Figure 25: Edit Wizard for variables and data type worksheets

The dockable Edit Wizard window is divided into two areas: The list box 'Group' and the selection area. Depending on the selected group the selection area displays the elements which can be inserted into the code body.

The list box 'Group' contains the available groups of code body elements for the corresponding language. The selection area displays the available elements contained in the selected group.

Depending on the editor, the following groups are provided in the list box:

- * 'Keywords', represented by the key symbol:  (only in ST).
- * 'Operators', represented by the key symbol:  (only in IL).
- * '<all FUs and FBs>', where functions are represented by the symbol  and FBs are represented by the symbol  .
The color of a symbol points to the origin of the related FU or FB: FIRMWARE FUs and FBs are displayed in red. LIBRARY FUs and FBs are displayed in blue. USER FUs and FBs are displayed in green.
- * 'Function blocks'. The selection area only offers function blocks. The symbols and their colors are described above.
- * 'Functions'. The selection area only offers functions. The symbols and their colors are described above.
- * 'PLC specific'. The selection area offers elements only available for a specific PLC.
- * 'PROCESSOR specific'. The selection area offers elements only available for a specific processor.
- * 'String FUs'. The selection area only offers string functions. The symbols and their colors are described above.
- * '*project name*'. One group is designated with the name of the project file. This group only contains the user defined functions and function blocks, which have already been created within the actual project. This is why only green symbols are available.
- * 'Favorites'. This group contains objects which you have added before as favorites.

How to add objects to the 'Favorites' group



For detailed information about 'Favorites' please refer to the corresponding help topic.

- * 'Type conv. FUs'. This group contains type converting functions, such as 'BOOL_TO_INT' or 'BOOL_TO_WORD'.
- * '*Libraries name*'. Each announced library is represented as an own group.

If the Edit Wizard is not visible in the main screen area, perform the following steps:



Calling the Edit Wizard with the mouse

- Click on the icon 'Edit Wizard' in the toolbar.



Calling the Edit Wizard with the keyboard

- Press the shortcut <SHIFT> + <F2>.



Detailed information about the Edit Wizard

The usage of the Edit Wizard is described in detail for each editor in the particular editor description chapter.



Calling help on a function or function block using the Edit Wizard

Before inserting a function or function block into a code body worksheet, you can call help information to the desired object.

For that purpose, mark the function/function block in the Edit Wizard selection area and open its context menu using the right mouse button. Select the context menu item 'Help on FB/FU'.

The context sensitive help topic appears. These help topics are available for IEC standard functions/FBs, PLC specific and firmware functions/FBs as well as for user defined functions and function blocks.



Which help topic appears and how to set up the different help files for user defined functions and function blocks is described in the section Using help in this chapter. Further detailed information are contained in the programming system online help.

3.9 Overview window for graphical worksheets

When using the graphic editor you can call the overview window to get an overview of the complete contents of the worksheet. This simplifies navigation in your worksheet.



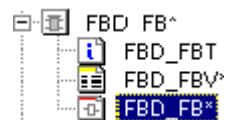
Calling the overview window with the mouse

- Open a graphical worksheet in the graphic editor by double clicking on the desired worksheet icon in the project tree.
- Click on the submenu 'Layout'.
- Select the menu item 'Overview window'.
The overview window appears.



Calling the overview window with its shortcut

- Open a graphical worksheet in the graphic editor. For this purpose mark the desired worksheet icon in the project tree and press <J>.
- Press <ALT> + <9>.
The overview window appears.



The overview window shows the entire contents of the current graphical worksheet. You can use the overview window to move the worksheet area which is visible in the editor.



Navigating in graphical worksheets using the overview window

- Click on the drawing in the overview window. A hand symbol is added to the cursor and the visible area is represented as a rectangle.
- Keep the left mouse button pressed and drag the rectangle to the area you want to be visible in the editor. When releasing the mouse button, the area which is covered by the rectangle is visible in the editor window.

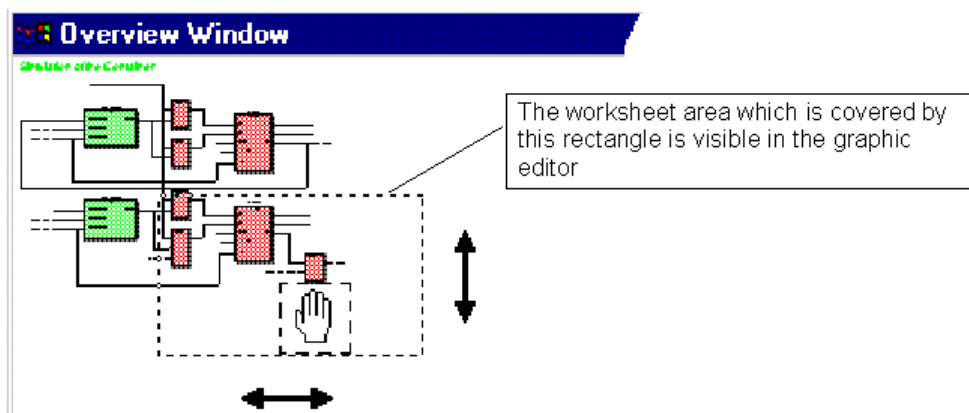


Figure 26: Navigating in a graphical worksheet using the overview window

3.10 Exiting worksheets

If you have finished editing the worksheet you can close the worksheet. A dialog appears where you can save your changes or not.

NOTE



For saving and exiting all opened worksheets choose the menu item 'Close all' in the sub-menu 'Window'.



Exiting the worksheet with the mouse

- Click on the submenu 'File'.
- Select the menu item 'Close'.
The dialog 'PROPROG wt II' appears.



Exiting the worksheet with the keyboard

- Press <CTRL> + <F4>.
The dialog 'PROPROG wt II' appears.



Figure 27: Dialog 'PROPROG wt II'



Using the dialog 'PROPROG wt II'



- Click 'Yes' to exit the worksheet with saving the changes.
- Click 'No' to exit the worksheet without saving the changes.
- Click 'Cancel' to return to the worksheet without exiting.

NOTE

When the worksheet is closed, an automatic compilation will be carried out.



3.11 Exiting the program

If you exit the program, it does not matter whether one or several editors are still open or all windows are already closed. If you have not saved the changes you have done, a dialog appears and you can either save the changes or close the corresponding windows without saving any changes.



Exiting the program with the mouse

- Choose the menu item 'Exit' in the submenu 'File'.
The program is closed.



Exiting the program using the keyboard

- Press <ALT> + <F4>.
The program is closed.



NOTE

To exit the program you can also double click on the icon of the system menu in the upper left window corner or click on the icon 'Close' in the upper right corner of the program window.

4

HANDLING AND EDITING PROJECTS

4.1 Creating a new project

The first step you have to do after calling the program is creating a new project or opening an existing project. This section describes the steps, which are necessary to create a new project.

There are two possibilities for creating a new project: Either by using the Project Wizard (recommended) or by manually selecting a template. Both methods are described in the following subsections.

4.1.1 Creating a new project using the Project Wizard

The easiest way of creating a new project is using the Project Wizard which guides you in 6 steps through the creation process. You can simply define the name and type of the project, the first POU, the configuration, the first resource and the first task.



Calling the Project Wizard with the mouse

- Click on the icon 'New Project' in the toolbar. The dialog 'New Project' appears, as shown in the following figure.
- In the dialog 'New Project' double click on the icon 'Project Wizard'. The dialog 'Project Wizard (Step 1 of 6)' appears.



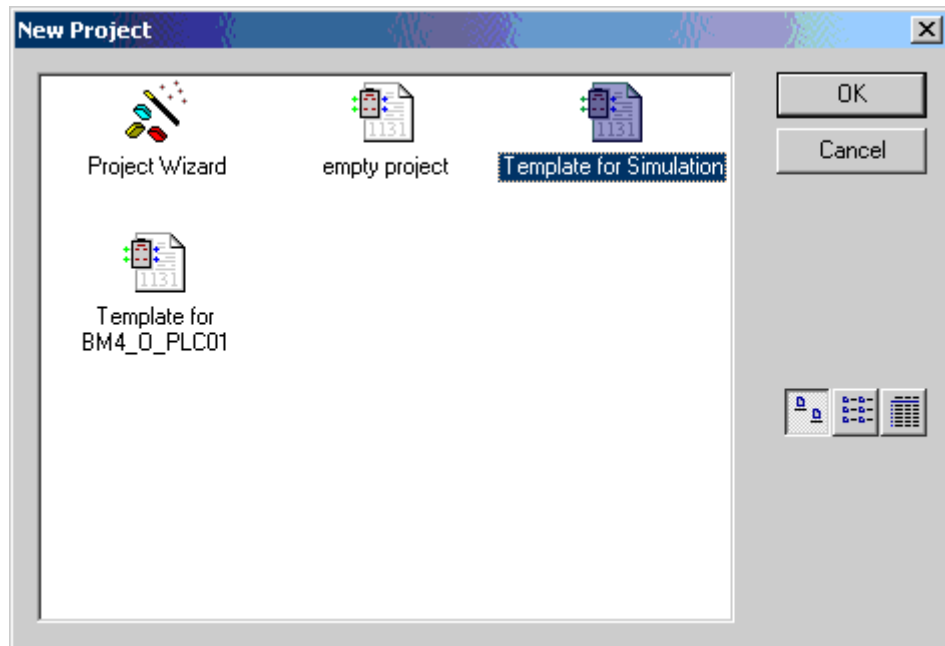


Figure 28: Dialog 'New Project' containing the available project templates and the Project Wizard



Calling the Project Wizard with the keyboard

- Press <CTRL> + <N>.
The dialog 'New Project' appears.
- In the dialog 'New Project' use the cursor keys to mark the icon 'Project Wizard'.
- Press <ENTER> to confirm the dialog.
The dialog 'Project Wizard (Step 1 of 6)' appears.



Using the Project Wizard



- Enter the desired information in the different step dialogs and confirm each step dialog by clicking on the button 'Next'.
- After you have confirmed the last step (6 of 6) by clicking on the button 'Finish', the new project is created automatically and appears in the project tree with the previously entered settings and identifiers.



For detailed information about the various step dialogs please refer to the online help system.

4.1.2 Creating a new project using a template

Using this way for creating a new project the program copies the template you have chosen to a project named 'Untitled'. The template consists of POU's, worksheets or configuration elements necessary for the PLC type.

You can edit the project 'Untitled' and save it afterwards under the name you want to use.



Creating a new project with the mouse

- Click on the icon 'New Project' in the toolbar. The dialog 'New Project' appears (see [▶Figure 28:◀](#) on page 54).



Creating a new project with the keyboard

- Press <CTRL> + <N>. The dialog 'New Project' appears (see [▶Figure 28:◀](#) on page 54).



Using the dialog 'New Project'



- Choose the template for your PLC type with a left mouse click.
- Confirm the dialog. The new project with the name 'Untitled' is created.

Your project tree should look like the following figure now:

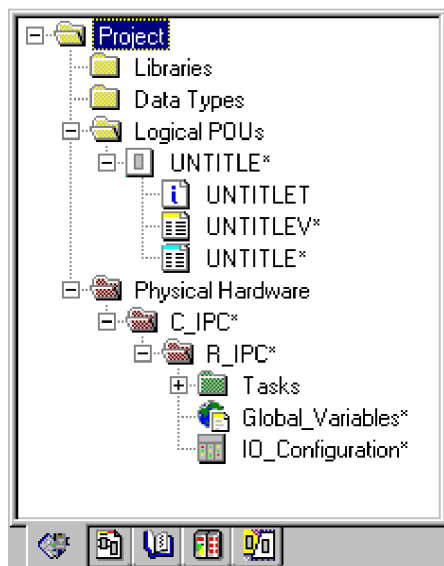


Figure 29: Project 'Untitled' with program 'Untitle' and its worksheets

The project 'Untitled' includes automatically one POU: the program 'Untitle'. The program has 3 worksheets:

4.2 Saving an existing project as a template

- The description worksheet 'UntitledT' for the POU documentation (optional).
- The variables worksheet 'UntitledV' for the declaration of variables and function block instances.
- The code body worksheet 'Untitled' for the code body definition.



NOTE

Some properties of this program can be changed. The default language for the first autoinserted program is IL, thus the language cannot be changed. If a program in another language is required, you have to insert a new POU.

4.2 Saving an existing project as a template

You can save an existing project as a template. After saving a project as template, the new template is contained in the dialog 'New Project' and can be used to create a new project.



Saving an existing project as a template



- Choose the menu item 'Save As Template...' in the submenu 'File'. The dialog 'Save As Template' appears.

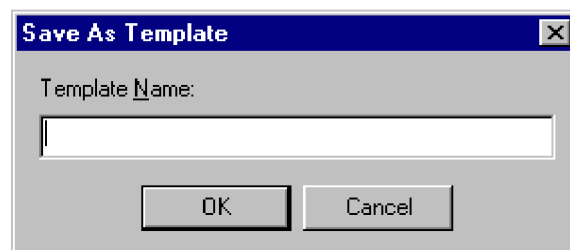


Figure 30: Dialog 'Save As Template'

- Enter the desired name for the new template.
- Confirm the dialog.
The new template is automatically saved in the default template directory.

How to delete an existing template



For detailed information about deleting templates please refer to the online help system.

4.3 Changing the properties of existing POUs

Let us assume that you want to use the program 'Untitled' just with a different name. In this case you have to change the properties of this program.

NOTE



Everywhere in the program you can change the properties of existing objects choosing 'Object Properties...' in the context menu of the object. In order to open the context menu of an object, mark the object and click the right mouse button. The context menu appears. The menu items always relate to the marked object.

Changing the properties of existing POUs with the mouse



- Choose 'Properties...' in the context menu of the icon '*Program name*'. The dialog 'Properties' appears.

Or

- Click with the right mouse button on the icon '*Program name*' in the project tree, to open its context menu.
- Select the menu item 'Properties...'. The dialog 'Properties' appears.



Changing the properties of existing POUs with the keyboard



- Press <↓> or <↑> to mark the icon '*Program name*'.
- Press <ALT> + <↵>. The dialog 'Properties' appears.



4.3 Changing the properties of existing POUs

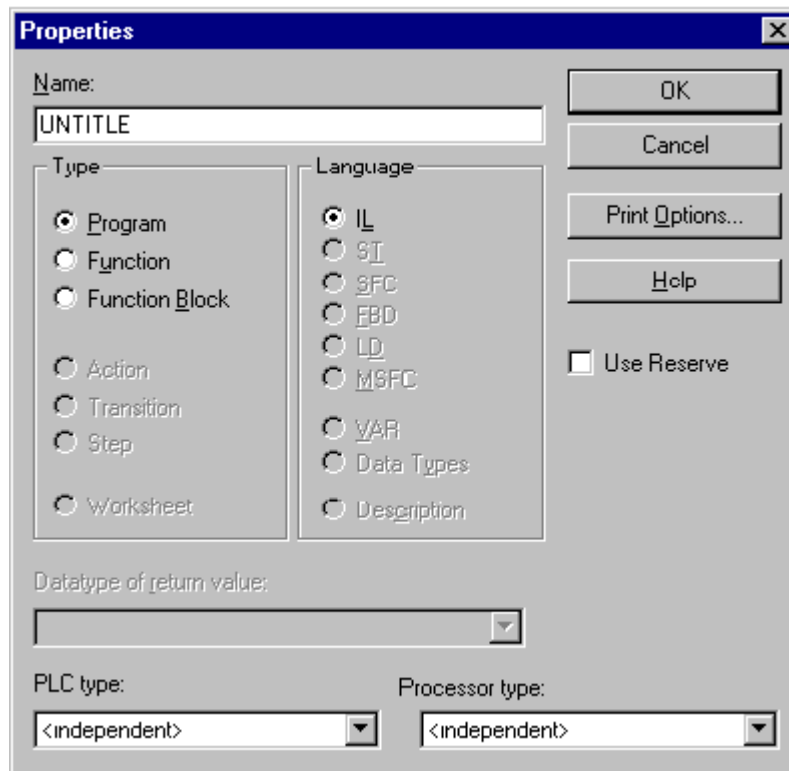


Figure 31: Dialog 'Properties' for changing the properties of existing POUs



Using the dialog 'Properties' for changing the properties of existing POUs



- If desired, enter a new name for the POU.
- If required, change the POU type. The preselected language cannot be changed for this POU.
- Confirm the dialog.



NOTE

You can also edit the name of a particular worksheet in the dialog 'Properties'. To perform this, mark the desired worksheet icon in the project tree, open its context menu using the right mouse button and select the menu item 'Properties...'. The dialog 'Properties' appears as shown above. Enter the new name for the worksheet in the field 'Name' and confirm the dialog.

**NOTE**

If you have changed the name of a program, you also have to change the name of the instantiated program in the task. Otherwise an error occurs when compiling the project, because the system is unable to find the renamed POU.

The steps to be performed for associating a program to a task are described in section 'Associating programs to tasks' in the chapter 'Compiling, downloading and debugging'.

4.4 Inserting new POUs

The next steps you would certainly like to do is to insert new POUs in different programming languages.

**Inserting a POU with the mouse**

- To insert a new program click on the icon 'Add program'. The dialog 'Insert' appears.
- To insert a new function block click on the icon 'Add function block'. The dialog 'Insert' appears.
- To insert a new function click on the icon 'Add function'. The dialog 'Insert' appears.

**Inserting a POU with the keyboard**

- Press <↓> or <↑>, to mark the icon 'Logical POU'.
- Press <INS>. The dialog 'Insert' appears.



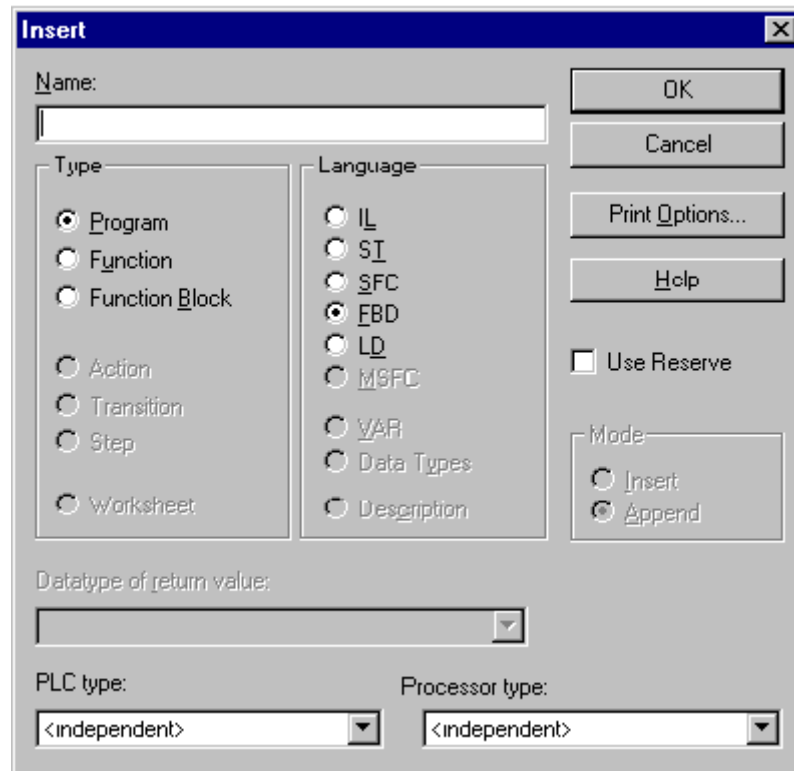


Figure 32: Dialog 'Insert' for inserting a new POU



Using the dialog 'Insert' while inserting a new POU



- Enter a name for your new POU.
- Choose the POU type.
- Choose the programming language. The default language for a new inserted POU is FBD.
- Confirm the dialog.
The new POU with its worksheets is inserted in the project tree.



NOTE

Some programming languages may be grayed according to the number of the available editors.

The new worksheets are marked with an asterisk in the project tree. These asterisks mean that the worksheets have been inserted or changed but not yet compiled.

**NOTE**

If a function block in a POU is not visible or not selectable, this can have the following reasons:

- The POU name is the same as the name of the function block which you want to insert.

Remedy: Change the POU name.

- The PLC type and the processor type of the library doesn't agree with the POU.

Remedy: Link the right library for the target system or select the right template.

- The library is damaged (e.g. subdirectories on the hard disk are deleted).

Remedy: Install the original library from the CD.

4.5 Inserting worksheets

It is also possible to insert new worksheets in POUs. This feature is necessary if you have large code bodies and you want to split them into several parts for better orientation. If you want to insert new worksheets in POUs the language of the new worksheet is determined by the POU language. This means all worksheets of a POU have the same language e.g. FBD. The language of a new inserted worksheet is set automatically.



Inserting a code body worksheet in a POU with the mouse

- Click on the code body worksheet icon where the new worksheet has to be inserted in the project tree.

- Choose the menu item 'Insert' from the context menu.

Or:

Click on the icon 'Add Object'.

In both cases, the dialog 'Insert' appears.

- Enter a name for the new worksheet in the field 'Name'.
- Determine, whether the new worksheet should be inserted after ('Append') or before ('Insert') the marked worksheet.
- Confirm the dialog 'Insert'.

The new code body worksheet is inserted in the project tree.





Inserting a variable worksheet in a POU with the mouse

- Click on the variable worksheet icon where the new worksheet has to be inserted in the project tree.
- Choose the menu item 'Insert' from the context menu.
Or:
Click on the icon 'Add Object'.
In both cases, the dialog 'Insert' appears.
- Click on the radio button 'VAR' in the 'Language' area of the dialog.
- Enter a name for the new worksheet in the field 'Name'.
- Determine, whether the new worksheet should be inserted after ('Append') or before ('Insert') the marked worksheet.
- Confirm the dialog 'Insert'. The new variable worksheet is inserted in the project tree.



The steps to insert a new description worksheet are similar to the steps described for code body or variable worksheets. The only difference is, that you have to mark an existing description worksheet before calling the dialog 'Insert'.

The steps to be done for inserting new data type worksheets in the subtree 'Data Types' are also similar to the steps described for inserting new code body worksheets.



Inserting a data type worksheet with the mouse

- Click on the icon 'Data Types' in the project tree.
- Choose the menu item 'Insert' from the context menu.
Or:
Click on the icon 'Add Object'.
In both cases, the dialog 'Insert' appears.
- Choose a name and confirm the dialog. The new data type worksheet is inserted in the subtree 'Data Types'.



Inserting of worksheets (i. e. all types of worksheets) can also be done using the keyboard.



Inserting a worksheet with the keyboard

- Press <↓> or <↑> to mark the desired icon. Which icon you have to mark depends on the worksheet type you want to insert (refer to the preceding descriptions).
- Press <INS>. The dialog 'Insert' appears.
- Fill in and confirm the dialog 'Insert'.



4.6 Announcing libraries

Having already finished one project, you can reuse these POU's and worksheets in a new project. This feature makes it unnecessary to define code bodies which already exist.

User libraries

To reuse POU's and worksheets of an existing project you have to announce this project as an **user library**. The file extension for user libraries is *.mwt.



NOTE

The worksheets of **user libraries** can be opened in view mode using the corresponding editor. View mode means, that the worksheet can be displayed but not edited.

Firmware libraries

It is also possible to use firmware functions and function blocks which are prepared by your PLC manufacturer. Firmware functions and function blocks have to be announced as **firmware libraries**. The file extension for firmware libraries is *.fwl.



NOTE

Worksheets of firmware libraries cannot be opened, neither in view mode nor in online mode.

In the following section it is described how to announce an user defined library. For announcing firmware libraries just choose the file extension '.fwl'.



Announcing a library with the mouse

- Click on the icon 'Libraries' to mark it.
- Choose the menu item 'Insert' from the context menu.



Or:

Click on the icon 'Add Object'.

In both cases, the dialog 'Include library' appears.



- Choose the project you want to announce as a library.
- Click on the button 'OK' to confirm the dialog.



Announcing a library with the keyboard

- Press <↓> or <↑> to mark the icon 'Libraries'.
- Press <INS>.
The dialog 'Include library' appears.
- Choose the project you want to announce as a library.
- Press <↵> to confirm the dialog.



Your project tree should look like the following figure now:

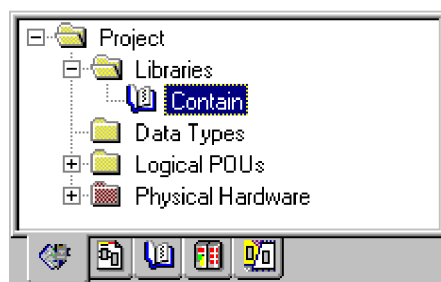


Figure 33: Project with announced library 'Contain'



Opening an user library worksheet in view mode with the mouse

- Mark the library you want to view. If not already opened double click on the 'Libraries' node icon.
- Browse to the desired library and open the POU folder.
- Double click on the desired worksheet icon. The worksheet is opened in the related editor. It cannot be edited.



NOTE



Since user library worksheets can be opened in view mode, it is also possible to switch them into online mode, i. e. it is possible to debug them.



The general procedure of switching library worksheets into online mode is equal to the handling of any other worksheet in your project. Please refer to the section **Calling worksheets in online mode** in the chapter **Compiling, downloading and debugging**.

4.7 Deleting worksheets, POU's or libraries

It is possible to delete worksheets, whole POU's or libraries. In every case you first have to mark the object, which has to be deleted.

For the following description let us assume, that you want to delete the POU 'Untitled'.

NOTE



When deleting a whole POU, keep in mind, that all worksheets contained in this POU are deleted too.

Deleting the POU 'Untitled'



- Click on the POU icon to be deleted.

Or:



Press <↓> or <↑> to mark the POU to be deleted.

- Press .
A message box appears in which you can confirm or abort the deletion.
- Confirm the dialog.
The POU is deleted.



NOTE



Having confirmed the message box there is no way of restoring the data. Use this feature very carefully!

4.8 Saving changes in worksheets while editing

While you are editing you should regularly save the changes you have done. In case of a power failure or other events you risk loss of data if you do not save your work. You can save the changes of the current worksheet at any time.

NOTE



The project tree is saved automatically.

4.9 Saving the existing project under a new name



Saving the changes in the current worksheet with the mouse

- Click on the icon 'Save' in the toolbar.
The worksheet is saved.



Saving the changes in the current worksheet with the keyboard

- Press <CTRL> + <S>.
The worksheet is saved.



NOTE

Autosave feature

The system provides an Autosave feature for unsaved worksheets. To enable this feature and to set the time interval call the dialog 'Options' and open the page 'Backup'. In the related dialog area activate the radio button 'Every' and enter a value.

Automatic compilation when closing

When saving a worksheet, the system automatically compiles this worksheet, i.e. a syntax checking is performed. Any detected errors are displayed in the error sheet of the message window. Additionally the first variables worksheet within the same POU is compiled. Therefore it is important, that the Input/Output parameters used are declared in the first variables worksheet of the POU.

Due to this automatic compilation each user defined function or function block is available in the Edit Wizard immediately after closing the corresponding worksheet.

4.9 Saving the existing project under a new name

The program provides the possibility to save an existing project under a new file name. This means that all files, belonging to the project (such as the project files, the POUs, etc.) are copied to a new project folder. The origin project is not changed.



Calling the dialog 'Save/Zip project as' with the mouse

- Click on the submenu 'File'.
- Select the menu item 'Save Project As/Zip Project as...'.
The dialog 'Save/Zip project as' appears.



Calling the dialog 'Save/Zip project as' with the keyboard

- Press <ALT> + <F>. The submenu 'File' is opened.
- Press <↓> or <↑> to mark the menu item 'Save Project As/Zip Project as...'.
• Press <↵>.
The dialog 'Save/Zip project as' appears.



Using the dialog 'Save/Zip project as'



- In the dialog 'Save/Zip project as' make sure that the entry 'Project Files (*.mwt)' is selected in the list box 'File type' .
- Enter the new project name in the field 'File name'. Use the extension 'mwt'. The entered name is used for the new project folder as well as for the new project file.
- Choose a new path if you want.
- Confirm the dialog.

The entire project is now copied to a new project folder and the copied project file is re-named. After saving the new project, the origin project is closed and the copied project remains opened with an empty workspace.

4.10 Zipping the project files into an archive file

The system provides the possibility to zip a whole project into an archive file using the dialog 'Save/Zip project as'. This zip archive contains then all files which belong to the project, i. e. the project file '*projectname.mwt*' itself, the code body files, variable declarations and some internal files which are necessary to restore the project from the archive. Therefore the zip functionality can be used to create project backups.

NOTE



It is recommended to zip your files regularly, e.g. once a day, and to save your archive on a floppy disk to make sure that no loss of data occurs.

Additionally to the possibility of zipping a project manually into an archive file, the system provides an automatic backup functionality. Using this feature you can create project backups automatically in certain time intervals or when closing a project. For details please refer to the procedure at the end of this section.



Zipping the project files with the mouse

- Click on the submenu 'File'.
- Select the menu item 'Save Project As/Zip Project as...'.
The dialog 'Save/Zip project as' appears.
- In the list box 'File type' select the entry 'Zipped Project Files (*.zwt)'.
The checkboxes 'Zip user libraries', 'Zip frontend code' and 'Zip page layouts' are now available.
- Enter a name for the zip file in the field 'File name'. Use the extension 'zwt' (for example contain.zwt).
- Activate the checkbox 'Zip user libraries' if you want to zip the library too.
- Activate the checkbox 'Zip frontend code' if you want to zip the compilation files too.
- Activate the checkbox 'Zip page layouts' if you want to zip page layouts and bitmaps too.
- Press 'Zip' to start the zipping process.
The progress of the zipping process is displayed in the status bar.
- Confirm the message box after zipping.

NOTE

Zipping projects and libraries



Via the checkboxes (see above) you can decide in detail what you want to zip in the .zwt file:

- „Zip user libraries“
- „Zip FW-Libraries“
- „Zip frontend code“ (should always be activated, if „Zip user libraries“ has been selected)
- „Zip page layouts“

All parts, inclusively of all used libraries and user-designed page layouts of the project are zipped completely, if all checkboxes are activated.

The following procedure should be used with very large projects and daily backups: The libraries, which are used in the project and the actual user code are to be governed separately.

Example:

You want to administrate your project reasonably.

Procedure:

1. The user creates a new project (e.g. „PrjBIB.mwt“) and inserts the same user libraries, FW-libraries there as well as self-created page layouts as in the main project „Prj.mwt“. Then you save the library project as .zwt, whereas you have selected all checkboxes.

Very important: The checkboxes „Zip user libraries“ and „Zip frontend code“ must be selected to zip the library code completely.

2. For the daily backup it is adequate to zip the project „Prj.mwt“ without an activated checkbox, so that the .zwt, which was generated can be minimized regarding its filesize.

Step 1 only must be carried out again, when new libraries are to be added or existing libraries are to be changed.

P.s.: Libraries delivered by Baumuller are dedicated via assigned names (e.g. „UNIVERSAL_20bd00.mwt“) to a defined version (in the example „20bd00“) and don't change. With the next update the version number will be incremented (e.g. „20bd01“). At delivery all Baumuller libraries are zipped with frontend code.

So it is ensured that there is only one library world-wide.



NOTE

Transfer of projects

If an user project is zipped without frontend code but with libraries, the libraries are zipped without code even if „Zip user libraries“ is activated.

To compile the overall project successfully after unzipping on another computer or another PROPROG wt software with different library indices, all there used libraries must saved with frontend code before.

At saving of the project without frontend code, these libraries mustn't be overwritten again.

If „Zip user libraries“ was selected, although „Zip frontend code“ should be activated by the user.



Detailed information about the dialog 'Save project as' can be found in the programming system help. The procedure how to unzip project files is also described in the help.



Activating the automatic backup feature

The system provides an automatic backup feature. This means that you can define, whether the system shall create a project backup each time you close the project or after an user defined time interval has elapsed.

To define these settings,

- call the dialog 'Options' by selecting the menu item 'Options...' in the submenu 'Extras'.
- Open the dialog page 'Backup'.
- In the dialog area 'Backup' select the desired option and confirm the 'Options' dialog.



NOTE

Important:

If the automatic backup feature is activated, the system **does not** overwrite backups which are created manually using the menu item 'Save project as/Zip project as...'. The system rather creates a backup file which is named *project_name_back.zwt*.

If you activate the checkbox „Zip user libraries“, you must although select „Zip frontend code“, otherwise the libraries are not zipped completely and you get „asterisks“ during saving (library not compiled).

4.11 Translating the project language

Assume that a project was developed in German, i.e. all comments and description worksheets are written in this language. Further assume that the maintenance staff has to expand or change this project but is not able to understand German.

Due to the system feature which allows you to translate project comments and descriptions into any language, situations like the above described are no problem any more.

Basically the translation process is divided into four steps.

- 1 Exporting all translatable text strings into a project translation export file with the file extension 'txt'. This file then contains all comments and the contents of the description worksheets.
- 2 Translating the project translation file. The translation can be done manually or by the help of a specific tool, such as a database supported translation tool.
- 3 Selecting the new project language and re-importing the translated txt-file into the programming system.
- 4 Switching the project language in the project properties dialog.

4.11.1 Exporting a project translation file

In order to translate a project into another language, you first have to export all translatable texts into a so called project translation file. Translatable texts are comments and the contents of the description worksheets. Of course it is not possible to translate text elements which are programming language specific.

The translation file is a usual ASCII-file with the extension 'txt', i.e. it can be opened and edited with any ASCII-editor. Therefore it is also possible to use customary translation tools for translating the file into the target language.



NOTE

Before exporting the file, the project default language has to be chosen from the 'Project Properties' dialog. (For further information refer to the chapter "Switching the project language").



Exporting a project translation file

- Open the submenu 'File' and select the menu item 'Export...'.
The dialog 'Import / Export' appears, showing all system add-ins which can be used for exporting files from the project.

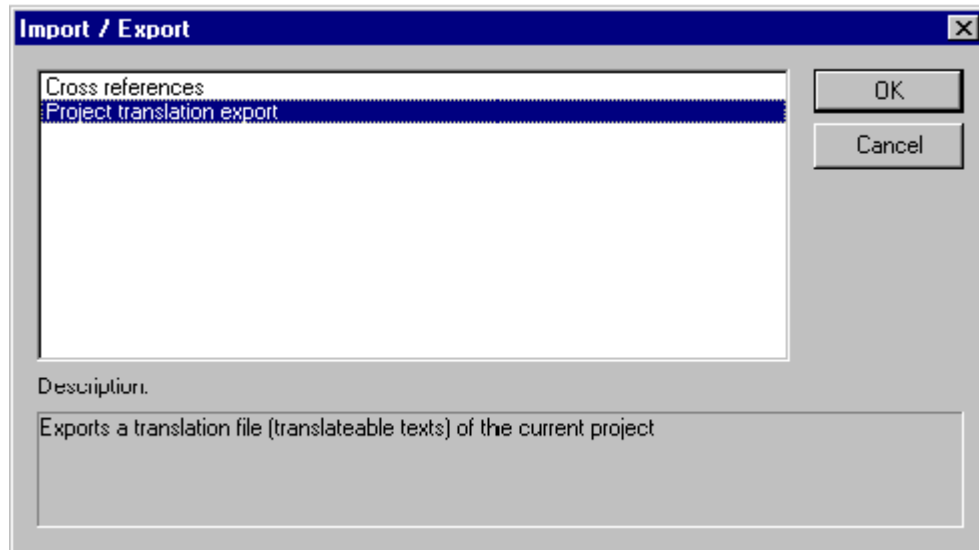


Figure 34: Dialog 'Import / Export' when selecting the menu item 'Export...'

- In the dialog list box mark the entry 'Project translation export' (as shown in the figure) and confirm the dialog.
The browse dialog 'Project translation export' appears.

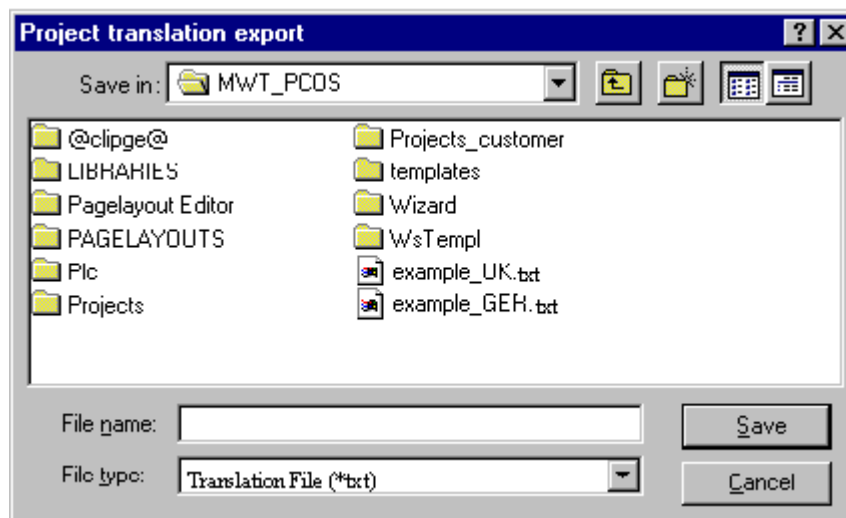


Figure 35: Dialog 'Project translation export'

- In the dialog 'Project translation export' browse to the desired destination directory and enter the name of the export file into the dialog field 'File name'.
- Confirm the browse dialog.
The system exports all translatable texts from the project into the project translation export file.

The next step is to open the export file and translate it either manually or with the help of a translation tool.

4.11.2 Importing a project translation file

When you have translated the project translation file, you first have to select the language of the file to be imported and then import the file into the project, before you can switch the project language.



Importing a project translation file

- Open the submenu 'File' and select the menu item 'Import...'.
The dialog 'Import / Export' appears, showing all system add-ins which can be used for importing files into the project.

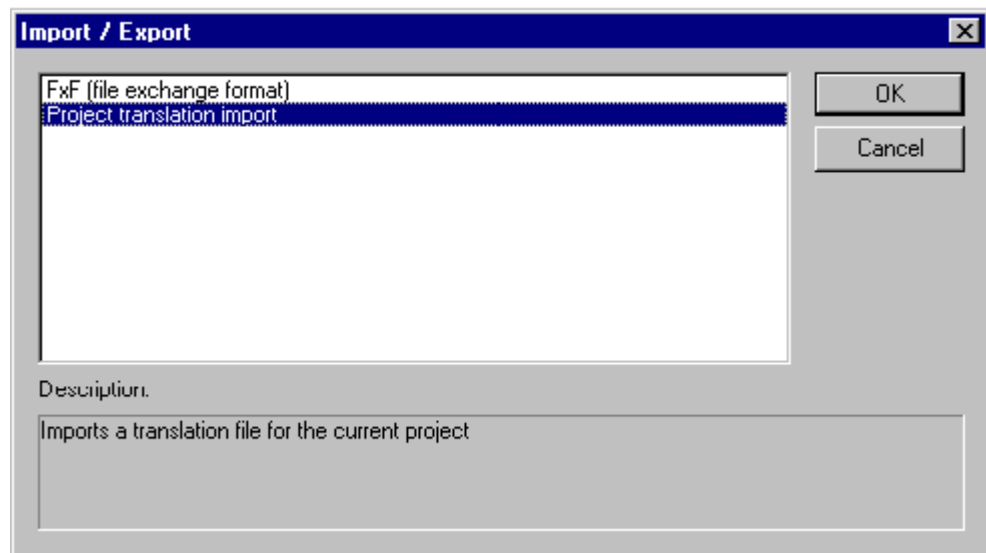


Figure 36: Dialog 'Import / Export' when selecting the menu item 'Import...'

- In the dialog list box mark the entry 'Project translation import' (as shown in the figure) and confirm the dialog.
The browse dialog 'Project translation import' appears.



Figure 37: Dialog 'Project translation import'

- In the dialog 'Project translation import' select the language of the text file to be imported. For that purpose open the list box 'Language' and choose the desired entry.



NOTE

The language selected here is then available in the properties dialog of the project, where you can switch the language.

- Browse to the desired directory and select the desired import file.
- Confirm the dialog.
The project translation file is imported into the project.
The system displays a message after the successful import process.


The language is now available in the project, but it is not switched automatically. The next step is to open the properties dialog and switch the project language.

4.11.3 Switching the project language

When you have imported a project translation file, you can switch the project language.



Switching the project language

- In the project tree open the context menu of the folder icon 'Project' by clicking with the right mouse button on it. 
- In the context menu select the menu item 'Properties...'. The dialog 'Project Properties' appears.

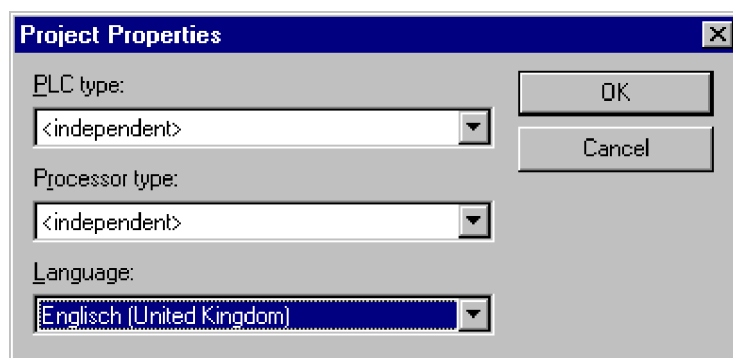


Figure 38: Dialog 'Project Properties'

- In the dialog open the list box 'Language' and select the desired language.



NOTE

The list box only contains languages, which have been selected in the dialog 'Project translation import' before re-importing the translation file.

The project language is switched to the selected language.

4.12 Source conversion for IL, FBD and LD

Additionally to the requirements given by IEC 61131-3, your programming system provides a powerful feature which is called 'Source conversion'. It can be used to cross compile an existing POU which is written in one of the languages IL, FBD or LD into each of the other two languages.

The following table shows the possible source conversions:

Source language	possible target languages
IL	FBD, LD
FBD	IL, LD
LD	IL, FBD

Prerequisite for each source conversion is that the project which contains the POU to be converted was already compiled before (using one of the commands 'Make' or 'Rebuild Project' in the submenu 'Build'). This is necessary, because the system does not convert the textual or graphical language elements which are visible in the corresponding worksheets, but the intermediate code, generated while compiling the project.



Converting the source language of a POU

- Make sure, that the project was already compiled before.
- In the project tree open the subtree 'Logical POUs' and mark the folder icon of the POU to be converted.
- Click with the right mouse button on the icon to open its context menu.
- Select the menu item 'Source conversion'.
The dialog 'Source conversion' appears.

Example:

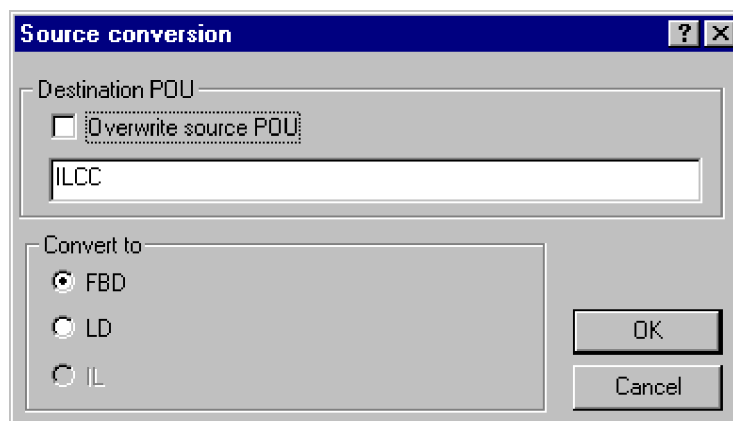
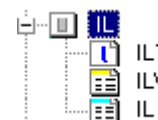


Figure 39: Dialog 'Source conversion'

- In the dialog mark the checkbox 'Overwrite source POU' if desired. If this checkbox is marked, the source POU is going to be overwritten by the converted POU. Otherwise the source POU is not changed and is available for future use.
- Determine the name of the target POU. After opening the dialog, the system automatically proposes a name. You can either leave this name unchanged or enter a new name.
- Mark the desired target language.
- Confirm the dialog.
According to your settings the source POU is converted and the new POU is inserted in the project tree.



NOTE

The new POU is not yet compiled after the conversion.

LITERALS, DATA TYPES AND VARIABLES

5.1 Literals

Literals can be used to enter external representation of data. Literals are necessary for the representation of numeric, character strings and time data. Every time you have to enter values you have to use literals.

5.1.1 Numeric literals

The numeric literals which can be used are shown in the following table:

Type	Examples
Integer literals	-12 0 123_456 +986
Real literals	-12.0 0.0 0.4560 3.14159_26
Real literals with exponents	-1.34E-12 -1.34e-12 1.0E+6
Base 2 literals	INT#2#1111_1111
Base 8 literals	INT#8#377
Base 16 literals	INT#16#FF SINT#16#ff
Boolean FALSE and TRUE	FALSE TRUE
Boolean 0 and 1	0, 1

Figure 40: Numeric literals

NOTE



Literals which are used in variable worksheets, literals of data type INT or BOOL can be used without keyword as it is shown in the following examples:

For INT#16#ff you can use 16#ff.

For BOOL#FALSE you can use FALSE.



NOTE

In variable declarations you can use
"var1 : DINT := 10"
but in the code body you have to use "LD DINT#10" .

5.1.2 Character string literals

A character string literal is a sequence of zero or more characters included in two single quote characters.

The character string literals which can be used are shown in the following table:

Type	Examples
Empty string	"
String with a blank	' '
Not empty string	'this is a text'

Figure 41: Character string literals

5.1.3 Duration literals

Duration data can be represented in hours, minutes, seconds, milliseconds and in combination.

The duration literals which can be used are shown in the following table:

Type	Examples
Short prefix	T#14ms t#14ms t#12m18s3ms T#25h_15m t#25h_15m
Long prefix	TIME#14ms time#14ms TIME#25h_15m time#25h_15m

Figure 42: Duration literals



NOTE

Numbers **after** decimal point are ignored at duration data represented in **milliseconds**, i.e. TIME#15.8ms is interpreted as TIME#15ms

5.2 Introduction to the IEC data types

Data types define the bitsize, the value range and the initial value of a variable. Elementary and user defined data types can be used. Generic data types are basically used for describing the available data types of inputs and outputs of functions.

5.2.1 Elementary data types

The value ranges and the bitsize of elementary data types are described in IEC 61131-3. Elementary data types are shown in the following table:

Data type	Description	Size	Range	Default initial value
BOOL	Boolean	1	0 up to 1	0
SINT	Short integer	8	-128 up to 127	0
INT	Integer	16	-32768 up to 32767	0
DINT	Double Integer	32	-2.147.483.648 up to +2147.483.647	0
USINT	Unsigned short integer	8	0 up to 255	0
UINT	Unsigned integer	16	0 up to 65535	0
UDINT	Unsigned double Integer	32	0 up to 4.294.967.295	0
REAL	Real numbers	32	1.18×10^{-38} up to 3.40×10^{38}	0.0
TIME	Duration	32	about 24 days	t#0s
BYTE	Bit string of length 8	8	16#00 up to 16#FF	0
WORD	Bit string of length 16	16	16#0000 up to 16#FFFF	0
DWORD	Bit string of length 32	32	16#00000000 up to 16#FFFFFFFF	0

Figure 43: Elementary data types



NOTE

The use of data types also depends on your hardware. Please refer to your hardware documentation for information about the supported data types.



NOTE

The data type STRING is also an elementary data type but does not belong to the above mentioned group. Its format in the memory depends on your PLC type.

The data type STRING has the following structure:

Byte 0-1	offset to maximum length (0 corresponds to 80)
Byte 2-3	current length
Byte 4-83	characters
Byte 84	zero terminator

5.2.2 Generic data types

Generic data types are data types which include hierarchical groups of elementary data types. ANY_INT includes the elementary data types DINT, INT, SINT, UDINT, UINT and USINT. If a function can be connected with ANY_INT it means that variables of the data types DINT, INT, SINT, UDINT, UINT and USINT can be handled.

Generic data types are shown in the following table:

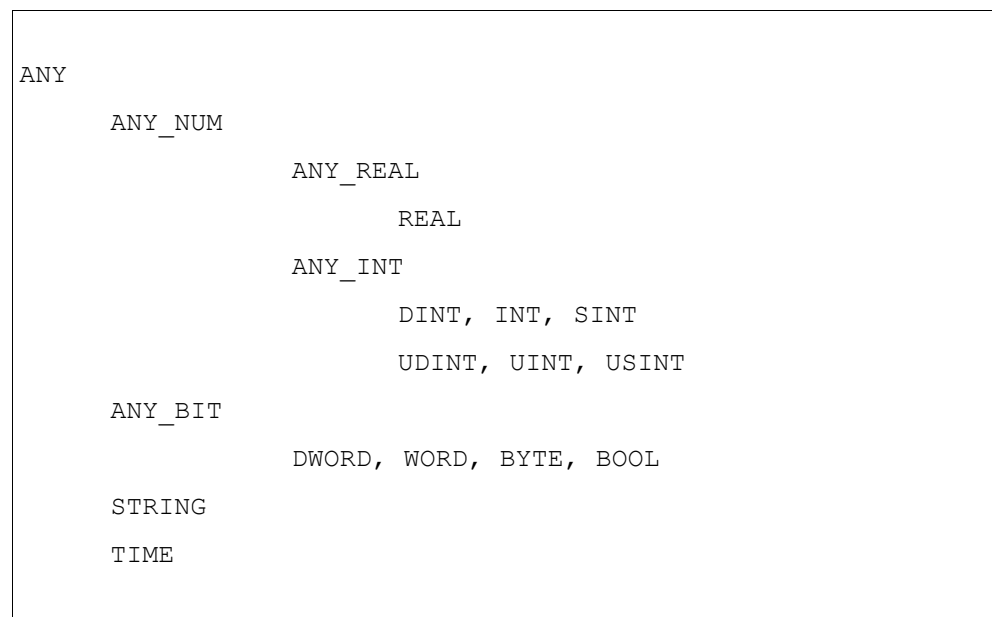


Figure 44: Generic data types



NOTE

The use of data types also depends on your hardware. Please refer to your hardware documentation for restrictions concerning data types.

5.2.3 User defined data types

User defined data types can be declared to create new data types in addition to the elementary data types defined in IEC 61131-3. This is useful because new data types with different properties can be created for an easier and quicker programming. User defined data types are also called derived data types.

Three types of user defined data types can be declared:

- * Array data types
- * Structured data types
- * String data types



Please refer to your PLC documentation for information about available user defined data types and restrictions. It is possible that the PLC does not support all described possibilities of using user defined data types!

5.3 Array data types

5.3.1 Declaring arrays

Array data types include several elements of the same data type. The particular elements within one array are identified and accessed using an array index. An array can be used to declare several elements of the same type with only one line in the type declaration.

```
TYPE
    graph      :      ARRAY[1..23] OF INT;
END_TYPE
```

Figure 45: Type declaration of an array data type

In the example the array 'graph' contains 23 elements of the data type 'INT'. All elements of an array are stored continuously one after another in the PLC memory.

NOTE



In order to declare an array, you first have to declare the array in the type declaration of the data type worksheet. Subsequently you can declare a variable in the variable worksheet of the POU (refer to the following programming example).

5.3.2 Programming example

An array should be typically used for data describing the same subject. Let us imagine that a process changes an input value every 3 seconds. It is necessary to store each of these input values to compare them with set points. All input values are of the same data type. In this case it is useful to declare an array because in the code body declaration the two values can be easily compared using an iteration statement (e.g. a FOR loop). The single components of the array can be accessed using the array index.

```
Type declaration:
TYPE
    graph      :      ARRAY[1..23] OF INT;
    set_point  :      ARRAY[1..23] OF INT;
END_TYPE

Variable declaration:
VAR
    input      :      graph; (* incoming values of the
                           machine *)
    values     :      set_point; (* values to compare
                           with *)
    i          :      INT :=1; (* variable for array
                           index *)
    run        :      BOOL :=TRUE;
    ERROR      :      BOOL;
    timer      :      FB_TIMER; (* declare FB instance *)
END_VAR

Code body declaration in ST:
timer (pt:=t#3s;in:=run);
IF timer.Q THEN (* provide input values to array
                'graph' *)
    input[i] := %IW0; (* assign input value to array *)
    run := 0; (* edge detection to start timer
              again *)
    i := i+1; (* higher array index *)
ELSE
    run :=1; (* count up *)
END_IF;
IF i = 23
    FOR i:=1 TO 23 BY 1 DO
        IF input[i] <> values[i] THEN (* compar-
            ing 'graph' to 'set_point' *)
            ERROR := TRUE;
        END_IF;
    END_FOR;
    i := 1;
END_IF;
```

Figure 46: Programming example of an array data type

5.3.3 Multi-dimensional arrays

If multi-dimensional arrays are needed, arrays of arrays can be used. An example for an array of an array is shown in the following figure:

```
TYPE
    graph      :      ARRAY [1..10] OF INT;
    my_array   :      ARRAY [1..3] OF graph;
END_TYPE
```

Figure 47: Type declaration of an array of array

In an array of an array a single element is accessed using two indexes as shown in the following figure:

```
Variablen declaration:

VAR
    var1      :      my_array;
    var2      :      INT;
END_VAR

Code body declaration in ST:

var2 := var1[1][3];
```

Figure 48: Accessing elements of an array of array

5.3.4 Initializing arrays

Arrays can be initialized, i. e. to each element contained in the array, a value can be assigned. As already mentioned above, each single element is accessed using its index. The initialization of an array can be done while editing the code body declaration.

```
Variable declaration:

VAR
    graph      :      ARRAY [1..10] OF INT;
END_VAR

Code body declaration in ST:

graph[1] :=7;
graph[2] :=1092;
.
.
.
graph[10] :=13;
```

Figure 49: Initializing elements of an array

It is not necessary to initialize all elements of an array. If no initial value is used, the array element is initialized with the default initial value when starting the program execution.

5.4 Structured data types

5.4.1 Declaring structures

Structured data types include several elements of the same or of different data types.

```
TYPE
    machine:
    STRUCT
        x_pos :      REAL;
        y_pos :      REAL;
        depth :      INT;
        rpm   :      INT;
    END_STRUCT;
END_TYPE
```

Figure 50: Declaration of a structured data type

In the example the structure 'machine' consists of 4 components. All components describe the characteristics of the machine.

5.4.2 Programming example

Structures should be used if data describing the same objects have to be declared. For example a drilling machine drills several holes in one work piece. All holes have a position in x and y position on the work piece, a drilling depth and different revolutions per minutes to drill with. The concrete values for the holes are different but the variables which are needed are always the same. In this case it is useful to declare a structure consisting of three components for the position, drilling depth and revolutions per minute. For each hole different values can be assigned to the components. The function block for the drilling process is just working on the same variable being a structure.

5.4.3 Arrays of structures

It is possible to use a structure within arrays as it is shown in the following example:

```
TYPE
  machine:
  STRUCT
      x_pos  :      REAL;
      y_pos  :      REAL;
      depth  :      INT;
      rpm    :      INT;
  END_STRUCT;
  my_array  :      ARRAY[1..10] OF machine;
END_TYPE
```

Figure 51: Declaration of an array of structure

An application example for arrays of structures could be a transfer line with several drilling machines. Via the array index the concrete drilling machine can be accessed and via the structure components the different values for drilling can be assigned.

5.4.4 Structures with arrays

It is possible to use arrays in structures as it is shown in the following example:

```
TYPE
  graph      :      ARRAY[1..10] of INT;
  antrieb:
  STRUCT
      rpm      :      INT;
      inputs   :      IN_BOOL;
      performance :      graph;
  END_STRUCT;
END_TYPE
```

Figure 52: Declaration of a structure of array

5.4.5 Initializing structures

Structures can be initialized by assigning the values to the components while editing the code body declaration. An example is shown in the following figure:

```
Variable declaration:

VAR
  var1      :     maschine;
  first     :      BOOL :=TRUE;
END_VAR

Code body declaration in ST:

IF first THEN
  var1.x_pos := REAL#1.3E+2;
  var1.rpm   := 3000;
  ...
  first := FALSE;
END_IF;
...
```

Figure 53: Assigning values to components of a structure

5.5 String data types

5.5.1 Declaring strings

User defined string data types are strings with a variable number of characters. Declaring a user defined string the length is set in parentheses behind the data type.

5.5.2 Programming example

In the following example the string length is 10 characters:

```
TYPE
    STRING10    :    STRING (10) ;
END_TYPE
```

Figure 54: Declaration of a string data type

5.6 Calling the text editor with the data type worksheet

The first step before editing user defined data types is to call the text editor with the data type worksheet. This step is done using the project tree. Let us assume for the following description that you want to edit the data type worksheet 'Type'. Therefore you have to insert first the data type worksheet 'Type' in the project tree as it is described in the chapter 'Handling and editing projects'.



Calling the data type editor with the mouse

- Double click on the icon 'Data type worksheet'.
The text editor with the data type worksheet appears.



Calling the data type editor with the keyboard

- Press <↓> or <↑> to mark the icon 'Data type worksheet'.
 - Press <↵>.
- The text editor with the data type worksheet appears.



5.7 Editing type declarations using the Edit Wizard

Type declaration of arrays and structures can be done using the Edit Wizard.

After calling the Edit Wizard in a data type worksheet, by pressing <SHIFT> + <F2>, the Edit Wizard is opened. The list box 'Group' contains only one entry: The group 'Data types'.

The selection area contains two keywords: ARRAY, STRING and STRUCT.

You can insert these keywords into your data type worksheet. The Edit Wizard inserts pre-edited constructions, where you just have to replace the green displayed comments by your actual values and variables.



Figure 55: Edit Wizard in a data type worksheet

The following figure shows both, an array and a structure, each inserted using the Edit Wizard. For each keyword, a new declaration block is inserted.



Figure 56: Data type worksheet 'Type' with pre-edited keywords, inserted using the Edit Wizard

NOTE

General information about the Edit Wizard can be found in the chapter 'Getting started'.

**Inserting a structure using the Edit Wizard with the mouse**

- Locate the position, where the new structure is to be inserted in the data type worksheet. Keep in mind, that every 'Wizard inserted' array or structure is included in an own declaration block.
Click the left mouse button to position the text cursor.
- Press <↓> to insert a new line.
- In the list of available keywords in the Edit Wizard double click on the keyword 'STRUCT'. It is automatically inserted at the text cursor position. The actual variables and values are replaced by comments (green text, enclosed by parentheses and asterisks) as shown in [▶Figure 56:◀](#) on page 88.
- Replace the comments by the actual names and values used in your project.

**Inserting a structure using the Edit Wizard with the keyboard**

- Press <↓> or <↑> to move the text cursor to the position, where the new structure is to be inserted in the data type worksheet. Keep in mind, that every 'Wizard inserted' array or structure is included in an own declaration block.
- Press <↓> to insert a new line.
- Press <ALT> + <3> to set your cursor into the Edit Wizard selection area.
- Press <↓> or <↑> to mark the keyword 'STRUCT'.
- Press <↓> to insert the structure. It is automatically inserted at the text cursor position. The actual variables and values are replaced by comments (green text, enclosed by parentheses and asterisks) as shown in [▶Figure 56:◀](#) on page 88.
- Replace the comments by the actual names and values used in your project.

5.8 Symbolic, located variables and directly represented variables

According to IEC 61131-3 variables are used for programming instead of direct addressing inputs, outputs or flags. Symbolic, located or directly represented variables can be declared.

A declaration of a symbolic variable consists of a variable name and a data type. A declaration of a located variable consists of a variable name, the variable location and a data type. A declaration of a directly represented variable consists of the variable location and a data type.



NOTE

Directly represented variables can only be declared in the declaration of global variables or in programs.

In the following figure an example for each variable type is given:

```
VAR
name                               :    data type;
name          AT    %location      :    data type;
              AT    %location      :    data type;
END_VAR
```

Figure 57: Declaration of a symbolic, a located variable and a directly represented variable

The location of the variable consists of a location prefix and a size prefix. Location prefixes are I for inputs, Q for outputs and M for internal memory. Size prefixes are X for single bits, B for byte, W for word and D for double word.



NOTE

Located and directly represented variables are stored at the declared logical address and it is up to the application programmer to check that no memory address is used twice.

5.9 Global and local variables

The scope of each variable which is determined by the use of the variable keyword is limited either to a POU or to the whole project. Therefore two types can be distinguished:

- * Local variables
- * Global variables

If a variable can be used only within a POU it is called local variable. In those cases the variable keywords VAR, VAR_INPUT and VAR_OUTPUT must be used.

If a variable can be used within the whole project it is called global variable. It has to be declared as VAR_GLOBAL in the global declaration and as VAR_EXTERNAL in each POU where it is used.

**NOTE**

It might be useful to declare all I/Os as global variables. In the global variable declaration they should be declared as located variables and in the VAR_EXTERNAL declaration of the POU they should be declared as symbolic variables. The typing effort in case of address changes is minimized doing it this way.

5.10 Retentive variables

Retentive variables are variables whose values are stored even if the power is switched off. In case of a warm start the last value of the variable is going to be used.

Retentive variables are declared using the keyword RETAIN as it is shown in the following example:

```
VAR RETAIN
    var1      :      BOOL      := TRUE;
END_VAR
```

Figure 58: Example for the declaration of a retentive variable

In this example the variable has got the initial value 'TRUE' which is the initial value for a cold start. In case of a warm start the current value of the variable is used.

The keyword RETAIN can be used in combination with the keywords VAR, VAR_OUTPUT and VAR_GLOBAL. It is not possible to declare retentive variables with the keywords VAR_INPUT and VAR_EXTERNAL.

5.11 Initializing variables

According to IEC 61131-3 initial values can be assigned to variables. Initial values can be given to all kind of variables except in VAR_EXTERNAL declarations. If the PLC is started the variable is processed using the initial value.

Initial values have to be inserted at the end of the declaration line of the variable using ':=' as it is shown in the following figure:

```
VAR
name           :      data type      := initial value;
name AT %location :      data type      := initial value;
      AT %location :      data type      := initial value;
END_VAR
```

Figure 59: Declaration and initialization of a symbolic, a located variable and a directly represented variable

The initial value has to fit to the data type. It is not possible to use e.g. the data type BOOL and an initial value '5'. In this case the system displays an error message.

The initial value is optional. If no initial value is used, the variable is initialized with the default initial value of the data type or with the retained value in case of retentive variables.

5.12 Variable declaration keywords

According to IEC 61131-3 different types of variable declarations exist. For each type a different keyword is used as you can see in the following table:

Keyword	Variable type / Explanation
VAR	<p>for internal variables which can be used only within a POU for declaring the instances of function blocks</p> <p>can be used for the declaration of directly represented and located variables in programs</p> <p>can be used for the declaration of symbolic variables</p> <p>can be used with the keyword 'RETAIN' for declaring retentive variables</p>
VAR_INPUT	<p>for variables which are inputs to functions, function blocks and programs</p> <p>to give a value to the POU coming e.g. from another POU</p> <p>its value is only read within the POU</p> <p>can be used only for the declaration of symbolic variables</p>
VAR_OUTPUT	<p>for variables which are outputs to function blocks and programs</p> <p>supplies an output value for e.g. other POUs</p> <p>its value is written within the POU</p> <p>it is also allowed to read the value</p> <p>can be used only for the declaration of symbolic variables</p> <p>can be used with the keyword 'RETAIN' for declaring retentive variables</p>
VAR_IN_OUT	<p>address of the variable is passed by reference</p> <p>the variable can be read or written</p> <p>typically used for complex data types such as strings, arrays and structures.</p>
VAR_EXTERNAL	<p>for global variables in the POU</p> <p>its value is supplied by the declaration of VAR_GLOBAL</p> <p>cannot be initialized</p> <p>its value can be modified within the POU</p> <p>can be used only for the declaration of symbolic variables</p>
VAR_EXTERNAL_PG	<p>for global variables in the program</p> <p>its value is supplied by the declaration of VAR_GLOBAL_PG</p> <p>cannot be initialized</p> <p>its value can be modified within the program</p> <p>can be used only for the declaration of symbolic variables</p>

5.12 Variable declaration keywords

Keyword	Variable type / Explanation
VAR_EXTERNAL_FB	for global variables in the function block its value is supplied by the declaration of VAR_GLOBAL_FB cannot be initialized its value can be modified within the function block can be used only for the declaration of symbolic variables
VAR_GLOBAL	for global variables which can be used in all programs and function blocks of the project can be used for the declaration of directly represented, located and symbolic variables can be used with the keyword 'RETAIN' for declaring retentive variables
VAR_GLOBAL_PG	for global variables which can be used in all programs of the project can be used for the declaration of directly represented, located and symbolic variables can be used with the keyword 'RETAIN' for declaring retentive variables
VAR_GLOBAL_FB	for global variables which can be used in all function blocks of the project can be used for the declaration of directly represented, located and symbolic variables can be used with the keyword 'RETAIN' for declaring retentive variables
END_VAR	to finish a variable declaration block

Figure 60: Table of keywords for variable declaration blocks



NOTE

Global variables have to be declared as VAR_GLOBAL in the global variable declaration of the resource and as VAR_EXTERNAL in the variable declaration of the POU.



NOTE

The keywords VAR_GLOBAL_PG, VAR_GLOBAL_FB, VAR_EXTERNAL_PG and VAR_EXTERNAL_FB are IEC extensions.

5.13 Declaring variables

You have two possibilities to declare variables:

- * Declaring a variable while editing a code body
- * Declaring variables in a variable worksheet using the text editor

The first method means inserting a variable in a code body worksheet which has not been declared before. In this case the dialog 'Variable' with its subdialog 'Automatic Variable Declaration' appears for the automatic declaration of the variable. Confirming this dialog the variable declaration is autoinserted in the variable worksheet and the new variable is inserted in the code body worksheet. This method is described in the corresponding chapters for the programming languages and SFC.

NOTE



If the corresponding variable worksheet is opened (i. e. the worksheet to which the new variable declaration has to be autoinserted), it is closed automatically when the dialog 'Variable' appears.

The second method means declaring variables just typing the declarations in the variable worksheet. To perform this the Edit Wizard can be used to insert the variable declaration keywords (VAR, VAR_INPUT, etc.).

After calling the Edit Wizard in a variable worksheet, by pressing <SHIFT> + <F2>, the Edit Wizard is opened. The list box 'Group' contains only one entry: The group 'Variable declaration'.

The selection area contains the available variable declaration keywords.

You can insert these keywords into your variable worksheet. The Edit Wizard inserts pre-edited constructions, where you just have to replace the green displayed comments by your actual variable declarations.

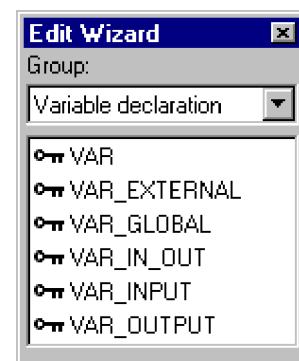
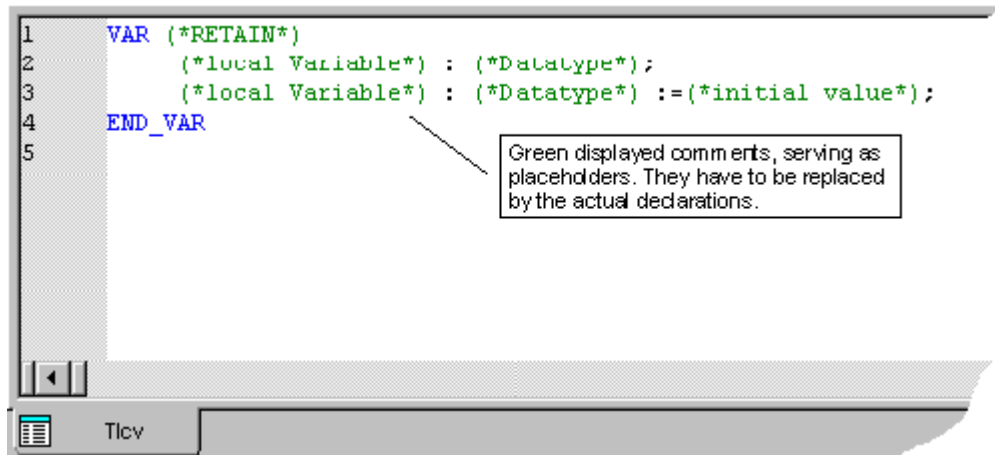


Figure 61: Edit Wizard in a variable worksheet

The following figure shows a VAR declaration inserted using the Edit Wizard. Note, that for each keyword a new declaration block is inserted.



```
1  VAR (*RETAIN*)
2      (*local Variable*) : (*Datatype*);
3      (*local Variable*) : (*Datatype*) :=(*initial value*);
4  END_VAR
5
```

Green displayed comments, serving as placeholders. They have to be replaced by the actual declarations.

Tlcv

Figure 62: Variable worksheet with pre-edited keyword 'VAR', inserted using the Edit Wizard

NOTE



General information about the Edit Wizard can be found in chapter 'Getting started'.



Inserting a VAR declaration using the Edit Wizard with the mouse

- Locate the position, where the VAR declaration is to be inserted in the variable worksheet. This position must not be within another declaration block. Click the left mouse button to position the text cursor.
- Press <↵> to insert a new line.
- In the list of available keywords in the Edit Wizard double click on the keyword 'VAR'. It is automatically inserted at the text cursor position. The actual variable declarations are replaced by comments (green text, enclosed by parentheses and asterisks) as shown in >Figure 62:4 on page 96.
- Replace the comments by the actual declarations.



Inserting a VAR declaration using the Edit Wizard with the keyboard

- Press <↓> or <↑> to move the text cursor to the position, where the VAR declaration is to be inserted in the variable worksheet. This position must not be within another declaration block.
- Press <↵> to insert a new line.
- Press <ALT> + <3> to set your cursor into the Edit Wizard selection area.
- Press <↓> or <↑> to mark the keyword 'VAR'.

- Press <⏏> to insert the keyword. It is inserted automatically at the text cursor position. The actual variable declarations are replaced by comments (green text, enclosed by parentheses and asterisks) as shown in ▶Figure 62:◀ on page 96.
- Replace the comments by the actual declarations.

5.14 Instantiation

IEC 61131-3 provides the possibility of instantiation. Instantiation means that a function block is defined once and can be used several times. As function blocks always have an internal memory it is necessary to store their values for each time the function block is used to a different memory region. This is done using instance names. The instance name is declared in the variable declaration of the POU where the function block is going to be used. In the following figure an example of a variable declaration for the function block 'FB_exam' with two instances is shown:

```
VAR
    drive1      :    FB_exam;
    drive2      :    FB_exam;
END_VAR
```

Figure 63: Instantiation of a function block

The function block 'FB_exam', whose code body has been defined somewhere in the project, has got two instances. The instance name of the first instance is 'drive1', of the second 'drive2'. In the corresponding code body worksheet you can use the function block 'FB_exam' twice, entering in both cases the correct instance name.

NOTE



Instance names are created automatically whenever a function block is inserted using the Edit Wizard.

Function blocks can be instantiated within programs or other function blocks. Functions can be called without instantiation because they do not have an internal memory.

The instance tree shows all instances used in your project as it is shown in the following example:

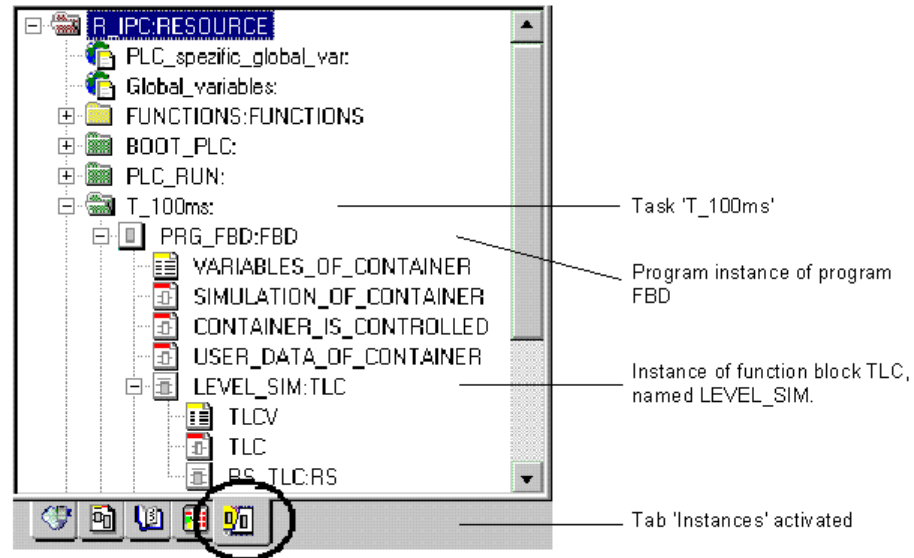


Figure 64: Example of an instance tree

In this example one program instance of the program 'FBD' is used in the task 'T_100ms'. The function block 'TLC' has been instantiated in the program 'FBD'.

The instance tree is made visible by clicking with the left mouse button on the tab 'Instances' at the bottom of the project tree (refer to [Figure 64:◀](#) on page 98).

NOTE



Program instances are created just associating a program to a task and entering an instance name in the corresponding dialog.



Associating programs to tasks is described in the chapter 'Compiling, downloading and debugging' of this manual.

6

EDITING IN ST

6.1 Calling the text editor with a ST worksheet

Before editing a ST code body worksheet you have to call the text editor with the ST worksheet, using the project tree.



A general description of handling the project tree and browsing through POU's and worksheets is contained in the chapters 'Getting started' and 'Handling and editing projects' in this manual.

As an example, let us assume that you want to edit the ST worksheet of a function block which is called ST_FB. For that purpose, you first have to insert a function block with this name as it is described in the chapter 'Handling and editing projects'. Keep in mind that the language of the worksheet is determined by the POU language. It is set by inserting the POU respectively the worksheet.



Calling the text editor for the ST code body worksheet with the mouse

- If the desired worksheet icon is not visible in the project tree, open the corresponding subtree, containing the POU worksheets. For this purpose double click on the POU name (e.g. 'ST_FB').
- Double click on the icon 'Worksheet in ST' of the function block 'ST_FB'. The text editor with the ST worksheet appears.





Calling the text editor for the ST code body worksheet with the keyboard

- If the desired worksheet icon is not visible in the project tree, open the 'ST_FB' subtree, containing the POU worksheets as follows: Press <↓> or <↑> to highlight the function block 'ST_FB'. Press <→> to open the function block subtree.
- Press <↓> or <↑> to mark the desired icon 'Worksheet in ST' of the function block 'ST_FB'.
- Press <↵>.
- The text editor with the ST worksheet appears.



6.2 Introduction to ST

A code body which is programmed in the textual language ST consists of statements and expressions.

Different types of statements can be used while editing.

An expression which is part of a statement returns one value for the execution of the statement. Expressions consist of operators and operands. The operators have to be applied to the operands in the way that the operator with the highest precedence is followed by the operators with the next lower precedence.

Statements and expressions can be entered by just typing them or using the Edit Wizard. The Edit Wizard simplifies editing in the text editors. In ST it contains a lot of standard keywords, functions and function blocks which can be inserted. When entering the statements by typing, it is recommended for a better orientation to start each statement in a new line and to use indents for statements and loops. Each statement has to end with a semicolon. When using the Edit Wizard this is done automatically. Comments can be inserted using asterisks and parentheses. Each line starts with the line number. The syntax highlighting represents the different elements by colors: keywords are blue; variables and instance names are black; comments are green.



The usage of the Edit Wizard is described in the section 'Editing statements using the Edit Wizard' in this chapter.

6.3 Inserting and editing assignment statements

An assignment statement is a specific type of statement. It copies the value of the expression on the right to the variable on the left as it is shown in the following figure:

```
variable name := expression;
```

Figure 65: Structure of an assignment statement in ST

Assignment statements are inserted using the Edit Wizard or by typing them.

For assignment statements it is important that the variable on the left and the value of the expression on the right are of the same data type. If not, a type conversion must be used.

The expression on the right is composed of operands and operators. Operands can be literals, variables, function calls or other expressions. IEC 61131-3 provides a list of possible operators which can be used to connect the operands. These operators have a certain priority. The operator with the highest priority is evaluated first. The list of possible operators is shown in the following table. The operator with the highest priority is explained in the first line, the operator with the lowest in the last.

Operator	Example	Value of example	Meaning
()	(2+3) * (4+5)	45	Parenthesization
**	3**4	81	Exponentiation
-	-10	-10	Negation
NOT	NOT TRUE	FALSE	Complement
*	10*3	30	Multiplication
/	6/2	3	Division
MOD	17 MOD 10	7	Modulo
+	2+3	5	Addition
-	4-2	2	Subtraction
<, >, <=, >=	4 > 12	FALSE	Comparison
=	T#26h = T#1d2h	TRUE	Equality
<>	8 <>16	TRUE	Inequality
&, AND	TRUE & FALSE	FALSE	Boolean AND
XOR	TRUE XOR FALSE	TRUE	Boolean exclusive OR
OR	TRUE OR FALSE	TRUE	Boolean OR

Figure 66: Table of operators in ST

6.4 Inserting and editing further statements

While editing in ST more statements can be used in addition to the assignment statements (such as selection statements, iteration statements or return statements). The keywords of the statements, examples for using the statements and their meanings are shown in the following table.



NOTE

You can use the Edit Wizard for entering statements in ST. In this case their structure is pre-edited.

For entering a statement using the Edit Wizard, perform the steps described in the following section 'Inserting statements using the Edit Wizard'.

Keyword	Example	Meaning
IF	IF a < b THEN c:=1; ELSIF a=b THEN c:=2; ELSE c:=3; END_IF ;	selection statement: A group of statements is executed only, if the associated expression 'a<b' is TRUE. If the condition is FALSE, either no statement is executed or the group of statements following ELSE is executed.
CASE	CASE f OF 1: a:=3; 2..5: a:=4; 6: a:=2; b:=1; ELSE a:=0; END_CASE ;	selection statement: A group of statements is executed according to the value of the expression following the keyword CASE. The variable or expression 'f' must be of the data type INT.
FOR	FOR a:=1 TO 10 BY 3 DO f[a] :=b; END_FOR ;	iteration statement: A group of statements is executed repeatedly increasing the variable 'a' by '3', beginning at '1' and finishing at '10'. The starting point is indicated by the assigned value of the control variable 'a'. The final value is indicated following 'TO' and the increments are indicated following 'BY'. All values must be of the data type ANY_INT. Note: If 'BY' is not indicated, the default value '1' is used. In this case, all values must be of the data type INT.
WHILE	WHILE b > 1 DO b:= b/2 END_WHILE ;	iteration statement: A group of statements is executed repeatedly, until the associated expression 'b>1' is FALSE. The condition of the statement is executed at the beginning of the loop. If the condition is FALSE, the loop is not executed.
REPEAT	REPEAT a := a*b; UNTIL a < 10000 END_REPEAT ;	iteration statement: A group of statements is executed repeatedly until the associated expression is TRUE. The condition of the statement is executed at the end of the loop. If the condition is FALSE, the loop is executed at least once.

Keyword	Example	Meaning
RETURN	RETURN;	return statement: The return statement exits the called function, function block or program and returns to the calling POU.
EXIT	<pre>FOR a:=1 TO 2 DO IF flag THEN EXIT; END_IF; SUM := SUM + a; END_FOR</pre>	exit statement: The exit statement can be used to abort the execution of an iteration statement.

Figure 67: Table of statements in ST

6.5 Inserting statements using the Edit Wizard

As already mentioned above, the most comfortable and fault preventing method to edit the code body (i. e. to insert statements) is to use the Edit Wizard.



NOTE

A general description of the Edit Wizard can be found in the section 'The Edit Wizard' in chapter 'Getting started' in this manual.

In ST worksheets the Edit Wizard contains keywords, functions and function blocks, which can be easily inserted into the code body. Using the Edit Wizard provides the following advantages:

- * It prevents from entering syntactical faults, such as missing semicolons, selection or iteration statements without END keywords, etc. This is done by inserting pre-edited statements, functions or function blocks, i. e. the statement structure is already completed by place holders. The specific variables and values are inserted as comments, which the user may simply overwrite.
- * It is not necessary, that the user knows the syntax of all different statements, such as functions or function blocks.

If the Edit Wizard is not visible in the workspace, perform the following steps:



Calling the Edit Wizard with the mouse

- Click on the icon 'Edit Wizard' in the toolbar.
The Edit Wizard window appears.



Calling the Edit Wizard with the keyboard

- Press <SHIFT> + <F2>.
The Edit Wizard window appears.

6.5 Inserting statements using the Edit Wizard

The following procedure describes the steps how to insert a CASE statement into the code body using the Edit Wizard with the mouse. It is assumed, that the Edit Wizard is visible. Otherwise, call the Edit Wizard as described above.



Inserting a CASE statement using the Edit Wizard with the mouse

- Locate the code body position, where the new statement is to be inserted. Click the left mouse button to position the text cursor.
- Press <_> to insert a new line.
- Open the Edit Wizard list box 'Group' and select the group 'Keywords'. The available keywords are displayed in the selection area of the Wizard.
- Locate the keyword 'CASE'.
- Double click on the CASE statement. It is inserted automatically at the text cursor position as shown below. The actual variables and values are replaced by comments (green text, enclosed by parentheses and asterisks).

```
CASE (^EXPRESSION (must return an INT value^)) OF
(* VALUE a*): (*STATEMENTS*); (* VALUE can be a single value*)
(* VALUES b*): (*STATEMENTS*); (* or a set of VALUES *)
(* . : . *) (* for Example: *)
(* . : . *) (* 1 : ....; *)
(* VALUE x*): (*STATEMENTS*); (* 2..4: ....; *)
ELSE (*STATEMENTS*);
END_CASE;
```

Figure 68: Pre-edited CASE statement, inserted using the Edit Wizard

The next step is to overwrite the comments, which serve as place holders with the necessary variables and values.



Editing the new inserted statement

- Mark the green colored place holders and replace them by statements and expressions (variables and values).



6.6 Inserting variables

While editing in ST there are three possibilities for using variables:

- * Inserting variables, which have already been declared in the variable worksheet of the POU.
- * Inserting variables, which have not been declared before, and declaring them afterwards in the variable worksheet of the POU.
- * Inserting a variable and declaring it while editing.

In the first two cases, the variable is inserted in the code body worksheet by just typing the variable name.

In the following section, the last case is described. Let us assume that you want to insert the variable 'T_value' in your code body worksheet.



Inserting a variable with the mouse

- Type the variable at the desired code body position.
- Mark it with a left mouse double click on the variable name.
- Click on the icon 'Variable' in the toolbar.
The dialog 'Variable' appears. The variable name is displayed in the field 'Variable list of POU *POUname*'.

```
T_value := 100;
```

```
T_value := 100;
```



Inserting a variable with the keyboard

- Type the variable at the desired code body position.
- Mark the variable name by holding the <SHIFT> key and pressing <→> repeatedly.
- Press <F5>. The dialog 'Variable' appears. The variable name is displayed in the field 'Variable list of POU *POUname*'.

```
T_value := 100;
```

```
T_value := 100;
```

NOTE



It is also possible to open the dialog 'Variable' without having marked the variable to be declared. In this case the dialog is opened with the empty field 'Variable list of POU *POUname*'.

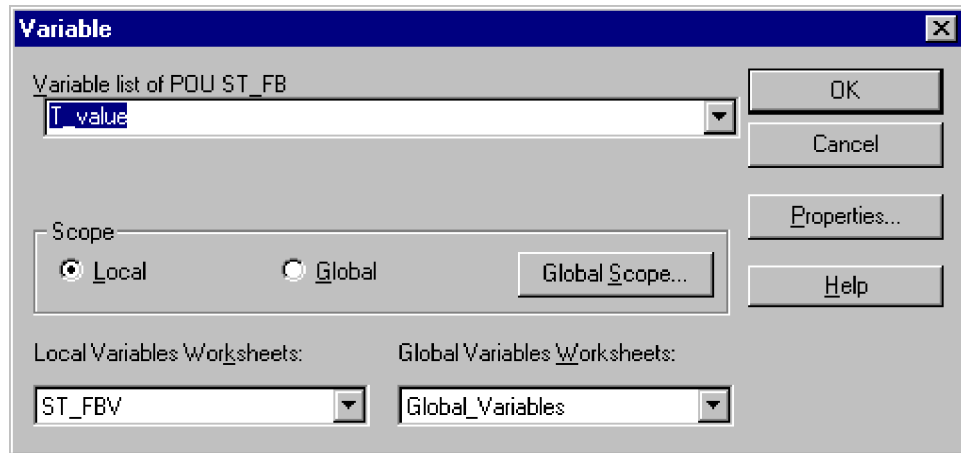


Figure 69: Dialog 'Variable'



Using the dialog 'Variable'



- * Enter a variable name if not already shown (refer to the note above).
- * Confirm the dialog 'Variable'.
The dialog 'Automatic Variables Declaration' appears.

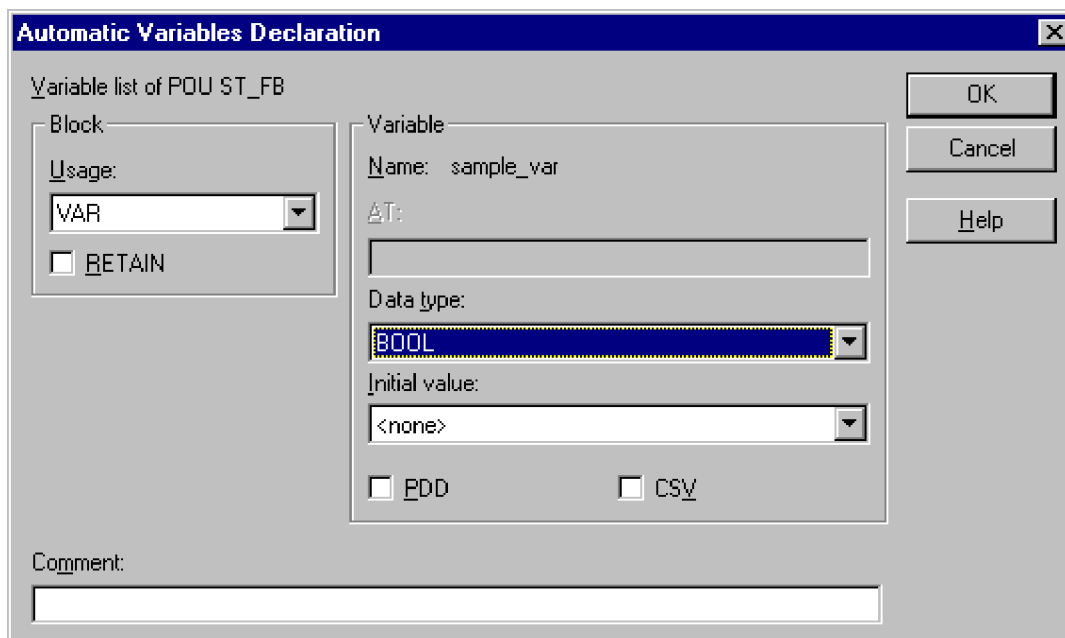


Figure 70: Dialog 'Automatic Variables Declaration'



Using the dialog 'Automatic Variables Declaration'



- Choose a variable keyword in the list box 'Usage'.
- Enter a location in the field 'AT' if you want to declare a located variable (only possible in programs or for global variables).
- Choose the correct data type in the field 'Data type'.
- If required, enter an initial value.
- Mark the checkbox 'PDD' if the variable should be stored in the OmegaOS PDD (Process Data Directory).
- Mark the checkbox 'CSV' if the variable should be stored in the CSV file, i.e. is intended to be used with the OPC Server. The OPC Server processes only variables, which are declared in the CSV file, in order to be used in an OPC client process (e. g. a visualization).
- Enter a comment if you want.
- Confirm the dialog.
The new variable is inserted in the code body worksheet and the declaration of the variable is autoinserted in the variable declaration of the POU.

NOTE



When inserting variables, which have already been declared before, the name of the variable appears in the listbox in the dialog 'Variable'. Confirming the dialog, the variable is directly inserted in the code body worksheet. The dialog 'Automatic Variables Declaration' does not appear.



Detailed information about the OPC Server can be found in the 'OPC Server Manual'.

6.7 Calling functions or function blocks using the Edit Wizard

Functions can be used in any expression e.g. within assignment statements as it is shown in the following figure:

```
variable name := function name(invar1, invar2);
```

Figure 71: Function call in ST

In the example the value of the function on the right is copied to the variable on the left. Instead of the variables 'invar1' and 'invar2' also constants can be used. In this way, either standard functions or user defined functions can be called.

Function blocks can be called as a single statement using the instance name, as it is shown in the following figure:

```
instance(invar1:=1, invar2:=2);  
a:= instance.outvar1;
```

Figure 72: Function block call in ST

The order of the formal parameters does not have any importance. If parameters are missing, the initial value or the value of the last call is used. There is no difference in calling either standard function blocks or user defined function blocks.



NOTE

Do not forget to declare the instance of the function block in the variable declaration of the POU, as it is described in the section 'Instantiation' in chapter 'Literals, variables and data types'.

The most comfortable and fault preventing method to insert function or function blocks into the code body is to use the Edit Wizard.

If the Edit Wizard is not visible in the workspace, perform the following steps:



Calling the Edit Wizard with the mouse

- Click on the icon 'Edit Wizard' in the toolbar.
The Edit Wizard window appears.



Calling the Edit Wizard with the keyboard

- Press <SHIFT> + <F2>.
The Edit Wizard window appears.

The following procedures describe the steps how to insert a MAX function and a CTU function block into the code body using the Edit Wizard with the mouse. It is assumed, that the Edit Wizard is visible. Otherwise, call the Edit Wizard as described above.



Inserting a MAX function using the Edit Wizard with the mouse

- Locate the code body position, where the new function is to be inserted. Click the left mouse button to position the text cursor.
- Press <⏏> to insert a new line.
- Open the Edit Wizard list box 'Group' and select the group 'Functions'. The available functions are displayed in the selection area of the Wizard.
- Locate the function 'MAX'.
- Double click on the function 'MAX'. It is inserted automatically at the text cursor position as shown in the following figure. Replace the green comments (enclosed by parentheses and asterisks) with the necessary elements.



Inserting a CTU function block using the Edit Wizard with the mouse

- Locate the code body position, where the new function block is to be inserted. Click the left mouse button to position the text cursor.
- Press <⏏> to insert a new line.
- Open the Edit Wizard list box 'Group' and select the group 'Function blocks'. The available function blocks are displayed in the selection area of the Wizard.
- Double click on the function block 'CTU'. The dialog 'FB Instances' appears. The field 'FB Instances' displays the default instance name (e.g. for a CTU the name 'CTU_n' is proposed, where n is the first available number which is free for this instance name). To define the name of the new FB you have the following possibilities:
 - Enter a new instance name in the field.
 - Accept the proposed name.
 - Select an already existing name in the list box 'FB Instances'.
- Press 'OK' to confirm the dialog. If you have entered a new instance name, the dialog 'Automatic FB Declaration' appears.
- Enter a comment if you want and press 'OK' to confirm the dialog. The function block 'CTU' is inserted automatically at the text cursor position as shown in the following figure. Replace the green comments (enclosed by parentheses and asterisks) with the necessary elements.

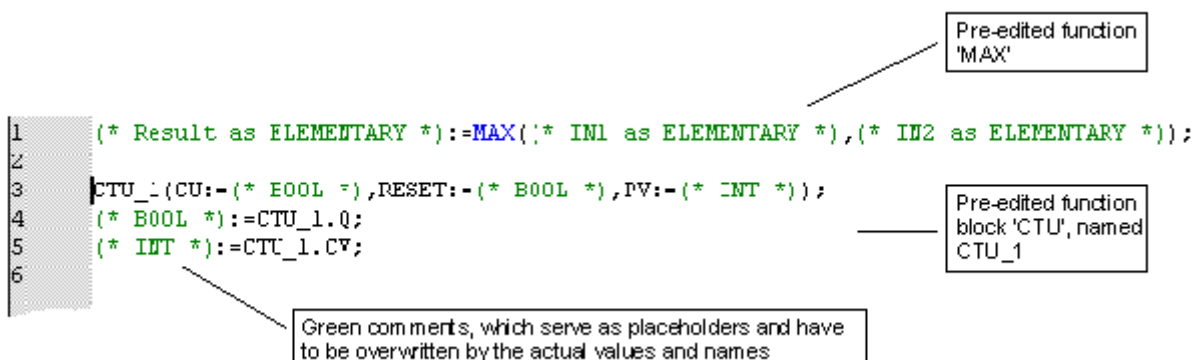


Figure 73: Figure 6-9: Pre-edited MAX function and CTU function block, inserted using the Edit Wizard

7

EDITING IN IL

7.1 Calling the text editor with an IL worksheet

The first step before editing an IL code body worksheet is to call the text editor with the IL worksheet, using the project tree.



A general description of handling the project tree and browsing through POU's and worksheets is contained in the chapters 'Getting started' and 'Handling and editing projects' in this manual.

As an example, let us assume that you want to edit the IL worksheet of a function block which is called IL_FB. For that purpose, you first have to insert a function block with this name as it is described in the chapter 'Handling and editing projects'. Keep in mind, that the programming language of the worksheet is determined by the POU language. It is set by inserting the POU respectively the worksheet.



Calling the text editor for the IL code body worksheet with the mouse

- If the desired worksheet icon is not visible in the project tree, open the corresponding subtree, containing the POU worksheets. For this purpose double click on the POU name (e.g. 'IL_FB').
- Double click on the icon 'Worksheet in IL' of the function block 'IL_FB'. The text editor with the IL worksheet appears.





Calling the text editor for the IL code body worksheet with the keyboard

- If the desired worksheet icon is not visible in the project tree, open the 'IL_FB' subtree, containing the POU worksheets as follows: Press <↓> or <↑> to highlight the function block 'IL_FB'. Press <→> to open the function block subtree.
 - Press <↓> or <↑> to mark the desired icon 'Worksheet in IL' of the function block 'IL_FB'.
 - Press <↵>.
- The text editor with the IL worksheet appears.



7.2 Instructions, operators, modifiers and operands

An instruction list consists of several instructions. Each instruction starts at a new line and contains an operator. Modifier and operand are optional. A comment can be entered at the end of the line using parentheses and asterisks. A line number is displayed in front of each line. The syntax highlighting represents the different elements by colors: operators and their modifiers are blue; variables and operands are black; comments are green. An instruction list with three instructions is shown in the following example:

```
LD    %IX2.2    (* input value *)
ADD   value     (* add value *)
ST    %QX2.2    (* result of addition *)
```

Figure 74: Example of an instruction list

In the example each instruction consists of an operator, an operand and a comment. While editing in IL the following operators, modifiers and operands can be used:

Operator	Modifier	Operand	Description
LD	N	ANY	Set current result equal to operand
ST	N	ANY	Store current result to operand location
S		BOOL	Set Boolean operand to 1 if the current result is 1
R		BOOL	Set Boolean operand to 0 if the current result is 1
AND	N,(ANY_BIT	Boolean AND
OR	N,(ANY_BIT	Boolean OR
XOR	N,(ANY_BIT	Boolean exclusive OR
ADD	(ANY_NUM	Addition

Operator	Modifier	Operand	Description
SUB	(ANY_NUM	Subtraction
MUL	(ANY_NUM	Multiplication
DIV	(ANY_NUM	Division
GT	(ANY_NUM + ANY_BIT	Comparison: >
GE	(ANY_NUM + ANY_BIT	Comparison: >=
EQ	(ANY_NUM + ANY_BIT	Comparison: =
NE	(ANY_NUM + ANY_BIT	Comparison: <>
LE	(ANY_NUM + ANY_BIT	Comparison: <=
LT	(ANY_NUM + ANY_BIT	Comparison: <
JMP	C, N	LABEL	Jump to label
CAL	C, N	NAME	Call function block
RET	C, N		Return to the calling function block or program
)			Evaluate deferred operation

Figure 75: Table of operators, modifiers and operands in IL

The meaning of the modifiers is described in the following table:

Modifier	Description
N	negated
(process the included expression first
C	conditional

Figure 76: Table of the modifiers in IL and their meaning

7.3 Inserting instructions using the Edit Wizard

Instructions can be inserted by just typing them or using the Edit Wizard. The Edit Wizard is a feature to simplify editing in the text editor. In IL it contains a lot of standard operators, functions and function blocks which can be inserted.



A general description of the Edit Wizard can be found in the section 'The Edit Wizard' in chapter 'Getting started' in this manual.

Using the Edit Wizard provides the following advantages:

7.3 Inserting instructions using the Edit Wizard

- * It prevents from entering syntactical faults, such as wrong instruction sequences. This is done by inserting pre-edited operators, functions or function blocks, i. e. the structure is already completed by placeholders. The specific variables and values are inserted as comments, which the user simply has to overwrite.
- * It is not necessary, that the user knows the syntax of all different operations, such as functions or function blocks.

Example: After inserting the operator 'ADD' and the function block 'RS' using the Edit Wizard, the following instruction list is visible in your IL worksheet:

```
1 LD (* IN1 as ANY_NUM *)
2 ADD (* IN2 as ANY_NUM *)
3 ST (* Result as ANY_NUM *)
4
5 LD (* BOOL *)
6 ST RS_1.SET
7 LD (* BOOL *)
8 ST RS_1.RESET1
9 CAL RS_1
10 LD RS_1.Q1
11 ST (* BOOL *)
12
```

Pre-edited operator 'ADD'

Pre-edited function block 'RS', named RS_1

Green comments, which serve as placeholders and have to be overwritten by the actual values and names

Figure 77: Operator 'ADD' and FB 'RS', both inserted using the Edit Wizard

If the Edit Wizard is not visible in the workspace, perform the following steps:



Calling the Edit Wizard with the mouse

- Click on the icon 'Edit Wizard' in the toolbar.
The Edit Wizard window appears.



Calling the Edit Wizard with the keyboard

- Press <SHIFT> + <F2>.
The Edit Wizard window appears.

The following procedure describes the steps how to insert an ADD operator into the code body using the Edit Wizard with the mouse. It is assumed, that the Edit Wizard is visible. Otherwise, call the Edit Wizard as described above.



To insert functions or function blocks into the code body, the same procedure must be performed with the exception of selecting the appropriate function or function block group in the Edit Wizard. This procedure is described in detail in section 'Calling functions or function blocks using the Edit Wizard' in this chapter.



Inserting an ADD operator using the Edit Wizard

- Locate the code body position, where the new instructions are to be inserted. Click the left mouse button to set a text cursor.
- Press <↵> to insert a new line.
- Open the Edit Wizard list box 'Group' and select the group 'Operators'. The available operators are displayed in the selection area of the Wizard.
- Locate the operator 'ADD'.
- Double click on the operator. It is inserted automatically at the text cursor position as shown in >Figure 77:◀ on page 114. The actual variables and values are replaced by comments (green text, enclosed by parentheses and asterisks).

The next step is to overwrite the comments, which serve as place holders with the necessary variables and values.



Editing the new inserted instructions



- Mark the green colored place holders and replace them by the actual operands (variables and values).
- Now you have to declare the variables which have been inserted with the operator. To perform this, double click on the variable to mark it. Then press <F5> to open the dialog 'Variable'. You can also change the names of the variables if necessary.

7.4 Inserting variables

If you are editing IL instructions manually by just typing them you have two possibilities to insert variables. The first possibility is typing the variable name in the code body worksheet and declaring the variable using the variable editor or the variable dialog.

You can also insert variables by clicking on the icon 'Variable' in the toolbar and using the dialog 'Variable'. In a second example at the end of this section, we assume, that you want to insert a variable which has already been declared in the global variable declaration of your project.

In the first example let us assume, that you have inserted the operator 'ADD' using the Edit Wizard. The required variable 'invar' is not yet declared in the variable worksheet 'IL_FBV'.



Declaring a variable with the mouse

- Double click on the variable name to be declared.
- Click on the icon 'Variable' in the toolbar.
The dialog 'Variable' appears. The field 'Variable list of POU *POUname*' displays the current *variable name*.

ADD invar





Declaring a variable with the keyboard

- Mark the variable name by holding the <SHIFT> key and pressing <→> **ADD** **invar**
- Press <F5>. The dialog 'Variable' appears. The field 'Variable list of POU *POUname*' displays the current *variable name*.



NOTE

It is also possible to open the dialog 'Variable' without having marked the variable to be declared. In this case the dialog is opened and the field 'Variable list of POU *POUname*' is empty.

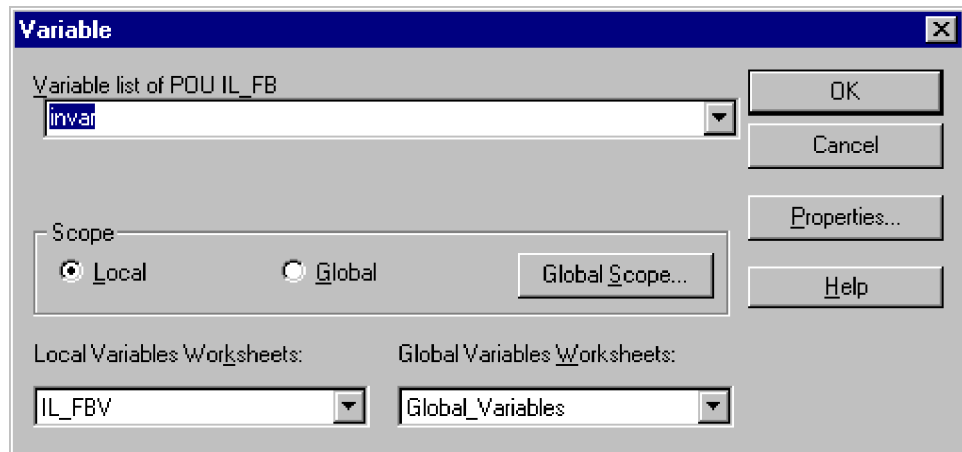


Figure 78: Dialog 'Variable', called with a variable marked



Using the dialog 'Variable'



- Enter a variable name if not already shown (refer to the note above).
- Confirm the dialog 'Variable'.
The dialog 'Automatic Variables Declaration' appears.

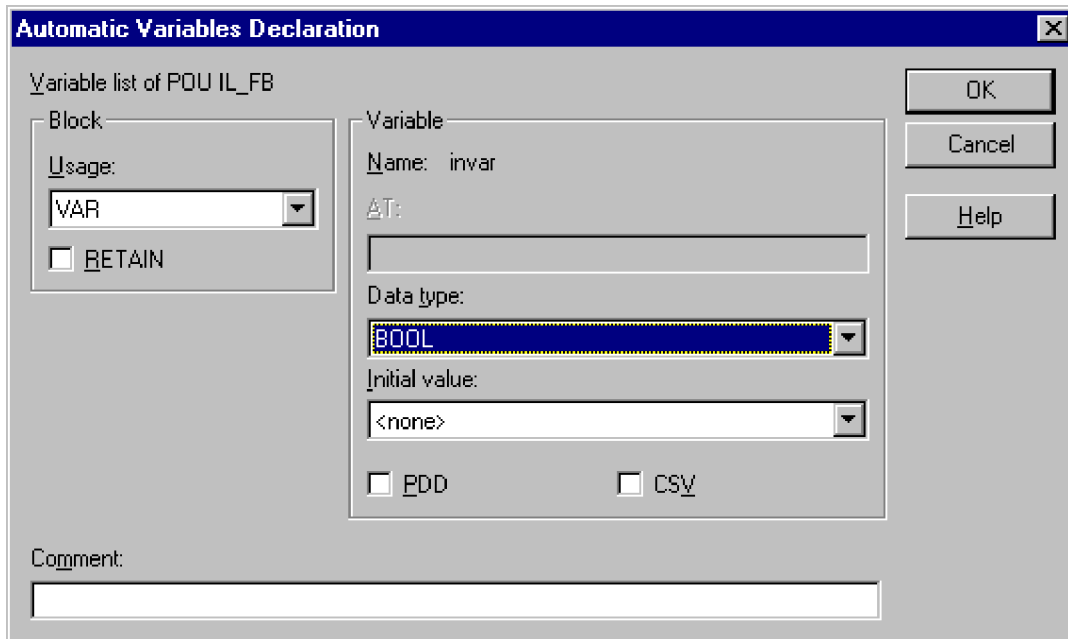


Figure 79: Dialog 'Automatic Variables Declaration'

Using the dialog 'Automatic Variables Declaration'



- Choose a variable keyword in the list box 'Usage'.
- Enter a location in the field 'AT' if you want to declare a located variable (only possible in programs or for global variables).
- Choose the correct data type in the field 'Data type'.
- If required, enter an initial value.
- Mark the checkbox 'PDD' if the variable should be stored in the OmegaOS PDD (Process Data Directory).
- Mark the checkbox 'CSV' if the variable should be stored in the CSV file, i. e. is intended to be used with the OPC Server. The OPC Server processes only variables, which are declared in the CSV file, in order to be used in an OPC client process (e. g. a visualization).
- Enter a comment if you want.
- Confirm the dialog.
The new variable is inserted in the code body worksheet and the declaration of the variable is autoinserted in the variable declaration of the POU.

NOTE



When inserting variables, which have already been declared before, the name of the variable appears in the listbox of the dialog 'Variable'. Confirming the dialog, the variable is directly inserted in the code body worksheet. The dialog 'Automatic Variables Declaration' does not appear.



Detailed information about the OPC server can be found in the 'OPC Server Manual'.

In a second example let us assume, that you want to insert a variable which has already been declared in the global variable declaration of your project.



Calling the dialog 'Variable' with the mouse

- Click on the icon 'Variable' in the toolbar.
The empty dialog 'Variable' appears.



Calling the dialog 'Variable' with the keyboard

- Press <F5>.
The empty dialog 'Variable' appears.

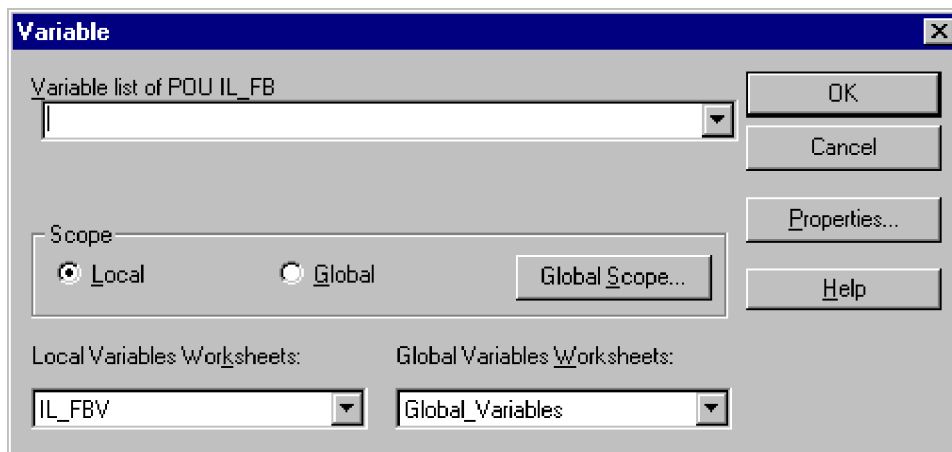


Figure 80: Dialog 'Variable'



Using the dialog 'Variable' to insert a variable, which has already been declared in the global variable declaration of the project



- Activate the radio button 'Global'.
- Check if the correct resource is listed.
- Choose the desired variable in the list box 'Variable list of *resource name*'.
- Confirm the dialog.

The variable is inserted in the code body worksheet and additionally in the variable declaration of the POU as VAR_EXTERNAL since it is a global variable.

7.5 Using jumps and labels

Jumps can be used to jump to a line of the instruction list. In these cases the operator 'JMP' and a label in front of the destination line is used as it is shown in the following example:

```
LD    value1
EQ    INT#100
JMPC  label
LD    value2
ADD   value3
ST    value4
label: LD    %IX2.2
```

Figure 81: Example of a jump

In the example a jump to the label is executed if 'value1' is 100. If it is not 100 no jump is executed and 'value2' is loaded.

7.6 Calling functions or function blocks using the Edit Wizard

Functions can be called in an instruction list by placing the function name in the operator field. In the following figure an example of calling a function with three input parameters and one output parameter is shown:

```
LD    inpar1
function name    inpar2, inpar3
ST    var1
```

Figure 82: Example of calling a function

The first declared input parameter has to be loaded in the preceding line of the function call. In the example the first input parameter is 'inpar1'. All other input parameters have to be used in the second line as operands separated by commas. The result is stored to the variable 'var1' as it is shown in the last line of the figure.

NOTE



This way either standard functions or user defined functions can be called.

Function blocks can be called using the operator CAL and the name of the function block in the operand field. Calling a function block needs more typing effort than calling a function. For the next example let us imagine a function block with two input parameters 'inpar1' and 'inpar2' and two output parameters 'outpar1' and 'outpar2'. The function block is called 'FB_exam'. Its instance name is '*instance*'.

For this example the function block call should look like the following figure:

```
LD    var1
ST    instance.inpar1
LD    var2
ST    instance.inpar2
CAL   instance
LD    instance.outpar1
ST    var3
LD    instance.outpar2
ST    var4
```

Figure 83: Example of calling a function block

The function block call consists of three parts: input parameter introduction, the proper call with the operator CAL and storage of the output parameters.

NOTE



Do not forget to declare the instances of function blocks in the variable declaration of the POU as it is described in the chapter 'Literals, data types and variables'.

The most comfortable and fault preventing method to insert functions or function blocks into the code body is to use the Edit Wizard.

If the Edit Wizard is not visible in the workspace, perform the following steps:

Calling the Edit Wizard with the mouse



- Click on the icon 'Edit Wizard' in the toolbar.
The Edit Wizard window appears.



Calling the Edit Wizard with the keyboard



- Press <SHIFT> + <F2>.
The Edit Wizard window appears.

The following procedures describe the steps how to insert a MAX function and a CTU function block into the code body using the Edit Wizard with the mouse. It is assumed, that the Edit Wizard is visible. Otherwise, call the Edit Wizard as described above.



Inserting a MAX function using the Edit Wizard with the mouse

- Locate the code body position, where the new function is to be inserted. Click the left mouse button to position the text cursor.
- Press <␣> to insert a new line.
- Open the Edit Wizard list box 'Group' and select the group 'Functions'. The available functions are displayed in the selection area of the Wizard.
- Locate the function 'MAX'.
- Double click on the function 'MAX'. It is inserted automatically at the text cursor position as shown in the following figure. Replace the green comments (enclosed by parentheses and asterisks) with the necessary elements.



Inserting a CTU function block using the Edit Wizard with the mouse

- Locate the code body position, where the new function block is to be inserted. Click the left mouse button to position the text cursor.
- Press <␣> to insert a new line.
- Open the Edit Wizard list box 'Group' and select the group 'Function blocks'. The available function blocks are displayed in the selection area of the Wizard.
- Double click on the function block 'CTU'. The dialog 'FB Instances' appears. The field 'FB Instances' displays the default instance name (e. g. for a CTU the name 'CTU_n' is proposed, where n is the first available number which is free for this instance name). To define the name of the new FB you have the following possibilities:
 - Enter a new instance name in the field.
 - Accept the proposed name.
 - Select an already existing name in the list box 'FB Instances'.
- Press 'OK' to confirm the dialog. If you have entered a new instance name, the dialog 'Automatic FB Declaration' appears.
- Enter a comment if you want and press 'OK' to confirm the dialog. The function block 'CTU' is inserted automatically at the text cursor position as shown in the following figure. Replace the green comments (enclosed by parentheses and asterisks) with the necessary elements.

```

1  LD  (* IN1 as ELEMENTARY *)
2  MAX (* IN2 as ELEMENTARY *)
3  ST  (* Result as ELEMENTARY *)
4
5  LD  (* BOOL *)
6  ST  CTU_1.CM
7  LD  (* BOOL *)
8  ST  CTU_1.RESET
9  LD  (* INT *)
10 ST  CTU_1.PV
11 CAL CTU_1
12 LD  CTU_1.Q
13 ST  (* BOOL *)
14 LD  CTU_1.CV
15 ST  (* INT *)

```

Pre-edited function 'MAX'

Pre-edited function block 'CTU', named CTU_1

Green comments, which serve as placeholders and have to be overwritten by the actual values and names

Figure 84: Pre-edited MAX function and CTU function block, inserted using the Edit Wizard

8

EDITING IN FBD

8.1 Calling the graphic editor with a FBD worksheet

Before editing a FBD code body worksheet you have to call the graphic editor with the FBD worksheet, using the project tree.



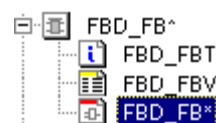
A general description of handling the project tree and browsing through POU's and worksheets is contained in the chapters 'Getting started' and 'Handling and editing projects' in this manual.

As an example, let us assume that you want to edit the FBD worksheet of a function block which is called FBD_FB. For that purpose you first have to insert a function block with this name as it is described in the chapter 'Handling and editing projects'. Keep in mind that the language of the worksheet is determined by the POU language. It is set by inserting the POU respectively the worksheet.



Calling the graphic editor for the FBD code body worksheet with the mouse

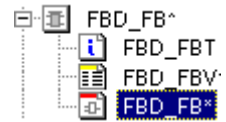
- If the desired worksheet icon is not visible in the project tree, open the corresponding subtree, containing the POU worksheets. For this purpose double click on the POU name (e. g. 'FBD_FB').
- Double click on the icon 'Worksheet in FBD' of the function block 'FBD_FB'. The graphic editor with the FBD worksheet appears.





Calling the graphic editor for the FBD code body worksheet with the keyboard

- If the desired worksheet icon is not visible in the project tree, open the 'FBD_FB' subtree, containing the POU worksheets as follows: Press <↓> or <↑> to highlight the function block 'FBD_FB'. Press <→> to open the function block branch.
 - Press <↓> or <↑> to mark the icon 'Worksheet in FBD' of the function block 'FBD_FB'.
 - Press <↵>.
- The graphic editor with the FBD worksheet appears.



8.2 Introduction to FBD

A code body which is programmed in the graphical language FBD (Functional Block Diagram) consists of functions, function blocks or variables which are connected by lines. A line can also be connected to a line. The set of connected objects is called a FBD network. In FBD networks it is not possible to connect outputs with outputs.

In FBD code bodies, comments can be inserted using the menu item 'Text (Comment)...' in the context menu.

Functions and function blocks can be inserted using the Edit Wizard and edited using the dialog 'Function/Function Block'. Both procedures are described in this chapter. The Edit Wizard is a feature to simplify editing in the graphical editor. It contains all functions and function blocks which can be inserted.

The graphical programming language FBD provides many additional features, which facilitate the creation of a program, FB or function code body, e.g.:

- * The graphic editor provides simple keyboard operations for insertion and scrolling (Cursor keys /CTRL Cursor keys for object mode and SHIFT/SHIFT CTRL Cursor keys for mouse cursor mode).
- * Duplication of inputs can be done directly via keyboard, toolbar and menu.
- * Negation of Inputs, Outputs, Contacts and Coils can be done directly via keyboard, toolbar and menu.
- * Easy auto routing for standard editing cases.
- * Items can be inserted directly on a line or can be connected to the inputs or outputs of present items.
- * Splitter and overview windows are available.
- * Freestyle editing allows to arrange items wherever you want.
- * Double clicking on user functions and function blocks opens the contents of the corresponding POU.

8.3 Inserting functions and function blocks using the Edit Wizard



A general description of the Edit Wizard can be found in the section 'The Edit Wizard' in chapter 'Getting started' in this manual.

In FBD worksheets the Edit Wizard contains functions and function blocks, which can be easily inserted into the graphical language code body.

If the Edit Wizard is not visible in the workspace, perform the following steps:



Calling the Edit Wizard with the mouse

- Click on the icon 'Edit Wizard' in the toolbar. The Edit Wizard window appears.



Calling the Edit Wizard with the keyboard

- Press <SHIFT> + <F2>. The Edit Wizard window appears.

The following procedure describes the steps how to insert a CTU function block into the FBD worksheet using the Edit Wizard with the mouse. It is assumed that the Edit Wizard is visible. Otherwise, call the Edit Wizard as described above.



Inserting the CTU function block using the Edit Wizard

- If desired activate the grid by selecting the menu item 'Grid' in the submenu 'Layout'.
- Click into the worksheet to set an insertion mark.
- Open the Edit Wizard list box 'Group' and select the group 'Function blocks'. The available function blocks are displayed in the selection area of the Wizard.
- Double click on the function block 'CTU'. The dialog 'FB Instances' appears. The field 'FB Instances' displays the default instance name (e.g. for a CTU the name 'CTU_n' is proposed, where n is the first available number which is free for this instance name). To define the name of the new FB you have the following possibilities:
 - Enter a new instance name in the field.
 - Accept the proposed name.
 - Select an already existing name in the list box 'FB Instances'.
- In this case enter a comment if you want and press 'OK'. The function block 'CTU' is automatically inserted into the worksheet and the declaration of the function block instance is autoinserted into the variable declaration of the POU.

With the next steps, the properties of the inserted function/function block are edited and the element is connected to the FBD network.

8.4 Changing the properties of functions and function blocks



The procedures how to change the properties of functions and function blocks are described in the following section 'Changing the properties of functions and function blocks'. For a description concerning the connection of objects refer to the section 'Connecting objects' in this chapter.

8.4 Changing the properties of functions and function blocks

You can change the properties of any function or function block inserted into the FBD worksheet by calling the dialog 'Function/Function Block'.



Calling the dialog 'Function/Function Block' for an existing function/function block

- In the FBD worksheet click on the function or function block to be edited. The marked object changes its color.
- Click with the right mouse button on the marked function or function block to open the context menu for this object.
- Select the menu item 'Object properties'.
The dialog 'Function/Function Block' appears.

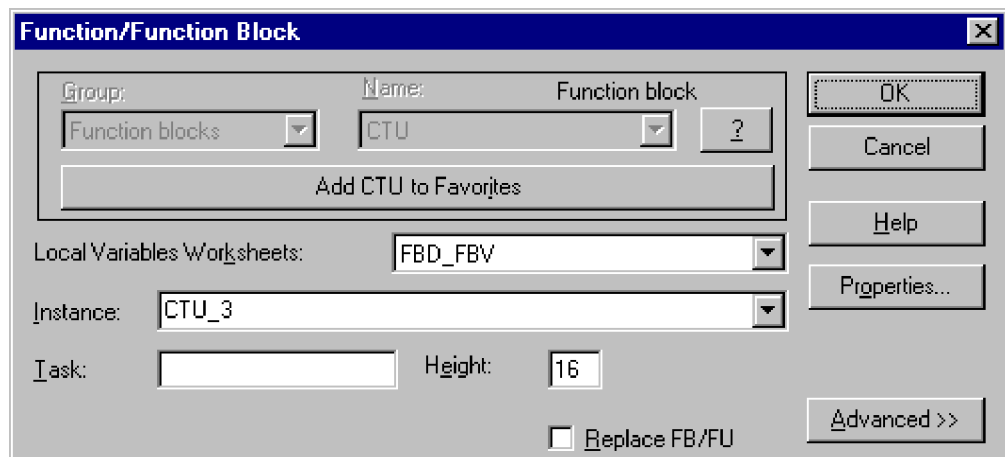


Figure 85: Dialog 'Function/Function Block'



Using the dialog 'Function/Function Block'



- The list boxes 'Group' and 'Name' are grayed and inactive because the object is already created.
- Change the instance name if you want.
- Confirm the dialog.

If you have changed the instance name of a function block, the dialog 'Automatic FB Declaration' appears. Enter a comment if you want and press 'OK' to confirm the dialog. The declaration of the function block instance is autoinserted into the variable declaration of the POU.

8.5 Replacing functions and function blocks

To replace a specific function or function block by another, you can either use the Edit Wizard or the dialog 'Function/Function Block'.



Replacing an object using the Edit Wizard

- Click on the specific object to be replaced. The marked object changes its color.
- Replace the marked object by double clicking on a function/function block in the selection area of the Edit Wizard as described in section 'Inserting functions and function blocks using the Edit Wizard'.
- Edit, if necessary, the new object as described in section 'Changing the properties of functions and function blocks'.



Replacing an object using the dialog 'Function/Function block'

- Click on the specific object to be replaced. The marked object changes its color.
- Click with the right mouse button on the marked object to open the context menu. Choose 'Object properties'. The dialog 'Function/Function Block' appears with the list boxes 'Group' and 'Name' grayed.
- Click on the check box 'Replace FB/FU'. The list boxes 'Group' and 'Name' become active.
- Define the new object to be inserted by selecting the required entries in the list boxes 'Group' and 'Name'. An instance name is proposed by the system.
- Continue editing the new object as described in section 'Changing the properties of functions and function blocks'.

8.6 Inserting variables

For inserting variables in a FBD worksheet you have two possibilities:

- * Inserting a variable which has already been declared as described in the section 'Declaring variables' in the chapter 'Literals, data types and variables'.

8.6 Inserting variables

* Inserting and declaring a new variable.

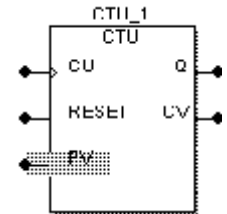
Variables can be inserted anywhere in the FBD worksheet or directly connected to an input/output (formal parameter) of functions or function blocks.

For the next steps let us assume, that you want to insert a variable directly connected to the formal parameter 'PV' of the function block 'CTU'. This variable has not yet been declared.



Inserting a variable at the formal parameter 'PV' with the mouse

- Click on the formal parameter 'PV' to mark it. The marked input changes its color.



- Click on the icon 'Variable' in the toolbar. The dialog 'Variable' appears.



Inserting a variable at the formal parameter 'PV' with the keyboard

- Press the cursor keys to mark the formal parameter 'PV'.
- Press <F5>. The dialog 'Variable' appears.

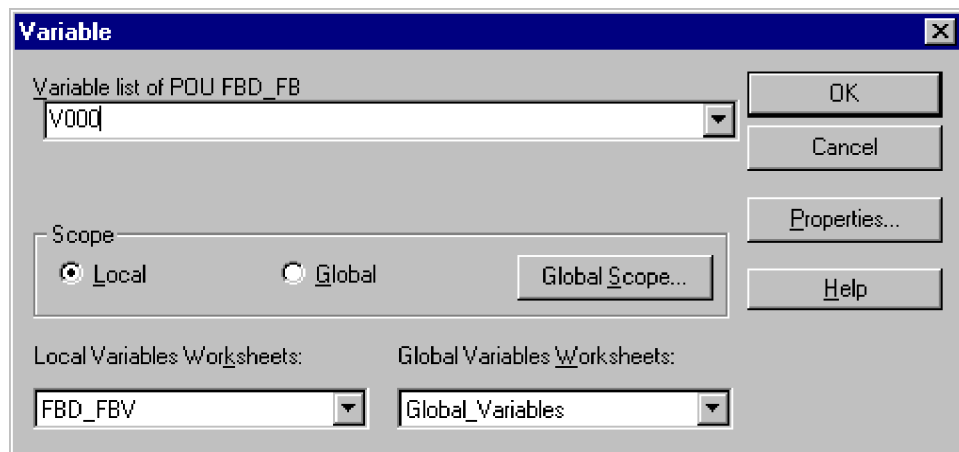


Figure 86: Dialog 'Variable'



Using the dialog 'Variable'

- Enter the variable name 'VAR1'.
- Confirm the dialog. The dialog 'Automatic Variables Declaration' appears.



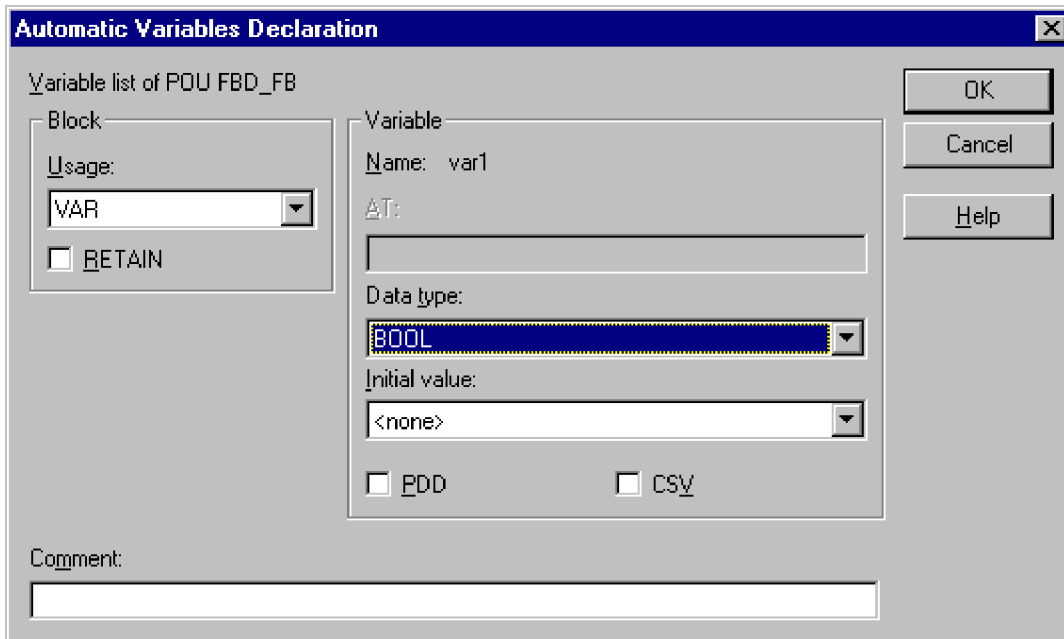


Figure 87: Dialog 'Automatic Variables Declaration'



Using the dialog 'Automatic Variables Declaration'

- Choose a variable keyword in the list box 'Usage'.
- Enter a location in the field 'AT' if you want to declare a located variable (only possible in programs or for global variables).
- Choose the correct data type in the field 'Data type'.
- If required, enter an initial value.
- Mark the checkbox 'PDD' if the variable should be stored in the ProConOS PDD (Process Data Directory), i. e. is intended to be used with IEC 61131-5 communication function blocks.
- Mark the checkbox 'CSV' if the variable should be stored in the CSV file, i.e. is intended to be used with the OPC Server. The OPC Server processes only variables, which are declared in the CSV file, in order to be used in an OPC client process (e. g. a visualization).
- Enter a comment if you want.
- Confirm the dialog.
The new variable is inserted in the code body worksheet and the declaration of the variable is autoinserted in the variable declaration of the POU.



NOTE

When inserting variables, which have already been declared before, the name of the variable appears in the list box of the dialog 'Variable'. Confirming the dialog, the variable is directly inserted in the code body worksheet. The dialog 'Automatic Variables Declaration' does not appear.



Detailed information about the OPC Server can be found in the 'OPC Server Manual'.

8.7 Connecting objects

If you have inserted functions, function blocks or variables into your worksheet, you have to connect them to create a legal FBD network. To connect objects in a FBD network you have the following possibilities:

- * connecting objects using the connection mode
- * connecting objects by drag & drop with the mouse
- * connecting objects while inserting a new object

The connection mode can be used to connect all objects which have free connection points. Connection points are represented as green and blue dots. It is also possible to connect a new line to an existing one using the connection mode.

NOTE



Objects can be linked only using a horizontal line. So your line should start at a connection point of an object and move away from the object in a horizontal way.

While connecting inputs and outputs of functions and function blocks the program provides an auto routing feature. Auto routing means that the program will determine the routing of the connection line between two free connection points automatically. In this case there is no need to mark any corner.

NOTE



Auto routing is only available if you connect an input to an output or vice versa. If you create a feedback or the connection starts at a connection line, auto routing is not possible.

For the following procedures let us assume that you want to connect the function block 'CTU' with the function 'ADD'.



Connecting a function block output to a function input using the connection mode with auto routing

- Click on the icon 'Connect objects' in the toolbar.
A symbol for a connection is added to the cursor. During the connection mode, the icon appears 'pressed'.



- Click on the output of the function block 'CTU', which you want to connect.
- Move the mouse to the input of the function 'ADD' you want to connect. The connection line is displayed red. If the program recognizes the desired connection point, the line is routed automatically and the color of the line changes to green.
- Click on the input. The new connection line is auto routed and inserted automatically.



Connecting two objects using the connection mode without auto routing

- Click on the icon 'Connect objects' in the toolbar. A symbol for a connection is added to the cursor. During the connection mode, the icon appears 'pressed'.
- Click at the point where you want to start drawing the line.
- Move the mouse to the object you want to connect.
- Click with the left mouse button to mark a corner if you want.
- Click at the point where you want the line to end.



Connecting two objects by drag & drop with the mouse

- Click on the function 'ADD' and keep the mouse button pressed.
- Move the mouse towards the function block 'CTU' so that the connection points overlap.
- Release the mouse button. The connection is set.
- If required, move either the function or the function block to a vacant position.

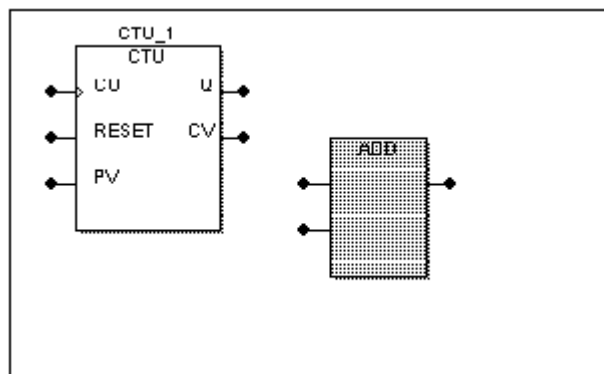


Figure 88: Function and function block before establishing the connection

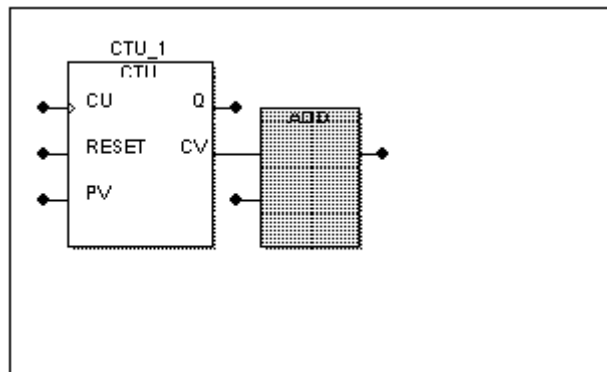


Figure 89: The connection is set

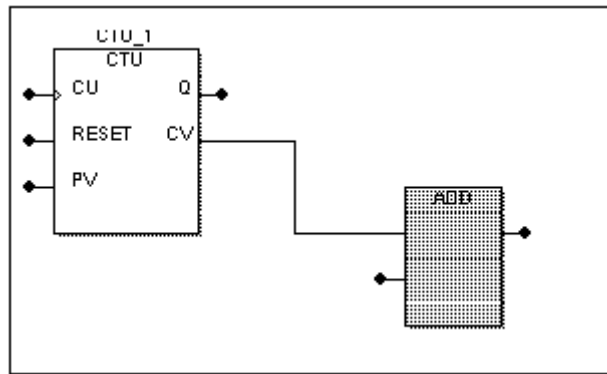


Figure 90: The function is moved to a vacant position.
The connection is routed automatically.

You can also connect a function or function block to another function or function block by marking an output parameter and then inserting the object as described in the following procedure.



Connecting an object to a new function/function block while inserting

- Click on the output 'CV' of the function block 'CTU', before the function 'ADD' is inserted. The marked output changes its color.
- Insert the new function 'ADD' using the Edit Wizard as described in the section 'Inserting functions and function blocks using the Edit Wizard'.
The connection between the highlighted output 'CV' and the first input of the 'ADD' function is established automatically.
- If required, move either the function or the function block to a vacant position.

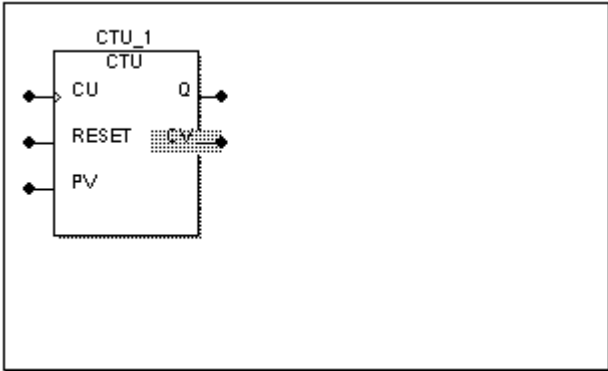


Figure 91: The output 'CV' of the FB 'CTU' is defined to be one end of the connection before the 'ADD' function is inserted

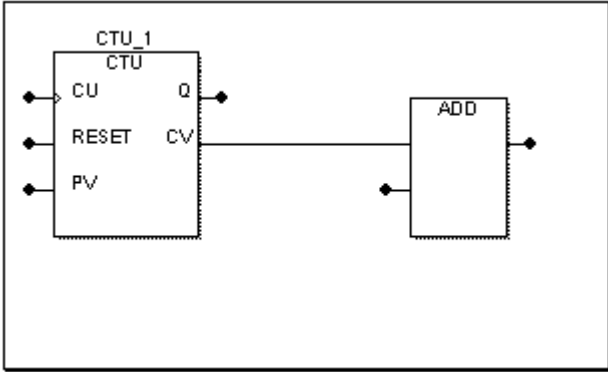


Figure 92: The connection is established automatically when the new 'ADD' function has been inserted

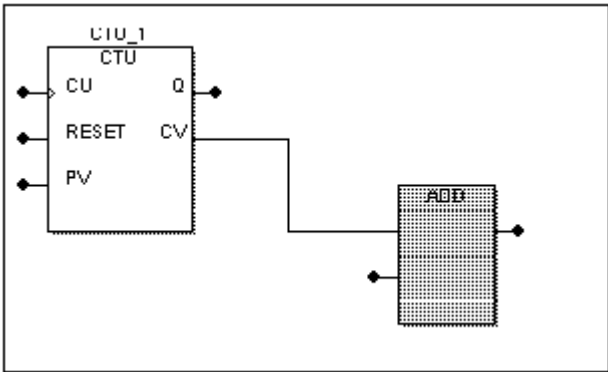


Figure 93: The function is moved to a vacant position. The connection is routed automatically.

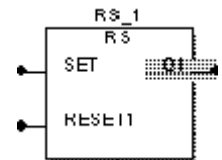
8.8 Negation of inputs and outputs

Using the graphic editor, it is easy to negate and to toggle the negation of inputs or outputs of functions and function blocks. There are several possibilities to negate a FP (Formal Parameter). They are described in this section using the output Q1 of an RS function block as an example.



Negating a formal parameter 'Q1' using the mouse

- Click on the formal parameter 'Q1' to mark it. The marked output changes its color.

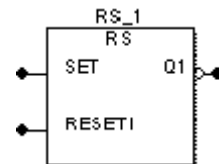


- Click on the icon 'Toggle negation of FP' in the toolbar.



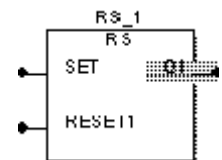
The output is shown with a circle as negation symbol.

To toggle the negation, just perform the described steps again for the same output.



Negating a formal parameter 'Q1' with the keyboard

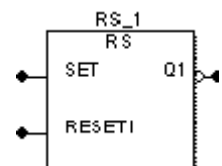
- Press the cursor keys to select the formal parameter 'Q1'. The marked output changes its color.



- Press the shortcut <SHIFT> + <F5>.

The output is shown with a circle as negation symbol.

To toggle the negation, just perform the described steps again for the same output.



8.9 Duplicating inputs of functions

The program allows to duplicate inputs of extensible functions. Using this feature you can add as many inputs to an extensible function as required for your application.

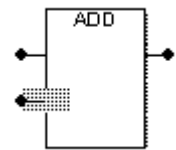
**NOTE**

It is only possible to duplicate the last input of a function.

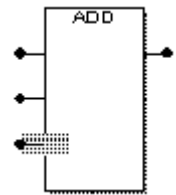
For the following example let us assume that you want to duplicate the input of the extensible function 'ADD'.

**Duplicating an input of the function 'ADD' with the mouse**

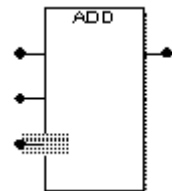
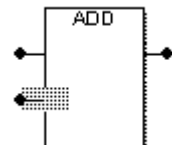
- Click on the last input of the function. The marked input changes its color.
- Click on the icon 'Duplicate FP' in the toolbar.



The new input is added below the previously selected input.

**Duplicating an input of the function 'ADD' with the keyboard**

- Press the cursor keys to select the last input. The marked input changes its color.
- Press <CTRL> + <F5>.
The new input is added below the previously selected input.



EDITING IN LD

9.1 Calling the graphic editor with a LD worksheet

The first step before editing a LD code body worksheet is to call the graphic editor with the LD worksheet, using the project tree.



A general description of handling the project tree and browsing through POU's and worksheets is contained in the chapters 'Getting started' and 'Handling and editing projects' in this manual.

As an example, let us assume that you want to edit the LD worksheet of a function block which is called 'LD_FB'. For that purpose you first have to insert a function block with this name as it is described in the chapter 'Handling and editing projects'. Keep in mind, that the programming language of the worksheet is determined by the POU language. It is set by inserting the POU respectively the worksheet.



Calling the graphic editor for the LD code body worksheet with the mouse

- If the desired worksheet icon is not visible in the project tree, open the corresponding subtree, containing the POU worksheets. For this purpose double click on the POU name (e.g. 'LD_FB').
- Double click on the icon 'Worksheet in LD' of the function block 'LD_FB'. The graphic editor with the LD worksheet appears.



Calling the graphic editor for the LD code body worksheet with the keyboard

- If the desired worksheet icon is not visible in the project tree, open the 'LD_FB' subtree, containing the POU worksheets as follows: Press <↓> or <↑> to highlight the function block 'LD_FB'. Press <→> to open the function block subtree.
 - Press <↓> or <↑> to mark the icon 'Worksheet in LD' of the function block 'LD_FB'.
 - Press <↵>.
- The graphic editor with the LD worksheet appears.

9.2 LD networks, contacts, coils and power rails

A code body programmed in the graphic language LD (Ladder Diagram) is composed of contacts and coils. According to IEC 61131-3 different types of contacts and coils can be used:

Symbol	Name	Description
-- --	Normally open contact	The Boolean value is copied from the left to the right if the state of the associated variable is ON.
-- / --	Normally closed contact	The Boolean value is copied from the left to the right if the state of the associated variable is OFF.
-- () --	Coil	The Boolean value is copied from the left to the right and to the associated variable.
-- (/) --	Negated coil	The Boolean value is copied from the left to the right. The negated Boolean value is copied to the associated variable.
-- (S) --	SET coil	The Boolean value is copied from the left to the right. The associated variable is set if the left link is TRUE.
-- (R) --	RESET coil	The Boolean value is copied from the left to the right. The associated variable is reset if the left link is TRUE.

Figure 94: Table of contacts and coils in LD

Contacts and coils are connected by lines and are bound on the left and on the right with power rails. The state of the left power rail is considered ON all the time. The right power rail is optional.

In addition to the serial connections of contacts and coils parallel branches can be created. Parallel branches are also called wired-ORs.

The set of connected objects is called a LD network. Every LD network shall contain at least one contact, one coil and a left power rail.

Variables in LD used with contacts and coils are always Boolean variables. While inserting contacts or coils the variable name can be entered. The variable name is displayed above the contact or coil in the worksheet.

In LD code bodies comments can be inserted using the menu item 'Text (Comment)...' in the context menu.

The following figure shows an example for a simple LD network.

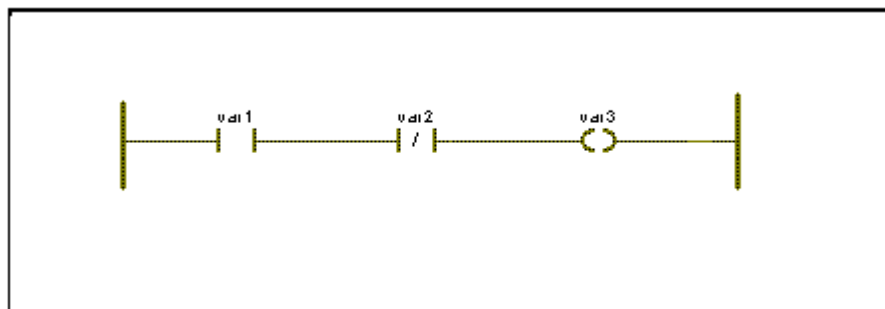


Figure 95: Example of a LD network

The example network consists of a left and right power rail which limits the LD network to the left and to the right. In the middle you can see two contacts and one coil connected with horizontal connection lines. The first contact, called the normally open contact, passes the incoming value from the left to the right if the value of variable 'var1' is TRUE. The second contact passes the incoming value if the value of the variable 'var2' is FALSE. The coil stores the incoming value to the linked variable 'var3'.

NOTE



For compiling your worksheet it is necessary to declare the variables as described in chapter 'Literals, data types and variables'.

In LD code body worksheets it is possible to insert FBD elements manually or using the Edit Wizard. The required steps are described in the section 'Calling functions or function blocks using the Edit Wizard' in this chapter.

9.3 Inserting contacts and coils

For the next steps let us assume that you have already inserted a LD function block called 'LD_FB' as it is described in the chapter 'Handling and editing projects'. Assuming furthermore that you want to insert a LD network with four contacts in a first step.



Inserting a first LD network with the mouse

- If desired activate the grid by selecting the menu item 'Grid' in the sub-menu 'Layout'.
- Click into the worksheet to set an insertion mark.
- Click on the icon 'Contact network' in the toolbar.
A first LD network with one contact and one coil is inserted.



Inserting a first LD network with the keyboard

- If desired activate the grid by pressing the shortcut <CTRL> + <.;>.
- Press <SPACE> to set an insertion mark.
- Press <F6>.
A first LD network with one contact and one coil is inserted.

9.4 Inserting serial contacts and coils

Your screen should look like the following figure:

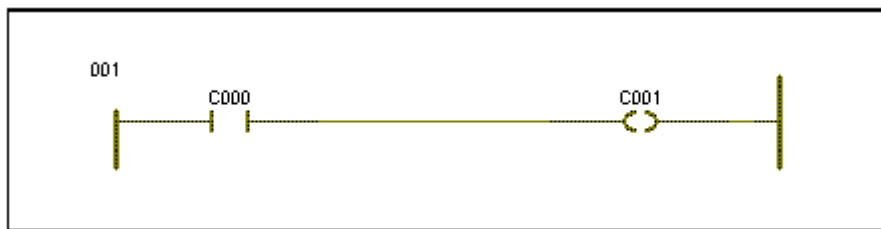


Figure 96: First LD network inserted

9.4 Inserting serial contacts and coils

For inserting more contacts and coils you can use the icons 'Contact left', 'Contact right' and 'Coil right' in the toolbar or the corresponding menu items or keyboard shortcuts. For the next steps let us assume that you want to insert one more serial contact in your LD network.



Inserting more serial contacts with the mouse

- Click on the contact 'C000' to mark it.
- Click on the icon 'Add contact right' in the toolbar.
The new contact 'C002' is inserted to the right of the marked contact.



Inserting more serial contacts with the keyboard

- Press the cursor keys to mark the contact 'C000'.
- Press <F7>.
The new contact 'C002' is inserted to the right of the marked contact.

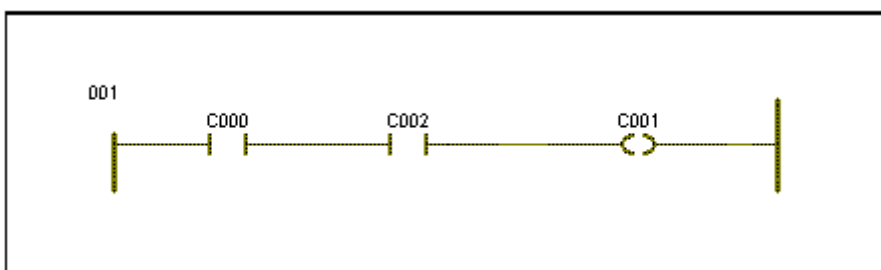


Figure 97: First LD network with inserted new contact 'C002'

9.5 Inserting parallel contacts or coils

Up to now only serial LD networks have been considered. In LD it is also possible to edit parallel branches or the so called Wired-ORs. For the next steps let us assume that you want to insert a branch parallel to contact 'C002' in the LD network described in the previous section.



Inserting a parallel branch with the mouse

- Click on the contact 'C002' to mark it.
- Click on the icon 'Add contact/coil below' to insert a parallel branch below contact 'C002'.



Or:

- Click on the icon 'Add contact/coil above' to insert a parallel branch above contact 'C002'.



In both cases the new contact 'C003' is inserted.



Inserting a parallel branch with the keyboard

- Press the cursor keys to mark the contact 'C002'.
- Press <CTRL> + <F7> to insert a contact below contact 'C002'.
The new contact 'C003' is inserted.

Your screen should look like the following figure now:

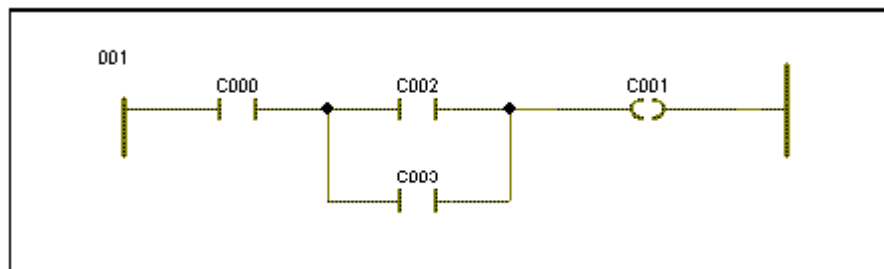


Figure 98: LD network with a parallel branch

9.6 Using the LD branch edit mode

For inserting new branches the LD branch edit mode is provided. The LD branch edit mode allows to create complex branch structures rapidly. It is used in existing LD networks. Before using the LD branch edit mode you first have to insert a network with at least one contact or coil.

For the next description let us assume that you want to insert another parallel contact.



Inserting a parallel branch with the mouse

- Click on the icon 'Insert LD branch' in the toolbar. A symbol for a LD branch is added to the cursor. While the LD branch edit mode is active, the icon appears 'pressed' in the toolbar.
- Click on a link between two objects where you want to start your parallel branch (see step 1. in the figure below).
- Move the mouse up or downwards to a free position (see step 2. in the figure below).
- Click the left mouse button at the position where you want to place the new object.
- Move the mouse to the desired end of the parallel branch (see step 3. in the figure below).
- Click the left mouse button to set the connection.

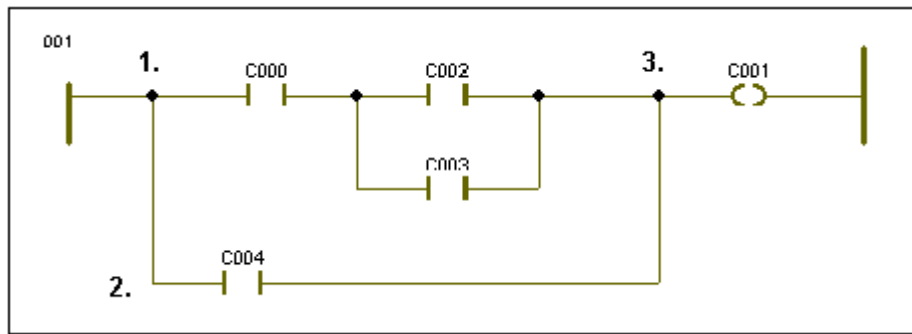


Figure 99: LD network with a parallel branch, inserted in LD branch edit mode

9.7 Changing the properties of contacts and coils

Having inserted all objects in your LD worksheet you may change the properties of the elements. You probably want to change normally open contacts into normally closed contacts or coils into SET coils etc. For changing the types of contacts and coils several icons in the toolbar can be used. You get a short description text to the corresponding function of any icon, if you position the mouse cursor on the particular icon (without clicking it). Additionally the status bar displays the function of any icon.

For the next steps let us assume that you want to change one contact into a normally closed contact.



Changing the properties of a contact with the mouse

- Click on a contact to mark it.
- Click on the icon 'Normally closed contact' in the toolbar.
The contact type is changed.



Changing the properties of a contact with the keyboard

- Press the cursor keys to mark a contact.
- Press <SHIFT> + <F7> to toggle the contact properties.
The contact type is changed.

NOTE



When toggling the properties of a coil, the various coil types (coil, negated coil, set coil and reset coil) are toggled.



Changing the properties of a contact using the dialog 'Contact/Coil'

- Double click on the contact to be changed.
The dialog 'Contact/Coil' appears.

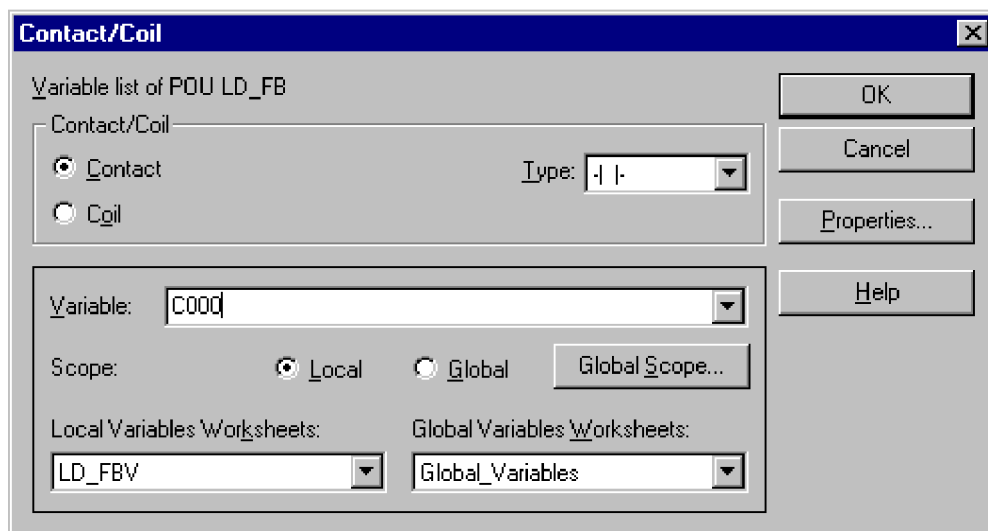


Figure 100: Dialog 'Contact/Coil'



Using the dialog 'Contact/Coil'



- In the list box 'Type' choose the required contact type.
- Enter a name for the variable if you want.
- Confirm the dialog.
The dialog 'Automatic Variables Declaration' appears.

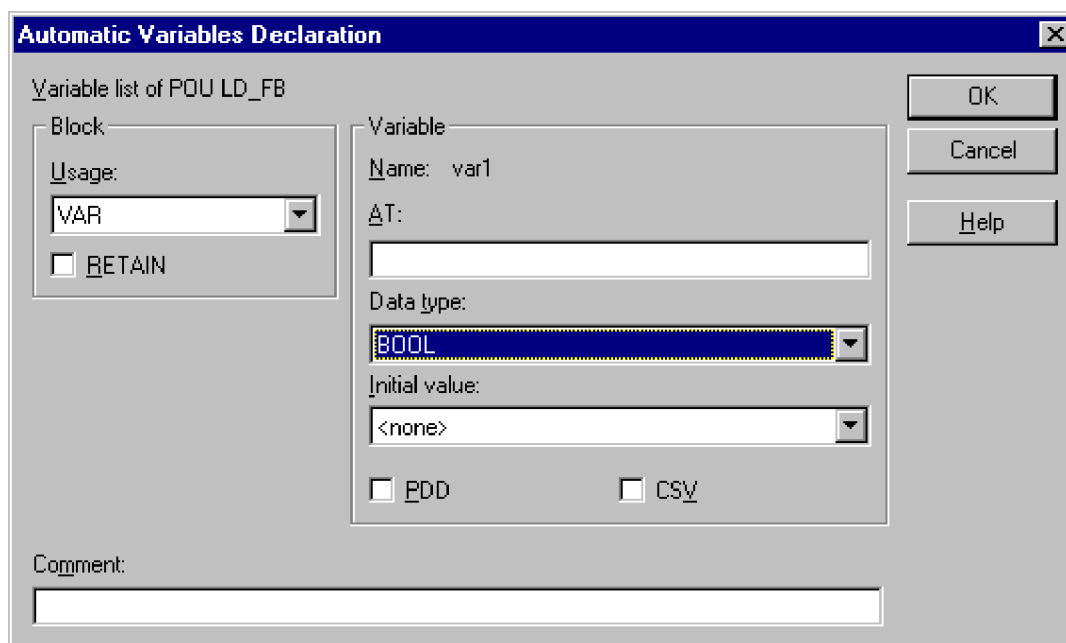


Figure 101: Dialog 'Automatic Variables Declaration'



Using the dialog 'Automatic Variables Declaration'



- Choose a variable keyword in the list box 'Usage'.
- Enter a location in the field 'AT' if you want to declare a located variable (only possible in programs or for global variables).
- Choose the correct data type in the field 'Data type'.
- If required, enter an initial value.
- Mark the checkbox 'PDD' if the variable should be stored in the OmegaOS PDD (Process Data Directory), i.e. is intended to be used with IEC 61131-5 communication function blocks.
- Mark the checkbox 'CSV' if the variable should be stored in the CSV file, i.e. is intended to be used with the OPC Server. The OPC Server processes only variables, which are declared in the CSV file, in order to be used in an OPC client process (e. g. a visualization).
- Enter a comment if you want.
- Confirm the dialog.
The new variable is inserted in the code body worksheet and the declaration of the variable is autoinserted in the variable declaration of the POU.

**NOTE**

When inserting variables, which have already been declared before, the name of the variable appears in the list box of the dialog 'Variable'. Confirming the dialog, the variable is directly inserted in the code body worksheet. The dialog 'Automatic Variables Declaration' does not appear.



Detailed information about the OPC server can be found in the 'OPC Server Manual'.

If you have done all steps of the programming example your screen should look like the following figure:

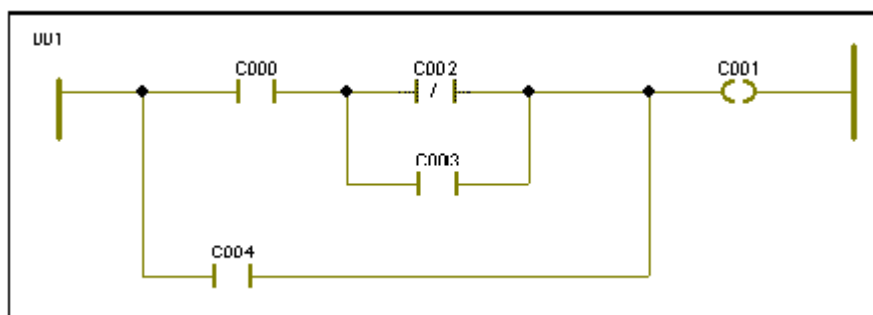


Figure 102: LD network with changed properties

9.8 Inserting variables

While inserting contacts and coils the graphic editor uses a place holder for the name of the variable. These place holders start with the letter 'C' followed by a number. In most cases you don't want to use these place holders but your own variable names. In these cases you have to do the same steps as described in the section 'Changing the properties of contacts and coils' of this chapter.

The first step to do is calling the dialog 'Contact/Coil' by double clicking on the contact or coil. In this dialog you first have to choose if you want to use a local or global variable by activating the corresponding radio button. If the variable is already declared, you may just select it in the listbox. If it is not declared you can enter a name for the new variable. The dialog 'Automatic Variables Declaration' appears. In both cases the variable name is displayed in the LD worksheet after confirming the dialog. If necessary the variable declaration is inserted automatically in the variable worksheet of the POU.

9.9 Calling functions or function blocks using the Edit Wizard

In a LD network also functions and function blocks can be called.



NOTE

Functions and function blocks are represented as FBD elements, i.e. LD and FBD elements are mixed in LD code body worksheets.

You can insert a function or a function block anywhere in the worksheet and connect it later or you can insert it directly in an existing LD network. In this case the function or function block will be directly connected to the network.

The most comfortable way to insert a FBD function or function block into the LD worksheet is to use the Edit Wizard.



A general description of the Edit Wizard can be found in the section 'The Edit Wizard' in chapter 'Getting started' in this manual.

If the Edit Wizard is not visible in the workspace, perform the following steps:



Calling the Edit Wizard with the mouse

- Click on the icon 'Edit Wizard' in the toolbar. The Edit Wizard window appears.



Calling the Edit Wizard with the keyboard

- Press <SHIFT> + <F2>.
The Edit Wizard window appears.

As an example, let us assume that you want to insert the function block 'CTU' connected to an existing LD network consisting of a left and a right power rail, a contact and a coil.



Inserting the function block CTU into an existing LD network using the Edit Wizard

- Click on a contact in the LD network to mark it.
- Open the Edit Wizard list box 'Group' and select the group 'Function blocks'. The available function blocks are displayed in the selection area of the Wizard.

- Double click on the function block 'CTU'. The dialog 'FB Instances' appears. The field 'FB Instances' displays the default instance name (e.g. for a CTU the name 'CTU_1' is proposed, where n is the first available number which is free for this instance name). To define the name of the new FB you have the following possibilities:
 - Enter a new instance name in the field.
 - Accept the proposed name.
 - Select an already existing name in the list box 'FB Instances'.
- Press 'OK' to confirm the dialog. If you have entered a new instance name, the dialog 'Automatic FB Declaration' appears.
- Enter a comment if you want and press 'OK' to confirm the dialog. The function block 'CTU' is inserted automatically into the LD network.



Inserting the function block CTU into an existing LD network with the keyboard

- Press the cursor keys to mark a contact in the LD network.
- Press <ALT> + <3> to activate the Edit Wizard.
- Press the <TAB> key to activate the list box 'Group' in the Edit Wizard. Then press the cursor keys to browse through the available functions and function blocks and select the group 'Function blocks'.
- Press the <TAB> key to activate the selection area and mark the function block 'CTU' with the cursor keys.
- Press <↵>. The dialog 'FB Instances' appears. The field 'FB Instances' displays the default instance name (e.g. for a CTU the name 'CTU_1' is proposed, where n is the first available number which is free for this instance name). To define the name of the new FB you have the following possibilities:
 - Enter a new instance name in the field.
 - Accept the proposed name.
 - Select an already existing name in the list box 'FB Instances'.
- Press <↵> to confirm the dialog. If you have entered a new instance name, the dialog 'Automatic FB Declaration' appears.
- Enter a comment if you want and press <↵> to confirm the dialog. The function block 'CTU' is automatically inserted into the LD network.

Your screen should look like the following figure now:

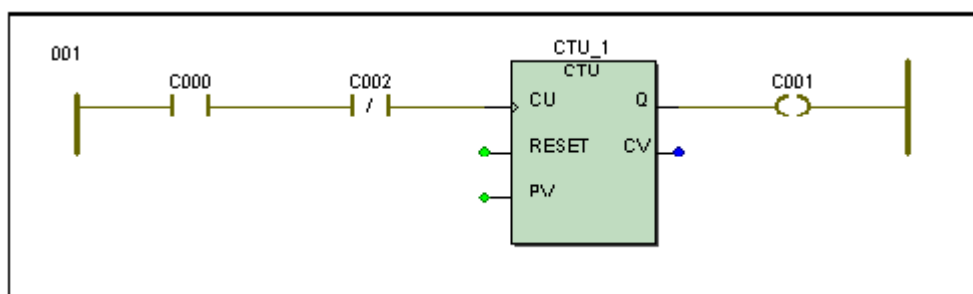


Figure 103: LD network with function block 'CTU'



NOTE

The unconnected formal parameters of the function block can be connected to other contacts, coils or variables.



The steps to edit an inserted FBD function or function block are described in the section 'Changing the properties of functions and function blocks' in the chapter 'Editing in FBD'.

10

EDITING IN SFC

10.1 Calling the graphic editor with a SFC worksheet

The first step before editing a SFC code body worksheet is to call the graphic editor with the SFC worksheet, using the project tree.



A general description of handling the project tree and browsing through POU's and worksheets is contained in the chapters 'Getting started' and 'Handling and editing projects' in this manual.

As an example, let us assume that you want to edit the SFC worksheet of a program which is called SFC_PROG. Therefore you first have to insert a program with this name as it is described in the chapter 'Handling and editing projects'.



Calling the graphic editor for the SFC code body worksheet with the mouse

- If the desired worksheet icon is not visible in the project tree, open the corresponding subtree, containing the POU worksheets. For this purpose double click on the POU name (e.g. 'SFC_PROG').
- Double click on the icon 'Worksheet in SFC' of the program 'SFC_PROG'.

The graphic editor with the SFC worksheet appears.



Calling the graphic editor for the SFC code body worksheet with the keyboard

- If the desired worksheet icon is not visible in the project tree, open the 'SFC_PROG' subtree, containing the POU worksheets as follows: Press <↓> or <↑> to highlight the program 'SFC_PROG'. Press <→> to open the subtree.
- Press <↓> or <↑> to mark the icon 'Worksheet in SFC' of the program 'SFC_PROG'.
- Press <↵>. The graphic editor with the SFC worksheet appears.



10.2 Introduction to SFC

A code body programmed in SFC is composed of steps and transitions which are connected by directed links.

One or several action blocks can be associated to a step. While the step is active the associated action is executed according to the action qualifier. The action can be a boolean variable. It is also possible to define code to be executed in an additional code body worksheet i.e. detail. In this case the name of the code body worksheet has to be used as the name for the action. An action representing a detail appears in green, an action representing a variable appears in red.

A transition represents the condition when execution moves from one step to another. If a transition becomes TRUE the preceding step is executed once again and the succeeding step becomes active. The transition can be either a boolean variable or a directly connected boolean expression in FBD or LD. It is also possible to define code to be executed in an additional code body worksheet i.e. detail. In this case the name of the code body worksheet has to be used as the name for the transition. A transition representing a direct connection is displayed like a normal transition but with a green connection point.

A set of connected objects is called SFC network. A SFC network must always have one initial step which is the first step to be executed after a cold start or warm start. An initial step is represented by a rectangle with a double line. All steps are represented in blue.

Simultaneous or alternative branches can be inserted in the SFC network.

In SFC code bodies comments can be inserted using the menu item 'Text (Comment)...' in the context menu.

Additionally it is possible to insert FBD or LD elements into a SFC worksheet. For this purpose, you can use the Edit Wizard, as described in the chapters 'Editing in FBD' or 'Editing in LD'.

NOTE



You should not use SFC in bypass tasks or event tasks, because this would increase the code execution time substantial! Instead you can use SFC in the DEFAULT task and in CYCLIC tasks.

10.3 Inserting a first SFC network

For the next descriptions let us start with a simple SFC network. In this first example you may insert a SFC network with one step, one transition and one action block.



Inserting a SFC network with the mouse

- If desired activate the grid by selecting the menu item 'Grid' in the sub-menu 'Layout'.
- Click into the worksheet to set an insertion mark.
- Click on the icon 'Create step transition sequence'.
A SFC network with one step and one transition is inserted.





Inserting a SFC network with the keyboard

- If desired activate the grid by pressing the shortcut <CTRL> + <.>.
- Press <SPACE> to set an insertion mark.
- Press <F8>. A SFC network with one step and one transition is inserted.

Your worksheet should look like the following figure:

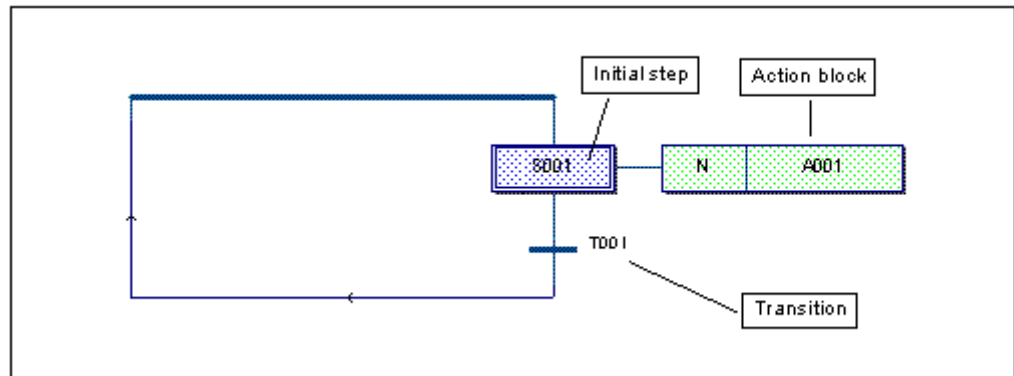


Figure 104: SFC network with one step and one transition

The figure shows a SFC network with one initial step (S001), the corresponding action block (A001) and one transition (T001).

10.4 Inserting more steps and transitions

Let us assume that you want to insert three more steps and transitions into your SFC network.

NOTE

To keep the legal network, steps and transitions are always inserted in pairs.



Inserting more steps and transitions with the mouse

- Click on the step, the new transition and step will succeed.
- Click on the icon 'Insert step transition sequence'. Another transition and step is inserted between the marked step and the next transition.
- Repeat the steps for two additional step and transition pairs.





NOTE

If you mark a **transition** instead of a step and you insert a new step-transition-sequence, the new step and transition is inserted below the marked transition.



Inserting more steps and transitions with the keyboard

- Press the cursor keys to mark the step, the new transition and step will succeed.
 - Press <F8>. Another transition and step is inserted after the marked step.
 - Repeat the steps for two additional step and transition pairs.
-



NOTE

If you mark a **transition** instead of a step and you insert a new step-transition-sequence, the new step and transition is inserted below the marked transition.

Your screen should look like the following figure now:



NOTE

In the following sample network, three new step-transition-sequences were inserted by marking step S001 and then pressing the icon/shortcut three times. Thus, the new sequences are inserted **between** step S001 and transition T001.

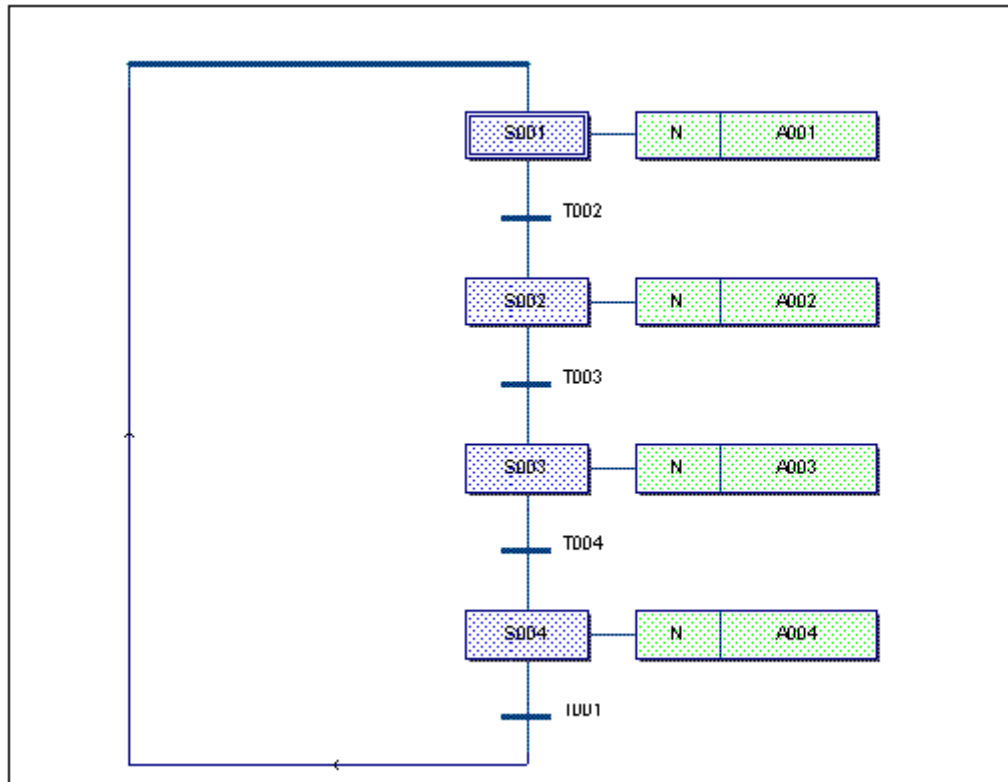


Figure 105: SFC network with four steps

10.5 Changing an initial step into a normal step or vice versa

Your first inserted step is an initial step. For the next sections let us assume that you want the second step to be the initial step. Therefore you have to change the first step into a normal step and the second step into an initial step. In the following section it is described how to change the initial step into a normal step.



Changing an initial step into a normal step with the mouse

- Double click on the initial step.
The dialog 'Step' appears.



Changing an initial step into a normal step with the keyboard

- Press the cursor keys to move to the position of the initial step.
- Press <_>.
The dialog 'Step' appears.





Figure 106: Dialog 'Step'



Using the dialog 'Step'

- Deactivate the checkbox 'Initial step'.
- Confirm the dialog.



10.6 Inserting alternative branches

Alternative branches mean that either one or another transition becomes true and only one of the branches is executed. For the next description let us assume that you want to insert an alternative branch following step S002.



Inserting alternative branches with the mouse

- Click on step S002 to mark it.
- Click on the icon 'Insert Simultaneous/Alternative Divergence' in the toolbar.
The dialog 'Divergence' appears.



Inserting alternative branches with the keyboard

- Press the cursor keys to move to the position of step S002.
- Press <CTRL> + <F8>.
The dialog 'Divergence' appears.

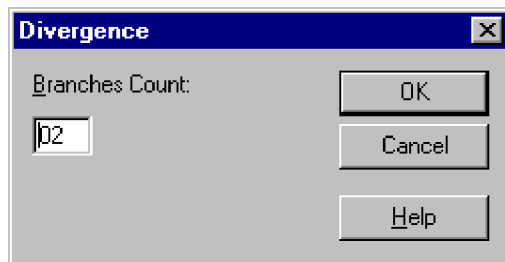


Figure 107: Dialog 'Divergence'



Using the dialog 'Divergence'

- The value in the field 'Branches Count' determines, how many branches are inserted below the marked step. In our example, you can confirm the dialog with the default value.



Your screen should look like the following figure:

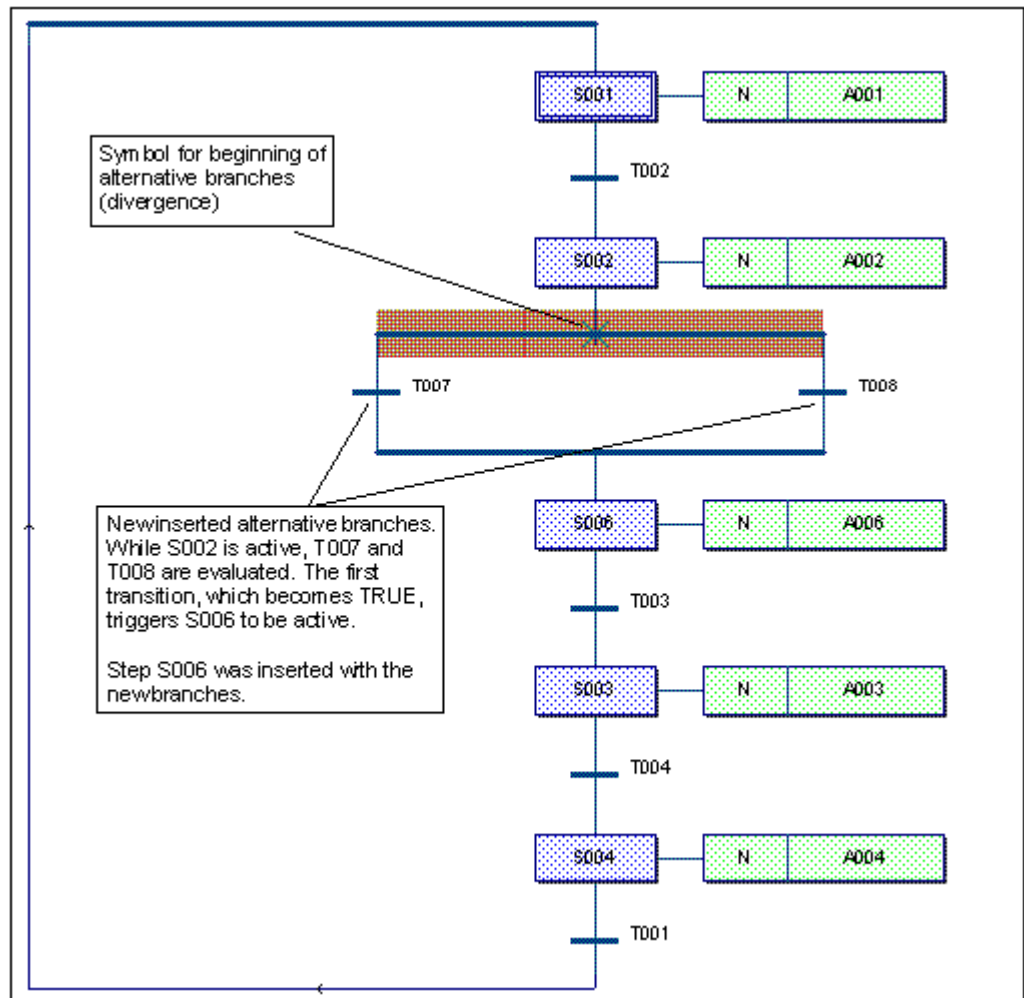


Figure 108: SFC network with 2 alternative branches

10.7 Inserting simultaneous branches

Simultaneous branches are branches where steps and their action blocks are executed simultaneously. For the next description let us assume that you want to insert a simultaneous branch following transition T004.



Inserting simultaneous branches with the mouse

- Click on transition T004 to mark it.
- Click on the icon 'Insert Simultaneous/Alternative Divergence' in the toolbar.
The dialog 'Divergence' appears.





Inserting simultaneous branches with the keyboard

- Press the cursor keys to move to the position of transition T004.
- Press <CTRL> + <F8>.
The dialog 'Divergence' appears.

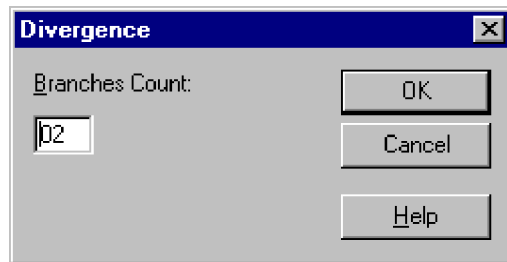


Figure 109: Dialog 'Divergence'



Using the dialog 'Divergence'

- The value in the field 'Branches Count' determines, how many branches are inserted below the marked transition. In our example, you can confirm the dialog with the default value.



Your screen should look like the following figure:

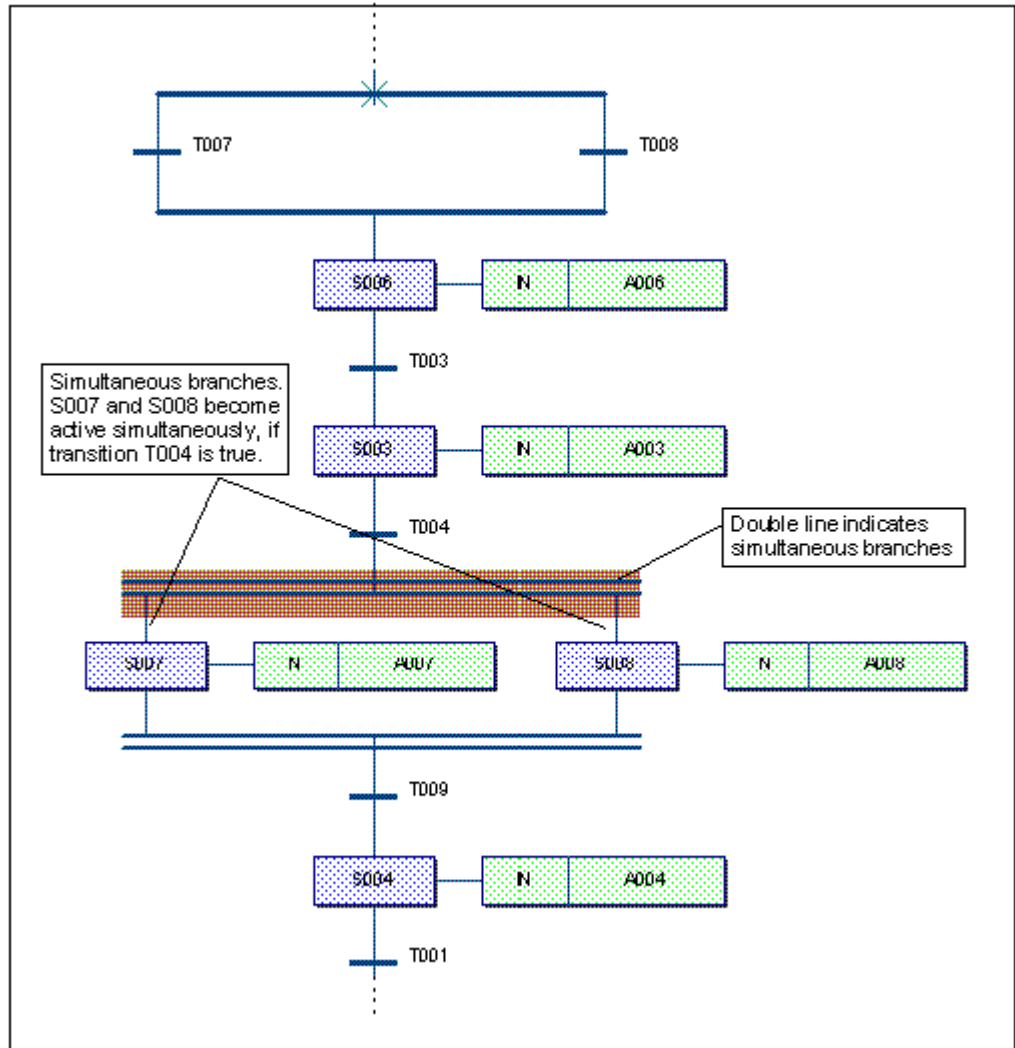


Figure 110: Part of SFC network with an alternative and a simultaneous branch

10.8 Using the SFC branch edit mode

You can insert branches using the SFC branch edit mode. The SFC branch edit mode is a comfortable tool for inserting simultaneous or alternative branches.



NOTE

While using the SFC branch edit mode always click on an object and not on a line.



Inserting an alternative branch with the mouse

- Click on the icon 'Insert SFC branch'.
A symbol for a SFC branch is added to the cursor. During the SFC branch edit mode, the icon appears 'pressed'.
- Click with the left mouse button in the **lower** half of a step to insert an alternative branch.
- Move the cursor to a free position where you want to drop the transition symbol. Avoid collisions with other symbols. Click the left mouse button to place the symbol.
- Move the cursor to the step where you want to close the branch. Click the left mouse button at this point.
The SFC branch edit mode is terminated automatically, after the new branch is inserted. For inserting another branch, reselect the SFC branch edit mode by clicking on the icon again.



Inserting a simultaneous branch with the mouse

- Click on the icon 'Insert SFC branch'.
A symbol for a SFC branch is added to the cursor. During the SFC branch edit mode, the icon appears 'pressed'.
- Click with the left mouse button in the **upper** half of a step to insert a simultaneous branch.
- Move the cursor to a free position where you want to drop the transition symbol. Avoid collisions with other symbols. Click the left mouse button to place the symbol.
- Move the cursor to the transition where you want to close the branch. Click the left mouse button at this point.
The SFC branch edit mode is terminated automatically, after the new branch is inserted. For inserting another branch, reselect the SFC branch edit mode by clicking on the icon again.



10.9 Inserting variables for actions

In SFC it is possible to connect boolean variables to actions. According to the action qualifier the value of the variable is set.



NOTE

Please refer to your PLC documentation or to the context-sensitive Help for detailed information about the available action qualifiers.



Connecting variables to action blocks with the mouse

- Click on the action block 'A004' with the right mouse button to open the context menu.
- Select the context menu item 'Object properties'.
The dialog 'Action' appears.



Connecting variables to action blocks with the keyboard

- Press the cursor keys to mark the action block 'A004'.
- Press <ALT> + <_ >.
The dialog 'Action' appears.

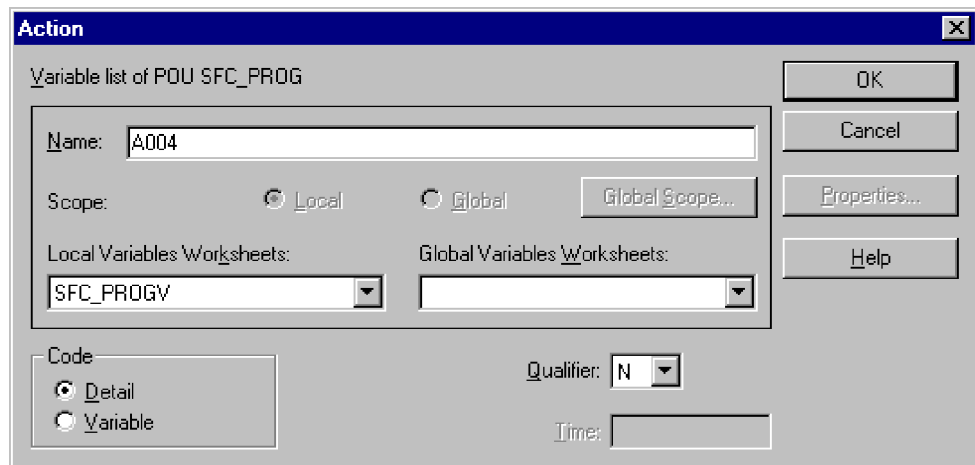


Figure 111: Dialog 'Action'



Using the dialog 'Action'



- Enter the name of the variable in the field 'Name'. For this purpose, delete or overwrite the current entry (A004 in our example).
- Activate the radio button 'Variable'.
- Confirm the dialog.
The dialog 'Automatic Variables Declaration' appears.

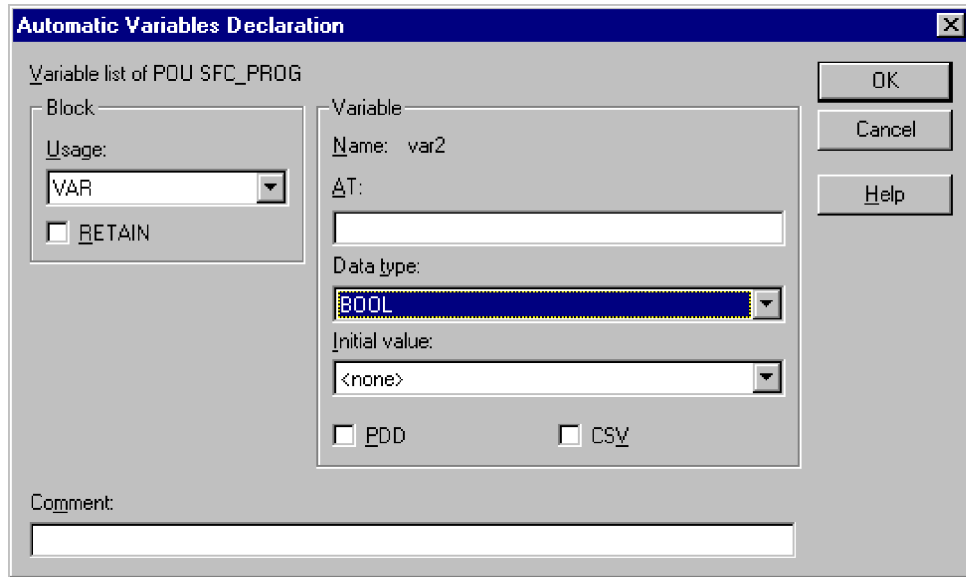


Figure 112: Dialog 'Automatic Variables Declaration'



Using the dialog 'Automatic Variables Declaration'

- Choose a variable keyword in the list box 'Usage'.
- Enter a location in the field 'AT' if you want to declare a located variable (only possible in programs or for global variables).
- Choose the correct data type in the field 'Data type'.
- If required, enter an initial value.
- Mark the checkbox 'PDD' if the variable should be stored in the ProConOS PDD (Process Data Directory), i.e. is intended to be used with IEC 61131-5 communication function blocks.
- Mark the checkbox 'CSV' if the variable should be stored in the CSV file, i.e. is intended to be used with the OPC Server. The OPC Server processes only variables, which are declared in the CSV file, in order to be used in an OPC client process (e. g. a visualization).
- Enter a comment if you want.
- Confirm the dialog.
The new variable is inserted in the code body worksheet and the declaration of the variable is autoinserted in the variable declaration of the POU.

NOTE



While inserting variables which have already been declared before the name of the variable appears in the listbox of the dialog 'Action'. Confirming the dialog, the variable is directly inserted in the code body worksheet. The dialog 'Automatic Variables Declaration' does not appear.



Detailed information about the OPC server can be found in the 'OPC Server Manual'.

In the following figure you can see the action block with the variable name 'var2':

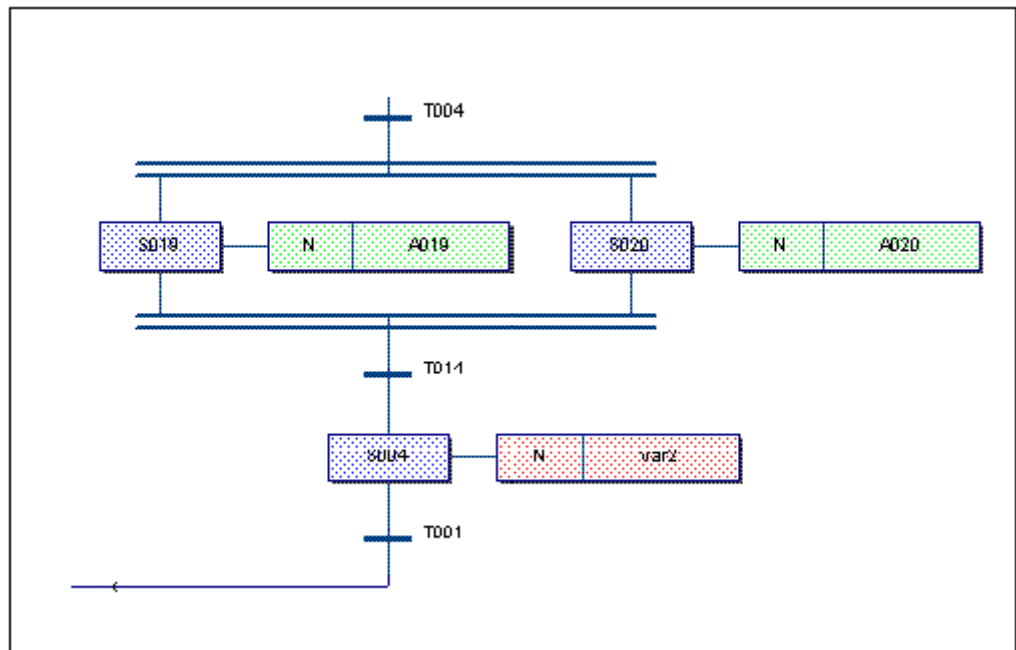


Figure 113: Action block with variable name

10.10 Inserting variables for transitions

If you want to connect a variable to a transition you have to do two main steps. First changing the properties of the transition into a direct connection and then inserting and connecting the variable. For the next procedures let us assume that you want to connect a variable called 'var1' to transition T003.



Changing the properties of the transition with the mouse

- Click on the transition 'T003' with the right mouse button to open the context menu.
- Select the context menu item 'Object properties'.
The dialog 'Transition' appears.



Changing the properties of the transition with the keyboard

- Press the cursor keys to mark the transition 'T003'.
- Press <ALT> + <_ >. The dialog 'Transition' appears.

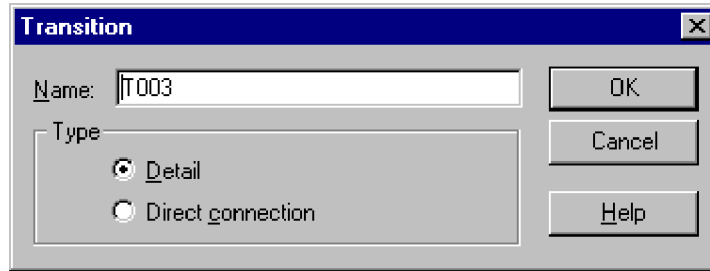


Figure 114: Dialog 'Transition'

Using the dialog 'Transition'



- Activate the radio button 'Direct connection'.
- Confirm the dialog.
The transition is shown with a green connection point.

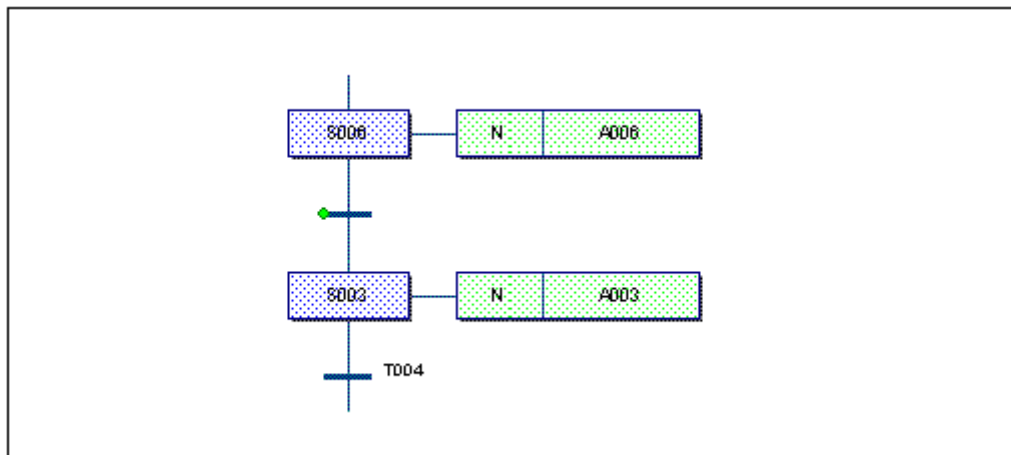


Figure 115: Transition, specified as direct connection with a green connection point

Inserting the variable with the mouse



- Click on the transition to mark it.
- Click on the icon 'Variable'.
The dialog 'Variable' appears.



Inserting the variable with the keyboard



- Press the cursor keys to mark the transition.
- Press <F5>.
The dialog 'Variable' appears.

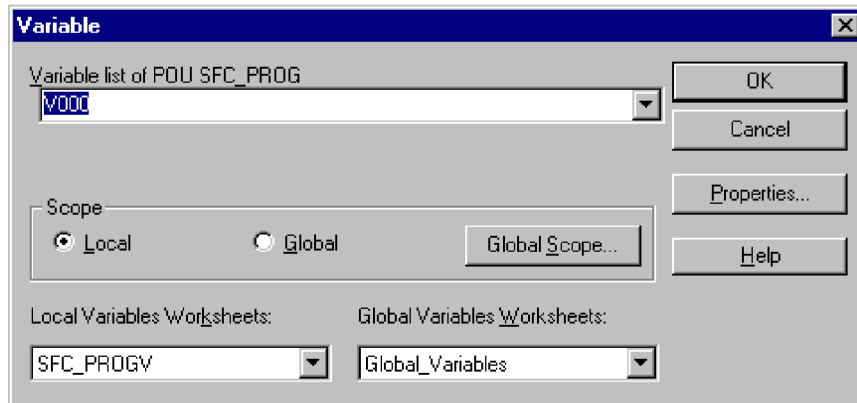


Figure 116: Dialog 'Variable'



Using the dialog 'Variable'



- Enter a variable name.
- Confirm the dialog.
The dialog 'Automatic Variables Declaration' appears.

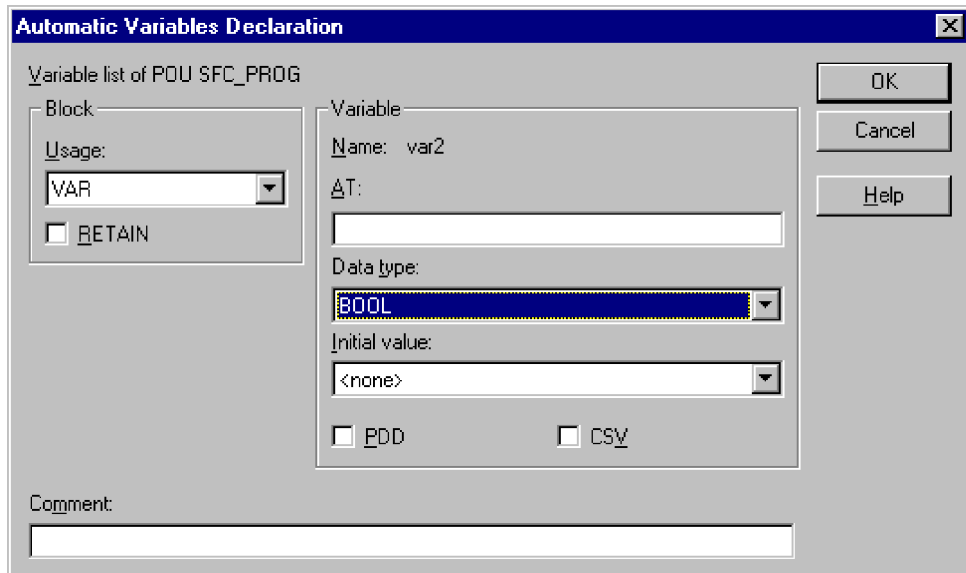


Figure 117: Dialog 'Automatic Variables Declaration'



Using the dialog 'Automatic Variables Declaration'



- Choose a variable keyword in the list box 'Usage'.
- Enter a location in the field 'AT' if you want to declare a located variable (only possible in programs or for global variables).
- Choose the correct data type in the field 'Data type'.
- If required, enter an initial value.
- Mark the checkbox 'PDD' if the variable should be stored in the OmegaOS PDD (Process Data Directory).
- Mark the checkbox 'CSV' if the variable should be stored in the CSV file, i.e. is intended to be used with the OPC Server. The OPC Server processes only variables, which are declared in the CSV file, in order to be used in an OPC client process (e. g. a visualization).
- Enter a comment if you want.
- Confirm the dialog.
The new variable is inserted in the code body worksheet and the declaration of the variable is autoinserted in the variable declaration of the POU.

NOTE



While inserting variables which have already been declared before the name of the variable appears in the listbox of the dialog 'Variable'. Confirming the dialog, the variable is directly inserted in the code body worksheet. The dialog 'Automatic Variables Declaration' does not appear.



Detailed information about the OPC server can be found in the 'OPC Server Manual'.

In the following figure you can see a variable connected to a transition:

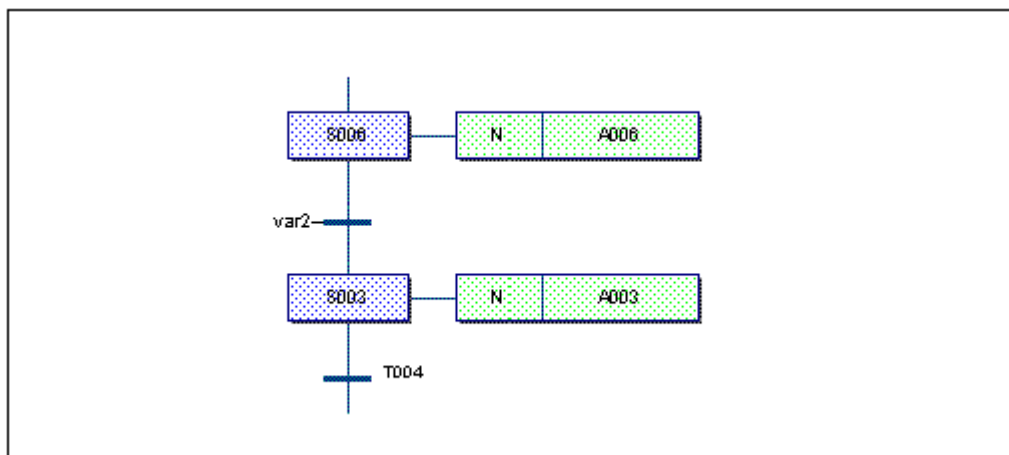


Figure 118: Variable connected to a transition

10.11 Calling functions

Instead of variables as it has been described in the section above also functions can be connected to transitions with direct connections. In these cases you first have to insert the function and then connect it with the connection point of the transition.



The steps how to insert and connect functions using the Edit Wizard are described in the section 'Inserting functions and function blocks using the Edit Wizard' in the chapter 'Editing in FBD' of this manual.

It is also possible to insert LD networks and connect them to the transition.



The steps how to insert contacts and coils are described in the chapter 'Editing in LD' of this manual.

10.12 Action and transition details

According to IEC 61131-3 it is possible to edit a code body for actions and transitions instead of connecting variables. These code bodies are edited in worksheets which you can find in the project tree below the directory nodes for actions and transitions. For the following steps let us assume that you have selected action A005 to be a detail.



Creating an action detail with the mouse

- Double click on the action block A005 in the SFC network.
The dialog 'Insert' appears.



Creating an action detail with the keyboard

- Press the cursor keys to mark the action block A005 in the SFC network.
- Press <_|>.
The dialog 'Insert' appears.

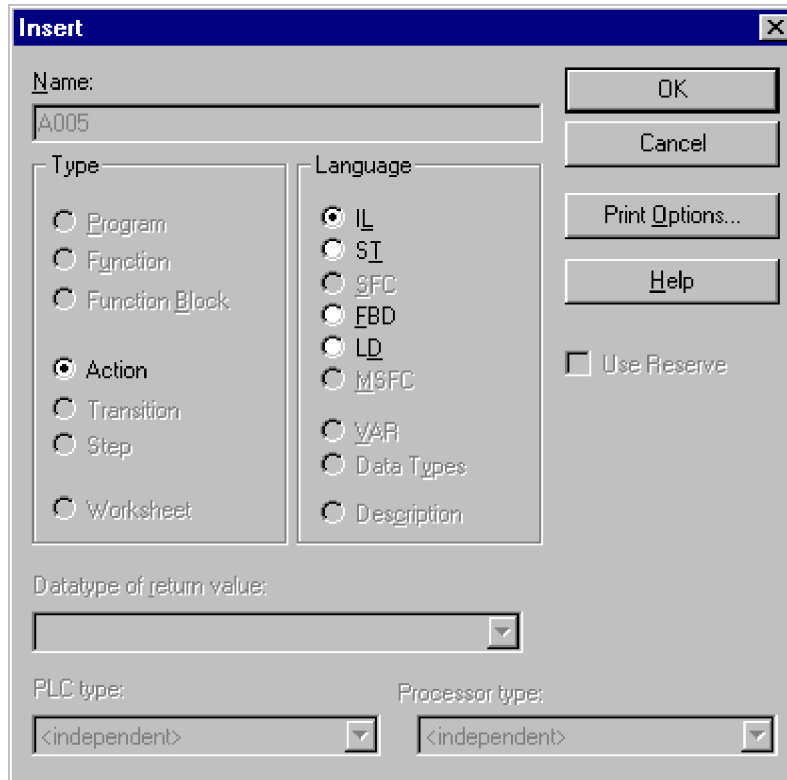


Figure 119: Dialog 'Insert'



Using the dialog 'Insert'



- Choose a programming language.
- Confirm the dialog.
The worksheet is inserted in the project tree and it is opened for immediate editing.
- Edit the code body for the action detail.



NOTE

For transitions the same steps as for actions have to be done.

COMPILING, DOWNLOADING AND DEBUGGING

11.1 Inserting configurations, resources and tasks

The project tree normally has one configuration, one resource and one task if you create a new project. It is possible to insert either new configurations, resources or tasks using the project tree editor. The steps to be done are almost the same independently if you are inserting configurations, resources or tasks.

NOTE



The PLC type of the configurations and the processor type of the resources depend on the connected PLC.

For the next steps let us assume that the project already contains a configuration and a resource and you want to insert a new resource called 'RES_2' for IPC as processor type.



Inserting a new resource with the mouse

- Click with the right mouse button on the icon '*Resource name*' in the project tree to open the context menu.
- Select the menu item 'Insert'.
The dialog 'Insert' appears.



Inserting a new resource with the keyboard

- Press <↓> or <↑> to mark the icon '*Resource name*' in the project tree.
- Press <INS>.
The dialog 'Insert' appears.



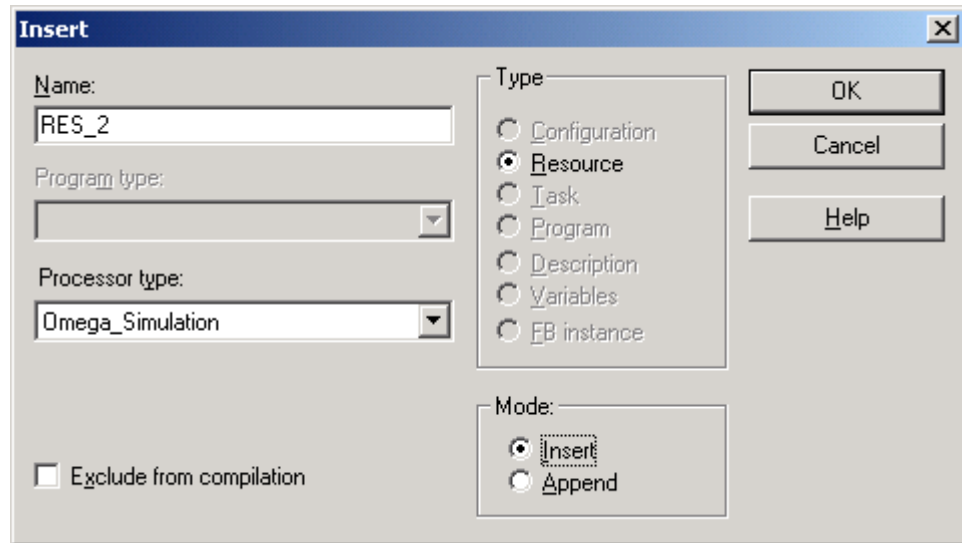
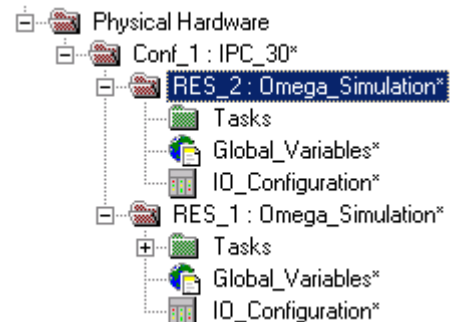


Figure 120: Dialog 'Insert'



Using the dialog 'Insert'

- Enter 'RES_2' as the name of the resource and select the desired processor type.
- Confirm the dialog.
The resource is inserted in the project tree.
- Select the menu item 'PLC Settings...' in the context menu of the resource (= right mouse button).
The dialog 'Resource Settings...' appears.



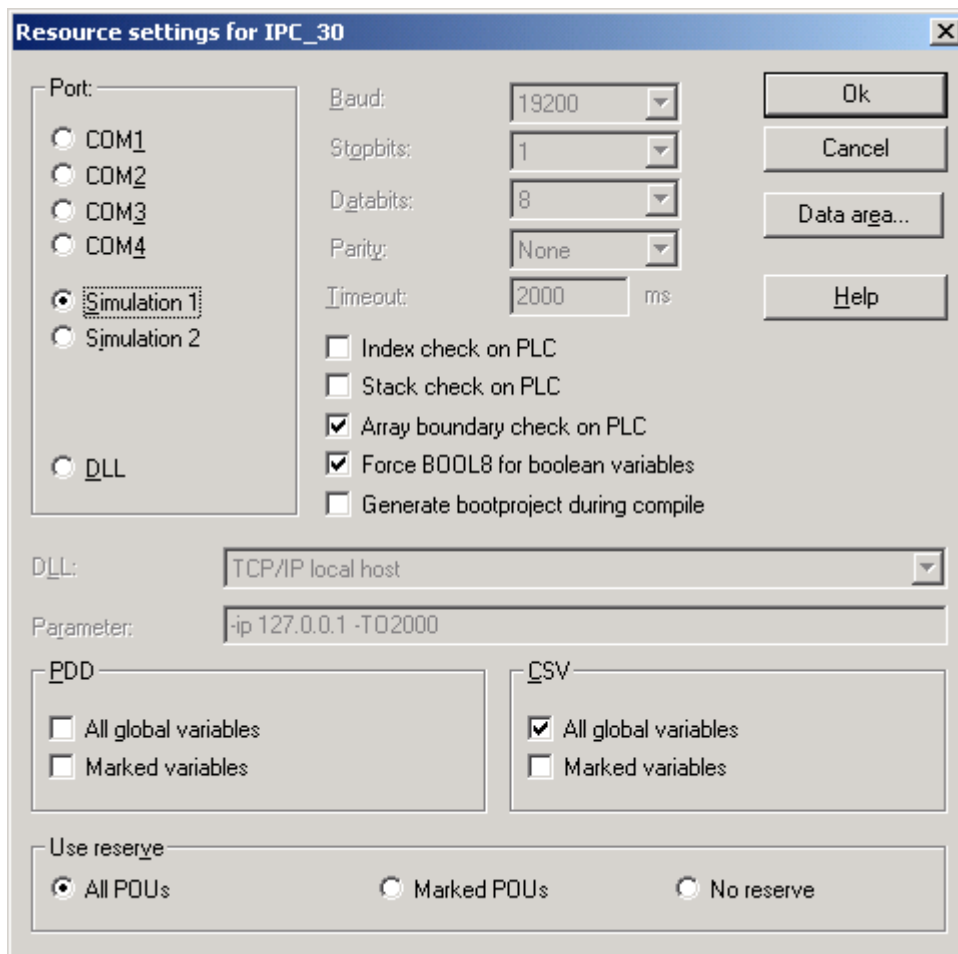


Figure 121: Dialog 'Resource Settings...'



Using the dialog 'Resource Settings...'

- Activate the radio button 'Simulation 1' to send the PLC program to simulation 1 while downloading.
- If desired, change the properties or the data area of the resource.
- Activate the CSV checkboxes, if the resource variables are intended to be used with the OPC Server. To store all global variables in the CSV file, activate the checkbox 'All global variables'. If you want to store variables explicitly marked as CSV variables in the CSV file, activate the checkbox 'Marked variables'. The OPC Server processes these variables and transfers their actual values to an OPC client (e. g. a visualization). Detailed information about the OPC Server are contained in the 'OPC Server manual'.
- Confirm the dialog.

11.1.1 Inserting a DEFAULT task

The DEFAULT task is executed whenever no other CYCLIC or EVENT task must be executed.

Thereby the DEFAULT task always is executed cyclic and alternates with the cyclic communication task over which the online communication is processed and which is activated automatically by the run time system.

With the cyclic execution it is ensured, that communication task and DEFAULT task do not interact and the temporal connections of the program execution of the IEC-code are independent of the user is online or offline.

The cyclic time of the DEFAULT task is twice the system tick of the used resource.

You can read the sampling interval of the system tick online in the information window of the current target system (in „Online/Resource control/Info“ in the entry „System tick“) and is dependent of the used resource.

The sampling interval of the system tick is e.g. for Drive Line II, b maXX PLC and controller PLC 10 ms.

For this reason the DEFAULT task is executed every 20 ms (and the communication task also every 20 ms, because of the alternation with the DEFAULT task at every system tick).

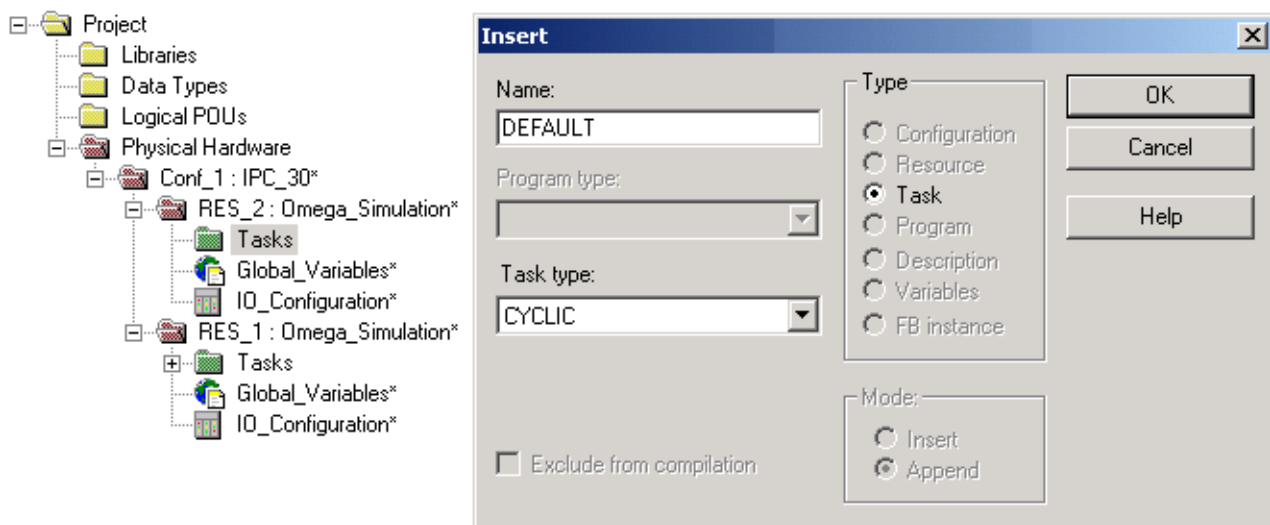


Figure 122: Insert a DEFAULT Task

The default watchdog time can be adapted after the acknowledge of the dialog. If the execution of the program assigned to the DEFAULT task takes longer than the watchdog time, the open-loop control goes to stop and a corresponding error message of the control can be read out.

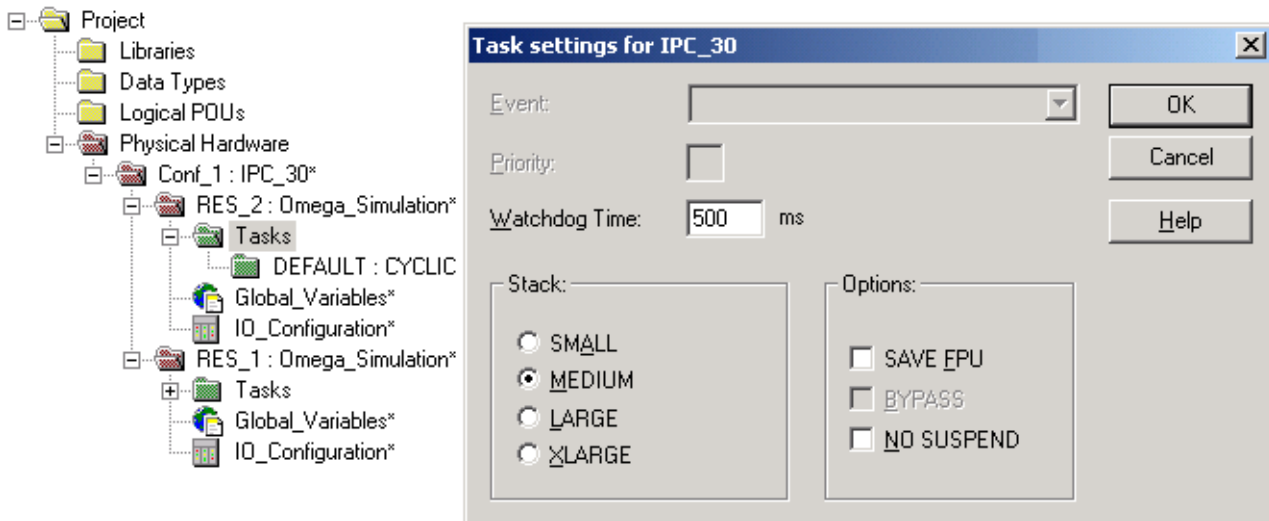


Figure 123: Task settings for a DEFAULT task

11.1.2 Insert a new task

In addition to the DEFAULT task also other tasks can be created.

It is differentiated between three groups:

- CYCLIC
- EVENT
- SYSTEM

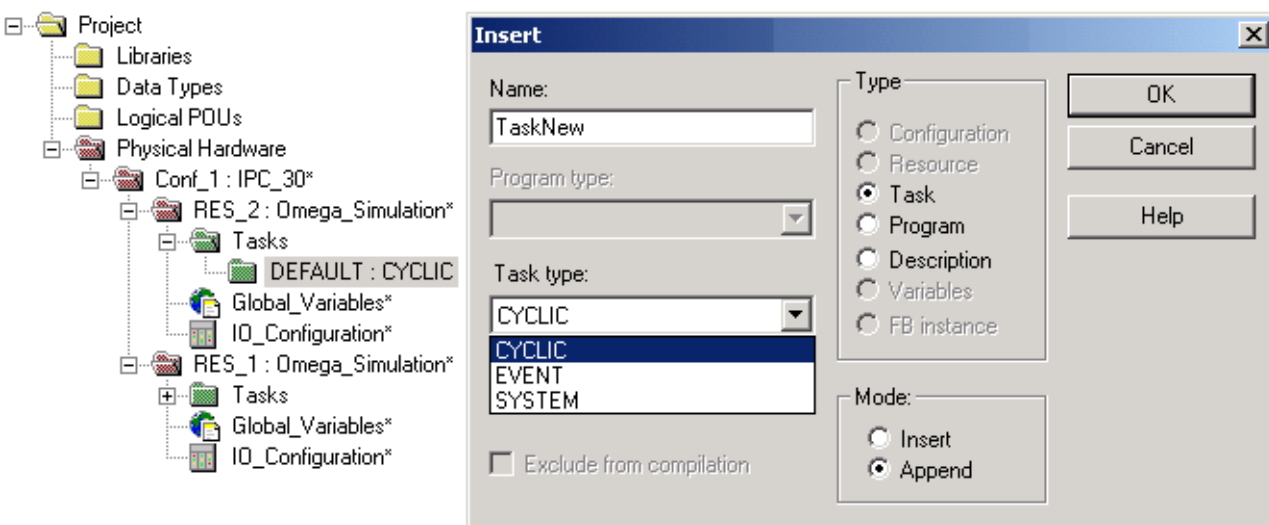


Figure 124: Insert a new task of the type CYCLIC

A task of the type "CYCLIC" is a cyclic task, where you can set a different sampling interval (excepting the DEFAULT task, whose sampling interval is forced by the used target system). The processing of these task types is managed by the run time system, whereby in the POU's assigned to this task type the Online/logic analysis and the Online/address status can be executed.

A task of the type „EVENT“ is a BYPASS task, without run time monitoring and dependent of the used resources. For this purpose see the chapters in the Operation Instructions and Application Manuals.

A task of the „SYSTEM“ type contains a special system task, as e.g. „Cold Start Task“ and „Warm Start Task“.

As an example we present a cyclic task „TaskNew“, which is inserted and is called every 50 ms.

Select task type „CYCLIC“ in the [Figure 124:](#) on page 173. After the acknowledge of the dialog a select menu appears in which the cycle time (= interval) and the Watchdog Time can be set.

The open-loop control sends an error message, if the run time of the task exceeds the watchdog time.

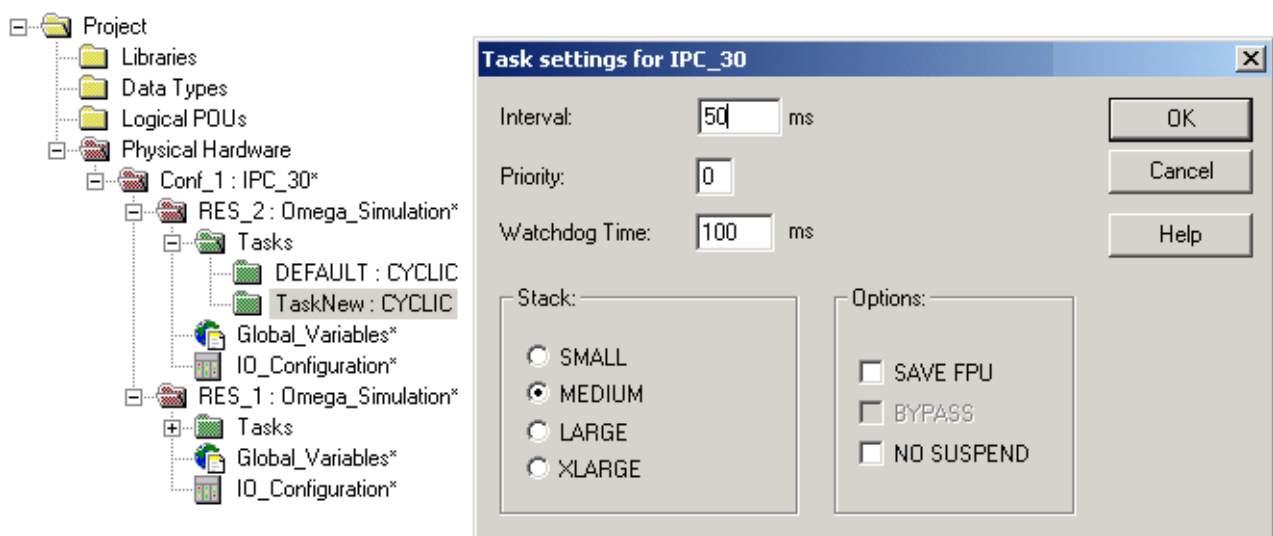


Figure 125: Task settings for a CYCLIC Task



NOTE

The cycle time must be a multiple of the system tick time (see above).

Example: At 10 ms system tick an entry of „11 ms“ or „19 ms“ internally acts as the value „10 ms“.

Values less than the system tick time are rounded up internally to the system tick time (here 10 ms).

11.1.3 Insert a system task

If the type „SYSTEM“ is selected as task type (special system tasks), it is to be selected from the following menu:

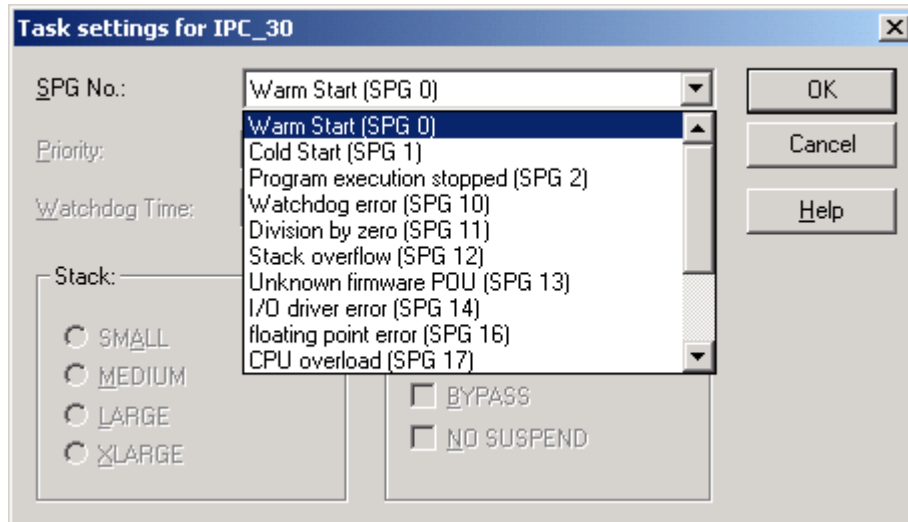


Figure 126: Select a system task

The descriptions according the single system tasks you will find in the corresponding chapters of the Operating Instructions and Application Manuals of the individual open-loop control.

11.2 Associating programs to tasks

If you have inserted a new resource you have to insert one or several tasks. For the next steps let us assume that you have already inserted the task 'DEFAULT' in 'res_2' following the steps described in the previous section.

The next step to be done before compiling is to associate programs to tasks. This means deciding in which task a program is processed.



Associating a program to a task with the mouse

- Click with the right mouse button on the task icon in the project tree to open the context menu.
- Select the menu item 'Insert'.
The dialog 'Insert' appears.





Associating a program to a task with the keyboard

- Press <↓> or <↑> to mark the task icon in the project tree.
- Press <INS>.
The dialog 'Insert' appears.

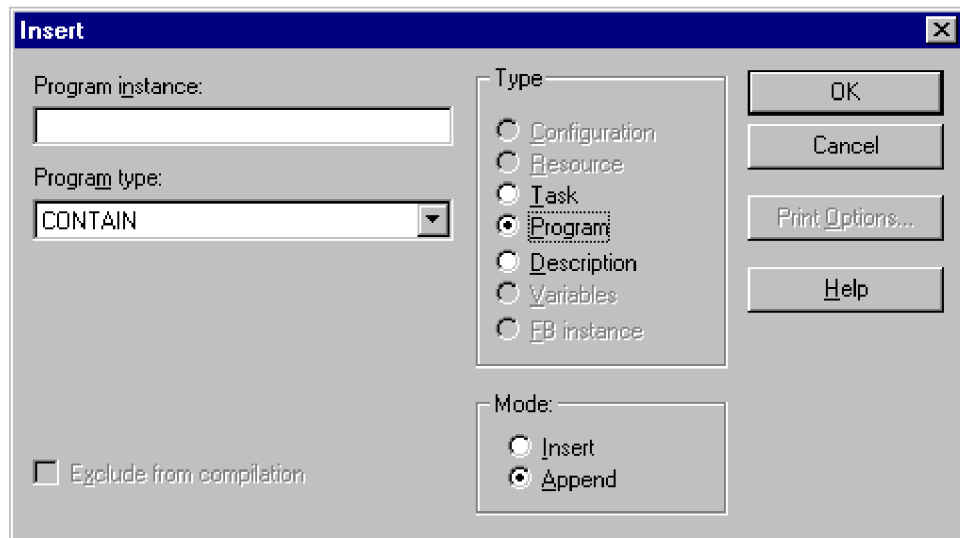


Figure 127: Dialog 'Insert'



Using the dialog 'Insert'



- Activate the radio button 'Program' as shown in figure 127.
- Enter the instance name of the program in the field 'Program instance'.
- Choose the name of the program which is contained in the list box 'Program type'.
- Confirm the dialog.
The icon of the program is inserted in the project tree.

11.3 Compiling a project

Compiling means translating and transforming the contents of the worksheets in special code which can be executed by your PLC. The compilation process is performed in several steps. It starts with compiling (i.e. syntax checking) the different worksheets. During the second main step the compiled worksheets are linked together and an intermediate IEC code is generated. The last step generates the PLC code.

You have several possibilities for compiling either the whole project or only parts of it. In the following list the different possibilities are described, which are called using the menu items in the submenu 'Build' or the corresponding icons in the toolbar.

- * 'Make' - This is the standard mode for compiling the project when you have finished editing. The menu item can be used to compile all worksheets which have been edited. These worksheets are marked with an asterisk in the project tree. After using 'Make' the PLC specific code is generated and the project is ready for downloading to the PLC.
- * 'Patch POU' - This menu item is used to compile changes, which have been made e.g. after debugging a project. The changes are automatically downloaded to the PLC, so that you can view them immediately after switching into online mode. While patching a POU it is not necessary to stop the program execution on the PLC.
- * 'Compile worksheet' - This menu item is used to compile a single worksheet after editing it. Choosing this menu item means, that syntax errors within the current code body worksheet and the related variable worksheet are going to be detected by the compiler. All detected errors and warnings are displayed in the message window. By double clicking on an error or warning you can open the related worksheet, where the error was detected.

NOTE



When closing or saving a worksheet, the system automatically compiles this worksheet. Additionally the first variables worksheet within the same POU is compiled. Therefore it is important, that the used variables are declared in the first variables worksheet of the POU. Due to this automatic compilation each user defined function or function block is available in the Edit Wizard immediately after saving the corresponding worksheet.

- * 'Rebuild Project' - This menu item is used to compile the whole project for the first time after editing. It should only be used, if 'Make' generates compiler errors, if you have unzipped your project without the front end code or if changes have been made in an announced user library.
- * Using 'Rebuild Project' all worksheets are going to be compiled and linked. Detected errors and warnings are displayed in the message window. After the syntax checking the IEC code as well as the PLC specific code is generated automatically. The project is then ready for downloading to the PLC.

The compilation modes 'Make' and 'Patch POU' are described in the following sections.

11.4 Compiling a project using 'Make'

This section describes the steps how to compile the changes you have made in the worksheets. The other possibilities are described in the context-sensitive Help.

Using the menu item/icon 'Make' the changed worksheets are compiled, linked and the changed PLC code is going to be generated. After successful execution the changed project is ready for downloading to the PLC.

NOTE



Before starting the compilation, ensure that the message window is visible. This window displays the compilation process, any detected errors and warnings and additional information to the process. If the window is not visible, press <CTRL> + <F2>.



Compiling the changes with the mouse

- Click on the icon 'Make' in the toolbar. The compilation process is displayed in the sheet 'Build' of the message window. Errors and warnings detected during compilation are logged in the corresponding sheets of the message window.



Compiling the changes with the keyboard

- Press <F9>. The compilation process is displayed in the page 'Build' of the message window. Errors and warnings detected during compilation are logged in the corresponding sheets of the message window.

In most cases while compiling for a first time the different compilers detect programming errors, such as a variable name which has been used twice or typing errors. In those cases a message appears in the sheet 'Build' of the message window, announcing the number of detected errors and warnings.

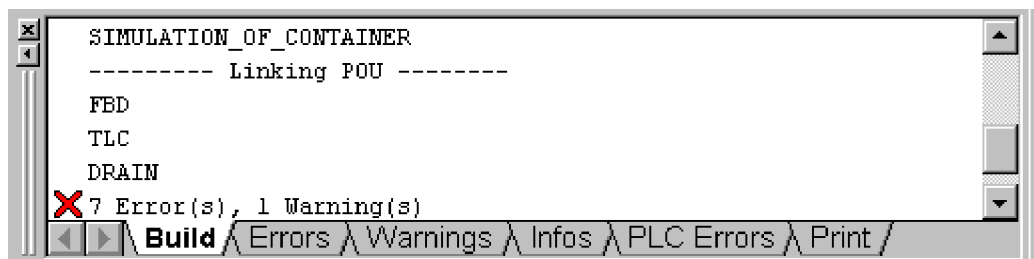


Figure 128: Announced errors after making a project

To display the detected errors, click on the tab 'Errors' in the message window. The error list is then shown in the message window.

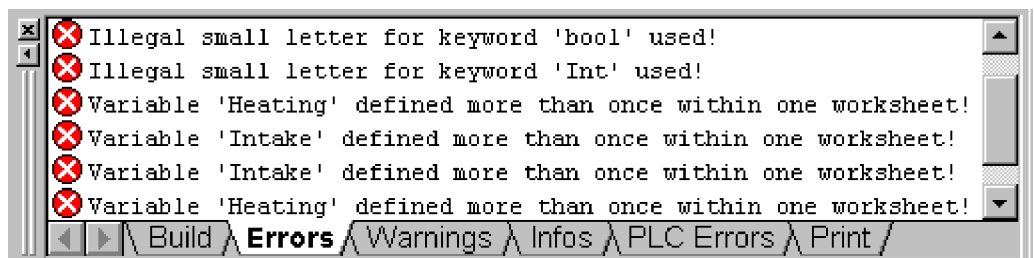


Figure 129: Error list, displayed in the message window

In order to display the list of warnings, just click on the tab 'Warnings'.

In most cases double clicking on a displayed error/warning will open directly the worksheet where the programming error/the reason for the warning is occurred. The corresponding line or the object is marked. You can also mark the error and press <F1> to get

the corresponding help topic with information about the cause of the error and what steps have to be done now.



NOTE

Having corrected the first error you can press <CTRL> + <F12> to go directly to the worksheet with the next error.

11.5 Patching POU's

To patch a POU means, that changes you have done while debugging a project are compiled, the corresponding code is generated and downloaded automatically to the PLC in one step. While downloading the changed project, it is not necessary to stop the program execution on the PLC.

If you have detected a programming error using the online mode and you have switched to the offline mode to remove the programming error you can use 'Patch POU' to compile these changes. The changes are downloaded automatically while the PLC keeps on running so that you can view them immediately having switched into online mode.

The following table lists the cases where Patch POU can be used.

Language	You can use Patch POU if ...
all	<ul style="list-style-type: none"> * new local or global variables have been inserted. * you have inserted a new function call if the function has already been called in the POU before. * you have inserted a function block call if the function block has already been instantiated in the POU before. * new, empty lines has been inserted. * comments have been changed or added. * you have deleted variables which are not used in *.csv.
IL	<ul style="list-style-type: none"> * you have changed or inserted IL operators. * you have changed the nesting level. * new local variables have been declared.
ST	<ul style="list-style-type: none"> * you have changed statements or expressions.
FBD	<ul style="list-style-type: none"> * existing networks have been changed.
LD	<ul style="list-style-type: none"> * existing networks have been changed.
SFC	<ul style="list-style-type: none"> * FBD networks, LD networks or variables connected to direct transitions have been changed. * the time interval of time action qualifier has been changed. * the variable name in action blocks has been changed.

Figure 130: Premises for the use of the 'Patch POU' operation

NOTE



If you want to use 'Patch POU' for POUs where new variables have been added, you should activate the POU memory reserve for Patch POU. Therefore activate the checkboxes 'Use reserve' in the properties dialog of the POU and in the dialog 'Resource settings'. In the dialog 'Data Area' the reserve size can be set.

Patch POU cannot be used in the following cases:

- * after changing string constants, user defined strings, variables, the physical hardware and the formal parameters of functions and function blocks (VAR_INPUT, VAR_OUTPUT and VAR_IN_OUT).

- * after inserting new string constants, user defined strings, functions, function block instances, POU's, libraries and CASE statements without using the reserve.
- * after deleting POU's or libraries.

NOTE



The menu item 'Patch POU' is only available if you switch the worksheet in offline mode by clicking on the icon 'Debug on/off' in the toolbar.



Patching POU's with the mouse

- Make sure that the worksheet is the active window.
- Make sure that the worksheet is in offline mode. In offline mode, the corresponding icon 'Debug on/off' appears not pressed (as shown beneath).
- Edit your worksheet and correct any detected programming errors.
- Click on the submenu 'Build' and select the menu item 'Patch POU'. The compilation process is started and its progress is displayed in the message window. Detected errors and warnings can be displayed in the corresponding sheets of the message window.



The steps how to display error and warning messages and to open the corresponding worksheets are described in the previous section 'Compiling a project using Make'.

11.6 Downloading the project

Having compiled your project using 'Make' or 'Rebuild Project' you have to download it to the simulation or PLC. If you have patched a POU the new generated code is downloaded automatically. While patching the POU it is not necessary to stop the program execution on the PLC.

NOTE



The target of the downloading process is set in the dialog 'Resource settings...'. Choose the menu item 'PLC Settings...' in the context menu of the resource to call the dialog. You can see to which simulation or port the PLC program is sent.



Downloading the project with the mouse

- Click on the icon 'Show Control Dialog' in the toolbar.
The control dialog appears directly if only one resource is available.
If the project tree contains several resources, the dialog 'Select resource' appears. In this case choose the desired resource to which the project is to be downloaded and confirm the dialog to display the control dialog.



Downloading the project with the keyboard

- Press <CTRL> + <F10>.
The control dialog appears directly if only one resource is available.
If the project tree contains several resources, the dialog 'Select resource' appears. In this case choose the desired resource to which the project is to be downloaded and press <↓> to display the control dialog.



Figure 131: Control dialog



Using the control dialog

- Press the button 'Download'.
The dialog 'Download' appears.



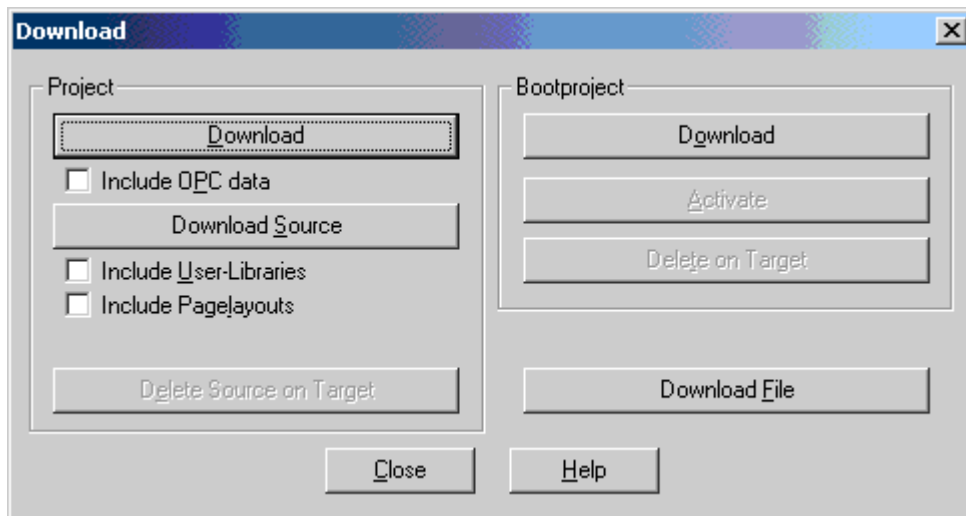


Figure 132: Dialog 'Download'

Using the dialog 'Download'



- Press the button 'Download' in the 'Project' area.
The full project including POU's and the configuration data is downloaded to the target. The PLC passes into the state 'STOP', which is displayed below the title bar in the control dialog.
- Press the button 'Cold' in the control dialog to start the program execution.

Please refer to the context-sensitive help for detailed information about downloading.



- In addition the online help contains detailed information about downloading a
- **bootproject** which is processed automatically by the PLC after the PLC is switched on or after a power failure.
 - **project source** which results in storing the zipped project files on the PLC. The downloaded source code can be uploaded again.
 - **file of any type** (e.g. an ASCII file) which can be stored on the PLC for future usage.

NOTE



The support of these functions depends on the target system. For further details contact the manufacturer of your PLC.

11.7 Calling worksheets in online mode

Having compiled and downloaded the project to the target it is possible to call the editors in online mode for debugging the worksheets. To call the worksheets in online mode you have the following possibilities:

- * activate the icon 'Debug on/off' in the toolbar or press <F10>. By activating this icon or pressing <F10> all worksheets already opened are switched automatically to online mode. If you open a new worksheet from the project tree or instance tree, this is also called in online mode. The icon 'Debug on/off' appears pressed if online mode is switched on. Refer also to the following note.
- * choose the menu item 'Open instance' in the context menu of a POU in the project tree or in the context menu of an already opened worksheet. Refer also to the following note.



NOTE

If already opened function block code bodies or program code bodies are instantiated several times and you want to debug these worksheets, i.e. calling in online mode by activating the icon 'Debug on/off', a warning message appears indicating that you have to use 'Open instance' to call these worksheets in online mode. In this case choose the menu item 'Open instance' in the context menu of the corresponding worksheets. Choosing this menu item the dialog 'Open Instance' appears where you have to choose the desired instance. The dialog 'Open Instance' also appears if online mode is switched on and you want to open a function block code body or program code body from the project tree, which is instantiated several times.

For the next steps let us assume that you want debug a textual worksheet in online mode.



Calling a textual worksheet in online mode with the mouse

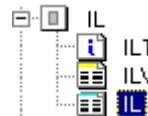
- If the desired worksheet is not yet opened, open the corresponding subtree in the project tree, containing the worksheet. For this purpose double click on the POU name.
- Double click on the worksheet icon (e.g. 'IL'). The worksheet is opened.
- Click on the icon 'Debug on/off'. All opened worksheets are switched to online mode. If online mode is switched on, the icon appears 'pressed'.





Calling a textual worksheet in online mode with the keyboard

- If the desired worksheet is not yet opened, open the corresponding subtree in the project tree, containing the worksheet as follows: Press <↓> or <↑> to highlight the POU name and press <→> to open the subtree.
- Press <↓> or <↑> to mark the worksheet icon (e.g. 'IL') and then press <↵>. The worksheet is opened.
- Press <F10>. All opened worksheets are switched to online mode. If online mode is switched on, the icon 'Debug on/off' appears 'pressed'.



```

1      FALSE LD      %IX0.2      (*direct variables*)
2      FALSE AND     %IX0.3
3      FALSE OR      Action_INIT
4      FALSE ST      IL_VAR
5
6      FALSE LD      Input_IX0_0
7      JMPC  MANUAL
8
9      (* Timer FB TON *)
10     TRUE  LD      Timer_start
11     TRUE  ST      TON_IL.IN
12     1.500 LD     PT_TON_IL
13     1.500 ST     TON_IL.PT
    
```

Figure 133: IL worksheet in online mode

In textual worksheets in online mode a gray line separates the worksheet into two parts. On the left the online values are displayed. On the right the code body is shown.

NOTE



You can change the width of the columns by positioning the mouse pointer on the gray separation line (when the correct position is reached, the cursor changes to a double line), pressing and holding the left mouse button while moving the cursor to the left or to the right.

In the following figure an example for a graphical worksheet in online mode is shown:

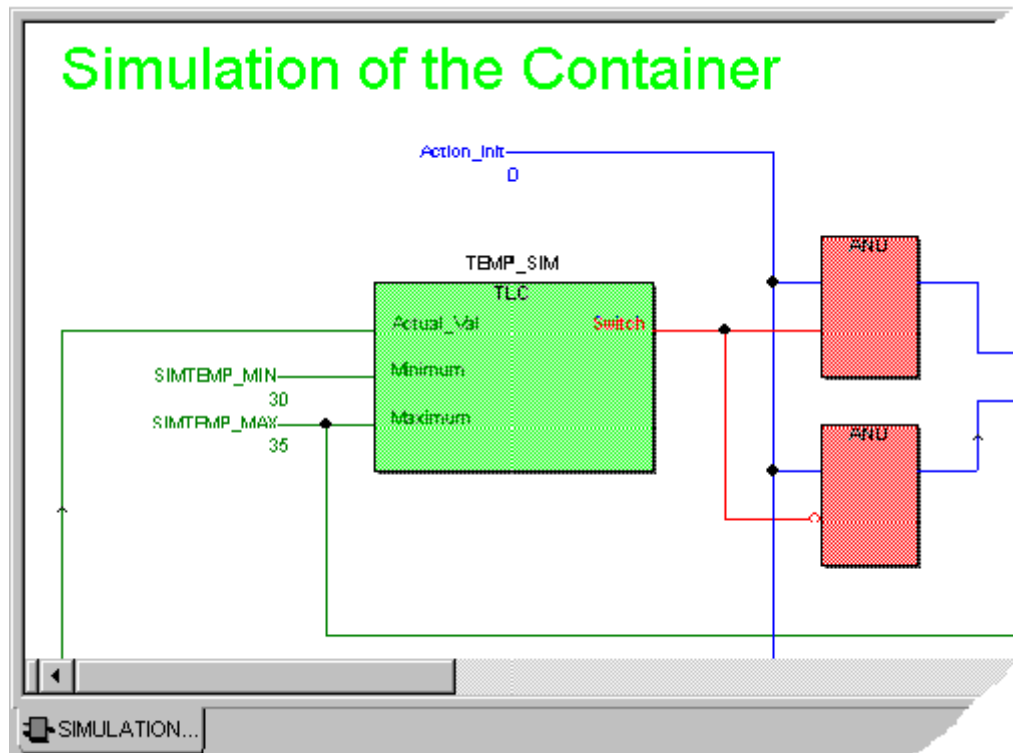


Figure 134: Graphical worksheet in online mode

The colors which are used in textual and in graphical worksheets have the following meanings:

Color	Meaning
Red	Boolean true
Blue	Boolean false
Green	integer values
Red	a set breakpoint
Orange	a reached breakpoint

Figure 135: Meaning of the colors in online mode

NOTE



In graphical worksheets the way how the online values are displayed can be changed choosing the menu item 'Online Layout' in the submenu 'Online'.



Calling an user library worksheet in online mode with the mouse

As it is possible to open the code body worksheets of announced user libraries in view mode (i.e. the worksheets can be viewed but not edited), it is also possible to call them in online mode. This means, they are treated like "normal" code body worksheets when debugging the project.



NOTE

Worksheets of firmware libraries cannot be opened, neither in view mode nor in online mode.



- Mark the icon of the library the worksheets you want to view. If not already opened double click on the 'Libraries' node icon.
- Browse to the desired library and open the POU folder.
- Double click on the desired user library worksheet icon. The worksheet is opened.
- Click on the icon 'Debug on/off'. All opened worksheets are switched to online mode. If online mode is switched on, the icon appears 'pressed'.



11.8 Adjusting the online layout and notation of online values

The programming system offers several possibilities for adjusting the representation of worksheets and values in online mode, in order to meet your requirements:

- * Switching the notation of online values in all online worksheets. The necessary steps are described in the following procedure.
- * Changing the online layout for graphical worksheets (please refer to the following note).
- * Highlighting feedbacks in graphical worksheets (please refer to the following note).
- * Displaying the execution order numbers in graphical worksheets (please refer to the following note).



For information about changing the online layout, highlighting feedbacks and displaying the execution order please refer to the corresponding topic in the PLC specific help file.



Switching the online value notation

This is done using the dialog 'Debug: *resource_name*'. Please note, that the selected value notation is applied to all graphical and textual worksheets.

- In an online worksheet mark any variable and open its context menu.
- Select the context menu item 'Debug dialog...'.
The dialog 'Debug: *resource_name*' appears.
- In the dialog area 'Value display' select one of the options 'Standard', 'Decimal', 'Hexadecimal' or 'Binary'.
- Confirm the dialog. The notation of the displayed online values is changed in all worksheets.

11.9 Switching between online and offline mode

If you have detected a programming error in any worksheet in online mode it is possible to switch to the offline mode to correct the programming error immediately. Having corrected the error the changes have to be compiled and sent to the target using 'Patch POU'.



Switching between online and offline mode with the mouse

- Click on the icon 'Debug on/off' in the toolbar.
All opened worksheets are switched automatically to offline mode. The icon appears not pressed if offline mode is switched on.
- Correct the programming errors.
- Choose the menu item 'Patch POU' in the submenu 'Build'. The changes you have done are compiled and sent to the target.
- Click on the icon 'Debug on/off' in the toolbar.
All worksheets are switched automatically to online mode. The icon 'Debug on/off' appears pressed if online mode is switched on.



Switching between online and offline mode with the keyboard

- Press <F10>.
All opened worksheets are switched automatically to offline mode. The icon 'Debug on/off' appears not pressed if offline mode is switched on.
- Correct the programming errors.
- Press <ALT> + <F9>.
The changes you have done are compiled and sent to the target.
- Press <F10>.
All worksheets are switched automatically to online mode. The icon 'Debug on/off' appears pressed if online mode is switched on.



NOTE

For information about calling function block code bodies and program code bodies in online mode, which are instantiated several times, refer to the previous section 'Calling worksheets in online mode'.

11.10 Switching to address status and powerflow

In online mode you use normally the variable status in which you can check the behavior of the variables. In the variable status you get the values stored in the I/O image at the end of a working cycle. But you can also switch to the address status. The address status displays the current values of the accumulator corresponding to the current moment of the program execution. Having activated the address status you see also the powerflow display. Powerflow means that it is displayed which program parts are actually executed and which ones not. This is e.g. important for debugging worksheets with conditional jumps.



NOTE

Activating the address status increases the cycle time of the considered program part. Therefore the address status should not be activated in the POU's of the bypass tasks and the event tasks!

It is not possible to switch to the address status without powerflow. The address status and powerflow can only be used together.

If you have opened a function worksheet in online mode the address status with powerflow is always switched on.



Switching to address status and powerflow with the mouse

- Click on the submenu 'Online'.
The submenu is opened.
- Choose the menu item 'Powerflow'.
The worksheet is switched to the address status with powerflow.

In the text editor in online mode powerflow is displayed with three different signs. The meaning of these signs is explained in the following table:

Powerflow sign	Meaning
Horizontal line	program part is not working
Vertical line	program part is working
Vertical, double line	program part is working in several POU's or in a loop, which is processed several times.

Figure 136: Meaning of the powerflow signs in text worksheets in online mode

In the graphic editor in online mode powerflow is displayed with colored lines. The meaning of these lines is explained in the following table:

Powerflow sign	Meaning
Red line	True
Blue line	False
Small, horizontal line	program part is working
Small, vertical line	program part is not working
Small, vertical double line	program part is working in several POU's or in a loop, which is processed several times.

Figure 137: Meaning of the powerflow signs in graphic worksheets in online mode

11.11 Forcing and overwriting variables

In online mode variables can be forced or overwritten. Forcing and overwriting means assigning to a variable a new value.

What is the difference between 'Forcing' and 'Overwriting'?

- * In case of **overwriting** the new value of the variable is only used for one working cycle. Having finished this cycle the variable is processed normally.
- * **Forcing** means using the new value for the variable until the forced variable is reset to its normal value by the user.

The steps to do for forcing and overwriting are nearly the same.

NOTE



Be very careful when forcing or overwriting variables while your PLC is running. Forcing and overwriting variables mean that the PLC program is executed with the values of the forced or overwritten variable.

Only physical inputs and outputs can be forced.

The support of these functions depends on the target system.
For further details contact the manufacturer of your PLC.



Forcing variables with the mouse

- Choose the menu item 'Debug dialog...' in the context menu of the variable in your worksheet in online mode.
The dialog 'Debug: *resource_name*' appears.



Forcing variables with the keyboard

- Press the cursor keys to go to the variable in your worksheet in online mode.
- Press <SHIFT> and keep it pressed. Press <→> or <←> to mark the variable.
- **In the graphical editor:**
Press <↓>.
The dialog 'Debug: *resource_name*' appears.

In the text editor:

Choose the menu item 'Debug dialog...' in the context menu of the variable in your worksheet in online mode.

The dialog 'Debug: *resource_name*' appears.

Or

Double-click on the corresponding online value of the variable in online mode.

The dialog 'Debug: *resource_name*' appears.

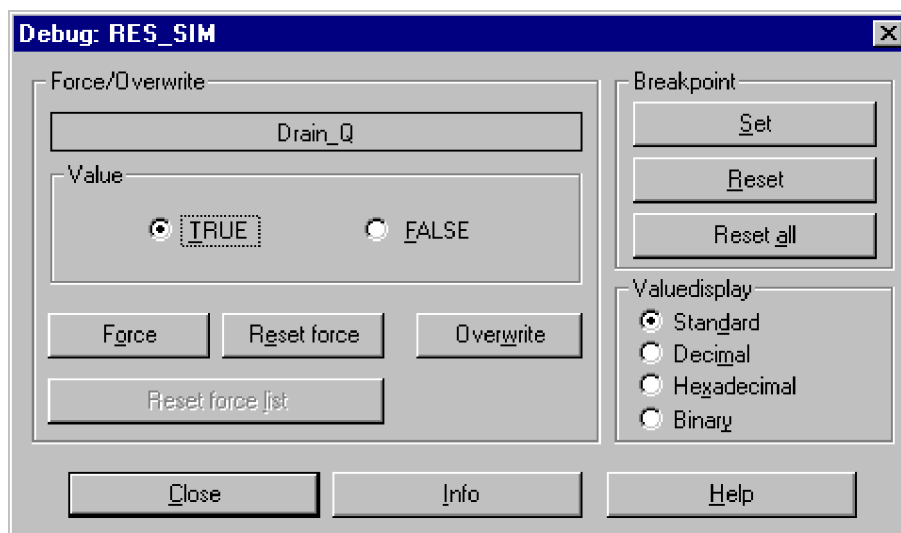


Figure 138: Dialog 'Debug: *resource_name*'



Using the dialog 'Debug: *resource_name*'



- Activate the radio button 'TRUE' or 'FALSE'.
If you force a non-Boolean variable the dialog includes an input field instead of the radio buttons. In this case enter the desired value.

NOTE



The dialog stores the value of a data type. In case of BOOL the actual value is read from the SPS and the opposite value is suggested, that is if the actual value is 'true', the value 'false' will be marked in the dialog.



- Press the button 'Force'.
- Confirm the dialog.
The variable is forced.



The programming system allows to reset a particular forced variable or all forced variables at the same time.



Resetting forced variables

- Choose the menu item 'Debug dialog...'
in the context menu of the forced variable in your worksheet in online mode.
- The dialog 'Debug: resource_name' appears as shown in [▶Figure 138:◀](#) on page 191.
- In the dialog area 'Force/Overwrite' press the button
 - 'Reset force' to reset the marked variable.
 - 'Rest force list' to reset all forced variables at the same time.

11.12 Setting and resetting breakpoints

In online mode breakpoints can be set or reset in code body worksheets. If a breakpoint is set the program execution halts at this point.

The control dialog now offers several new buttons to control the PLC in debug mode: The program execution continues at the breakpoint if the button 'Go' in the control dialog is pressed. To restart the program execution at the program beginning press the button 'Restart' in the control dialog.



NOTE

Be very careful using breakpoints while your PLC is running. Breakpoints mean that the program execution is halted at the point where the breakpoint has been set.

The behavior of the I/O's when reaching a breakpoint depends on the PLC type.

Breakpoints cannot be set in function block instances. Setting a breakpoint in an instance means automatically that the program execution is halted each time when the function block is used.



Setting a breakpoint with the mouse

- Double click on a line or an object in your code body worksheet in online mode.
The dialog 'Debug: resource_name' appears.



Setting a breakpoint with the keyboard

- Press the cursor keys to go to the line or object in your code body worksheet in online mode.

- **In the graphical editor:**

Press <↵>.

The dialog 'Debug: *resource_name*' appears.

- **In the text editor:**

Choose the menu item 'Debug dialog...' in the context menu of the variable in your worksheet in online mode.

The dialog 'Debug: *resource_name*' appears.

Or

Double-click on the corresponding online value of the variable in online mode.

The dialog 'Debug: *resource_name*' appears.

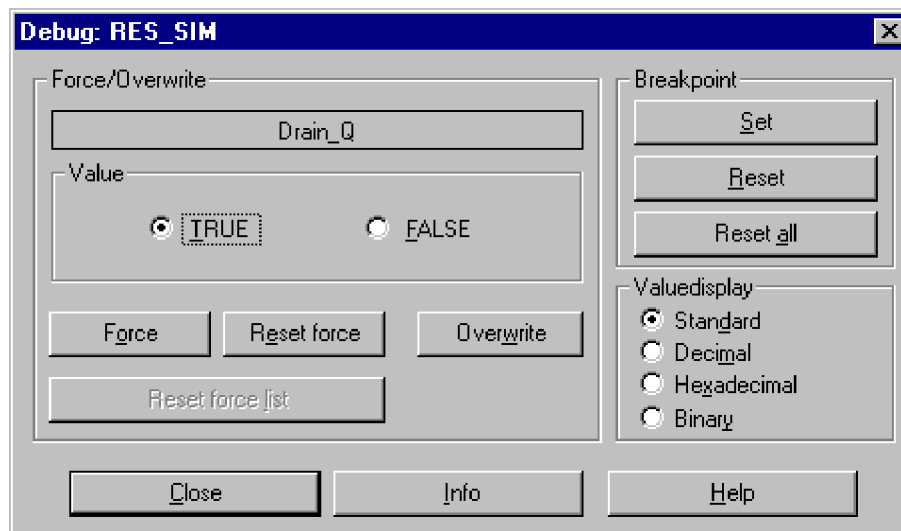


Figure 139: Dialog 'Debug: *resource_name*'



Using the dialog 'Debug: *resource_name*'

- Press the button 'Set'.
The breakpoint is set.



NOTE

In any worksheets a set breakpoint is represented in red, a reached breakpoint in orange color.





If you have set any breakpoints the programming system provides several functions for debugging your PLC program. These functions are described in the following section 'Debugging with set breakpoints'.

The programming system allows to reset a particular breakpoint or all breakpoints at the same time.



Resetting breakpoints

- **In the graphical editor:**

Press <_J>.

The dialog 'Debug: *resource_name*' appears.

- **In the text editor:**

Choose the menu item 'Debug dialog...' in the context menu of the variable in your worksheet in online mode.

The dialog 'Debug: *resource_name*' appears.

Or

Double-click on the corresponding online value of the variable in online mode.

The dialog 'Debug: *resource_name*' appears as shown in [▶Figure 139:◀](#) on page 193.

- In the dialog area 'Breakpoint' press the button
 - 'Reset' to reset the breakpoint for the marked object.
 - 'Reset all' to reset all set breakpoints at the same time.

11.13 Debugging with set breakpoints

The programming system provides a step and trace function which can be used for debugging your PLC program. These functions allow to continue the program execution line by line after a breakpoint has been reached.

What is the difference between stepping and tracing?

- * **Stepping** means that if a function or a function block call is reached, the call is stepped over, i.e. the code body of the function or function block is not debugged.
- * **Tracing** means that if a function or a function block call is reached, the function or function block code body is opened for debugging.

Stepping and tracing is performed using the control dialog.

NOTE



Stepping and tracing is only possible if a breakpoint has been set.

For the next steps let us assume that you have set a breakpoint in a ST code body worksheet in online mode.



Stepping and tracing after a breakpoint is reached using the mouse

- If the control dialog is not visible, click on the icon 'Show Control Dialog' in the toolbar.
The control dialog appears.



Stepping and tracing after a breakpoint is reached using the keyboard

- If the control dialog is not visible, press <CTRL> + <F10>.
The control dialog appears.



Figure 140: Control dialog if a breakpoint is set



Using the control dialog



- Press the button 'Step' to go to the next line or object and to halt the program execution at this point. If a function or function block call is reached, the call is stepped over and the next line is highlighted.
- Press the button 'Trace' to go to the next line or object and to halt the program execution at this point. If a function or function block call is reached, the function or function block code body is opened for debugging.

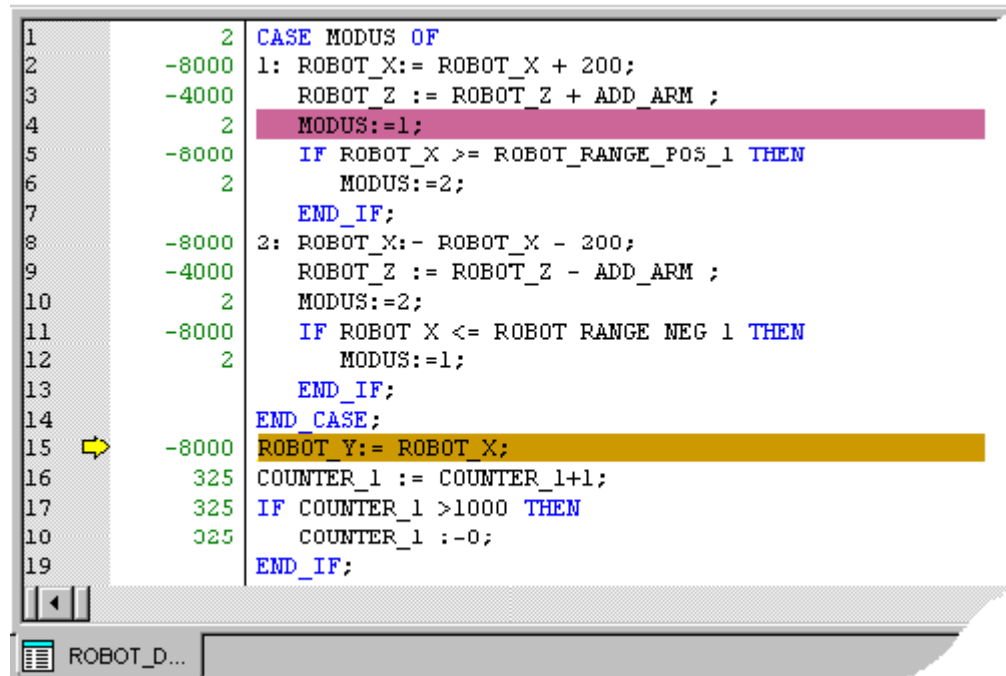
NOTE



The line or the object where the program execution is halted after pressing 'Step' or 'Trace' is highlighted in orange color. In addition in textual worksheets a yellow arrow is used to highlight the line where the program execution is halted (as shown in the following figure).

11.14 Using the watch window

The following figure displays an example for a ST worksheet in online mode with two set breakpoints (line 4 and line 15) and halted program execution (line 15).



```
1      2  CASE MODUS OF
2      -8000 1: ROBOT_X:= ROBOT_X + 200;
3      -4000 ROBOT_Z := ROBOT_Z + ADD_ARM ;
4      2     MODUS:=1;
5      -8000 IF ROBOT_X >= ROBOT_RANGE_POS_1 THEN
6      2     MODUS:=2;
7      END_IF;
8      -8000 2: ROBOT_X:= ROBOT_X - 200;
9      -4000 ROBOT_Z := ROBOT_Z - ADD_ARM ;
10     2     MODUS:=2;
11     -8000 IF ROBOT X <= ROBOT RANGE NEG 1 THEN
12     2     MODUS:=1;
13     END_IF;
14     END_CASE;
15     -8000 ROBOT_Y:= ROBOT X;
16     325  COUNTER_1 := COUNTER_1+1;
17     325  IF COUNTER_1 >1000 THEN
18     325  COUNTER_1 :=-0;
19     END_IF;
```

Figure 141: ST worksheet in online mode with set breakpoint

11.14 Using the watch window

The watch window can be used to collect variables from different worksheets in online mode to get an appreciation about how these variables work together. If a variable is added once to the watch window, the associated worksheet must not be opened to monitor the current value. Furthermore the watch window is used to debug elements of user defined data types such as arrays and structures.

NOTE



The procedures how to debug user defined data types using the watch window is described in the following section 'Debugging user defined data types using the watch window'.

In the watch window you can insert and delete variables. There is no limit for the number of variables inserted in the watch window. In addition you can use the watch window to force and overwrite variables. This is performed by choosing the menu item 'Debug dialog...' in the context menu of a variable.



NOTE

If you are calling the debug dialog from the watch window, it is not possible to set or reset breakpoints for the marked variable. For that purpose you have to call the debug dialog using the context menu of the corresponding variable.

For the next description let us assume that you want to insert a variable to the watch window. The first step to do is to call the watch window and then to write the variable into it.



Calling the watch window with the mouse

- Click on the icon 'Watch Window' in the toolbar.
Depending on the previous state, the message window is now visible or hidden.



Calling the watch window with the keyboard

- Press <ALT> + <F10>.
Depending on the previous state, the message window is now visible or hidden.

The watch window is displayed as follows:

Variable	Value	Type	Instance
Level	750	INT	C_IPC.R_IPC.T_100ms.PRG_FBD.Level
Temp	34	INT	C_IPC.R_IPC.T_100ms.PRG_FBD.Temp
Intake Q	FALSE	BOOL	C_IPC.R_IPC.T_100ms.PRG_FBD.Intake_Q
Heater Q	FALSE	BOOL	C_IPC.R_IPC.T_100ms.PRG_FBD.Heater_Q
Drain Q	TRUE	BOOL	C_IPC.R_IPC.T_100ms.PRG_FBD.Drain_Q
Container States		Container_Struct_Array	C_IPC.R_IPC.T_100ms.PRG_FBD.Container_States

◀ Watch 1 | Watch 2 | Watch 3 | Watch 4 ▶

Figure 142: Watch window with several variables inserted

The watch window contains the following information:

- * 'Variable': Displays the variable name. User defined data types such as arrays and structures are marked with a '+' (as for example 'Container States' in the above figure). To display the elements of these data types, click on the '+' sign. Normal variables are displayed such as 'Level' in the above example.
- * 'Value': Displays the current value of the variable.
- * 'Type': Displays the data type.
- * 'Instance': Displays the instance path where the variable is used. The path always contains the configuration, resource, task, the associated program name and variable name.

The watch window allows to manage several pages where every page can be used independently. The individual pages are called by clicking on the sheet tabs at the bottom of the watch window.



Inserting variables in the watch window with the mouse

- Mark the variable in the worksheet in online mode.
- Click the right mouse button to open the context menu.
- Choose the menu item 'Add to Watch Window'.
The variable is inserted in the watch window.

If you want to delete variables in the watch window, perform the following procedure.



Removing variables from the watch window with the mouse

You can either remove a single variable from the watch list or clear the whole list.

- * To **remove a single variable** from the watch list, mark the variable to be deleted and press the key . Alternatively you can open its context menu using the right mouse button and select the context menu item 'Delete'. This is also possible for a single element within a user defined structure or array.
- * To **remove all variables** open the context menu of any variable in the watch window and select the menu item 'Clear'.

11.15 Debugging user defined data types using the watch window

For debugging user defined data types such as arrays and structures the watch window has to be used. The first step is to call the watch window and then to write the variable being an array or a structure into it. The second step is debugging the elements of the array or structure in the watch window.

For the next steps let us assume that you want to debug an array of structure. The first thing to do is to call the watch window and to insert the variable being the array of structure as it is has been described in the previous section 'Using the watch window' of this chapter. [▶Figure 142:◀](#) on page 197 shows how an array of structure (e.g. 'Container States') is displayed in the watch window. Having inserted the array of structure in the watch window the components of the array of structure must be displayed.



Debugging an array of structure with the mouse

- In the watch window click on the '+' sign associated to the array of structure.

+ ...	Container States
-------	------------------

The components of the array of structure are displayed.

- Click on the '+' sign associated to a structure (e.g. [0]).

- ...	Container States
+ ...	[0]
+ ...	[1]

The components of the structure are displayed as shown in the following figure.

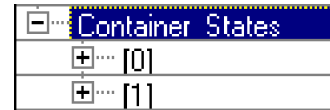


Debugging an array of structure with the keyboard

- In the watch window press <↑> or <↓> to go to the line with the array of structure.



Press <→> to display the components of the array of structure.



- Press <↓> to go to a structure (e.g. [0]) and press <→> to display the components of the structure.

The components of the structure are displayed as shown in the following figure.

Variable	Value	Type	Instance
[-] Container States		Container_Struct_Array	C_IPC.R_IPC...
[-] [0]		Container_Struct	C_IPC.R_IPC...
[+] Drain	FALSE	BOOL	C_IPC.R_IPC...
[+] Heater	TRUE	BOOL	C_IPC.R_IPC...
[+] Intake	FALSE	BOOL	C_IPC.R_IPC...
[+] Level	650	INT	C_IPC.R_IPC...
[+] Temp	30	INT	C_IPC.R_IPC...
[+] State of container	Intake: CLOSE Heater: ON Drain: CLOSE	STRING	C_IPC.R_IPC...
[+] Values of container	Level: 650 Liter Temperature: 30 Grad	STRING	C_IPC.R_IPC...
[+] [1]		Container_Struct	C_IPC.R_IPC...

Figure 143: Watch window with opened array of structure

NOTE



Having displayed the components of an user defined data type you can use the watch window to force and overwrite variables and to set and reset breakpoints. This is performed by choosing the menu item 'Debug dialog...' in the context menu of a variable in the watch window.



The procedures how to force and overwrite variables and to set and reset breakpoints are described in the sections 'Forcing and overwriting variables' and 'Setting and resetting breakpoints' in this chapter.

11.16 Using the Logic Analyzer

The Logic Analyzer is a powerful tool for recording values of variables during a certain time interval which the user has to determine by a specified number of sample cycles.

Using this feature you can check the behavior of your PLC program by recording variables (while the PLC runs) and displaying them graphically in the Logic Analyzer window. It is possible to display several curves of variable values together depending on their trigger conditions and other settings.

All recorded values and settings of the Logic Analyzer are stored automatically with the project.

The system supports the use of the Logic Analyzer with one or several resources. Each resource gets a separate Logic Analyzer worksheet. A new resource is added or deleted in the Logic Analyzer window after compiling the project.

NOTE



Recording values is only possible in online mode.



Calling the Logic Analyzer window with the mouse

- Click on the icon 'Logic Analyzer' in the toolbar. Depending on the previous state, the Logic Analyzer window is now visible or hidden.



Calling the Logic Analyzer window with the keyboard

- Press <ALT> + <F11>. Depending on the previous state, the Logic Analyzer window is now visible or hidden.

The window is empty after you have opened it **for the first time** because no variables are added to the Logic Analyzer.

The following figure shows the Logic Analyzer window after capturing the curves for the variables `Level` and `Temp`.

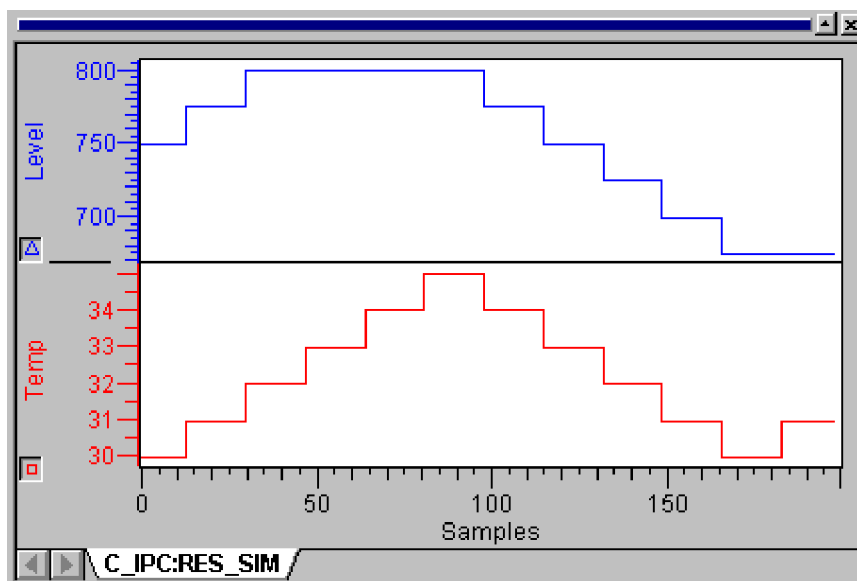
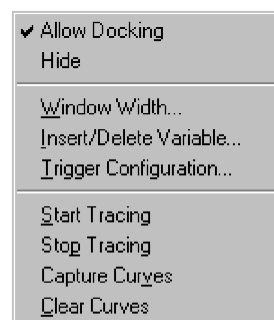


Figure 144: Logic Analyzer window after capturing two curves

By clicking on the bottom bar of the Logic Analyzer window with the right mouse button you can open the Logic Analyzer context menu. The menu items provide all available commands and dialogs of the Logic Analyzer which are represented as icons in the toolbar too. In the following descriptions the toolbar icons are used.



The following procedure describes how to use the Logic Analyzer, i.e. how to add variables, to set the trigger configuration and start the recording process, etc.



If you need more detailed information please refer to the general PLC help file.



NOTE

Prerequisites for using the Logic Analyzer:

- Before calling the Logic Analyzer make sure that
- the compiled project is downloaded to the PLC.
 - the PLC or the simulation is in running state.
 - the system is in online mode.

After the Logic Analyzer window is visible on the screen, you first have to add the variables to the Logic Analyzer for which the values should be recorded. When starting the Logic Analyzer for the first time, no variables are added. To connect a variable please perform the following steps:



Adding variables to the Logic Analyzer

- Make sure that the system is in online mode.
- Select the desired variable in any textual or graphical code body worksheet or in a variable worksheet with a right mouse click. The context menu appears.
- Select the menu item 'Add to Logic Analyzer'.
- The marked variable is added to the Logic Analyzer window.
- Repeat these steps for adding new variables as often as required.



Removing variables from the Logic Analyzer

Maybe it is necessary to remove one or several variables from the Logic Analyzer which are already added but no longer used. For that purpose you have to call the dialog 'Variables' of the Logic Analyzer.

- Click on the icon 'Insert/Delete Variable...' in the toolbar.
The dialog 'Variables' appears containing a list of all variables which are connected to the Logic Analyzer.

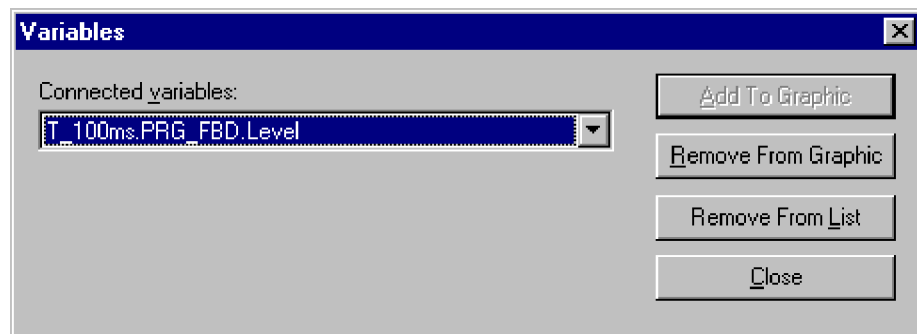


Figure 145: Dialog 'Variables' of the Logic Analyzer window

The dialog 'Variables' provides two ways for deleting a single variable in the Logic Analyzer:

* **Removing from the Logic Analyzer window:**

The variable does not yet appear in the Logic Analyzer window but it is still in the list of variables. You can add the removed variable to the analyzer again by selecting the variable name in the list box of the dialog 'Variables' and pressing the button 'Add to Graphic'.

* **Removing from variable list of the Logic Analyzer:**

Removing from list means, that the variable is deleted in the graphical presentation and in the variable list of the Logic Analyzer. If you need the deleted variable later

again, you have to add it again using the variables context menu.
To remove a variable, select it in the list box and press the desired button.



Setting up the Logic Analyzer window width

Now the Logic Analyzer contains the desired variables. The next step is to set up the Logic Analyzer window width, in order to define the x-Axis.

- Open the Logic Analyzer context menu and select the menu item 'Window Width'.
The dialog 'Data Window' appears.

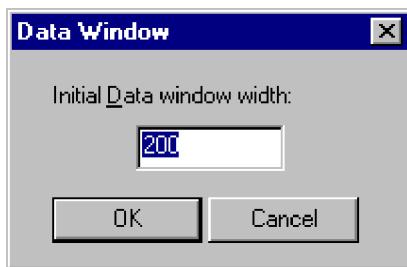


Figure 146: Dialog 'Data Window' of the Logic Analyzer window

- Set the initial window width by entering a value into the text field. The default value is 200 samples.



Defining the trigger configuration

An important step before recording values using the Logic Analyzer is to set the trigger configuration.

- Click on the icon 'Trigger configuration' in the toolbar.
The dialog 'Trigger configuration' appears.



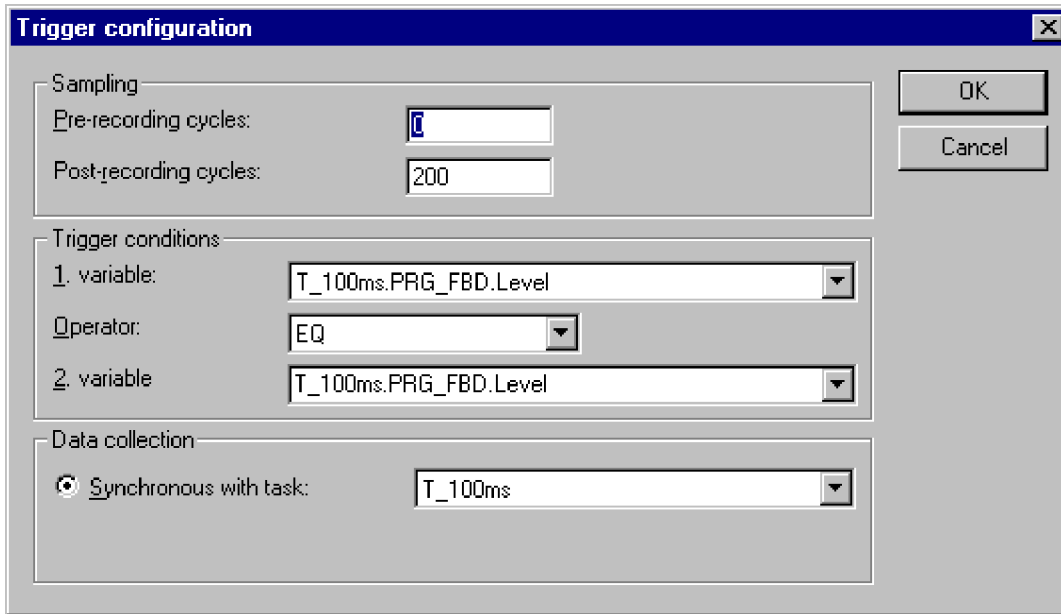


Figure 147: Dialog 'Trigger configuration' of the Logic Analyzer



For detailed information about the meaning of each dialog field and button please refer to corresponding topic in the general PLC help file.

- In the dialog area **Sampling** you have to determine the time interval in which the Logic Analyzer records values. This time interval is determined by the number of sample cycles. Enter the number of sample cycles before and after the trigger event in which the Logic Analyzer is intended to record values.
- In the dialog area **'Trigger conditions'** select the '1. Variable' and the '2. Variable' as well as the 'Operator' which specifies the trigger condition depending on the first and the second variable value.

NOTE



When setting the trigger condition make sure that the selected variables and the operator are combined in a way which makes it possible that the trigger condition becomes true. If the trigger condition does not become true the Logic Analyzer cannot record values although the message 'recording samples...' is displayed.

Example: If '1. Variable' = '2. variable' and 'Operator' = NE the trigger condition **never** becomes true.

- In the dialog area **'Data collection'** specify a task. The Logic Analyzer records values synchronously to the execution of the task selected in this list box. Therefore it is important that the trigger conditions and the selected task relate to each other.



Starting the recording of values

Once the trigger conditions are set, you can start the recording of values for the variables connected to the Logic Analyzer.

- Click on the icon 'Start recording values' in the toolbar.
The Logic Analyzer starts recording values for the time interval defined in the dialog 'Trigger configuration'.
In a valid record process the recording of values stops automatically at the end of the defined sample cycles.



Aborting the recording of values

Assuming that you have entered a wrong number of cycles or any other wrong parameter you can cancel the recording before the time interval has elapsed.

- To abort the recording press the icon 'Stop recording values' in the toolbar.



Adjusting the Logic Analyzer window

After recording one or several variables the value curves are automatically displayed in the Logic Analyzer window. There are several possibilities to change the graphical presentation.

Stretching or compressing the vertical or horizontal axis

You can stretch or compress the value axis of each variable in order to zoom in or out. For that purpose

- press the left or right mouse button on the desired axis and keep it pressed while adjusting the axis by moving the mouse.

Shifting the visible range of the vertical or horizontal axis

In addition it is possible to shift the visible range. This is useful if you have zoomed in the axis (by stretching it) and therefore the required values are shifted out of the visible range. For that purpose

- press the right and the left mouse button simultaneously on the desired axis and keep it pressed while shifting the axis range by moving the mouse.

NOTE



Shifting the range is only possible if the range is larger than the displayed range.

Reading off certain values within the Logic Analyzer graphic

To get information about exact variable values you can use a horizontal or a vertical cursor line.

For that purpose,

- move the mouse cursor to the top/bottom boundary (for getting a horizontal line) or to the left/right boundary (in order to set a vertical line).
A double line symbol is added to the mouse cursor.
- Click the left mouse button and keep it pressed while dragging the cursor line into the signal curve.

After the cursor line is moved to the desired point in the window, a little message window appears, showing you the values at this coordinates.

When using a **vertical cursor line**, the window shows the values of all variables within the window in relation to the number of samples. In the example shown beside, the Logic Analyzer contains two variables: Temp and Level.

[Samples]:	118.3
△ Temp :	4.00
□ Level :	100.0

If you have set a **horizontal line**, the window shows the values of all variables in the window without their relation to the x-axis, as shown beside.

[Temp]:	29.695
[Level]:	570.1



Clearing the Logic Analyzer window

If you want to delete all curves and all listed variables from the Logic Analyzer in one step proceed as follows:

- Open the Logic Analyzer context menu.
- Select the menu item 'Clear curves'.

After selecting this menu item, you have to add new variables to the Logic Analyzer before you can start a new recording.

PRINTING YOUR PROJECT WITH A CUSTOMIZED PAGE LAYOUT

12.1 Printing the project

The submenu 'File' contains the commands used to define the printer settings, to display the preview of the current page and to print the entire project or parts of it.

The program offers several possibilities to print your project documentation. To set the printing options, the program provides two dialogs:

- In the dialog 'Print Project' you can select the items to be printed and the used print mode.
- On the page 'Default Page layout' in the dialog 'Options' you can define the page layout which is used to print the data.

The above mentioned dialogs and the various print modes are described in the following sections.

12.1.1 Controlling the print process using the dialog 'Print Project'

If you want to print only parts of your project, you first have to select the desired folder icon in the project tree (e.g. a POU icon) before calling the dialog 'Print Project'.

NOTE



It is not possible to print libraries and worksheets in online mode.



Selecting the desired icons in the project tree

- Select the desired folders in the project tree, which you want to print. This could be for example one or several worksheets (see note below), a single POU or the whole subtree 'Logical POUs'. If you want to print the whole project, skip this step and proceed with the next step.



NOTE

How to mark several adjacent objects in the project tree:

Mark the first object in the project tree by clicking on it with the left mouse button. Then press the <SHIFT> key and keep it pressed. Click the last object you want to mark and release the <SHIFT> key. All objects between the first and the last marked icon are marked.

How to mark several objects which are not side by side:

Mark the first object in the project tree by clicking on it with the left mouse button. Press the <CTRL> key and keep it pressed. Click on each object you want to mark.



Displaying the page borders

Before printing it is recommended to display the page borders for the worksheets to be printed. If the page borders are activated they are displayed as blue lines.

Especially for graphical worksheets the borders allow you to check, whether the whole content of the worksheet fits on the page. For that purpose it may be necessary to zoom out of the worksheet, until the whole worksheet and the page borders are visible.

- Choose the menu item 'Borders' in the submenu 'Layout'. Please note, that page borders must be activated separately for each worksheet.



Calling the dialog 'Print Project'

- Select the menu item 'Print Project' in the submenu 'File'. The dialog 'Print Project' appears as shown in the following figure.

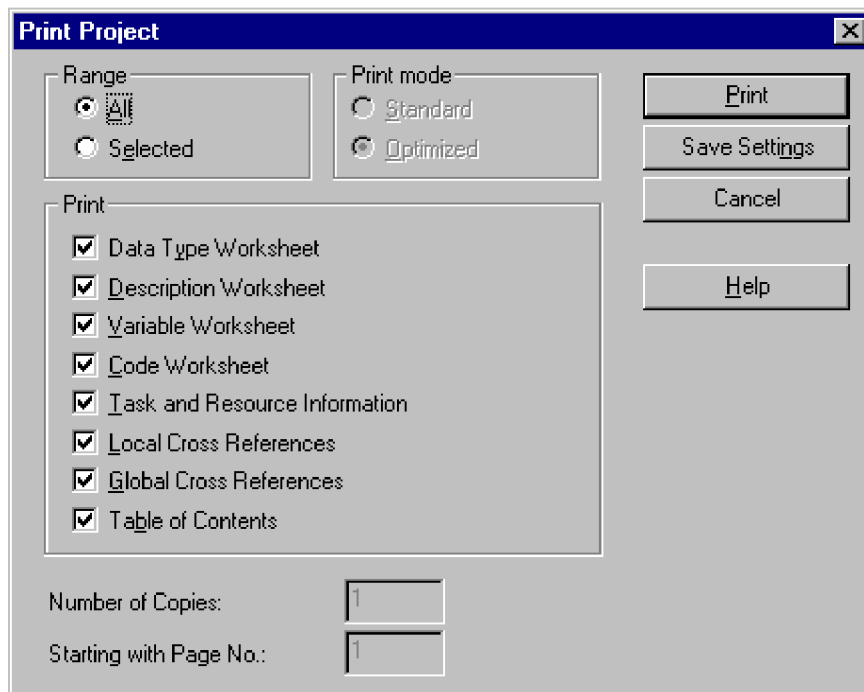


Figure 148: Dialog 'Print Project'



Using the dialog 'Print Project'

- Select the range to be printed.
 - 'All' means the entire project, including all subtrees in the project tree.
 - 'Selected' means, that only the icons which are marked in the project tree before calling the dialog 'Print Project' are printed.
- Select the items to be printed by activating/deactivating the corresponding checkboxes in the area 'Print'. Keep in mind, that each checkbox relates to the marked range ('All' or 'Selected').

Example: If you mark the checkboxes 'Description Worksheet', 'Variable Worksheet' and 'Code Worksheet' only those worksheets are going to be printed, which are within the selected range.

- Select the desired **print mode**:
 - '**Standard**' means, that each selected item is printed with the associated default page layout. In order to set the default page layout for the standard print mode call the dialog 'Options', open the page 'Default Page layout' and edit the fields 'Text' and 'Graphic'.
 - '**Optimized**' means that the selected items are printed with less paper as possible. All worksheets are printed one after the other. A new page is only used, if a new POU or a new step (in SFC) starts. While using optimized printing a table of contents and cross references can be printed. The default page layout which is used for optimized printing is set in the dialog 'Options' on the page 'Default Page layout' (list box 'Optimized Printing').



The steps to define a certain page layout as default page layout for the different print modes are described in the section 'Defining a page layout as default page layout' in this chapter.



The procedures how to create and edit a page layout are described in the section 'Editing a page layout' in this chapter.



- If desired, change the number of copies to be printed.
- Enter another page number for the first page to be printed.

NOTE



The number of copies and the first page number to be printed cannot be changed in the current system version.



- Confirm the dialog.
The selected data are printed on the standard printer.

12.1.2 Defining a page layout as default page layout

You can define a page layout as default page layout (e.g. a customized page layout) using the page layout editor. This default page layout is used automatically when printing your project or parts of it. The default page layout is set in the dialog 'Options'.



Calling the page 'Page layouts' in the dialog 'Options'

- Select the menu item 'Options' in the submenu 'Extras'.
The dialog 'Options' appears.
- Click on the tab 'Page layouts'.

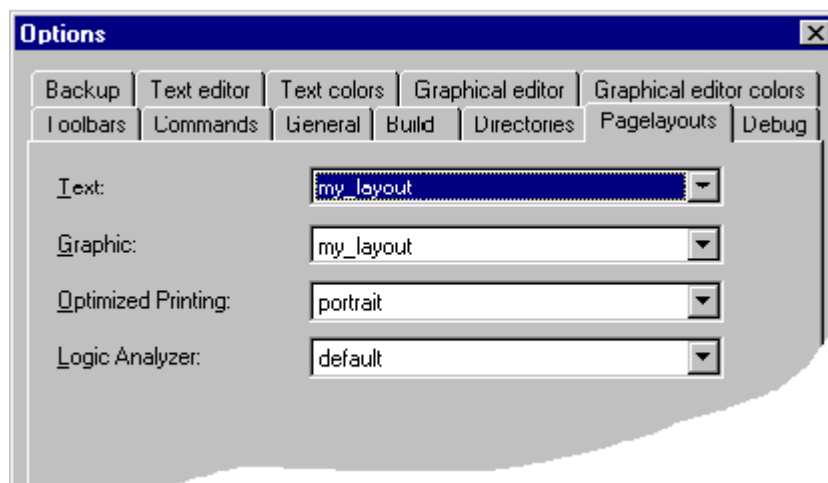


Figure 149: Page 'Page layouts' in the dialog 'Options'



Using the dialog 'Page layouts'

The page provides four list boxes used to define different default page layouts. Each list box contains all page layouts which are stored in the corresponding page layout directory. This can either be predefined layouts (delivered with the program) or customized layouts, which you have created using the page layout editor.

In [▶Figure 149:◀](#) on page 210 the customized page layout 'my_layout' is defined as default page layout for text and graphic in standard print mode. For the print mode 'Optimized' the predefined layout 'Portrait' is selected, the Logic Analyzer prints with the default page layout.



- Open each list box and select the desired page layout.
 - **'Text'**: Defines the page layout for textual worksheets, such as description worksheets, variable worksheets, textual code bodies, cross references, etc. The setting in the field 'Text' is only relevant when using the print mode 'Standard'.
 - **'Graphic'**: Defines the page layout for graphical worksheets, such as code bodies in FBD, LD or SFC. The setting in the field 'Graphic' is only relevant when using the print mode 'Standard'.
 - **'Optimized Printing'**: Defines the page layout for text and graphic when using the print mode 'Optimized'.
 - **'Logic Analyzer'**: Defines the page layout for the printed output of the Logic Analyzer.
- Confirm the dialog.

12.2 Using the page layout editor

The page layout editor is a tool which allows you to create new page layouts or edit existing page layouts.

After saving a new page layout, it is stored in the corresponding directory and can be selected as default page layout in the dialog 'Options'.

The first step before editing your own page layout is to call the page layout editor.



Calling the page layout editor

- Choose the menu item 'Page layout Editor' in the submenu 'Extras'.
The page layout editor is opened.

NOTE

The page layout editor is normally opened with the default page layout.



12.2.1 Creating a new page layout

To create a new page layout, perform the following steps:



Creating a new page layout with the mouse

- Choose the menu item 'New' in the submenu 'File'.
The page layout editor with the page layout 'untitled' is opened.



Creating a new page layout with the keyboard

- Press <ALT> + <F>.
The submenu 'File' is opened.
- Press <N>.
The page layout editor with the page layout 'untitled' is opened.

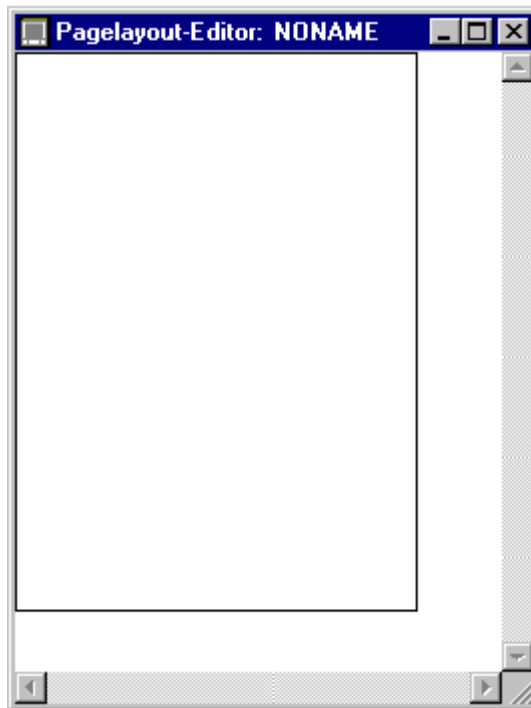


Figure 150: Page layout editor with a new page layout

12.2.2 Defining the source area

Having created a new page layout you should first define the area where you want to place the contents of the worksheets. This area is called 'source area' and is represented with a red rectangle.



Defining the source area with the mouse

- Click on the icon 'Source area' in the toolbar.
A rectangle is added to the shape of the cursor.
- Place the mouse cursor to the position of one intended source area corner.
- Press the left mouse button and keep it pressed.





- Move the mouse drawing a rectangle.
- Release the mouse button.
The source area is drawn.



Defining the source area with the keyboard

- Press <S>.
A rectangle is added to the shape of the cursor.
- Press the cursor keys to move to the position of one intended source area corner.
- Press <SPACE> and keep it pressed.
- Press the cursor keys to move to the position of the opposite corner of the rectangle.
- Release <SPACE>.
The source area is drawn.

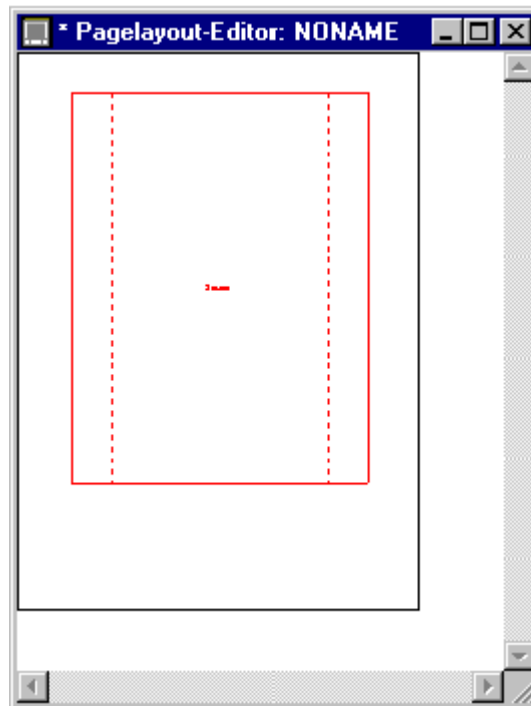


Figure 151: Page layout with the new source area

NOTE



To move the new source to another position on the page, mark it with a left mouse click and keep the mouse button pressed. Move the rectangle to the desired position by dragging the mouse.

To change the source code area size after creating it, click on the right mouse button. In the context menu choose 'Object properties...'. The dialog 'Settings source area' appears. Using this dialog you can change the position, size and other source area properties.

12.2.3 Inserting elements in your page layout

You can insert several elements in your page layout, such as rectangles, lines or texts. For the next steps let us assume that you want to insert a horizontal line at the bottom of the page.



Inserting a line with the mouse

- Click on the icon 'Line' in the toolbar.
A line is added to the shape of the cursor.
- Press the left mouse button and keep it pressed.
- Move the mouse drawing a line.
- Release the mouse button.
The line is drawn.



Inserting a line with the keyboard

- Press <I>.
A line is added to the shape of the cursor.
- Press <SPACE> and keep it pressed.
- Press the cursor keys to move to the position where you want the line to end.
- Release <SPACE>.
The line is drawn.



NOTE

The ways how rectangles, texts and bitmaps can be inserted in the page layout are described in the context-sensitive Help.

The bitmaps must be available as 16 color bitmaps!

12.2.4 Editing environment items

Another kind of objects which can be inserted in a page layout are environment items. Environment items are place holders for e.g. page numbers, the company name or other kind of texts. Several standard environment items are available which are called 'system items'.



Editing environment items with the mouse

- Click on the icon 'Environment items' in the toolbar.
A 'T' is added to the shape of the cursor.
- Click in the editing field where you want to insert the environment item.
The dialog 'Settings Environment Text' appears.





Editing environment items with the keyboard

- Press <V>.
A 'T' is added to the shape of the cursor.
- Press the cursor keys to go to the position where you want to insert the environment item.
- Press <SPACE>.
The dialog 'Settings Environment Text' appears.



Inserting an item using the dialog 'Settings Environment Text'



- Select the desired item.
- Press the button 'Font...!' to choose a new font or font size if you want.
- Confirm the dialog.
The place holder with the name of the item is inserted in the page layout.



NOTE

The system items and more possibilities how to modify or delete existing items are explained in the context-sensitive Help.

To view the contents of the place holder you can use preview.

12.3 Using preview

Preview allows to get an appreciation of the appearance of your worksheet as it would be for printing. It helps you to organize the elements on the page in a clear and structured way.

Preview means that the content of the active window is shown with the associated page layout. If you have not associated a page layout it is shown with the default page layout.



NOTE

Cross references are not displayed using preview.

It is not possible to use preview for worksheets in online mode.



Calling preview with the mouse

- Make sure that the desired worksheet is the active window.
- Choose the menu item 'Print Preview...' in the submenu 'File'.
Preview is called and the active worksheet is displayed.



Calling preview with the keyboard

- Make sure that the desired worksheet is the active window.
- Press <ALT> + <F>.
The submenu 'File' is opened.
- Press <V>.
Preview is called and the active worksheet is displayed.

SYSTEM LIMITS FOR PROPROG WT II

13.1 Project limits

Nodes in the project tree	8000
Configurations in the project tree	100
Resources in the project tree	100
Program instances per resource	1000
Tasks per resource	15
Program instances per task	500
Global variables	15000
Local variables per POU	15000
Amount of I/O parameters of functions and function blocks	128
Embedded libraries	30
POUs in a library	1000
POUs in a project	2000
Code body worksheets per POU	64
Variable worksheets per POU	20
Global variable worksheets per resource	20
Global program instance variable worksheets per resource	20
Amount of functions and function blocks of a different kind per POU	620

13.2 Supported programming languages

Functions and function blocks of the same kind per POU	1024
Jumps and labels per POU	750
Jumps and returns per network	20
Steps / transitions per POU	110
Actions per POU	165
Contacts / coils per POU	3000
Nesting of functions / function blocks	128
Nesting level for user-defined data types	8
Elements in a structure	200
Elements in an array	32766
Nesting level FOR / CASE / IF	8
I/O groups	200
Open online windows	30



NOTE

To get a better project structure, complex variable declarations should not be made in one single worksheet. It is better to distribute them to several worksheets. In principle it is possible to define the given number of variables in a worksheet. However, to receive maximum support of the variable declaration dialog when defining new variables, you should not use more than 5000 lines and 300000 bytes per variable worksheet.

In addition only one variable should be declared in a line, because a determinate display in the online mode is impossible.

13.2 Supported programming languages

Textual programming languages	<ul style="list-style-type: none">• Instruction List (IL)• Structured Text (ST)
Graphic programming languages	<ul style="list-style-type: none">• Function Block Diagram (FBD)• Ladder Diagram (LD)• Sequential Function Chart (SFC)

13.3 Data types

ANY_BIT	BOOL(1/8), BYTE(8), WORD(16), DWORD(32)
ANY_NUM	SINT(8), INT(16), DINT(32), USINT(8), UINT(16), UDINT(32), REAL(32) The limits for REAL are between 3.4 E38 and -3.4 E38
Time (TIME), arrays (ARRAY), structures (STRUCT) and strings (STRING)	

13.4 User-defined data types depending on the implementation

- User-defined data types can be used within user function blocks and programs. They cannot be used in user functions.
- While debugging complete arrays and structures the variables have to be written into the watchlist. Writing these variables into the watchlist can only be done in the variable worksheet and not in the code body worksheet. The components of the data types have to be opened in the watchlist as it is described in the context-sensitive help.
- Multi-dimensional arrays must be declared. However it is possible to declare arrays of arrays as shown in the following example:

```

Type declaration:

TYPE
    graph      :   ARRAY[1..10] OF INT;
    my_array   :   ARRAY[1..3]  OF graph;
END_TYPE

Variable declaration:

VAR
    var1      :   my_array;
    var2      :   INT;
END_VAR

Code body declaration in ST:

var2 := var1[1][3];

```

Figure 152: Accessing elements of an array of array

- Only the data type 'INT' can be used as an array index. Expressions like 'i + 1' are not possible.
- In arrays and structures the data type BOOL corresponds to one byte.
- While using local variables for arrays and structures it has to be observed that corresponding address blocks of the I/O configuration stand in one single definition.
- To declare an array element within a structure the corresponding array has to be declared before the structure is declared as shown in the following example.

13.5 Data type STRING depending on the implementation

```
TYPE
    Array_Used_In_machine: ARRAY[1..23] OF INT;
    machine      :
    STRUCT
        x_pos      : INT;
        performance : Array_Used_In_machine;
    END_STRUCT;
END_TYPE
```

Figure 153: Type declaration of elements of an array in a structure

NOTE



Alias, subrange and enumerated data types are not available yet.

13.5 Data type STRING depending on the implementation

- If a variable of the data type STRING is used in PROPROGRAM wt II, it is always stored with a size of 85 characters. 80 bytes correspond to the maximum length of the string and 5 bytes are reserved for internal use.
- If a string constant in a code body worksheet is used, it needs one byte per character plus 5 bytes in the OmegaOS memory.
- String functions and function blocks cannot be used in networks. You can use only single functions and function blocks storing the result to variables.
- String functions cannot be nested while editing in ST.
- Strings cannot be used in function POU's.
- While overwriting strings in the dialog 'Online Debug' only 80 characters can be used.
- It is not possible for string constants to use Patch Worksheet.



For detailed information about STRING data types and STRING functions please refer to the OmegaOS documentation.

13.6 ST depending on the implementation

- In a FOR statement it is only possible to count upwards. Therefore the increment value after the keyword BY must always be positive.
- In a FOR statement the variable 'i' and the increment value can be changed within the body of the statement. No error message or warning is displayed if you enter the following example:

Example: FOR i=0 TO 10 BY5
 DO i:=10;

Be very careful using such a statement structure.

- No assignments can be used in a function call.
The following example is not possible: a:= SIN(b:=3);
- In an expression all operations are processed. Parts of an expression which have no impact on the result are also processed. In the following example c>d is also processed if a<b is FALSE:

Example: a:=a<b AND c>d;

13.7 VAR_IN_OUT depending on the implementation

- A VAR_IN_OUT output cannot be connected with a VAR_IN_OUT input of another FB
- Variable-status is not displayed in the graphic worksheets. In textual worksheets it is only displayed if powerflow is enabled. Variable-status is always displayed in the watchlist.

Please observe the following points recommended by the IEC 61131-3:

- VAR_IN_OUT can only be used for FBs
- A VAR_IN_OUT parameter is declared once in the FB and is visible (in FBD) as input-parameter and as output-parameter of the FBs
- Both, the input and the output of a VAR_IN_OUT parameter, have to be connected.
- The input and the output of the VAR_IN_OUT parameter have to be connected to a variable.
- The input and the output parameter have to be connected to the same variable.

13.8 Specials in PROPROG wt II

Please observe the following specials in PROPROG wt II:

- After having taken over a project from an older version of PROPROG wt, the project has to be compiled using "Recreate project".

- The watch window cannot handle variables of the groups VAR_GLOBAL_PG and VAR_GLOBAL_FB.
 - The watch window cannot handle directly represented variables.
 - It is not possible to delete page layouts from within the page layout editor.
 - It is not allowed to have blanks in the project path and/or name. The project name is limited to 24 characters.
 - The project tree's clipboard contents will be deleted if a node is deleted in the project tree.
 - It is not possible to select variables with more than 30 characters from inside the variable dialog.
-
- Multi-selection in the project tree is not possible.
 - Firmware libraries are now always located below the PLC directory.



List of Figures

An example of configuration elements	13
Diagram of a default task with two programs	15
Subtree 'Logical POUs'	15
Project tree with the instances within the resource 'R_IPC'	16
Worksheets of a function block in FBD	16
Icons of a SFC POU	17
Subtree 'Data Types'	18
The project with its subtrees	19
Subtree 'Library'	19
Program user interface with sample project 'example'	24
Sample tooltip (icon 'Save') and the corresponding description displayed in the status bar	27
Example of a detached toolbar window	28
The 'Shortcut Manager' for adding/modifying keyboard shortcuts	30
Dialog 'Assign Shortcut' before pressing the desired shortcut keys	31
Dialog 'Assign Shortcut' after pressing the desired shortcut keys	31
Default shortcuts in alphabetical order	33
Worksheet tabs in the workspace	34
Message window with its different page tabs	35
Cross reference window	37
Symbols in the cross reference window	38
Dialog 'Cross Reference Filter', used to filter the displayed entries in the cross reference list ..	38
Using the wildcard symbol in the dialog 'Cross Reference Filter'	39
Project tree window with sheet tabs	43
Edit Wizard for the different languages	46
Edit Wizard for variables and data type worksheets	46
Navigating in a graphical worksheet using the overview window	49
Dialog 'PROPROG wt II'	50
Dialog 'New Project' containing the available project templates and the Project Wizard	54
Project 'Untitled' with program 'Untitled' and its worksheets	55
Dialog 'Save As Template'	56
Dialog 'Properties' for changing the properties of existing POUs	58
Dialog 'Insert' for inserting a new POU	60
Project with announced library 'Contain'	64
Dialog 'Import / Export' when selecting the menu item 'Export...'	71
Dialog 'Project translation export'	71
Dialog 'Import / Export' when selecting the menu item 'Import...'	72
Dialog 'Project translation import'	73
Dialog 'Project Properties'	74
Dialog 'Source conversion'	75
Numeric literals	77
Character string literals	78
Duration literals	78
Elementary data types	79
Generic data types	80
Type declaration of an array data type	81
Programming example of an array data type	82
Type declaration of an array of array	83
Accessing elements of an array of array	83
Initializing elements of an array	84
Declaration of a structured data type	84
Declaration of an array of structure	85
Declaration of a structure of array	86



List of Figures

Assigning values to components of a structure	86
Declaration of a string data type	87
Edit Wizard in a data type worksheet	88
Data type worksheet 'Type' with pre-edited keywords, inserted using the Edit Wizard	88
Declaration of a symbolic, a located variable and a directly represented variable	90
Example for the declaration of a retentive variable.....	91
Declaration and initialization of a symbolic, a located variable and a directly represented variable....	92
Table of keywords for variable declaration blocks	94
Edit Wizard in a variable worksheet.....	95
Variable worksheet with pre-edited keyword 'VAR', inserted using the Edit Wizard.....	96
Instantiation of a function block	97
Example of an instance tree	98
Structure of an assignment statement in ST.....	101
Table of operators in ST	101
Table of statements in ST	103
Pre-edited CASE statement, inserted using the Edit Wizard.....	104
Dialog 'Variable'	106
Dialog 'Automatic Variables Declaration'	106
Function call in ST	107
Function block call in ST.....	108
Figure 6-9: Pre-edited MAX function and CTU function block, inserted using the Edit Wizard	109
Example of an instruction list.....	112
Table of operators, modifiers and operands in IL	113
Table of the modifiers in IL and their meaning.....	113
Operator 'ADD' and FB 'RS', both inserted using the Edit Wizard	114
Dialog 'Variable', called with a variable marked.....	116
Dialog 'Automatic Variables Declaration'	117
Dialog 'Variable'	118
Example of a jump	119
Example of calling a function	119
Example of calling a function block.....	120
Pre-edited MAX function and CTU function block, inserted using the Edit Wizard.....	121
Dialog 'Function/Function Block'	126
Dialog 'Variable'	128
Dialog 'Automatic Variables Declaration'	129
Function and function block before establishing the connection	131
The connection is set.....	132
The function is moved to a vacant position.	
The connection is routed automatically.	132
The output 'CV' of the FB 'CTU' is defined to be one end of the connection before the 'ADD' function is inserted	133
The connection is established automatically when the new 'ADD' function has been inserted	133
The function is moved to a vacant position.	
The connection is routed automatically.	133
Table of contacts and coils in LD.....	138
Example of a LD network.....	138
First LD network inserted.....	140
First LD network with inserted new contact 'C002'	140
LD network with a parallel branch.....	141
LD network with a parallel branch, inserted in LD branch edit mode.....	142
Dialog 'Contact/Coil'.....	143
Dialog 'Automatic Variables Declaration'	144



LD network with changed properties	145
LD network with function block 'CTU'	147
SFC network with one step and one transition	151
SFC network with four steps	153
Dialog 'Step'	154
Dialog 'Divergence'	155
SFC network with 2 alternative branches	156
Dialog 'Divergence'	157
Part of SFC network with an alternative and a simultaneous branch	158
Dialog 'Action'	160
Dialog 'Automatic Variables Declaration'	161
Action block with variable name	162
Dialog 'Transition'	163
Transition, specified as direct connection with a green connection point	163
Dialog 'Variable'	164
Dialog 'Automatic Variables Declaration'	164
Variable connected to a transition	165
Dialog 'Insert'	167
Dialog 'Insert'	170
Dialog 'Resource Settings...'	171
Insert a DEFAULT Task	172
Task settings for a DEFAULT task	173
Insert a new task of the type CYCLIC	173
Task settings for a CYCLIC Task	174
Select a system task	175
Dialog 'Insert'	176
Announced errors after making a project	178
Error list, displayed in the message window	178
Premises for the use of the 'Patch POU' operation	180
Control dialog	182
Dialog 'Download'	183
IL worksheet in online mode	185
Graphical worksheet in online mode	186
Meaning of the colors in online mode	186
Meaning of the powerflow signs in text worksheets in online mode	189
Meaning of the powerflow signs in graphic worksheets in online mode	190
Dialog 'Debug: resource_name'	191
Dialog 'Debug: resource_name'	193
Control dialog if a breakpoint is set	195
ST worksheet in online mode with set breakpoint	196
Watch window with several variables inserted	197
Watch window with opened array of structure	199
Logic Analyzer window after capturing two curves	201
Dialog 'Variables' of the Logic Analyzer window	202
Dialog 'Data Window' of the Logic Analyzer window	203
Dialog 'Trigger configuration' of the Logic Analyzer	204
Dialog 'Print Project'	208
Page 'Page layouts' in the dialog 'Options'	210
Page layout editor with a new page layout	212
Page layout with the new source area	213
Accessing elements of an array of array	219
Type declaration of elements of an array in a structure	220





Index

A	
Action	150, 159
detail	166
qualifier	150
Action block	150, 151
Action qualifier	159
ADD	112
Addition	101, 112
Address status	189
Alternative branch in SFC	150, 155, 158
AND	112
Announcing a library	63
ANY_BIT	80
ANY_INT	80
ANY_NUM	80
Archive	67
Arithmetic function	14
Array data types	81
array of structure	85
debug	198
multi-dimensional array	83
structures with arrays	86
Assignment statement	101
Associating programs to tasks	175
Automatic backup feature	69
Axis in the Logic Analyzer	205
B	
Bistable function block	14
Bitmaps in page layout	214
Bit-string function	14
BOOL	79
Boolean AND	101, 112
Boolean exclusive OR	101, 112
Boolean OR	101, 112
Borders	208
Branch	
alternative in SFC	150
parallel in LD	138
simultaneous in SFC	150
Breakpoint	192
BY	102
BYTE	79
C	
CAL	113, 120
Calling	
function block	113
functions and function blocks in IL	119
CASE	102
Character string function	14
Character string literal	78
Code body part	16
Coil	138
change properties	142
Colors in online mode	186
Colors of the status bar	40
Comment	70
FBD	124
IL	112
LD	138
SFC	150
ST	100
Comparison	101
Comparison function	14
Compiling	176
errors	178
Make	177
patch POU	179, 188
Complement	101
Configuration	12, 13
insert	169
Configuration elements	12, 13
Connecting graphical objects while inserting	132
Connecting objects in FBD	130
Connection line	138
Contact	138
change properties	142
inserting	139
Context-sensitive help	41
Correcting programming errors	188
Counter function block	14
Creating a new project	53
Cross Reference Window	
symbols (icons) in the column 'Variable'	37
Cross reference window	36
csv (file extension)	70
Cyclic tasks	12
D	
Data type	
array	81
derived	81
generic	80
initial value	92
string	87
structure	84
user defined	81
Data type declaration	
editing using the Edit Wizard	88
Data type STRING	220
Data type worksheet	44, 87
Data types	17, 219
elementary	79
user-defined	219
Data types declaration	18



Index

Debugging	188, 198	Duplicating function inputs	134
Declaration part	16	Duration literal	78
Default initial value	92	DWORD	79
Default page layout	210	E	
Derived data types	81	Edge detection function block	14
Description worksheet	70	Edit Wizard	115, 124
Detail	150, 166	calling functions/FBs in IL	119
Dialog		calling functions/FBs in ST	108
Action	160	editing data type declarations	88
Assign Shortcut	30	general description	45
Automatic FB Declaration	109, 121, 147	in ST	100
Automatic Variables Declaration	107, 117, 129, 144, 145, 161, 165	inserting instructions in IL	113
Contact/Coil	144	inserting statements in ST	103
Control dialog	182, 195	inserting variable declaration keywords	95
Cross Reference Filter	38	list box 'Group'	47
Debug 'resource_name'	191, 193	replacing functions/FBs	127
Divergence	155, 157	selection area	47
Download	183	Editor	
FB Instances	121, 125, 147	editing FBD worksheets	123
Function/Function Block	127	editing IL worksheet	111
Import / Export	71, 72	editing LD worksheet	137
Insert	60, 167, 170, 176	editing SFC worksheet	149
New Project	54, 55	editing ST worksheets	99
Page layout	210	worksheets in online mode	184
Print Project	209	Elementary data types	17, 79
Project Properties	74	ELSE	102
Project translation export	71	ELSIF	102
Project translation import	73	END_CASE	102
Properties	58	END_FOR	102
Resource Settings...	171	END_IF	102
Save as	66	END_REPEAT	102
Save changes?	50	END_VAR	94
Save project as	68	END_WHILE	102
Settings Environment Text	215	Environment item	214
Settings source area	213	EQ	113
Shortcut Keys	29	Equality	101
Source conversion	75	Error list (message window)	178
Step	154	Errors	
Transition	163	correct programming errors	188
Trigger configuration	203	displayed in the message window	35
Variable	106, 115, 116, 118, 128, 164	while compiling	178
Variables (of the Logic Analyzer)	202	Event task	12
DINT	79	EXIT	103
Direct connection	150	Exiting	
Directly represented variable	89	PROPROG wt II	50
Disk space, display in the status bar	40	worksheet	49
DIV	113	Exponentiation	101
Division	101, 113	Expression	100, 101
DO	102	F	
Documentation		FBD	124
context-sensitive help	8, 41	call editor	123
Downloading	181		
Drag & drop in graphic worksheets	130		



comment	124	Help on FB/FU	41
connect objects	130		
insert variable	127	I	
mix with SFC	166	I/Os, declaring	91
negating inputs and outputs	134	Icon	
network	124, 130	Add contact right	140
FBD worksheets	43	Add contact/coil above	141
Filtering the cross reference window	38	Add contact/coil below	141
Firmware library	19, 63	Add function	59
FOR	102	Add function block	59
Forcing	190	Add Object	61, 62, 63
Formal parameter	119, 128, 134	Add program	59
Function	41	code body worksheet name	61
call in IL	119	Connect objects	130
call in LD	146	Contact network	139
call in ST	107	data type folder	62
changing the properties	126	Data type worksheet	87
different types	14	Debug on/off	181, 184, 188
duplicating inputs	134	Duplicate FP	135
firmware	63	Edit Wizard	103, 108, 114, 120, 125, 146
general description	14	Environment items	214
negating inputs and outputs	134	FBD worksheet name	123
replacing using the Edit Wizard	127	IL worksheet name	111
type conversion	14	Insert LD branch	142
Function block	41	Insert SFC branch	159
call in IL	113, 120	Insert Simultaneous/Alternative Divergence	155, 156
call in LD	146	Insert step/transition	150, 151
call in ST	108	Insert/Delete Variable...	202
changing the properties	126	LD worksheet name	137
declare instance	93, 97	libraries folder	63
firmware	63	Line	214
general description	14	Logic Analyzer	200
instance	15	Make	178
negating inputs and outputs	134	Normally closed contact	143
replacing using the Edit Wizard	127	POU name	65
types	14	Program name	57
Functional Block Diagram		Resource name	169
see FBD		Save	66
		SFC worksheet name	149
G		Show Control Dialog	182, 195
GE	113	Source area	212
Generic data types	17, 80	ST worksheet name	99
Global variable	90	Start recording values	205
Graphic editor		Stop recording values	205
edit FBD	123	task name	175
edit LD	137	Toggle negation of FP	134
edit SFC	149	Trigger configuration	203
general description	43	Variable	105, 115, 118, 128, 163
GT	113	variable worksheet name	62
		Watch Window	197
H		Icons	
Hardware requirements	21	description in the status bar	27
Help function		tooltips, activating	27
see context-sensitive help			



Index

Icons in the Cross Reference Window	37		
IEC 61131	11		
configuration	12		
data types	17		
function	14		
function block	14		
instantiation	15		
program	14		
programming language	20		
resource	12		
task	12		
variable	17		
IEC 61131-3			
configuration elements	12		
IEC code	176		
IF	102		
IL			
call editor	111		
call function	119		
call function block	120		
comment	112		
insert variable	115		
instruction	112		
jump	119		
label	119		
modifier	112, 113		
operand	112		
operator	112		
IL worksheets	44		
Inequality	101		
Initial step	150, 151		
change	153		
Initial value	92		
Initializing variables	91		
Installation	22		
Instance	15, 97		
Instance name	15, 97, 176		
Instance tree	42, 98		
Instantiation	15, 97		
of function blocks	121, 125, 147		
Instruction			
IL	112		
inserting in IL using the Edit Wizard	113		
Instruction list			
see IL			
INT	79		
Internal memory	14, 15, 97		
Interrupt task	12		
Iteration statement	102		
J			
JMP	113, 119		
Jump	113, 119		
K			
Keyboard Shortcuts			
see shortcuts			
Keyboard, general usage	23		
Keywords in ST	100		
L			
Label	119		
Ladder Diagram			
see LD			
Language			
exporting a project translation file	70		
importing a project translation file	72		
switching project language	74		
Translating project language	70		
Language elements	13		
LD	138		
branch edit mode	141		
call editor	137		
call functions or function blocks	146		
change properties of contacts/coils	142		
coil	138		
comment	138		
connection line	138		
contact	138		
insert coil	139		
insert contact	139		
insert variable	145		
mix with SFC	166		
network	138		
parallel branch	138, 141		
power rail	138		
variable	138		
wired-OR	138, 141		
LD (Operator)	112		
LD worksheets	43		
LE	113		
Library	19		
announce	63		
announced	20		
print	207		
Viewing user library worksheets	63		
Line			
page layout	214		
Literal			
character string	78		
duration	78		
general description	77		
numeric	77		
Local variable	90		
Located variable	89		
Location	90		
prefix	90		
Logic Analyzer			



adjusting the window	205	IL	112
Dialog 'Trigger configuration'	203	ST	100, 101
Dialog 'Variable'	202	Operator	
general description	200	IL	112
Prerequisites for the usage	201	ST	100, 101
submenu in the submenu 'Online'	26	Optimized printing	209, 211
LT	113	OR	112
		Overview window	48
		Overwriting variables	190
M		P	
Main screen	34	Page borders	208
Make	177	Page layout	
Menu bar	25	creating a new	211
Message window	35	customize	210
errors while compiling	178	defining as default	210
warnings while compiling	178	preview	215
Mixing		Page layout editor	
FBD and LD	146	call	211
Modifier		define source area	212
IL	112, 113	general description	45
Modulo	101	insert a bitmap	214
Mouse, general usage	23	insert a line	214
MUL	113	insert a rectangle	214
Multiplication	101, 113	insert a system item	214
		insert a text	214
		insert an environment item	214
		Parallel branch in LD	138, 141
		Parenthesization	101
		Patch POU	177, 179, 188
		PLC code	176
		POU	
		change properties	57
		delete	65
		insert	59
		types	13
		Power rail	138
		Powerflow	189
		Preview	215
		Printing	207
		displaying page borders before printing	208
		print mode 'Optimized printing'	209
		print mode 'Standard'	209
		select items	209
		select range	209
		Program	
		associate to task	175
		general description	14
		instance	16, 98, 176
		internal memory	14
		Program module	13
		Program organization unit	
		see POU	
N			
NE	113		
Negated coil	138		
Negating inputs and outputs	134		
Negation	101		
Network			
FBD	124, 130		
insert SFC network	150		
LD	138		
SFC	150		
Normally closed contact	138		
Normally open contact	138, 139		
Numeric literal	77		
Numerical function	14		
O			
Online mode	184		
address status	189		
colors	186		
debug arrays and structures	198		
function worksheet	189		
Layout	186, 187		
powerflow	189		
powerflow in graphic worksheets	190		
powerflow in text worksheets	189		
switching on/off	188		
variable status	189		
watch	196		
Open Instance	184		
Operand			



Index

Programming errors	188	REPEAT	102
Programming language		RESET coil	138
FBD	124	Resource	12, 13
graphical	20	control dialog	182, 195
IL	112	insert	169
LD	138	settings	171
ST	100	RET	113
textual	20	RETAIN	91
Programming languages	218	Retentive variable	91
Project		initial value	92
compile	176	RETURN	103
create	53	Return	113
Creating using Project Wizard	53		
downloading	181	S	
exporting a translation file	70	S	112
importing a translation file	72	Save As Template...	56
Save as	66	Saving	65
Save as template	56	Scope	
switching project language	74	variable	90
template	55	Selection function	14
Translating project language	70	Selection statement	102
Untitled	55	Sequential function chart	
Zip automatically	69	see SFC	
Zip into an archive file	67, 68	SET coil	138
Project limits	217	Setup	22
Project tree	16, 18	SFC	150
configuration elements	13	action	150
insert configuration	169	action block	150
insert library	63	action qualifier	159
insert POU	59	alternative branche	155
insert resource	169	branch edit mode	158
insert task	169	call editor	149
insert worksheet	61	change network	153
subtree 'Data Types'	18, 42	comment	150
subtree 'Library'	20, 42	detail	150, 166
subtree 'Logical POUs'	15, 42	insert network	150, 151
subtree 'Physical Hardware'	42	insert variable	159, 162
Project tree editor	42	mix with FBD	166
Project Wizard	53	mix with LD	166
Properties	57	network	150
PROPROG wt II		simultaneous branch	156
exit	50	step	150
installation	22	transition	150
starting the program	22	SFC worksheets	43
		Shortcut Manager	29
Q		Shortcuts	29
Qualifier in SFC	150, 159	adding/modifying	29
		assigning new shortcuts	30
R		list of default shortcuts	32
R	112	Simultaneous branch in SFC	150, 156, 158
Rail	138	SINT	79
REAL	79	Size prefix	90
Rebuild Project	177	Software requirements	22
Rectangle in page layout	214	Sorting the cross reference list	39



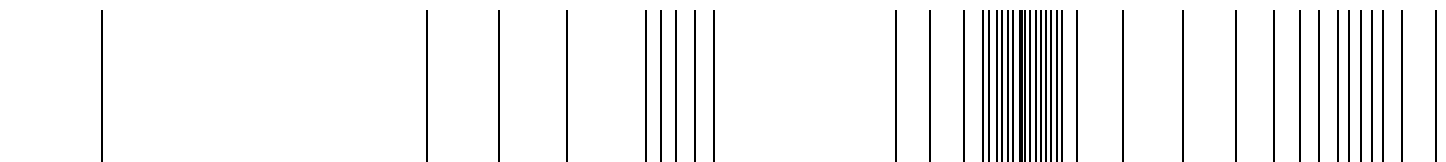
Source area	212	Subtree 'Data Types'	18, 42
Source conversion	75	Subtree 'Instances'	16
Specials in PROPROG wt II	221	Subtree 'Library'	42
ST	112	Subtree 'Logical POU's'	15, 42
assignment statement	101, 107	Subtree 'Physical Hardware'	13, 42
call editor	99	Symbolic variable	89
call function	107	Symbols in the Cross Reference Window	37
call function block	108	System item	214
CASE statement	102	System task	12
comment	100		
depending on the implementation	221	T	
edit statement	102	Task	12, 13
EXIT statement	103	associate a program	175
expression	100, 101	Template	55
FOR statement	102	Saving a project as template	56
general description	100	Text editor	
IF statement	102	declaring data types	87
insert variable	105	edit IL	111
operand	100, 101	edit ST	99
operator	100, 101	general description	44
REPEAT statement	102	TIME	79, 80
RETURN statement	103	Time scheduling	12
statement	100	Timer function block	14
statement keywords	102	TO	102
WHILE statement	102	Toolbar	27
ST worksheets	44	Tooltips	27
Standard function block	14	Trace	195
Standard print mode	209	Transition	150, 151
Starting the program	22	changing properties	162
Statement	100	detail	166
assignment statement	101, 107	direct connection	150
control statement	102	insert	151
inserting in ST using the Edit Wizard	103	variable	162
iteration statement	102	Translating project language	70
keywords statement	102	Translation file	
selection statement	102	exporting	70
Status bar	40	importing	72
Step	150	switching project language	74
change properties	153	Trigger configuration (Logic Analyzer)	203
initial	150		
insert	150, 151	U	
Step (debugging)	195	UDINT	79
STRING	80	UINT	79
String data types	87	Undocking/Docking the cross reference window	40
Structured data types	84	UNTIL	102
debug	198	User defined data types	17, 81
initialized	86	array	81
structures with arrays	86	string	87
Structured text		structures	84
see ST		User Interface	
SUB	113	menu bar	25
Submenus, general description	25	User interface	
Subtraction	101, 113		
Subtree 'Library'	19		



Index

cross reference window	36		
Edit Wizard	45		
general description	24		
main screen	34		
message window	35		
overview window	48		
status bar	40		
submenus	25		
toolbar	27		
workspace	34		
User interface, customize	24		
User library	19, 63		
USINT	79		
V			
VAR	93		
VAR_EXTERNAL	90, 93		
VAR_EXTERNAL_FB	94		
VAR_EXTERNAL_PG	93		
VAR_GLOBAL	90, 94		
VAR_GLOBAL_FB	94		
VAR_GLOBAL_PG	94		
VAR_IN_OUT	93		
depending on the implementation	221		
VAR_INPUT	90, 93		
VAR_OUTPUT	90, 93		
Variable	17		
adding to the Logic Analyzer	202		
declare	95		
declare using the variable editor	95		
declare while editing a code body	95		
delete from Logic Analyzer	202		
directly represented	89		
force	190		
global	90		
initialize	91		
insert in FBD	127		
insert in IL	115		
insert in LD	145		
insert in SFC	159, 162		
insert in ST	105		
inserting into watch window	196		
keywords	93		
local	90		
located	89		
overwrite	190		
retentive	91		
scope	90		
status in online mode	189		
symbolic	89		
watch window	196		
Variable declaration	16, 17		
function block instance	97		
inserting keywords using the Edit Wizard	95		
instantiation	93		
keyword	93		
location	90		
location prefix	90		
size prefix	90		
types	93		
Variable worksheets	44		
View mode for user library worksheets	63		
W			
Warnings			
displayed in the message window	35		
Watch window	196		
debug arrays and structures	198		
insert variable	196		
WHILE	102		
Wildcards	39		
Wildcards (filtering cross references)	39		
Wired-OR	138, 141		
Wizard			
Project Wizard	53		
WORD	79		
Worksheet			
delete	65		
description	56		
exit	49		
insert	61		
online mode	184		
preview	215		
print in online mode	207		
save	65		
variable	56		
Workspace	34		
X			
XOR	112		
Z			
Zipping a project	68		
zwt files	68		

be in motion



Baumüller Nürnberg GmbH Ostendstraße 80-90 90482 Nuremberg Tel: +49(0)911-5432-0 Fax: +49(0)911-5432-130 www.baumueller.de

All the information in these Operating Instructions is non-binding customer information; it is subject to ongoing further development and is updated on a continuous basis by our permanent change management system. Note that all the data/numbers/information that are quoted are current values at the time of printing. This information is not legally binding for dimensioning, calculation and costing. Before using the information listed in these Operating Instructions as the basis for your own calculations and/or applications, make sure that you have the latest most current information. This means that we accept no responsibility for the accuracy of the information.