

Rexroth IndraLogic XLC IndraMotion MLC 13VRS HMI Connection

R911336362
Edition 02

Application Manual



Title Rexroth IndraLogic XLC
IndraMotion MLC 13VRS
HMI Connection

Type of Documentation Application Description

Document Typecode DOK-XLCMLC-HMI*****V13-AP02-EN-P

Internal File Reference RS-dcfa675221d6f7cb0a6846a00062316e-2-en-US-4

Change history

Edition	Release Date	Notes
Edition 01	2012-04	First edition
Edition 02	2014-08	Revision; with BRWS driver documentation

Copyright © Bosch Rexroth AG 2014

This document, as well as the data, specifications and other information set forth in it, are the exclusive property of Bosch Rexroth AG. It may not be reproduced or given to third parties without its consent.

Liability The specified data is intended for product description purposes only and shall not be deemed to be a guaranteed characteristic unless expressly stipulated in the contract. All rights are reserved with respect to the content of this documentation and the availability of the product.

Editorial Department System Development Automation Motion Logic Control, AHo (FrWe, vha/MePe)

Table of Contents

	Page
1 About this Documentation.....	9
1.1 Validity of the Documentation.....	9
1.2 Using the Safety Instructions.....	9
1.2.1 Safety Instructions – Structure.....	9
1.2.2 Explaining Signal Words and Safety Alert Symbol.....	10
1.2.3 Symbols Used.....	10
1.3 Required and Supplementing Documentations.....	11
1.4 Names and Abbreviations.....	16
 2 Visualization Devices.....	 17
2.1 Overview.....	17
2.2 IndraControl VCP.....	17
2.2.1 Brief Description.....	17
2.2.2 Technical Data.....	19
2.2.3 Variants.....	19
2.2.4 Accessories.....	20
Connecting Cables.....	20
Battery kit.....	21
2.2.5 Documentation.....	21
2.3 IndraControl VCH.....	22
2.3.1 Brief Description.....	22
2.3.2 Field of Application.....	22
2.3.3 Technical Data.....	22
2.3.4 Variants.....	23
2.3.5 Accessories.....	23
2.3.6 Documentation.....	23
2.4 IndraControl VR21.....	23
2.4.1 Brief Description.....	23
2.4.2 Variants.....	25
2.4.3 Technical Data.....	26
VR 21 Single Touch.....	26
VR 21 Multi Touch.....	26
2.4.4 Housing and Mounting Dimensions.....	27
2.4.5 Interfaces.....	28
View of Interfaces.....	28
Interface overview.....	28
DC 24 V Voltage Supply.....	28
USB Interfaces X9, X10.....	29
Ethernet Interface X5.....	29
Slot for SD Memory Card.....	29
2.4.6 Ordering Information.....	30
Accessories and Spare Parts.....	30
Type Code VR21xx.01.....	30
2.4.7 Documentation.....	31

Table of Contents

	Page
2.5	IndraControl VEP..... 31
2.5.1	Brief Description..... 31
2.5.2	Field of Application..... 31
2.5.3	Technical Data..... 31
2.5.4	Wear Parts..... 32
2.5.5	Variants..... 32
2.5.6	Accessories..... 33
2.5.7	Documentation..... 33
2.6	IndraControl VPB and VDP..... 34
2.6.1	Brief Description..... 34
2.6.2	Field of Application..... 34
2.6.3	Technical Data..... 34
	VPB 40.3..... 34
	VDP 16.3/40.3/60.3..... 35
2.6.4	Wear Parts..... 35
2.6.5	Variants..... 35
2.6.6	Accessories..... 36
	Connecting Cables (CDI Interface)..... 36
	Accessories for VPB 40.3..... 37
2.6.7	Documentation..... 37
2.7	IndraControl VPP..... 37
2.7.1	Brief Description..... 37
2.7.2	Field of Application..... 38
2.7.3	Technical Data..... 38
2.7.4	Wear Parts..... 38
2.7.5	Variants..... 39
2.7.6	Accessories..... 39
2.7.7	Documentation..... 40
3	Visualizations..... 41
3.1	Overview..... 41
3.2	VI-Logic..... 41
3.2.1	Field of Application..... 41
3.2.2	Properties of the Integrated Visualization..... 41
3.2.3	Visualization Elements..... 42
3.2.4	Supported Visualization Devices..... 43
3.2.5	Protocol..... 43
3.2.6	Documentation..... 43
3.3	VI Composer..... 43
3.3.1	Field of Application..... 43
3.3.2	Configurable Small Operating Terminals..... 43
3.3.3	Communication Protocol BRC-Symbolic..... 44
	Brief Description..... 44
	Data Objects..... 44
	Individual Variables..... 44
	String Variables..... 44

Table of Contents

	Page
Tables.....	44
3.3.4 Connection Settings.....	44
Protocol Parameters for PROFIBUS-DP.....	44
Protocol parameter for Ethernet.....	45
3.3.5 Documentation.....	45
3.4 WinStudio.....	45
3.4.1 Overview.....	45
3.4.2 WinStudio Elements in the Project Explorer.....	46
3.4.3 General WinStudio Settings and Functions.....	47
3.4.4 WinStudio Screens.....	59
Creating and Editing WinStudio Screens	59
Screen Properties.....	59
Object Properties and Variable Selection.....	60
3.4.5 WinStudio Variables.....	60
3.4.6 Linked Symbols.....	61
WinStudio Library.....	61
User Symbols.....	62
3.4.7 WinStudio Tasks.....	63
Mathematic Sheets.....	63
Alarms.....	64
Recipes.....	64
Reports.....	65
Trends.....	66
Scheduler.....	66
External Databases.....	66
Scripts.....	66
Overview.....	66
Global Procedures.....	67
Start Script.....	67
Graphic Script.....	67
Screen Scripts.....	68
User Scripts.....	68
3.4.8 WinStudio Tools.....	69
Verifying Application.....	69
Starting Application.....	69
3.4.9 Database Spy.....	70
3.4.10 LogWin.....	70
3.4.11 Debugger.....	71
3.4.12 Cross Reference Search.....	71
3.4.13 Variable Import Wizard.....	72
3.4.14 Export/Import.....	72
3.4.15 WinStudio Licensing.....	73
Introduction.....	73
WinStudio License Overview.....	73
Activating a WinStudio License.....	74
Consequences of a Missing or a Lower Class License.....	74

Table of Contents

	Page
3.4.16 Multilingualism.....	75
3.4.17 WinStudio Applications.....	75
Overview.....	75
Creating WinStudio Application.....	75
User Management.....	75
Web Functionality.....	75
Transferring Application and Activation on Target Device.....	76
3.4.18 More Information.....	77
4 Communication Interfaces.....	79
4.1 Overview.....	79
4.2 BRWS Driver (Bosch-Rexroth WinStudio Driver).....	79
4.2.1 Properties.....	79
4.2.2 Areas of Application.....	80
4.2.3 Configuration.....	80
4.2.4 Driver Sheets.....	81
4.2.5 Protocol Structure.....	83
4.2.6 Conventions.....	83
4.2.7 Supporting PLC Variables and Parameters.....	84
PLC variables.....	84
XLC/MLC Parameters.....	85
4.2.8 Error Messages.....	87
4.2.9 Message Information.....	87
4.2.10 Documentation.....	88
4.3 OPC.lwSCP.....	88
4.3.1 Properties.....	88
4.3.2 SCP-OPC Configurator.....	88
General.....	88
Operation.....	89
Creating Device Configuration.....	89
Opening Device Configuration.....	89
Adding Devices.....	89
Displaying Device Properties.....	90
Changing Devices OPC Alias.....	92
Removing Devices.....	92
Saving Device Configuration.....	92
Saving Device Configuration As.....	92
Activating Device Configuration.....	93
Closing Applications.....	93
4.3.3 lwSCP and OPC.lwSCP Items.....	93
General Information.....	93
IndraLogic Items.....	95
Overview.....	95
sercos Items.....	95
Overview.....	95
4.3.4 Exclusion of Liability	96

Table of Contents

	Page
4.4	IndraLogic OPC.DA..... 96
4.4.1	Properties..... 96
4.4.2	Configuring IndraLogic OPC.DA Server..... 96
	General Information..... 96
	OPC.DA Server Configuration..... 97
	Determining Communication Address..... 97
	OPC.DA Server Configuration..... 97
	OPC.DA Server Configuration File "OPCServer.ini"..... 100
	Storage directory of the OPC.DA Server Configuration File..... 104
	General Information..... 104
	Windows XP..... 104
	Windows 7..... 105
	Windows CE..... 105
	OPC.DA Server - Item Syntax..... 107
4.4.3	Exclusion of Liability..... 107
4.5	Configuring OPC Client Connection with WinStudio 107
4.6	OPC UA..... 110
4.6.1	Overview..... 110
4.6.2	Introducing OPC Unified Architecture..... 111
	General Information..... 111
	Specification Overview..... 111
	OPC UA Information Model..... 112
	Service-Oriented Architecture..... 112
4.6.3	Bosch Rexroth OPC UA Server..... 116
	General Information..... 116
	Bosch Rexroth OPC UA Server Properties..... 116
	Configuration..... 117
	Configuration by the Client..... 118
	PLC Namespace..... 119
4.6.4	OPC UA Communication Example..... 121
	Introduction..... 121
	Prerequisites..... 121
	UaExpert..... 121
	Establishing a Connection..... 123
	Browsing..... 124
	Subscribe and Write..... 125
4.7	Configuring OPC UA Client Connection Using WinStudio..... 128
4.8	Modbus TCP..... 130
4.8.1	Introduction and Overview..... 130
5	Application Examples..... 133
5.1	Overview..... 133
5.2	Symbol Configuration..... 133
5.2.1	General Information..... 133
5.2.2	Symbol Configuration with Wizard..... 134
5.2.3	Symbol Configuration via Pragma..... 135

Table of Contents

	Page
5.3	VI-Composer..... 136
5.4	WinStudio with IndraLogic XLC..... 136
5.4.1	General Information..... 136
5.4.2	Technical Requirements..... 137
5.4.3	Creating the IndraWorks Project..... 137
5.4.4	Integrating the WinStudio Visualization in IndraWorks..... 143
5.4.5	Creating the WinStudio Visualization..... 150
5.4.6	Starting the Interface..... 156
5.5	IndraLogic Visualization..... 159
5.6	HMI of Third Party Manufacturers..... 159
5.6.1	OPC.DA Server..... 159
	Tools to Configure the OPC.DA Server on a PC..... 159
	Tools to Transfer the OPC.DA Server Configuration to a Visualization Device VEPxx.4 with Win- dowsCE..... 160
5.6.2	ModbusTCP..... 160
	General..... 160
	Configuration with Siemens WinCC-flexible 2007..... 162
	Configuration with Pro-face GP-Pro EX 2.2..... 166
6	Client Development..... 173
6.1	OPC Client Visual Basic .NET..... 173
6.1.1	General Information..... 173
6.1.2	Visual Basic Project "OPC Client"..... 173
	New Project..... 173
	Adding References..... 174
	Adding Windows Form and Control Element..... 175
	Implementing OPC Client..... 177
	General Information..... 177
	Declaring Variables..... 178
	Listing OPC server..... 179
	Creating and Canceling OPC Server Instances..... 179
	Creating and Canceling OPC Subscription..... 181
	Listing Control Items..... 183
	Adding and Deleting OPC Items..... 185
	Reading Control Data..... 187
	Writing Control Data..... 190
	Cyclic Reading of Control Data..... 192
	Error Handling..... 194
6.2	OPC Client Visual C# .NET..... 194
6.2.1	General Information..... 194
6.2.2	Visual C# Project "OPC Client "..... 195
	New Project..... 195
	Adding References..... 195
	Adding Windows Form and Control Elements..... 196
	Implementing OPC Client..... 198
	General Information..... 198

Table of Contents

	Page
Declaring Variables.....	199
Listing OPC Server.....	200
Creating and Deleting OPC Server Instance.....	200
Creating and Canceling OPC Server Subscription.....	203
Listing Control Items.....	205
Adding and Removing OPC Items.....	206
Reading Control Data.....	209
Writing Control Data.....	211
Cyclic Reading of Control Data.....	213
Error Handling.....	215
7 Service and Support.....	217
Index.....	219

1 About this Documentation

1.1 Validity of the Documentation

Target group This documentation is intended for users who connect the supported visualization systems of IndraLogic XLC, IndraMotion MLC, IndraMotion MTX or IndraMotion MLD (IL 2G).

This documentation supports the user during the following phases:

- Selection
- Engineering
- Commissioning

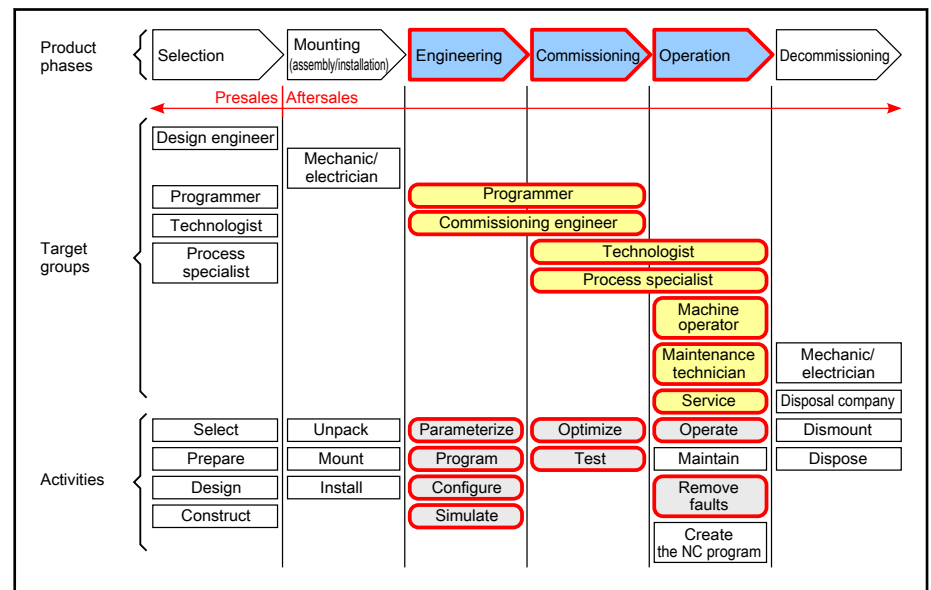


Fig. 1-1: Application phases

Required qualification: Individual who is able to assess the tasks assigned and identify possible safety risks owing to qualification in the subject, knowledge and experience. The individual should also be familiar with the standards and regulations.

1.2 Using the Safety Instructions

1.2.1 Safety Instructions – Structure

The safety instructions are structured as follows:

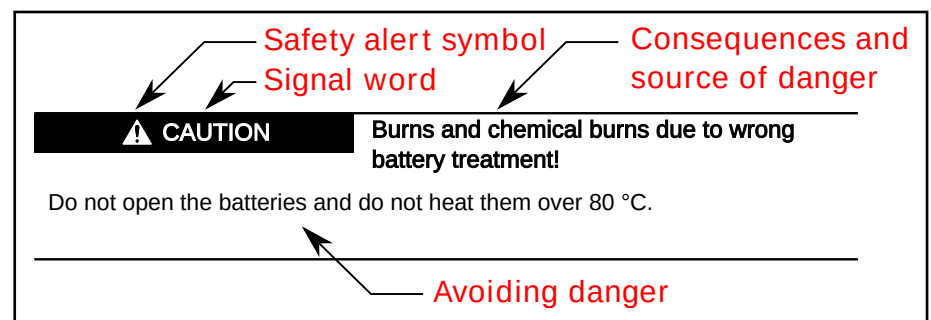


Fig. 1-2: Safety instructions – Structure

About this Documentation

1.2.2 Explaining Signal Words and Safety Alert Symbol

The safety instructions in this documentation contain specific signal words (danger, warning, caution, notice) and, if necessary, a safety alert symbol (according to ANSI Z535.6-2006).

The signal word is meant to draw the reader's attention to the safety instruction and signifies the degree of danger.

The safety alert symbol (a triangle with an exclamation point), which precedes the signal words danger, warning and caution is used to alert the reader to personal injury hazards.

DANGER

In case of non-compliance with this safety instruction, death or serious injury **will** occur.

WARNING

In case of non-compliance with this safety instruction, death or serious injury **can** occur.

CAUTION

In case of non-compliance with this safety instruction, minor or moderate injury could occur.

NOTICE

In case of non-compliance with this safety instruction, property damage could occur.

1.2.3 Symbols Used

Notes are displayed as follows:



This is a note.

Tips are displayed as follows:



This is a tip.

1.3 Required and Supplementing Documentations

Documentation titles with type codes and parts numbers

IndraWorks		MLC	XLC
/36/	Rexroth IndraWorks 13VRS Software Installation DOK-IWORKS-SOFTINS*V13-CORS-EN-P, R911336880 This documentation describes the IndraWorks installation.	X	X
/5/	Rexroth IndraWorks 13VRS Engineering DOK-IWORKS-ENGINEE*V13-APRS-EN-P, R911336870 This documentation describes the application of IndraWorks in which the Rexroth Engineering tools are integrated. It includes instructions on how to work with IndraWorks and how to operate the oscilloscope function.	X	X
/20/	Rexroth IndraMotion MLC 13VRS Functional Description DOK-MLC***-FUNC****V13-APRS-EN-P, R911336295 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraMotion MLC.	X	
/20/	Rexroth IndraLogic XLC 13VRS Functional Description DOK-XLC***-FUNC****V13-APRS-EN-P, R911336352 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraLogic XLC.		X
/7/	Rexroth IndraWorks 13VRS CamBuilder DOK-IWORKS-CAMBUIL*V13-APRS-EN-P, R911336291 This documentation describes the basic principles and operation of the CamBuilder, the cam editing tool.	X	X
/37/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Automation Interface DOK-XLCMLC-AUT*INT*V13-APRS-EN-P, R911336356 This documentation describes the script-based access to IndraWorks project data via the interface of the Automation Interface.	X	X
/38/	Rexroth IndraWorks 12VRS FDT Container DOK-IWORKS-FDT*CON*V12-APRS-EN-P, R911334398 This documentation describes the IndraWorks FDT Container functionality. It includes the activation of the functionality in the project and working with DTMs.	X	X
/29/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Project Conversion DOK-XLCMLC-PROCONV*V13-APRS-EN-P, R911336366 This documentation describes the project conversion of IndraLogic 04VRS and IndraMotion MLC04VRS on IndraWorks version 12 with IndraLogic 2G. Changes with regard to Motion and PLC are described in detail.	X	X
/28/	Rexroth IndraMotion MLC 13VRS Commissioning DOK-MLC***-STARTUP*V13-CORS-EN-P, R911336308 This documentation describes the steps to commission and service the IndraMotion MLC system. It includes checklists for frequent tasks and a detailed description of the steps.	X	

Tab. 1-1: XCL/MLC documentation overview

About this Documentation

Motion		MLC	XLC
/23/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS PLCopen Libraries DOK-XLCMLC-FUNLIB**V13-LIRS-EN-P, R911336360 This documentation describes the function blocks, functions and data types of the RIL_Common-Types, ML_Base and ML_PLCOpen libraries for the IndraLogic XLC/IndraMotion MLC. It also includes the error reactions of function blocks.	X	X
/27/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Generic Application Template DOK-XLCMLC-TF*GAT**V13-APRS-EN-P, R911336370 This documentation provides a structured template to the IndraLogic PLC programmer. This template can be used to add and edit the PLC programming code. It includes the template, the template wizard and example applications.	X	X
/31/	Rexroth IndraMotion MLC 13VRS RCL Programming Instruction DOK-MLC***-RCL*PRO*V13-APRS-EN-P, R911336297 This documentation provides information on the RobotControl. The programming language RCL (RobotControl Language) is focused. The program structure, variables, functions, motion statements and the required system parameters are described.	X	
/32/	Rexroth IndraMotion MLC 13VRS Robot Control V2 DOK-MLC***-ROCO***V13-RERS-EN-P, R911341035 The documentation provides information about the Robot Control V2. The focus of this documentation is on PLCopen programming. The program structure, variables, functions and motion commands as well as the required system parameters are described. Furthermore, the documentation contains the description of the ML_Robot library.	X	
/21/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Parameters DOK-XLCMLC-PARAM***V13-RERS-EN-P, R911336364 This documentation describes the parameters of the XLC/MLC systems as well as the interaction between parameterization and programming. It includes the axis parameters, control parameters, kinematic parameters, touch probe parameters and programmable limit switch parameters.	X	X
/10/	Rexroth IndraDrive Firmware for Drive Controllers MPH-, MPB-, MPD-, MPC-07 DOK-INDRV*-MP*-07VRS**-FKRS-EN-P, R911328670 This documentation describes all functional properties of the IndraDrive firmware in the variants MPH-07, MPB-07, MPD-07 and MPC-07.		
/11/	Rexroth IndraDrive MPx-17 Functions DOK-INDRV*-MP*-17VRS**-APRS-EN-P, R911331236 This documentation describes all functional properties of the IndraDrive firmware in the variants MPB-17, MPM-17, MPC-17 and MPE-17.		

Tab. 1-2: XCL/MLC documentation overview

About this Documentation

Field Buses		MLC	XLC
/4/	Rexroth IndraWorks 13VRS Field Buses DOK-IWORKS-FB*****V13-APRS-EN-P, R911336872 This documentation describes the supported field buses and their diagnostic function blocks.	X	X
/4a/	Rexroth IndraWorks 13VRS Field Buses Libraries DOK-IWORKS-FB*LIB**V13-LIRS-EN-P, R911337857 This manual describes the field bus libraries: RIL_ProfibusDP_02, RIL_ProfibusDP Slave, RIL_Profine-tIO, RIL_ProfinetIODevice, RIL_EtherNetIPAdapter, RIL_MappingList, RIL_SERCOSIII, RIL_Inline including their diagnostics and error reactions of the function blocks.	X	X
/63/	SERCOS System Manual for I/O Devices DOK-CONTRL-ILS3*****-APRS-EN-P, R911333512 This documentation describes the configuration, parameterization, commissioning and diagnostics of I/O devices with a SERCOS interface.	X	X

Tab. 1-3: XCL/MLC documentation overview

HMI		MLC	XLC
/8/	Rexroth IndraWorks 13VRS HMI DOK-IWORKS-HMI*****V13-APRS-EN-P, R911336874 This documentation describes the functions, configuration and operation of the user interfaces IndraWorks HMI Engineering and IndraWorks HMI Operation.	X	X
/6/	Rexroth IndraWorks 13VRS WinStudio DOK-IWORKS-WINSTUD*V13-APRS-EN-P, R911336882 This documentation describes the installation of the software, working with WinStudio and the creation and operation of applications.	X	X
/50/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS HMI Connection DOK-XLCMLC-HMI*****V13-APRS-EN-P, R911336362 This documentation describes the visualization systems supported by the IndraLogic XLC and IndraMotion MLC and their connection.	X	X

Tab. 1-4: XCL/MLC documentation overview

PLC		MLC	XLC
/3/	Rexroth IndraWorks 13VRS IndraLogic 2G Programming Instruction DOK-IWORKS-IL2GPRO*V13-APRS-EN-P, R911336876 This documentation describes the PLC programming tool IndraLogic 2G and its use. It includes the basic use, first steps, visualization, menu items and editors.	X	X
/33/	Rexroth IndraWorks 13VRS Basic Libraries IndraLogic 2G DOK-IL*2G*-BASLIB**V13-LIRS-EN-P, R911336285 This documentation describes the system-comprehensive PLC libraries.	X	X

Tab. 1-5: XCL/MLC documentation overview

About this Documentation

Technology		MLC	XLC
/30/	Rexroth IndraMotion MLC 13VRS Technology Libraries DOK-MLC***-TF*LIB**V13-LIRS-EN-P, R911336324 This documentation describes the function blocks, functions and data types of the "ML_TechInterface.library", "ML_TechMotion.library", "RMB_TechCam.library" and "ML_TechBase.library". It also includes libraries for the winder functionality, register controller functionality and CrossCutter functionality.	X	
/60/	Rexroth IndraMotion MLC 13VRS RegisterControl (Library) DOK-MLC***-REGI*CO*V13-LIRS-EN-P, R911336306 This documentation describes the inputs and outputs of the individual function blocks and provides notes on their use.	X	
/62/	Rexroth IndraMotion MLC 13VRS RegisterControl (Application Manual) DOK-MLC***-REGI*CO*V13-APRS-EN-P, R911336304 This documentation describes the application of the integrated register control for a rotogravure printing machine. The components of the mark stream sensor, the HMI application and the error recovery options are described. This instruction provides information on how to operate the register control, react to errors and query diagnostics. This documentation is written for machine setters and machine operators.	X	
/49/	Rexroth IndraMotion MLC 13VRS Winder Function Application DOK-MLC***-TF*WIND*V13-APRS-EN-P, R911336326 This application-related system documentation describes the application of the winder technology functions.	X	

Tab. 1-6: XCL/MLC documentation overview

Hardware		MLC	XLC
/1/	Rexroth IndraControl L45/L65/L85 Control DOK-CONTRL-ICL45L65L85-PRRS-EN-P, R911332116 This documentation describes the IndraControl L45/L65/L85 controls.	X	X
/2/	Rexroth IndraControl L25 DOK-CONTRL-IC*L25*****-PRRS-EN-P, R911328474 This documentation describes the IndraControl L25 controls.	X	X
/24/	Rexroth IndraControl Lxx 13VRS Function Modules DOK-CONTRL-FM*LXX**V13-APRS-EN-P, R911336283 This documentation describes all function modules of the Lxx controls including engineering and diagnostics.	X	X
/12/	Rexroth IndraDrive Drive Controllers MPx-02 to MPx-07 DOK-INDRV*-GEN-**VRS**-PARS-EN-P, R911297317 This documentation describes all parameters implemented in the firmware for drive controllers of the IndraDrive family. It supports the parameterization of the drive controllers.		

About this Documentation

Hardware		MLC	XLC
/13/	Rexroth IndraDrive MPx-02 to MPx-07 and HMV DOK-INDRV*-GEN-**VRS**-WARS-EN-P, R911297319 This documentation describes all diagnostics implemented in the following firmwares: • Drive controller firmwares from MPx-02 to MPx-08 and • firmwares of the supply devices of the type "HMV". It supports the operating crew as well as the programmer at troubleshooting.		
/35/	Rexroth IndraDrive Drive Controller Control Sections CSB01, CSH01, CDB01 DOK-INDRV*-CSH*****-PRRS-EN-P, R911295012 This documentation is a project planning manual to select and use the control sections CSB01, CDB01 and CSH01 for drive controllers of the product families Rexroth IndraDrive M and Rexroth IndraDrive C.		

Tab. 1-7: XCL/MLC documentation overview

Diagnostics and service		MLC	XLC
/26/	Rexroth IndraWorks 13VRS IndraMotion Service Tool DOK-IWORKS-IMST****V13-RERS-EN-P, R911337707 This documentation describes the IndraMotion Service Tool (IMST). It is a web-based diagnostic tool to access the control systems IndraMotion MLC L25, L45 or 65 as well as the MLP via an Ethernet high-speed connection. The IMST allows OEMs, end users and service engineers to access and remotely diagnose a system. A PC with Internet Explorer version 6 or 7 is required.	X	X
/22/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnostics DOK-XLCMLC-DIAG****V13-RERS-EN-P, R911336358 This documentation includes all control parameters implemented in the control systems IndraLogic XLC and IndraMotion MLC.	X	X

Tab. 1-8: XCL/MLC documentation overview

System Overview		MLC	XLC
/48/	Rexroth IndraMotion for Printing 13VRS System Overview DOK-IM*PR*-SYSTEM**V13-PRRS-EN-P, R911336289 This documentation describes the product IndraMotion for Printing. It introduces the control systems, drive systems and I/O systems as well as the commissioning and programming.	X	
/48/	Rexroth IndraMotion for Packaging 13VRS System Overview DOK-IM*PA*-SYSTEM**V13-PRRS-EN-P, R911336287 This documentation describes the product IndraMotion for Packaging. It introduces the control systems, drive systems and I/O systems as well as the commissioning and programming.	X	
/9/	Rexroth IndraMotion MLC 13VRS System Overview DOK-MLC***-SYSTEM**V13-PRRS-EN-P, R911336310 This documentation provides an overview on the hardware/software components of the automation system IndraMotion MLC in the mentioned version. It helps assembling a system.	X	
/9/	Rexroth IndraLogic XLC 13VRS System Overview DOK-XLC***-SYSTEM**V13-PRRS-EN-P, R911336354 This documentation provides an overview on hardware/software components of the automation system IndraLogic XLC in the respective version. It helps assembling a system.		X

Tab. 1-9: XCL/MLC documentation overview

About this Documentation

First Steps		MLC	XLC
/25/	Rexroth IndraMotion MLC 13VRS, First Steps DOK-MLC***-F*STEP**V13-CORS-EN-P, R911336293 This documentation describes the first steps of the IndraMotion MLC and the RobotControl. It includes the hardware and software prerequisites as well as the creation of a project.	X	
/25/	Rexroth IndraLogic XLC 13VRS, First Steps DOK-XLC***-F*STEP**V13-CORS-EN-P, R911336350 This documentation describes the first steps of the IndraLogic XLC. It includes the hardware and software prerequisites as well as the creation of a project.		X

Tab. 1-10: XCL/MLC documentation overview

1.4 Names and Abbreviations

Term	Explanation
CANopen	Field bus
DeviceNet	Field bus
Ethernet	Communication interface
IWE	IndraWorks Engineering
IWO	IndraWorks Operation
OWG	Optical waveguide
NC	Numerical Control
ModbusTCP Protocol	Communication protocol based on a master/slave or client/server architecture (IEC 61158)
OEM	Original Equipment Manufacturer
PROFIBUS-DP	Field bus
sercos	Sercos (SErial Realtime COmmunication System) interface is a world-wide standardized interface for the communication between controls and drives

Tab. 1-11: Names and abbreviations used

2 Visualization Devices

2.1 Overview

Rexroth IndraControl V is the comprehensive HMI device portfolio for the individual control, operation and monitoring in all industrial sectors. With scalable hardware and software, IndraControl V can be precisely adapted to your machine-specific requirements.

IndraControl V integrates all necessary functions for efficient automation starting with a convenient operation and clear and transparent visualization via integrated controls up to diagnostics. Together with Rexroth system solutions, get a complete automation solution for:

- Transfer machinery
- Printing and processing machines
- Packaging and food machines
- Metal forming machines
- Wood working machines
- Textile machines
- Handling and mounting systems

The following IndraControl V variants are available:

- Controller-based
 - [IndraControl VCP, page 17](#)
 - [IndraControl VCH, page 22](#)
 - [IndraControl VR21, page 23](#)
- Embedded PC-based
 - [IndraControl VEP, page 31](#)
- PC-based
 - [IndraControl VPB and VDP, page 34](#)
 - [IndraControl VPP, page 37](#)

2.2 IndraControl VCP

2.2.1 Brief Description

The VCP small control panels are operating and visualization terminals to operate and monitor machines. The devices are suitable for a number of different applications due to their compact design. It is communicated with the superordinate control via PROFIBUS DP or Ethernet interface.



Fig. 2-1: Small control panel VCP 02

Visualization Devices



Fig. 2-2: Small control panel VCP 05



Fig. 2-3: Small control panel VCP 08

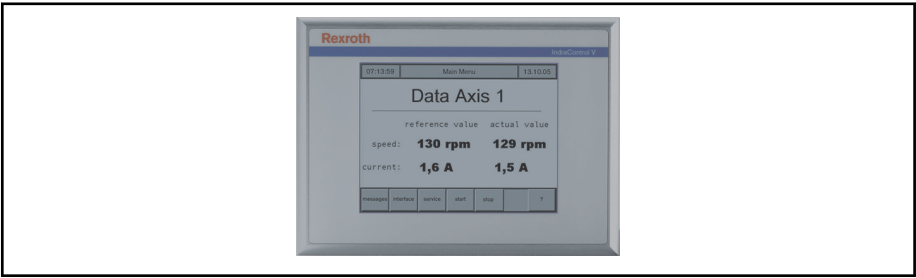


Fig. 2-4: Small control panel VCP 11

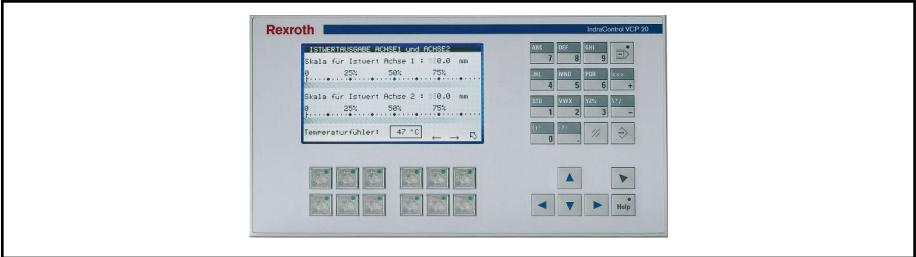


Fig. 2-5: Small control panel VCP 20



Fig. 2-6: Small control panel VCP 25

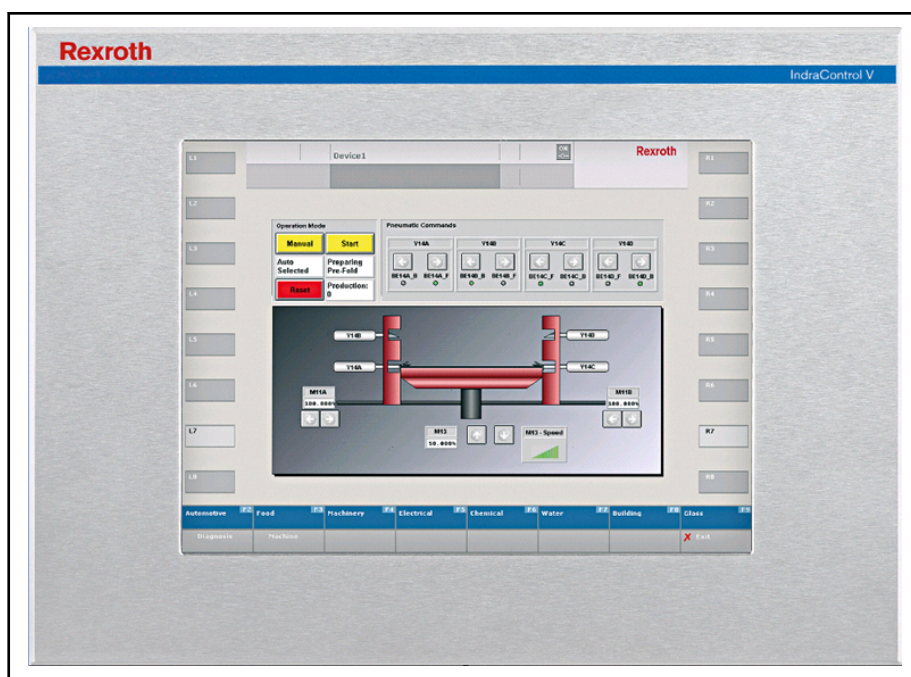


Fig. 2-7: Small control panel VCP 35

2.2.2 Technical Data

	VCP 02	VCP 05	VCP 08	VCP 11	VCP 20	VCP 25	VCP 35
Display	3" graphic display, 160 x 80 pixels		3.8" graphic display, 320 x 240 pixels	3.8" graphic display, 320 x 240 pixels	5.7" graphic display, 320 x 240 pixels	5.7" graphic display, 320 x 240 pixels	10.4" graphic display, 640 x 480 pixels
Front panel	100 x 148 mm	168 x 120 mm	159 x 209 mm	96 x 130 mm	160 x 300 mm	180 x 234	249 x 328 mm
Function/system keys	4 / 7	6 / 24	15/24	Touch screen	16 / 22	Touch screen	Touch screen
Processor	ARM 200 MHz						
Application memory	3 MB						
Interfaces	1x Ethernet (10/100 Base T) 2x USB host 2.0 Profibus DP interface RS232/RS485 (option)						
Power supply	24 V DC						

Tab. 2-1: Technical data for VCP small control terminal

2.2.3 Variants

The following variants of the VCP small control panels are available:

Visualization Devices

Type	Remarks
VCP02.2DRN-003-NN-NN-PW	Keys, communication via Ethernet
VCP02.2DRN-003-PB-NN-PW	Keys, communication via PROFIBUS DP
VCP02.2DRN-003-SR-NN-PW	Keys, communication via serial interfaces
VCP05.2DSN-003-NN-NN-PW	Keys, communication via Ethernet
VCP05.2DSN-003-PB-NN-PW	Keys, communication via PROFIBUS DP
VCP05.2DSN-003-SR-NN-PW	Keys, communication via serial interfaces
VCP08.2DTN-003-NN-NN-PW	Keys, communication via Ethernet
VCP08.2DTN-003-PB-NN-PW	Keys, communication via PROFIBUS DP
VCP08.2DTN-003-SR-NN-PW	Keys, communication via serial interfaces
VCP11.2DWN-003-NN-NN-PW	Touch screen, communication via Ethernet
VCP11.2DWN-003-PB-NN-PW	Touch screen, communication via PROFIBUS DP
VCP11.2DWN-003-SR-NN-PW	Keys, communication via serial interfaces
VCP11.2ECN-003-NN-NN-PW	TFT
VCP11.2ECN-003-PB-NN-PW	TFT with PROFIBUS DP slave
VCP20.2DUN-003-NN-NN-PW	Keys, communication via Ethernet
VCP20.2DUN-003-PB-NN-PW	Keys, communication via PROFIBUS DP
VCP20.2DUN-003-SR-NN-PW	Keys, communication via serial interfaces
VCP25.2DVN-003-NN-NN-PW	Touch screen, communication via Ethernet
VCP25.2DVN-003-PB-NN-PW	Touch screen, communication via PROFIBUS DP
VCP25.2DVN-003-SR-NN-PW	Keys, communication via serial interfaces
VCP35.2ECN-003-NN-NN-PW	Touch screen, communication via Ethernet
VCP35.2ECN-003-PB-NN-PW	Touch screen, communication via PROFIBUS DP
VCP35.2ECN-003-SR-NN-PW	Keys, communication via serial interfaces

Tab. 2-2: VCP device variants

2.2.4 Accessories

Connecting Cables

The VCPs communicate with the controls either using PROFIBUS DP or the Ethernet. The following cable is available for PROFIBUS DP:

Visualization Devices

Type	Notes
IKB0034/000,0	Connection between control and VCP; ready-made cables on one side, variable length

Tab. 2-3: PROFIBUS connecting cable

The following cables are available to communicate via Ethernet:

Type	Notes
RKB0008/002.5	Bus cable, ready-made Ethernet cable, ready-made 100-Base-T, CAT 6+, UL, drag chain-compatible, cable length 2.5 m
RKB0008/005,0	As above, cable length 5 m
RKB0008/010,0	As above, cable length 10 m
RKB0008/015,0	As above, cable length 15 m
RKB0008/020,0	As above, cable length 20 m
RKB0008/025,0	As above, cable length 25 m
RKB0008/030,0	As above, cable length 30 m
RKB0008/035,0	As above, cable length 35 m
RKB0008/040,0	As above, cable length 40 m
RKB0008/050,0	As above, cable length 50 m

Tab. 2-4: Ethernet connecting cable

Battery kit

A battery kit is available under the following name:

Type	Notes
VAS04.1-001-002-NN	Battery kit

Tab. 2-5: Battery kit for VCP devices

2.2.5 Documentation

The following documentation is available to provide comprehensive descriptions of the VCP small control panels:

- DOK-SUPPL*-VCP*02.2***-PRxx-EN-P
- DOK-SUPPL*-VCP*05.2***-PRxx-EN-P
- DOK-SUPPL*-VCP*08.2***-PRxx-EN-P
- DOK-SUPPL*-VCP*11.2***-PRxx-EN-P
- DOK-SUPPL*-VCP*20.2***-PRxx-EN-P
- DOK-SUPPL*-VCP*25.2***-PRxx-EN-P
- DOK-SUPPL*-VCP*35.2***-PRxx-EN-P

xx - Edition

2.3 IndraControl VCH

2.3.1 Brief Description

Operating, setting up, parameterizing and diagnosing from anywhere – IndraControl VCH 08 makes it possible using Ethernet TCP/IP. When connecting and unplugging, the "Stop" function is automatically bridged and guarantees trouble-free operation.

Low weight and optimum design allow to work without effort and provide excellent operating comfort.

The integrated 3-stage enabling button and 2-circuit design of the Stop button are looped through the VAC 30 switch-on box and thus ensure maximum safety.

Special features:

- Safety functions for operators and machinery
- Handles with integrated enabling and stop buttons
- Robust construction for industrial use
- Ergonomically optimized for a safe and comfortable handling
- Brilliant 3.8" graphic display and an all-purpose usable membrane keyboard for convenient operation and visualization



Fig. 2-8: IndraControl VCH 08

2.3.2 Field of Application

The VCH 08 is designed for handling applications.

2.3.3 Technical Data

	VCH 08
--	--------

Visualization Devices

Display	3.8" gray scale, 320 x 240 pixels
Front panel design	15 function keys 25 system keys 40 operating keys
Enabling button	Number: 2 2-circuit, 3-stage
Stop button	Number: 1 2-circuit
Handwheels	Number: 1 (option)
Override potentiometer	Number 1: (Option)
Processor	CPU PXA 270/416 MHZ (standard)
Memory	64 MB (application) 64 MB (flash)

Tab. 2-6: Technical data for VCH 08

2.3.4 Variants

The following VCH 08 variants are available:

Type	Remarks
VCH08.1EAB-064ET-A1D-064-CS-E2-PW	With handwheel (0.5 m Ethernet control cable)
VCH08.1EAB-064ET-A1D-064-FS-B2-PW	With E-Stop (0.5 m Ethernet control cable)
VCH08.1EAB-064ET-A1D-064-CS-E4-PW	8 m Ethernet control cable
VCH08.1EAB-064ET-A1D-064-DS-E4-PW	With handwheel (8 m Ethernet control cable)

Tab. 2-7: VCH 08 variants

2.3.5 Accessories

The following accessories are available for the VCH 08:

Type	Remarks
VAS01.1-002-NNN-NN	Wall bracket VAS 01
VAC30.2N-NN	VAC 30

Tab. 2-8: Accessories for the VCH 08

2.3.6 Documentation

The following documentation is available for the comprehensive description of the VCH 08 hand-held terminal:

- DOK-SUPPL*-VCH*08.1***-PRxx-EN-P

xx - Edition

2.4 IndraControl VR21

2.4.1 Brief Description

The VR21xx small control panels are operating and visualization terminals to operate and monitor machines. The devices are suitable for a number of different applications due to their compact design. With the XLC/MLC, the devi-

Visualization Devices

ces are available with a screen diagonal of 4.3", 7" and 9". Configuring operating screens using Rexroth WinStudio. Communication with a parent control takes place via Ethernet interface.

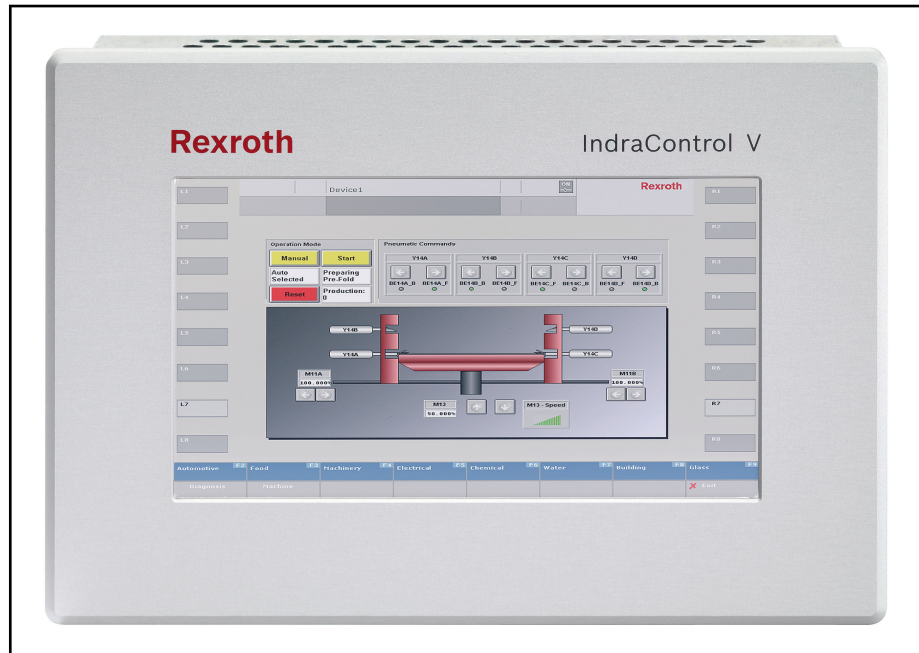


Fig. 2-9: Small control panel VR2104

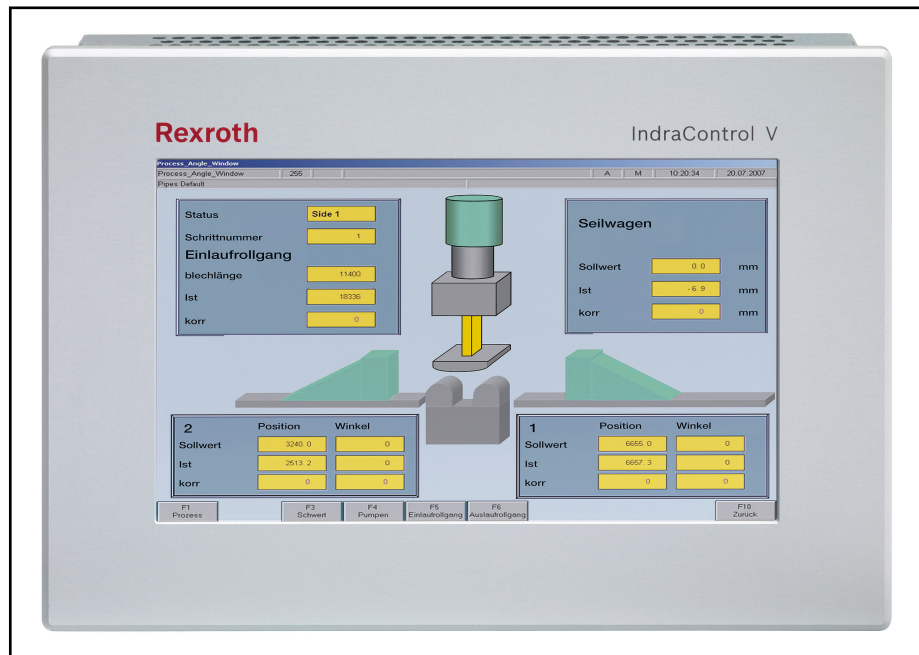


Fig. 2-10: Small control panel VR2107

Visualization Devices

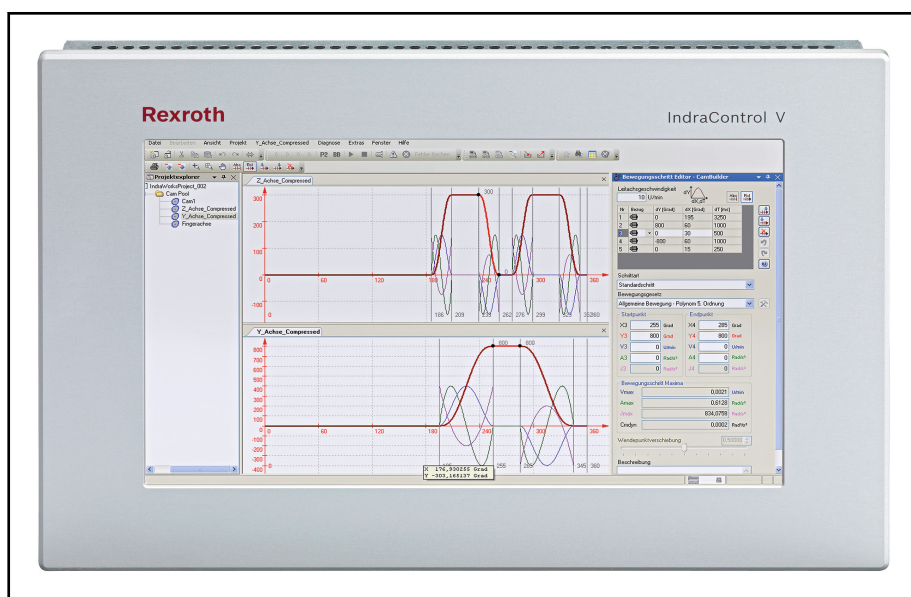


Fig. 2-11: Small control panel VR2109

2.4.2 Variants

IndraControl VR 21 devices are available in different variants. The devices differ in display size and in the touch technology (single touch and multi touch).

Type code	Display	Resolution	Touch technology
VR2104.01-00-01-N2-NNN-AA Parts number R911340500	4,3"	480 × 272 pixels	Resistive, single touch
VR2107.01-00-01-N2-NNN-AA Parts number R911340503	7"	800 × 480 pixels	Resistive, single touch
VR2107.01-00-01-N2-NNN-CA Parts number R911340505	7"	800 × 480 pixels	Capacitive, multi touch
VR2109.01-00-01-N2-NNN-AA Parts number R911340051	9"	800 × 480 pixels	Resistive, single touch
VR2109.01-00-01-N2-NNN-CA Parts number R911340506	9"	800 × 480 pixels	Capacitive, multi touch

Tab. 2-9: Variants of IndraControl VR 21

Visualization Devices

2.4.3 Technical Data

VR 21 Single Touch

	VR2104 single touch	VR2107 single touch	VR2109 single touch
Display	107.95 mm TFT (4.3") 480 × 272 pixels 65536 colors Brightness 450 cd/m ² Display size 53.8 × 95 (H × W)	177.8 mm TFT (7") 800 × 480 pixels 262144 colors Brightness 350 cd/m ² Display size 91.4 × 152.4 (H × W)	228.6 mm TFT (9") 800 × 480 pixels 16.77 millions of color Brightness 360 cd/m ² Display size 117 × 195 (H × W)
Touch technology	Analog resistive, 4-wire-technology		
Touch activation pressure	15 g (standard) with R8 HS60 silicone rubber		
Housing	Sheet steel, electroplated		
Front panel material	Aluminum, brushed, naturally anodized		
Degree of protection	Front panel IP 65 acc. to DIN EN 60 529, rear panel IP 20		
Central unit	ARM Cortex™-A8, 800 MHz with real-time clock		
Flash memory	256 MBytes		
LPDDR memory	512 MBytes		
Power supply	DC 24 V (use a 24 V industrial power supply acc. to DIN EN 60742, classification VDE 0551, for example VAP01.1H-W23-024-010-NN, parts number R911171065)		
Current consumption	0.2 A (typical for 24 V) 0.3 A (maximum)	0.3 A (typical for 24 V) 0.4 A (maximum)	0.6 A (typical for 24 V) 0.7 A (maximum)
Connected load	4.8 W	7.2 W	14.4 W
Fuse	Semiconductor fuse, self-resetting		
Reverse voltage protection	Integrated		
USB	Maximum of 500 mA for each USB socket. Maximum of a total current of 1 A at all USB sockets.		
Weight	Ca. 0.6 kg	Ca. 0.8 kg	Ca. 1.3 kg

Tab. 2-10: Technical data of the VR 21 single touch

VR 21 Multi Touch

	VR2107 multi touch	VR2109 multi touch
Display	177.8 mm TFT (7") 800 × 480 pixels 262144 colors Brightness 350 cd/m ² Display size 91.4 × 152.4 (H × W)	228.6 mm TFT (9") 800 × 480 pixels 16.77 millions of color Brightness 360 cd/m ² Display size 117 × 195 (H × W)
Touch technology	Projected capacitive, operable with fingers (touch)	
Touch activation pressure	15 g (standard) with R8 HS60 silicone rubber	
Housing	Sheet steel, electroplated	
Front panel material	Aluminum, brushed, naturally anodized	

	VR2107 multi touch	VR2109 multi touch
Surface of the front panel	Glass	
Degree of protection	Front panel IP 65 acc. to DIN EN 60 529 Rear panel IP 20	
Central unit	ARM Cortex™-A8, 800 MHz with real-time clock	
Flash memory	256 MBytes	
LPDDR memory	512 MBytes	
Power supply	DC 24 V (use a 24 V industrial power supply acc. to DIN EN 60742, classification VDE 0551, for example VAP01.1H-W23-024-010-NN, parts number R911171065)	
Current consumption	0.4 A (typical for 24 V) 0.5 A (maximum)	0.7 A (typical for 24 V) 0.8 A (maximum)
Connected load	9.6 W	16.8 W
Fuse	Semiconductor fuse, self-resetting	
Reverse voltage protection	Integrated	
USB	Maximum of 500 mA for each USB socket. Maximum of a total current of 1 A at all USB sockets.	
Weight	Ca. 0.8 kg	Ca. 1.3 kg

Tab. 2-11: Technical data of the VR 21 multi touch

2.4.4 Housing and Mounting Dimensions

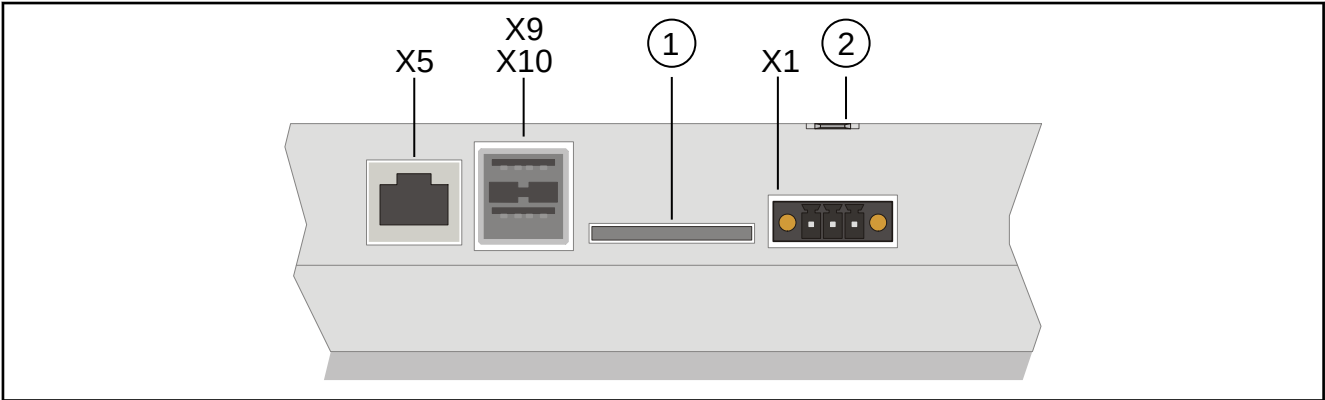
Variant	Display	Device height	Device width	Mounting depth	Mounting cut-out
VR2104	4,3 "	100 mm	140 mm	42 mm	H: 92 mm, B: 132 mm
VR2107	7 "	150 mm	211 mm	46.3 mm	H: 142 mm, W: 203 mm
VR2109	9 "	178 mm	263 mm	52.3 mm	H: 170 mm, W: 255 mm

Tab. 2-12: Housing and mounting dimensions of the VR 21xx

Visualization Devices

2.4.5 Interfaces

View of Interfaces



① SD memory card
② Functional earth
Fig. 2-12: Interfaces at the IndraControl VR 21

Interface overview

Housing label	Connection type	Connector type, installed	Mating connector or connection (external)
X1	DC+24 V power supply	3-pin, Phoenix MINI COMBICON	3-pin, FK-MCP 1,5/3-ST-3,5
X9, X10 USB host	2 USB interfaces	USB socket, 4-pin, type A	USB plug, 4-pin, type A
X5	Ethernet 10/100 Base-T	RJ45 socket	RJ45 male connector
 Functional earth	Functional earth	Tab connector	Blade receptable
SD memory card	Insert for SD and SDHC memory card	-	

Tab. 2-13: Interfaces

NOTICE

Malfunctions due to insufficient shielding!
Only use shielded cables and metallic, conductive plug connectors or coupling housings with a large shield connection.

DC 24 V Voltage Supply

Connect cables with a conductor cross section between 0.75 and 2.5 mm² to the connecting terminal.

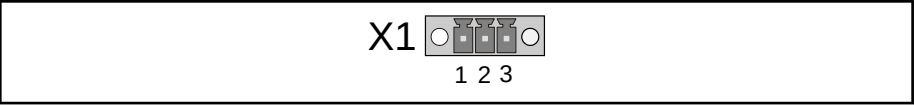


Fig. 2-13: Power supply connector X1

Visualization Devices

Pin	Name	Function
1		Low-noise earth, functional earth (FE)
2	0 V	Supply voltage 0 V (GND)
3	24 VDC	Supply voltage 24 VDC

Use a 24 V industrial power supply acc. to DIN EN 60742, classification VDE 0551, for example VAP01.1H-W23-024-010-NN, parts number R911171065)

USB Interfaces X9, X10

Operating USB ports at operating device

There are two USB host interfaces at the operating device.



When using input devices (e.g. keyboard, mouse) not suitable for the industrial use, operating safety might be limited. Input devices designed for home and office use are also included.



Only connect the USB devices complying with the USB2.0 specification.



For the specification of a suitable cable, refer to "Universal Serial Bus Specification Rev. 2.0".

The maximum cable length is 2.5 m.



Not all USB devices are detected.

The operating system does not support all USB devices. Devices - requiring a special USB driver not part of the system - cannot be operated at the USB interfaces.

Ethernet Interface X5

There is a 10/100Base-T Ethernet interface at the operating device.



Use the twisted pair cable of category 5 or 6 (CAT 5 or 6). The maximum cable length is 100 m.

Ethernet interface – Status and diagnostic displays

LED ACT/LNK (green)

- On: Connection to network
- Off: No connection
- Flashing: Communication running

LED SPD 10/100 (yellow)

- On: 100 MBit/s mode
- Off: 10 MBit/s mode or connection interrupted

Slot for SD Memory Card

Plug in the SD card on the lower side of the operating device.



When using hardware (e.g. keyboard, mouse, memory card) not suitable for the industrial use, operating safety might be limited. This includes also hardware designed for home and office.

Visualization Devices

Plugging in memory card

Insert the memory card with the upper side on top. Insert the memory card until engaged.

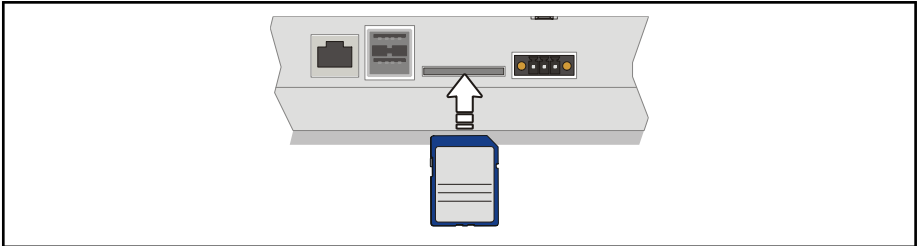


Fig. 2-14: Insert memory card

Remove memory card

Press the memory card until it is engaged in the operating device. Remove the disengaged memory card.

2.4.6 Ordering Information

Accessories and Spare Parts

For ordering information on accessories and spare parts refer to Rexroth IndraControl VR 21 Operating Panel, DOK-SUPPL*-VR21**-.01**-ITRS-EN-P, R911339476.

Type Code VR21xx.01

Short text column	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Example:	V	R	2	1	0	4	.	0	1	-	0	0	-	0	1	-	N	2	-	N	N	N	-	A	A	.	.	.
Unit type 1																												
HMI..... = V																												
Unit type 2																												
Box (preconfigured).. = R																												
Rating class																												
Rating class S..... = 2																												
Processor version																												
Processor version 1..... = 1																												
Display size (16:9)																												
4,3"..... = 04																												
7"..... = 07																												
9"..... = 09																												
Design																												
HW-Design..... = 01																												
Field bus																												
None Field bus..... = 00																												
Interface (Ethernet, USB...)																												
2 x USB Host + 1 x Ethernet 100 Mbit..... = 01																												
Remanent Memory																												
Non-existent..... = N																												
Mass storage (Harddisk, Flash, SSD)																												
Onboard Flash min.128 MB..... = 2																												
Hardware configuration																												
None extension slot..... = NNN																												
Design & HMI-Display character																												
Single-Touch in Rexroth-Design..... = AA																												
Multi-Touch in Rexroth-Design..... = CA																												

Fig. 2-15: Type code

2.4.7 Documentation

The following documentation is available to provide comprehensive descriptions of the VR21 small control panels:

- DOK-SUPPL*-VR21**.01**-IT02-EN-P

2.5 IndraControl VEP

2.5.1 Brief Description

These terminals allow to operate the machine easily and conveniently using a touch screen or a virtual keyboard. These devices, similar to PCs, use only embedded components for a robust design.

Due to the large number of interfaces and slots, the IndraControl VEP devices can be ideally adapted to comply with machine and system requirements.

Special features:

- Compact device design for the installation in control cabinets or on a support arm
- Non-hard drive hardware without rotating media
- Visualization, operation and monitoring with WinStudio visualization software

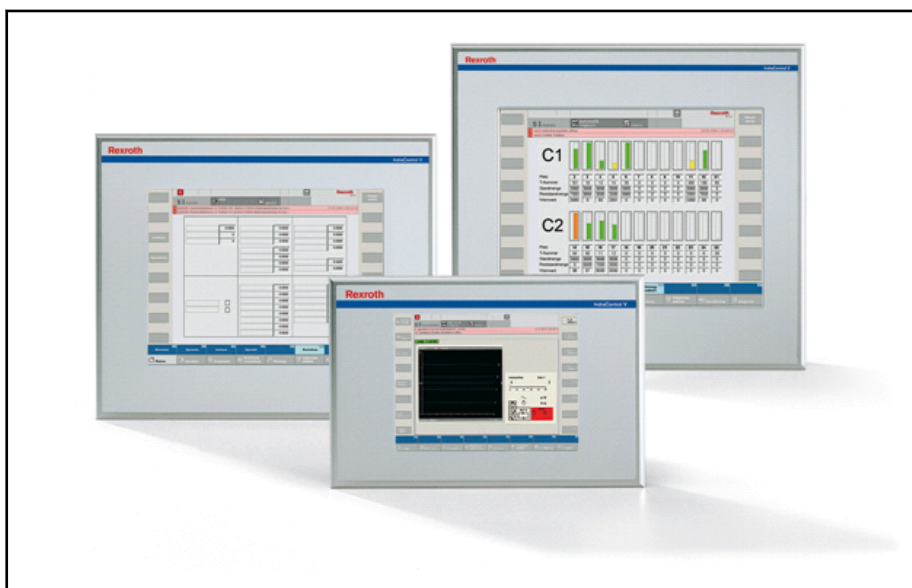


Fig. 2-16: IndraControl VEP variants

2.5.2 Field of Application

The standard industrial operating terminals VEP are primarily used for all tasks in the PC-based automation.

2.5.3 Technical Data

	VEP 40.5	VEP 50.5
Display	12" color display	15" color display
Keyboard	Via virtual keyboard	
Module slots	--	

Visualization Devices

	VEP 40.5	VEP 50.5
Interfaces	2 x USB 2.0 - Connection (type A) 2 x Ethernet ports (RJ 45, 10/100 Base-T)	
Field bus	--	
Processor	Atom N455 1.66GHz	
Main memory	2 GB	
Compact Flash	Equipped with CF socket / 8 GB CF card	
Supply voltage	24 V DC	
Operating system	Windows XPe / Windows 7e	

Tab. 2-14: Technical data VEP 40.5/50.5

	VEP 30.4	VEP 40.4	VEP 50.4
Display	8.4" color display	12" color display	15" color display
Keyboard	Touch control	Touch control or touch and key control	Touch control or touch and key control
Module slots	--		
Interfaces	2 x USB 2.0 - Connection (type A) 2 x (Intel Celeron) or 1 x (Intel Atom) Ethernet port (RJ 45, 10/100 Base-T)		
Processor	Intel Celeron 600Mhz / Intel Atom 1.1 GHz		
Main memory	512MB / 1 GB		
Compact Flash	2 CF socket / Equipped with 1GB CF-card or 4 GB CF-card		
Supply voltage	24 V DC		
Operating system	Windows CE 6.0 / Windows XPe		

Tab. 2-15: Technical data for VEP 30.4/40.4/50.4

2.5.4 Wear Parts

Parts with a limited service life are installed in the standard industrial terminals VEP and are therefore not subject to warranty.

This applies to the following components:

- CF card
- CMOS battery
- Background lighting

The service life of the individual components is described in the device documentation (see [Chapter 2.5.7 Documentation, page 33](#)).

2.5.5 Variants

The following variants of the standard industrial operator terminals VEP xx.5 are available:

Type	Remarks
VEP40.5DBN-2G02E-A3D-NNN-NN-FW	Touch front
VEP40.5EIN-2G02E-A3D-NNN-NN-FW	Combined front
VEP50.5DFN-2G02E-A3D-NNN-NN-FW	Combined front

Tab. 2-16: VEP xx.5 variants

The following variants of the standard industrial operating terminals VEP xx.4 are available:

Type	Remarks
VEP30.4EFN-512NN-A2D-NNN-NN-FW	Touch front
VEP40.4DBN-512NN-A2D-NNN-NN-FW	Touch front
VEP40.4BKN-512NN-A2D-NNN-NN-FW	Combined front
VEP50.4DEN-512NN-A2D-NNN-NN-FW	Touch front
VEP50.4BIN-512NN-A2D-NNN-NN-FW	Combined front

Tab. 2-17: VEPxx.4 variants

2.5.6 Accessories

The following accessories are available for VEP devices:

Type	Remarks
Compact flash module 1 GByte	CFM01.1-01G0-N-LBA-NN-NW
Compact flash module 4 GBytes	CFM01.1-04G0-N-LBA-NN-NW
Compact flash module 8 GBytes	CFM01.1-08G0-N-LBA-NN-NW
VAK40.1E-DE-U-MPNN	IndraControl VAK 40 [(drawer keyboard USB, DE (German))]
VAK40.1E-EN-U-MPNN	IndraControl VAK 40 (drawer keyboard USB, EN)

Tab. 2-18: VEP accessories

2.5.7 Documentation

The following documentation is available for a detailed description of the compact industrial PC VEPxx.5:

- DOK-SUPPL*-VEP**.5****-PRxx-EN-P
xx - Edition

The following documentation is available for a detailed description of the compact industrial PC VEPxx.4:

- DOK-SUPPL*-VEP**.4****-PRxx-EN-P
xx - Edition

2.6 IndraControl VPB and VDP

2.6.1 Brief Description

The VPB 40.3 is a high-end industrial PC, which is - together with the passive VDP 16.3, VDP 40.3 or VDP 60.3 operating panels - a PC-based operating and visualization terminal with excellent industrial suitability. The VPB 40.3 is intended to be mounted into a control cabinet. The operating devices are designed to be mounted to a control panel or on a control cabinet wall. VDPxx.3 and VPB 40.3 are connected via a CDI interface.

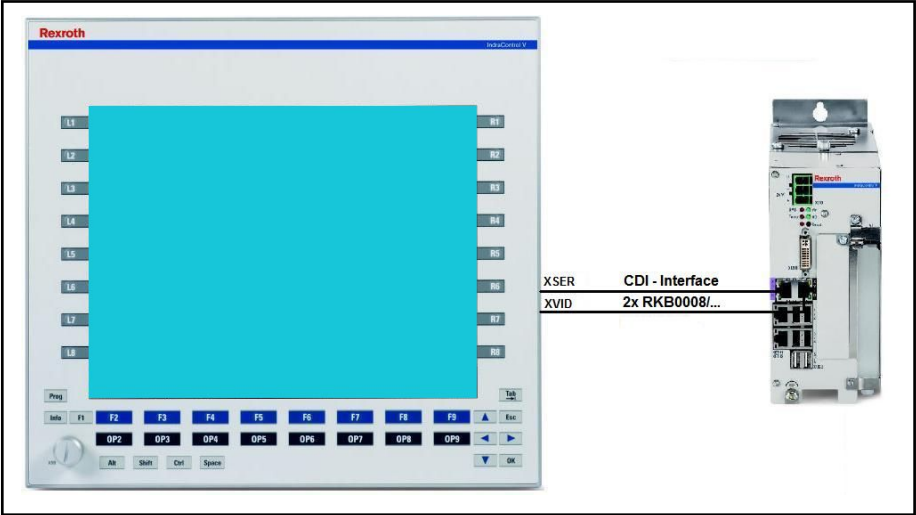


Fig. 2-17: High-end industrial PC VPB 40.3 with the operating device VDP 40.3

2.6.2 Field of Application

The compact industrial PC VPB 40.3 is used in industrial environments with higher vibration and shock requirements.

2.6.3 Technical Data

VPB 40.3

Name	VPB40.3
PC box	1, 2 or 4 slots
Processor	Intel Celeron™ P4500 1.86 GHz Intel Core i5™ 520M 2.4 GHz or Intel Core i7™620M 2.66 GHz
Main memory	2 GB or 4 GB RAM
Supply voltage	24 V DC

Tab. 2-19: Technical data for the VPB 40.3

VDP 16.3/40.3/60.3

	VDP 16.3	VDP 40.3	VDP 60.3
Display	12" color display	15" color display	19" color display
Front panel design	16 M-keys, touch screen or combined		
Supply voltage	24 V DC		

Tab. 2-20: VDPxx.3 variants

2.6.4 Wear Parts

Parts with a limited service life are installed in the compact industrial PC VPB 40.3 and the operating device VDPxx.3 and are therefore not subject to warranty.

This applies to the following components:

- Hard drive
- CMOS battery
- Fan
- Background lighting

The service life of the individual components is described in the device documentation (see [Chapter 2.6.7 Documentation, page 37](#)).

2.6.5 Variants

The following models of high-end industrial PCs VPB 40.3 are available:

Type	Slots	Remarks
VPB40.3D1N-2G0NN-C3D-DN-NN-FW	1	Celeron P4500, 2GB RAM, 160GB HDD
VPB40.3D1N-2G0NN-D4D-DN-NN-FW	1	Core i5-520M, 2GB RAM, 160GB HDD
VPB40.3D1L-2G0NN-C3D-DN-NN-FW	2	Celeron P4500, 2GB RAM, 160GB HDD
VPB40.3D1L-2G0NN-D4D-DN-NN-FW	2	Core i5-520M, 2GB RAM 160GB HDD
VPB40.3D1L-2G0NN-D4D-HN-NN-FW	2	Core i5-520M, 2GB RAM 100GB SSD
VPB40.3D1L-2G0NN-D4D-EE-NN-FW	2	Core i5-520M, 2GB RAM 2x160GB HDD (RAID1), DVD burner
VPB40.3D1L-2G0NN-D4D-DE-NN-FW	2	Core i5-520M, 2GB RAM 160GB HDD, DVD burner
VPB40.3D1L-2G0NN-D4D-EN-NN-FW	2	Core i5-520M, 2GB RAM, 2x160GB HDD (RAID1)
VPB40.3D1M-2G0NN-D4D-DE-NN-FW	2	Core i7-620M, 4GB RAM, 160GB HDD
VPB40.3D1L-4G0NN-D5D-DN-NN-FW	2	Core i7-620M, 4GB RAM, 2x160GB HDD (RAID1), DVD burner
VPB40.3D1L-4G0NN-D5D-EE-NN-FW	2	Core i7-620M, 4GB RAM, 2x160GB HDD (RAID1)
VPB40.3D1L-4G0NN-D5D-HN-NN-FW	2	Core i7-620M, 4GB RAM, 100GB SSD
VPB40.3D1Q-4G0NN-D5D-DN-NN-FW	2	Core i7-620M, 4GB RAM, 160GB HDD
VPB40.3D1Q-4G0NN-D5D-EE-NN-FW	2	Core i7-620M, 4GB RAM, 2x160GB HDD (RAID1), DVD burner

Visualization Devices

Type	Slots	Remarks
VPB40.3D1M-2G0NN-C3D-DE-NN-FW	4	Celeron P4500, 2GB RAM, 160GB HDD, DVD burner
VPB40.3D1M-2G0NN-C3D-HN-NN-FW	4	Celeron P4500, 2GB RAM, 100GB SSD
VPB40.3D1M-2G0NN-D4D-DE-NN-FW	4	Core i5-520M, 2GB RAM, 160GB HDD, DVD burner
VPB40.3D1M-2G0NN-D4D-EE-NN-FW	4	Core i5-520M, 2GB RAM, 2x160GB HDD (RAID1), DVD burner
VPB40.3D1M-4G0NN-D5D-DE-NN-FW	4	Core i7-620M, 4GB RAM, 160GB HDD, DVD burner
VPB40.3D1M-4G0NN-D5D-EE-NN-FW	4	CPU Core i7-620M, 4GB RAM, 2x160GB HDD (RAID1), DVD burner
VPB40.3D1M-4G0NN-D5D-HN-NN-FW	4	CPU Core i7-620M, 4GB RAM, 100GB SSD

Tab. 2-21: VPD 40.3 selection

The following variants of the VDPxx.3 operating devices are available:

Type	Remarks
VDP16.3BKN-D1-NN-NN	30.5 cm (12") display, membrane keys
VDP16.3DBN-D1-NN-NN	30.5 cm (12") display, touch screen
VDP40.3DEN-D1-NN-NN	38.1 cm (15") display, touch screen
VDP40.3BIN-D1-NN-NN	38.1 cm (15") display, membrane keys
VDP40.3DFN-D1-NN-NN	38.1 cm (15") display, touch screen, membrane keys
VDP60.3FEN-D1-NN-NN	48.3 cm (19") display, touch screen

Tab. 2-22: VDPxx.3 selection

2.6.6 Accessories

Connecting Cables (CDI Interface)

The compact PC VPB 40.3 and the operating device VDPxx.3 are connected via CDI interface. The following ready-made cables are available:

Type	Notes
RKB0008/002.5	Ready-made bus and Ethernet cables, ready-made 100-Base-T, CAT 6+, UL, drag chain-compatible, cable length 2.5 m
RKB0008/005,0	As above, cable length 5 m
RKB0008/010,0	As above, cable length 10 m
RKB0008/015,0	As above, cable length 15 m
RKB0008/020,0	As above, cable length 20 m
RKB0008/025,0	As above, cable length 25 m
RKB0008/030,0	As above, cable length 30 m

Visualization Devices

Type	Notes
RKB0008/035,0	As above, cable length 35 m
RKB0008/040,0	As above, cable length 40 m
RKB0008/050,0	As above, cable length 50 m

Tab. 2-23: Connecting cable VPB 40.3 - VDPxx.3



Two connecting cables of type RKB0008/... are required to connect a VPB 40.3 and a VDPxx.3.

Accessories for VPB 40.3

Type	Notes
VAP01.1H-W23-024-010-NN	24V DC mains adapter; 10A
VAU01.1U-024-024-240-NN	Uninterruptible power supply with 24 V input/24 V output/240 W
RKB0019/...	USB interface cable to connect the UPS; lengths: 0.5m; 1.0m; 3.0m; 5.0m

Tab. 2-24: Accessories for VPB 40.3

2.6.7 Documentation

The following documentation is available for a detailed description of the high-end industrial PC VPB 40.3:

- DOK-SUPPL*-VPB*40.3***-PRxx-EN-P
- DOK-SUPPL*-VAU*01.1U**-PRxx-EN-P

The following documentation provides comprehensive descriptions of the operating devices VDPxx.3:

- DOK-SUPPL*-VDP*XX.3***-PRxx-EN-P

xx - Edition

2.7 IndraControl VPP

2.7.1 Brief Description

The VPP 16.3/40.3/60.3 operating devices are active PC-based operating and visualization terminals with excellent industrial suitability. These devices are predominantly mounted on a control panel or a control cabinet wall.

Visualization Devices

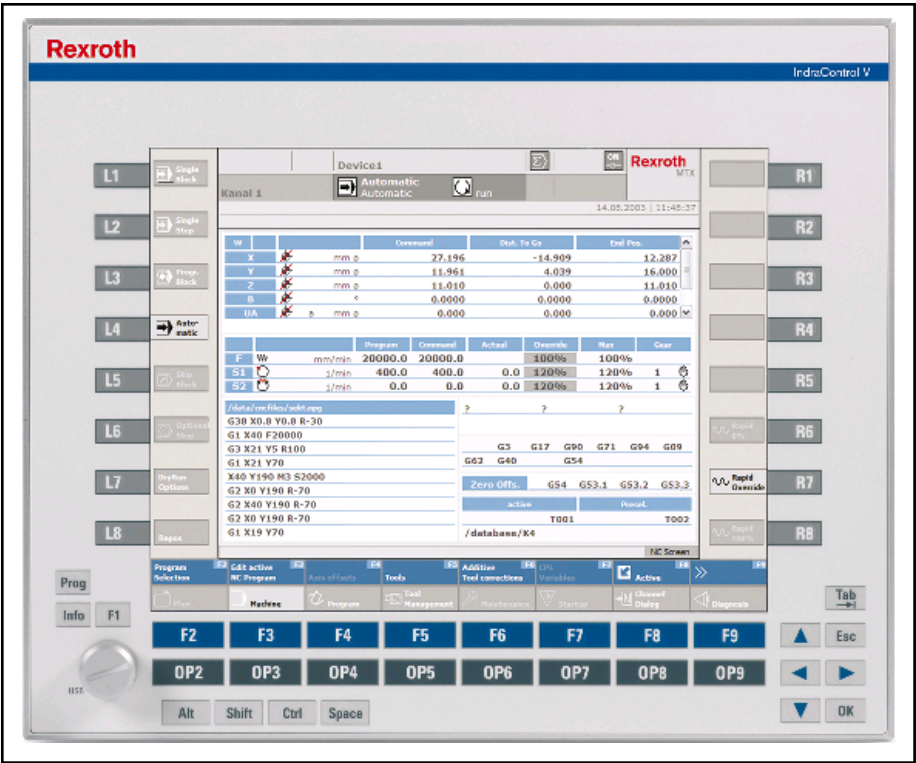


Fig. 2-18: Compact industrial PC VPP 40.3 with 15" color display

2.7.2 Field of Application

Compact Industrial PCs VPP 16.3/40.3/60.3 are used for standard as well as high-end applications.

2.7.3 Technical Data

	VPP 16.3	VPP 40.3	VPP 60.3
Display	12" color display	15" color display	19" color display
Front panel design	16 machine function keys		
PC box	1, 2 or 4 slots		
Processor	Intel Celeron™ P4500 1.86 GHz Intel Core i5™ 520M 2.4 GHz Intel Core i7™ 620M 2.66 GHz		
Main memory	2 GB or 4 GB RAM		
Supply voltage	24 V DC		

Tab. 2-25: Technical data for the VEP 16.3/40.3

2.7.4 Wear Parts

Parts with a limited service life are installed in the compact industrial PC VPP 16.3/40.3 and are therefore not subject to warranty.

This applies to the following components:

- Hard drive
- CMOS battery
- Fan

- Background lighting

The service life of the individual components is described in the device documentation (see [Chapter 2.7.7 Documentation, page 40](#)).

2.7.5 Variants

The following variants of compact industrial PCs VPP 16.3/40.3 are available:

Type	Slots	Remarks
VPP16.3DBM-4G0NN-D5D-DN-NN-FW	4	30.5 cm (12") display, 160 GB HDD, Core i7-620M, 4 GB RAM
VPP40.3DEK-2G0NN-C3D-DN-NN-FW	1	38.1 cm (15") display, touch screen, Celeron P4500, 2GB RAM, 160GB HDD
VPP40.3BIK-2G0NN-C3D-DN-NN-FW	1	38.1 cm (15") display, membrane keys, Celeron P4500, 2GB RAM, 160GB HDD
VPP40.3BIM-2G0NN-D4D-DN-NN-FW	4	38.1 cm (15") display, membrane keys, Core i5-520M, 2GB RAM, 160GB HDD
VPP40.3DEM-2G0NN-D4D-HN-NN-FW	4	38.1 cm (15") display, membrane keys, Core i5-520M, 2GB RAM, 100GB SSD
VPP40.3DFL-2G0NN-D4D-HN-NN-FW	2	38.1 cm (15") display, membrane keys and touch screen, Core i5-520M, 2GB RAM, 100GB SSD
VPP40.3BIM-4G0NN-D5D-DN-NN-FW	1	38.1 cm (15") display, membrane keys, Core i5-520M, 2GB RAM, 100GB SSD
VPP40.3DEM-4G0NN-D5D-DN-NN-FW	4	38.1 cm (15") display, touch screen, Core i7-620M, 4GB RAM, 160GB HDD
VPP40.3BIK-4G0NN-D5D-HN-NN-FW	1	38.1 cm (15") display, membrane keys, Core i5-520M, 2GB RAM, 100GB SSD
VPP40.3DEN-4G0NN-D5D-DN-NN-FW	1	38.1 cm (15") display, touch screen, 160 GB HDD, Core i7-620M, 4 GB RAM
VPP60.3FEL-2G0NN-D4D-DN-NN-FW	2	48.3 cm (19") display touch screen, 160 GB HDD, Core i5-520M, 2 GB RAM
VPP60.3FEL-4G0NN-D5D-HN-NN-FW	2	48.3 cm (19") display, touch screen, 100 GB SSD, Core i7-620M, 4 GB RAM

Tab. 2-26: VPP 16.3/40.3 variants

2.7.6 Accessories

The following accessories are available for VPP devices:

Type	Remarks
VAP01.1H-W23-024-010-NN	Power supply unit 24 V DC; 10 A
VAU01.1U-024-024-240-NN	Uninterruptible power supply with 24 V input/24 V output/240 W
RKB0019/...	USB interface cable to connect the UPS; lengths: 0.5 m; 1.0 m; 3.0 m; 5.0 m

Tab. 2-27: Accessories for the VPP 16.3/40.3

Visualization Devices

2.7.7 Documentation

The following documentation is available for the detailed description of the compact industrial PCs VPP 16.3/40.3/60.3:

- DOK-SUPPL*-VPP*XX.3***-PRxx-EN-P
- DOK-SUPPL*-VAU*01.1U**-PRxx-EN-P

xx - Edition

3 Visualizations

3.1 Overview

The IndraLogic XLC and IndraMotion MLC systems provide several visualizations. Each visualization has its own scope of application. The following visualizations are supported:

- **VI-Logic:** This visualization is primarily intended for the development and commissioning phase. The visualization runs in the IndraWorks engineering environment. Text as well as graphic elements are supported.

Refer to [VI-Logic page 41](#).

- **VI-Composer:** This software is used to configure all small operating terminals of type VCP and VCH. The functional scope is limited to text representation and simple graphic elements.

Refer to [VI Composer page 43](#).

- **WinStudio:** This visualization software is used to configure all PC-based and embedded systems. Text as well as graphic elements are supported.

Refer to [WinStudio page 45](#).

3.2 VI-Logic

3.2.1 Field of Application

Integrated visualization for testing, commissioning and diagnostics.



The IndraLogic commissioning visualization "VI-Logic visualization" is only intended to support the commissioning of machinery.



The IndraLogic commissioning visualization "VI-Logic visualization" can run on the PC as well as on the control. Different features are integrated in the visualization.

3.2.2 Properties of the Integrated Visualization

- Within a visualization, any desired expressions, even function calls, are permitted.
- A visualization can be created as an object below an application (or in the "General module" folder).
- In contrast to IndraLogic 1.x, all visualization elements as well as the visualization objects are implemented as IEC61131-3 function blocks. Thus, visualizations can be instanced in other visualizations, i.e. they can be referenced and used to extend other function blocks.
- A placeholder concept ensures that placeholders are treated as input variables of a visualization.
- Interfaces can be edited in an interface editor. Frame elements allow to easily alternate between a display of one visualization to another within one visualization object.
- Visualizations in the Project Explorer are managed by a Visualization Manager for each application. The manager defines common settings for all application-specific visualizations.

Visualizations

- Some general settings for all visualizations in one project (file paths, size adjustment, etc.) can be made in the visualization options.
- Each individual visualization has properties, such as its planned use (as "visualization", "numpad/keypad", "dialog" or display size).
- The visualization editor also includes the "ToolBox" tab providing the available visualization elements and a properties editor to configure the added elements.
- It is easy to arrange and group the visualization elements. The standard elements are provided using the visualization libraries in the project. IndraLogic 1.x visualizations can be imported.
- Visualization elements can be animated using IndraLogic project variables entered directly or using expressions (combinations of variables with operators and constants). This allows variable values to be scaled for example, so that the variable values are suitable for the visualization.
- Language selection (ANSI or UNICODE) in a visualization can be executed by using text lists.
- The variable values of an application can be modified and displayed using a text field element. Formatting these inputs and output is based on the standard function "sprintf" (C-library).
- External data sources can also be used.
- A tool tip can be assigned to each visualization element.
- In addition to the zoom function, automatic scaling is also possible (visualization adjusts to screen size).
- Visualizations can be saved in libraries to provide them in other projects.
- Multilingualism

The VI-Logic visualization supports multilingualism of static texts via "GlobaleTextLists" and dynamic texts via "TextLists". For dynamic texts, the text displayed depends on a variable value.

3.2.3 Visualization Elements

Commissioning visualizations can be created by various visualization elements divided in "Basic", "General Control Elements" "Measuring Devices", "Lamps/ Switches/ Screens" and "Special Control Elements". For the available visualization objects, refer to the following list.

Visualization elements

Basic	General Control Elements	Measuring Devices	Lamps/ Screens	Switches/	Special Control Elements
Pointer	Button	Pointer instrument	Screen changer		Trace
Rectangle	Scroll bar	Histogram	Lamp		ActiveX element
Rounded rectangle	Table	Bar graph			
Ellipse	Text field	Lever button			
Line		Energy button			
Polygon		Pushbutton			
Polyline		LED pushbutton			
Bézier curve		Toggle switch			

Visualizations

Screen		Rotary switch		
Frame				

Tab. 3-1: Visualization elements acc. to sections

The visualization objects are configured via static and dynamic properties as well as event handling for user inputs in the "Properties" window.

Predefined actions, such as executing limited ST code (only assignments, but no loops or branches are permitted), affecting variables, screen and language selection, internal commands as well as opening and closing of dialogs, can be selected from the actions provided for user entries.

3.2.4 Supported Visualization Devices

No additional hardware is required for the visualization, as visualization and engineering run on one device.

Engineering is simplified, as the visualization and the PLC application are configured on one interface.

3.2.5 Protocol

The integrated visualization can directly access variables from the control. Externally generated lists of variables or symbols are not required.

Automatic configuration of the communication. An error-prone communication configuration is not required.



The "VI-Logic" visualization and its visualization objects provide all variables for project planning used in the PLC program (implementation) or declared with the attribute "{attribute 'linkalways'}".

Thus, variables which are only in the the declaration are optimized by the compiler from runtime code and can therefore not be visualized or changed.

3.2.6 Documentation

For a detailed description of the integrated visualization software, refer to the IndraWorks online help.

3.3 VI Composer

3.3.1 Field of Application

This VI-Composer software uniquely configures all small operating terminals of type VCP and VCH.



The **VI Composer** software is not part of IndraWorks and has to be ordered separately.

3.3.2 Configurable Small Operating Terminals

The following small operating terminals can be configured with the VI-Composer software:

- VCP:
 - IndraControl VCP 02
 - IndraControl VCP 05
 - IndraControl VCP 08

Visualizations

- IndraControl VCP 11
- IndraControl VCP 20
- IndraControl VCP 25
- IndraControl VCP 35
- VCH
 - IndraControl VCH 08

See also [IndraControl VCP](#), page 17.

3.3.3 Communication Protocol BRC-Symbolic

Brief Description

The protocol provides read and write access to all global data objects of the control.

The configuration software applies the data objects of the symbol file (file-name.SYM or filename.xml) generated during the compilation of the control object.

The connected operating device accesses a file object via the symbolic name.

Data Objects

Generally, the variable length is specified by a definition in the "IndraLogic" configuration software.

Individual Variables

Variables of type BOOL, SINT, INT, DINT, BYTE, USINT, WORD, UINT, LWORD, DWORD, UDINT, REAL, STRING, WSTRING, LINT, ULINT, LREAL, BITORBYTE, ENUM, TIME, TOD, DATE and DT can be accessed.

String Variables

Use the variable type STRING(N) or WSTRING(N) for string variables. N specifies the string length.



String variables cannot exceed a length of 64 characters.

Tables

To display ARRAY variables in table elements, use data type ARRAY [1..N]. The ARRAY [1..N] has to consist of one of the following basic types:

BOOL, BYTE, WORD, LWORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, LINT, ULINT, LREAL, REAL, STRING, WSTRING, ENUM

3.3.4 Connection Settings

Protocol Parameters for PROFIBUS-DP

- Device address

This parameter specifies the operating device address.

Adjustable values	Default value
1 to 126	2

Tab. 3-2: Device address

- Delay until connection is established.

This parameter specifies a waiting time until the communication is started by the operating device.

Adjustable values	Default value
5s to 255s	5s

Tab. 3-3: Delay time when establishing a connection

- Response time of slave

This parameter specifies how long the operating device waits for control response

Adjustable values	Default value
10ms to 65535ms	1000ms

Tab. 3-4: Response time of slave

Protocol parameter for Ethernet

Connection table:

Connections up to 16 controls can be defined in the connection table.

The connection names of the main dialog are applied to the "Connection Name" column.

Enter a valid IP address for each connection name to the "IP Address" column of the control.

3.3.5 Documentation

The following documentation is available to provide comprehensive descriptions of the VI-Composer configuration software:

DOK-SUPPL*-VIC*BED*0... en-EN

3.4 WinStudio

3.4.1 Overview

WinStudio is a visualization software for all PC-based and embedded systems.

The product consists of two parts.

WinStudio Engineering

- Configuration tool to create individual HMI screens up to complete user interfaces.
- One-time investment (license) for any number of WinStudio projects.
- Scalable
 - WinStudio Lite license (limited functions; included in the standard scope of the IndraWorks systems).
 - Scaled license model acc. to number of tags used (process variables).
- WinStudio Engineering package characteristics.
 - Integral part in IndraWorks Engineering to create visualization applications.
 - Addition to IndraWorks HMI interface (IndraWorks OPD).
 - Creation of individual visualizations (WinStudio Runtime).

Visualizations

- Stand-alone editor to create individual visualizations (WinStudio Engineering stand-alone).

WinStudio Runtime

- Software on visualization devices
 - Already installed or
 - prepared for installation (software download).
- Scaling
 - WinStudio Lite license (limited functions; included in the standard scope of the IndraWorks systems).
 - Scaled license model acc. to number of tags used (process variables).
- WinStudio runtime environment characteristics.
 - WinStudio in IndraWorks HMI interface (IndraWorks OPD).
(Ready user interface of the systems MLC, MTX and IL under Windows CE/Windows 7 embedded compact, Win XP/ Win XPe, Win 7/ Win 7e)
 - WinStudio Runtime (IndraWorks-independent user interface).

3.4.2 WinStudio Elements in the Project Explorer

To create WinStudio screens and to configure advanced function (e.g. processes, alarms) and the communication connection, different editors and dialogs are available. Open them via the context menu of the "WinStudio" or other subfolders.

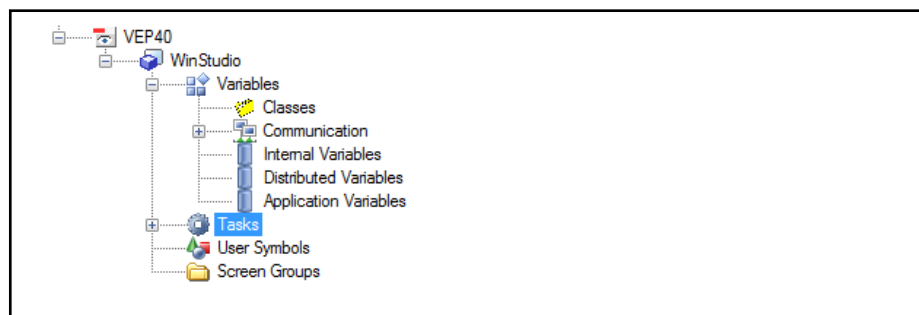


Fig. 3-1: WinStudio folder

For the following subfolder, got to the "WinStudio" node in the Project Explorer:

- **Variables**
In this folder, there are editors for different variable types (application variables, system variables), editors to configure the process communication (driver sheets) and to create individual data types (classes).
- **Tasks**
Extended WinStudio objects (e.g. mathematic and alarm sheets) can be created and edited in this folder.
- **User Symbols**
in this folder, there are self-created WinStudio objects that can be used for WinStudio screens.
- **Screen Groups**
In this folder, screen groups can be created that summarize several WinStudio screens to a common screen.

The following WinStudio elements are available if at least one license "WinStudio-Engineering and Runtime 1.5k" is activated and "WinStudio 1.5k" or higher is selected as target system of the WinStudio project.

- [Alarms, page 64](#)
- [Recipes, page 64](#)
- [Reports, page 65](#)
- [Trends, page 66](#)
- TCP/IP driver sheets
- DDE driver sheets

When opening a project with "WinStudio 1.5k" or higher selected as target system, but no active matching WinStudio license, the elements requiring a license are displayed. However, they are disabled and cannot be edited.

3.4.3 General WinStudio Settings and Functions

Project properties

Select "Project Settings" to open the dialog with the same name to make basic project settings.

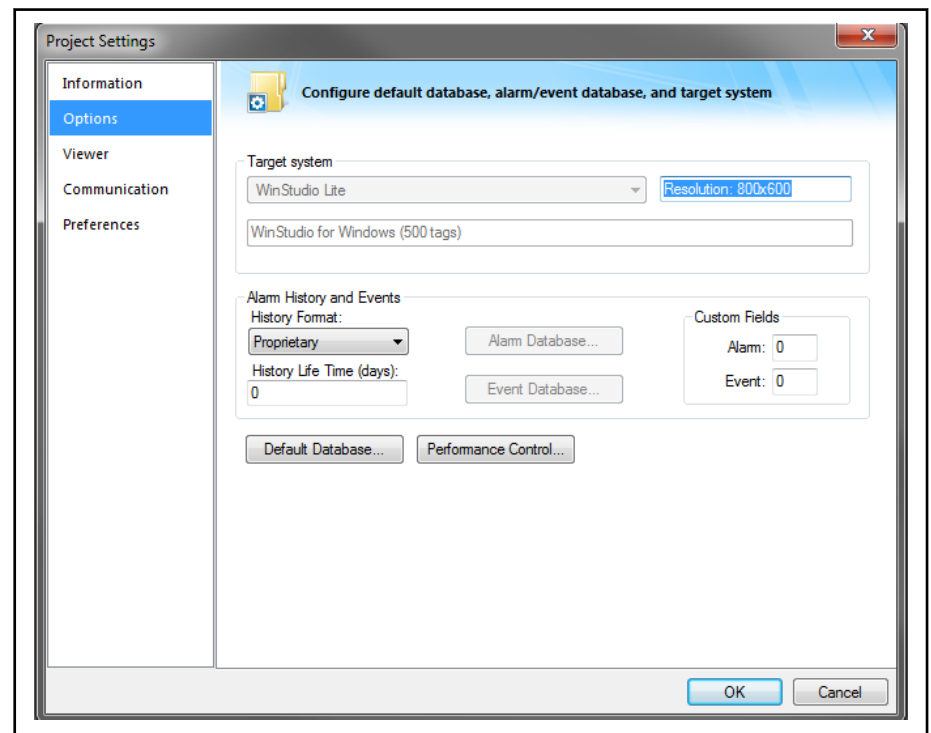


Fig. 3-2: Dialog: Project Settings

Subsequently, the tabs are written:

- [Information, page 48](#)
- [Options, page 48](#)
- [Runtime Desktop, page 49](#)
- [Communication, page 53](#)
- [Web, page 54](#)

Visualizations

Information

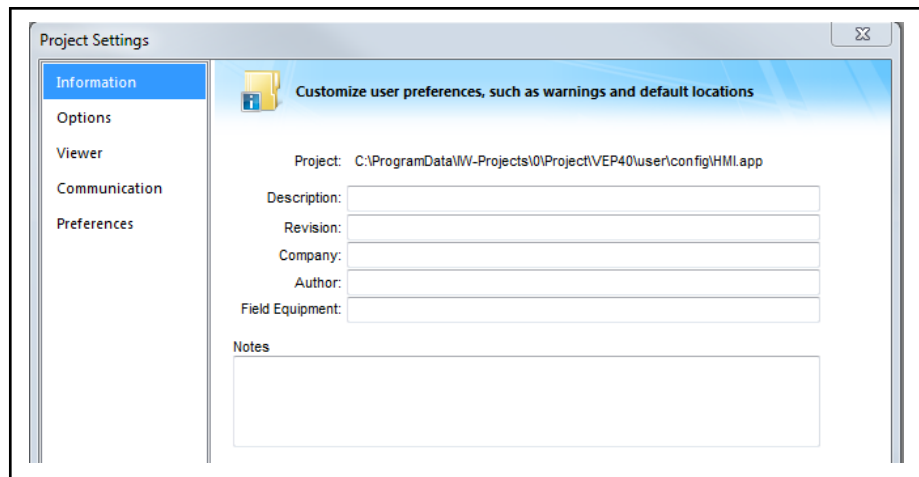


Fig. 3-3: Dialog: Project Settings (tab: Information)

The tab in the "Project Settings" dialog describes the project for documentation purposes.

Options

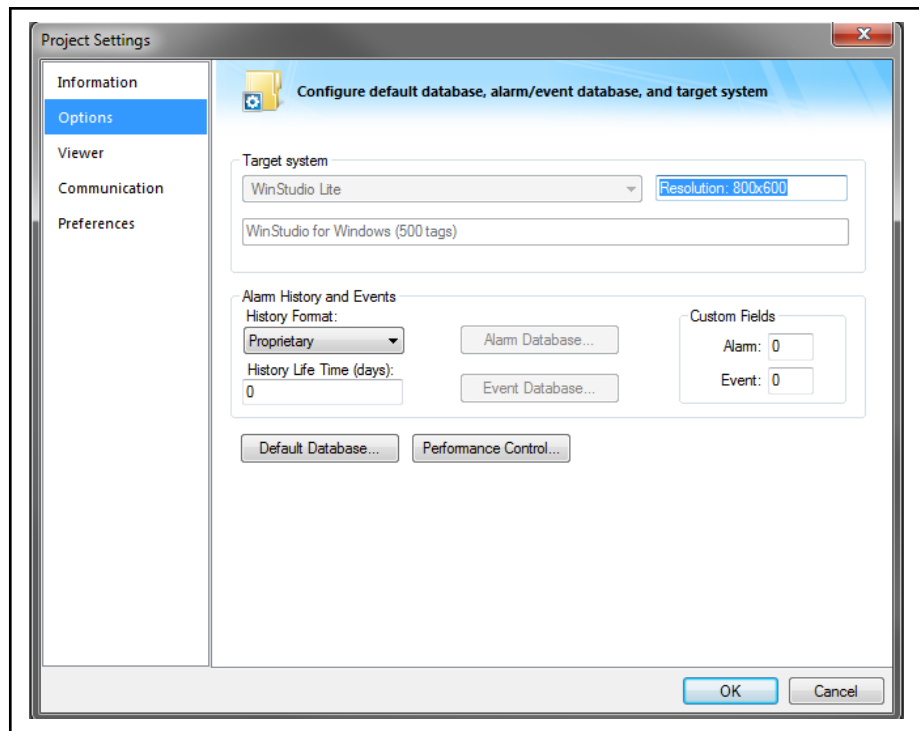


Fig. 3-4: Dialog: Project Settings (tab: Options)

Specify general project parameters in this tab:

- **Target system**

Select the target system and thus specify the number of available WinStudio variables, the available functional scope as well as the project type matching the target system (Win XP (e)/ Win 7(e) or Windows CE/ Windows 7 embedded compact). Default setting: "WinStudio Lite". More target systems with a higher number of variables and an advanced functional scope are available after the activation of the WinStudio Engineering license.

The screen resolution is selected via the configured HMI device.



If a license is selected that does not match the current license on the operating station, the application might not be executed correctly.

• Alarm History and Events

Enter a value (in days) in the "History Life Time (days)" field to specify the period the alarm and event history files should be available. After this period, WinStudio automatically deletes the present alarm and event history (older than the period specified).



If 0 is entered as value, the history files are not automatically deleted. In this case, ensure that the files are deleted, as memory space is continuously assigned to the system.

Select "History Format" to specify how and where to save the alarm files.

Property	Saves history data to the subfolder \alarm of the application (by default) in text files by using a proprietary WinStudio format.
Database	Saves the history data to a user-defined SQL relational databases by using the integrated ADO interface.
Binary	Saves history data to the subfolder \alarm of the application (by default) in binary-coded files by using a proprietary WinStudio format.

• Default Database

A default database can be configured to be used by different tasks and objects.

Runtime Desktop

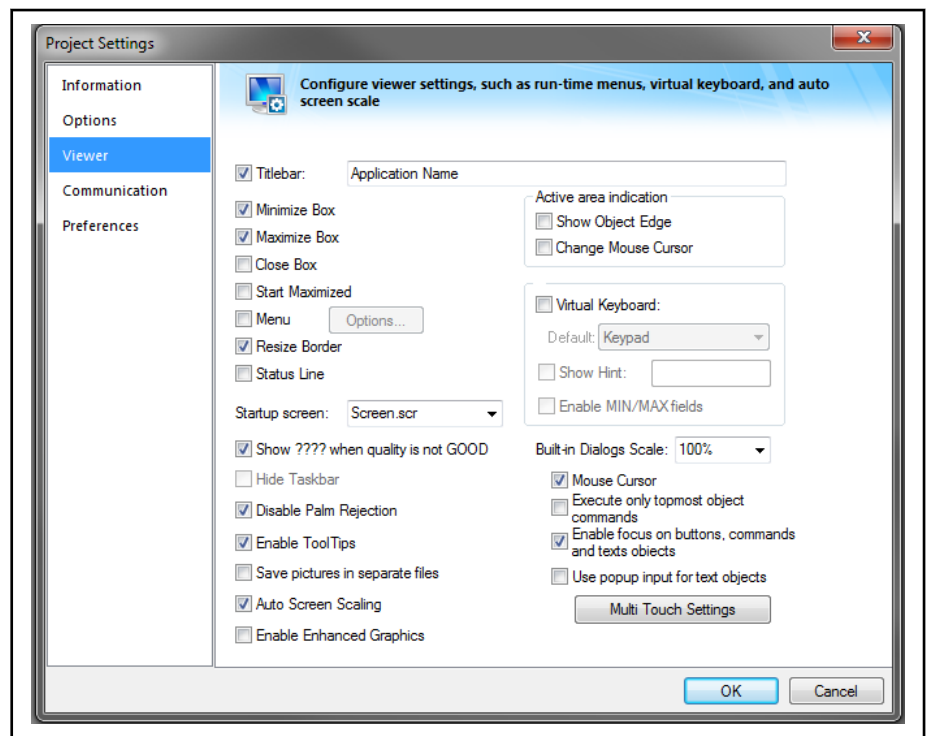


Fig. 3-5: Dialog: Project Settings (tab: Viewer)

Visualizations

The runtime desktop parameters can be enabled/disabled in this tab:

- **Checkbox and text field Title bar**
Specify the default text for the title bar of the "Viewer" window.
- **Checkboxes Minimize Box, Maximize Box and Close Box**
Enable these checkboxes to show (enable) or hide (disable) these buttons in the "Viewer" window.
- **Checkbox Maximize Box**
The application is maximized when executed.
- **Checkbox Menu and Options button**
Enable the checkbox and the "Options" button to specify the menu options available at runtime. In the "Runtime Menu Options" dialog, menu options can be shown (enable) or hidden (disable).
- **Checkbox: Resize Border**
Resizes the "Viewer" window at runtime.
- **Checkbox: Status Line**
Display the status bar in the runtime application.
- **Field: Startup screen**
Enter the name of the WinStudio screen file to be used as startup screen when opening the runtime application.
- **Checkbox: Show ??? when quality is not GOOD**
Question marks (???) are displayed instead of the tag value if problems occur at runtime while processing data
- **Checkbox: Hide Taskbar**
Hides the taskbar (only Windows CE/ Windows 7 embedded compact)

Disable Palm Rejection: Disables "Palm Rejection" at project runtime. "Palm Rejection" is a Windows 7 touch screen device characteristic. Accidental touching is detected and the signal is not applied. This can slow down the triggering response of the touch screen.



Palm Rejection is only available on Windows 7 touch screen devices. Thus, selecting **Disable Palm Rejection** has no effects on projects running on other target systems.



With Windows 7 touch screen devices, a right-click can be performed by touching the screen for a longer period (press and hold function) This option is enabled by default. Disable the "press and hold" function as follows:

1. Click on "Start" -> "Control Panel". The "Control Panel" window is displayed.
2. Double-click on stylus and touch. The stylus and touch dialog is displayed.
3. Click on the "Touch" tab.
4. Select "Press and hold" from the touch action list and click on "Settings". "Press and Hold - Settings" dialog is displayed.
5. Delete "Enable press and hold for right click".
6. Click on "OK" to close the "Press and Hold - Settings" dialog.
7. Click on "OK" to close the "Stylus and Touch" dialog.

- Checkbox: **Enable ToolTips**

Enable this option to display the Windows tooltips at runtime. The tooltips can be entered for each object into the "Object Properties" dialog as note.

- Checkbox: **Auto Screen Scaling**

The application screens can be automatically resized to the display size. This function is available for local applications under Windows XP/ Windows 7 (or Web Thin Clients). This parameter is not available for local applications under Windows CE/Windows 7 embedded compact.



Only use "Auto Screen Scaling" if the site proportion of the target display corresponds to the site proportion of the WinStudio screens, as the screen elements are distorted otherwise.

- Section: **Active area indication**

Enable the options "Show Object Edge" and "Change Mouse Cursor" in this area to change the object edge and the mouse cursor when the cursor moves over an object.

- **Input section**

Checkbox: **Virtual Keyboard**

Shows the mouse cursor in the runtime application.

- Checkbox **Show Hint**: A tip is shown in the title bar of the keyboard. This tip is specified for the individual objects in the "Object Properties" dialog. Otherwise, enter a string value to the "Show Hint" field. It is then used as default tip.
- Checkbox **Enable Min/Max fields**: Displays the permitted minimum and maximum values at the lower edge of the keyboard. These values can be specified for the individual WinStudio objects in the "Object Properties" dialog. Otherwise, the "Min" and "Max" properties of the corresponding tag are used.

Visualizations



The Min and Max fields are only displayed on the keyboard type and also only if the assigned tag is defined as Integer or Real. If Min is higher than Max, the input is disabled. If the Min/Max pair set at the object differs from the Min/Max properties in the tag, it is tried to convert the input variable accordingly.

- Checkbox **Mouse Cursor**: Displays the mouse cursor at runtime.



The "Mouse Cursor" option is not supported under Windows CE/Windows 7 embedded compact if executed on the Armv4l processors.

- Checkbox **Execute only topmost object commands**: Specifies project response if the user clicks on a section with two or more overlapping screen objects. If this checkbox is enabled, only commands assigned to the object at the topmost screen level are executed. If this checkbox is disabled, the commands of all overlapping objects are executed.



The object with the highest object ID is the topmost one. The object ID is displayed in the status bar at the lower edge of the engineering environment. Using the Tools "Move backward/forward", the sequence the objects overlap each other can be changed.

- Checkbox **Enable focus for buttons, commands and text objects**: The object as shown in the following figure is focused when clicking on the command button or entering a text at runtime.



Fig. 3-6: Focus on text input (left). No focus on text input (right)

Use the **Tab** key to subsequently navigate to all objects that can be focused in the screen or enable the object currently focused with **Enter**. This option is enabled by default.

- Checkbox **Use popup input for text objects**: Select this option if text entries are to be shown for a small popup window.

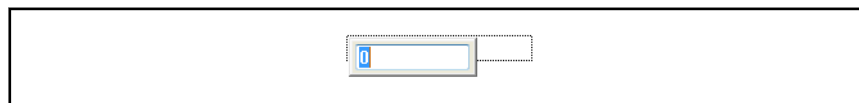


Fig. 3-7: Popup for text entries



If the above described checkbox **Virtual Keyboard**, it disables this checkbox.

It also applies: If the above described checkbox **Enable focus on buttons, commands and text objects** is "not" set, this checkbox is automatically selected. This ensures that there is a note in the screen indicating the currently active text input.

- **Multi Touch Settings** dialog. When projecting multi touch-capable visualization hardware, configure them via the "Multi Touch Functions" dialog

Communication

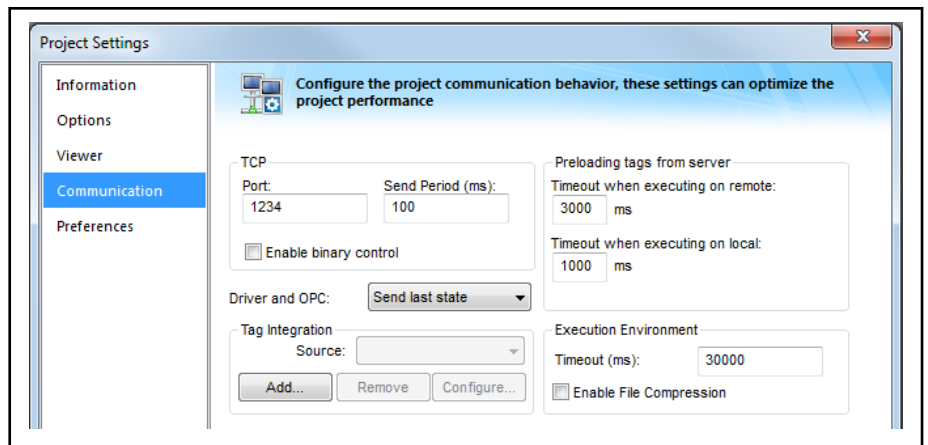


Fig. 3-8: Dialog: Project Settings (tab: Communication)

Specifies general communication parameters:

- **Driver and OPC**

Selecting the method used by all communication drivers and OPC client worksheets in the current application.

- **Send every status**
Each value change of a tag is transmitted to the control.
- **Send last status**
Only the last value is transmitted to the control.

- **TCP**

Configuration dialog for TCP/IP client and TCP/IP server module settings

- **Port**
TCP port used by the TCP/IP client and the TCP/IP server modules running on the current computer. If this value is changed in the local application, ensure that the value is also changed in the remote application that communicates with the local application.
- **Send Period (ms)**
Update interval in ms between two subsequent messages sent from the TCP/IP client or the TCP/IP server modules.
- **Enable binary control**
Transmits information sent by the TCP/IP client or the TCP/IP server module to the remote station in encrypted form. An encryption increases system security and decreases communication performance. If two stations have been connected via the TCP/IP client and the TCP/IP server module, ensure an identical setting for both applications.

- **Execution Environment** section: Configure the communication settings for the execution environment (platform). It is the tool used to transfer project data to the target system.

- **Timeout (ms)** box: Specifies the waiting time (ms) the execution environment waits for a successfully established connection to the target system.
- **Enable File Compression** Checkbox: Compresses system and project files before sending them to the target system. Only useful in case of slow connections. File compression is disabled by default.

Visualizations

- **Preloading Tags from Server** section: To improve performance, all server tags of the screen to be displayed are downloaded to the memory before displaying a screen. Configure timeout settings for remote and local viewer:
 - **Timeout when executing on remote** field: Specifies the period (ms) the secure viewer executed on a remote station or the thin Client wait for tags to be loaded.
 - **Timeout when executing on local** box: Specifies the period (ms) the viewer executed on a local station (i.e. the server) waits for tags to be loaded.
- **Tag integration** section: Used to automatically generate a driver sheet. For more information, refer to the IndraWorks WinStudio Online help "Rexroth WinStudio 7.3 Technical Reference".

Web

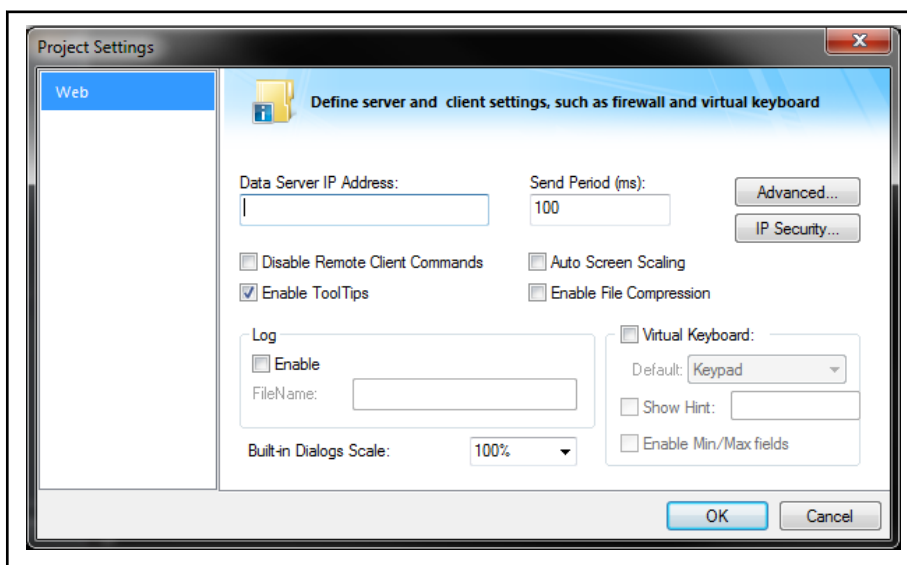


Fig. 3-9: Dialog: Project Settings (tab: Web)

Call the tab via **WinStudio ► Tools ► Web Settings**, and the parameters are specified for a web thin client:

- **Field: Data Server IP Address**
Enter IP address (or the host name) of the data server station. The data server station is the device on which the TCP/IP server module of WinStudio is running.
- **Field: Send Period (ms)**
Specifies the update rate (ms) between server and web thin client
The default value is 100 ms. Decreasing the value increases the communication work in the network.
- **Checkbox: Disable Remote Client Commands**
Enabled checkbox prevents that a remote client passes commands from the web thin client to the server. If enabled, the web thin client can read data from the server but cannot send any data (tag values, set points) to the data server. The web thin client station becomes a "Read-Only station".
- **Checkbox: Enable ToolTips**
Checkbox **Log**:

Enable and **FileName**: Enter the file name into the text field to create a log file on the thin client station. This log file can be used for troubleshooting.

- **Virtual Keyboard** section: If enabled, the virtual keyboard is enabled for thin clients. Use this keyboard to enter data on the touch screen at run-time. No hardware keyboard is required.

The following settings can be made for the virtual keyboard:

- **Default**: Specifies a default keyboard type to be used in the project if no keyboard type was specified by the calling object or function.
- **Scaling**: Zooms keyboard in and out to adapt to the size of the target screen. 100% indicates the default size of the keyboard type.
- Checkbox **Show Hint**: A tip is shown in the title bar of the keyboard. Enter a text or a string into the **Show Hint** field then used as default tip.
- **Enable Min/Max fields**: The minimum and maximum values are displayed at the lower edge of the keyboard. These values are specified for objects in the "Object Properties" dialog. Otherwise, the "Min" and "Max" properties of the corresponding tag are used.



The **Min** and **Max** fields are only displayed on the keyboard type **Keypad**. The assigned tag has to be of type Integer or Real. If Min is higher than Max, the input is disabled. If the Min/Max values set at the object differ from the Min/Max values in the tag, WinStudio tries to convert the input variable accordingly.

- Tooltips configured for objects on the screens, can be displayed if to be viewed on the web thin client (web browser).
- Checkbox: **Auto Screen Scaling**
Autosizes the screens in a web browser window. Web browser size adjusts to the screens irrespective of the screen resolution.



The checkbox "Automatic Screen Scaling" does not apply to web browsers running in Windows CE/ Windows 7 embedded compact.

- Checkbox: **Enable File Compression**
The files saved in the \Web folder of the application are compressed when enabling this checkbox. This is useful to reduce download time in case of slow connections.

IP security settings

IP Security: Click on this button to open the "IP Security" dialog.

Visualizations

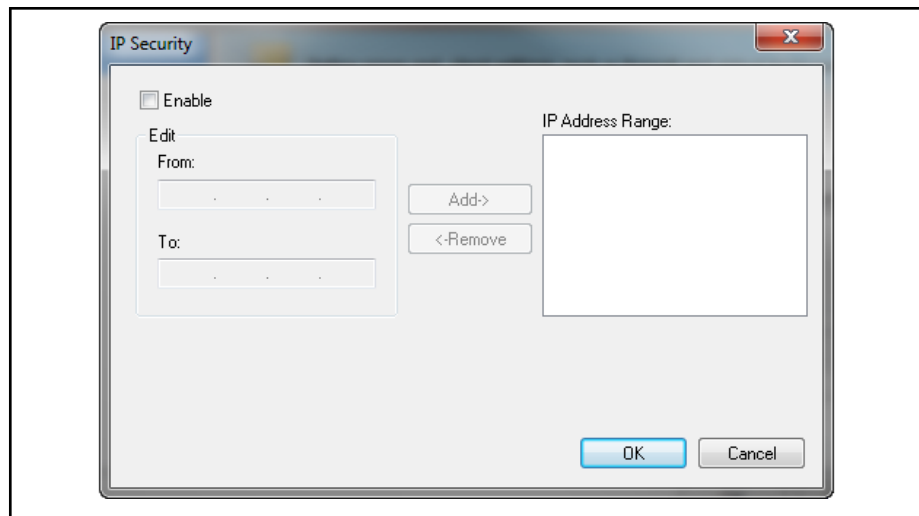


Fig. 3-10: "IP Security" dialog

Use these dialog parameters to specify the IP address range for the computers that can access the project as thin client.

Select **Enable**. Once the parameters of the **Edit** range become active, enter the IP addresses into the **From** and **To** fields to name the IP address range. Use **Add** and **Remove** to apply the IP addresses to the **IP Address Range**. If a thin client tries to connect to the server, it checks the defined address range and does not allow to establish a connection to the thin client station if its address is not available.



By default, IP security only applies to thin clients connecting to the data server. IP security can also be implemented for database synchronization of projects running on different stations. Add the following parameters to the project file (**project_name.app**):

[TCP]

UseWebIPSecurity=1

Advanced Settings

Advanced: Opens a dialog to edit more web settings. Modifying these settings is normally not required. Depending on the architecture used in the project, more settings can be configured.

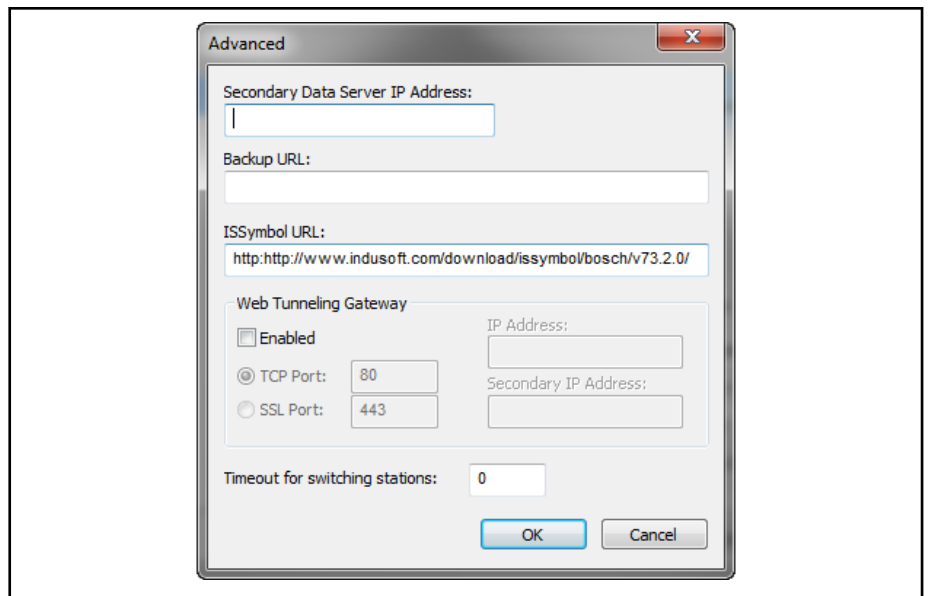


Fig. 3-11: Advanced settings dialog

- **Secondary Data Server IP Address** field: Enter the IP address (or host name) of the secondary data server station. A data server station is the computer or device on which the "WinStudio" TCP/IP server module is executed. Fill out this field if redundant data servers are used coming from thin clients. If the primary data server fails, the thin client automatically tries to connect to the secondary data server.
- **Backup URL** field: Enter the URL of the location the web files are saved (files from the "Web" subfolder of the project folder). This URL is used to download the files from the secondary web server if the primary web server is not available.
- **ISSymbol URL**: field If the thin client connects to the server, it tries to load Control ISSymbol. If ISSymbol is not registered on the local PC (thin client), the browser tries to download it from the URL specified in this field. The default URL website is the website "Bosch Rexroth" provides the current ISSymbol version for download. You might configure a different location, especially if the thin client computer is not connected to the internet. The **ISSymbolVM.cab** file (stored in the subdirectory \BIN of "WinStudio") has to be available with the URL configured in this field.



If the thin client stations cannot access the internet, it is recommended to provide the **ISSymbolVM.cab** file on the web server station and to configure the URL in this field.



Web browsers under Windows CE cannot automatically download the Control ISSymbol (**ISSymbolCE.ocx**) from a remote location. **ISSymbolCE.ocx** has to be registered manually in the Windows CE device used as thin client.

- **Web Tunneling Gateway**: Enables the web tunneling gateway. Depending on the project architecture, the connection from the thin client computers to the data server might have to run via a web tunneling gateway.

Visualizations

- **TCP Port:** Select this radio button if HTTP is used with the Microsoft IIS web server. Specify the TCP port used by the HTTP web server ("80" is the default TCP port for the HTTP protocol).
- **SSL Port:** Select this radio button if SSL (Secure Socket Layer) is used with the Microsoft IIS web server. Specify the TCP port used by the HTTPS web server ("443" is the default TCP port for the HTTPS protocol).
- **IP Address:** The IP address of the web server computer used to execute the web tunneling gateway. This has to be the IP address of the web server accessible from the thin client stations.
- **Secondary IP Address:** The IP address of the web server computer used to execute the secondary web tunneling gateway. This has to be an IP address of the secondary web server accessible from the thin client station(s). Configure this field if redundant web servers are used.



Most web settings can be changed dynamically via the "SetWeb-Config" function. This is especially useful when a project is created only once in order to ensure that the web settings are automatically configured if the project runs on different working stations with different IP addresses.

Project status

The runtime components have to be enabled to correctly execute some of the WinStudio functions. For an overview on all WinStudio runtime components, go to **WinStudio ► Status....**

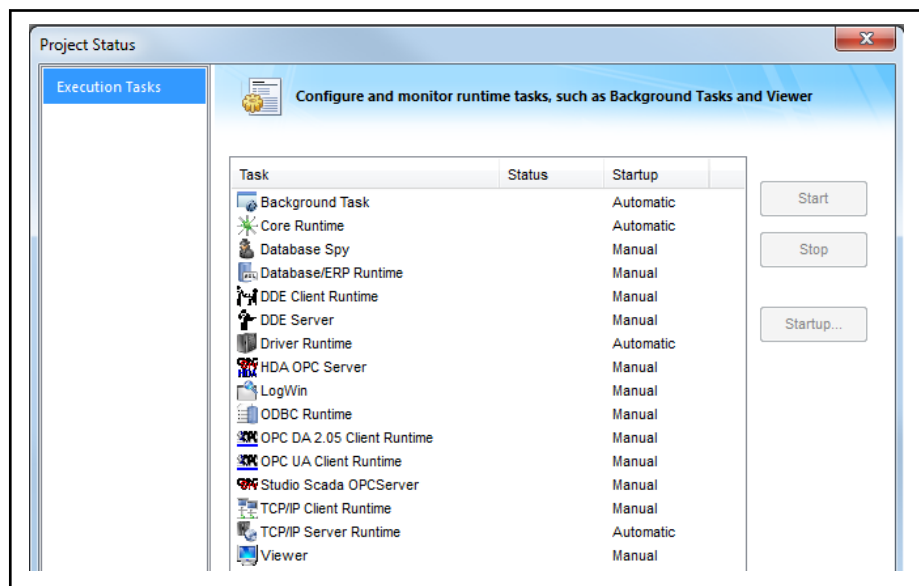


Fig. 3-12: Dialog: Project Settings (tab: Execution Tasks)

All available WinStudio runtime components are displayed in the "Status" dialog. The status as well as the start mode are displayed for each runtime component. The following runtime components are enabled by default at startup:

- Background Task
- Core Runtime
- TCP/IP Server

Tasks can be started and stopped at runtime and it can be selected which tasks are automatically or manually started via the "Start", "Stop" and "Startup" buttons.

Loading WinStudio project The current WinStudio project including all WinStudio objects can be replaced by another WinStudio project to use an existing WinStudio project as template. To replace the current WinStudio project, go to **WinStudio ► Load project...** A file browser opens used to select a project file (file extension "*.app"). After the file selection has been confirmed, all current WinStudio elements are deleted and new elements of the selected project are inserted.

3.4.4 WinStudio Screens

Creating and Editing WinStudio Screens

To create a WinStudio screen, go to **WinStudio ► New Screen**. A new element with the name "Screen" is added to the "WinStudio" folder. This name can be changed now or later.

To edit an available WinStudio screen, open the WinStudio screen editor by double-clicking on the element in the project tree.



Some elements can only be used if the WinStudio-Engineering license 1,5k or higher is activated. Elements requiring with a license are disabled if license is not adequate.

When opening a WinStudio screen, additional toolbars with frequently used function are displayed in the presetting. The presetting can be changed via **Tools ► Customize...** and the provided toolbars can be hidden or further toolbars can be added. All editing functions of a WinStudio screen are also available via the IndraWorks Engineering menu bar via **WinStudio Screen**.

Screen Properties

More options to change the screen size and to use a background graphic are available in the screen properties. Open the "Screen Attributes" dialog via the context menu of the "Screen Editor" via the menu item **Screen attributes**.

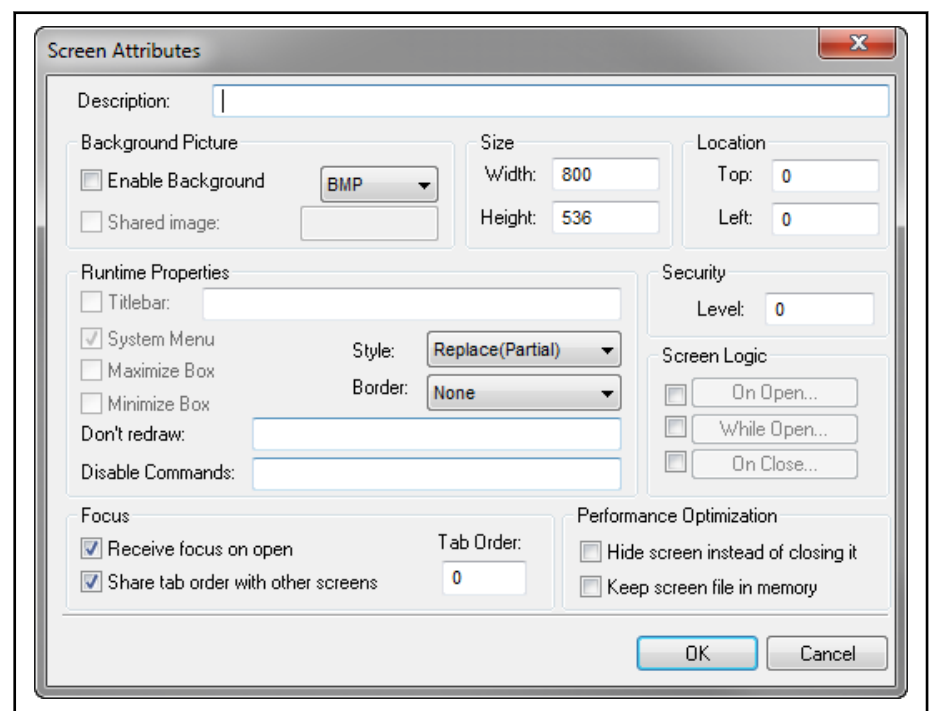


Fig. 3-13: Attributes of a WinStudio screen

Dynamic object properties

Visualizations

Some properties (color, size, position) of WinStudio objects that can be added to a WinStudio screen can be changed dynamically. Moreover, functions - that can be executed when activating the WinStudio object in IndraWorks Operation - can be configured.

Several dynamic properties can be assigned to a WinStudio object by enabling them individually. The following dynamic properties are available:

- **Command:** Function to be executed while enabling the element
- **Hyperlink:** Opens when enabling the hyperlink
- **Bar diagram:** Represents a variable value as bar diagram
- **Text input/output:** shows the value of a WinStudio variable or applies an entered value
- **Color envelope:** Changes the color depending on a WinStudio variable
- **Position:** Changes the X- and Y-position of the object depending on a WinStudio variable
- **Size:** Changes width/height of the object depending on a WinStudio variable
- **Line rotation:** Rotates a line depending on a WinStudio variable

Object Properties and Variable Selection

The properties of a WinStudio object can be edited using the property dialog. Basic properties (e.g. line width) can be edited. The scope of the provided properties depends on the respective object type.

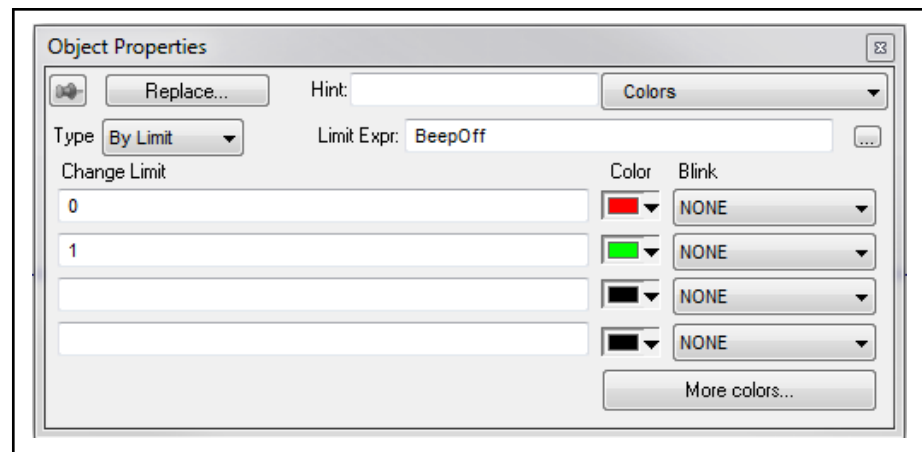


Fig. 3-14: Object properties

Variables of the following areas can be selected in the selection dialog:

- Project - Tags from the application variable database created during configuration
- System - WinStudio system variables (e.g. minute) from the internal variable database
- Distributed- Variables from automatic driver sheet generation (if configured) from the distributed variable database

3.4.5 WinStudio Variables

WinStudio allows to execute action (e.g. computations) with regard to a variable. Computation results can be assigned to variables and WinStudio variables can be linked to process variables.

In the "Variable" folder, the following variable categories are available:

- **Internal Variables:** Predefined, fixed variables

- **Distributed Variables:** Variables linked to process variables (automatic driver sheet generation)
- **User Variables:** Individual, user-specific variables

Classes *WinStudio provides the following data types for own variables:*

- Bool
- Integer - 32 bit, signed integer
- Real - 64 bit, signed floating point
- String - Character string up to 1024 characters

Moreover, individual data types, so-called "classes" can be defined that are composed of standard types.

	Name	Type	Description
1	 yPos	Integer	x-Position
2	 xPos	Integer	y-Position
3	 Title	String	Title
*		Integer	
*		Integer	
*		Integer	
*		Integer	
*		Integer	

Fig. 3-15: WinStudio class editor

In the class editor, element names can be added to the "Name" column and a data type can be assigned under "Type". After saving the new data class, they can be used for WinStudio application variables.

Communication WinStudio variables can be linked to PLC variables to show the PLC variable value in a WinStudio screen, to use it for calculations or to change it. WinStudio variables are linked to PLC variables via so-called driver sheets. In addition to the variable list, a driver sheet contains different settings required to communicate with the PLC. The driver sheet name consists of a unique number and an optional description text. OPC-DA, OPC-IwSCP, OPC-UA as well as BRWS driver are available for the communication.



The value ranges of the WinStudio variables have to cover the value ranges of the control variables linked with the application.

3.4.6 Linked Symbols

WinStudio Library

WinStudio provides a comprehensive library of symbols that can be inserted into WinStudio screens. Self-defined symbols (user icons) can be created and used again.

The WinStudio library is represented in a tool window in the lower section of IndraWorks Engineering (see next figure). The library consists of many folders. WinStudio symbols are sorted in these folders.

Inserting symbol Open the WinStudio tool window via the WinStudio node **WinStudio ► Tools ► WinStudio Library**. To insert a symbol into a WinStudio screen, open the WinStudio screen. Subsequently, select a symbol by selecting it in the preview window. By clicking on the WinStudio screen editor, the symbol can be inserted into the desired position.

Visualizations

Parameterizing symbol

Each WinStudio symbol has several parameters that have to be adjusted correspondingly. Double-click on the symbol in the WinStudio screen to open the "Parameterization" dialog. All symbol parameters are sorted by category and displayed by clicking on the category. Change individual parameters by clicking on the "Value" column and entering a value. A fixed value or a WinStudio variable can be entered.

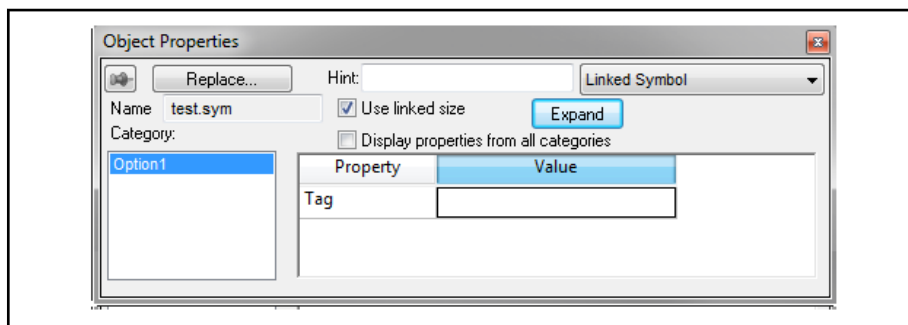


Fig. 3-16: "Parameterization" dialog of a WinStudio symbol

User Symbols

Customized symbols can be created in addition to predefined WinStudio symbols. If an object is to be used multiple times in a WinStudio project, creating a new symbol can be really helpful. If a symbol is changed, the changes are applied to all WinStudio screens in which the symbol is used. The symbol does not have to be inserted again into the WinStudio screens.

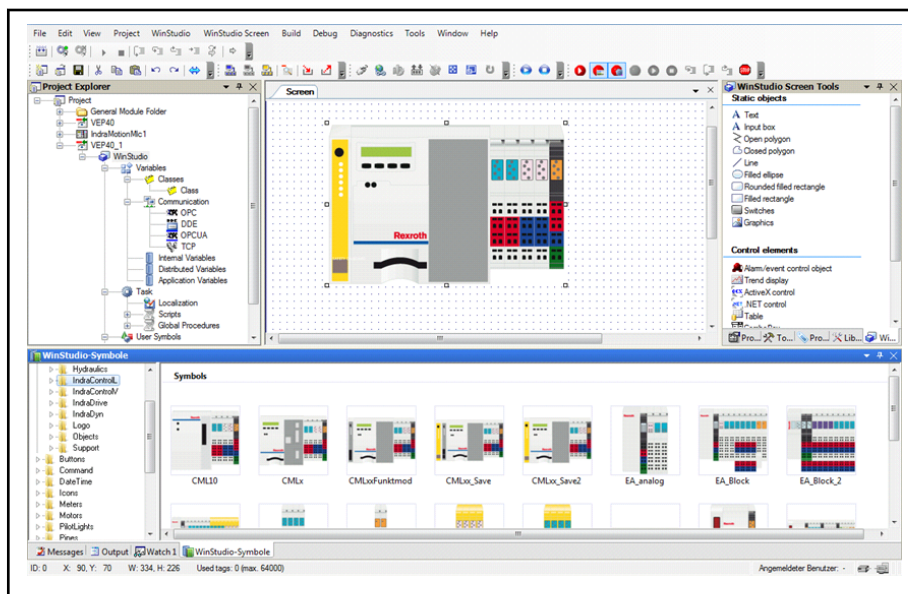


Fig. 3-17: IndraWorks WinStudio with opened symbol window

Creating a symbol

To create a symbol, insert one or more objects into a WinStudio screen, highlight them and click on the context menu item **Group**. Click on the context menu item **Create Linked Symbol** and enter a name in the subsequent file dialog.

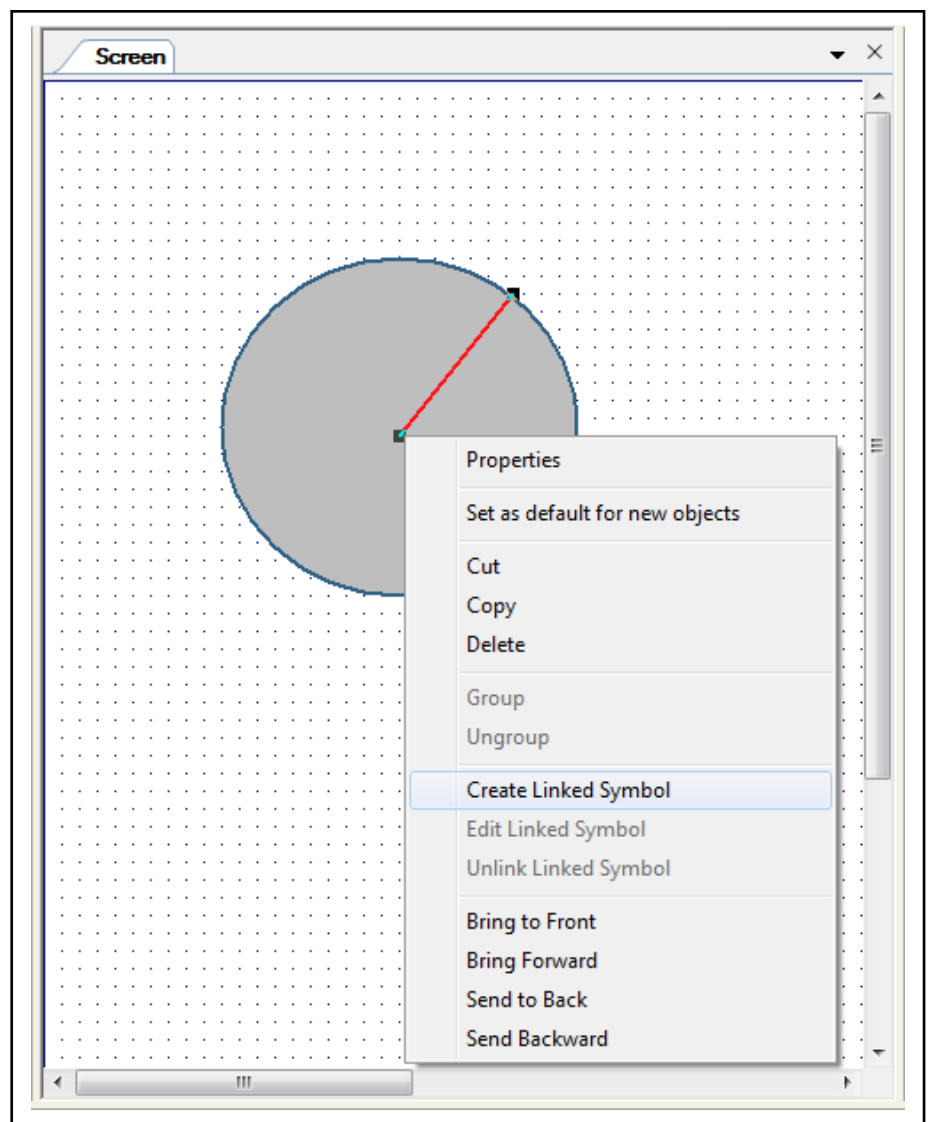


Fig. 3-18: Creating a user symbol

Apply symbol to library

A created WinStudio symbol is initially only available in the same project. To use it in other projects, transfer it to the WinStudio library by clicking on the element in the project tree under **WinStudio ► User Symbols** and press **Apply to WinStudio Library**. The symbol is copied to the "User Objects" folder in the WinStudio library.

Protect symbol with password

Own user symbols can be password-protected. A protected user symbol can thus be used and parameterized in WinStudio screens, but the internal functionality is not visible.

When opening a protected user symbol, an input dialog is displayed prompting the password. The editor only opens if the password is correct. For more information on the password protection of user symbols, refer to the IndraWorks online help "Rexroth WinStudio 7.3 Technical Reference".

3.4.7 WinStudio Tasks

Mathematic Sheets

A WinStudio mathematic sheet is used to program sequences, variable assignments and to execute other WinStudio functions.

Visualizations

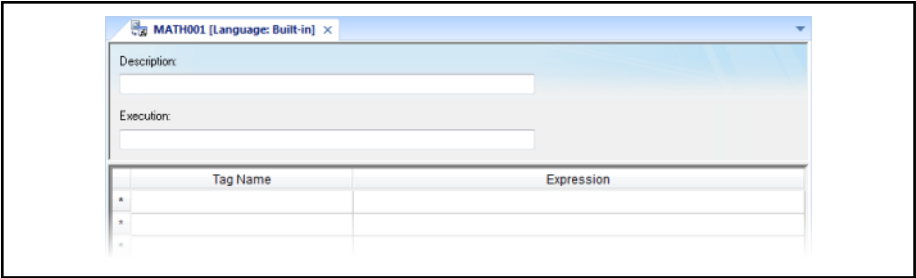


Fig. 3-19: WinStudio mathematic sheet

The mathematic sheet is executed cyclically if the variable or the expression in the "Execution" field is TRUE.

For more information on the mathematic sheets, refer to the IndraWorks on-line help "Rexroth WinStudio 7.3 Technical Reference".

Alarms

Using WinStudio Alarms, messages can be defined that are output if a variable exceeds or falls below a value.

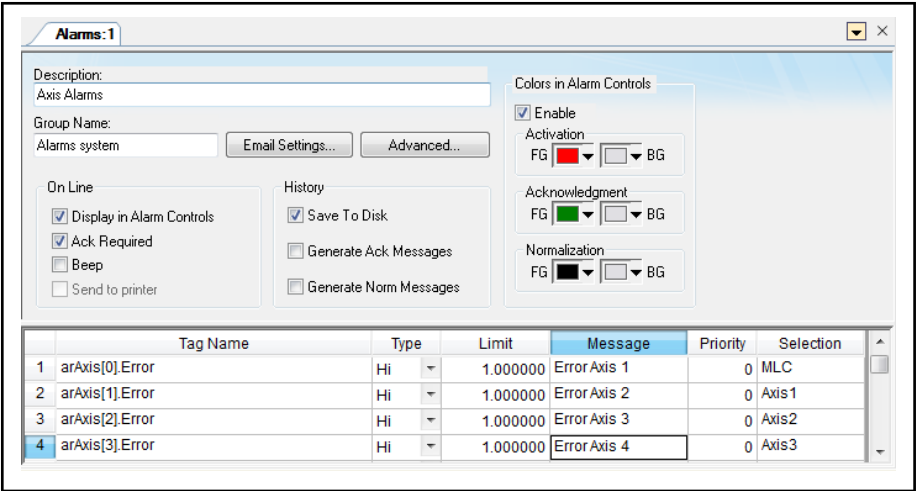


Fig. 3-20: WinStudio alarm editor

Alarm messages can be displayed using the alarm control element in WinStudio screens.

For more information on the alarms, refer to the IndraWorks online help "Rexroth WinStudio 7.3 Technical Reference".

Recipes

In WinStudio, variable values can be saved in files. Variable values can also be loaded from files using recipes.

A WinStudio recipe is a list of variables and the assignment of variables.

Visualizations

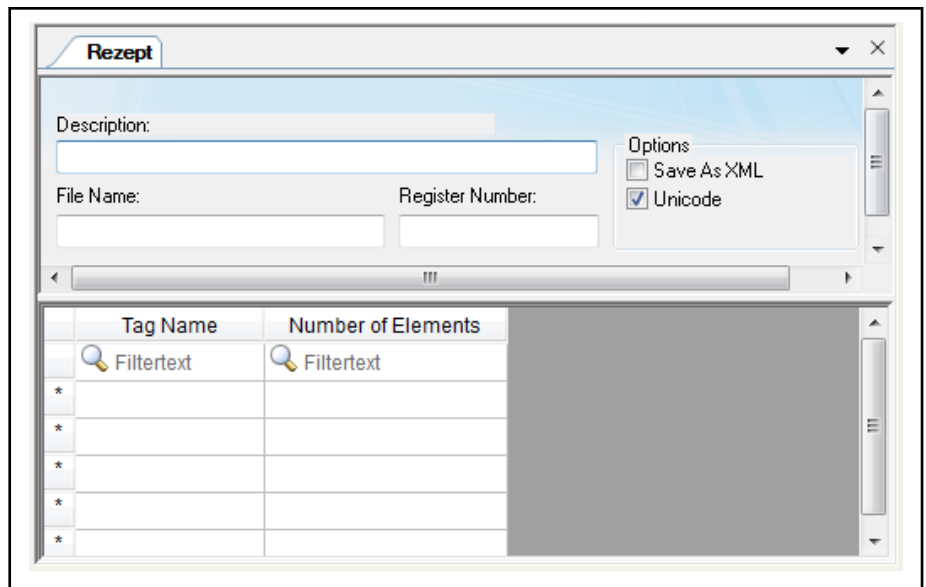


Fig. 3-21: WinStudio recipe editor

In the "Tag Name" column of the recipe editor, the names of variables, whose values are to be saved or loaded, can be entered. If a variable is an array, the number of elements has to be entered into the "Number of elements" column.

For more information on the recipes, refer to the IndraWorks online help "Rexroth WinStudio 7.3 Technical Reference".

Reports

A WinStudio report creates a documentation that can contain any text as well as current WinStudio variable values. The report can be saved in a file or printed.

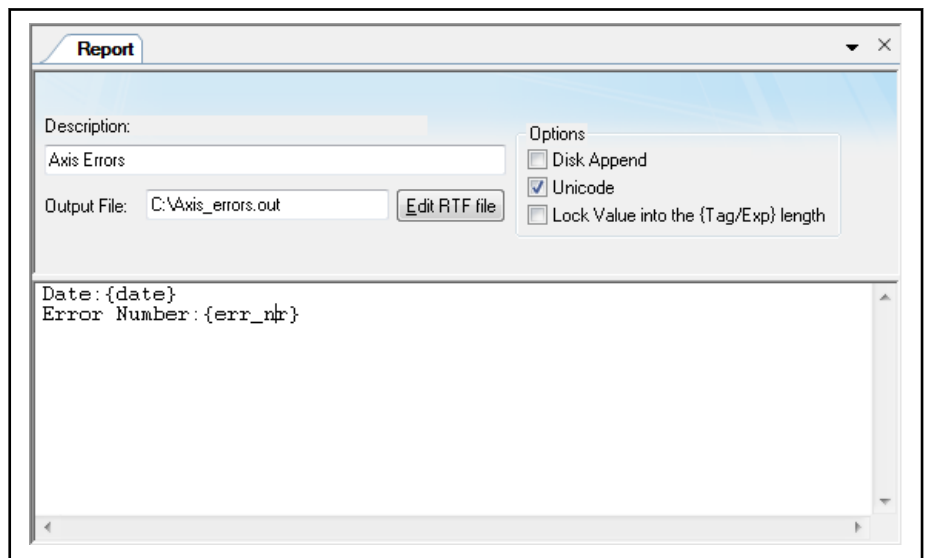


Fig. 3-22: WinStudio report editor

For more information, refer to the IndraWorks online help "Rexroth WinStudio 7.3 Technical Reference".

Visualizations

Trends

WinStudio trend data records WinStudio variables. These variables can be represented in a WinStudio screen using a trend object to monitor the chronological sequence of specified variables.

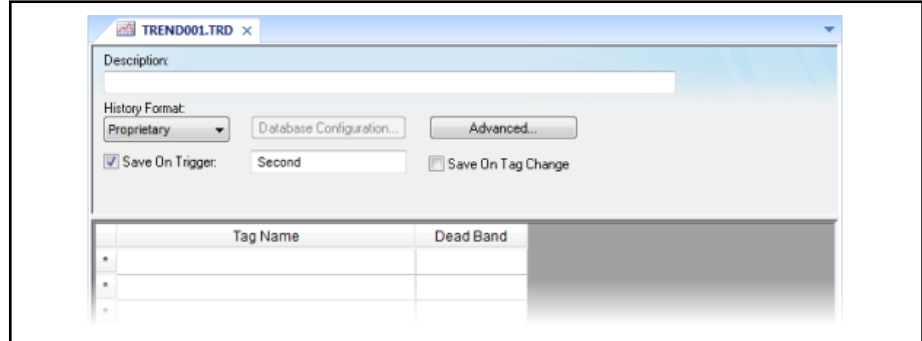


Fig. 3-23: WinStudio Trend Editor

For details on the configuration of trend data objects and to visualize trend data, refer to the WinStudio online help "WinStudio 7.3 Technical Reference".

Scheduler

Actions can be executed either time or event-triggered using WinStudio Schedules. For example, it is possible to automatically generate a report each day or to initialize a variable again every 30 seconds.

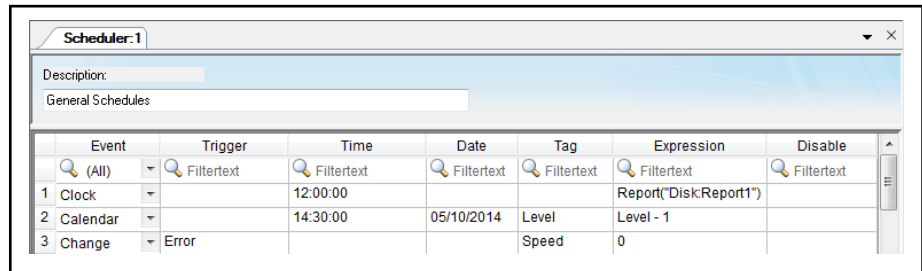


Fig. 3-24: WinStudio Schedule Editor

For a detailed description of how to work with the scheduler, refer to the IndraWorks online help "WinStudio 7.3 Technical Reference".

External Databases

Use External Databases to link historic data, created via alarms, events or trends for example, with external databases. Multiple database interfaces are supported (ODBC, Microsoft SQL Server, Microsoft Access, etc.)

For a detailed description about the configuration of such an interface, refer to the IndraWorks online help "WinStudio 7.3 Technical Reference".

Scripts

Overview

WinStudio supports the creation and execution of user-defined functions in VisualBasic scripts. The VisualBasic script language is a simple and flexible tool to program functions and sequences. An introduction to VB (VisualBasic) and a language reference can be found in the MSDN (Microsoft Developers Network).

For more information on VB scripts and the VB script language, refer to the IndraWorks online help "WinStudio 7.3 Technical Reference".

Global Procedures

Some VB functions can be declared as "Global Procedure" and a function library is created. Functions declared under "Global Procedures" can be opened in all WinStudio objects with VB support (e.g. as command on buttons).

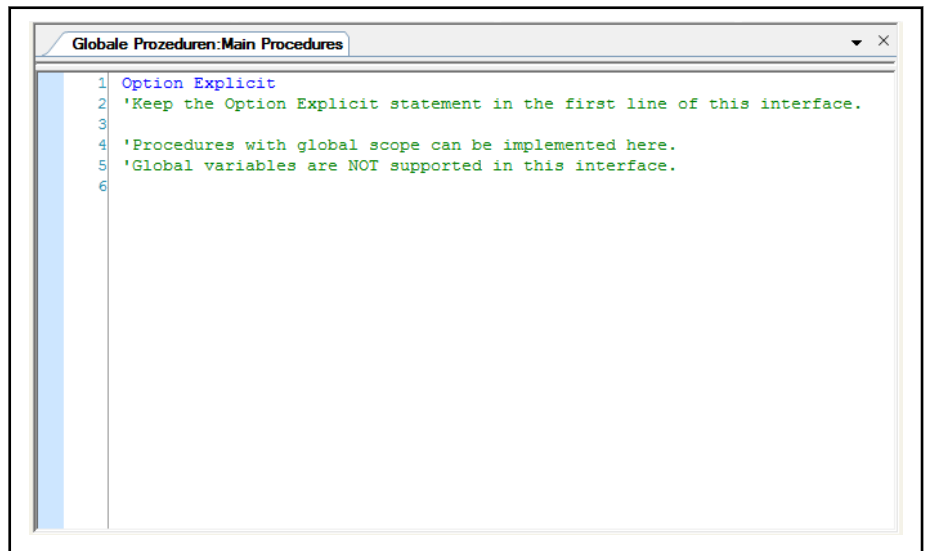


Fig. 3-25: Dialog: Global Procedures (editor)

Any number of functions and subfunctions can be declared in the editor for global procedures.

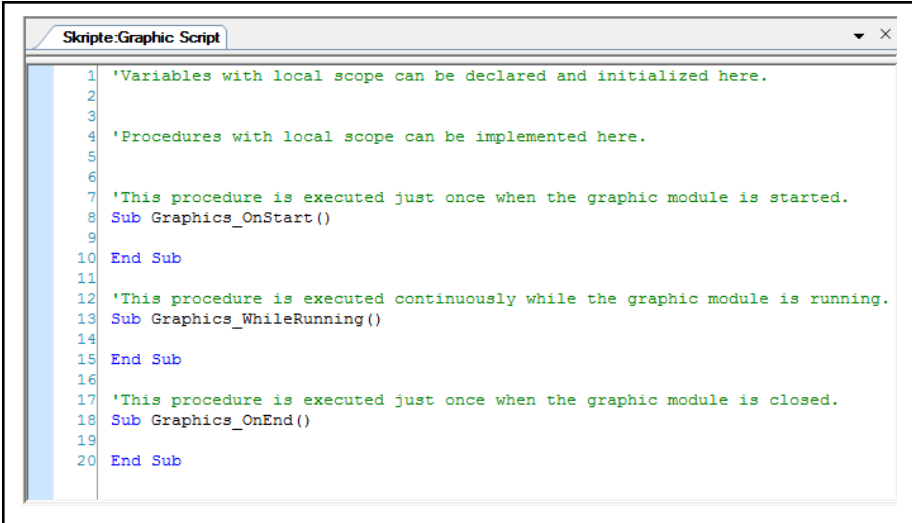
Start Script

In the "Start script" element, VB functions and user-defined global procedures can be called. These are to be executed once upon application startup. The start script is particularly suitable when initializing variables.

Graphic Script

The VB script "Graphic script" is used for a one-time execution of VB statements when starting or closing the WinStudio graphic module or for continuous execution of the graphic module at runtime. The graphic script already contains all required subfunctions that can be inserted into the own VB code. To correctly execute functions in the graphic script, do not change the function names. More functions, which can be called in the predefined graphic functions, can be added.

Visualizations



```

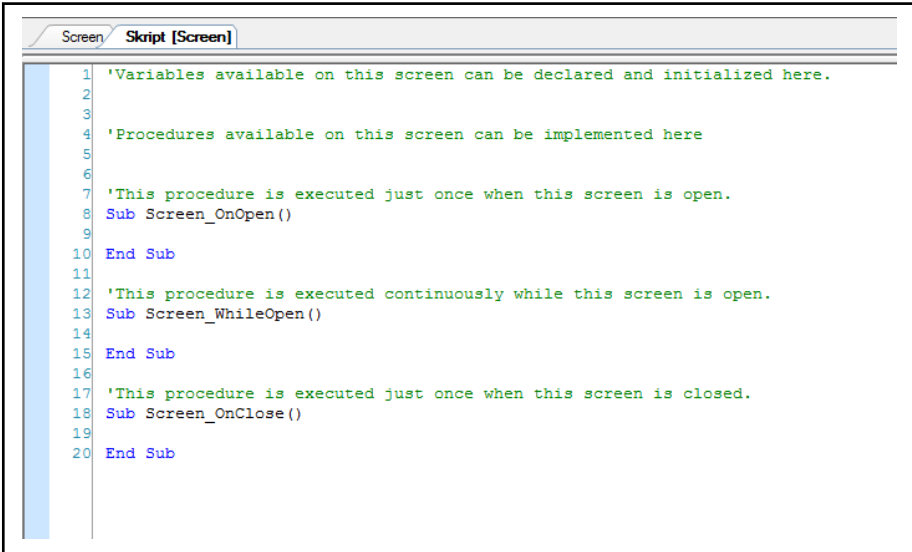
1 'Variables with local scope can be declared and initialized here.
2
3
4 'Procedures with local scope can be implemented here.
5
6
7 'This procedure is executed just once when the graphic module is started.
8 Sub Graphics_OnStart()
9
10 End Sub
11
12 'This procedure is executed continuously while the graphic module is running.
13 Sub Graphics_WhileRunning()
14
15 End Sub
16
17 'This procedure is executed just once when the graphic module is closed.
18 Sub Graphics_OnEnd()
19
20 End Sub

```

Fig. 3-26: Dialog: Graphic script (editor)

Screen Scripts

As in case of the graphic script, VB instructions can be programmed for each WinStudio screen. These instructions are either executed once when opening or closing the screen or continuously while the WinStudio screen is open. The screen scripts already contain all required function bodies. An own VB code can be inserted into those function bodies. The names of the predefined functions may not be changed.



```

1 'Variables available on this screen can be declared and initialized here.
2
3
4 'Procedures available on this screen can be implemented here
5
6
7 'This procedure is executed just once when this screen is open.
8 Sub Screen_OnOpen()
9
10 End Sub
11
12 'This procedure is executed continuously while this screen is open.
13 Sub Screen_WhileOpen()
14
15 End Sub
16
17 'This procedure is executed just once when this screen is closed.
18 Sub Screen_OnClose()
19
20 End Sub

```

Fig. 3-27: Dialog: Screen script (editor)

There is a VB script for each WinStudio screen. Open and edit the VB script in the context menu of the opened screen editor via **Screen Script**.

User Scripts

In addition to the predefined VB scripts ("start scripts", "graphic script" and "screen script"), any number of VB scripts, into which VB instructions can be inserted, can be added under the "Scripts" node. Functions cannot be declared in user scripts.

User scripts contain a continuous number that specifies the execution sequence of the user scripts. To exclude a script from the execution at runtime, enter a WinStudio variable or expression into the "Execution" field. In this

case, the script is only executed if the variable or the expression has a value greater than zero.

3.4.8 WinStudio Tools

Verifying Application

Variables required for the respective function can be used in all WinStudio elements. During the configuration, inconsistencies can occur by deleting or re-naming variables so that the affected WinStudio functions and elements do not function correctly anymore.

Use the function "Verify Application" to check whether variable names are used in WinStudio elements (mathematics sheets, WinStudio screens, driver sheets, scripts, etc.) that are not in the variable database or if variables were declared in the variable database not used in WinStudio elements.

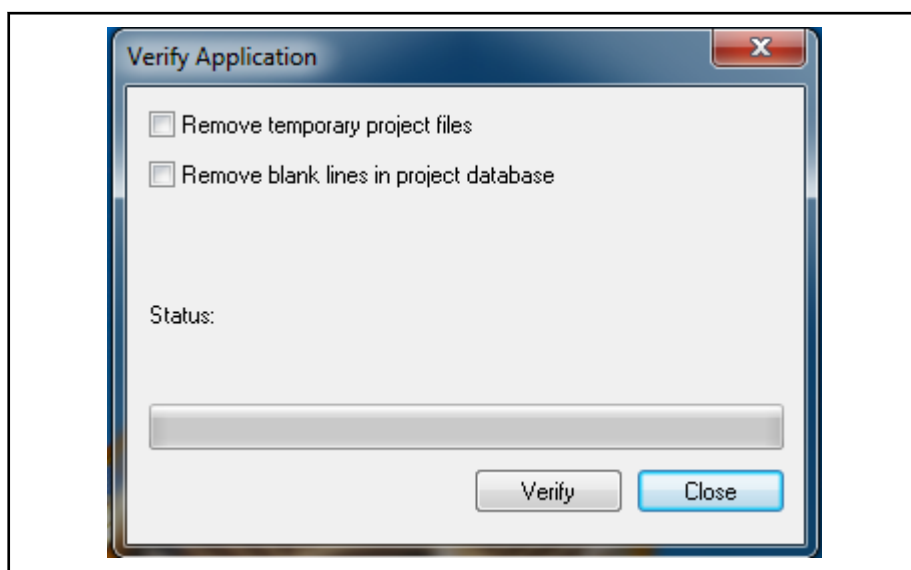


Fig. 3-28: Verify application

Information, warnings and error messages are output in the message window of IndraWorks Engineering after the verification dialog is closed.

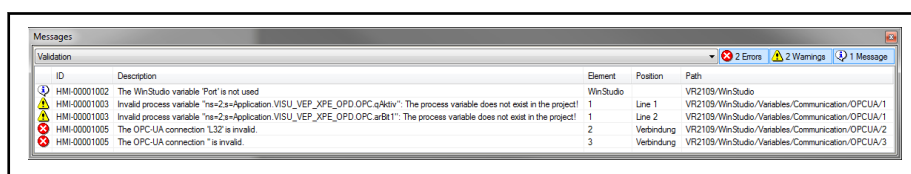


Fig. 3-29: Outputting messages of the "Verify Application"

Detailed information on each message in output line by line. Double-click on the message line to open the corresponding tool to correct the message cause. Thus, all inconsistencies can successively be corrected and the function "Verify Application" can be restarted if required.

Starting Application

The function of individual WinStudio screens can be checked using the function "Start Application" before transferring the entire package to the target device and enabling it. Open the test window via **WinStudio ► Start Application**.

Visualizations

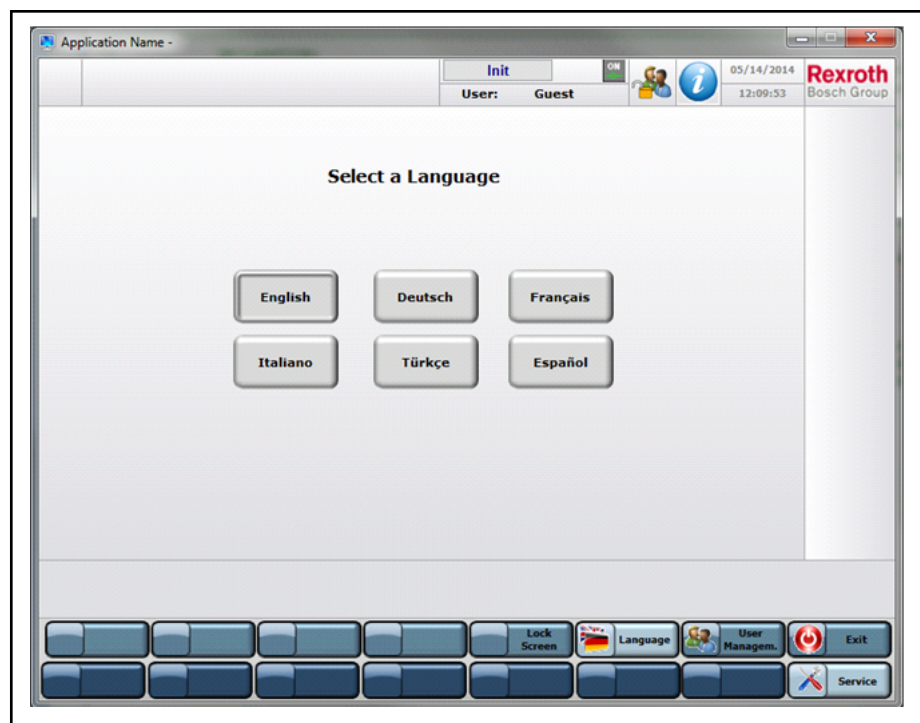


Fig. 3-30: Testing WinStudio screens

After executing "Start Application", the WinStudio Viewer Task as well as corresponding background tasks open and the complete application (display of variables and images, commands on buttons) and the WinStudio Task execution can be tested. Close the application via "Close Application", as closing the Viewer Task background processes does not close the application.

3.4.9 Database Spy

Using WinStudio database spy, value and status of individual WinStudio variables of an active WinStudio application can be monitored. In the input field of the database spy, enter the WinStudio variable name and apply it to the list of variables to be monitored via **Add**. The current value as well as the connection status are displayed for each variable in the variable list.

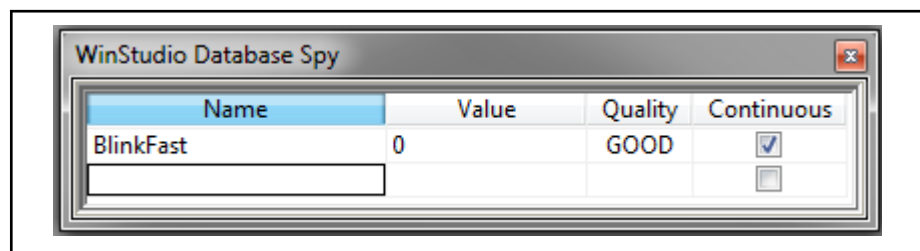


Fig. 3-31: WinStudio database spy

Two variants of database spies can be used:

1. **Local database spy:** To access a locally running WinStudio application
2. **Remote database spy:** To access a WinStudio application within the network

3.4.10 LogWin

Use WinStudio LogWin to analyze sequences in a WinStudio application and to troubleshoot.

Visualizations

In LogWin, communication connection messages (e.g. BRWS, OPC), individual trace messages and the recipe execution can be logged.

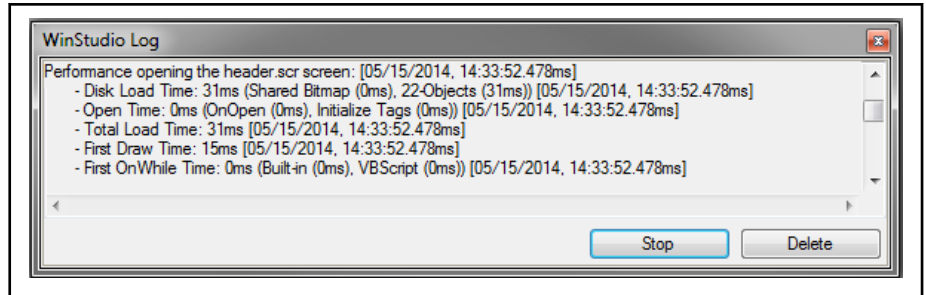


Fig. 3-32: Monitoring WinStudio variables in LogWin

In the context menu of the LogWin window, open a configuration dialog to select messages to be recorded via **Settings**. Furthermore, WinStudio variables, whose modification result in a message outputting the current value, can be selected.

For more information on the LogWin use, refer to the IndraWorks online help "Rexroth WinStudio 7.3 Technical Reference".

3.4.11 Debugger

For process analysis as well as for troubleshooting in the "Logic" script of a WinStudio application, the WinStudio debugger is a useful tool. Global procedures, graphic, start, user and screen scripts are supported. To start a debug session, open the respective script in the script editor and right-click to set a breakpoint at the left, light blue edge of the script window. Close the editor and open **WinStudio ► Debug Application** in the project node. Use the WinStudio debug bar under **View ► Toolbars ► Debug** to debug the running program.

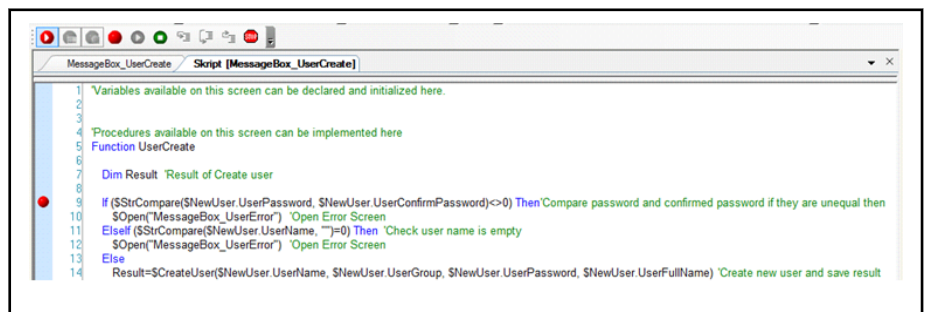


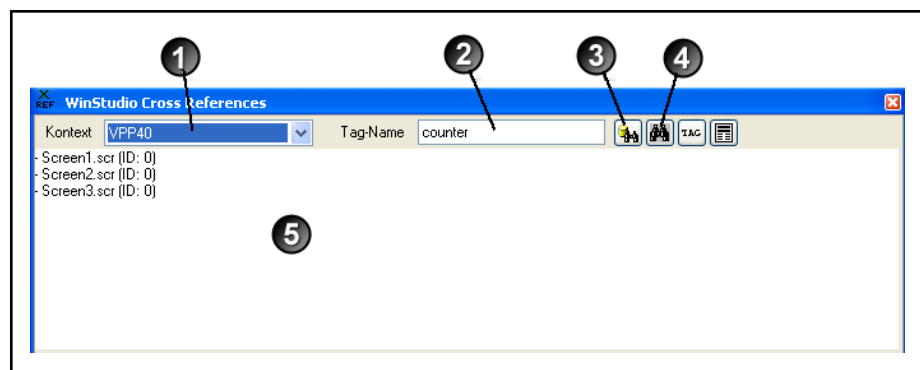
Fig. 3-33: WinStudio "Debug" functionality

For more information on the debugger, refer to the IndraWorks online help "Rexroth WinStudio 7.3 Technical Reference".

3.4.12 Cross Reference Search

Apply the cross reference search to find the usage of a certain WinStudio variable.

Visualizations



- 1 Search environment
- 2 Variable
- 3 Search in file
- 4 Search in project
- 5 Display

Fig. 3-34: Cross reference search of WinStudio variables

3.4.13 Variable Import Wizard

It is often useful to apply variable lists from another WinStudio application or a database to avoid that they have to be configured manually again. Applying variables is supported by the import wizard.

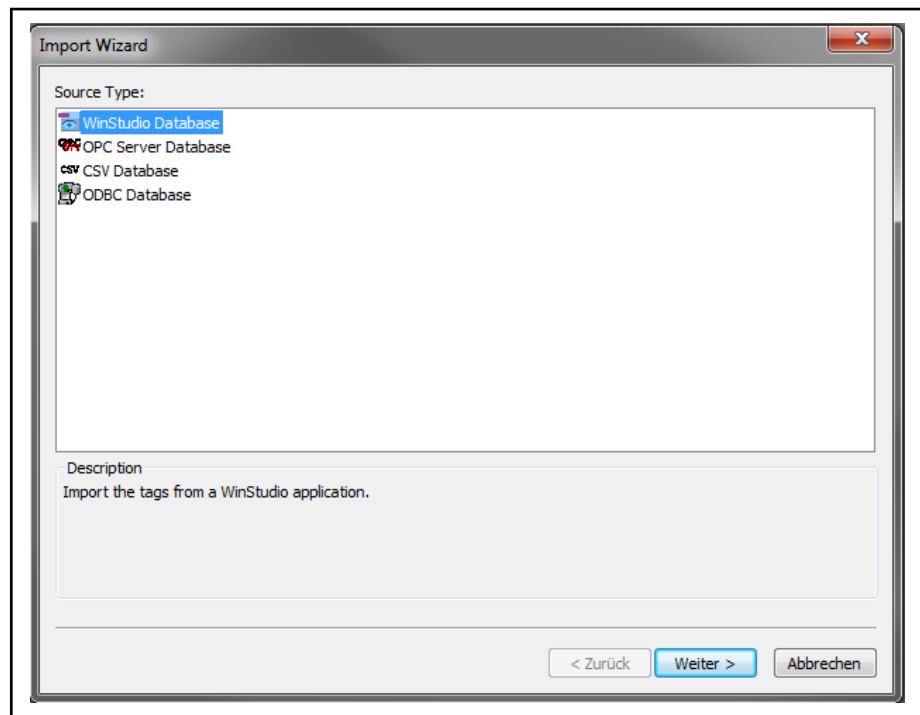


Fig. 3-35: Selecting a data source in the WinStudio import wizard

3.4.14 Export/Import

The "WinStudio" folder as well as all corresponding subelements support the export/import functionality in IndraWorks. This functionality allows to export the entire WinStudio project or only parts of it and to import them into another IndraWorks project.

Please note the following characteristics when using the export/import function on WinStudio elements:

- The WinStudio variables are determined and automatically exported for each WinStudio element. When importing this WinStudio element, the required WinStudio variables are inserted again. If the variable to be imported already exists in the new project, all corresponding variable properties (e.g. description text, initial values) are overwritten. If the variable type differs, the variable remains unchanged. To still import the variables, change the variable type or rename the variable and repeat the import.

If an existing variable was overwritten during import or if the new variable could not be imported, the import dialog displays a corresponding message.

- If the WinStudio variable to be exported is a user-defined WinStudio data class, the corresponding class definition has to be exported as well to create the variable correctly.

3.4.15 WinStudio Licensing

Introduction

The graded license model permits an optimized adjustment to the required number of variables and the required functional scope. The larger license package covers the functional scope of the smaller package. An upgrade to the next higher license class is possible.

The WinStudio licensing procedure changed with the introduction of WinStudio 7.x (from IndraWorks 09VRS). In the following, please find a description of the new license management.

WinStudio License Overview

The following licenses are available:

- **WinStudio Lite**

This license is part of the IndraWorks installation. The user can create and execute WinStudio projects with a maximum of 500 variables (tags). Moreover, all CE devices are equipped with a Lite license to execute Lite runtime projects. With this license, alarms, recipes, reports, trends as well as TCP/IP and DDE driver sheets are not executed and cannot be created or edited.

- **WinStudio Runtime 1K5 / 4K / 64K / 512K (RUN licenses)**

For WinStudio runtime projects, licenses with up to 1.500, 4.000, 64.000 and 512.000 WinStudio variables (tags) are available

- **WinStudio Engineering and Runtime 1K5 / 4K / 64K / 512K (RUD licenses)**

With this license, WinStudio projects with up to 1.500, 4.000, 64.000 or 512.000 WinStudio variables (tags) can be created and executed

- **WinStudio demo license**

With this license, the WinStudio functional scope can be tested within 30 days. After the test period, the last valid license (e.g. Lite license) is active

The demo license can only be activated once.



Only Lite, 1K5 and 4K runtime licenses are available for the activation on an IndraControl VEx device.

Visualizations

Activating a WinStudio License

The individual "WinStudio Runtime" and "WinStudio Engineering and Runtime" licenses are subject to additional costs and have to be activated by entering a license key. The license sheet contains your personal license key.

To activate the license, open the licensing dialog via **Tools ► Options ► General ► Software Licenses** and select "WinStudio". Open the activation dialog via **Licensing...** and enter the activation key.

After license activation, WinStudio Engineering or IndraWorks Operation might have to be restarted.

Saving license on dongle

The activation key is only valid for one PC. To use a license on several PCs, save it on a dongle. The license is valid as long as the dongle is plugged into a USB interface of the computer. Windows CE licenses (VEx) cannot be transmitted to a dongle.

License activation on VEx devices

To activate the WinStudio runtime license on a visualization device under Windows XP (embedded)/ Windows 7(embedded), go to the Windows start menu **All Programs ► Rexroth ► IndraWorks ► Common Tools ► License Manager** on the device, open the licensing dialog and enter the activation key. The licensing software is already installed on new VEx devices. If you have a VEx device without pre-installed licensing software, install the IndraWorks visualization software on the VEx device and enable the WinStudio license.

For a VEx device under Windows CE, activate the license manager via the Desktop shortcut "LicenseImport.CE".

When using a VRxx device, use remote licensing in the IndraWorks Engineering environment. Go to the context menu item **Licensing** of the respective device node. Note that you have to be connected to the device.



The license activation via the WinStudio remote agent is not possible anymore with the new licensing procedure in WinStudio 7.x. Use the IndraWorks licensing dialog to activate the license (as described above).



The license key cannot be transmitted anymore via the integrated WinStudio due to the new licensing procedure in WinStudio 7.x. The license key has to be directly entered at the VEx device.

Consequences of a Missing or a Lower Class License

When opening a project with the WinStudio that was editor created on another computer with a higher class WinStudio license, the elements subject to license in WinStudio Engineering are not activated. Thus, these elements can neither created nor edited.

When starting IndraWorks Operation, a message displays that the available WinStudio license is insufficient for the current project. No WinStudio screens are displayed in case of an insufficient or missing license.



If you still want to use the current project, activate the demo license. Use the IndraWorks licensing dialog to activate the license (as described above).

3.4.16 Multilingualism

WinStudio user screens can have multiple languages. All texts in the WinStudio screen are displayed in the selected project language at runtime.

Create a translation file via the node **WinStudio ► Tasks ► Localisation**. A new column is created for each target language (multi-language function).

For more information on the WinStudio multi-language function, refer to the IndraWorks online help "Rexroth WinStudio 7.3 Technical Reference".

3.4.17 WinStudio Applications

Overview

In IndraWorks, applications can be created that are exclusively based on the WinStudio runtime environment. This is required when using the WinStudio web functionality. Creating and enabling WinStudio applications differ from the previously described application within the IndraWorks runtime environment. These differences are explained in the following.

Creating WinStudio Application

To create a WinStudio application, insert a VPP or VEx device from the device library to an IndraWorks project. Select "WinStudio Application" in the HMI wizard. Confirming the entries with **Complete** and the WinStudio elements are created in the project tree. Elements such as F-, M- and OP-panels and operating screens requiring an IndraWorks runtime environment cannot be selected.

User Management

The WinStudio user management can protect different WinStudio functions in WinStudio applications. Configure the WinStudio user management via **WinStudio ► Tools ► Security Settings**. Enable the safety system in the configuration dialog. Groups and users can be created and rights can be assigned.

For more information on how to use the WinStudio user management, refer to the WinStudio online help "Rexroth WinStudio 7.3 Technical Reference".



Users and groups configured in the IndraWorks user management are not available in the WinStudio user management. If required, create them again in the WinStudio user management.

Web Functionality

WinStudio applications can be visualized using a web browser at runtime (web application). Applications have to be saved in HTML format.

To save a WinStudio screen in HTML format, open the context menu of the WinStudio screen and execute the function "Save as HTML". Optionally, the function "Save all documents as HTML" saves all existing WinStudio screens in HTML format.

After the configuration and saving of all required WinStudio screens in HTML format, configure the web server. Open the WinStudio project settings and select the "Web" tab.

For information on the configuration and use of the WinStudio web server, refer to the WinStudio online help "Rexroth WinStudio 7.3 Technical Reference".

Visualizations



The functions "Save as HTML" and "Save all documents as HTML" as well as the "Web" tab in the WinStudio project settings are only available for a "1,5 k" WinStudio license or higher.

Transferring Application and Activation on Target Device

To transfer a WinStudio application to a target device, start the WinStudio remote agent on the target device.

For devices under Windows CE / Windows 7 embedded compact, the short-cut "WinStudio"CE-Server" to start the remote agent is located on the desktop. For devices under Windows 7 embedded / Windows XP embedded, start the remote agent via **Start ▶ All Programs ▶ Rexroth ▶ IndraWorks ▶ WinStudio ▶ Remote Agent**

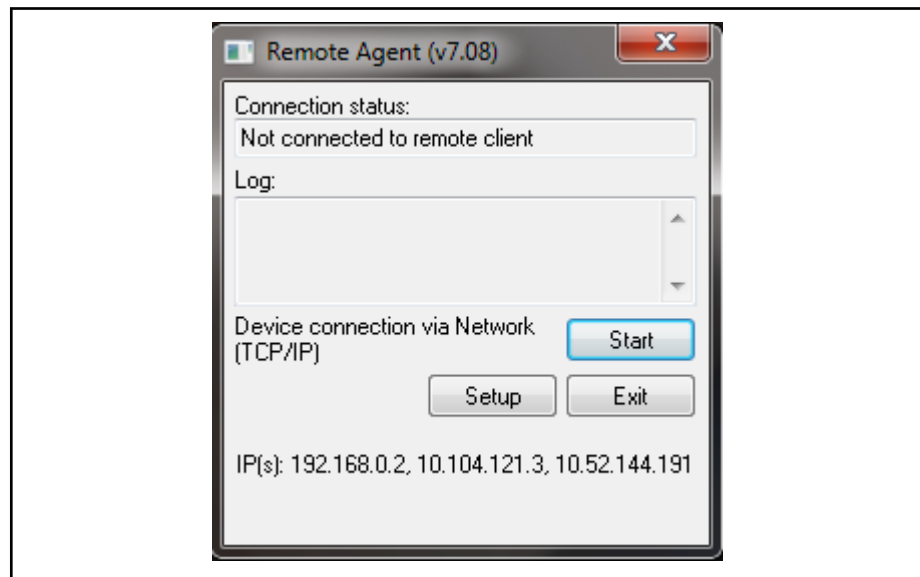


Fig. 3-36: WinStudio remote agent on target device

Open the Management Remote dialog in IndraWorks via the context menu of the WinStudio node **WinStudio Target Platform**. Select the target device to connect to in the "Target" tab in this dialog. When setting the target station "Local host", connect to the remote agent started on the local PC. To connect to the remote agent running on a device in the network, enter the device IP address. Click on **Connect** to connect to the target device.

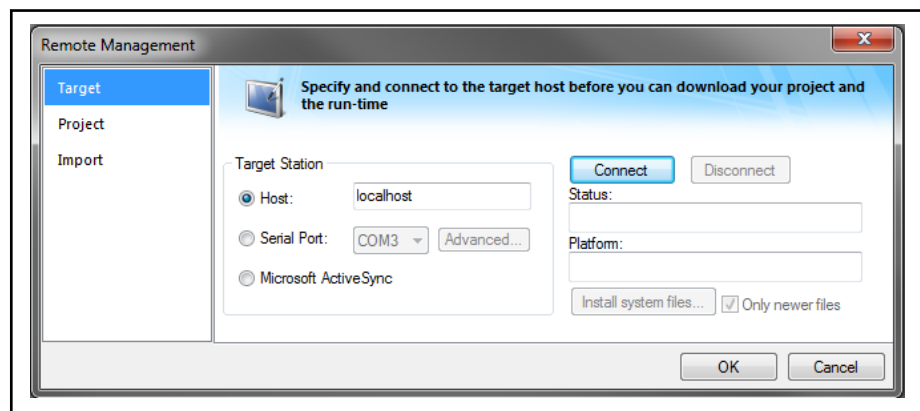


Fig. 3-37: WinStudio Remote Management

Select the target path the project files should be transferred to the target device in the "Project" tab.



Ensure that the connection is established before selecting the tab. Then, the target path is provided by the system.

Note that the target directory is deleted before transferring project files. Select the checkbox "Keep user files" to prevent this.

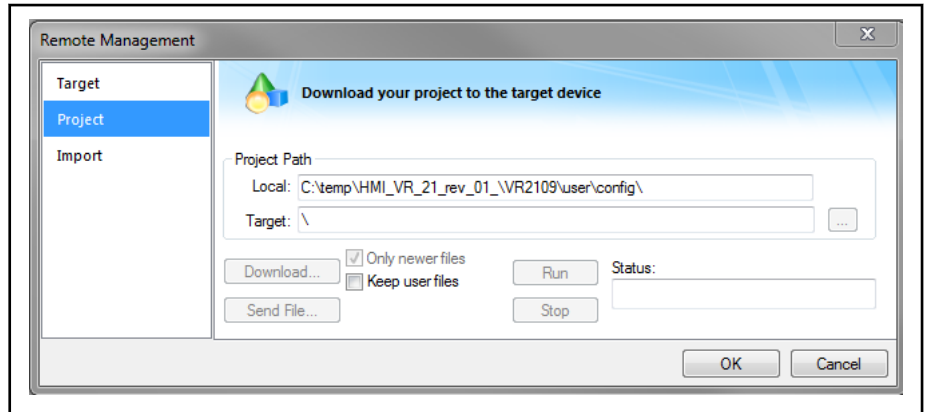


Fig. 3-38: WS Remote Management "Project" tab

Press **Download** to transfer the project files. After transfer, start and stop the project with the **Start** and **Stop** buttons

3.4.18 More Information

WinStudio provides comprehensive functions to create user screens. For the complete functional scope, refer to the IndraWorks online help WinStudio 7.3 Technical Reference.

4 Communication Interfaces

4.1 Overview

The following table provides an overview on the possible combination of visualization devices, visualizations and communication protocols.

Device and visualization	Communication protocol					
	OPC.DA	OPC.UA	OPC.lwSCP	BRWS	ModbusTCP	IndraLogic Gateway
VCP/VCH						
VI-Composer				x		
VEP/VEH (WinXP/Win7)						
WinStudio	X	X	X*	X		
Extern. visu	X*	X**	X*		X	
VEP/VEH (WinCE)						
WinStudio	X	X		X		
Extern. visu					X	
VPx						
WinStudio	X*	X	X*	X		
VI-Logic						X
Extern. visu	X*	X**	X*		X	
VR21						
WinStudio		X		X		

* Available after installing the installation variant "Communication - IndraLogic, IndraMotion MLC" by the IndraWorks installer

** If visualization software is provided with an OPC UA client

Tab. 4-1: Overview on the possible visualization combinations

The controls of the XLC/MLC family always allow the following number of communication channels:

- XLC/MLC L25: 4
- XLC/MLC L45: 8
- XLC/MLC L65: 16

4.2 BRWS Driver (Bosch-Rexroth WinStudio Driver)

4.2.1 Properties

For WinStudio version 6.2 and higher, the BRWS driver is integrated and allows to access variables and parameters of the XLC/MLC systems on Ethernet basis for PC-based and embedded PC-based visualization devices.

The driver uses the options of the communication server.

The BRWS protocol only uses synchronous data services. The client sends a request and the server immediately sends a response.

Communication Interfaces

4.2.2 Areas of Application

4.2.3 Configuration

The BRWS driver is configured in the WinStudio engineering environment.

Open the context menu of the WinStudio communication node in the IndraWorks Project Explorer and select **Insert Driver**. The driver (BRWS connection to communication server) can be selected in the dialog of the communication driver. The driver is then displayed in the lower dialog window.

Confirm with **OK**, and a new folder "BRWS" is created in the WinStudio workspace. The driver can be configured and new driver sheets can be created.

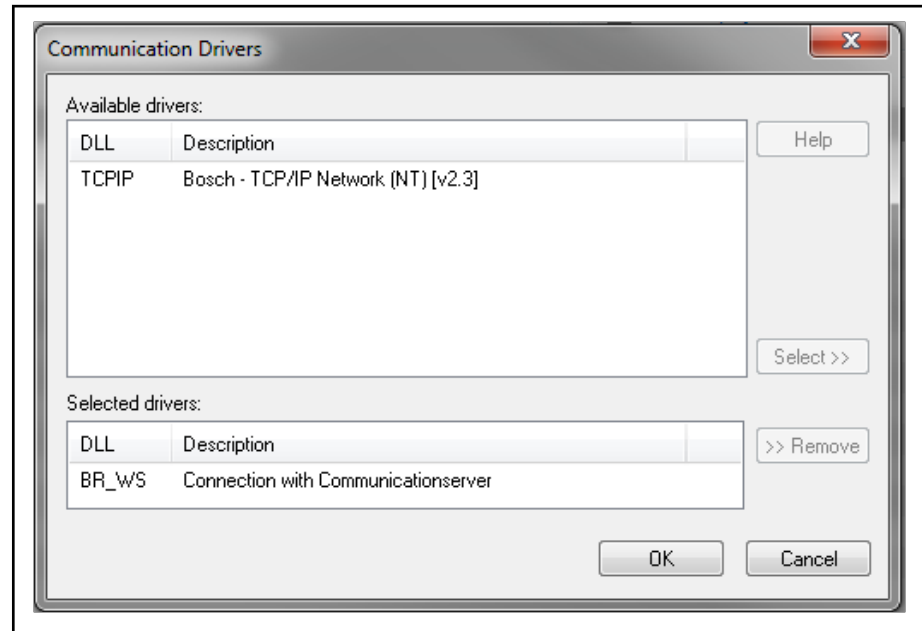


Fig. 4-1: Driver selection

The TCP/IP address of the target control is specified in the driver settings dialog. Select "None" as "Serial Encapsulation".

A timeout can be specified in ms in the "Timeout in ms" field. The set time is used for the time monitoring of communication connections. The timeout value is only significant if the visualization is connected to two or more controls and if one control fails or cannot be reached.

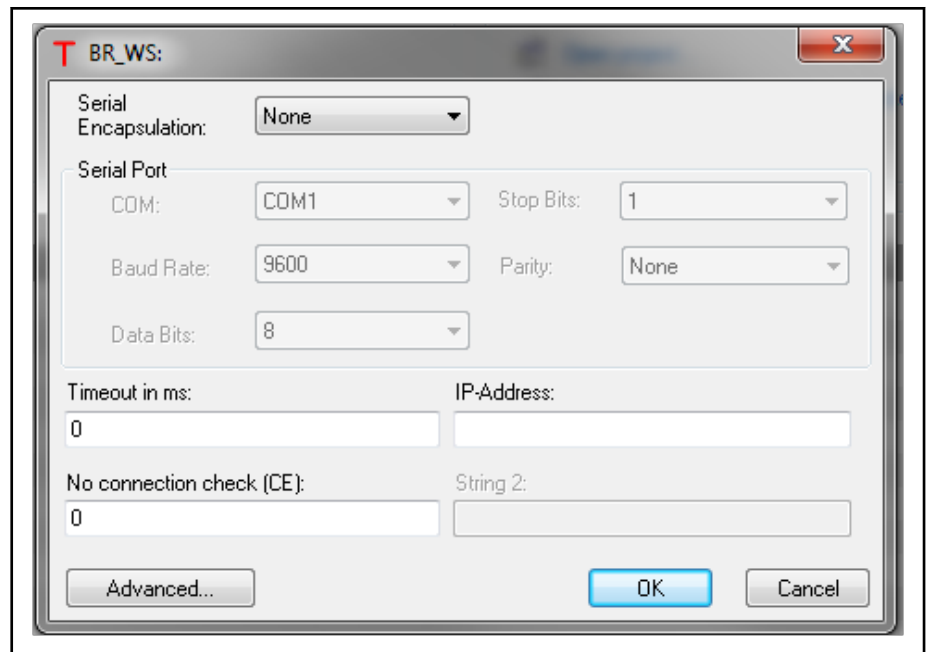


Fig. 4-2: Driver properties



In case of slow LAN connections, short network failures can occur in XP/Win7 devices. These failures are logged with error number 1005. In this case, increase the timeout to 300 - 500 ms!

If multiple embedded PC-based (CE6.0) visualization devices communicate with the same control and short network failures (1005) occur, disable the connection check to relieve the communication channel.

Recommendation: If projects with large symbol tables are loaded to the control, e.g. during the configuration phase via "Online Change", a timeout of at least 1000 to 2000 milliseconds is recommended!

4.2.4 Driver Sheets

Data to be communicated between the control and the HMI are specified in the WinStudio driver sheets. It can also be defined, how data is to be communicated.

Right-click on the BRWS folder to create new driver sheets. They are displayed as elements below the "BRWS" node and can also be opened and edited there.

The driver sheets are divided into two parts:

- **Header:**
It is specified in the header at which points in time and with which priority it is to be read or written.
- **List area:**
Defines the relation of the variables in the application to the variables in the control.

Communication Interfaces

Fig. 4-3: BRWS driver sheet

Header of driver sheets:

- **Description**
Describes driver sheet for documentation purposes.
- **Checkbox: Increase priority**
Increases the priority for reading and writing driver sheet variables.
- **Read Trigger**
Driver sheet variables are read when changing the entered tag.
- **Enable Read when Idle**
If a constant or a tag unequal to zero is entered, the driver sheet variables are read continuously.
- **Read Completed**
The entered tag is set if the read command is completed.
- **Read Status**
The status of the last read command is applied to the entered tag.
- **Write Trigger**
Driver sheet variables are written whenever changing the entered tag.
- **Enable Write on Tag Change**
If a constant or a tag unequal to zero is entered, the driver sheet variables are continuously checked for changes. Upon a change, the driver sheet variables are written.
- **Write Completed**
The entered tag is set if the write command is completed.
- **Write Status**
The status of the last write command is applied to the entered tag.
- **Station**
Provides extended driver options to communicate to other controls.
If the field is left blank, the IP address of the target control previously set for the driver configuration (see [Driver properties, page 81](#)) is used to read or write data.
If the IP address of a control is entered into the station, it is communicated with this control. The IP address can be changed at runtime. This

means that the currently entered IP address is always selected. The selected controls have to have loaded the symbols used and should also be accessible physically via the TCP/IP. Otherwise, no communication can be established.

It is thus possible to use the selected visualization on different controls without adjusting the project.

- **Header**

A unique ID has to be entered into the header.

The ID structure is irrelevant. Only numbers and characters can be used. The ID has to be unique, i.e. an ID may only be assigned once in a project. If the number of the driver sheet is used in the ID, no duplicate IDs can be assigned.



Avoid simultaneous "Enable Read when Idle" and "Enable Write on Tag Change" if possible! Read variables via "Read Trigger".

List area of the driver sheets:

- **Variable name**

Enter the WinStudio variables to be written with the variable value from the "Address" column (reading) or whose value has to be written to the variable in the "Address" column (writing).

- **Address**

Enter control variable name and type.

The variable name consists of <program block name>, <variable name> and <type>

- **Example:** "Global variable"

.Glob_int;T_INT16

.Glob_write;T_UTF8STR

- **Example:** "Local variable in program FB "PLC_PRG""

PLC_PRG.Byt;T_INT8

PLC_PRG.Ind;T_INT16

4.2.5 Protocol Structure

The request length in UTF-8 format precedes each protocol. The request length is not identical to the data security layer (transport/network) if the request is fragmented the transport and/or several requests are sent simultaneously.

Header	Body
UTF-8 data length 0-7FFF FFFF (1-6 bytes)	Requests/responses

4.2.6 Conventions

The BRWS driver can communicate with up to 20 different controls (IP addresses) at the same time.

General driver sheet information:

- Max. 512 items per driver sheet
- No blank lines in the driver sheet
- All variables have to be in the control

Communication Interfaces

- Case sensitivity in the address fields has to be identical to the notation in the symbol file
- Max. 960 characters per address field

Information on EIS:

- The number of the handles available for communication is limited to 10,000 in case of EIS (DeviceXCC.xml or DevicePlc.cfg).
- The number of the required handles is composed as follows:
Each registered PLC variable requires a handle. Additionally, several group handles are registered for each driver sheet.
- The TCP window size in MLC BSP is set to 1460 bytes (1 telegram). This means that if the data volume of a driver sheet is greater than 1460 bytes, it has to be partitioned. Partitioning results in performance loss.

4.2.7 Supporting PLC Variables and Parameters

PLC variables

- Item syntax
 - 'POE_NAMESPACE'.PlcVariable';BRWS_DataType'
 Example: Application.PLCPROG.Value;T_INT16
 The "Value" variable from the "PLCPROG" POU of the namespace "Application" is addressed. It is of data type INT

PLC data type	BRWS data types	WinStudio
BOOL	T_BOOL	Boolean
BYTE	T_UINT8	Integer
WORD	T_UINT16	Integer
DWORD	T_UINT32	Integer
LWORD	T_UINT64	"not supported" Configure variable with BRWS data type "T_UTF8STR".
SINT	T_INT8	Integer
INT	T_INT16	Integer
DINT	T_INT32	Integer
LINT	T_INT64	"not supported" Configure variable with BRWS data type "T_UTF8STR".
USINT	T_UINT8	Integer
UINT	T_UINT16	Integer
UDINT	T_UINT32	Integer
TIME	T_UINT32	Integer
LTIME	T_UTF8STR	String
DATE	T_UINT32	Integer
DATE_AND_TIME	T_UINT32	Integer
TIME_OF_DAY	T_UINT32	Integer

Communication Interfaces

ULINT	T_UINT64	"not supported" Configure variable with BRWS data type "T_UTF8STR".
REAL	T_FLOAT32	Real
LREAL	T_FLOAT64	Real
STRING	T_UTF8STR	String
WSTRING	T_UTF8STR	String
TIME	T_UINT32	Integer
LTIME	T_UTF8STR	String
DATE	T_UINT32	Integer
DATE_AND_TIME	T_UINT32	Integer
TIME_OF_DAY	T_UINT32	Integer

Tab. 4-2: Assigning PLC data types



Conversion to a data type with a greater value range is possible.

E.g. "T_UINT8 ⇒ T_UINT16 ⇒ T_INT32 ⇒ T_UTF8STR"

XLC/MLC Parameters

- Item syntax

'CtrlNo','AxisNo','SercosParameter'. 'SercosElement'; 'BRWS_DataType'

Example: 0,15,A-0-0002.7;T_UTF8STR

The axis name is addressed with the axis number 15 (A-0-0001) using the axis parameter A-0-0002. The axis was created in the control with the control number 0 (C-0-0031). Element 7 (Data) of the parameter A-0-0002 is accessed. This element is a STRING variable.

BRWS item element	Value	Description	
CtrlNo	0	Local control	Optional
	1 - 999	Control address in the link	
AxisNo	0	Control parameters	
	1 - 999	Axis parameters	
SercosParameter	C	System parameters	
	K	Kinematics	
	M	Touch probe	
	N	Cam switch	
	O	Oscilloscope	
	A	Axis-specific parameter	
	S	Standard drive parameter	
	P	Product drive parameter	

Communication Interfaces

BRWS item element	Value	Description	
SercosElement	'Empty'	Data	
	1	sercos ID	
	2	Description	
	3	Attributes	
	4	Unit	
	5	Min. value	
	6	Max. value	
	7	Data	
	8	Data status (only drives)	

Tab. 4-3: BRWS item structure for parameters

Parameter type	PLC data type	BRWS data types	WinStudio
BINARY, 2BYTE	WORD	T_UINT16	Integer
UINT_DEC, 2BYTE	UINT	T_UINT16	Integer
INT_DEC, 2BYTE	INT	T_INT16	Integer
UINT_HEX, 2BYTE		T_UINT16	Integer
IDN, 2BYTE		T_UINT16	Integer
BINARY, 4BYTE	DWORD	T_UINT32	Integer
UINT_DEC, 4BYTE	UDINT	T_UINT32	Integer
INT_DEC, 4BYTE		T_INT32	Integer
UINT_HEX, 4BYTE	UDINT	T_UINT32	Integer
MLC_IDN, 4BYTE		T_UINT32	Integer
FLOAT, 4BYTE	REAL	T_FLOAT32	Real
TEXT, 1BYTE_VAR	STRING	T_UTF8STR	String
BINARY, 8BYTE	LWORD	T_UINT64	"not supported" Configure variable with BRWS data type "T_UTF8STR".
UINT_HEX, 8BYTE	ULINT	T_UINT64	"not supported" Configure variable with BRWS data type "T_UTF8STR".
DOUBLE, 8BYTE	LREAL	T_FLOAT64	Real
MLC_IDN, 8BYTE	MB_IDN (LINT)	T_UINT64	"not supported" Configure variable with BRWS data type "T_UTF8STR".

Tab. 4-4: Parameter data type assignment



Conversion to a data type with a greater value range is possible.
E.g. "T_UINT8 ⇒ T_UINT16 ⇒ T_INT32 ⇒ T_UTF8STR"

4.2.8 Error Messages

The communication state can be monitored and/or logged to a file using the engineering environment of WinStudio via Remote LogWin.

The following messages are output by the Bosch Rexroth WinStudio driver:

0	; OK
1003	; No IP address available for this data source
1004	; Login at COM server failed
1005	; Communication error Data source not available, connection to control configured incorrectly, no physical connection to the control, network problems etc.
1006	; Response timeout to a query at Com server (> 30 seconds)
1008	; Invalid handle Variable in driver sheet not in control symbols. Symbol file loaded? Reload program, incorrect symbol file

Tab. 4-5: Messages of the Bosch Rexroth WinStudio driver

For more information on the remote LogWin, refer to the IndraWorks online help "Rexroth WinStudio 7.3 Technical Reference".

4.2.9 Message Information

- 0 No errors. Communication online.
- 1003 The IP address is entered into the target control during driver configuration. If this entry is blank or incorrect, the driver cannot connect to the communication server on the target control.
- 1004 Login at COM server failed. The basic control connection was established. The login request response is negative or there is no response at all. Either no or an old status of the Com server is installed in the control or the control is in an undefined state and the Com server task cannot be executed anymore.
- 1005 The communication error is always displayed if sending and receiving Com server data failed.
- 1006 The Com server does not react on client requests. The control is busy or an exception is reported.
- 1008 Invalid handle. This message is sent by the communication server if a variable, which does not exist in the control, is requested by the client. This should not occur in normal mode, as it can be checked with the "Check" browser function whether all variables on the driver sheets have a counterpart in the control if the driver sheets are filled out correctly.

If the program is reloaded, the communication server discards all previously registered handles. With feedback 1008 of the communication server, the client detects that its handle is not valid anymore. The communication is re-established by the client. All handles are registered again. When reloading the program, the client reports an 1008 error for several seconds until the client is synchronized again.

Communication Interfaces

If the "1008" error is continuously present, there are three causes:

1. At least one driver sheet was not correctly filled out and contains variable descriptions not present in the symbol file of the control.
Solution: Check driver sheets using the "Check" function of the browser.
2. An incorrect or incomplete symbol file was loaded to the control.
Solution: Check whether the project download was executed with symbol information and whether the correct symbol file was loaded.
3. The Com server does not have enough handles.

4.2.10 Documentation

The following documentation is available for a detailed description of the BRWS driver (Bosch Rexroth WinStudio driver):

DOK-CONTRL-IL2GPRO*Vxx-APyy-EN-P

xx - Version

yy - Edition

4.3 OPC.lwSCP

4.3.1 Properties

OPC.lwSCP is the Bosch Rexroth standard OPC server.

It is used for the communication connection of IndraWorks projects with the standard user interface "IndraWorks Operation" on PC-based hardware systems (see [Visualization Devices, page 17](#)).

It can also be used for WinStudio and "Third Party" applications. The IndraWorks Com package has to be installed.

If OPC.lwSCP is used without IndraWorks Operation Desktop, the OPC.lwSCP has to be configured for devices it has to connect to.

The following documentation [SCP-OPC Configurator, page 88](#), is only relevant for this use case.



For detailed information on the OPC communication, the items that can be used and a possibly required client programming, refer to DOK-IM*ML*-OPC*COM*Vxx-APxx-EN.

4.3.2 SCP-OPC Configurator

General

The SCP-OPC configurator declares devices provided by SCP via OPC. The SCP-OPC configurator can be used to create configuration files and to label them as active configuration. When starting the OPC server, the active configuration is analyzed and the relevant SCP devices are created. Selected devices are highlighted in blue. Modifications made at an active configuration applies at next OPC server restart.

The SCP-OPC configurator displays the list of devices in an SCP-OPC configuration file in the main application window.

The following device properties are displayed:

Column	Description
Device	SCP device name, cannot be edited
Type	SCP device type, cannot be edited
OPC alias	OPC alias name of the device, can be edited, optional

Tab. 4-6: Device properties displayed

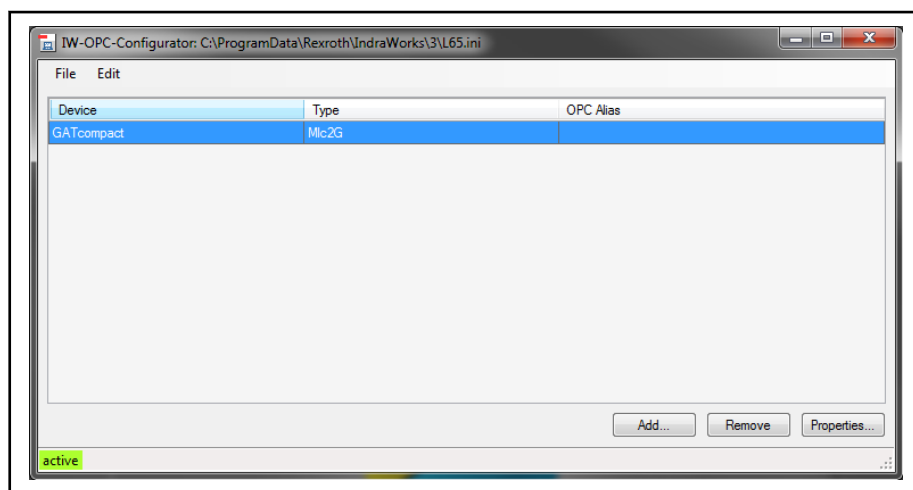


Fig. 4-4: IW-OPC Configurator: Device properties

Operation

Creating Device Configuration

Create a new SCP-OPC configuration file via the main menu items **File ► New**. Enter the file name when saving the configuration. If there is unsaved configuration data when exiting the application, you are prompted to save the data.



The title bar of the application shows path and file name of the currently opened file. If a file was modified, since it was opened or since it was last saved, the symbol "*" is displayed at the end of the file name in the title bar of the dialog.

Opening Device Configuration

Create a new SCP-OPC configuration file via the main menu items **File ► Open**. Select the directory and the configuration file in the file selection dialog. Only files with the extension ".ini" are displayed. The SCP-OPC configuration is displayed as a list of devices in the main application window. If the open file is located the currently active SCP-OPC configuration, a light-green field with the label "active" is displayed in the status bar of the application.



The title bar of the application shows path and file name of the currently opened file. If a file was modified, since it was opened or since it was last saved, the symbol "*" is displayed at the end of the file name in the title bar of the dialog.

Adding Devices

Via the main menu items **Edit ► Add**, the menu item **Add** or the **Add** button, a selection dialog is displayed after the data source was entered. Select the devices to be added to the configuration. Devices already included in the configuration are grayed out in the selection dialog. IndraWorks projects, SCP-OPC

Communication Interfaces

configuration files or an active SCP-OPC configuration are available as data sources. Using the "Properties" option, both in the selection dialog and in the main window of the configurator, a dialog showing the properties of an SCP device can be displayed for each device.



The devices have to be unique within one configuration. If devices are selected in the selection dialog which are already included in the configuration (shown in grey), they are not be added again.

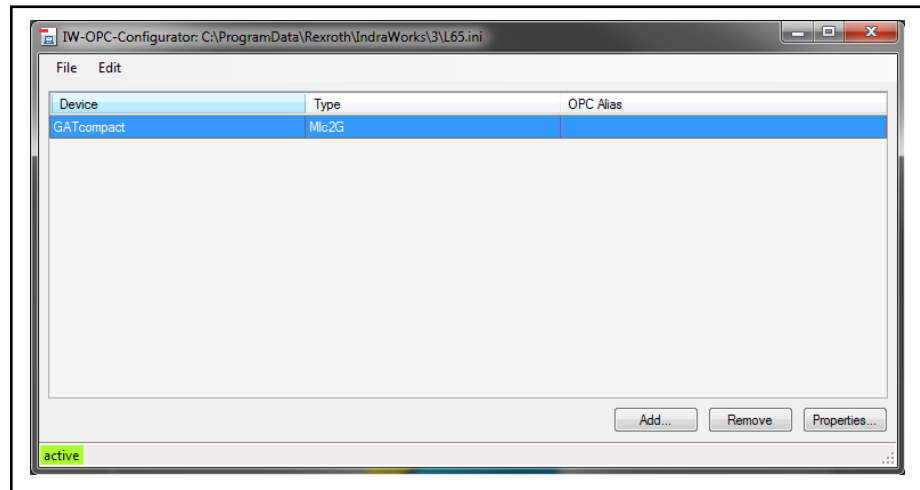


Fig. 4-5: Dialog to select devices

Displaying Device Properties

The "Properties" dialog of the selected device in the device list is displayed with all properties via the main menu items **Edit ► Properties**, the context menu item **Properties** or the **Properties** button. The option is only enabled if exactly one device of the list is selected.



Open the "Device Properties" dialog either from the main window or from the "Select Devices" dialog.

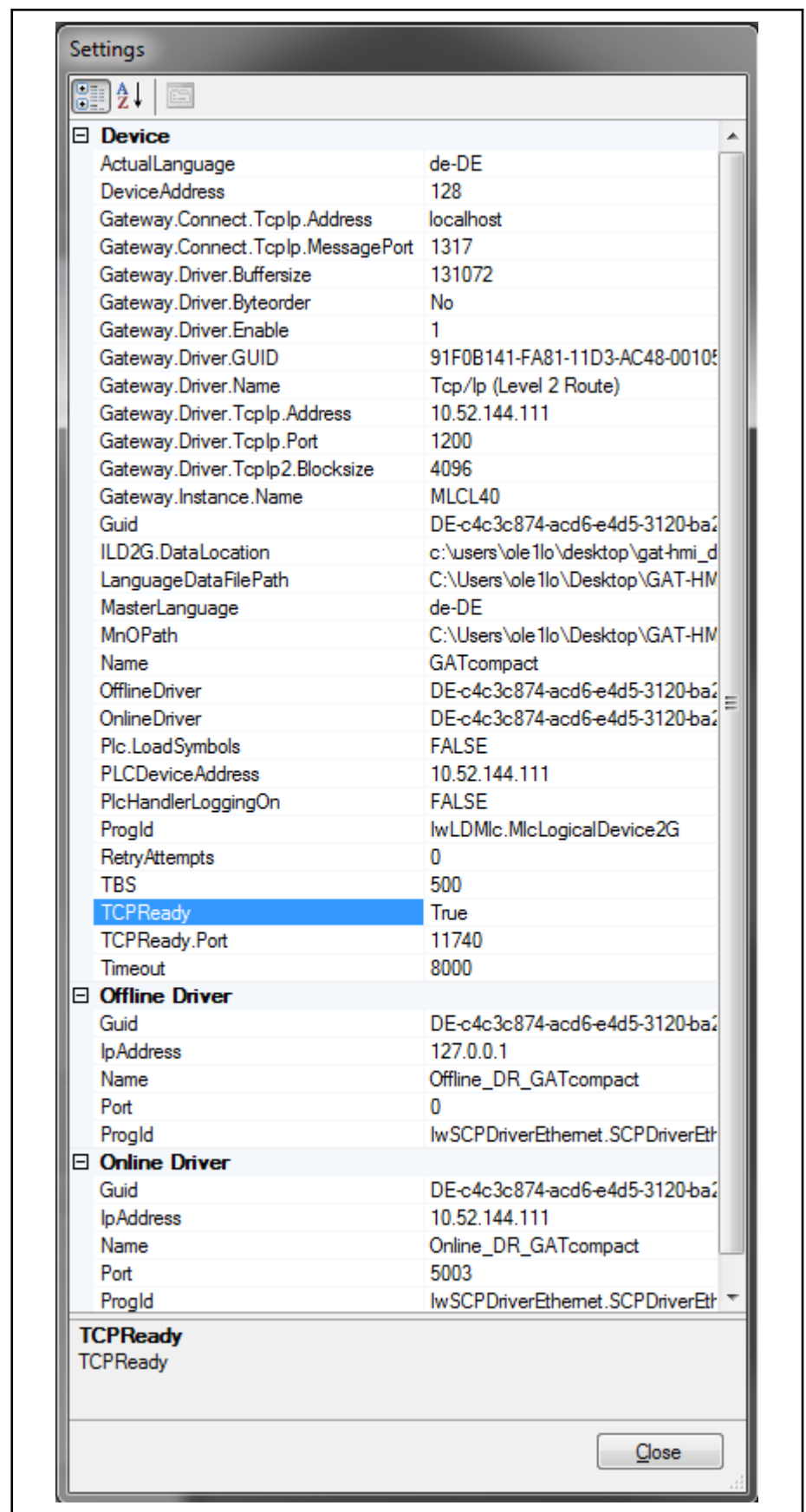


Fig. 4-6: "Settings" dialog

Communication Interfaces

Changing Devices OPC Alias

Device name and device type cannot be changed by the user. These fields are displayed as non-editable fields with a grey background. To change the OPC alias name of a device, select the device and click on the OPC alias name field. Exit editing mode with <Return> or by selecting another field.



The OPC alias name may include only the following characters:

"a-z, A-Z, 0-9, _, -".

If the naming convention is violated, a message is displayed, the background of the field turns red and an error symbol is displayed in the field. A description of the present error is also shown in the tool tip of the field marked as incorrect.

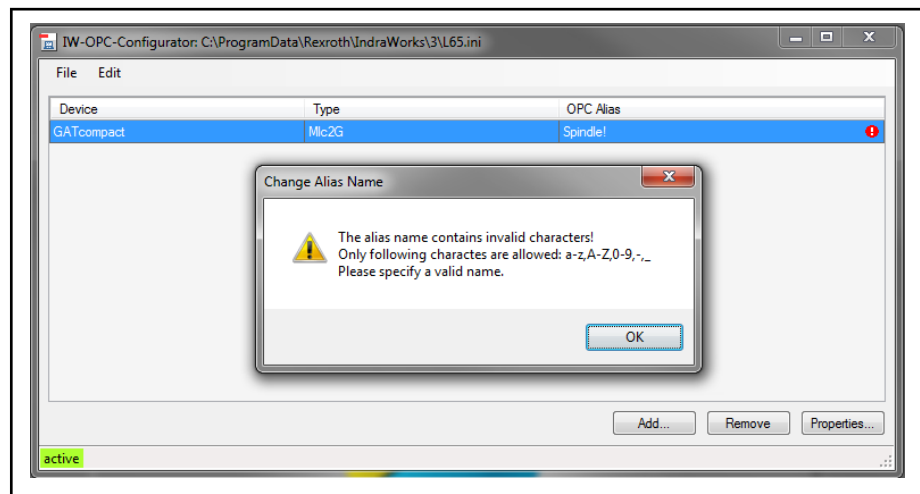


Fig. 4-7: Dialog: Chang Alias Name

Removing Devices

Remove the selected devices from the device list - after confirming a security prompt - via the main menu items **Edit ► Remove** or the context menu item **Remove** or the **Remove** button.



This option is only enabled if at least one device is selected from the list.

Saving Device Configuration

Save the configuration currently being worked on under the directory and file name given in the title bar of the application via the main menu items **File ► Save**.



This option is only available if any changes were made since the data was opened or last saved.

Saving Device Configuration As

Save the configuration currently being worked on under a different file name via the main menu items **File ► Save As**. Select the directory in a file selection dialog and enter the file name (for the file to be saved). If the file already exists in the target directory, overwriting the file has to be confirmed.



The "Save" option is not available for newly created configurations, as no file name was specified. The option "Save As" is always available if a configuration file is open.

Activating Device Configuration

Specify that the file currently open is used as active configuration via the main menu item **File ▶ Activate**. That means that this configuration is used to generate the device list of the OPC server at next restart. Only devices defined in this configuration file are available via the OPC server.

If the open file is located the currently active SCP-OPC configuration, a light-green field with the label "active" is displayed in the status bar of the application.



A configuration can only be labeled as active if it was previously saved as file.

Closing Applications

Close the application via the main menu item **Close** or via the system menu x. If there is unsaved data, you are prompted to save the data.

4.3.3 IwSCP and OPC.IwSCP Items

General Information

This chapter provides an overview on the items relevant for the OPC client and available in the "IwSCP server".

When using the items in the OPC client, ensure that the item syntax described for the "IwSCP server" is extended with the specification of the "requested data type" and the "control name or alias".

For direct comparison, the item syntax is shown first for the "IwSCP" server and subsequently for the "OPC.IwSCP" server.

IwSCP Server

- PLC variables
"[IwSCP service],[fully qualifying variable name]"
Example:
"Plc.PVL,Application.HMIVARS.bLifeSign_hmi"
- Sercos parameter
 - MLC/MLP
"[IwSCP service],[drive address],[parameter number],[element coding]"
Example:
"SPA1,0,C-0-0001,D"
"SPA1,0,C-0-0001,40"
 - MLD
"[IwSCP service],[parameter number],[element coding]"
Example:
"SPA2,S-0-0030,D"
"SPA2,S-0-0030,40"

OPC.IwSCP Server

- PLC variables

Communication Interfaces

"![required data type],[control name or alias],[lwSCP service],[fully qualifying variable name]"

Example:

"!Bool,ImMlc2G,Plc.PVL,Application.HMIVARS.bLifeSign_hmi"

- sercos parameter

- MLC/MLP

"![required data type],[control name" or

"alias],[lwSCP service],[drive address],[parameter number],[element coding]"

Example:

"!UI4,ImMlc2G,SPA1,0,C-0-0001,D"

"!UI4,ImMlc2G,SPA1,0,C-0-0001,40"

- MLD

"![required data type],[drive name or alias],[lwSCP service], [parameter number], [element coding]"

Example:

"!BSTR,Drive01,SPA2,S-0-0030,D"

"!BSTR,Drive01,SPA2,S-0-0030,40"

The "OPC.lwSCP data types" listed in the following table can be used for the parameter "required data type".

OPC.lwSCP Data types	IEC data types	Value range Memory size
BOOL	BOOL	TRUE (1) and FALSE (0) 8 bits
I1	SINT	-128 to 127 8 bits
I2	INT	-32768 to 32767 16 bits
I4	DINT	-2147483648 to 2147483647 32 bits
I8	LINT	-2 ⁶³ to 2 ⁶³ -1 64 bits
UI1	USINT	0 to 255 8 bits
UI2	UINT	0 to 65535 16 bits

Communication Interfaces

OPC.lwSCP Data types	IEC data types	Value range Memory size
UI4	UDINT DATE TIME DATE_AND_TIME DT TIME_OF_DAY TOD	0 to 4294967295 32 bits
UI8	ULINT LTIME	0 to 2 ⁶⁴ -1 64 bits
R4	REAL	±(1.175494351e-38 to 3.402823466e+38) 32 bits
R8	LREAL	±(2.225073858072014e-308 to 1.7976931348623158e+308) 64 bits
BSTR	STRING WSTRING	Any string with 80 characters if no length is specified in the PLC decla- ration.

Tab. 4-7: OPC.lwSCP data types

IndraLogic Items

Overview

The IndraLogic items relevant for the OPC client are listed below.

Item overview

- **Plc.State**
Reading current status information of the IndraLogic Runtime.
- **Plc.PVT**
Reads symbol information from the PLC project.
- **Plc.PVF**
Reads symbol information of a variable.
- **Plc.PVL**
Reads and writes variable values.
- **Plc.PVBS**
Access to PLC variables (structures, arrays) as byte stream.
- **Plc.PVLK**
Access to PLC variables (structures, arrays)

sercos Items

Overview

The sercos items relevant for the OPC client are listed below.

Item overview

- **SPH1**
Reads and writes the sercos phase.

Communication Interfaces

- **SPA1**
Reads and writes elements of the sercos parameters.
- **SPA2**
Reads and writes elements of the sercos parameters of a drive.

4.3.4 Exclusion of Liability

WARNING

No liability for damages caused by customer OPC clients

Via OPC, you have unlimited access to all device data and the entire device functionality.

For damages and loss of data due to improper OPC handling in "third-party" OPC client applications and WinStudio user screens or due to a non-compliance of device-specific safety guidelines, Bosch Rexroth is not liable. This affects writing of device-sensitive and safety-relevant data in particular.

4.4 IndraLogic OPC.DA

4.4.1 Properties

The IndraLogic OPC server OPC.DA is an OPC server to access the PLC system of the control.

IndraLogic OPC.DA can be used for PC-based and embedded-PC-based visualizations.

The server has to be configured to operate an IndraLogic OPC.DA server. This specifies the devices to be connected with the OPC.DA server at start (see [Configuring IndraLogic OPC.DA Server](#), page 96).



For detailed information on the OPC communication, the items that can be used and a possibly required client programming, refer to DOK-IM*ML*-OPC*COM*Vxx-APxx-EN.

4.4.2 Configuring IndraLogic OPC.DA Server

General Information

This chapter describes the configuration of the IndraLogic OPC.DA server "IndraLogic.OPC.DA" for PC systems (Windows XP/Windows 7) and CE systems (Windows CE 6) to communicate with IndraLogic 2G control systems.

The OPC.DA server is configured using the "OPCServer.ini" file in the "%CommonAppData%\Rexroth\IndraWorks" directory:

Example:

Paths:

"C:\ProgramData\Rexroth\IndraWorks"(Windows7) or "C:\Documents and Settings\All Users\Application Data\Rexroth\IndraWorks" (WindowsXP) on PC systems or in the "\StorageCard\Rexroth\Controls\OPCServer" directory of CE systems.

The "OPCServer.ini" is created on the PC-based control panels using a configurator. See also [OPC.DA Server Configuration](#), page 97.

For the manual configuration of the "OPCServer.ini" (for example required for WindowsCE-based visualization devices if no configurator is available during

Communication Interfaces

the IndraWorks Communication installation), refer to [OPC.DA Server Configuration File OPCServer.ini, page 100](#).



To communicate with the control, a symbol configuration has to be on the control. See also [Symbol Configuration, page 133](#).



The IndraLogic OPC.DA server supports the communication to the IndraLogic 2G control systems.

This chapter describes only the relevant entries in the OPC.DA server configuration file "OPCServer.ini" to communicate with IndraLogic 2G control systems.

OPC.DA Server Configuration

Determining Communication Address

To communicate with the control, the OPC.DA server or the underlying communication component "GatewayService" require a "communication address" partially determined from the IP address and the subnet mask of the control.

To exclude the communication address as source of error in the configuration file, determine it using a communication test with the control.

The following figure shows how the communication address "0000.c0a8.0001" was determined for the IndraMotion MLC with the IP address "192.168.0.1" and for the PLC communication option UDP. After a successful communication test, the communication address is displayed in the "Address:" line in the "PLC communication successful:" section of the "Result of connection test" field as hexadecimal number and used to configure the OPC.DA server.

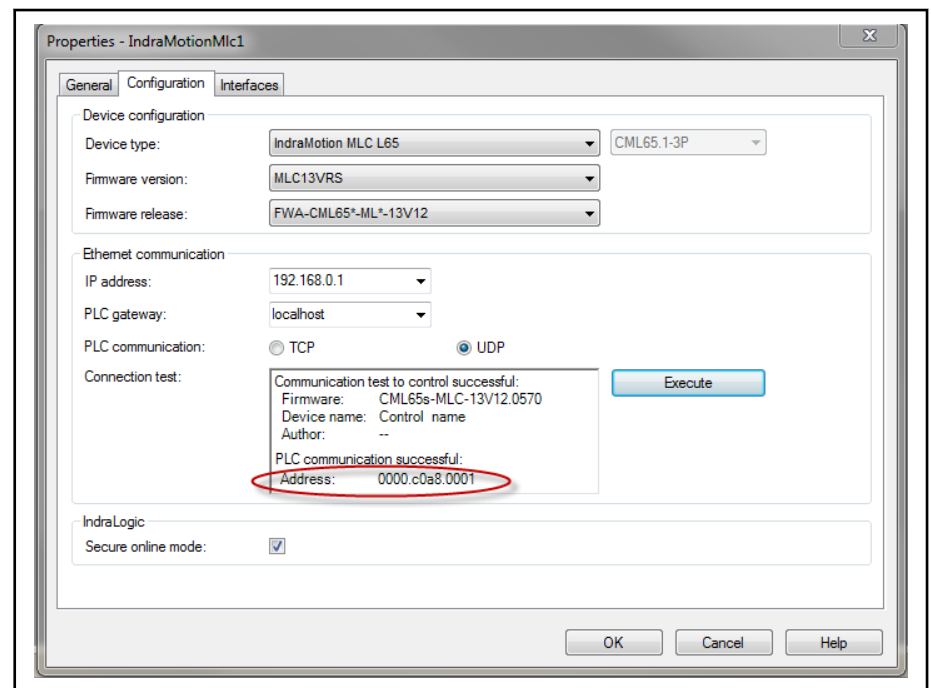


Fig. 4-8: "Properties" dialog IndraMotion MLC - Determining communication address

OPC.DA Server Configuration

A configurator is available for PC-based visualization devices to configure the OPC.DA server

Communication Interfaces

Open it via the Windows Start menu of the installed IndraWorks version

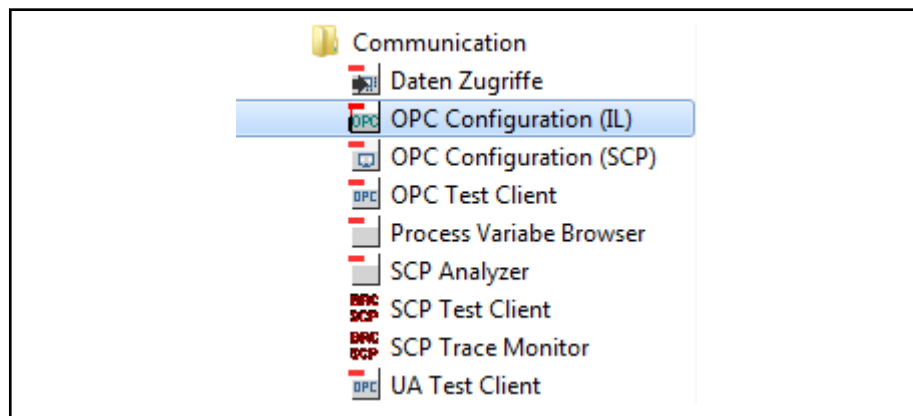


Fig. 4-9: Opening OPC.DA server configurator

Create a new configuration via the context menu and by attaching a new PLC.

- Disable logging for the "normal mode" as it is only significant during commissioning.
- The standard update rate of 200ms is recommended when registering groups and items at the OPC.DA server. Higher update rates are generally not required.



For a detailed description of the configuration parameters, refer to [OPC.DA server configuration file - Serversection \[server\], page 101](#).

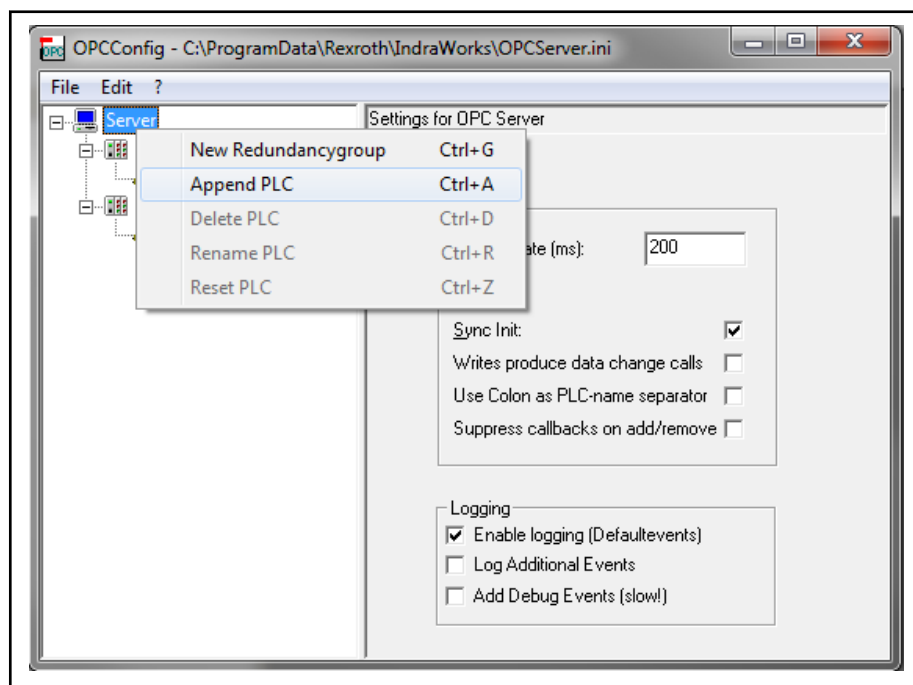


Fig. 4-10: Creating a new control within the OPC.DA server configuration

For the communication of the OPC.DA server with 2G control systems, select an interface "GATEWAY3" (= default) used establish the connection to the control.

Subsequently, make the connection settings where the IP address of the 2G control has to be entered in HEX.

Communication Interfaces

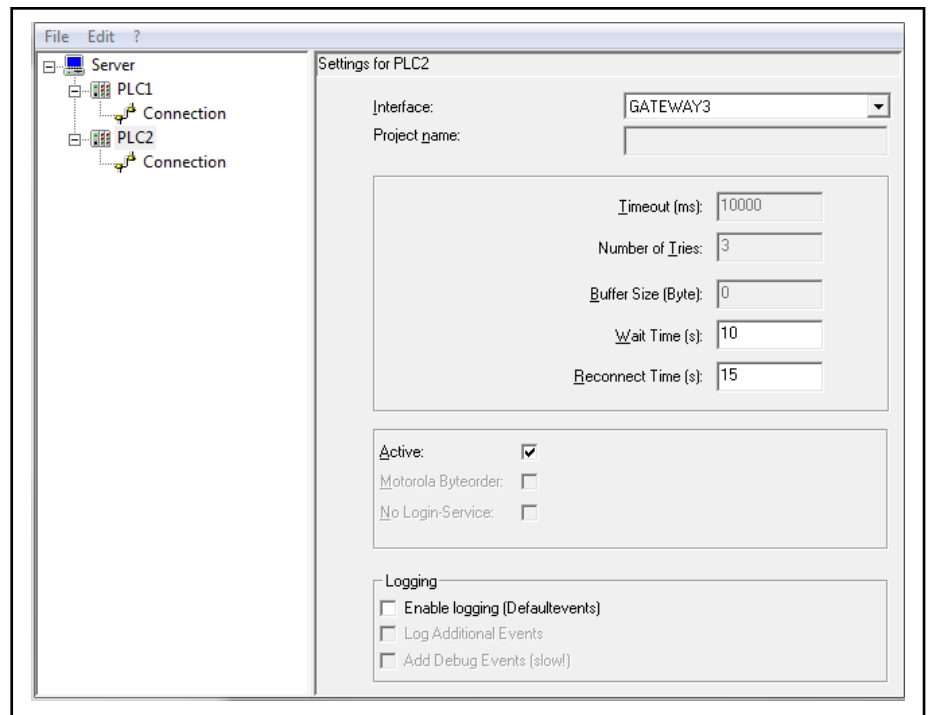


Fig. 4-11: Configuring interface (gateway) for the communication in the OPC.DA server configurator

The address as determined under [Determining Communication Address](#), page 97, in IndraWorks Engineering was used.

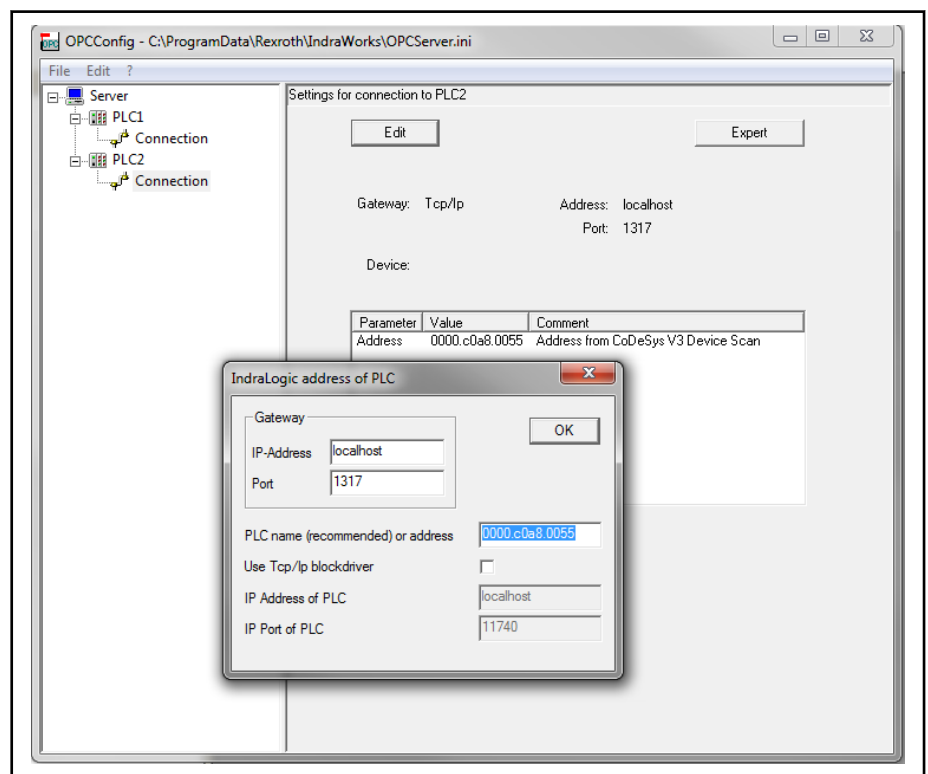


Fig. 4-12: Configuring connection settings in the OPC.DA server configurator

Save the configuration under "OPCServer.ini" to apply it upon next OPC.DA server start

For configuration file paths, see [chapter "General Information"](#) on page 96

Communication Interfaces

OPC.DA Server Configuration File "OPCServer.ini"

The OPC.DA server configuration file "OPCServer.ini" is structured like an "INI file", i.e. it contains sections and parameter with names and values. The character ";" (semicolon) is used to insert comment lines for documentation purposes or to hide parameters.

To create an OPC.DA server configuration, use the template file "OPCServer.ini.template". The template file is stored in the directory "%CommonProgramFiles%\Rexroth\" with every IndraWorks Suite "Engineering" or "Communication" installation version.

The placeholder or the environment variable "%CommonProgramFiles%" is set by the operating system depending on the installed language and can be queried by entering the line "echo %CommonProgramFiles%" in a prompt. In case of English interfaces or MUI (Multilingual User Interface) operating systems, the path "C:\Program Files\Common Files" is output.

The location of the directory containing the configuration file "OPCServer.ini" for the OPC.DA is described in [Windows XP, page 104](#), and [Storage directory of the OPC.DA Server Configuration File, page 104](#) for PC systems and in [Windows CE, page 105](#) for CE systems.

```
[Server]
Version=0x305040A
updaterate=200
logevents=1
logfilter=15
syncinit=1
ReadCyclesAfterWrite=0
UseColonAsPlcDevId=0
GroupUpdateBehaviour=0
PLCs=1
PLC0=PLC1

[PLC:PLC1]
interfacetype=GATEWAY3
reconnecttime=15
waittime=10
active=1
logevents=1
logfilter=15
gateway=Tcp/Ip
gatewayaddress=localhost
gatewayport=1317
parameters=1
parameter0=Address
value0=0000.c0a8.0001

[Redundancy]
NumOfGroups=0
```

Fig. 4-13: Configuration file template of the IndraLogic OPC.DA server

The configuration file always has a server section and any number of control sections. The following tables [OPC.DA server configuration file - Serversection \[server\], page 101](#), and [OPC.DA server configuration file - Control section \[PLC:...\], page 102](#), describe the essential configuration parameters for the server and control section of the file "OPCServer.ini".

Communication Interfaces

Parameter name and parameter value	Description
[Server]	Section identifier of the "Server" section
updaterate=200	Update rate of the OPC.DA server in milliseconds. This is the update interval of the internal OPC cache. The setting for the OPC client update is determined by the properties of the OPC group or subscription. Depending on the number of registered variables, it is necessary to change the update rate to reserve enough resources for other processes running on the PC. The update interval should not fall below 200 ms.
logevents=1	Enabling log mechanism. All server events are written to the file "OPCServer.log" according to the settings of the parameter "logfilter". The log file is located in the same directory as the "OPCServer.ini" file. During installation or testing, the parameter "logevents" can be set to "1". Logging has to be deactivated again at runtime, i.e. "0".
logfilter=15	Filter definition for server events. <i>Possible values are:</i> • 15 0x0000 0000 0000 000F: Standard logging • 47 0x0000 0000 0000 002F: Standard logging and additional entries • 63 0x0000 0000 0000 003F: Standard Logging, additional entries and debug entries
syncinit = 1	If this option is enabled, a synchronous connection is initialized.
ReadCyclesAfterWrite=0	After each written item, the OPC server sends a callback to the client. Previously, when values changed fast, a callback might not have been sent for each change and the value was thus not updated in the client. To ensure an update, use the "ReadCyclesAfterWrite" setting. It indicates the number of read cycles after the latest writing the values are to be updated in the client (even if they have not changed). (Default: callback created after 0 reading cycles. For a different number of cycles, enter it manually to the .ini file)
UseColonsAsPlcDevideir=0	If this value is not 0, the first "." (e.g. the dot after the PLC name) is replaced by a colon.
GroupUpdateBehaviour=0	In this case, data-changed callbacks are created for the activated/deactivated item. If the entry is 0, callback functions are created for all items in the group whenever one or multiple items are activated or deactivated within the group.
PLCs=1	Number of configured controls.
PLC0=PLC1	List of all configured controls in the configuration file.
PLC...	Note that the counter index (parameter name) is continuous and that a unique control section exists for all listed controls (parameter value e.g. "PLC1").

Tab. 4-8: OPC.DA server configuration file - "Server" section [server]

Communication Interfaces

Parameter name and parameter value	Description
[PLC:PLC1]	Section identifier of the control section "PLC1". If a connection to this control is to be established when starting the OPC.DA server, it has to be available as value in the list of all controls of the server section in one of the parameters "PLC..." (e.g. "PLC0=PLC1").
interfacetype=GATEWAY3	Control type definition. This parameter defines the control system that is communicated with. GATEWAY3 is used for IndraLogic 2G control systems
reconnecttime=15	Time interval in seconds during which an attempt is made to re-establish the connection.
waittime=10	Waiting time in seconds of the subordinate communication layer "PLC Handler". Maximum waiting time to establish a connection to the control.
active=1	Activate control. Use this parameter to enable or disable the communication with the control. e.g. the configuration of a system can already be created and unavailable modules can be deactivated.
logevents=1	Enabling log mechanism. All server events are written to the "OPCServer.log" file according to the settings of the "logfilter" parameter. The log file is located in the same directory as the "OPCServer.ini" file. During installation or testing, the parameter "logevents" can be set to "1". Logging has to be deactivated again at runtime, i.e. "0".
logfilter=15	Filter definition for server events. <i>Possible values are:</i> 15 0x0000 0000 0000 000F: Standard logging 47 0x0000 0000 0000 002F: Standard logging and additional entries 63 0x0000 0000 0000 003F: Standard Logging, additional entries and debug entries
gateway=Tcp/Ip	Communication settings from OPC.DA server to "GatewayService" - gateway - Protocol used for communication. - gatewayaddress - IP4 address or host name to the gateway establishing the connection to the control. - gatewayport - Network port where the "GatewayService" accepts incoming connections. This setting has to match the gateway configuration.
gatewayaddress=localhost	
gatewayport=1317	
parameters=1	Number of parameters used. The parameter selection can vary with regard to the configuration.
parameter0=Address	Definition of the communication address of the control. It is applied for the parameter "value0" as described in Determining Communication Address, page 97 ,.
value0=0000.c0a8.0001	
Advanced parameter settings	

Communication Interfaces

Parameter name and parameter value	Description
parameter1=Port	Port of TCP/IP block driver (not used in default configuration)
value1=1234	Address is applied via the parameter "value1"
parameter1=DirectAddressesOnly	Not used for IndraLogic 2G controls
parameter2=CheckData-ChangeOnPlc	If this parameter is active ("value<Number>" = 1), the control provides only the modified values. The option requires more resources on the control and might cause increased communication load. Only to be used in closed systems and with known application data
parameter3=ClientAddress-Resolution	Not used for IndraLogic 2G controls
parameter4=DontExpand-SimpleTypeArrays	While the connection is established, all data types available in the symbol configuration of the control are loaded. Complex data types (arrays, structures) are subsequently expanded to represent all elementary data types in the symbol configuration. For intelligent client applications able to expand elementary data types themselves, this feature can be suppressed ("Value<Number>" = 1)
parameter5=WriteThrough-ReadCache	By default, all synchronous write commands to the control do not affect the data to be cyclically read If a value is written from the client (visualization) and from the control at the same time, the following can result: Depending on the client, there can be a short flickering between the current and the previous variable value on the control after writing. If it cannot be excluded that variables are never simultaneously written from the client to the control, it can be checked in the control after each writing task whether the variables to be written are also part of the cyclic reading list. If this is the case, the cache value of the respective variables is set to the written value and the data change callback is called for each list. Note that a registered callback is called more often. This feature requires additional computing time during each writing and the computing time also increases proportionally to the variables to be written as well as to the number of variables for each cyclic reading list. To enable this option, set "Value<Number>" = 1
parameter6=AllowAlternate-Symbolfile	This option is not supported
parameter7=DontLoadSymbolsFromPlc	If the client knows all symbol names and types and browsing can thus be omitted, this option can be enabled (value<Number> = 1)
parameter8=DontExpand-ComplexTypeArrays	Use this option to suppress the array expansion of complex data types (structures). The "DontExpandComplexTypeArrays" option can normally only be used in close systems, as the client has to be familiar with the alignment of the members. If the options "DontExpandComplexTypeArrays" and "DontExpandSimpleTypeArrays" are active at the same time, only the top level nodes of the GVLs or POU's are returned. To enable this option, set "Value<Number>" = 1

Tab. 4-9: OPC.DA server configuration file - "Control" section [PLC:...]

An IndraLogic OPC.DA server configuration file establishing a communication connection on PC and CE systems with the control IP address "192.168.1.31" is shown.

Communication Interfaces

```

[Server]
Version=0x305040A
updateRate=200
logEvents=1
logFilter=15
syncInit=1
ReadCyclesAfterWrite=0
UseColonAsPlcDevId=0
GroupUpdateBehaviour=0
PLCs=1
PLC0=PLC1

[PLC:PLC1]
interfacetype=GATEWAY3
reconnectTime=15
waitTime=10
active=1
logEvents=1
logFilter=15
gateway=TCP/IP
gatewayAddress=localhost
gatewayPort=1317
parameters=1
parameter0=Address
value0=0000.c0a8.0001

[Redundancy]
NumOfGroups=0

```

Fig. 4-14: Configuration file of the IndraLogic OPC.DA server - "OPCServer.ini"



The OPC.DA server configuration file "OPCServer.ini" is the same for PC and CE systems in connection with IndraLogic 2G control systems.

Storage directory of the OPC.DA Server Configuration File

General Information



The OPC.DA server configuration file "OPCServer.ini" of Windows XP and Windows CE are only compatible for IndraWorks version 12 and higher.

Windows XP

The OPC.DA server configuration file "OPCServer.ini" has to be located under
 C:\Documents and Settings\All Users\Application Data\Rexroth\IndraWorks
 for an English or for MUI operating system installations.

In the following figure, the storage directory for an operating system installation is shown:

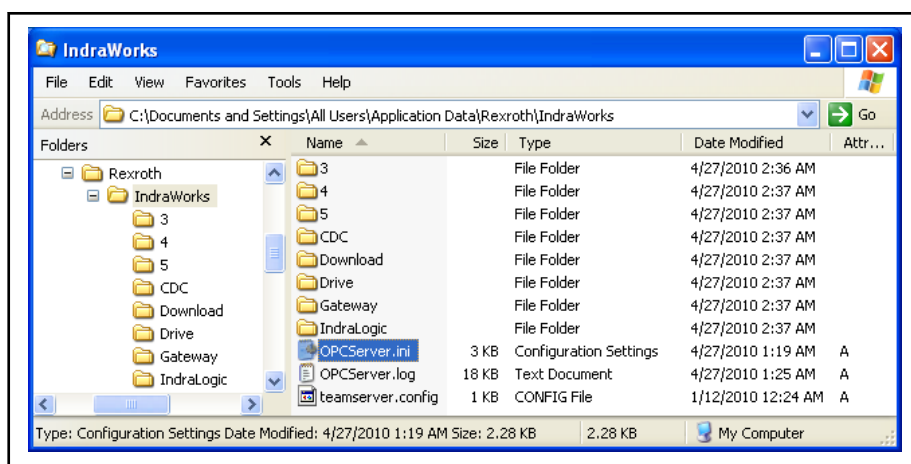


Fig. 4-15: Storage directory of the configuration file "OPCServer.ini" on PC systems with Windows XP

Windows 7

Save the OPC.DA server configuration file "OPCServer.ini" under C:\ProgramData\Rexroth\IndraWorks.

The storage directory is shown in the following:

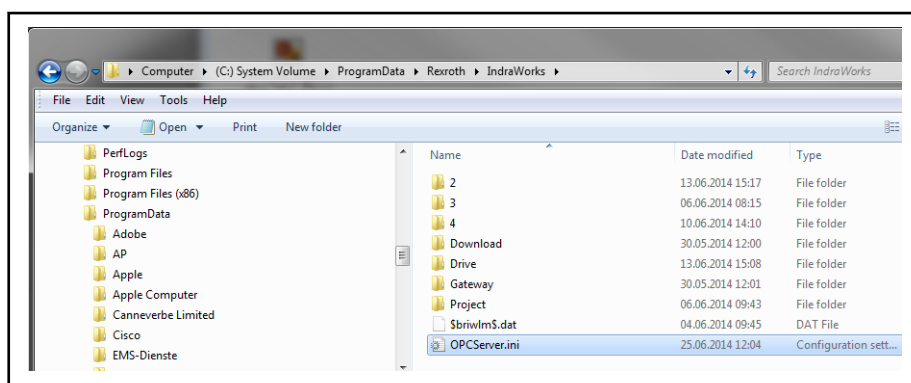


Fig. 4-16: Storage directory of the configuration file "OPCServer.ini" on PC systems with Windows 7

Windows CE

For Windows CE devices, the OPC.DA server configuration file "OPCServer.ini" for the visualization device "VEPxx.4" has to be stored under \StorageCard\Rexroth\Controls\OPCServer

The figures [fig. 4-17 "Storage directory of the configuration file OPCServer.ini on CE systems - CE Explorer"](#) on page 106 and [fig. 4-18 "Storage directory of the configuration file OPCServer.ini on CE systems - XP Explorer via FTP connection"](#) on page 106 show the storage directory with "CE Windows Explorer" and "XP Windows Explorer" for a "VEPxx.4" visualization device.

Communication Interfaces

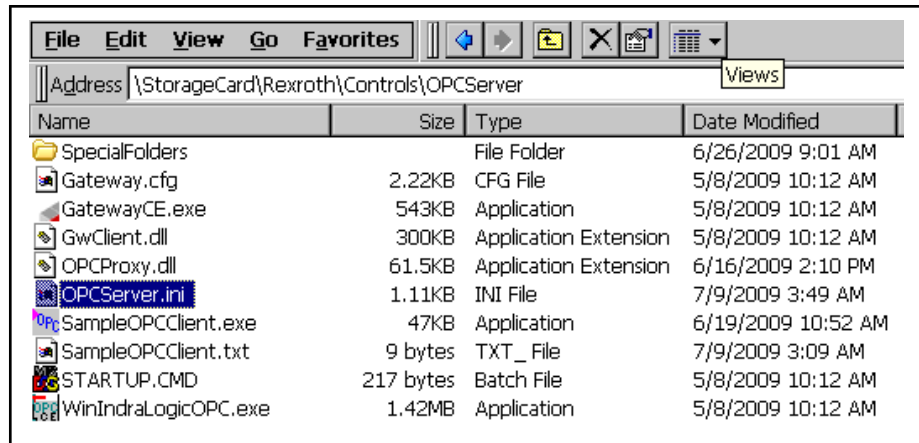


Fig. 4-17: Storage directory of the configuration file "OPCServer.ini" on CE systems - CE Explorer

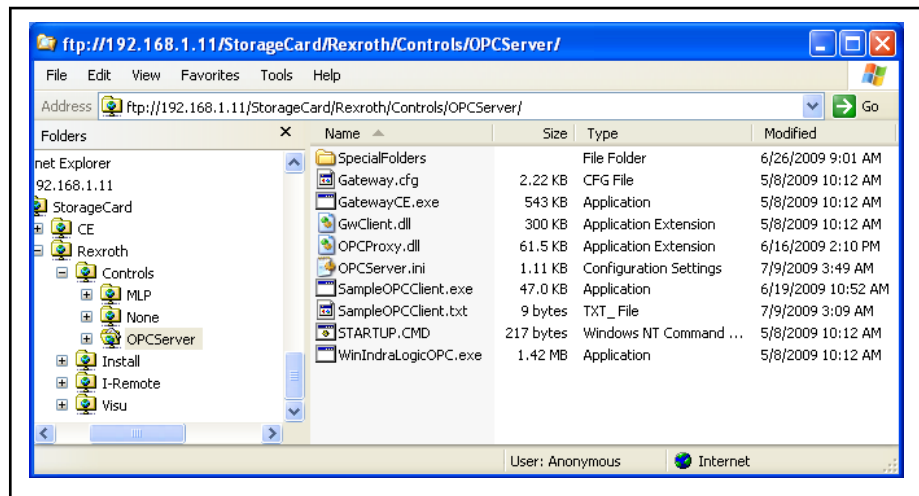


Fig. 4-18: Storage directory of the configuration file "OPCServer.ini" on CE systems - XP Explorer via FTP connection

Select "OPC.DA server" in "CE Rexroth Settings" (link "RxSettings") in the grouping field "Control" of the "Application Settings" tab to activate the OPC.DA server communication. The selection in the "Visualization" grouping depends on the visualization system. Refer to fig. 4-19 "Dialog: CE Rexroth settings (configuration of an OPC communication)" on page 106.

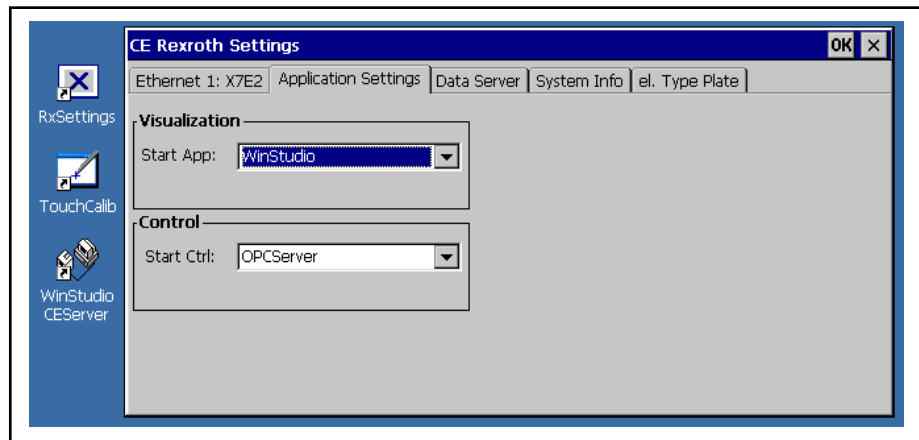


Fig. 4-19: Dialog: CE Rexroth settings (configuration of an OPC communication)

OPC.DA Server - Item Syntax

The IndraLogic OPC.DA server can access all PLC variables configured for access in the symbol configuration or via pragmas in the IndraWorks project. The item syntax consists of the "control name" (configuration file "OPCServer.ini") and the "fully qualifying variable name". The variable name is preceded by all namespaces in which the variable was declared.

A dot "." is used as a separator for the control name and the fully qualifying variable name.

See the following summary.

- Item syntax
 - "[Control name].[fully qualifying variable name]"

Example:

"PLC_IL2G.Application.HMIVARS.bLifeSign_hmi"



When using the IndraLogic function for namespace replacement with pragma

{attribute 'namespace' := 'NewNamespace'}

or via the symbol configuration, it is accessed via the newly defined namespace.

Example:

"PLC_IL2G.NewNamespace.bLifeSign_hmi"

4.4.3 Exclusion of Liability

WARNING

No liability for damages caused by customer OPC clients

Via OPC, you have unlimited access to all device data and the entire device functionality.

For damages and loss of data due to improper OPC handling in "third-party" OPC client applications and WinStudio user screens or due to a non-compliance of device-specific safety guidelines, Bosch Rexroth is not liable. This affects writing of device-sensitive and safety-relevant data in particular.

4.5 Configuring OPC Client Connection with WinStudio

To configure a connection to an OPC server (OPC.lwSCP, OPC.DA), the corresponding driver sheets are used in WinStudio. Multiple driver sheets can be created for each configured server. Communication parameters (update rates, variables to communicate, etc.) can be individually specified for each created driver sheet.

To create such a connection, add a new OPC driver sheet to the Project Explorer in the node **WinStudio ► Variables ► Communication ► OPC** of the corresponding visualization project.

Communication Interfaces

Fig. 4-20: OPC driver sheet

Use the following parameters to configure an OPC driver sheet:

- **Description:** For documentation purposes, enter a description for the OPC work sheet.
- **Server Identifier:** Depending on the previous configuration, select either the OPC.lwSCP or the IndraLogic OPC.DA server for IndraLogic 2G controls.
- **Disable:** Enter a tag or constant with a value unequal to "0" to disable the communication with the OPC server. Enter "0" or leave the field empty to enable the communication.
- **Read Update Rate:** Determine the number the server updates this group (in ms). The server uses the fastest rate suitable when entering "0".
- **Percent Deadband** (only valid for analog elements): Determine the element value change in percent the server should start sending a report.
- **Status:** Enter the tag name to get the connection status. The status "Good" corresponds to the value 1.
- **Remote Server Name:** Node name or IP address of the server in the node network.
- **Read before writing:** When enabling this option, the original values are retrieved from the OPC directly before writing new variable values to the server. The project buffers the new values to be written and subsequently reads the original values from the server. After synchronizing the project with the server, new values are written from buffer to server.
- **Read after writing:** When enabling this option, the values are retrieved from the OPC directly after writing new variable values to the server. For more information on this setting, refer to the IndraWorks online help "Rexroth WinStudio 7.3 Technical reference".
- **Accept tag name in item column:** If this option is enabled, the text in curly brackets in the "Element" field is interpreted as tag name (of type string). In this case, this tag value is used as element name by the OPC server. Thus, the link between a tag and the OPC elements can be changed at runtime by changing the value of the configured tag. If **Accept tag name in item column** is not enabled, all characters configured in the "Element" column are part of the element name (including the characters in curly brackets).
- **Variable Name:** Enter the name of the tag to be linked with the server elements.
- **Element:** Specify the item strings of the variables to be exchanged with the control.
 1. Using the WinStudio OPC browser: There has to be a connection to the control

Communication Interfaces

Right-click on **Element** and select **OPC Browser**. The controls from the respective OPC configuration are displayed. The variables in the symbol configuration of the respective control are shown in a tree structure.

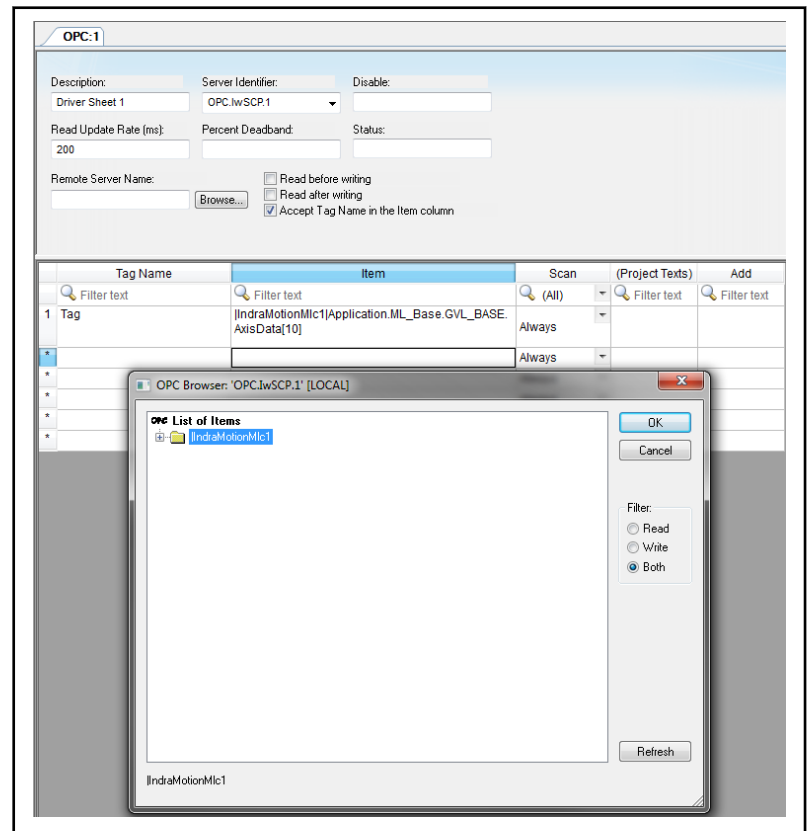


Fig. 4-21: OPC browser with OPC driver sheet. OPC.IwSCP server is selected

- Using IndraWorks tool window for process variables: No connection to the control required

Enable the tool window via **View ► Other Windows ► Process Variables** in the IndraWorks toolbar.

If the configuration of the OPC driver sheet was saved, the symbol variables of the controls available in the project are displayed. Drag and drop them to the driver sheet line. If the control is not in the project, load the symbol configuration file of the external control and thus project them to the HMI project. For more information on the tool window for process variables, refer to the IndraWorks online help

Communication Interfaces

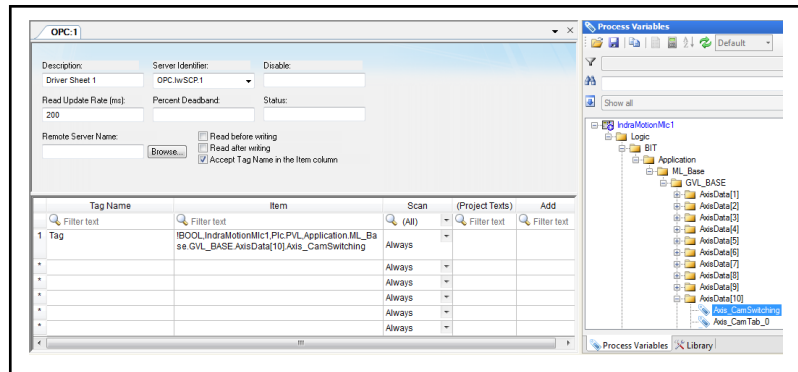


Fig. 4-22: OPC driver sheet configuration and tool window for process variables

- **Scan:** Specifies the conditions when the tag value is read from the remote device or server and then updated in the project database. The following options apply:
 - **Always** means that the communication worksheet is read and updated at each scan irrespective of whether the tag is used in WinStudio screens, scripts or worksheets. It is recommended for tags that have to be monitored continuously in the background such as for triggering alarms, in recipes or tags in the historic database.
 - **Screen** means that the tag is only read and updated if it is at least used in one open screen either locally or on another client station. It is recommended for tags that are exclusively used in screens, as those are not always to be updated. Selecting this option can improve the project performance.
 - **Auto** means that the project automatically selects either **Always** or **Screen** depending on whether the tag is used in the project. If the tag is only used in a WinStudio screen, the scan is reset to **Screen** by default. If the tag is also used in other worksheets (e.g. script, computation, alarm, trend, recipe, report or scheduler), the scan is set to **Always** at runtime.

If you are not sure what to select, select **Always**. This ensures that the tag is read and updated.

- **Div:** Specify the division constant if scaling has to be adapted. It is a division factor in a read operation and a multiplication factor in a write operation.
- **Add:** Specify the addition constant if scaling has to be adapted. It is an addition factor in a read operation and a subtraction factor in a write operation.

To execute the OPC client task at project runtime, open the task configuration (**Tools ► Status** in the context menu of the "WinStudio" folder) and select the initial setting "Automatic" for "OPC Client Runtime". To test the task manually in this dialog, click on **Start**. After executing this program, a small icon is displayed in the "Info" section.

4.6 OPC UA

4.6.1 Overview

The OPC UA server is available for 12VRS and higher in the "IndraMotion MLC" and "IndraLogic XLC" systems as communication server to connect HMIs, SCADA and control systems.

It permits access to all PLC variables enabled for access in the IndraLogic symbol configuration by the respective control.

The current version of the Rexroth visualization "WinStudio" has an OPC UA client for the communication connection to the OPC UA server in the control systems.

An overview is provided on:

- the OPC UA architecture
[Introducing OPC Unified Architecture, page 111](#)
- the Bosch Rexroth OPC UA server
[Bosch Rexroth OPC UA Server, page 116](#)
- the creation of a communication connection to the Bosch Rexroth OPC UA server
[OPC UA Communication Example, page 121](#)
- the creation of a communication connection to the Bosch Rexroth OPC UA server via WinStudio



Use the "Wireshark" software (version 1.6 or higher) to diagnose connection problems between an "OPC UA client" and the "Bosch Rexroth OPC UA server".

4.6.2 Introducing OPC Unified Architecture

General Information

OPC UA is the further development of the OPC industry standard. The service-oriented architecture ensures platform independency, scalability and high-availability by dispensing with a DCOM basis. OPC UA ensures an uninterrupted, vertical integration from the control level up to the automation component - irrespective of the programming language or the operating system. OPC UA is a client/server system. Several clients can simultaneously access one server. A client can access several servers.

Specification Overview

OPC Unified Architecture was published by the OPC Foundation as multi-part specification in 13 parts <https://opcfoundation.org/developer-tools/specifications-unified-architecture/>. Can only be downloaded by members of the OPC foundation.

Part	Specification
Part 1	OPC UA Part 1 - Overview and Concepts 1.01 Specification
Part 2	OPC UA Part 2 - Security Model 1.01 Specification
Part 3	OPC UA Part 3 - Address Space Model 1.01 Specification
Part 4	OPC UA Part 4 - Services 1.01 Specification
Part 5	OPC UA Part 5 - Information Model 1.01 Specification
Part 6	OPC UA Part 6 - Mappings 1.00 Specification
Part 7	OPC UA Part 7 - Profiles 1.00 Specification
Part 8	OPC UA Part 8 - Data Access 1.01 Specification
Part 9	OPC UA Part 9 - Alarms and Conditions 1.00 Specification

Communication Interfaces

Part	Specification
Part 10	OPC UA Part 10 - Programs 1.00 Specification
Part 11	OPC UA Part 11 - Historical Access 1.00 Specification
Part 12	OPC UA Part 12 - Discovery
Part 13	OPC UA Part 13 - Aggregates

OPC UA Information Model

The OPC UA information model is based on nodes describing an object-oriented context. A node can consist of attributes, methods and events. The content depends on the NodeClass. The type model allows mapping of all types from user data up to meta data. Each node has a unique NodeID. The node connections are described by references. The reference semantics is specified by the ReferenceType.

The OPC UA information model is used as basis by other organizations such as PLCopen and extended as individual profile. For more information on these extensions, refer to Companion Specification in the respective organizations or the OPC foundation. Each OPC UA server reports the profiles it supports upon request.

Service-Oriented Architecture

The OPC UA architecture is divided into logical levels.

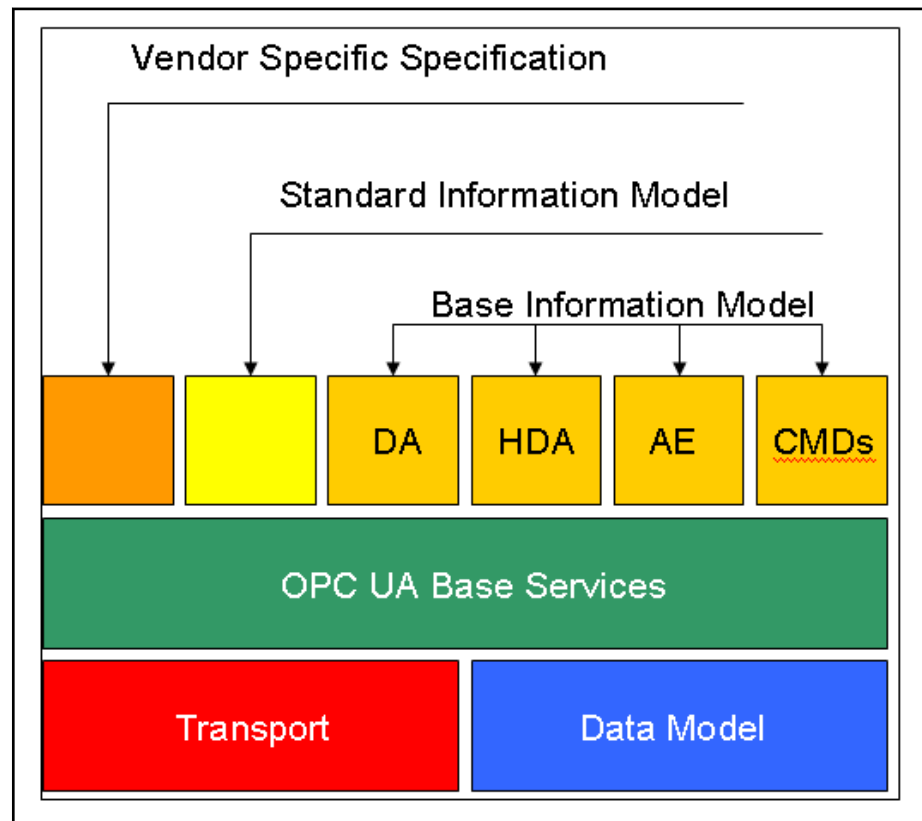


Fig. 4-23: OPC UA architecture

Transport layer

The "transport layer" serializes and deserializes data and sends or receives them. Data is transferred using an XML stream via the HTTP-HTTPS protocol or via a high-performance binary TCP protocol.

Communication Interfaces

OPC UA basic services All OPC UA basic services are abstract method descriptions. They do not depend on the protocol and form the basis for all OPC UA functions. The methods are summarized in service sets.

The most important service sets are mentioned in the following section:

- Discovery Service Set**
- The "Discovery Service Set" defines services which a client uses to determine the end points of a server and their security configuration. An end point is a variant consisting of a communication protocol, a host address, a port number and security setting.

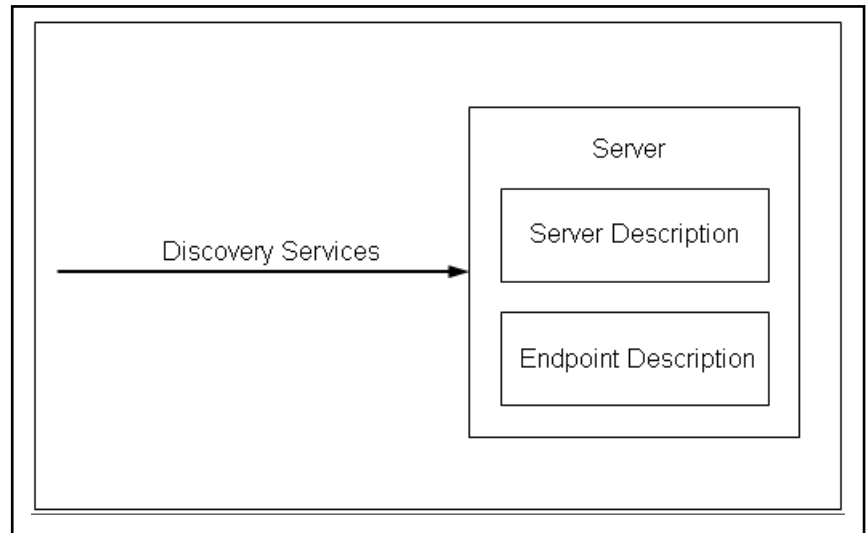


Fig. 4-24: Discovery Service Set

- Session Service Set**
- A session establishes a logic connection between client and server on application level. The connection is established by the client and can contain client-specific user and language settings.

A session is created using the "CreateSession" service. A session ID is assigned to the client. The client uses this session ID to call the following data services (Read, Write, Subscribe etc.).

A session is closed automatically by the server after client request (CloseSession) or timeout.

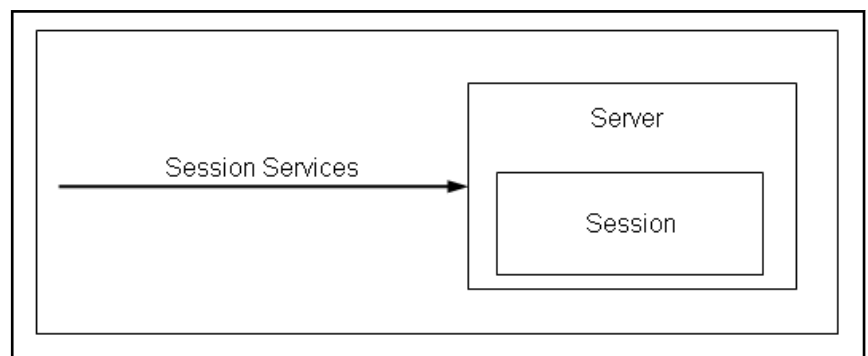


Fig. 4-25: Session Service Set

- View Service Set**
- The client can use the "View Service Set" to browse through the restricted address spaces (so-called views) of the OPC UA server. The address space presents available OPC UA server information to a client. It is generally generated by multiple nodes. The nodes are standardized objects and map "real world" objects. The most important node classes are "Variable", "Object" and "Method".

Communication Interfaces

The node connections are established using references. It is differentiated between hierarchical and non-hierarchical references. Hierarchical references are used to map a structural node order for example. Non-hierarchical references are for example used to refer to a node type definition.

The type system of nodes and references include types for objects, references, variables, events and data. It can be extended by the derivation of existing types and thus be provided with deeper semantics.

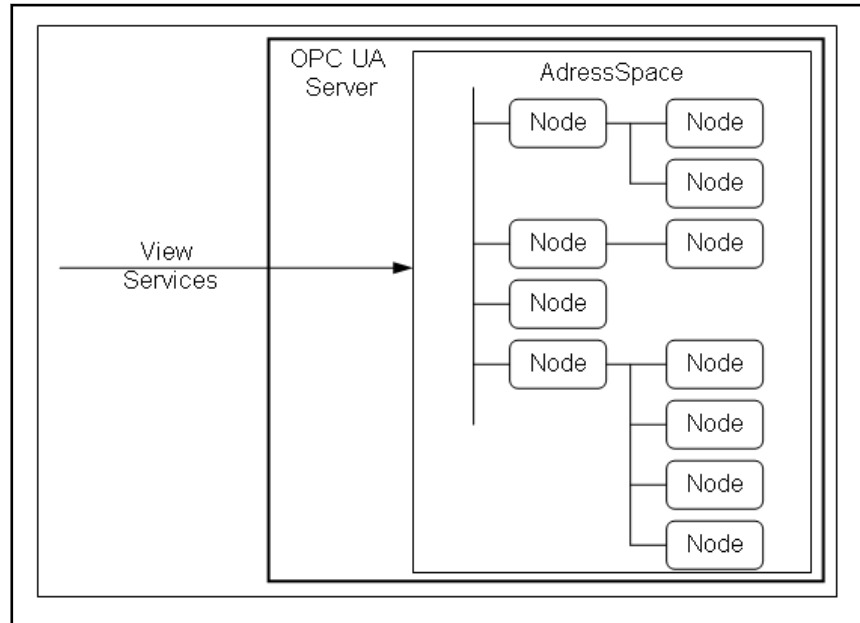


Fig. 4-26: View Service Set

- Attribute Service Set**
- A node consists of multiple attributes. These attributes describe a node in detail. The node class specifies the node attributes. Only nodes of the "Variable node" class are assigned the "Value" attribute.

All node classes have a uniform set of basic attributes. These comprise among others: NodeID to address a node, NodeClass and Browse-Name.

Using the "Service Set" attribute, read and write access to these attributes is possible.

Communication Interfaces

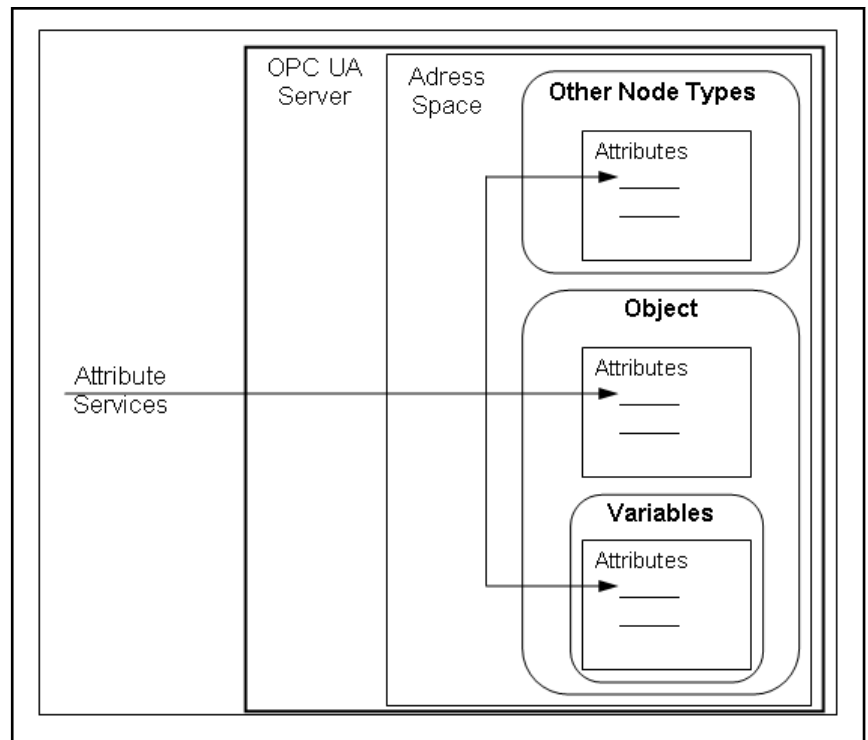


Fig. 4-27: Attribute Service Set

Subscription Service Set

- Subscriptions are an efficient method in OPC UA to receive information on value changes in the OPC UA server by the server. Using the "Subscription Service Set", clients can create, modify and delete subscriptions.

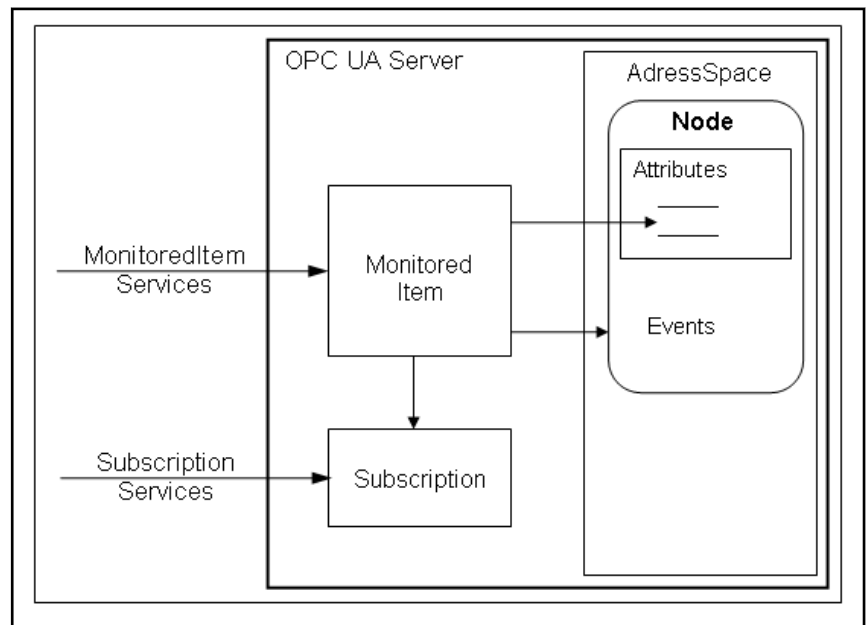


Fig. 4-28: Subscription Service Set

MonitoredItem Service Set

- A "MonitoredItem" is a "Value" attribute, an aggregation of one or several "Value" attributes or an event brought into the context of a subscription. If the value of these "Value" attributes change or if a corresponding event is output, it is automatically transferred to the client.

Communication Interfaces

Using the "MonitoredItem Service Set", clients can create, modify or delete "MonitoredItems" in the OPC UA server.

4.6.3 Bosch Rexroth OPC UA Server

General Information

The Bosch Rexroth OPC UA server is a firmware-integrated component. It ensures a secure connection via a standardized communication protocol to access the control. Adjustments or driver developments in the client are not required. The OPC UA server is initialized and started upon control startup by default. If not required, disable it to reduce CPU load.

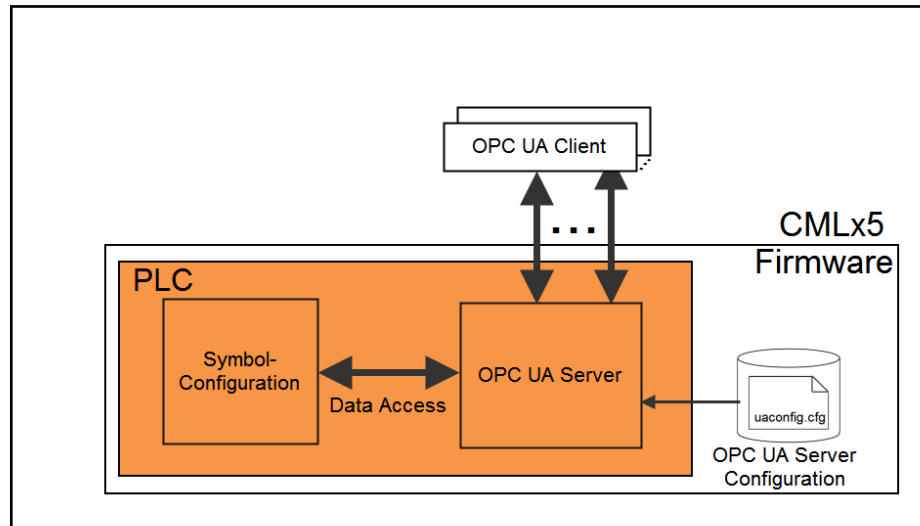


Fig. 4-29: Integrating OPC UA server

Bosch Rexroth OPC UA Server Properties

In the following, the Bosch Rexroth OPC UA server properties are described.

EndpointURL	The "EndpointURL" is part of the end point. It consists of the communication protocol and the host address. OPC UA defines a high-performance binary protocol (opc.tcp://) as communication protocol and XML-based protocol (http://). The Bosch Rexroth OPC UA server supports the binary protocol and uses the port 4840 by default. EndpointURL structure: opc.tcp://<hostaddress>:4840
MessageSecurityMode	The security level is specified via the "MessageSecurityMode". <i>Possible values are:</i> • Sign: This security level ensures the message integrity. The message can be protected against third-party changes by adding a signature. • SignAndEncrypt: The message confidentiality is ensured in addition to the "Sign" security level. Messages are encrypted using crypto algorithms. Thus, third parties do not have read access. • None: The messages are neither signed nor encrypted. The Bosch Rexroth OPC UA server only supports the MessageSecurityMode None. Entries under SecurityPolicy are ignored.
SecurityPolicy	The "SecurityPolicy" describes the message encryption type. OPC UA defines the policies "Basic128Rsa15" and "Basic256". The "SecurityPolicy" is not supported.

Communication Interfaces

PublishingInterval	Defines the cycle in which a subscription tries to send a notification to a client. The "PublishingInterval" value is specified upon the subscription creation between client and server. The Bosch Rexroth OPC UA server supports the following PublishingIntervals[ms]: 1, 2, 5, 10, 50, 100, 500, 1000, 2000, 5000
SamplingInterval	The "SamplingInterval" defines the cycle in which the server samples the subordinate data sources. The "SamplingInterval" value is specified between client and server upon creation of a MonitoredItem. The Bosch Rexroth OPC UA server supports the following SamplingIntervals [ms]: 1, 2, 5, 10, 50, 100, 500, 1000, 2000, 5000

Tab. 4-10: Bosch Rexroth OPC UA Server Properties



"PublishingInterval" and "SamplingInterval" are values specified by the client. These values affect the control load by the OPC UA server. By the client selected time intervals that are too short can affect the overall response of the control.

Configuration

Configure the OPC UA server via the parameters listed in the "Configuration" table.

The parameters are stored in a file and can be changed in that file.

The file name is specified as "uaconfig.cfg". Depending on the control system used, the file can be found in the following paths:

MTX: /plc

XLC/MLC: /ata0b or USER

Configuration

Parameters	Description
TaskRun	This parameter enables or disables the OPC UA server. Permitted values are: 0 : Server not active 1 : Server active
ShortestSamplingInterval	This parameter specifies the minimum value for the PublishingInterval of the subscriptions and the SamplingInterval of MonitoredItems. It defines the fastest interval value changes can be detected in. Note that smaller values cause a higher system load. If the parameter is set too high, value changes might not be detected anymore. Permitted values are: 1, 2, 5, 10, 50, 100, 500[ms] To avoid a control overload, this value is limited with regard to the control type used. The smallest permitted value is defined in the configuration file of the IndraLogic, Section "OPCUAStack" with the entry "ShortestSamplingInterval" and cannot be modified.
TcpPort	This parameter specifies the port number of the OPC UA server. Port number 4840 is standardized for the OPC UA communication.

Communication Interfaces

Logging	This parameter is reserved for future extensions.
LoggingLevel	This parameter is reserved for future extensions.

Tab. 4-11: Configuration parameters

The configuration parameters are applied at control startup. Parameter changes are thus only applied after control reboot.

Configuration by the Client

There are properties in the OPC UA server that are configured by the client. Which properties can be configured by the client depends on the implementation of the client. Configurable properties are shown in the following.

Properties configurable by the client

Parameters	Description
Sampling Interval	The "SamplingInterval" defines the cycle in which the server samples the subordinate data sources.
Publishing Interval	Describes the interval during which one or multiple value changes are transferred to the client for one or multiple "Monitored Items".
Queue Size / Discard Oldest	The queue size indicates the number of MonitoredItem values to be saved in one subscription. The maximum value is 1,000. The parameter "Discard Oldest" specifies how a new value is entered if the queue is full. <i>Possible values for "Discard Oldest":</i> • TRUE: The oldest value in the queue is replaced by a new value. • FALSE: The latest value in the queue is replaced by a new value.
Filter	Different filter types can be set. <i>Possible values are:</i> • DataChangeFilter • EventFilter • AggregateFilter For information on the types, refer to the respective client documentation or the OPC UA specification.

Communication Interfaces

Monitoring Mode	Indicates whether the values of a "Monitored Item" are entered into the queue and reported. <i>Possible values are:</i> • Disabled: The "Monitored Item" value is not sampled and thus not entered into the queue. No notification by this item. • Sampling: The "MonitoredItem" value is sampled and entered into the queue. No notification by this item. • Reporting: The "MonitoredItem" value is sampled and entered into the queue. No notification by this item.
Enable Publishing	This parameter determines whether subscription data is transferred to the client.

Tab. 4-12: Possible configuration parameters by the client

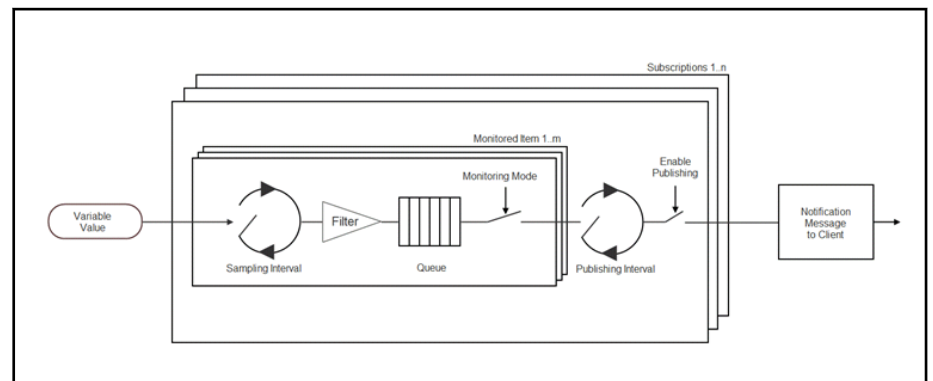


Fig. 4-30: Simplified subscription representation

PLC Namespace

The OPC UA server supports the client access to PLC data points. PLC variables can be read and changed via "Read" and "Write Services". The "Publish" service provides an efficient mechanism to inform a client via value changes of PLC variables. The PLC namespace can be analyzed using the "Browse Service". Note that the PLC namespace is based on the symbol configuration of the PLC program. Only symbolically exported data points are mapped in the OPC UA server namespace (see symbol configuration of the IndraLogic documentation).

Central entry point to the PLC namespace is the "Logic" node

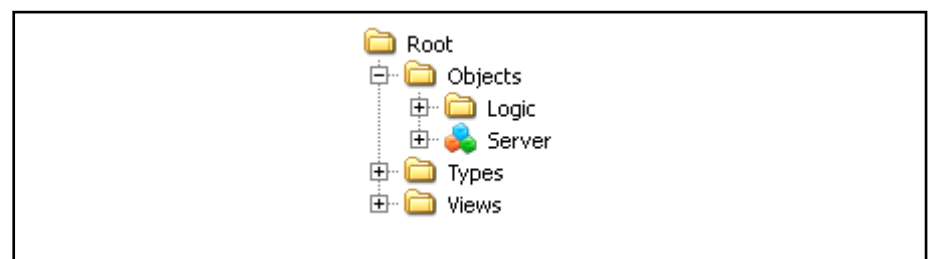


Fig. 4-31: Root node "Logic"

The structure under this node complies with the specification "PLCopen: OPC UA For IEC 61131-3 1.00 Companion Specification" (under <http://www.opcfoundation.org>). It corresponds to the PLC application structure.

Communication Interfaces

OPC UA and the PLCopen use a different basic data system. The types are mapped according to the following table.

Type mapping

IEC 61131-3	OPC UA "build in" data type	Extended OPC UA data types by "OPC UA For IEC 61131-3 1.00 Companion Specification"	Comment
BOOL	Boolean		8 bit data type
SINT	SByte		8 bit data type
USINT	Byte		8 bit data type
INT	Int16		16 bit data type, signed
UINT	UInt16		16 bit data type, unsigned
DINT	Int32		32 bit data type, signed
UDINT	UInt32		32 bit data type, unsigned
LINT	Int64		64 bit data type, signed
ULINT	UInt64		64 bit data type, unsigned
BYTE	Byte	BYTE	BYTE is a data type specified by the PLCopen and derived by the byte with the bit string length 8.
WORD	UInt16	WORD	WORD is a data type specified by the PLCopen and derived by the UInt16 data type with the bit string length 16.
DWORD	UInt32	DWORD	DWORD is a data type specified by the PLCopen and derived by the UInt32 data type with the bit string length 32.
LWORD	UInt64	LWORD	LWORD is a data type specified by the PLCopen and derived by the UInt64 data type with the bit string length 8.
REAL	Float	-	32 bit floating point
LREAL	Double	-	64 bit floating point
STRING	String	STRING	STRING is a data type specified by the PLCopen and derived by the string. It is a one-byte character string of variable length.
WSTRING	String	-	A UTF 8 string of variable length
DT DATE_AND_TIME	DateTime	-	64 bit data type representing the time in 100ns intervals since January 1st 1601.
DATE	DateTime	DATE	DATE is a data type specified by the PLCopen and derived from the DateTime 64 bit data type. It represents a date

IEC 61131-3	OPC UA "build in" data type	Extended OPC UA data types by "OPC UA For IEC 61131-3 1.00 Companion Specification"	Comment
TOD TIME_OF_DAY	DateTime	TOD	TOD is a data type specified by the PLCopen and derived from the DateTime 64 bit data type. It represents a time.
TIME	Double	Duration	Duration is a data type specified by the PLCopen and derived from the double 64 bit data type. It represents a time interval in milliseconds.

Tab. 4-13: Type mapping IEC 61131-3 OPC UA

4.6.4 OPC UA Communication Example

Introduction

The following section shows how to access the Bosch Rexroth OPC UA server easily.

It is described how to parameterize a connection and how the different services of the server can be used.

Prerequisites

To simulate the communication sample, the UaExpert of the Unified Automation GmbH is required.

The UaExpert is implemented for the Win32 and Linux platforms and can be downloaded as freeware via <http://www.unified-automation.com>.

This documentation refers to version 1.2.0 132 of the UaExpert.

UaExpert

UaExpert is a universal OPC UA test client. The OPC UA server address space can be analyzed with the UaExpert and simple tests can be conducted with the OPC UA server.

The UaExpert supports

- browsing address spaces,
- reading and writing of nodes (value attribute),
- monitoring value changes and events (subscription),
- calling methods,
- reading historic data

The UaExpert is divided into several windows.

Communication Interfaces

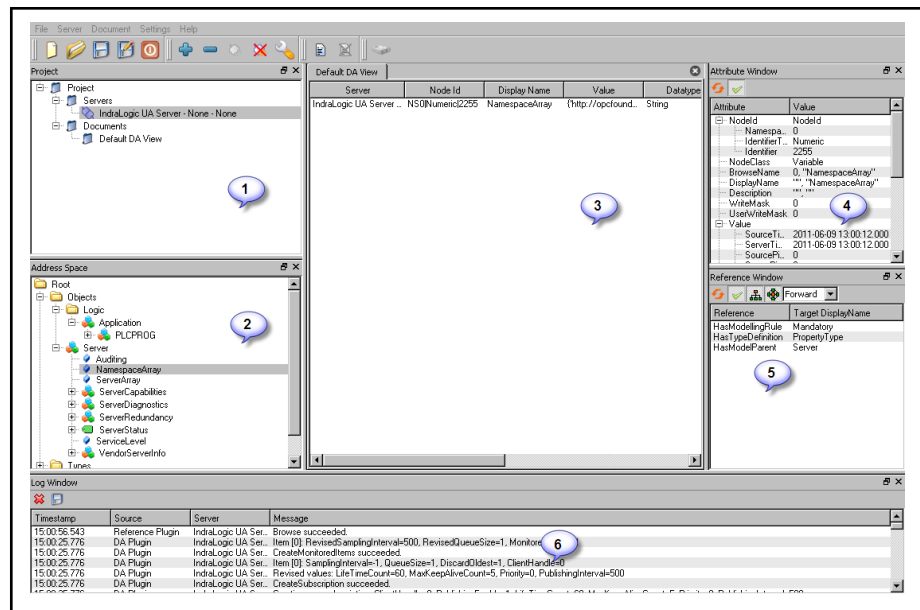


Fig. 4-32: UAExpert

The subwindows have the following significance:

- ① Project Window. Manages server connections and documents.
- ② Address Space Window. Allows navigation and browsing in the address space of the server selected in the Project Window.
- ③ Data View Window. Maps server data.

The following views are supported:

– DataAccess View

Describes the content of a subscription. All nodes monitored for a value change are listed.

– Alarms&Condition View

Events and alarms triggered by the server are configured and displayed. Alarms and events are not supported in the current Bosch Rexroth OPC UA server version.

– HistoricalTrend View

The chronological value change can be displayed or parameterized as diagram for one or several nodes.

– Performance View

Used to analyze the OPC UA server performance. The Read, Write, CreateMonitoredItem and DeleteMonitoredItem services are supported.

- ④ Attribute Window

Displays all attributes of the node selected in the Address Space Window.

- ⑤ Reference Window

Displays all references of the node selected in the Address Space Window. It is differentiated between forward references (linking from this node) and inverse references (linking to this node).

- ⑥ Log Window

Diagnoses communication problems.

Establishing a Connection

An end point has to be specified to establish a connection to the OPC UA server. The end point consists of the communication protocol, the host address and the security settings.

To define an end point for an OPC UA server in UaExpert, select **Server ► Add...** in the menu. In the dialog, enable the "Discovery" tab und select "<Double-Click to Add Server...>" in the tree view under "Custom Discovery".

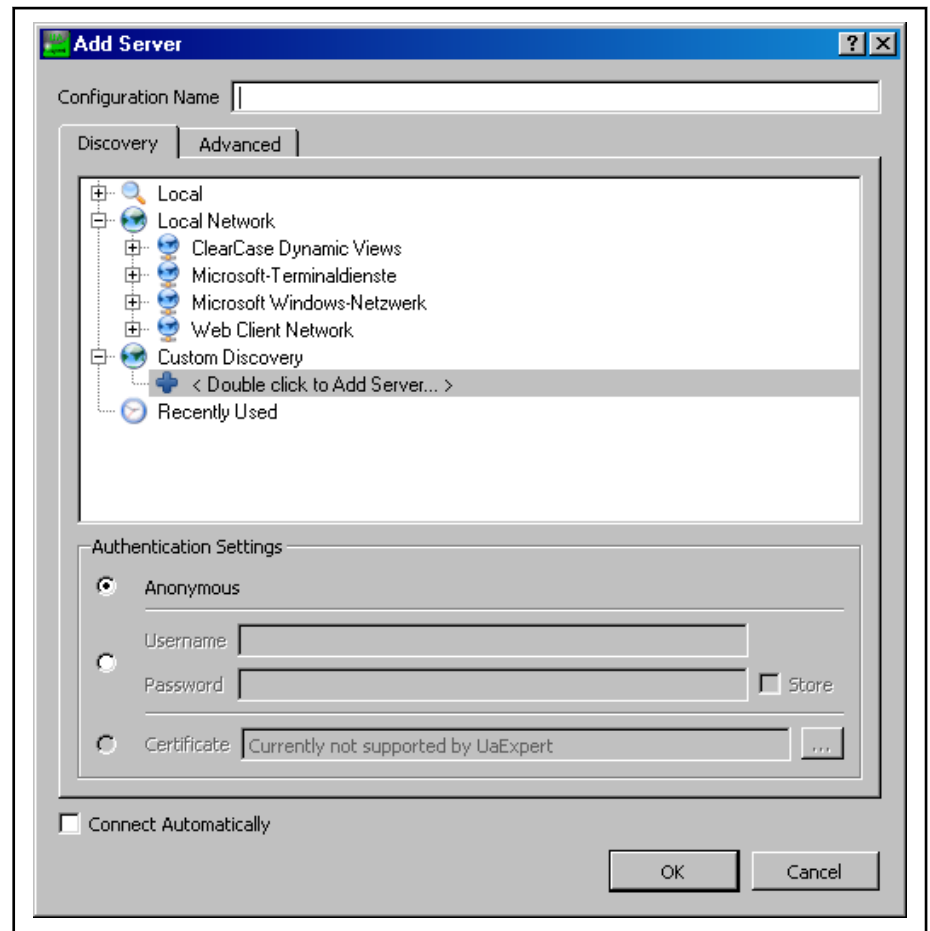


Fig. 4-33: Dialog: Add Server

Double-click on the entry to go to the entry mask of the communication protocol and the host address. The communication protocol is already assigned as UA binary protocol by the "opc.tcp://" entry in the input line. Add the IP address or the host name for the OPC UA server. The port number, separated by a colon, can also be specified. The Bosch Rexroth OPC UA server uses the port 4840.

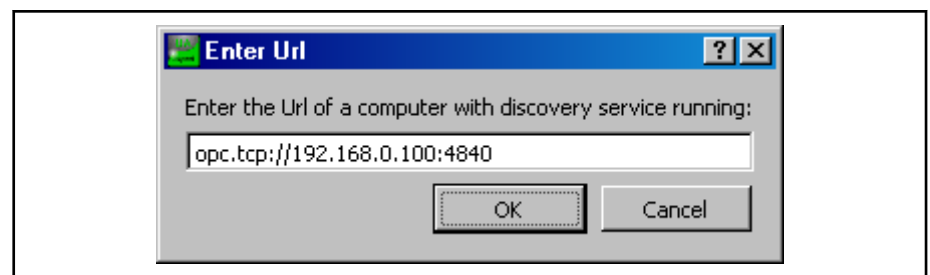


Fig. 4-34: Entering EndpointUrl

Return to the previous dialog with **Ok**.

Communication Interfaces

The EndpointUrl is now displayed under "Custom Discovery".

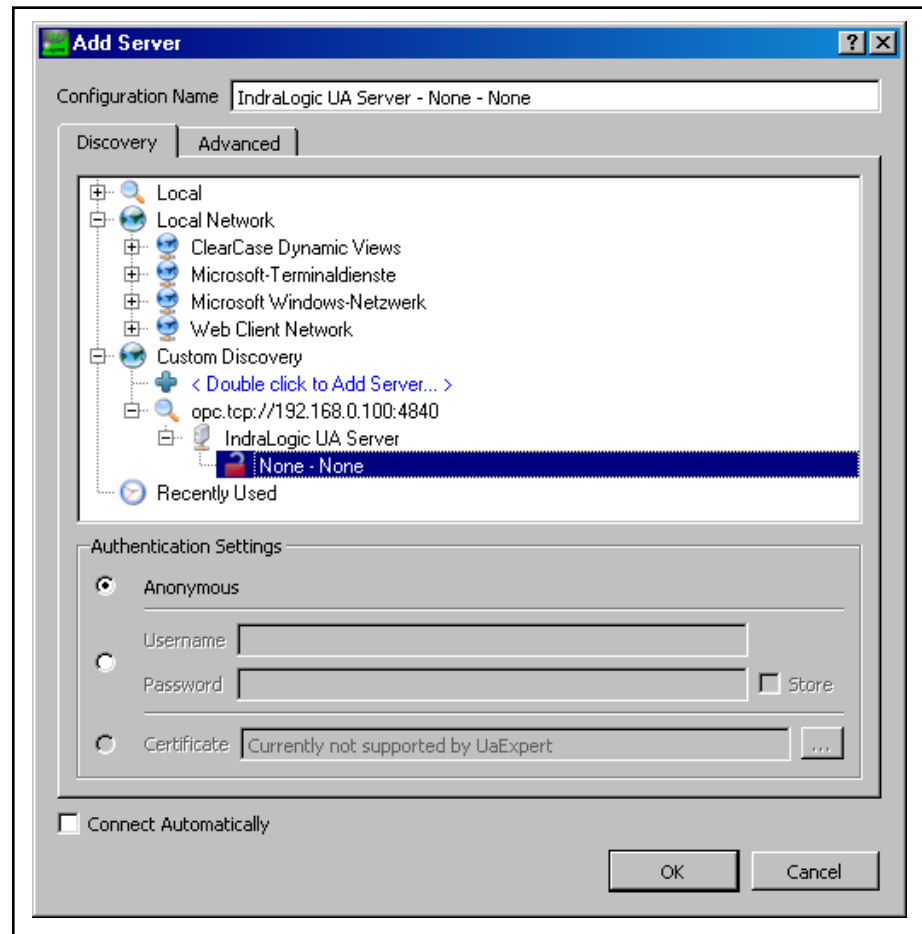


Fig. 4-35: Selecting "No Security Endpoint"

If a connection is established to the server, the entry can be expanded as tree. The ServerUrl "IndraLogic UA Server" and the security end points should be visible. The Bosch Rexroth OPC UA server does not support any security. Thus, only the "None - None" entry exists. Click on this entry and close the dialog.

Browsing

The ServerUrl is now visible for the newly parameterized connection in the Project Window below the "Server" node. Establish a connection to this OPC UA server via the context menu of this entry or via the "Server Connect" menu item.

After an established connection, the address space of the OPC UA server is displayed in the Address Space Window.

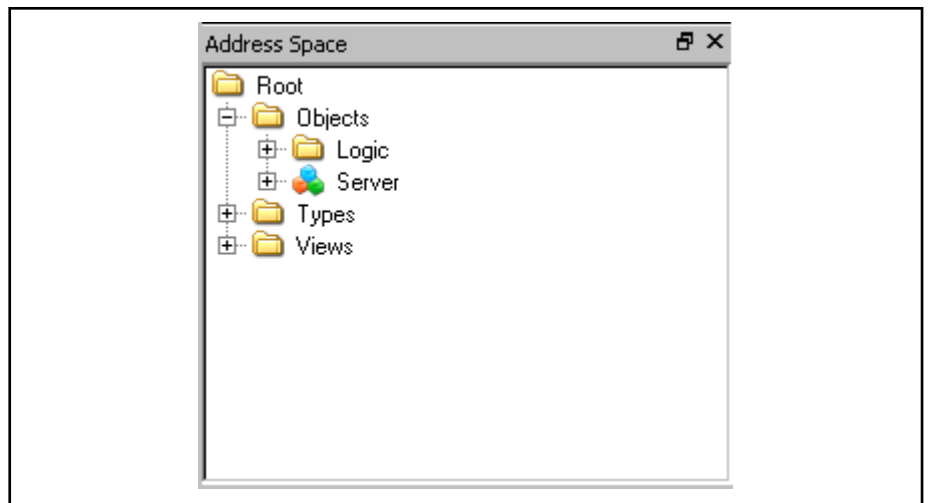


Fig. 4-36: Address Space Window

The entries in the tree view are OPC UA nodes. These nodes can now be expanded and thus facilitate a navigation in the address space of the OPC UA server. All nodes in the address space are displayed.

Click to select a node and all node attributes are listed with their in the Attribute Window. The values in this window are read-only (a client cannot change these values).

All references from the node (forward) and all references to this node (inverse) can be read in the Reference Window.

Subscribe and Write

The UaExpert supports the mechanism to monitor changes in data (subscription) in the Data View Window. Create one or multiple of these views via the menu **Document - Add** and by selecting "Data Access Classic View" in the "Add Document" dialog.

"Value" attributes monitored on value changes by the server are named "Monitored Item". To generate a "Monitored Item", the node in the "Address Space Window" has to be dragged to the "Data View Window".

Communication Interfaces

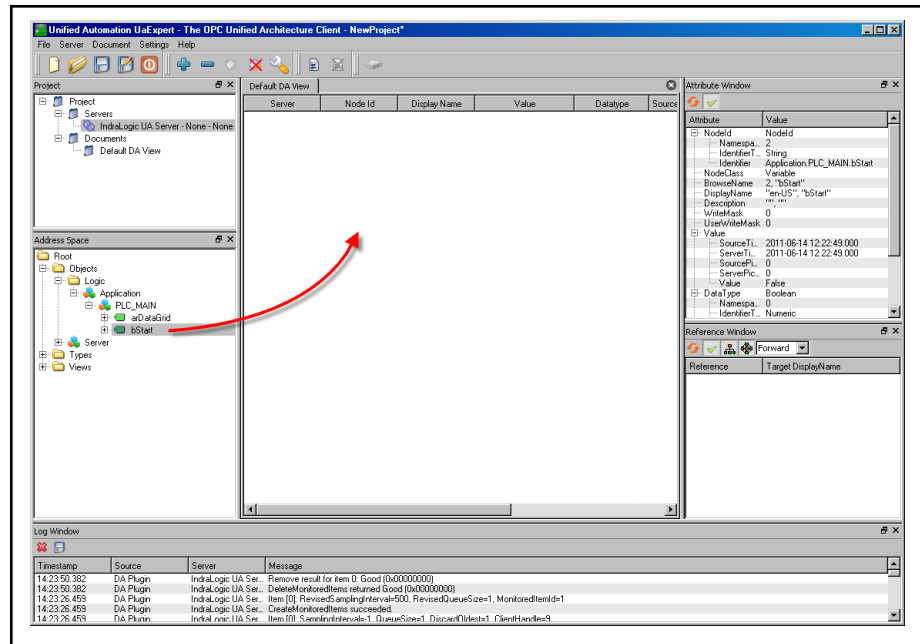


Fig. 4-37: Create Monitored Item

The attribute "Value" is automatically used as Monitored Item by UaExpert. Thus, the nodes have to be part of the node class "Variable".

The node value is now displayed and regularly updated in the Data View Window. A "Data Access Classic View" can be viewed as subscription. For the dialog to set the subscription properties, go to the context menu of this window. Right-click on the Data View Window and select "Subscription Settings..."

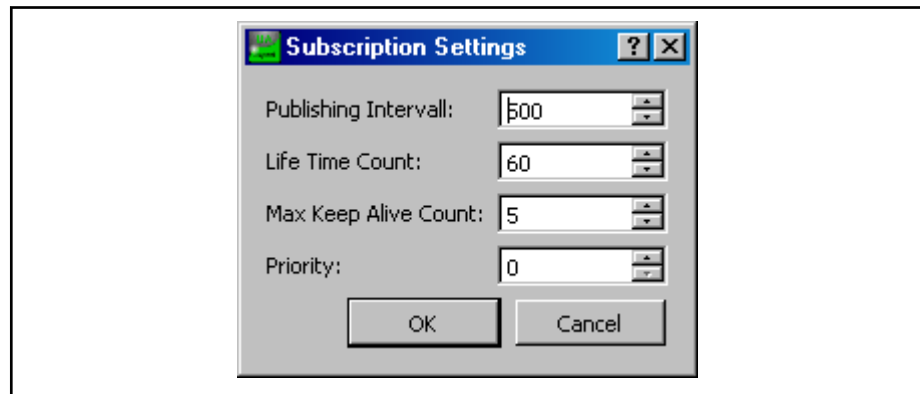


Fig. 4-38: Dialog: Subscription Settings

The following settings can be selected in the dialog. Note that all values are "negotiated" between client and server.

Setting values for a subscription

Publishing Interval	Describes the interval in milliseconds during which the "Monitored Items" are applied to the client in case of changes. If the Monitored Item values do not change within the publishing interval, no data is transferred to the client. The Bosch Rexroth OPC UA server uses fixed publishing intervals: 10 ms, 100 ms, 200 ms, 500 ms, 1.0 s, 2.0 s, 5.0 s.
Life Time Count	Monitors the client-server connection. Indicates the number of publishing intervals without value transfer to the client until all monitored items of this subscription and the subscription are deleted on the server. The "Life Time Count" has to be at least three times the "Max Keep Alive Count"
Max Keep Alive Count	Indicates the number of publishing intervals without transferring the value changes from the server. The server sends an "empty" message to the client. Thus, the client knows that the subscription is still active in the server.
Priority	A priority (the higher the value, the higher the priority) can be assigned to a subscription via this value. If several subscriptions are active in a server, subscriptions with a higher priority are also processed with priority.

Tab. 4-14: Subscription settings

Further setting options refer to the Monitored Items. To select the settings, right-click on the Monitored Item and select "Monitored Items Settings...". The values are also "negotiated" between client and server.

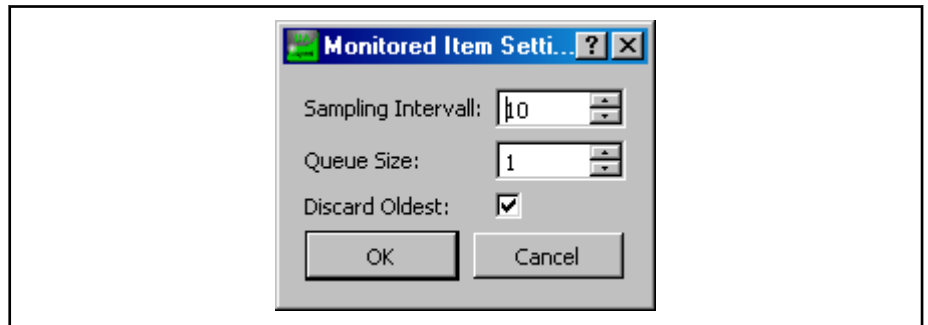


Fig. 4-39: Dialog: Monitored Item Settings

Communication Interfaces

Setting values for Monitored Items

Sampling Interval	Specifies the the server samples the subordinate data source to determine a value for the Monitored Item. If the value is set to -1, the publishing interval value is set.
Queue Size	Using this entry, the queue size can be set for the Monitored Item. These values are only entered into the queue if values change. All queue values are transmitted to the client within the publishing interval.
Discard Oldest	Describes the rule for the server response if the queue is full. If the value is set, the oldest value of the queue is replaced by the new value. Otherwise, the latest value is replaced. If the queue size is 1, this parameter is not significant.

Tab. 4-15: Monitored Item settings

To write a value, double-click to select the "Value" column of a "Monitor Item". The value can now be edited. Click on "Enter", and the new value is written to the server.

4.7 Configuring OPC UA Client Connection Using WinStudio

To configure a connection to an OPC-UA server, the corresponding driver sheets are used in WinStudio. Driver sheets include the variables and communication parameters to be communicated. Multiple driver sheets can be created for each server configured in WinStudio. Communication parameters (update rates, variables to communicate, etc.) can be individually specified for each created driver sheet. Each driver sheet is a subscription.

To configure an OPC-UA connection to an OPC UA client, proceed with the following two steps:

- 1. Configure the OPC-UA connection
- 2. Configure a driver sheet. As already mentioned, multiple driver sheets can be created.

A) Configuring an OPC-UA connection

To communicate with a device via OPC-UA server, configure the connection to this server. Proceed as follows:

- 1. Open the context menu of the "OPC-UA" folder and select "Configure OPC-UA Connections".

A dialog displays all configurations created. To create a new connection to the OPC-UA server, click on "Add".

The dialog "UA Server Configuration" is displayed.

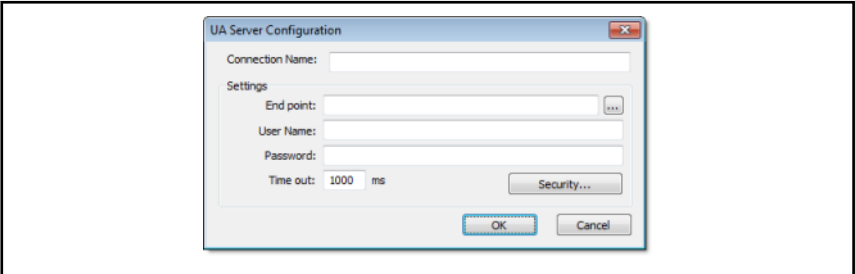


Fig. 4-40: "UA Server Configuration" dialog

- 2. Enter a connection name into the field **Connection Name**.

Communication Interfaces

- Enter the URL of the OPC-UA server to connect with into the **End point** field. Click on ... to open the "UA Discovery" server dialog.

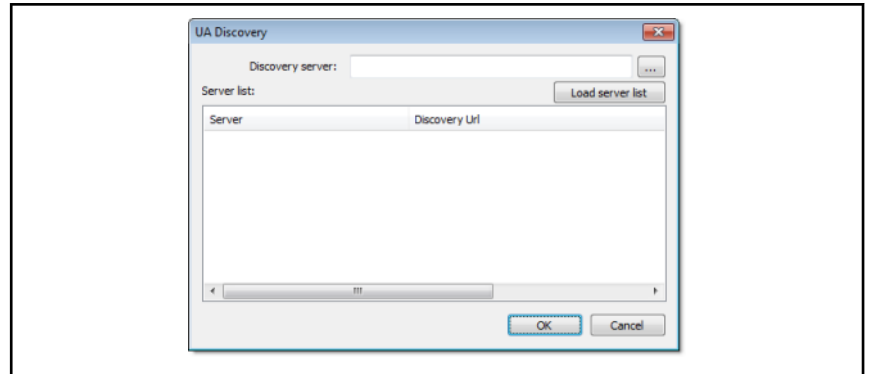


Fig. 4-41: "UA Discovery" dialog

Use this dialog to find the "Discovery Server" publishing a list with OPC-UA servers in the network and select the target server for the connection. To connect to a Rexroth control, enter the IP address of the control. Click on **Load server list** to update the server list. Select the control from the server list and confirm with **OK**.

B) Configuring an OPC-UA driver sheet

To create a new driver sheet for the OPC-UA client, proceed as follows:

- Open the context menu of the node **WinStudio** ► **Variables** ► **Communication** ► **OPC-UA** and select "New Driver Sheet"

A new element is added to the OPC-UA folder. Double-click to open the driver sheet editor:

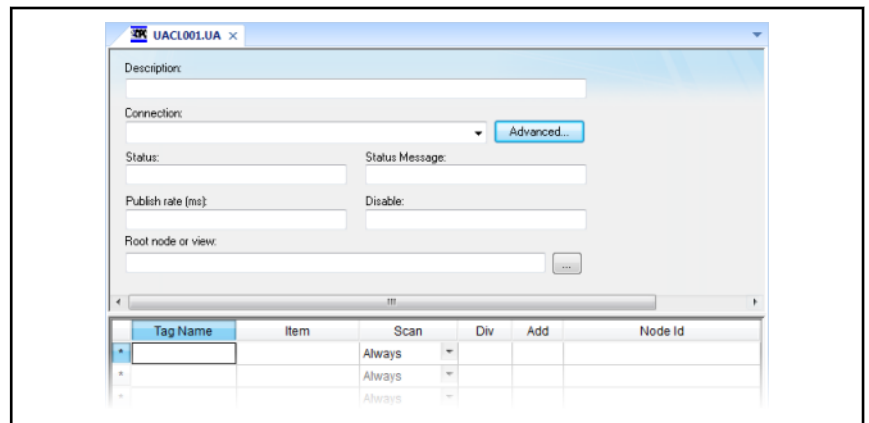


Fig. 4-42: "OPC UA Client" worksheet

- Enter a worksheet description into the **Description** field. Only used for documentation purposes.
- Select the server connection configured in A) from the **Connection** list and save the driver sheet.
- Set **Publish rate (ms)**:. The driver sheet variables are updated in this interval at runtime
- The variables to be communicated can be selected.

Two options are available:

- Using the WinStudio OPC UA browser: There has to be a connection to the control

Right-click on **Element** and select **OPC Browser**. The controls from the respective OPC configuration are displayed. The variables in

Communication Interfaces

the symbol configuration of the respective control are shown in a tree structure.

- Using IndraWorks tool window for process variables: No connection to the control required

Enable the tool window via **View ► Other Windows ► Process Variables** in the IndraWorks toolbar.

If the configuration of the OPC driver sheet was saved, the symbol variables of the controls available in the project are displayed. Drag and drop them to the driver sheet line. If the control is not in the project, load the symbol configuration file of the external control and thus project them to the HMI project. For more information on the tool window for process variables, refer to the IndraWorks on-line help

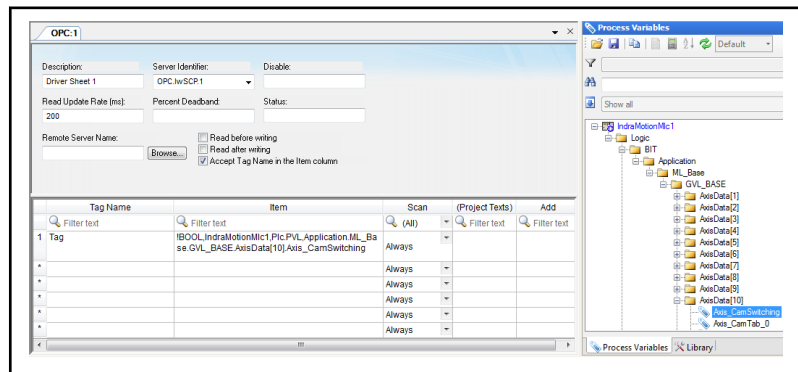


Fig. 4-43: OPC driver sheet configuration and tool window for process variables

For more connection options as well as for a detailed description of the advanced settings, refer to the IndraWorks online help "Rexroth WinStudio 7.3 Technical Reference"

4.8 Modbus TCP

4.8.1 Introduction and Overview

The Modbus TCP exchanges data between Ethernet devices supporting the Modbus protocol. It is especially suitable to connect multiple HMI devices to a control. The Modbus slave (server) is the control used for the Modbus masters (clients) to connect.

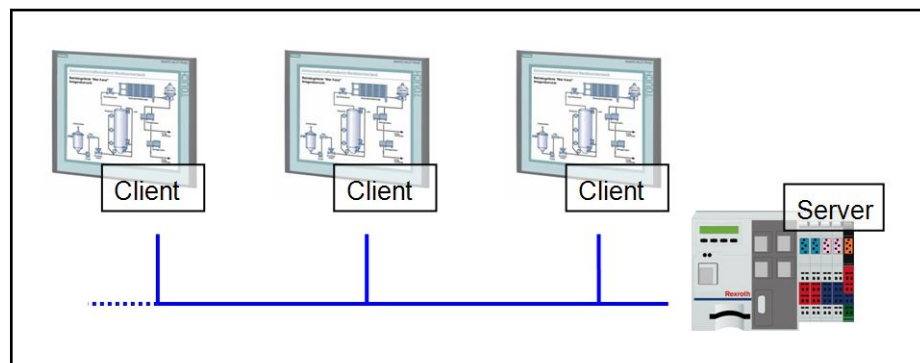


Fig. 4-44: Connecting several HMI devices (Modbus clients) to a control (Modbus server)

Modbus TCP differentiates between four different data types:

- Discrete input: Bit-addressed binary input (array of Boolean)

Communication Interfaces

- Coil: Bit-addressed binary output (array of Boolean)
- Input Register: Word-based output (array of word)
- Holding Register: Word-based output (array of word)

Transmitting medium	Ethernet
Protocol	Modbus TCP on TCP-Port 502
Max. number of HMI devices	16 per Modbus TCP server
Transmission efficiency	1000 I/O operations per second for 1ms cycle time
I/O area	65536 Bit Discrete Input 65536 Bit Coil 65536 Word Input Register 65536 Word Holding Register

Tab. 4-16: Modbus TCP properties

The IndraMotion MLC and IndraLogic XLC controls support the following Modbus TCP-compatible HMI devices:

- Siemens WinCC-flexible 2007
- Pro-face GP-Pro EX2.2.

Devices of other manufacturers might also be supported (not yet tested).

A Modbus TCP server is implemented in each of the IL_ModBusTCPServer and IL_ModBusTCPServerType02 function blocks. The RIL_ModbusTCP.library contains these function blocks.

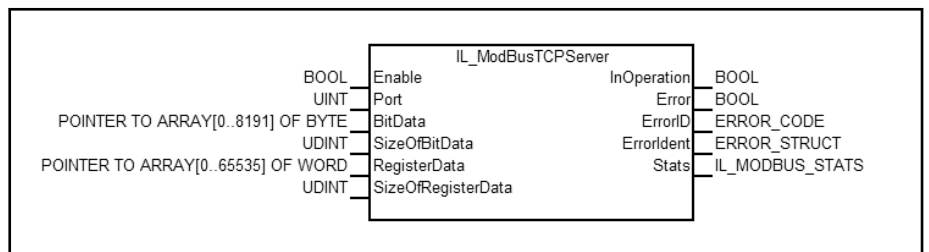


Fig. 4-45: "IL_ModBusTCPServer" function block

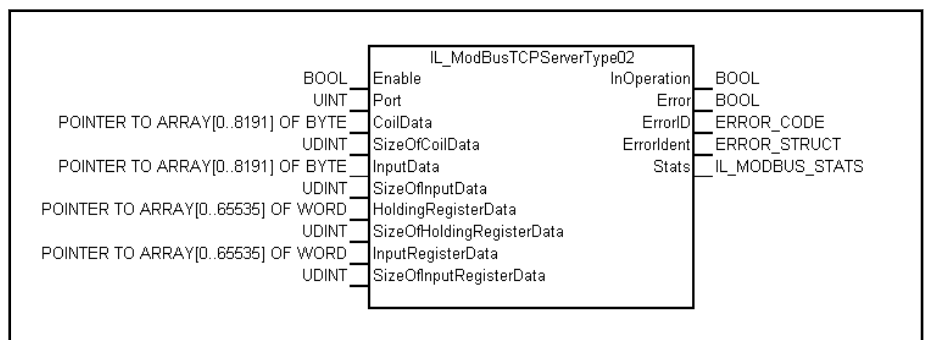


Fig. 4-46: "IL_ModBusTCPServerType02" function block



The functions blocks "IL_ModBusTCPServer" and "IL_ModBusTCPServerType02" differ in the following:

In contrast to the Modbus specification, the data ranges for "Coil Data" and "Discrete Input Data" are summarized under "BitData". The "Holding Register" and "Input Register" are summarized under "RegisterData" in the "IL_ModBusTCPServer" function block.

Communication Interfaces

For a more detailed description of the Modbus TCP functionality, refer to

- Rexroth IndraWorks xxVRS, Basic Libraries IndraLogic 2G

xx = Version

5 Application Examples

5.1 Overview

This chapter describes simple sample applications for individual visualization types. The short character of the samples is intentional. For more detailed information, please refer to the corresponding documentations of the visualization systems.

5.2 Symbol Configuration

5.2.1 General Information

In order for an OPC client to access the PLC variables of a control, the variables required by the OPC client or the variables provided for access by the PLC project have to be configured in the symbol configuration of IndraLogic.

The variables can be configured using the following access authorizations:

- No access
- Read
- Write
- Read and write

Symbol configuration can be carried out with a wizard via the "Symbol Configuration" window, via "pragma directives" in the variable declaration section of the POU or via variables lists.

A prerequisite for symbol configuration is that the PLC object "Symbol configuration" exists below the PLC application in the IndraWorks project tree. The object listed in the following table from the "PLC Objects / General Objects" library window has to be added below the "Logic / Application / HMI" device node using drag & drop. The following figure shows the Project Explorer once the object has been added.

Object type	Name	Description
Symbol configuration	"_"	Configuration of the variables for access via the OPC server with access rights.

Tab. 5-1: Adding PLC library object "Symbol configuration"

Application Examples

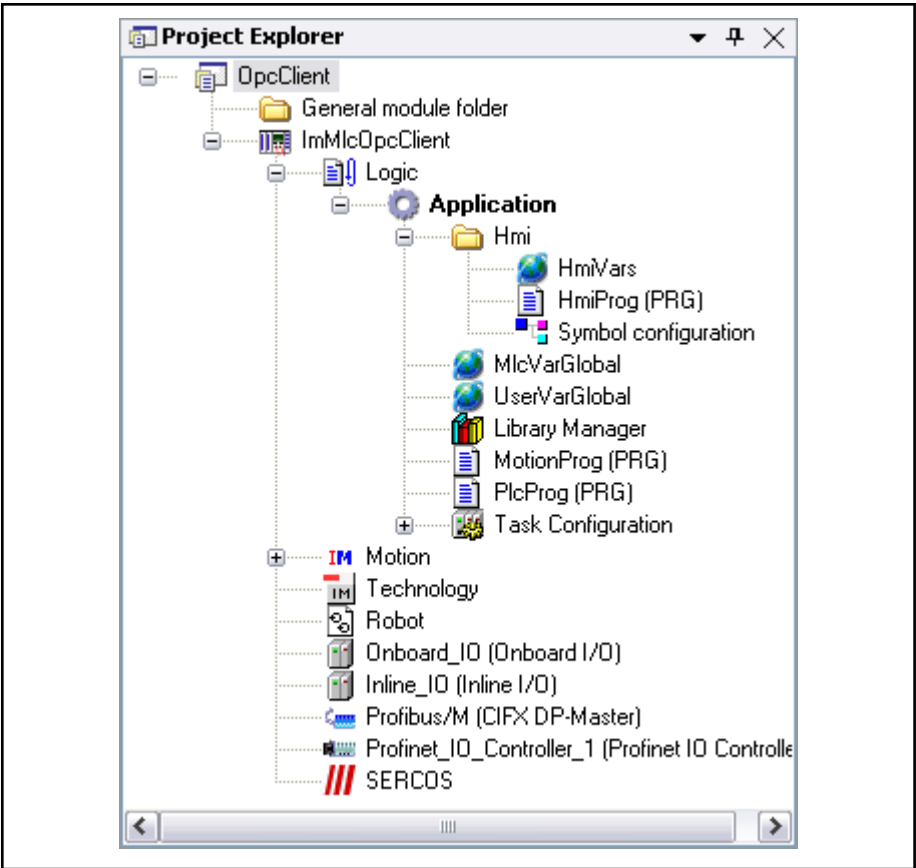


Fig. 5-1: Added library object "Symbol configuration"

5.2.2 Symbol Configuration with Wizard

The symbol configuration wizard is opened via the newly created "Symbol Configuration" node in the IndraWorks Project Explorer.

The symbol configuration window is divided into two areas:

Available variables	List of all of variables in the PLC program with the respective maximum access authorization level. If no variables are listed, use the "Refresh" link to update the list. Note that it has to be possible to compile the project without errors.
Selected variables	List of all variables which were configured for access via the OPC server, with the respective configured access authorization. Select the "Display invalid variables" link to highlight all variables that no longer exist in the PLC project.

Tab. 5-2: Structure of the "symbol configuration wizard"

Select or delete variables by double-clicking on the element or by using the ">" and "<" buttons. This selects or removes the selected element and all subordinate elements.

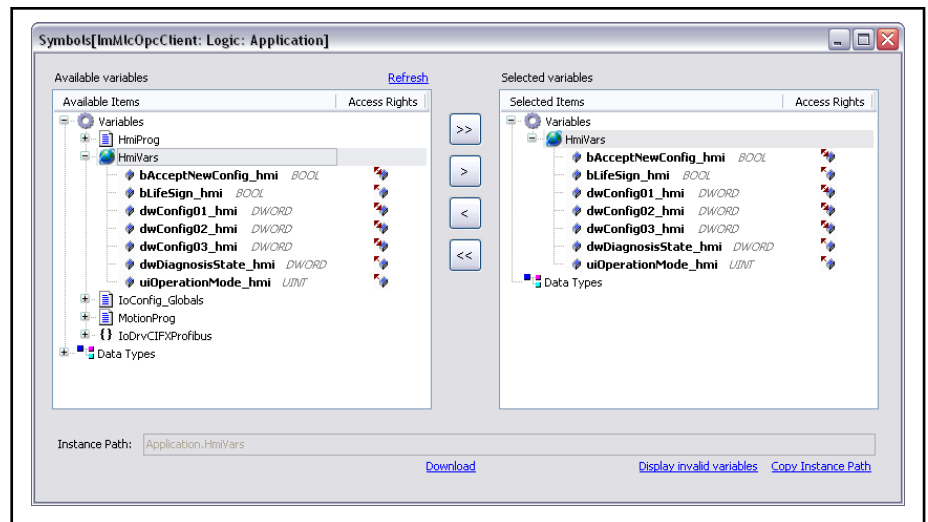


Fig. 5-2: Symbol configuration wizard



The "Symbol configuration" window lists only those variables which are used in the PLC program (POU), if this is called in a task. This means that variables which are not used are not in the project tree.

The access rights for the respective variable can be changed by clicking on the symbol in the column "Access rights" in the section "Selected variables". Access rights are switched in the following order:



Read and write access

Write access

Read access

5.2.3 Symbol Configuration via Pragma

Symbol configuration using "pragma directives" is carried out directly in the declaration area of the variables, e.g. in variable lists or in POU's.

The {attribute 'symbol'} attribute with the parameters for access authorization is available for pragma configuration. The possible configurations are shown in the following figure.

Pragma directive	Description
{attribute 'symbol' := 'none'}	Variables are blocked for access and are not displayed in the OPC server browsing list.
{attribute 'symbol' := 'read'}	Variables are only enabled for read access and are displayed in the OPC server browsing list.
{attribute 'symbol' := 'write'}	Variables are only enabled for write access and are displayed in the OPC server browsing list.
{attribute 'symbol'} {attribute 'symbol' := 'readwrite'}	Variables are enabled for read and write access and are displayed in the OPC server browsing list.

Tab. 5-3: Pragma directives for "symbol configuration"

Depending on the position or the line in which the Pragma is written, it is valid for all variables in the declaration range or for just one variable.

Application Examples

- In order for the pragma to affect only a single variable it has to be written into the line before the variable.
- If it is to affect all variables in the declaration range, it has to be in the first line of the declaration. In this case individual variables can still be assigned a different access authorization.

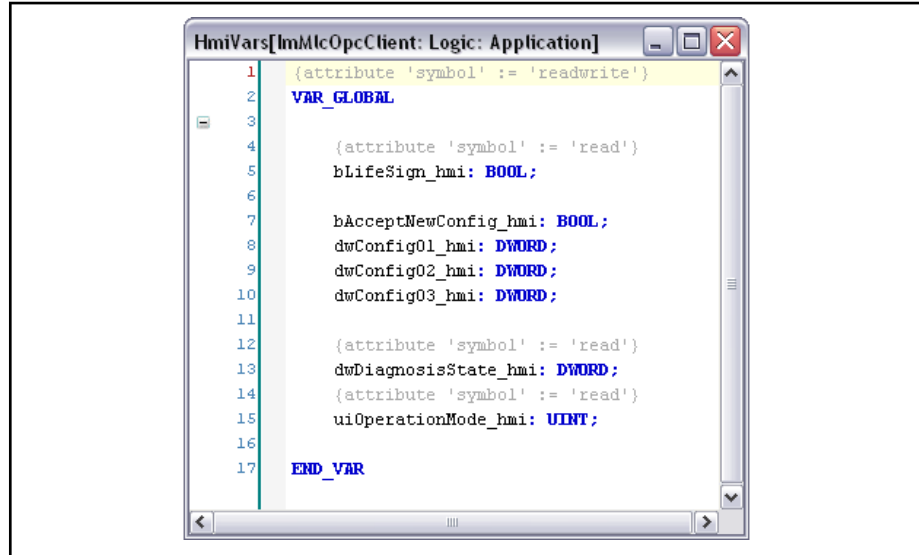


Fig. 5-3: HMI variable declaration with pragma



If symbol configuration is carried out via pragma directives these symbols are **not** displayed in the symbol configuration dialog.

When the PLC application is downloaded, the symbol information is created in the IndraWorks "IndraLogic" project subdirectory as an XML file and is loaded into the control as a symbol application.

5.3 VI-Composer

In preparation.

5.4 WinStudio with IndraLogic XLC

5.4.1 General Information

The aim is to create a WinStudio interface that can be used to control a drive and display the control status information.

This project is a first step to using "WinStudio" in connection with IndraLogic XLC or IndraMotion MLC. The "WinStudio" interface is created step by step.

Subtasks Task description:

- Creating an IndraWorks project in which a drive is operated,
- Integrating WinStudio in IndraWorks,
- Creating the WinStudio application to visualize data of the control project (operating panel on the control).
- Start the interface.

5.4.2 Technical Requirements

Preconditions

- PC with installed IndraWorks from version 13,
- IndraLogic XLC or IndraMotion MLC as control with sercos interface,
- IndraDrive C (MPx-07VRS) as drive,
- WinStudio V 7.3 under IndraWorks,
- IndraWorks Engineering 13Vxx.

Meeting these prerequisites is described in

- Rexroth IndraLogic XLC 13VRS Functional Description, DOK-IM*ML*-SYSTEM**V13-RExx-EN-P and
- IndraWorks online help, **System IndraLogic XLC/IndraMotion MLC ▶ XLC/MLC Functional Description ▶ Commissioning and Operation**,

HMI, type VEP xx.4/5 or VPP xx.3

- The current firmware is loaded.
- The IP address is known.
- The control and the HMI are connected to the programming device (PC) via Ethernet.

Prerequisites:

- A bridged sercos bus if there is no drive. Use a virtual axis instead of a real axis in this case.

5.4.3 Creating the IndraWorks Project

Start the IndraWorks Engineering environment using the icon on the desktop or the respective selection in the Windows program folder.

To create a new IndraWorks project, select

File ▶ New ▶ Project...

The dialog "Create New IndraWorks Project" is displayed.

Enter the name "WinStudio" (any name accepted by IndraWorks can be chosen) for the project and select a directory to store the project in.

Insert the "IndraMotion XLC Lxx" control into the project by dragging the device from the library to the project. Please note that the device can only be dragged directly into the project folder.

Application Examples

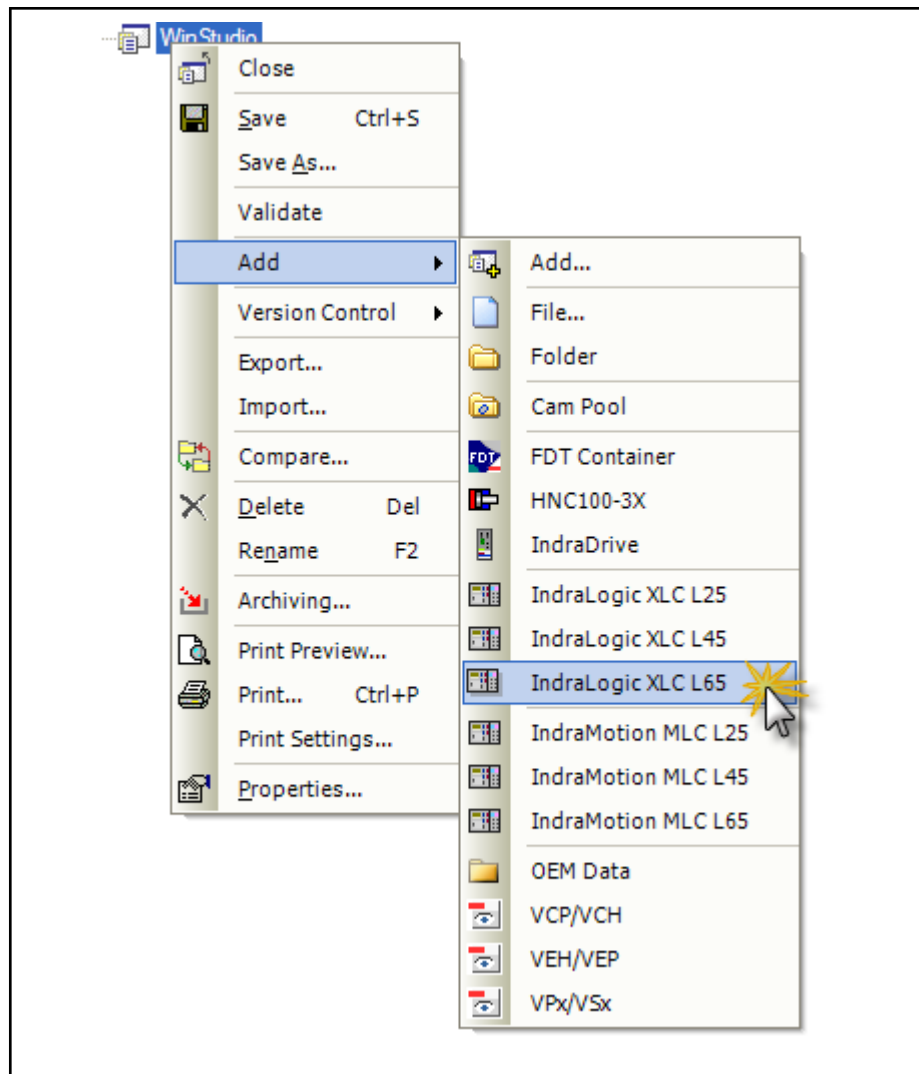


Fig. 5-4: Selecting the IndraLogic XLC L65 control

Specify your device name (IndraWorks will notify you if any characters chosen are invalid) or use the one already specified.

Click on "Next" to make the communication settings and hence, establish a connection to the control.

The settings under "Firmware version" and "Firmware release" have to be adapted to the control.

The option "CML65.1-3P" has to be selected under device type.

Enter the IP address of your control into the field **IP address**. The IP address of the control can be found on the IndraLogic XLC VPP under "Shortcut to RxSettings".

If the option "Safe Online Mode" is enabled, another prompt is displayed before any safety-relevant activities can be carried out.

Subsequently, a **connection test** to the control should be carried out.

If the connection test was successful, determine which drives can be operated at your control.

Open the context menu of the folder "SERCOS" **Scan bus configuration**.

"Drive1" was detected on the figure below.

Application Examples

By double-clicking on the symbol in column IPO (1), editing of the motion functionality in the control was selected.



Interpolation in the control  or drive  :

- Switching by double-clicking on the icon
- The IndraDrive (interpolation in the drive) becomes a SerCosDrive (interpolation in the control)

Its name was changed to "RA1" (2).

"Create Drives" (3) creates the drive in the Project Explorer.

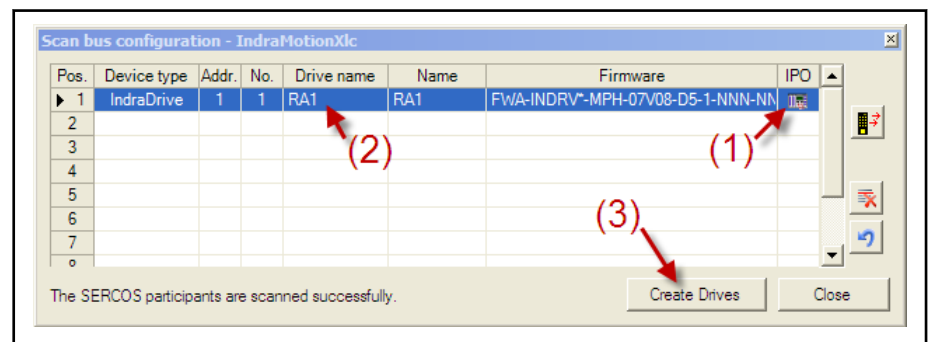


Fig. 5-5: Drive scanning result

If control and drive have been inserted into the project, you should see the following in IndraWorks.

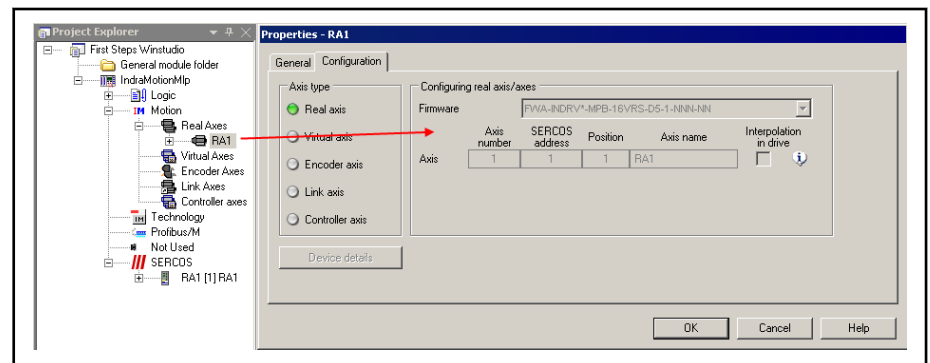


Fig. 5-6: IndraWorks – Control and drive configured correctly

Create a control program in structured text under **Logic** ► **Application** ► **MotionProg (PRG)**. A double-click on the "MotionProg" file automatically opens IndraLogic in the IndraWorks middle part.

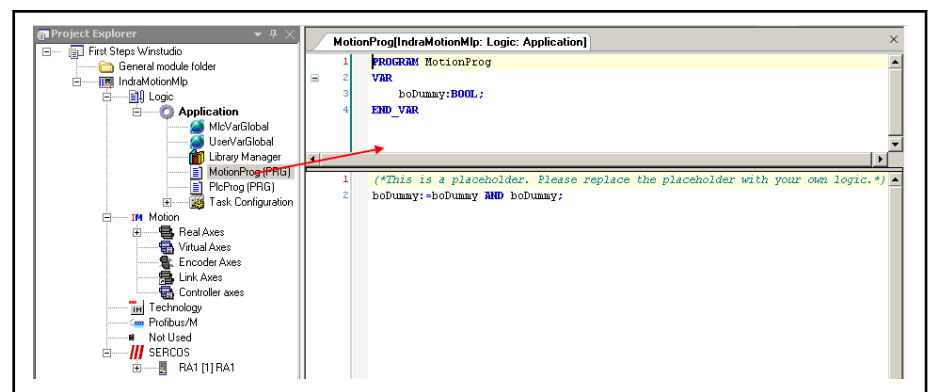


Fig. 5-7: IndraLogic – MotionProg

Application Examples

Delete the code predefined in the file from the editor.

Insert the following function blocks into the program: MC_Power, MC_MoveVelocity, MC_Stop and MC_Reset.

Start with MC_Power. Use the F2 input assistance.

Enter the instance name (PowerRA1) in the declaration part and select type (User-defined Types ► ML_PLCopen ► POU ► Motion ► MC_Power).

Disable the option "Insert with Arguments".

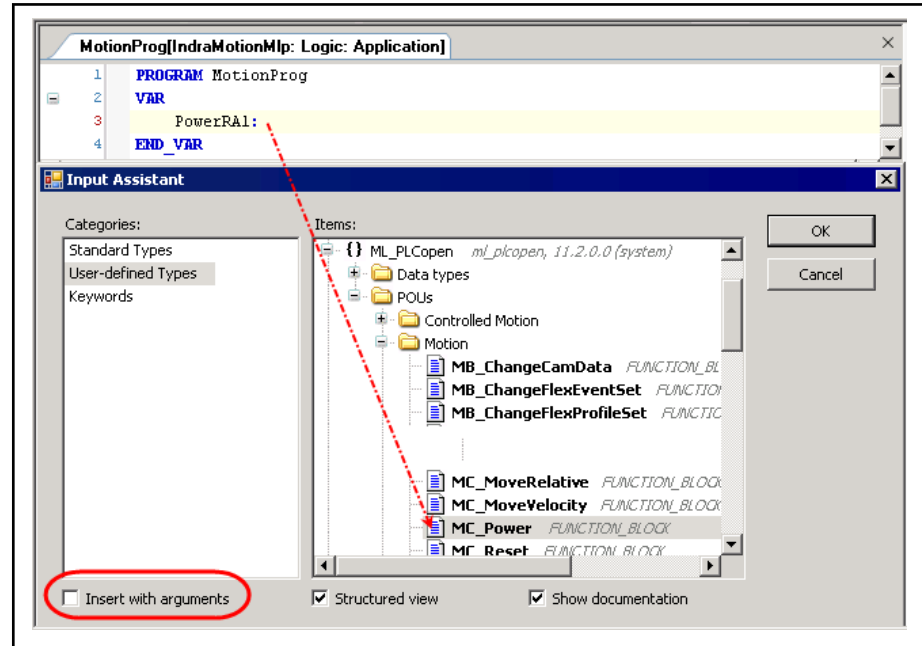


Fig. 5-8: F2 input assistance, declaration, selection of the function block type (without argument)

Add ";" at the end of the declaration line.

Go to the implementation part and open the F2 input assistance (Instance calls ► PowerRA1).

Enable the option "Insert with Arguments".

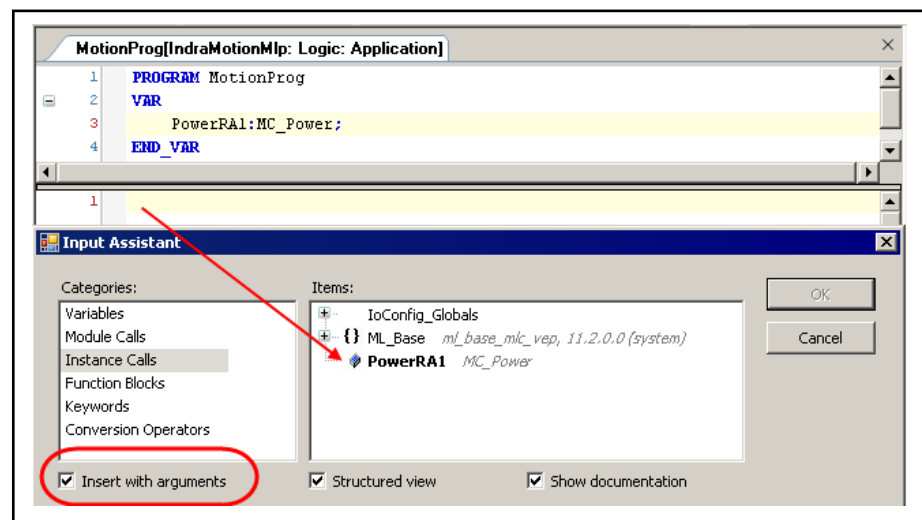


Fig. 5-9: F2 input assistance, implementation, selection of the function block instance (with arguments)

We will provide the template of the block for wiring.

Application Examples

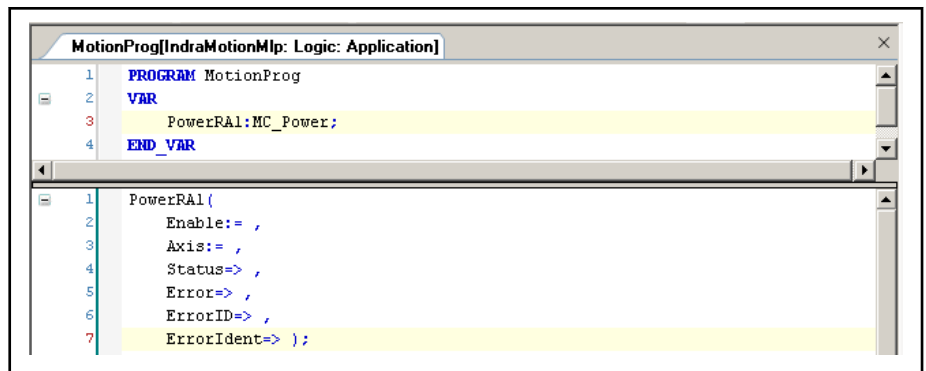


Fig. 5-10: Implementation, template of the PowerRA1 block: MC_Power

Please assign "Enable" to "en_PowerRA1". When exiting the line, a declaration window is displayed in the Editor.

Please assign "Axis" to the corresponding axis reference. Use the F2 input assistance again.

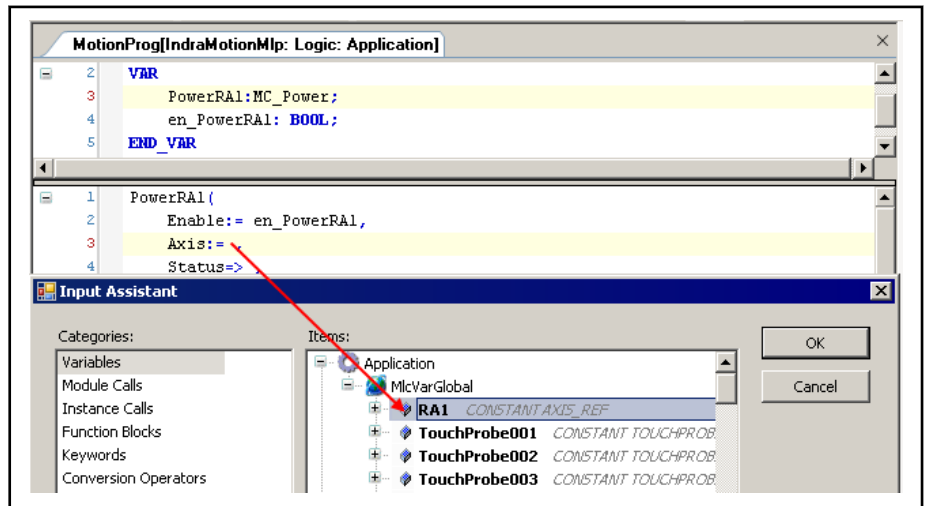


Fig. 5-11: F2 input assistance, area of the global constants to be selected

The axis reference is entered as Global Variable (constant) under the name of the variable list and the axis description.

Complete all other variables.

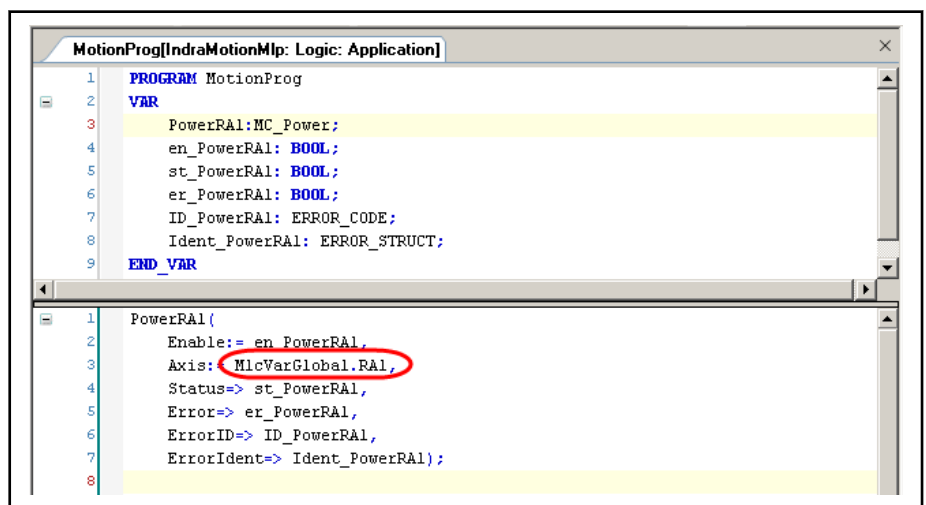


Fig. 5-12: PowerRA1: MC_Power: Complete in declaration and implementation.

Application Examples

Insert the following blocks into the program: MC_MoveVelocity, MC_Stop and MC_Reset.

```

1 // Power on, RA1
2 PowerRA1(
3     Enable:= en_PowerRA1,
4     Axis:= MlcVarGlobal.RA1,
5     Status=>st_PowerRA1,
6     Error=> er_PowerRA1,
7     ErrorID=>ID_PowerRA1,
8     ErrorIdent=> Ident_PowerRA1);
9 // Run RA1 with given velocity
10 MoveVelRA1(
11     Execute:= ex_MoveVelRA1,
12     Velocity:= 300,
13     Acceleration:= 20,
14     Deceleration:= 20,
15     Axis:= MlcVarGlobal.RA1,
16     InVelocity=> in_MoveVelRA1,
17     Active=> ac_MoveVelRA1,
18     CommandAborted=> ca_MoveVelRA1,
19     Error=> er_MoveVelRA1,
20     ErrorID=> ID_MoveVelRA1,
21     ErrorIdent=> Ident_MoveVelRA1);
22 // Stop RA1 with given deceleration
23 StopRA1(
24     Execute:= ex_StopRA1,
25     Deceleration:= 20,
26     Axis:= MlcVarGlobal.RA1,
27     Done=> do_StopRA1,
28     Active=> ,
29     CommandAborted=> ,
30     Error=> er_StopRA1,
31     ErrorID=> ID_StopRA1,
32     ErrorIdent=> Ident_StopRA1);
33 // Clear error, RA1
34 ResetRA1(
35     Execute:= ex_ResetRA1,
36     Axis:= MlcVarGlobal.RA1,
37     Done=> do_ResetRA1,
38     Error=> er_ResetRA1,
39     ErrorID=> ID_ResetRA1,
40     ErrorIdent=> Ident_ResetRA1);
41

```

Fig. 5-13: MotionProg – Structured text

The part of the declaration containing the variables is filled during the program creation.

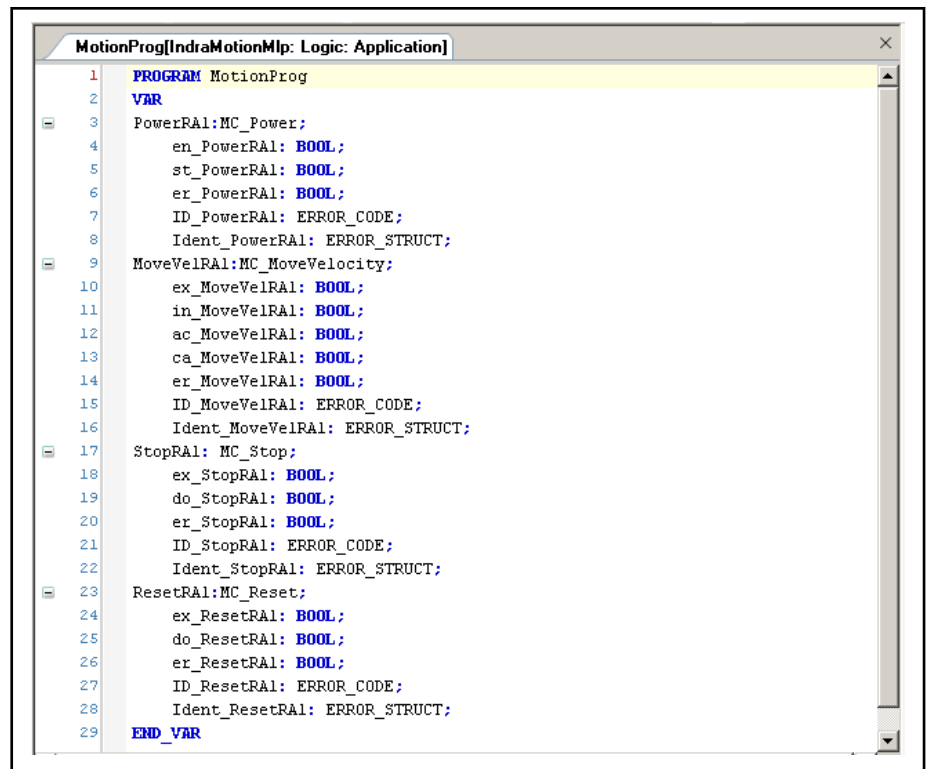


Fig. 5-14: MotionProg – Variables

If the control program was created, it should be compiled:

1. **Create ► Build Application** to ensure that there are no errors in the source code.
2. If the compilation has been executed successfully (see task list), go on-line with the control (**right mouse button on Control ► Switch online** or click on the icon in the toolbar).
3. A subsequent phase switching from P2 to BB is required.

In the context menu of the Sercos node, select **SERCOS State ► P4**

4. At the end, switch the application online (the program created in the IndraLogic uploaded to the PLC, **Debug ► Log in**) and start the program (**Debug ► Start**).

The control program creation is now completed. The variables enabled via the WinStudio interface at a later point in time can be seen in the IndraLogic.

5.4.4 Integrating the WinStudio Visualization in IndraWorks

To access program variables externally (as it is the case with WinStudio, external application), symbols with certain access rights must be defined. This is performed via Symbol Configuration (**Application ► Symbol Configuration**)

If the symbol configuration is inserted under "Application", the available variables are shown on the left after the symbol configuration has been selected and after clicking on refresh (1). The variables can be moved to the **"Selected variables"** area using the arrow keys (2). The variables selected can only be activated in an external application. If the variables were selected from the **MotionProg**, the following figure is displayed:

Application Examples

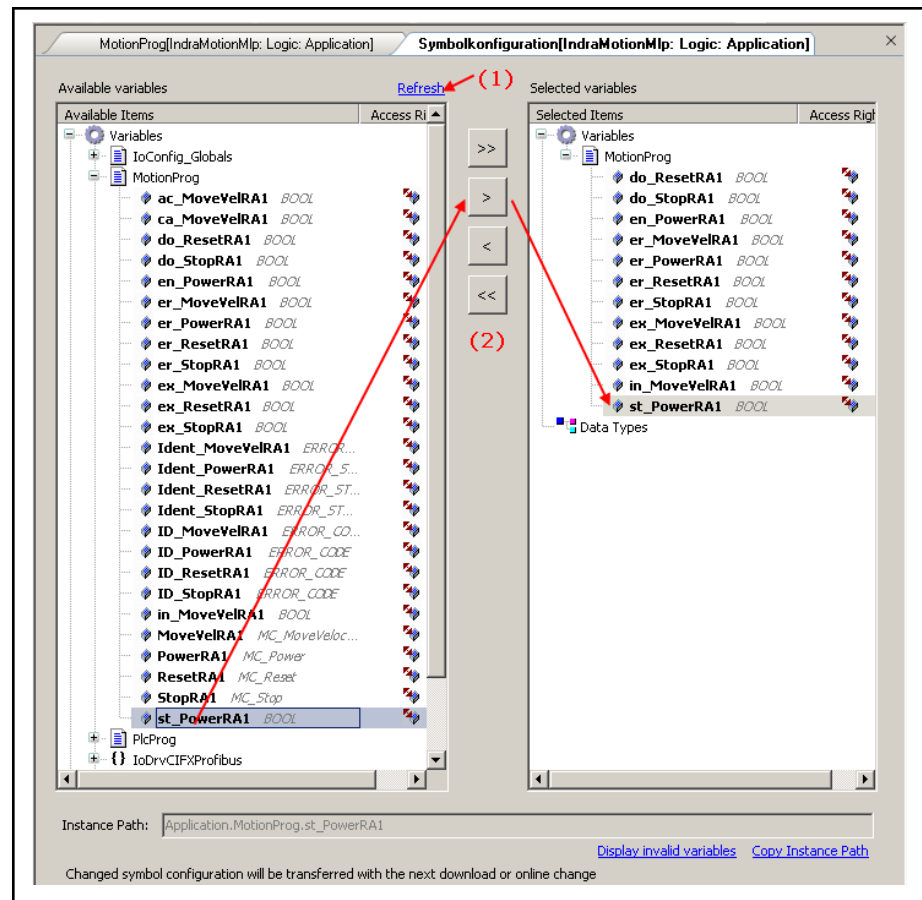


Fig. 5-15: Symbol configuration – Variables selected

Three steps are required to write the symbol configuration correctly into the PLC.

1. The option **Create ► Clean All** has to be selected to delete the old data in the PLC.
2. The application has to be compiled again. The program changed (**Create ► Compile**).
3. Go online with the application to write the changed program into the PLC (**Debug ► Login**).

Now, the selected variables can be read by the PLC by an external application (here: WinStudio).

Drag the "VEH/VEP" visualization from the library and drop it in the project node.

Select the device types in this window, specify a name for the visualization, enter the IP address of the control and select the type of application.

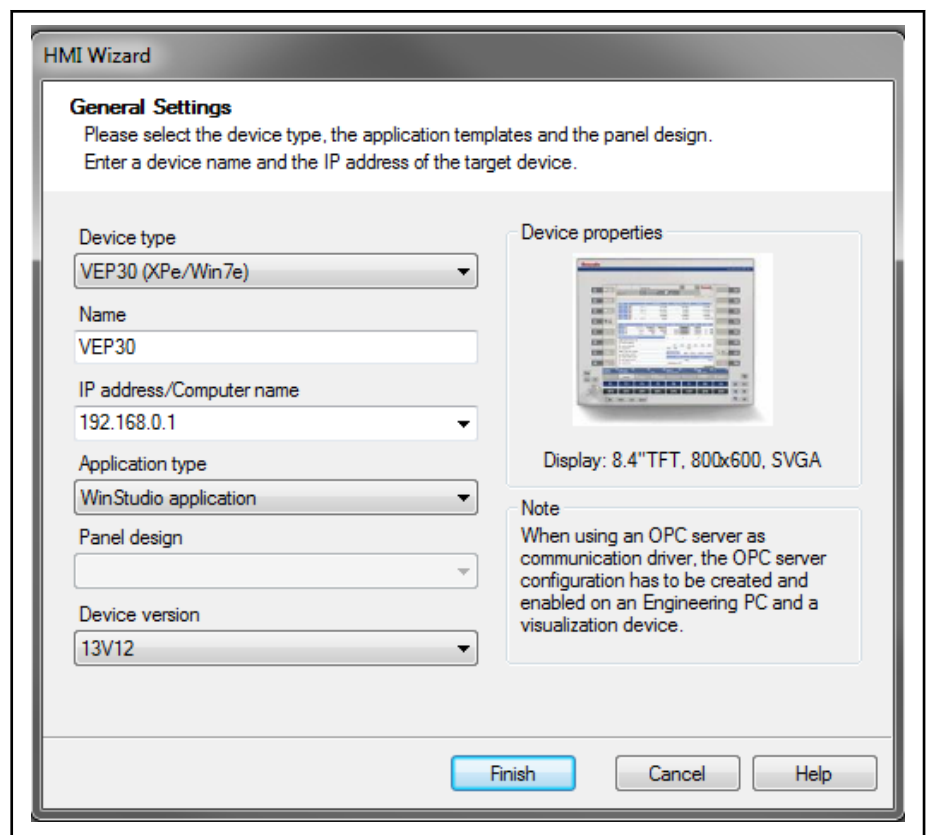


Fig. 5-16: Inserting the visualization VEP – General Settings

- The **device type** is to be selected according to the HMI (in this case **VEP30 (XPe/Win7e)**).
- Enter the name in the **Name** field under which the visualization is executed in the project (in this case **VEP30**).

You can specify a name (IndraWorks will notify you if any characters chosen are invalid) or select the default name.

- Enter the IP of the VEP30 below the **IP address**.
- Select a WinStudio application as **Application type**.

Now, establish the communication between WinStudio and the PLC.

Open WinStudio by clicking on the "WinStudio" object in the project tree under visualization.

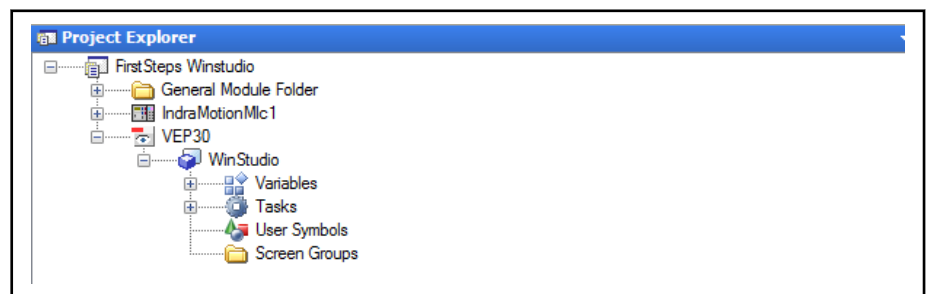


Fig. 5-17: Projekt Explorer with "WinStudio" node

To read the variables from the control, a special driver screen has to be integrated into WinStudio. The BRWS driver is used in this example project. Include the BRWS driver to the project via **VEP30 ► WinStudio ► Variables ► Communication ► Add/Remove Driver**. Right-click to highlight the

Application Examples

BRWS driver in the "Available drivers" field and press "Select". The driver is displayed in the "Selected drivers" field. Confirm with "OK".

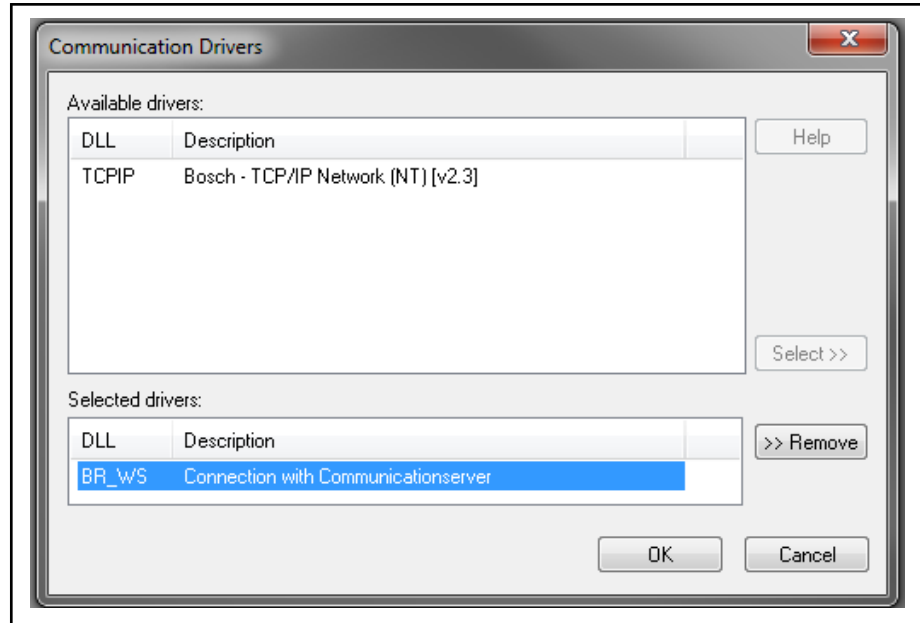


Fig. 5-18: The BRWS driver was integrated via the "Select" button

Create the driver sheet in the workspace under **VEP30 ► WinStudio ► Variables ► Communication ► BRWS ► New BR_WS Driver Sheet**.

Click on the BRWS driver with the right mouse button, the context menu "New BR_WS driver sheet" is displayed.

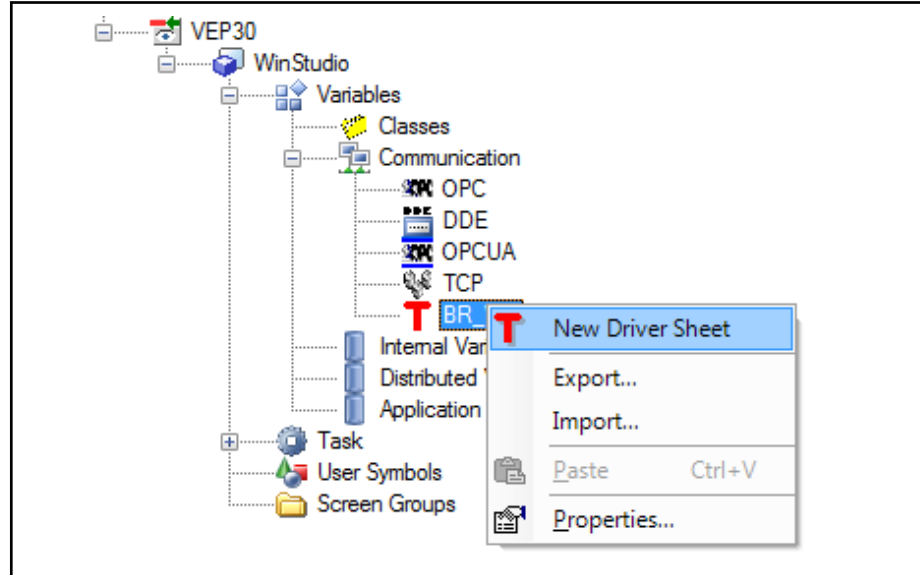


Fig. 5-19: Creating a new BR_WS driver sheet

The driver must be specifically configured.

Select the "BRWS" folder with the right mouse button. A context menu is displayed in which you have to select **Properties** again.

Enter the new IP address of the control into the window

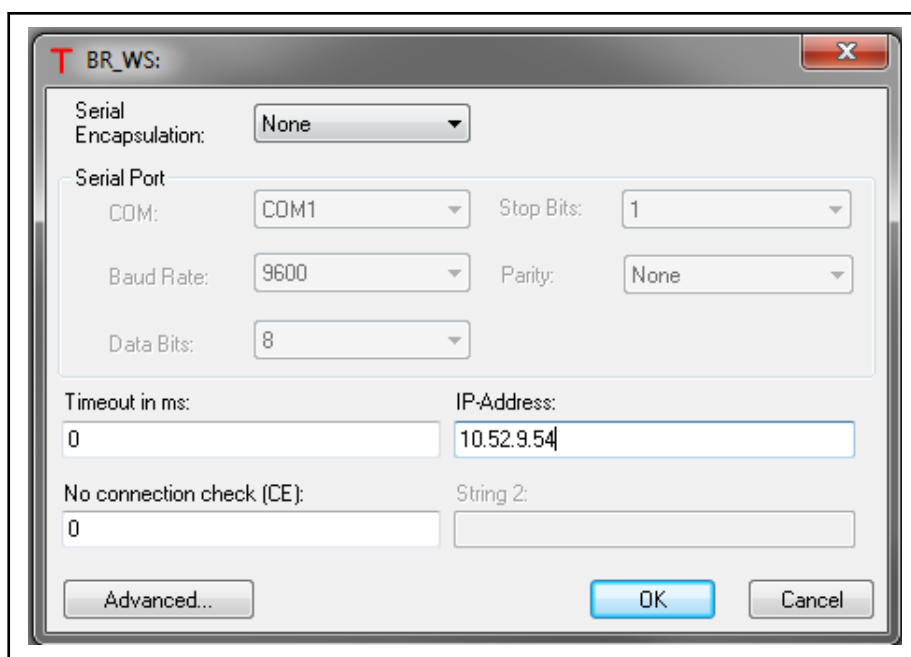


Fig. 5-20: BRWS properties, here IP address

In the following step, all PLC variables of the WinStudio environment are shown.

Therefore, click on the driver sheet "1". A new window opens on the right, next to the project tree.

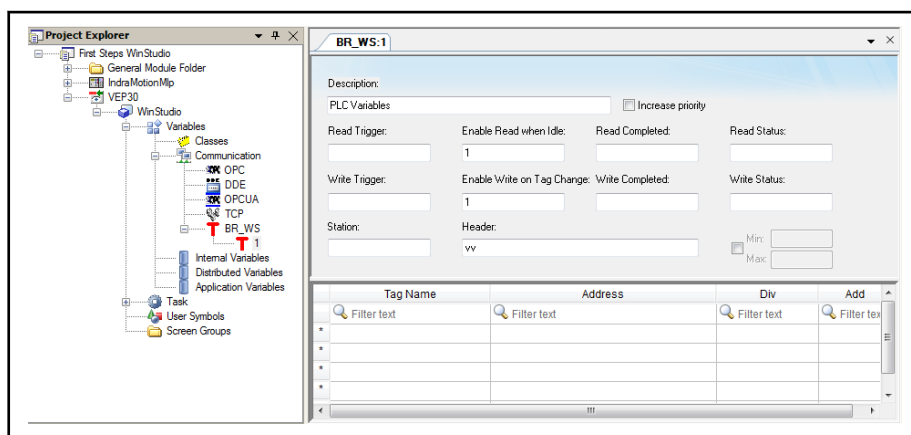


Fig. 5-21: BRWS driver sheet configuration, Sheet "1"

Fill out the fields as shown above.

The fields **Enable read when idle** and **Enable write on tag change** have to assume the value 1 in order to work with the variables.

Enter a unique entry for each driver sheet in the **Description** field ("vv" is exemplarily used, entry can be freely selected). Save the driver sheet (save project with Ctrl+S for example). The driver sheet configuration is now known to the system.

Read the PLC variables enabled in the symbol configuration in WinStudio. There are two options:

1. Using the IndraWorks process variable browser:

Application Examples

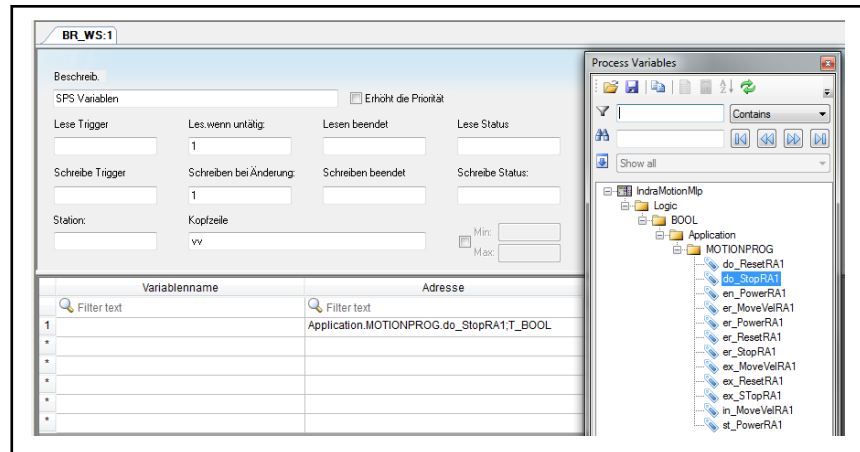


Fig. 5-22: BRWS driver sheet and IndraWorks process variable browser

Open the tools window for the process variables via **View ► More Windows ► Process Variables**. Refresh the process variable browser with



to see the PLC variables available in the symbol configuration of the control. Drag&drop them into the driver sheet columns.

The variable name with access path is specified as follows: ";" and the variable type "T_BOOL".

2. using the BRWS Variable browser

Set the mouse cursor to the table section of the driver sheet (e.g. to the "Address" column) and right click. A new window, the variable browser, opens.

Select "Read Variables (online)".

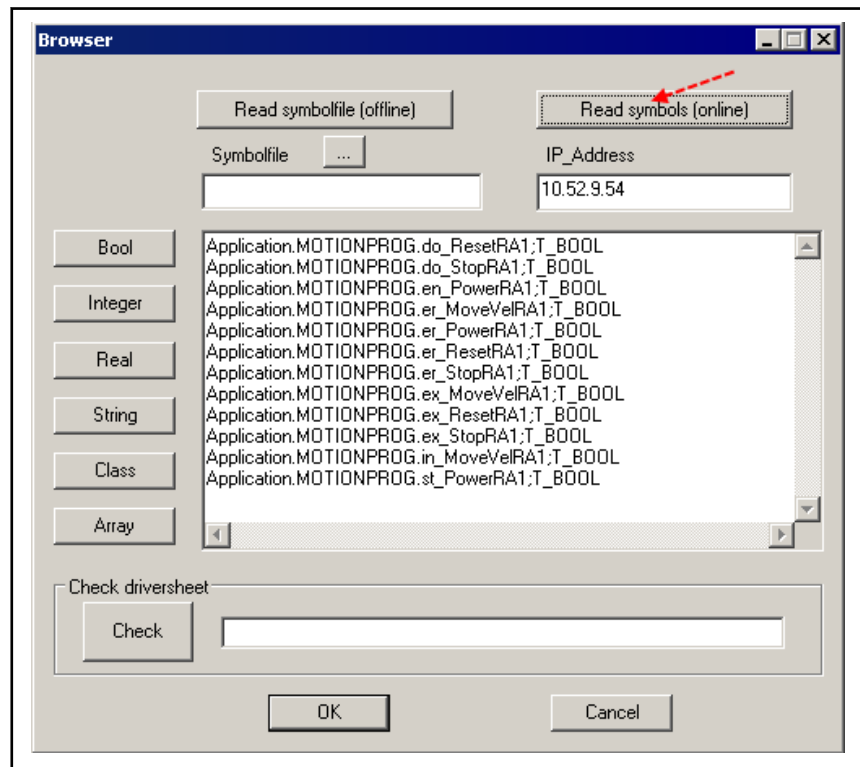


Fig. 5-23: Browser, Read variables online

Application Examples

All PLC variables are shown in the browser that have earlier been released under the **symbol configuration** and that can be activated via WinStudio.

Highlight and copy the variables. Close the window by clicking on "OK"

Copy this variable to the "Address" (1) column, top cell.

WinStudio creates its own namespace. This is required as the visualization should be created independently and possibly before the PLC program.

Both name lists are assigned variable per variable (2).

The **Variables seen from the WinStudio application** are contained in the **Application variables** (3).

The WinStudio list can be filled in directly (Application variable list) or as shown in the following.

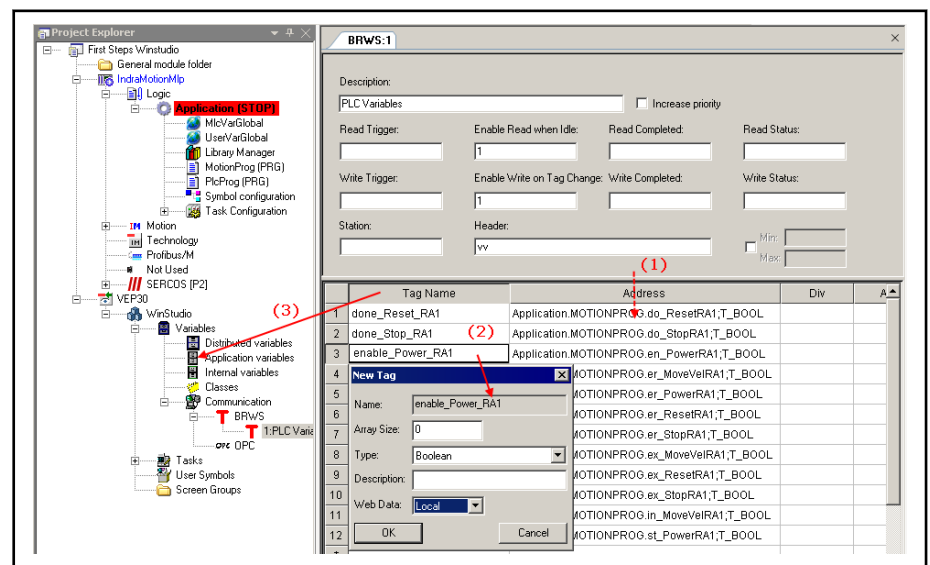


Fig. 5-24: BRWS properties, here screen 1, with variable names and addresses

BRWS:1 Application variables					
	Name	Size	Type	Description	Scope
1	done_Reset_RA1	0	Boolean		Local
2	done_Stop_RA1	0	Boolean		Local
3	enable_Power_R...	0	Boolean		Local
*			Integer		Server
*			Integer		Server
*			Integer		Server
*			Integer		Server
*			Integer		Server

Fig. 5-25: WinStudio: Application variables, preliminary result

Application Examples

	Tag Name	Address	Div	Add
1	done_Reset_RA1	Application.MOTIONPROG.do_ResetRA1;T_BOOL		
2	done_Stop_RA1	Application.MOTIONPROG.do_StopRA1;T_BOOL		
3	enable_Power_RA1	Application.MOTIONPROG.en_PowerRA1;T_BOOL		
4	error_MoveVelocity_RA1	Application.MOTIONPROG.er_MoveVelRA1;T_BOOL		
5	error_Power_RA1	Application.MOTIONPROG.er_PowerRA1;T_BOOL		
6	error_Reset_RA1	Application.MOTIONPROG.er_ResetRA1;T_BOOL		
7	error_Stop_RA1	Application.MOTIONPROG.er_StopRA1;T_BOOL		
8	execute_MoveVelocity_RA1	Application.MOTIONPROG.ex_MoveVelRA1;T_BOOL		
9	execute_Reset_RA1	Application.MOTIONPROG.ex_ResetRA1;T_BOOL		
10	execute_Stop_RA1	Application.MOTIONPROG.ex_StopRA1;T_BOOL		
11	in_Velocity_RA1	Application.MOTIONPROG.in_MoveVelRA1;T_BOOL		
12	status_Power_RA1	Application.MOTIONPROG.st_PowerRA1;T_BOOL		
*				

Fig. 5-26: BRWS properties, here screen 1, with variable names and addresses

If the PLC variables have been integrated correctly, all variables are listed under

WinStudio ► Variables ► Application Variables

BRWS:1 Application variables					
	Name	Size	Type	Description	Scope
1	done_Reset_RA1	0	Boolean		Local
2	done_Stop_RA1	0	Boolean		Local
3	enable_Power_R...	0	Boolean		Local
4	error_MoveVelocit...	0	Boolean		Local
5	error_Power_RA1	0	Boolean		Local
6	error_Reset_RA1	0	Boolean		Local
7	error_Stop_RA1	0	Boolean		Local
8	execute_MoveVel...	0	Boolean		Local
9	execute_Reset_R...	0	Boolean		Local
10	execute_Stop_RA1	0	Boolean		Local
11	in_Velocity_RA1	0	Boolean		Local
12	status_Power_RA1	0	Boolean		Local
*			Integer		Server

Fig. 5-27: Checking the transfer: WinStudio – Application variables

5.4.5 Creating the WinStudio Visualization

After the variables have been integrated and are now available as user variables, you can start with the graphic interface.

Close the current window. Select the "WinStudio" folder with the right mouse button. Insert a new screen in the next step.

Application Examples

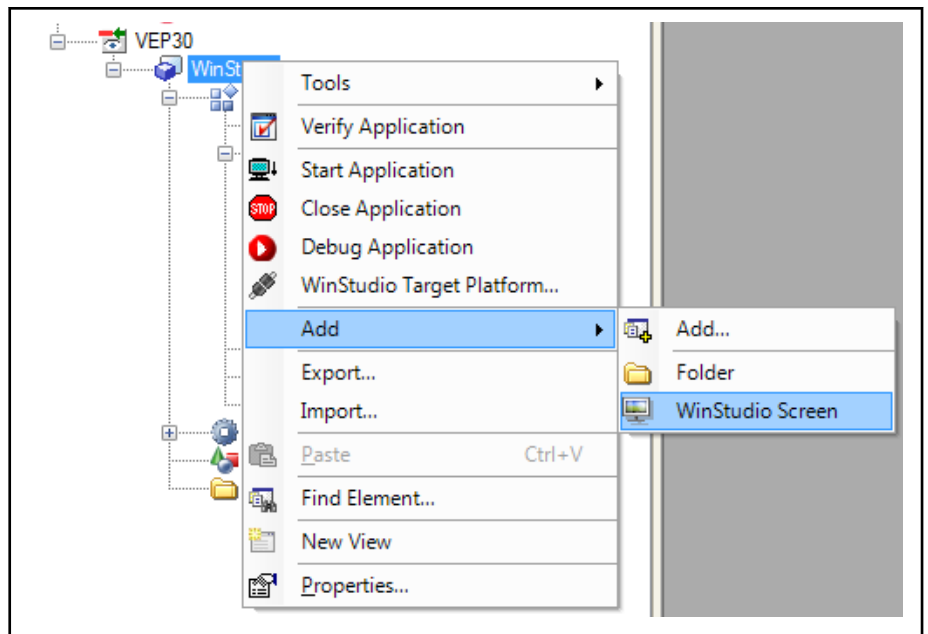


Fig. 5-28: WinStudio – Insert screen

You can freely select the name of the screen: here "start screen".

Open the screen. The "Screen Attributes" can be selected after clicking on the screen with the right mouse button.

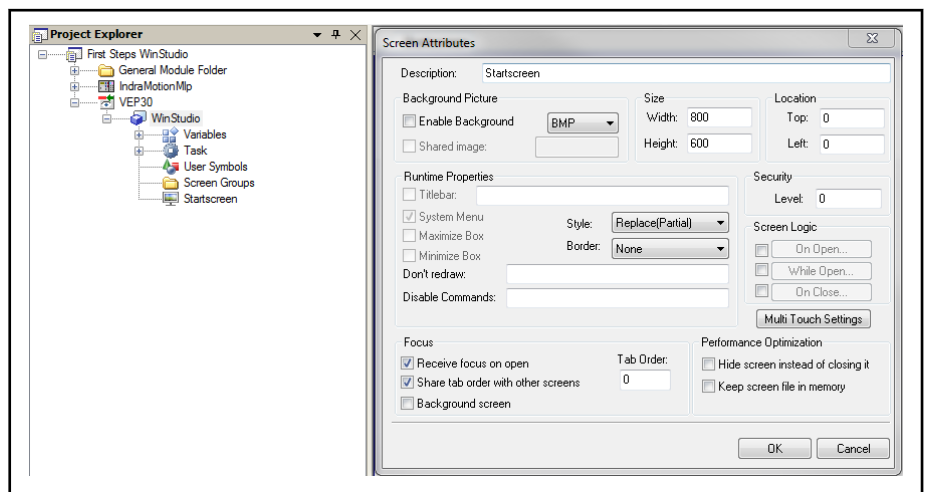


Fig. 5-29: WinStudio – Screen/Settings

The workspace next to the tree opens as well. Objects can be inserted into the screen using the individual toolbars at the right WinStudio margin.

Application Examples

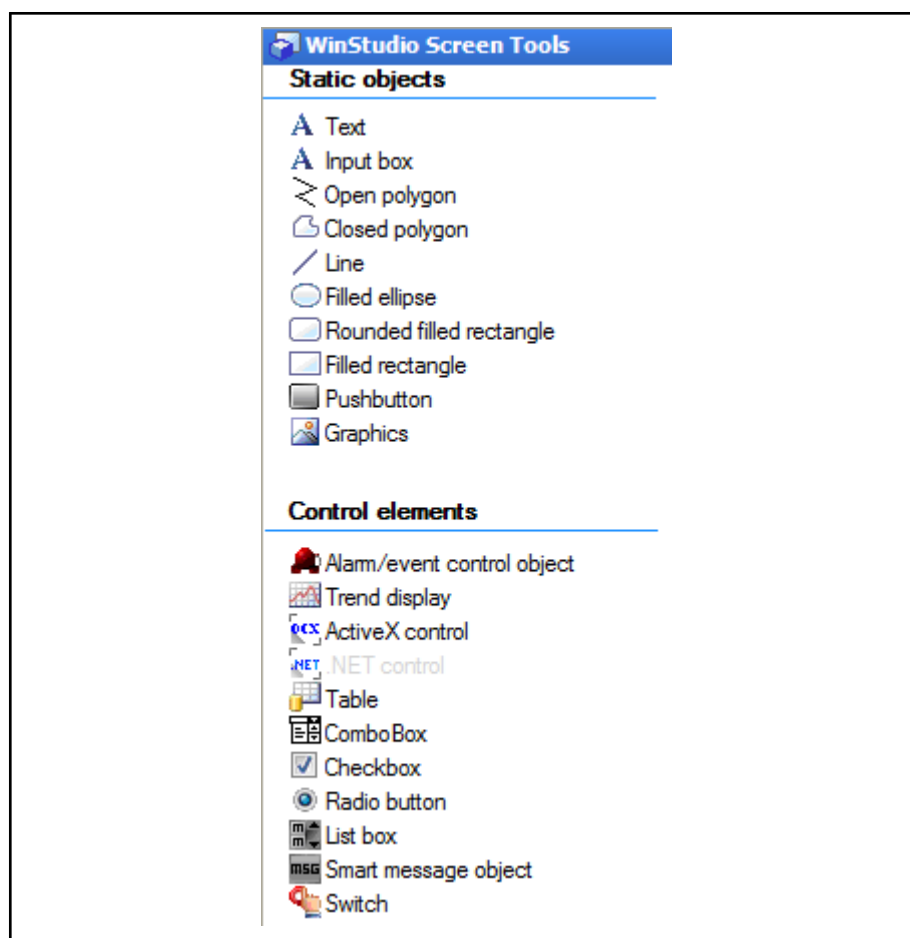


Fig. 5-30: WinStudio – Screen Tools

The objects are prepared for a small demonstration interface. Therefore, change the background color of the later interface with the right mouse button.

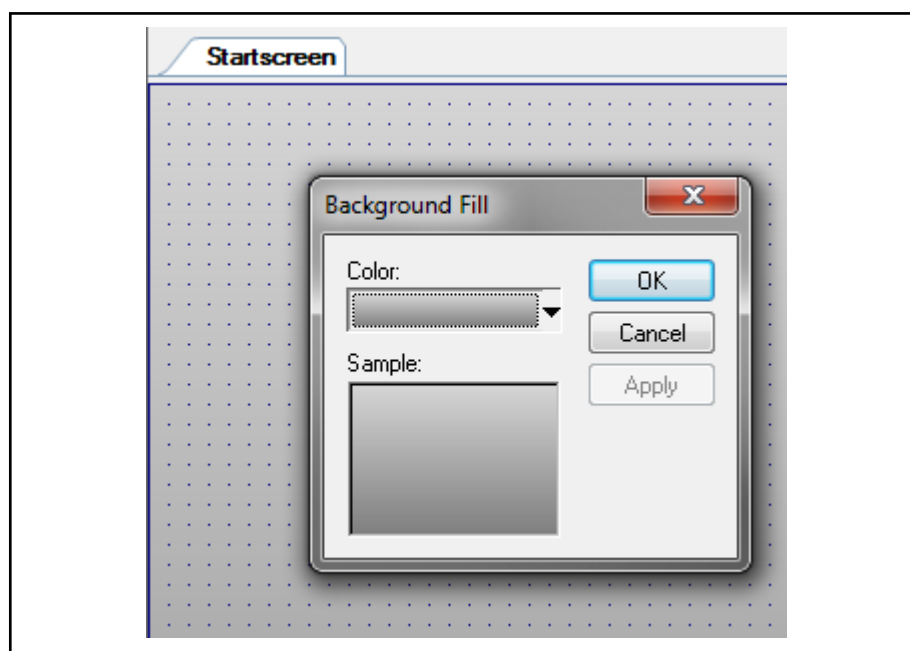


Fig. 5-31: "Startup" screen - Change background color

Application Examples

To optically divide the interface, insert the "Filled rectangle" object into the screen. This rectangle should cover the upper quarter of the work space.

The object can be adapted more specifically if you open the context menu with the right mouse button. Select "Properties".

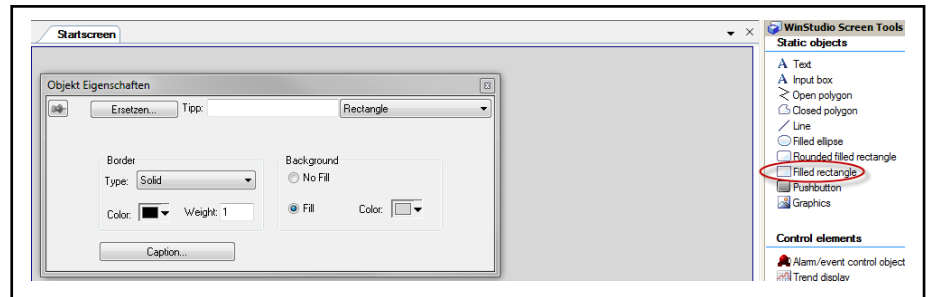


Fig. 5-32: WinStudio – Insert the "Filled rectangle" object

As shown, the filling color was adapted and the rectangle was cut to the complete screen length.

WinStudio provides the user with a library containing symbols for example.

The library can be accessed via the main menu "WinStudio" or via the context menu of the WinStudio folder.

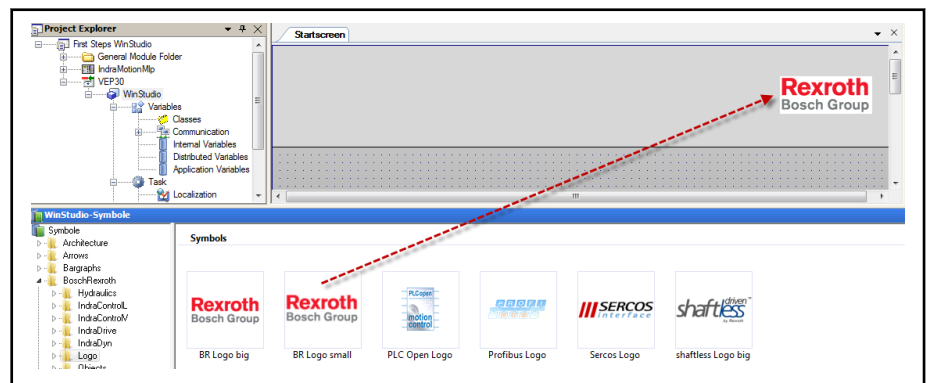


Fig. 5-33: WinStudio – Library opened, logo applied

The requested symbol, here "BR Logo small", is inserted via drag&drop.

Insert the "Text" object from the WinStudio ScreenTools into the image. Formatting can be selected via the object settings (right mouse click ► Settings).

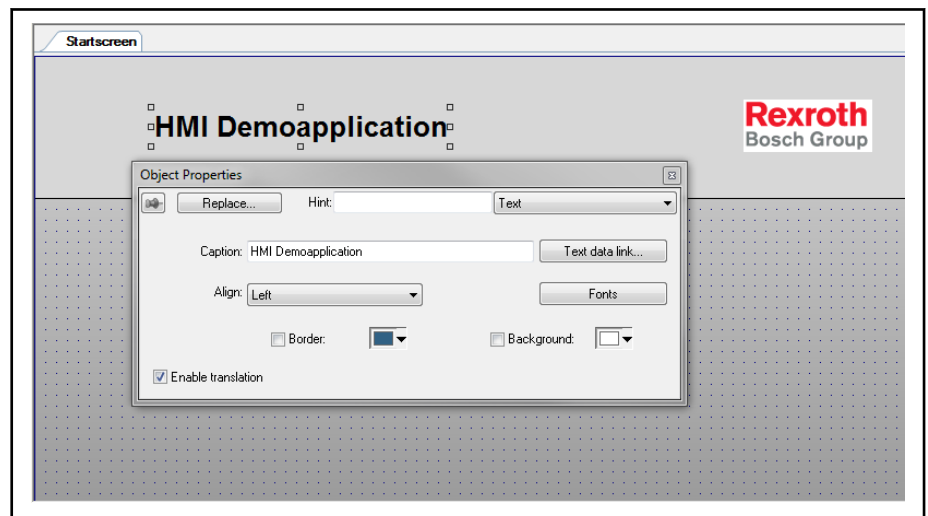


Fig. 5-34: WinStudio – Insert text object

Application Examples

The control elements are now inserted into the screen. A switch (control element "Switch") from the WinStudio ScreenTools is used as example.

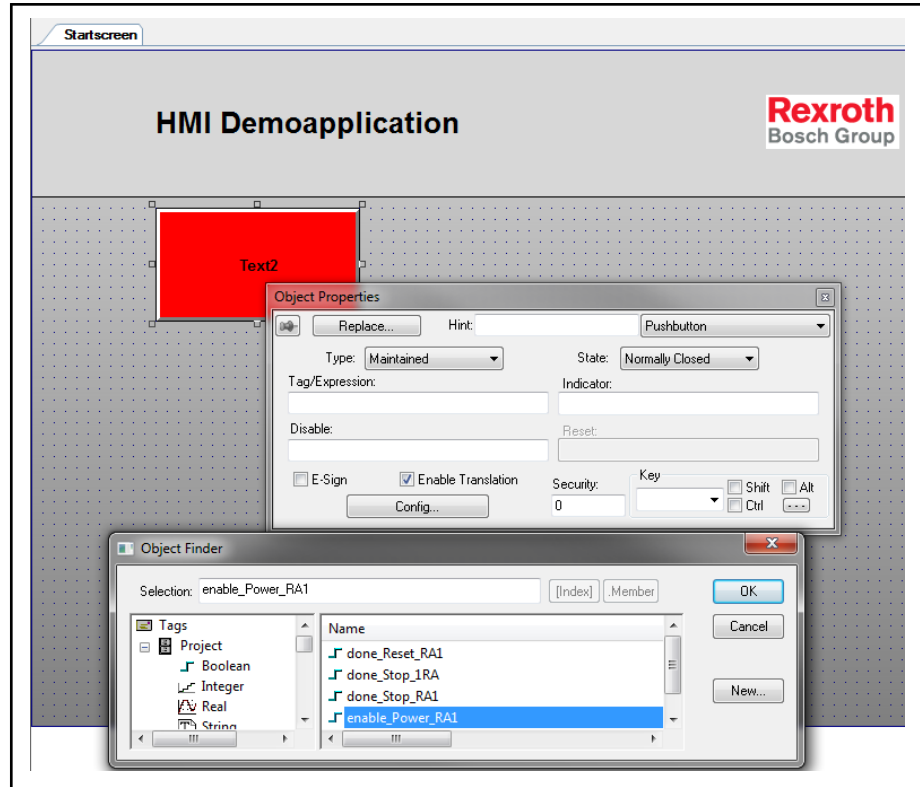


Fig. 5-35: WinStudio – "Switch" inserted

Different settings have to be made for the control elements.

As shown in the figure above, the different fields must be set.

Configuring the switch as **type "Switch"** in the object properties.

Insert the variable to be triggered by the switch object into the field "Var/Expression" via the button in the **Tag/Exp**

(in this case **enable_Power_RA1**).

The **"Variable Selection"** opens to select the variable.

Set the behavior in case of an open and closed switch under **Config..**



The settings "Color selection", "Status" (normally closed/ normally open) and the "Labeling" are to be assigned user-specifically.

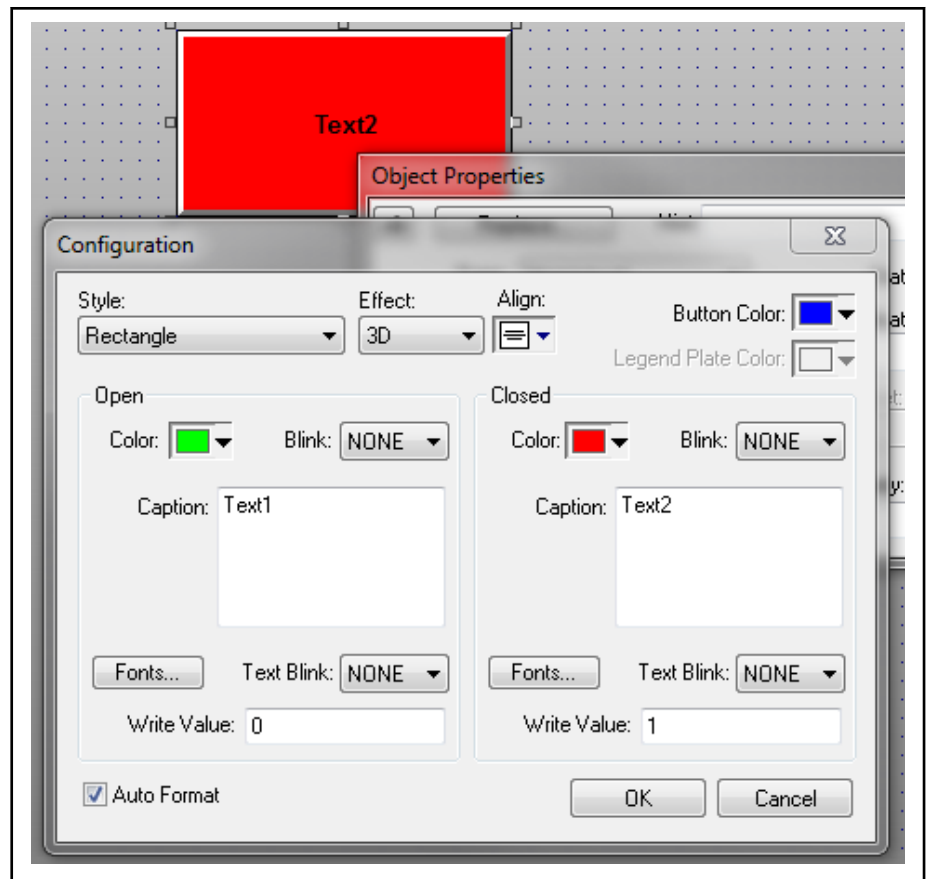


Fig. 5-36: Configuration dialog of the "Switch" object

The four switches are implemented with the following configurations:

Connect power

- Type: Switch
- Status: Normally closed
- Variable: enable_Power_RA1
- Label (open): Power ON / green
- Label (closed): Power OFF / grey

Reset drive

- Type: Switch
- Status: Normally closed
- Variable: execute_Reset_RA1
- Label (open): Reset drive / green
- Label (closed): Reset OFF / grey

Switch on drive

- Type: Switch
- Status: Normally closed
- Variable: execute_MoveVelocity_RA1
- Label (open): Drive RUN / green
- Label (closed): Drive ON / grey

Stop drive

- Type: Switch

Application Examples

- Status: Normally closed
- Variable: execute_Stop_RA1
- Label (open): Drive OFF / green
- Label (closed): Drive STOP / grey

To display the current status of the drive, add checkboxes to the interface.

Make the settings **"Description"** and **"Variable"**.

See the configuration of all checkboxes below:

Power status

- Description: Power ON
- Variable: status_Power_RA1

Power status (error)

- Description: Power ERROR
- Variable: error_Power_RA1

Drive status

- Description: Drive ON
- Variable: in_Velocity_RA1

Drive status (error)

- Description: Drive ERROR
- Variable: error_MoveVelocity_RA1

Status reset drive

- Description: Reset ON
- Variable: done_Reset_RA1

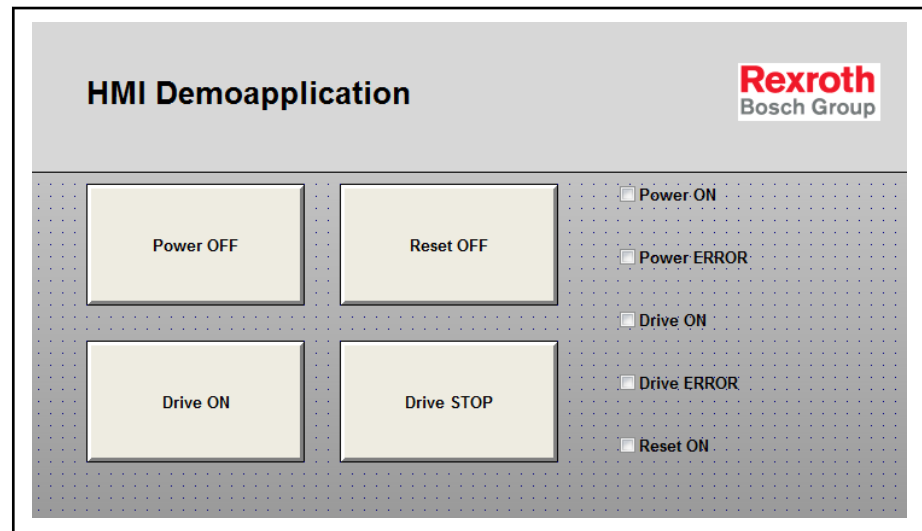


Fig. 5-37: WinStudio – Complete interface

5.4.6 Starting the Interface

Just a few steps are left and the interface is running on the control.

The startup screen is determined vis the project settings.

Select the **"Viewer"** tab and enter the created startup screen in the **"Startup Screen"** Window below the **WinStudio ► Properties** path (Otherwise, the interface is not loaded in the viewer. Also select the property via the context menu of the **"Image"** node in the Project Explorer).

Application Examples

More display options can be selected in this tab (e.g. displaying WinStudio menu bar).

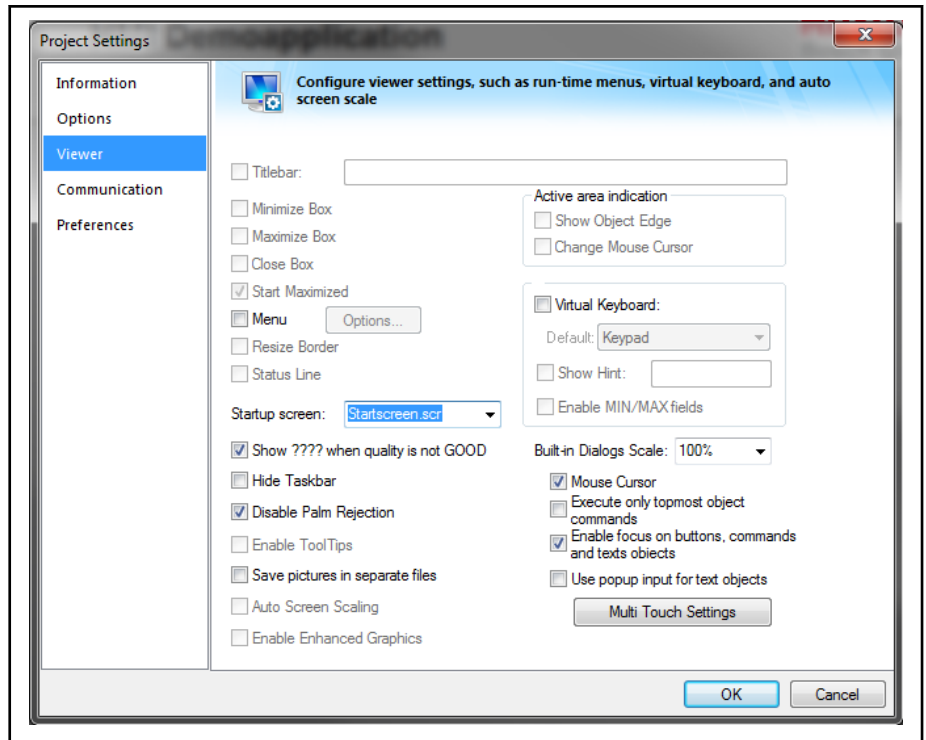


Fig. 5-38: Winstudio project settings; "Viewer" tab

The settings made can now to be saved in the project.

Transfer your PLC project in a ready for operation state:

1. Go online with the control.
2. Log in.
3. Start your application.
4. Switch the sercos bus to phase 4 (P4).

Transfer the WinStudio project to the target platform.

Select the context menu item "WinStudio Target Platform" on the "WinStudio" node

Connect to the HMI **(1)**. For device running with Windows CE6.0 or Windows 7 embedded compact, the runtime system has to be transferred to the HMI device **(2)**. For devices with Windows 7 embedded or Windows XP embedded, step **(2)** is obsolete.

Application Examples

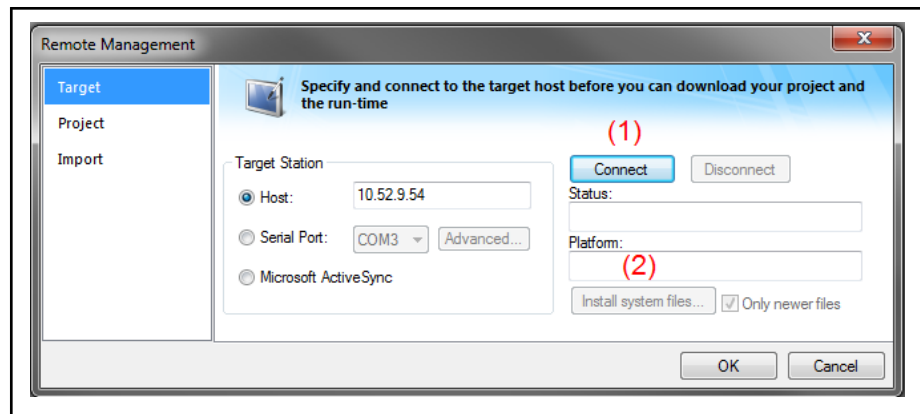


Fig. 5-39: Execution platform: Applying project data

Specify the target path for the project data on the target device (1). The following default paths are used for the different target devices:

- **VEPxx.4/5, Windows Xp embedded/ Windows 7 embedded:**
D:\Documents\Rexroth\My Documents\WinStudio for IndraWorks Projects\config
- **VR21xx, Windows 7 embedded compact:**
\\FlashDrv\Rexroth\WinStudio\config\
- **VEPxx.4, Windows CE6.0:**
\\StorageCard\Rexroth\Visu\WinStudio\config\

If connected to the respective device, the path is automatically provided for selection. Ensure that the target path specification (1) is correctly entered and korrekt it if required via the ... button.

Ensure that the **Keep user files** option is enabled. This avoids files previously located in the specified target path to be deleted when transferring project files.

Download project files via the **Download(2)** button (loading is only visible in the "Remote Agent" window on the target device).

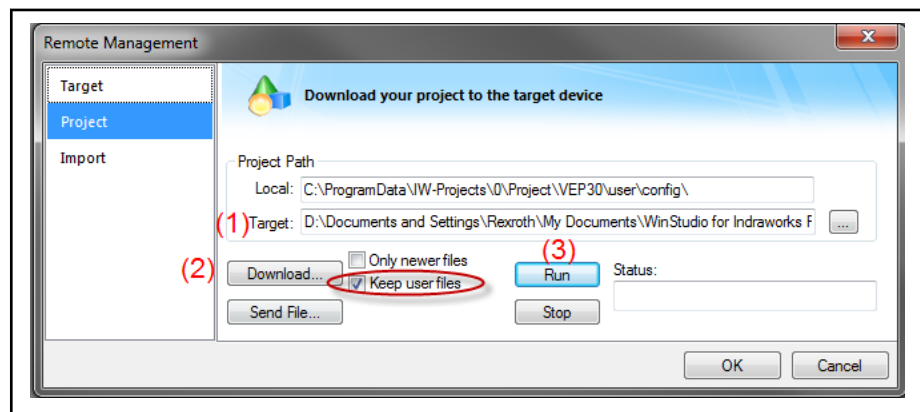


Fig. 5-40: Loading and starting the application

Start the application (3).

Start interface automatically when booting the control

Proceed as follows to start the HMI application when booting Windows XP or Windows 7 embedded:

Go to **Start ► All Programs ► Rexroth ► IndraWorks ► WinStudio**.

Open the context menu of the "Startup" service and create a shortcut. Add the shortcut to the Windows Startup folder.

5.5 IndraLogic Visualization

An application sample for the VI-Logic visualization is contained in "Rexroth IndraLogic XLC 13VRS First Steps", DOK-XLC***-F*STEP**V13-CO01-EN-P.

5.6 HMIs of Third Party Manufacturers

5.6.1 OPC.DA Server

Tools to Configure the OPC.DA Server on a PC

The program listings "Batch processing program for opening and editing the file OPCServer.ini" on page 159 and "Batch processing program for opening the file OPCServer.log" on page 159 can be used for a more user-friendly handling of the OPC.DA server configuration file "OPCServer.ini" as well as the "OPCServer.log" log file on the PC. The file paths can be determined using the registry and the environment variables. Open them in the "Notepad" editor ("SET EDITOR_FILE=notepad.exe").

Batch processing program for opening and editing the file "OPCServer.ini"

```
@ECHO OFF
REM
REM Open IndraLogic OPC Server Configuration File.
REM If it does not exist, copy it!
REM

SET EDITOR_FILE=notepad.exe
SET REGKEY=HKLM\Software\Rexroth\IndraWorks
SET REGKEYNAME=CurrentIndraLogicOPCDAIniFileName

REM 'delims' is set with TAB and SPACE
FOR /F "tokens=2* delims=" "%a" IN ('REG QUERY "%REGKEY%" /v %REGKEYNAME%') DO SET OPCFILE="%~b"

IF EXIST "%OPCFILE:~1=" GOTO fileopen

COPY "%CommonProgramFiles%\Rexroth\OPCServer.ini.template" "%OPCFILE:~1="

:fileopen
START %EDITOR_FILE% "%OPCFILE:~1="
```

Fig. 5-41: Batch processing program for opening and editing the file "OPCServer.ini"

Batch processing program for opening the file "OPCServer.log"

```
@ECHO OFF
REM
REM Open IndraLogic OPC Server Log File.
REM

SET EDITOR_FILE=notepad.exe
SET REGKEY=HKLM\Software\Rexroth\IndraWorks
SET REGKEYNAME=CurrentIndraLogicOPCDALogFileName

REM 'delims' is set with TAB and SPACE
FOR /F "tokens=2* delims=" "%a" IN ('REG QUERY "%REGKEY%" /v %REGKEYNAME%') DO SET OPCFILE="%~b"

IF EXIST "%OPCFILE:~1=" GOTO fileopen

GOTO :EOF

:fileopen
START %EDITOR_FILE% "%OPCFILE:~1="
```

Fig. 5-42: Batch processing program for opening the file "OPCServer.log"

Application Examples

Tools to Transfer the OPC.DA Server Configuration to a Visualization Device VEPxx.4 with WindowsCE

If the visualization device ("VEPxx.4") is in the same network as the development PC the script "[Batch processing program for copying the file OPCServer.ini to a VEPxx.4](#)" on page 160 can be used to transfer the OPC.DA server configuration file "OPCServer.ini" which is active on the PC.

Adjust the IP address in the following assignment line:

"SET IPADDR_VEP=192.168.1.11"

Batch processing program for copying the file "OPCServer.ini" to a "VEPxx.4"

```
@ECHO OFF
CLS
REM
REM Copy IndraLogic OPC Server Configuration File to VEPxx.4
REM
REM
REM Set IP Address of the VEP Device
SET IPADDR_VEP=192.168.1.11

SET REGKEY=HKLM\Software\Rexroth\IndraWorks
SET REGKEYNAME=CurrentIndraLogicOPCAIniFileName

REM 'delims' is set with TAB and SPACE
FOR /F "tokens=2* delims=" " %a IN ('REG QUERY "%REGKEY%" /v %REGKEYNAME%') DO SET OPCFILE="%%b"

IF EXIST "%OPCFILE:=" GOTO filecopy

ECHO.
ECHO !!! ERROR !!!
ECHO - File
ECHO %OPCFILE%
ECHO does not exist!
ECHO.

PAUSE
GOTO :EOF

:filecopy

REM Create FTP command file
SET FTPCMDFILE="%-dp0FtpCmd.txt"
SET VEPDSTDIR=/StorageCard/Rexroth/Controls/OPCServer

ECHO open %IPADDR_VEP%> %FTPCMDFILE%
ECHO anonymous>> %FTPCMDFILE%
ECHO 123@456.com>> %FTPCMDFILE%
ECHO cd %VEPDSTDIR%>> %FTPCMDFILE%
ECHO ascii>> %FTPCMDFILE%
ECHO put "%OPCFILE:=">> %FTPCMDFILE%
ECHO close>> %FTPCMDFILE%
ECHO bye>> %FTPCMDFILE%

ECHO.
ECHO Copy file
ECHO %OPCFILE%
ECHO.
ECHO.
CALL ftp -s:%FTPCMDFILE%

PAUSE
```

Fig. 5-43: *Batch processing program for copying the file "OPCServer.ini" to a "VEPxx.4"*

5.6.2 ModbusTCP

General

It is useful for the configuration to execute first tests with the Modbus test clients.

Test client examples available in the internet for free (date 09.2008):

- IPC@CHIP Modbus Client V1.4
- Modbus TCP/IP Request Tool Pro

The following chapters describe the configuration of a TCP connection with
Details about the configuration of a Modbus TCP connection is explained in the following chapters:

- [Configuration with Siemens WinCC-flexible 2007, page 162](#) and
- [Configuration with Pro-face GP-Pro EX 2.2, page 166](#)

Addressing data from a PLC project

In the PLC project, data ranges for the register and bit data are transferred to function block instance IL_ModbusTCPServer using the pointer.



The data ranges of the Modbus TCP server are accessed via the following addressing by the Modbus client

Register data range:

Addressing the "Holding Register" data using **%MW** (read and write access) or the "Input Register" data using **%IW** (only read access).

Bit data range:

Addressing the "Coil" data using **%M** (read and write access) or the "Discrete Input" data using **%I** (only read access).

Bit data

In the test project, the bit data range was assigned as array of bytes from where every single bit can be addressed continuously.



No ARRAY OF BOOL can be used for the bit data range since 8 bits of memory are assigned to a variable of BOOL in the IndraLogic.

The individual bits of a byte can be copied on variables of type BOOL (and vice versa)

The address of a variable from a bit data range corresponds to the bit number from the byte array used for bit data.

Example:

Bit data (%M)

%M0 address corresponds to the **BitData[0].0** variable

%M7 address corresponds to the **BitData[0].7** variable

%M8 address corresponds to the **BitData[1].0** variable

%M12 address corresponds to the **BitData[1].4** variable

Register data

The register data range can consist of up to 65536 words and can be organized by the user without any restrictions.

For a better overview on the register data, a structure containing arrays of different PLC data types was used for the register data range in the test project. This structure was created in such a way that the individual data type areas are always provided with a length of 1000 words and do therefore begin with a defined word address.

Example of a data structure for RegisterData

Program:

```
TYPE REGISTER_DATA :  
STRUCT
```

Application Examples

```

arWord:   ARRAY[0..999]OF WORD;      (* %MW0000 - %MW0999 *)
arInt:    ARRAY[0..999]OF INT;       (* %MW1000 - %MW1999 *)
arUInt:   ARRAY[0..999]OF UINT;      (* %MW2000 - %MW2999 *)
arDint:   ARRAY[0..499]OF DINT;      (* %MW3000 - %MW3999 *)
arUdint:  ARRAY[0..499]OF UDINT;     (* %MW4000 - %MW4999 *)
arReal:   ARRAY[0..499]OF REAL;      (* %MW5000 - %MW5999 *)
arString: ARRAY[0..99]OF STRING(19); (* %MW6000 - %MW6999 *)
(* String(19): because 19 byte string + 1 byte end sign *)
(* together 7000 WORDS *)
END_STRUCT
END_TYPE

```

Text string Texts (strings) can also be transferred via the register data area.



In the IndraLogic, the length of a string (in byte) is always the given length + 1 byte of end ID

To define the strings with their total word length, the length is always to be specified with an odd number of bits.

(e.g. string(19) => 19 byte string + 1 byte End ID = 20 byte = 10 words)

The address of a variable from the register data range results from the design of the structure used for the register data.

If the previously shown ["Example of a data structure for RegisterData" on page 161](#) is used, the following addresses result:

Example:

Register data (%MW)

%MW0 address corresponds to the **RegisterData.arWord[0]** variable

%MW1000 address corresponds to the **RegisterData.arInt[0]** variable

%MW1001 address corresponds to the **RegisterData.arInt[1]** variable

%MW5000 address corresponds to the **RegisterData.arReal[0]** variable

%MW5002 address corresponds to the **RegisterData.arReal[1]** variable

%MW6000 address corresponds to the **RegisterData.arString[0]** variable

%MW6010 address corresponds to the **RegisterData.arString[1]** variable

Configuration with Siemens WinCC-flexible 2007

Communication settings To configure the connection in WinCC-flexible, the following settings are required:

1. Open **Communication** ► **Connections** in the project tree menu of the operating device
2. Enter the name of the connection (and optionally a comment) and select the communication driver **"Modicon MODBUS TCP/IP"**
3. Subsequently, the communication parameters have to be configured in the dialog

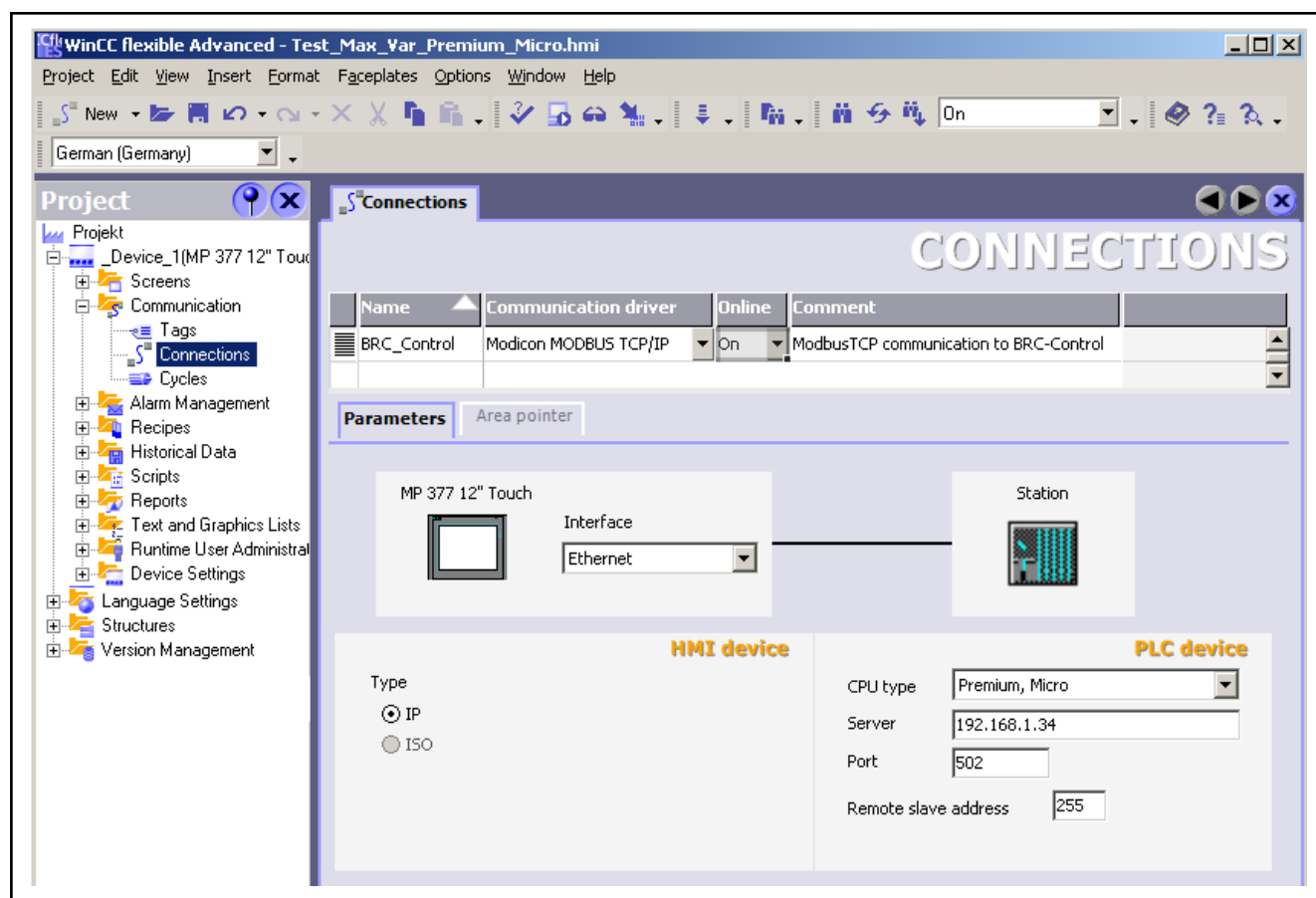


Fig. 5-44: WinCC-flexible Modbus TCP communication parameters

Section	Name	Value	Description
HMI device	Interface	Ethernet	Selection of the communication interface
	Type	IP	Only IP is supported
PLC device	CPU type	Premium, Micro	Schneider Modicon control type operating without address offset and regular bit counting (refer to the Siemens user manual "WinCC-flexible 2007 Communication Volume 2 - Chapter 6")
	Server	"192.168.1.34"	IP address of the control
	Port	"502"	Port of the communication connection (port of the IL_ModbusTCP-Server function block instance)
	Remote slave address	"255"	Maintain default setting (255) ! not relevant for the Modbus TCP - refer to "MODBUS Messaging on TCP/IP Implementation Guide V1.0b" on www.Modbus-IDA.org

Tab. 5-4: WinCC-flexible Modbus TCP communication parameters

Variable Declaration

After configuring the connection, the variables can be declared. Open the menu **Communication** ► **Tags** below the project tree of the operating device.
Example of a variable declaration in WinCC-flexible 2007:

Application Examples

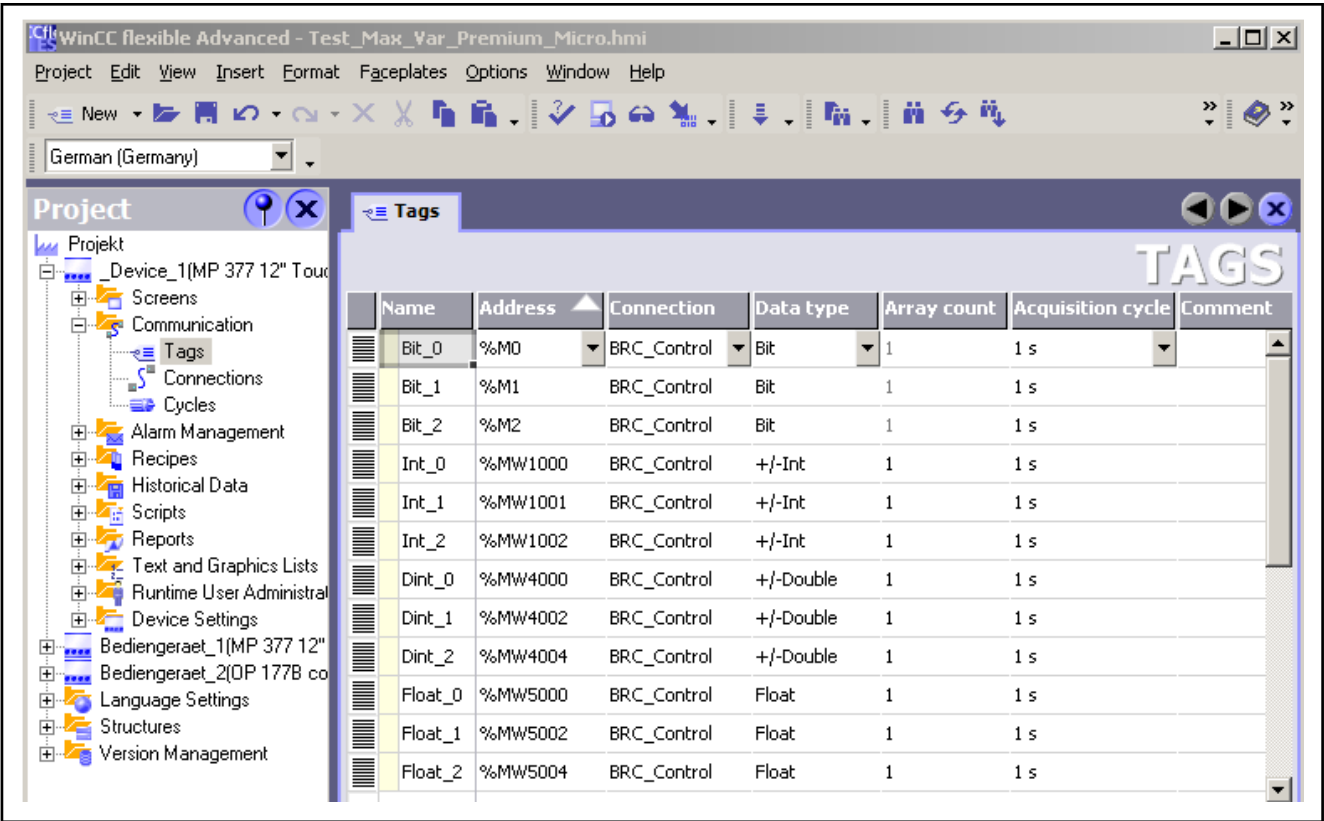


Fig. 5-45: WinCC-flexible variable declaration - Example

WinCC - flexible variable declaration

Input field	Example	Description
Name	"Bit_0"	Variable name
Address	"%M0"	Modbus address of the variable ("Addressing data from a PLC project" on page 161)
Connection	"BRC_Control"	Connection name
Data type	"Bit"	WinCC-flexible variable data type (tab. 5-6 "WinCC-flexible data types/IndraLogic data types" on page 165)
Array count	"1"	Number of array elements if variable is to be assigned as array
Acquisition cycle	"1s"	Update rate in which the variable values are to be read from the control

Tab. 5-5: WinCC - flexible variable declaration

In WinCC-flexible 2007, the following data types are available. The corresponding IndraLogic data types are listed below.

Application Examples

WinCC - flexible data types

WinCC - flexible	IndraLogic	Description
Int	UINT	Unsigned integer length 16 bit Value range: 0 to 65535
+/-int	INT	Signed integer length 16 bit Value range: -32768 to 32767
Double	UDINT	Unsigned double integer length 32 bit Value range: 0 to 4294967295
+/-double	DINT	Signed double integer length 32 bit Value range: -2147483648 to 2147483647
Float	REAL	Floating point length 32 bit Value range: 1.175494351e-38F to 3.402823466e+38F
Bit	Single bit of a byte	Single bit length 1 bit ("Bit data" on page 161)
16 bit group	WORD	WORD length 16 bits Attention: For 16 bit groups, the bit counting in WinCC-flexible is vice versa to the one in IndraLogic
ASCII	STRING	The string length (in bytes) is to be configured in WinCC-flexible via the Tag Properties. ("Text string" on page 162)

Tab. 5-6: WinCC-flexible data types/IndraLogic data types

The declared variables can now be used for different WinCC-flexible functionalities.

Application Examples

Example:

Displaying and entering different variables

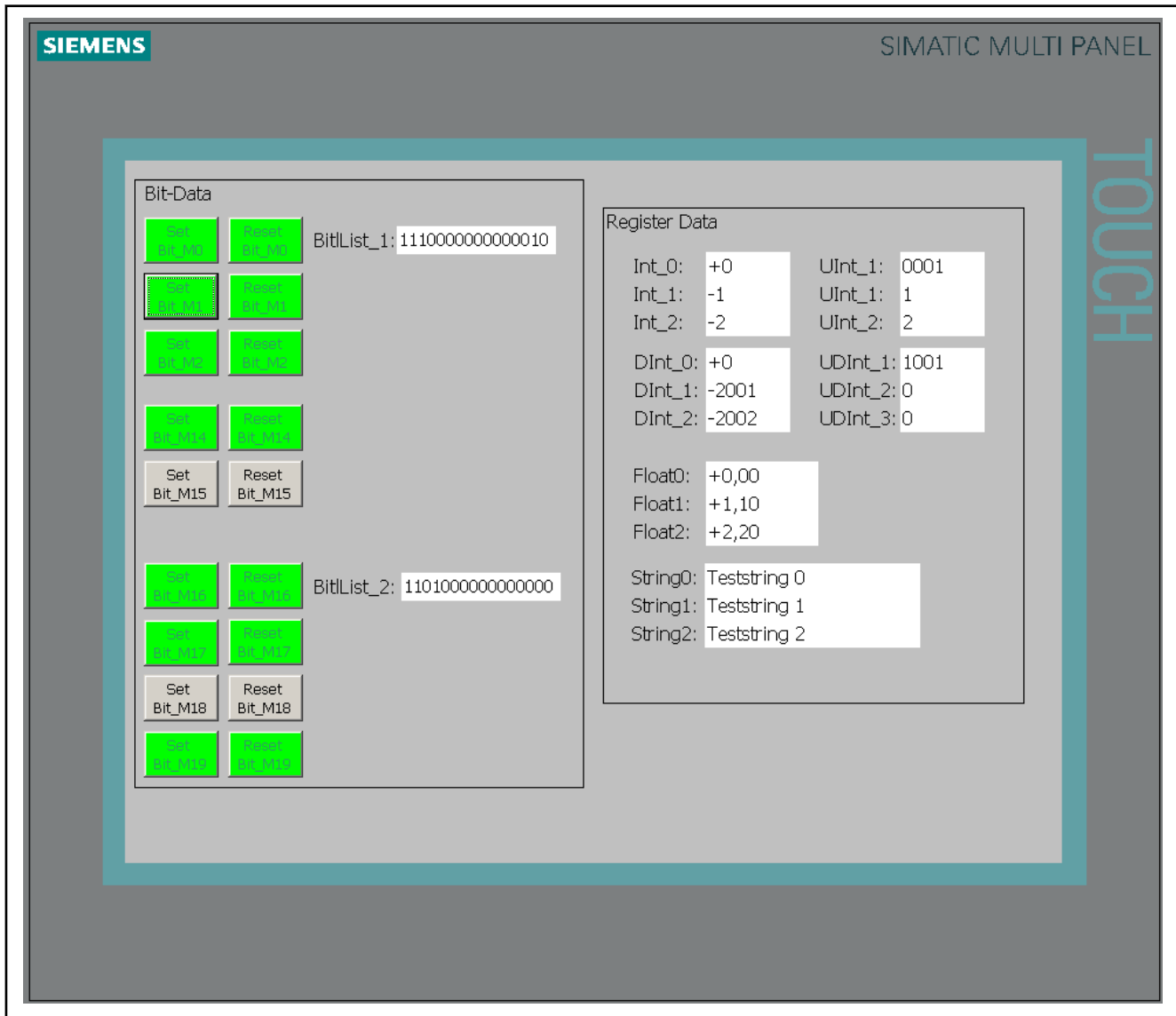


Fig. 5-46: WinCC-flexible - Visualization example

Configuration with Pro-face GP-Pro EX 2.2

Communication settings

The following settings are required to configure the connection in Pro-face GP-Pro EX 2.2:

1. Select the **Device/PLC** entry in the **"System Settings"** menu.
2. Execute the **"Add Device/PLC"** link in the working area of the window
3. Subsequently, the following settings have to be selected in the **"Add Device/PLC"** dialog:

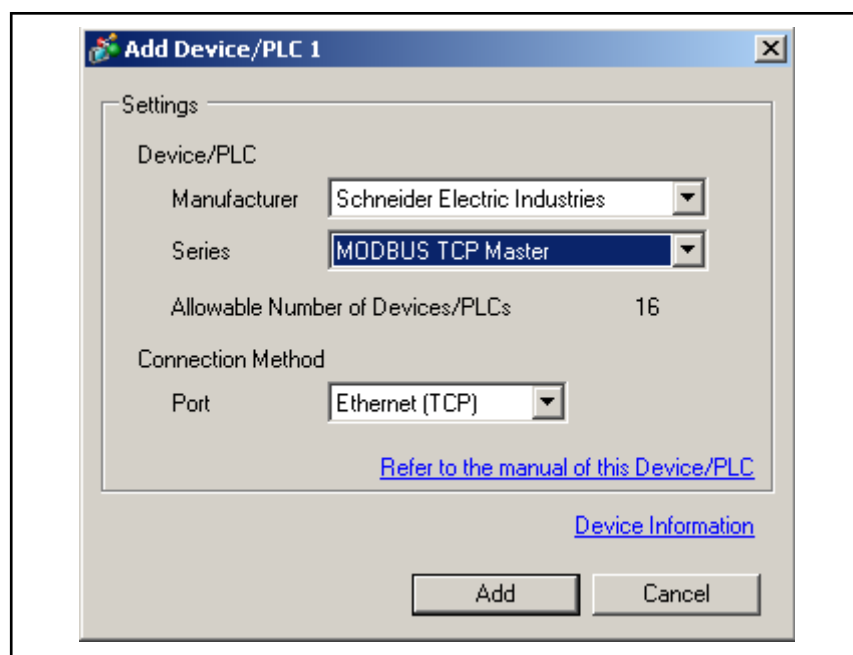


Fig. 5-47: GP-Pro EX communication settings - Device selection

Drop-down menu	Selection
Manufacturer	Schneider Electric Industries
Series	MODBUS TCP master
Port	Ethernet (TCP)

Tab. 5-7: Communication settings - Device selection

Acknowledge your selection by clicking on **Add**

- Enter the connection name in the extended window of the working area under "Device Name" and click on  **"Settings"**.

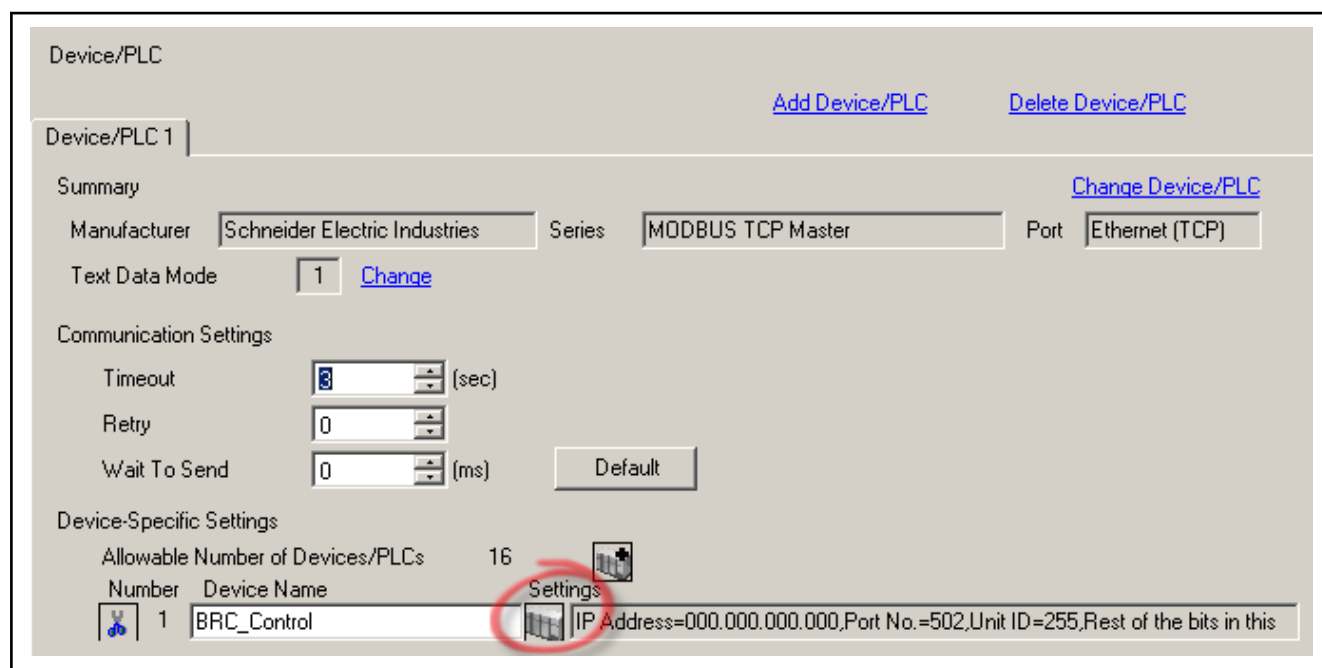


Fig. 5-48: GP-Pro EX communication settings - Working range

Application Examples

5. Subsequently, the following settings have to be selected in the "Equipment Configuration" tab in the "Individual Device Settings" dialog:

The screenshot shows the 'Individual Device Settings' dialog box for 'BRC_Control'. The 'Equipment Configuration' tab is selected. The 'Equipment Address' section contains three input fields: 'IP Address' with the value '192.168.1.34', 'Port No.' with the value '502', and 'Unit ID' with the value '255'. Below this, the 'Bit manipulation (set/reset) to Holding Register' section has two radio buttons: 'Clear' and 'Do not clear', with 'Do not clear' being selected. A note below states: 'Note on when selecting "Do not clear": If the ladder program writes data to Holding Register during the read/write process, the resulting data may be incorrect.' The 'IEC61131 Syntax' section has a checked checkbox and a dropdown for 'Address Mode' set to '0-based (Default)'. Below it is a warning: 'If you change the series, please reconfirm all address settings.' The 'Variables' section has a dropdown for 'Double Word word order' set to 'Low word first(L/H)'. At the bottom, there is an unchecked checkbox for 'Low Security Level' and three buttons: 'Default', 'OK (0)', and 'Cancel'.

Fig. 5-49: GP-Pro EX communication settings - Device parameters

Section	Name	Value	Description
Equipment Address	IP Address	"192.168.1.34"	IP address of the control
	Type	"502"	Port of the communication connection (port of the IL_ModbusTCPServer function block instance)
	Unit ID	"255"	Maintain default setting (255) ! not relevant for the Modbus TCP - refer to "MODBUS Messaging on TCP/IP Implementation Guide V1.0b" on www.Modbus-IDA.org
Bit manipulation to Holding Register	Reset of the bits in this word	"Do not clear"	Maintain default setting (Do not clear)

Application Examples

Section	Name	Value	Description
IEC61131 Syntax	Address Mode	"0-based (default)"	Activate IEC61131 Syntax and maintain Address Mode for the default setting (0-based)
Variables	Double Word word order	"Low word first (L/H)"	Word order within a double word

Tab. 5-8: GP-Pro EX communication settings - Device parameters

The "MaxQuery" register defines the maximum number of data that can be packed in the Modbus request. The default settings can be retained.

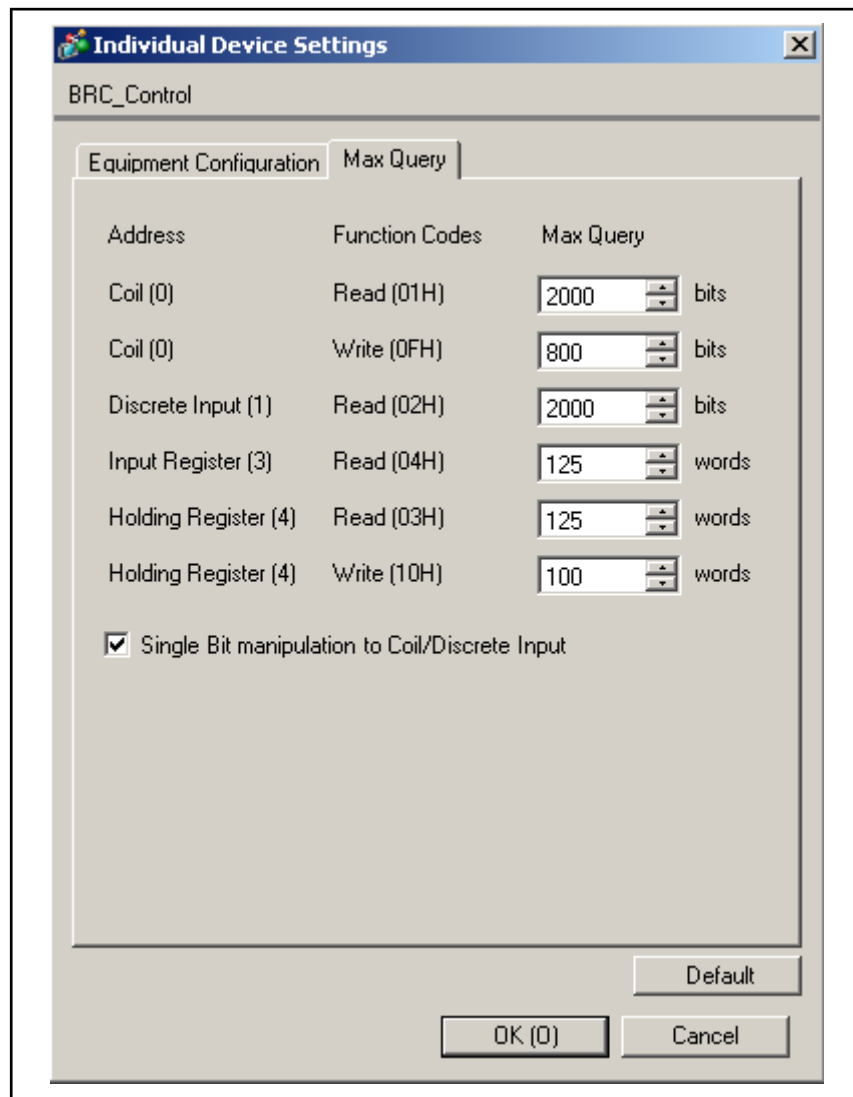
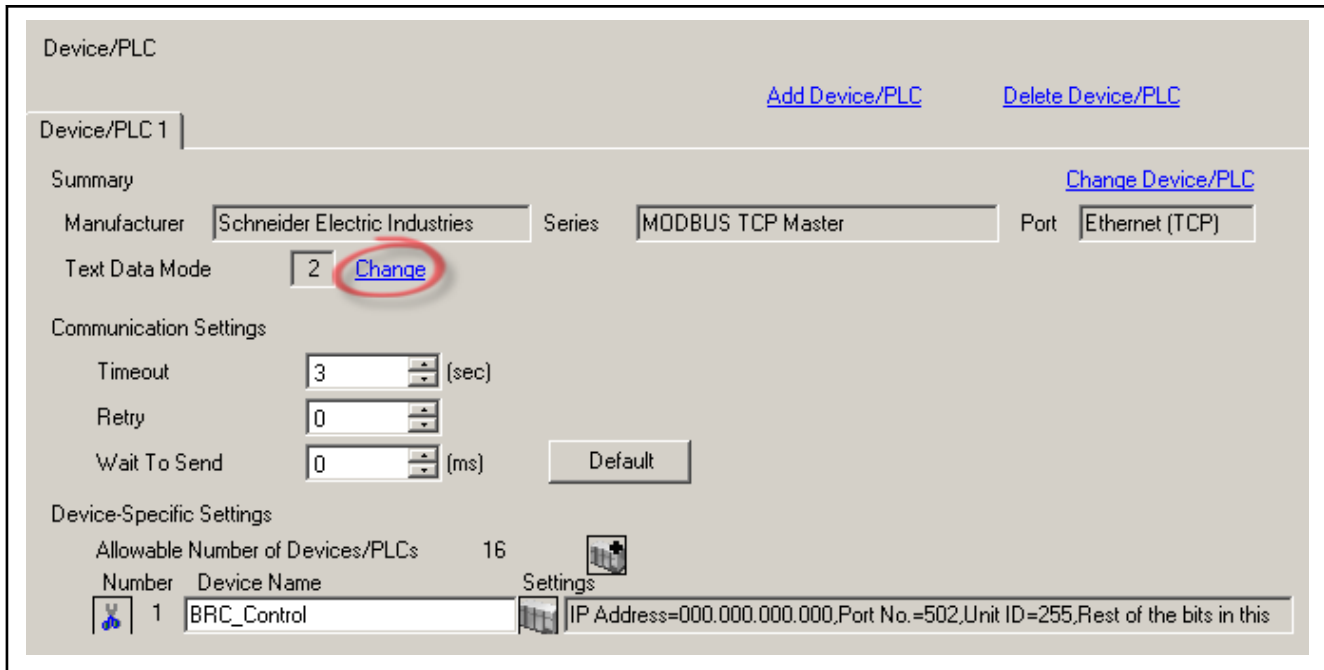


Fig. 5-50: GP-Pro EX communication settings - Device parameters

Confirm selection with <OK>

- The "Text Data Mode" have to be adjusted to allow interpretation of texts. Therefore, click on the **"Change"** link in the working area window.

Application Examples



Device/PLC

[Add Device/PLC](#) [Delete Device/PLC](#)

Device/PLC 1

Summary [Change Device/PLC](#)

Manufacturer Series Port

Text Data Mode [Change](#)

Communication Settings

Timeout (sec)

Retry

Wait To Send (ms)

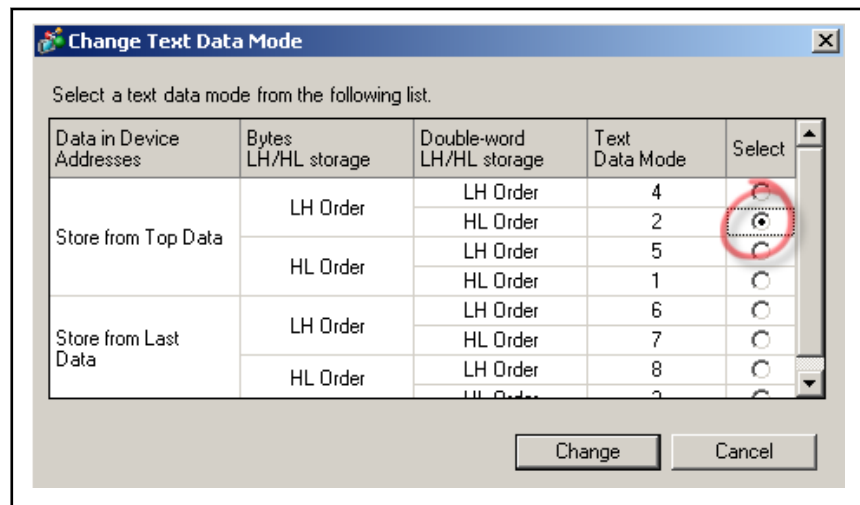
Device-Specific Settings

Allowable Number of Devices/PLCs 16

Number	Device Name	Settings
1 1	BRC_Control	IP Address=000.000.000.000,Port No.=502,Unit ID=255,Rest of the bits in this

Fig. 5-51: GP-Pro EX communication settings - Working range

Subsequently, select 2 in the "Change Text Data Mode" dialog and confirm with the <Change> button.



Change Text Data Mode

Select a text data mode from the following list.

Data in Device Addresses	Bytes LH/HL storage	Double-word LH/HL storage	Text Data Mode	Select
Store from Top Data	LH Order	LH Order	4	☐
		HL Order	2	☒
	HL Order	LH Order	5	☐
		HL Order	1	☐
Store from Last Data	LH Order	LH Order	6	☐
		HL Order	7	☐
	HL Order	LH Order	8	☐

Fig. 5-52: Dialog: Change Text Data Mode (GP-Pro EX communication settings)

Using addresses

After configuring the connection, the addresses can be used directly in different GP-Pro EX functionalities, refer to ["Addressing data from a PLC project"](#) on page 161.

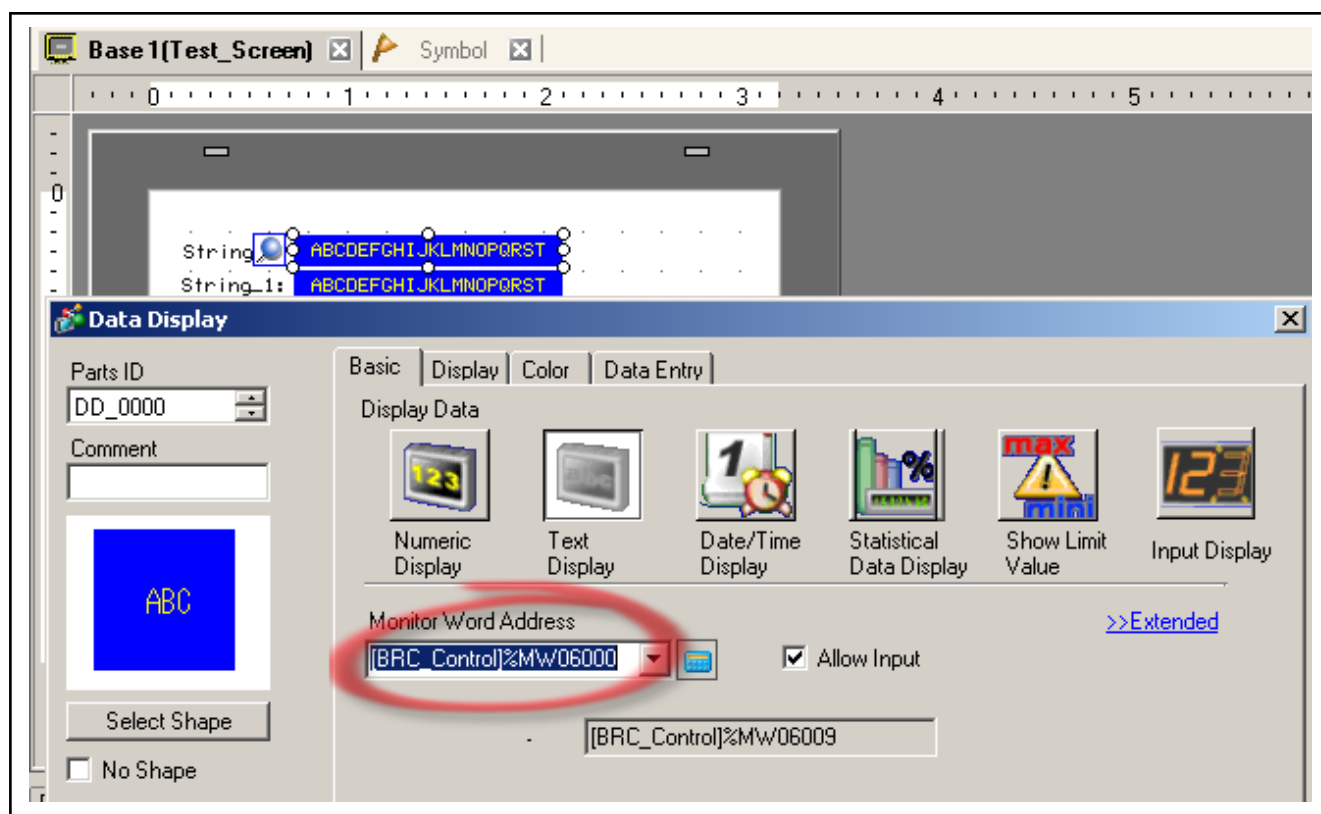


Fig. 5-53: GP-Pro EX - Direct address use

Variable Declaration

Symbol variables of type bit address and word address can also be declared. For details, please refer to ["Addressing data from a PLC project" on page 161](#). The address content can be used in different GP-Pro EX functionalities via these symbol variables.

To declare the symbol variables in the working area of the window, open **Common Settings ► Symbol Variables** in the menu.

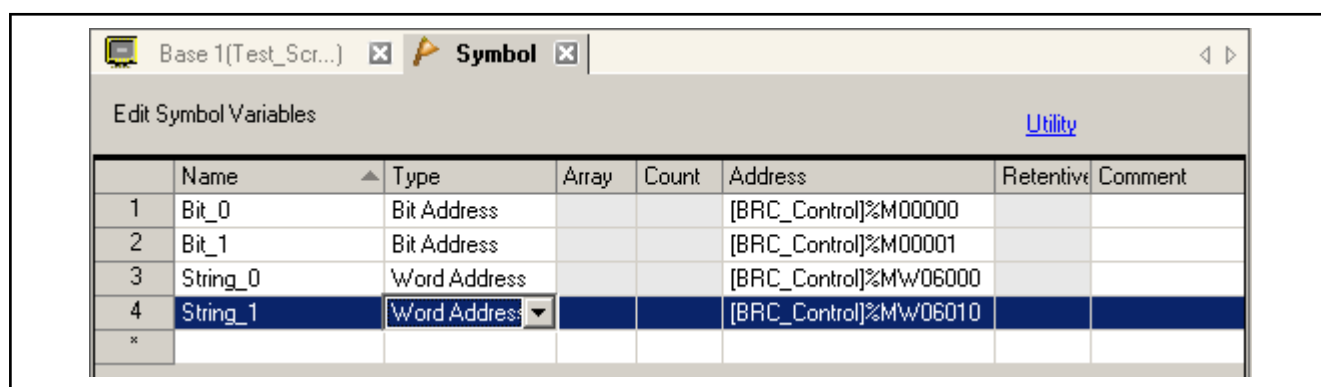


Fig. 5-54: GP-Pro EX - Create a symbol variable

After creating the symbol variable, it can be used as "placeholder" for the address.

Application Examples

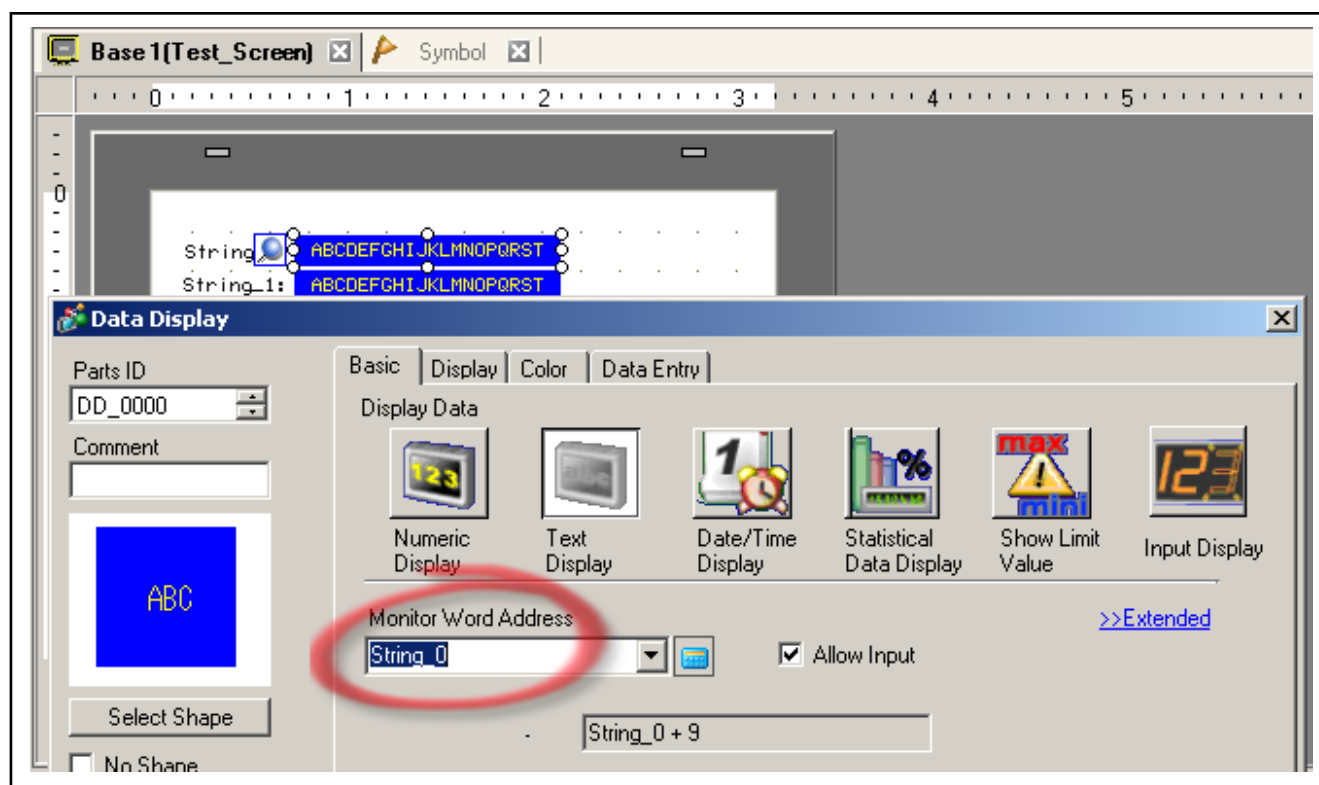


Fig. 5-55: GP-Pro EX - Using symbol variables



The number of the registers to be interpreted is defined via the data type used in the individual display fields.

For display fields of type "Text Display", the number of characters to be displayed are defined via the number of the registers to be interpreted.

Example:

Displaying and entering different variables

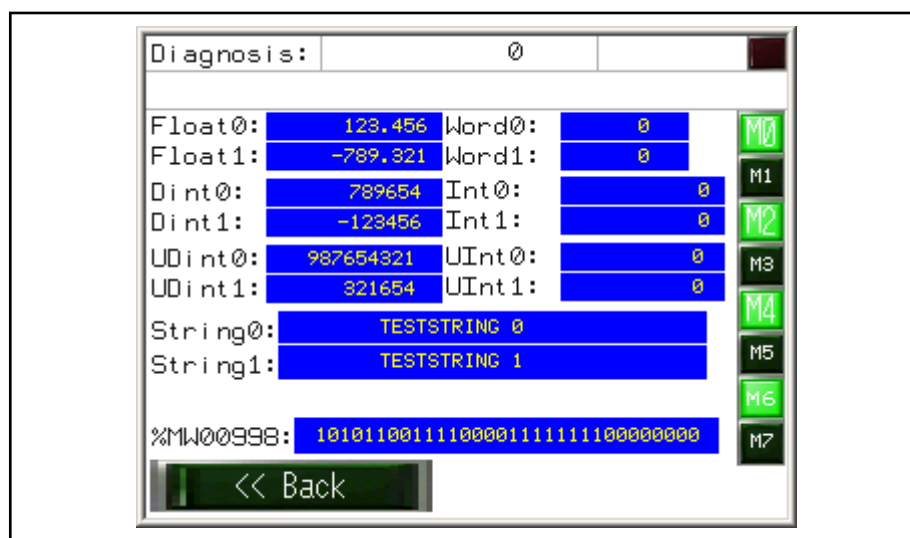


Fig. 5-56: Variable display

6 Client Development

6.1 OPC Client Visual Basic .NET

6.1.1 General Information

This section deals with the development of the OPC Client "Visual Basic .NET" using "Microsoft Visual Studio .NET 2005".

This section deals with the following topics:

- Creating projects
- Adding References
- Adding forms and control elements
- OPC Client implementation

The following diagram shows the OPC Client as active Windows application.

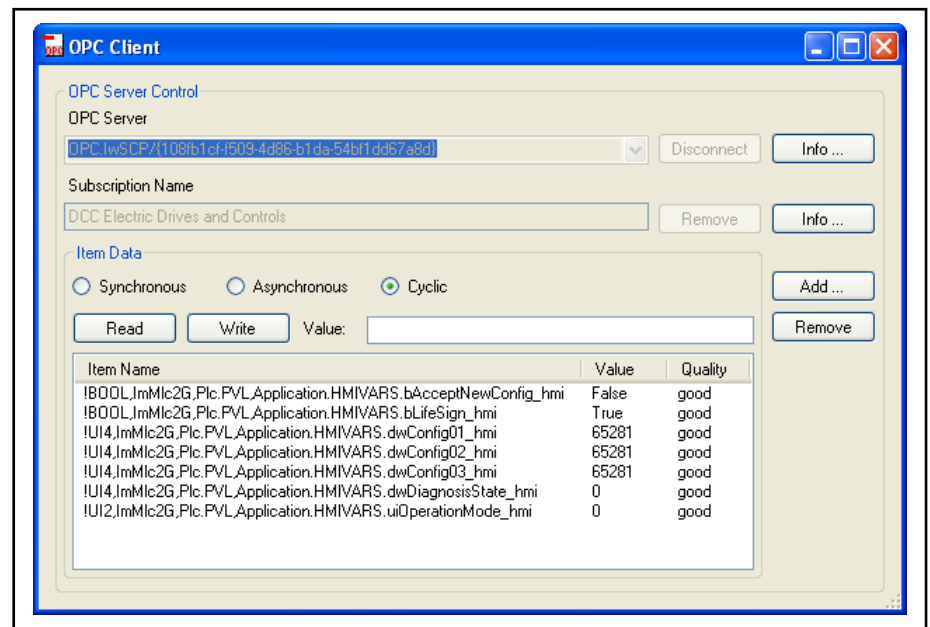


Fig. 6-1: OPC Client interface

6.1.2 Visual Basic Project "OPC Client"

New Project

To create a new project in the "Microsoft Visual Studio .NET 2005" development environment, click on the link "New Project" on the starting page or on **File ► New and Project....**

The "New Project" dialog is displayed as shown below.

Select project type and template:

- "Other Languages -> Visual Basic -> Windows" or
- "Visual Basic -> Windows" and
- "Windows Application as template"

Enter "OPCClient" as project name and adapt the suggested storage location.

Create the with **OK**.

Client Development

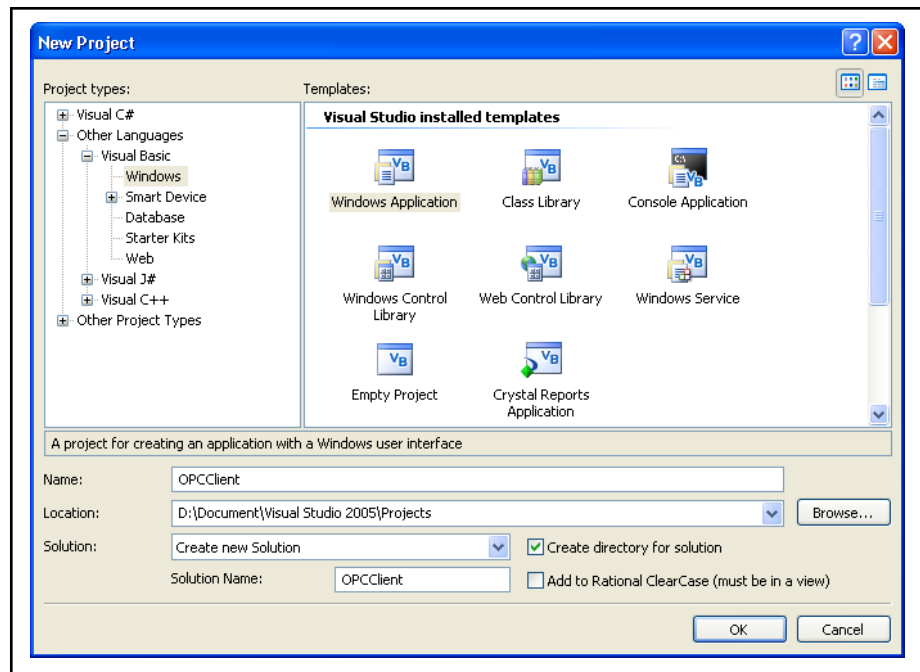


Fig. 6-2: Dialog: New Project

Adding References

The project has to contain a reference to "OpcNetApi.dll" and "OpcNetApi.Com.dll" to access the OPC .NET API functionalities.

"References" can be managed on the "Reference" page of the project properties. Additional dependencies can be added via the "Add Reference" dialog. Open them with "Add...". In the "Add Reference" dialog, the two files "OpcNetApi.dll" and "OpcNetApi.Com.dll" can be added to the project on the "Browse" tab.

Both files are provided with the "IndraWorks Suite" installation and saved in the installation folder (e.g. "C:\Program Files\Rexroth\IndraWorks").

Open the files to apply them to the "References:" list.

The list "Imported namespaces:" shows all namespaces available for the project.

All selected namespaces can be used in the project without explicit declaration using the "Import" statement.

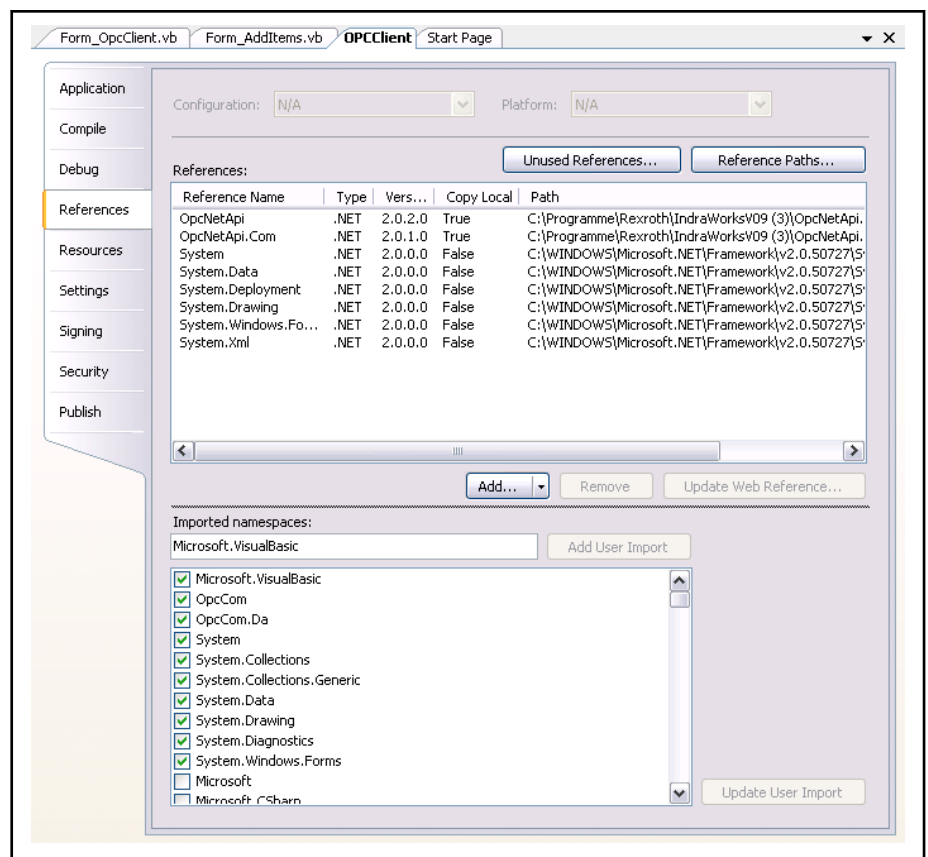


Fig. 6-3: Project properties -References

Adding Windows Form and Control Element

Add a Windows to the starting project via **Project ► Add Windows Form....**

In the "Add New Element - OPCClient" dialog, the "Windows form" template is already selected and only the name of the form has to be entered.

All Windows forms created in the OPC Client example have the identifier name "Form_" as prefix.

The following figure shows an overview on the "OPC Client" project in the "Solution Explorer".

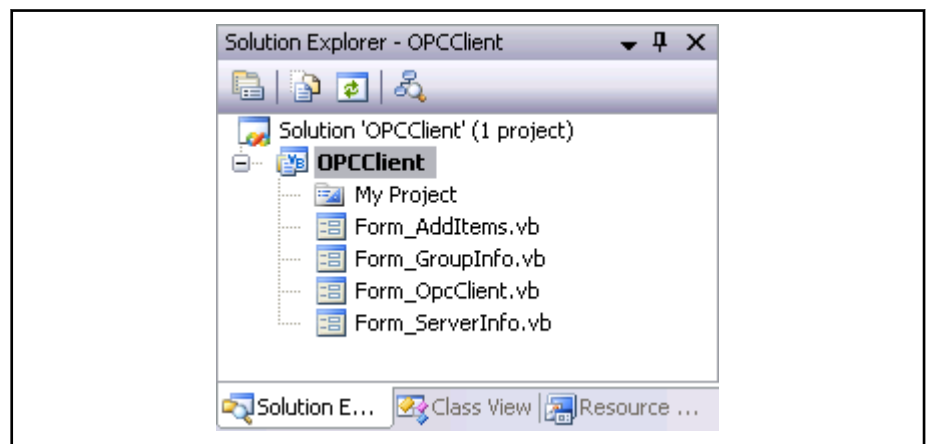


Fig. 6-4: Project overview

Client Development

"Windows forms" are designed with the "Windows Forms Designer" and the control elements in the "Toolbox". Select the control elements and place them on the form.

The control elements for the selected object are configured in the "Properties" window.

Size and position of the control elements can also be changed using the mouse in the Windows Forms Designer.

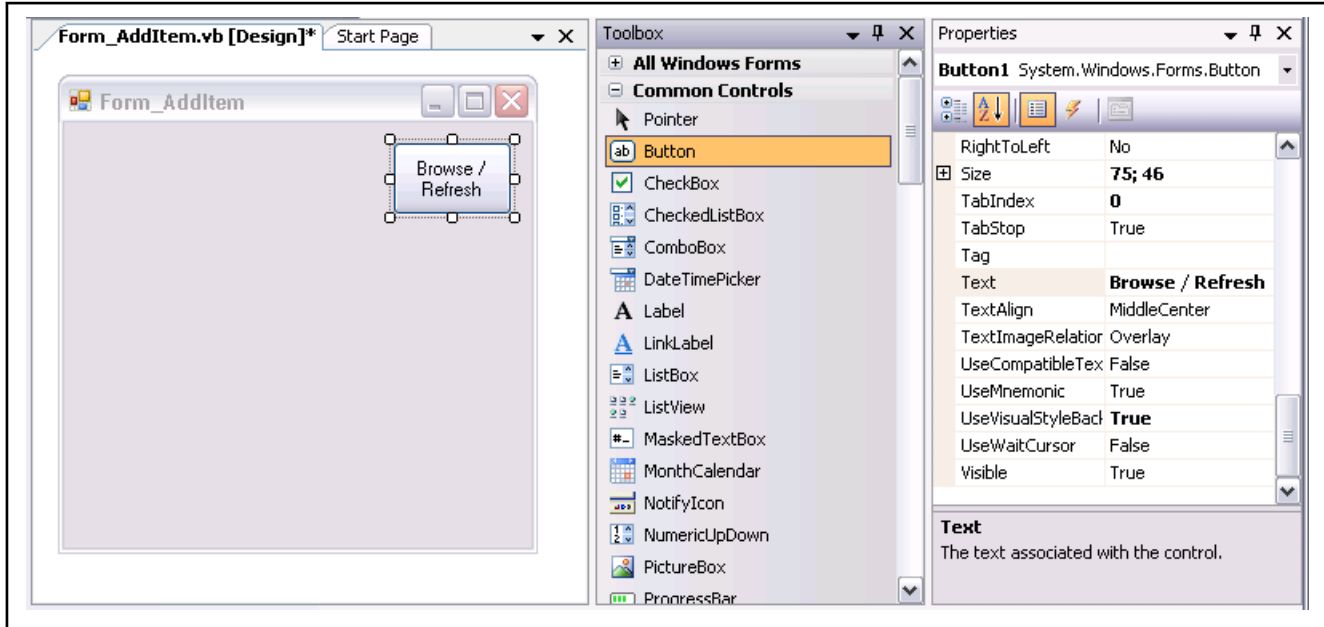


Fig. 6-5: Adding and configuring control elements

The following figure shows the "OPC Client" form in the "Windows Forms Designer".

To maintain clarity, a detailed description of the form and the individual control elements is not provided. All configurable properties can be taken from the example project.

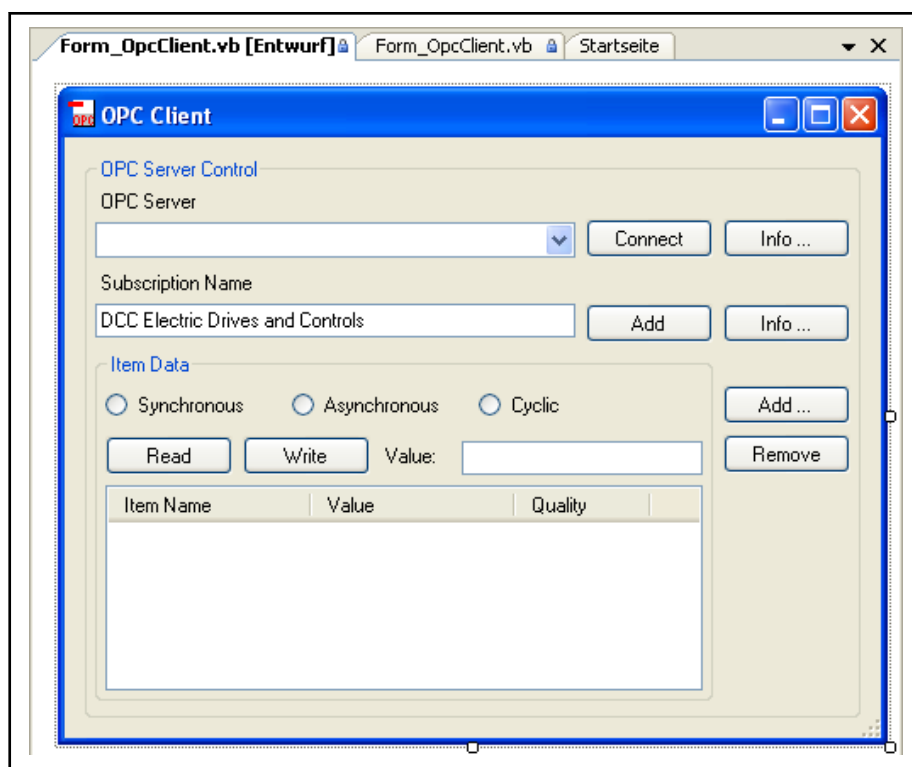


Fig. 6-6: Dialog: OPC Client (Windows Forms Designer)

Implementing OPC Client

General Information

The following section describes the procedure when programming the OPC Client.

To provide a better overview in the program, individual sections of the class are summarized with the "#Region" directive. These "regions" can be expanded or collapsed when creating the program.

The figure below shows the defined areas using the example of the "Form_OpcClient" form.

Area	Description
"#Region" -"#End Region"	
"Declaration"	Contains class declaration (member variables and constants).
"Form Events"	Procedures for handling the events triggered by the form and the included control elements.
"OPC Events"	Procedures for handling the events triggered by the OPC server.
"Additional Internal Functions"	Additional internal definitions, structures and functions to optimize user guidance.

Tab. 6-1: Program structuring using "#Region" directives

The relevant program lines in the example project are also identified with the keyword **"Step:"** and numbered consecutively. Using the search "Search in Files" (<Ctrl>+<Shift>+<F>) in Visual Studio allows branching directly to the implementation described by entering the search word e.g. **"Step: 2"**.

Client Development

The implementation described in the following was documented as the operating order of the "OPC Client" application.

The structure is shown in the following:

Chapter heading	Description
chapter "Declaring Variables" on page 178	Declaring variables to manage OPC server and Windows form.
chapter "Listing OPC server" on page 179	Listing of all OPC servers installed on the local PC.
chapter "Creating and Canceling OPC Server Instances" on page 179	Creating and canceling the OPC server instance and displaying the OPC server information.
chapter "Creating and Canceling OPC Subscription" on page 181	Creating and canceling an OPC subscription and displaying the subscription information.
chapter "Listing Control Items" on page 183	Reading and displaying all control items (browsing).
chapter "Adding and Deleting OPC Items" on page 185	Adding and deleting OPC items.
chapter "Reading Control Data" on page 187	Reading "synchronous" and "asynchronous" control data.
chapter "Writing Control Data" on page 190	Writing "synchronous" and "asynchronous" control data.
chapter "Cyclic Reading of Control Data" on page 192	Cyclic control data reading.

Tab. 6-2: Overview on the implementation structure

Declaring Variables

In the "Step: 1" declaration section, the "Form_OpcClient" variables are declared. The syntax chosen for the names ensures that all class variables can be written with the prefix "m_" ("member variables") and that the constant definitions are "capitalized".

The variables are

- the object for the OPC server instance and
- variables to manage the form.

The line **"Dim m_OpcServer As Opc.Da.Server = Nothing"** is the declaration relevant to the example. The "m_OpcServer" variable contains the created OPC server object.

```

' Step: 1
'
' declaration of members used for opc administration
#Region " Declaration "
' Step: 1
'
' declaration of members used for opc server administration
Dim m_OpcServer As Opc.Da.Server = Nothing

'
' members to store configuration
Dim m_DialogAddItems As OPCClient.Form_AddItems
Dim m_InternalStateControl As InternalStateControl

'
' constant values
Const BUTTON_DESCRIPTION_SERVER_CONNECT As System.String = "Connect"
Const BUTTON_DESCRIPTION_SERVER_DISCONNECT As System.String = "Disconnect"
Const BUTTON_DESCRIPTION_GROUP_ADD As System.String = "Add"
Const BUTTON_DESCRIPTION_GROUP_REMOVE As System.String = "Remove"
Const MAX_ITEMS As System.Int32 = 10

#End Region

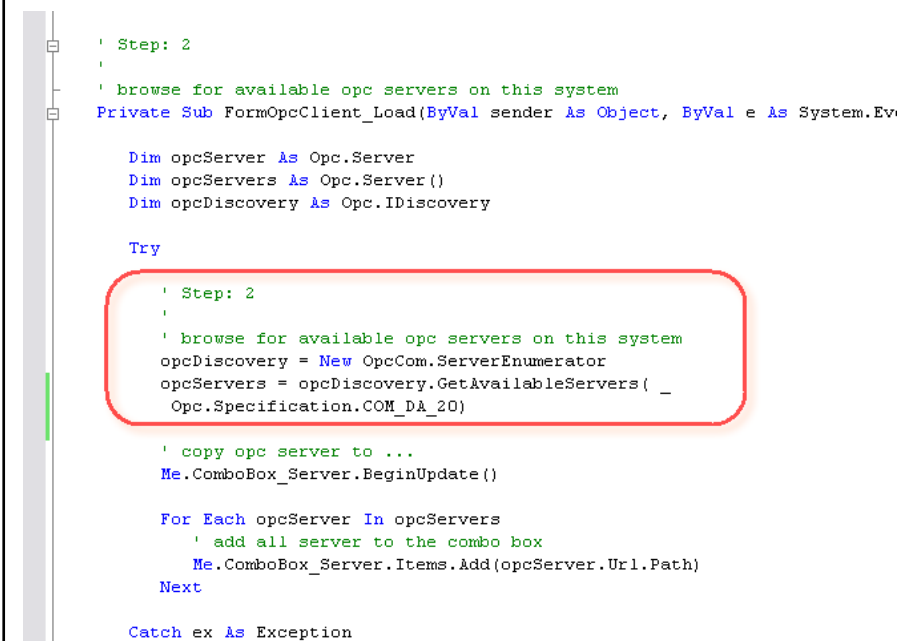
```

Fig. 6-7: Declaring class variables

Listing OPC server

All OPC servers installed locally on the PC are listed in the "Load Event" **"Step: 2"** of the "Form_OpcClient" form. All OPC servers are requested via the "opcDiscovery.GetAvailableServers()" method of the Discovery interface "Opc.IDiscovery" of the "OpcEnum.exe" service supporting the specification "Data Access 2.0". The available OPC specifications are defined in the "Opc.Specification" structure.

The list provided by the Discovery interface contains the complete URL path with ProgID (without version index) and GUID to clearly identify the server.



```

' Step: 2
'
' browse for available opc servers on this system
Private Sub FormOpcClient_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles MyBase.Load

    Dim opcServer As Opc.Server
    Dim opcServers As Opc.Server()
    Dim opcDiscovery As Opc.IDiscovery

    Try

        ' Step: 2
        '
        ' browse for available opc servers on this system
        opcDiscovery = New OpcCom.ServerEnumerator
        opcServers = opcDiscovery.GetAvailableServers( _
            Opc.Specification.COM_DA_20)

        ' copy opc server to ...
        Me.ComboBox_Server.BeginUpdate()

        For Each opcServer In opcServers
            ' add all server to the combo box
            Me.ComboBox_Server.Items.Add(opcServer.Url.Path)
        Next

    Catch ex As Exception
    End Try
End Sub

```

Fig. 6-8: Listing installed OPC servers

Creating and Canceling OPC Server Instances

General Information

This section describes how to create and cancel the OPC server instance and how to read the server information. The OPC Server instance is managed in the event handler "Button_Connect_Disconnect_Click" either of the button **"Connect"** or **"Disconnect"**. Server information is ready and displayed via the function "Button_ServerInfo_Click" of the **"Info ..."** button.

The relevant code segments in the OPC Client example project are:

- **"Step: 3"** - Create OPC server instance
- **"Step: 4"** - Delete OPC server instance
- **"Step: 5"** and **"Step: 21"** - Read server information

Creating OPC Server Instance - Connect

Creating an OPC server instance and establishing the connection is described in **"Step: 3"** (see the following figure).

The first step creates an OPC server instance "New Opc.Da.Server" with "OpcCom.Factory" application parameter and the "Opc.URL".

The "Opc.URL" provides information on the OPC specification "Opc.UrlScheme.DA", the PC name "localhost" or the IP address the server is to be started with as well as the server identifier.

Client Development

In the example project, the server ID is retrieved from the combination list field "ComboBox_Server".

The server ID can be specified as follows:

- "OPC.lwSCP"
ProgID without version index
- "OPC.lwSCP.1"
ProgID with version index
- "OPC.lwSCP/{108fb1cf-f509-4d86-b1da-54bf1dd67a8d}"
ProgID without version index and GUID
- "OPC.lwSCP.1/{108fb1cf-f509-4d86-b1da-54bf1dd67a8d}"
ProgID with version index and GUID

The OPC server is instantiated by calling the function "Me.m_OpcServer.Connect()".

Deleting OPC Server Instance - Disconnect

The deletion of the OPC server instance is shown in "Step: 4".

Calling the function "Me.m_OpcServer.Disconnect()" causes logout from OPC server. If no further client is connected to the server at this point, it is closed.

The function "Me.m_OpcServer.Dispose()" is then called to release resources kept in the OPC server instance and the instance is canceled by "Me.m_OpcServer = Nothing".

```
' show hour glas
Me.Cursor = Cursors.WaitCursor

If Me.Button_Connect_Disconnect.Text = BUTTON_DESCRIPTION_SERVER_CONN
    ' Step: 3
    '
    ' select opc server and
    Me.m_OpcServer = New Opc.Da.Server(New OpcCom.Factory, _
    New Opc.URL(Opc.UrlScheme.DA + "://localhost/" + _
    Me.ComboBox_Server.Text))
    ' connect with ...
    Me.m_OpcServer.Connect()

    ' Enable or disable controls
    Me.ComboBox_Server.Focus()
    Me.SetInternalStateControl(InternalStateControl.Connected)

ElseIf Me.Button_Connect_Disconnect.Text = BUTTON_DESCRIPTION_SERVER_
    ' Step: 4
    '
    ' disconnect from server and clean up
    Me.m_OpcServer.Disconnect()
    Me.m_OpcServer.Dispose()
    Me.m_OpcServer = Nothing

    ' Enable or disable controls
    Me.SetInternalStateControl(InternalStateControl.Disconnected)

    ' clear browse info
    If Not Me.m_DialogAddItems Is Nothing Then Me.m_DialogAddItems.Cle

End If
```

Fig. 6-9: Creating and canceling OPC server instance

Reading Server Information

The information provided by the OPC Server is read and displayed in the "Load Event" **"Step: 21"** of the form "Form_ServerInfo".

The form is called in **"Step: 5"** (without figure).

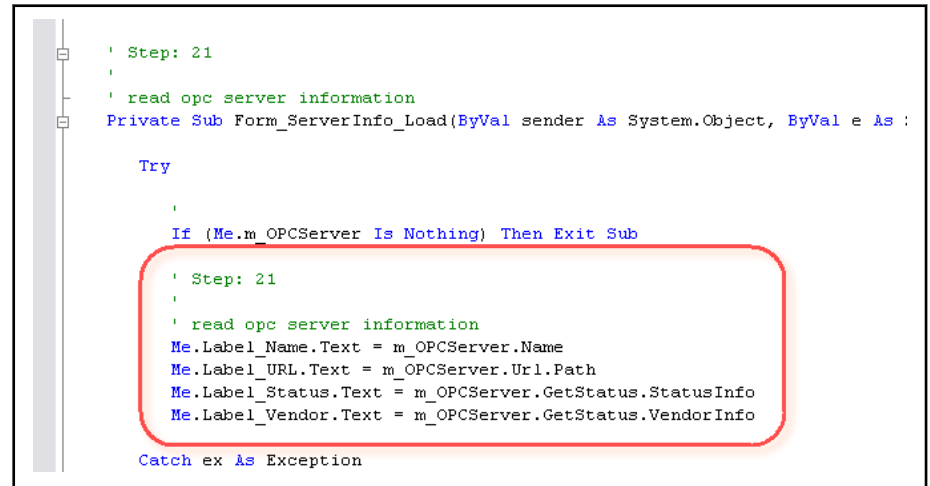


Fig. 6-10: Reading OPC server information

Creating and Canceling OPC Subscription

General Information

This section describes login and logout of a "subscription" (group) on the OPC server. The subscription is managed in the "Button_Add_Remove_Group_Click" event handler of the **"Add"** or **"Remove"** button and the subscription information is read and displayed in the "Button_GroupInfo_Click" function of the **"Info ..."** button.

The relevant code segments in the OPC Client example project are:

- **"Step: 6"** - Create subscription
- **"Step: 7"** - Cancel subscription
- **"Step: 8"** and **"Step: 22"** - Display subscription information

Creating a Subscription - CreateSubscription

Create a subscription in the OPC server collection using the function "Me.m_OpcServer.CreateSubscription()" and the application of the parameter "Opc.Da.SubscriptionState" (**"Step: 6"**, see [fig. 6-11 "Creating OPC subscription" on page 182](#)).

The elements of the "Opc.Da.SubscriptionState" class clearly identify the subscription and configure with regard to behavior or processing by the OPC server.

The following table describes the class elements.

Class element	Description
Active	The control cyclically reads the subscription items and saves them in the server cache.
ClientHandle	"Unique Identifier" (specified by the client).
Deadband	Smallest possible change in percentage of a value to trigger an event to the client.

Client Development

Class element	Description
KeepAlive	Largest possible time interval between two events to the client (life sign).
Local	Definition of localization in the format "[language Code]-[country/region code]", e.g. "de-DE", "en-US"
Name	"Unique Name" (specified by client). Specified by the "TextBox_Group.Text" text box.
ServerHandle	"Unique Identifier" (specified by server).
UpdateRate	Time interval the server checks data and sends an event to the client.

Tab. 6-3: Elements of the "Opc.Da.SubscriptionState" class

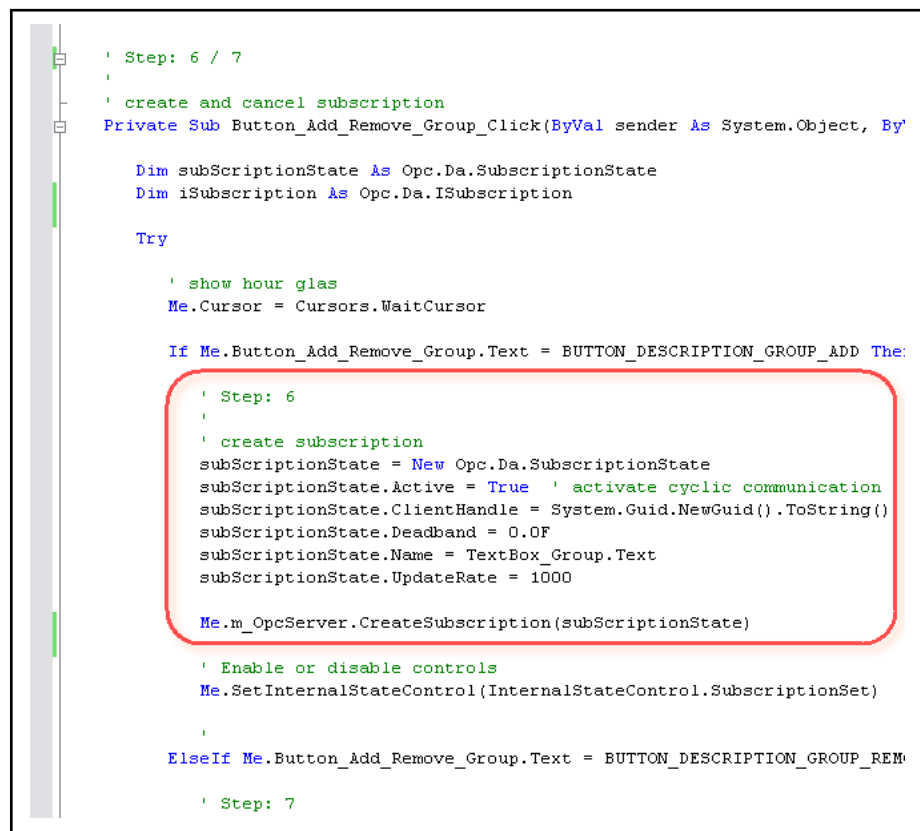


Fig. 6-11: Creating OPC subscription

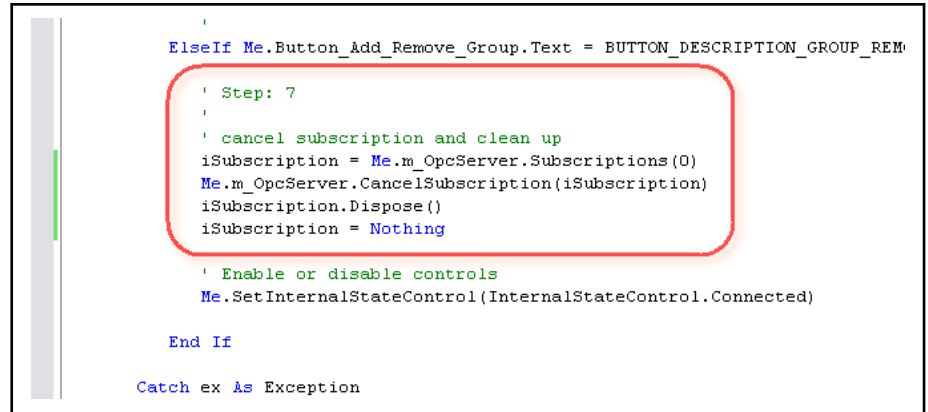
Canceling Subscription - CancelSubscription

The function "Me.m_OpcServer.CancelSubscription()" is called with the application of the parameter "Opc.Da.ISubscription" to cancel a subscription ("Step: 7").

Once the subscription has been canceled from the OPC server collection, call the "iSubscription.Dispose()" function. This enabled the resources used by the subscription. The reference of the interface variables is canceled by the assignment "iSubscription = Nothing".

The local assignment "iSubscription = Me.m_OpcServer.Subscription()" performed at the start of the code segment is necessary, as the subscription was removed from the OPC server collection after calling the function

"Me.m_OpcServer.CancelSubscription()". The subscription is no longer available.



```

ElseIf Me.Button_Add_Remove_Group.Text = BUTTON_DESCRIPTION_GROUP_REM
    ' Step: 7
    '
    ' cancel subscription and clean up
    iSubscription = Me.m_OpcServer.Subscriptions(0)
    Me.m_OpcServer.CancelSubscription(iSubscription)
    iSubscription.Dispose()
    iSubscription = Nothing

    ' Enable or disable controls
    Me.SetInternalStateControl(InternalStateControl.Connected)

End If

Catch ex As Exception

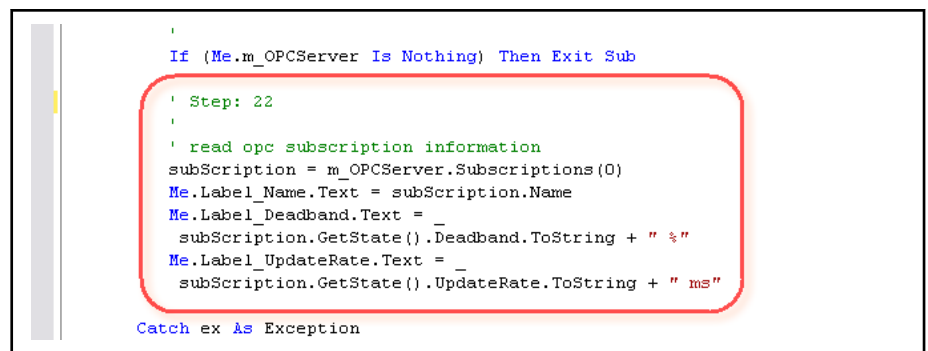
```

Fig. 6-12: Canceling OPC subscription

Displaying Subscription Information

The information provided by the subscription is read and displayed in the "Load Event" ("Step: 22") in the form "Form_GroupInfo".

The form is called in "Step: 8" (without figure).



```

If (Me.m_OPCTServer Is Nothing) Then Exit Sub

' Step: 22
'
' read opc subscription information
subScripton = m_OPCTServer.Subscriptions(0)
Me.Label_Name.Text = subScripton.Name
Me.Label_Deadband.Text = _
    subScripton.GetState().Deadband.ToString + " %"
Me.Label_UpdateRate.Text = _
    subScripton.GetState().UpdateRate.ToString + " ms"

Catch ex As Exception

```

Fig. 6-13: Reading OPC subscription information

Listing Control Items

General Information

This section describes the reading of all items configured in the control for the access via OPC server.

The relevant code segments in the OPC Client example project are:

- "Step: 20", "Step: 20.1" and "Step: 20.2" - Reading control items

Reading Control Items

Clicking on the "Add ..." button on the "Form_OpcClient" form, the "Form_AddItems" form is called. As it can take some time to read the items, the items on the OPC server are not read in the "Load Event" of the form but by using the "Button_BrowseItems_Click" function activated via the **"Browse / Refresh"** button.

The two classes "Opc.ItemIdentifier" and "Opc.Da.BrowseFilters" of the "Me.m_OPCTServer.Browse()" function are applied as parameters to read the items ("Step: 20.2").

The classes are provided with the following function:

Client Development

Opc.ItemIdentifier The class "Opc.ItemIdentifier" sets the "starting point" for the search in the "tree hierarchy". Consequently, when e.g. a search was canceled, it can be continued at this point.

Opc.Da.BrowseFilter The class "Opc.Da.BrowseFilter" is used as filter criterion for the function call result. Only variables beginning with a certain character sequence can be read for example.

As there is no default specification for the format in which an OPC server returns the "browsing" information, reading the control items was divided into two functions ("Step: 20.1" and "Step: 20.2"). This division allows reading flat hierarchies (tree structure without sub-nodes) and structured hierarchies (tree structure with sub-nodes) (see following figures).

```
' clear list, stop updating
Me.ListBox_Available_Items.BeginUpdate()
Me.ListBox_Available_Items.Items.Clear()

' Step: 20.1
'
' browse items
filters.ElementNameFilter = ""
filters.BrowseFilter = Opc.Da.browseFilter.all
filters.ReturnAllProperties = True
filters.ReturnPropertyValues = True
itemID.ItemName = ""
Me.BrowseItems(itemID, filters)

Catch ex As Exception
```

Fig. 6-14: Reading control items - Filter configuration

```
' Step: 20.2
'
' browse items
elements = m_OPCTServer.Browse(itemID, Filters, position)

' adds each element to the list
For Each element In elements
'
' avoid errors while using .NET API 1.30
If Not element.ItemName Is Nothing Then
itemName = element.ItemName
Else
itemName = element.Name
End If
Me.ListBox_Available_Items.Items.Add(itemName)

If (element.HasChildren = True) Then
' check on subordinate items

' set branch to search
item.ItemName = itemName & element.ItemName
item.ItemPath = element.ItemPath
Me.BrowseItems(item, Filters)

End If

Next
```

Fig. 6-15: Reading control items - Recursion

Adding and Deleting OPC Items

General Information

This section describes the addition and deletion of items to/from a subscription.

The relevant code segments in the OPC Client example project are:

- **"Step: 9", "Step: 9.1" and "Step: 9.2"** - Adding items
- **"Step: 10"** - Deleting items

Adding Items - AddItems

In the form event "Button_Add_Item_Click" (**"Step: 9"**) the form "Form_AddItems" is called and the items selected in the form are added.

The variables can be entered separately in the text field of the "Add Items" form by ";" or they can be selected from the list. When entering the variables, the item syntax of the respective OPC server is to be taken into account.

The following shows the item syntax for the "OPC.lwSCP" server and the "IndraLogic OPC" server.

"OPC.lwSCP" Server

- **PLC variables**
 "[[required data type],[control name or alias],[lwSCP service],[fully qualifying variable name]]"
Example:
 "!Bool,ImMlc2G,Plc.PVL,Application.HMIVARS.bLifeSign_hmi"
- **sercos parameter**
 "[[required data type],[control name or alias],[lwSCP service],[drive address], [parameter number], [element coding]]"
Example:
 "!UI4,ImMlc2G,SPA1,0,C-0-0001,D"
 "!UI4,ImMlc2G,SPA1,0,C-0-0001,40"



The "OPC.lwSCP" server allows the access to parameter and PLC variables!



For further information on the item syntax of the "OPC.lwSCP" server, refer to [chapter 4.3.3 "lwSCP and OPC.lwSCP Items" on page 93](#).

"IndraLogic OPC" server

- **PLC variables**
 "[Control name].[fully qualifying variable name]"
Example:
 "PLC_IL2G.Application.HMIVARS.bLifeSign_hmi"



The "IndraLogic OPC" server allows the access to PLC variables.

For the description of the "Browse" functionality, refer to [chapter "Listing Control Items" on page 183](#).

Once the items are entered or selected in the "Add Items" form, in **"Step: 9.1"** an "Opc.Da.Item" class is created and initialized for every item. The item name is assigned in the line "opcltems(index).ItemName = items(index)".

The class elements of class "Opc.Da.Item" used in the example are shown in the following table:

Client Development

Class element	Description
Active	"DataChanged" events for the item are sent to the client.
ActiveSpecified	The "Active" property was defined.
ItemName	Fully qualified item name. The syntax depends on the OPC server.
ItemPath	Fully qualified item path. This property is not analyzed by the "OPC.lwSCP".
MaxAge	When retrieving the item from the OPC server cache, the value may not be older than defined in "MaxAge".
MaxAgeSpecified	The "MaxAge" property was defined. If this property is set to "True", the value is retrieved from the OPC server "cache" or the "Device" depending on "MaxAge". If the property is set to "False", the value is always read from the "Device".

Tab. 6-4: Elements of class "Opc.Da.Item"

Once all OPC items are created, the OPC item array is logged in to the subscription on the OPC server with the function request "Me.m_OpcServer.Subscriptions(0).AddItems()" ("Step: 9.2").

```

' Step: 9
'
' add items to the subscription
ReDim opcItems(items.Length - 1)
For index = 0 To items.Length - 1

    ' Step: 9.1
    '
    ' creating new item
    opcItems(index) = New Opc.Da.Item
    ' set item properties
    Randomize()
    opcItems(index).ClientHandle = CInt(Int((1000 * Rnd()) + 1))
    opcItems(index).Active = True
    opcItems(index).ActiveSpecified = True
    opcItems(index).ItemName = items(index)
    opcItems(index).ItemPath = ""
    opcItems(index).MaxAge = -1
    'opcItems(index).MaxAgeSpecified = False ' read from device
    opcItems(index).MaxAgeSpecified = True ' read from cache

Next

' Step: 9.2
'
' add the item to the subscription and to the list view
itemResults = Me.m_OpcServer.Subscriptions(0).AddItems(opcItems)
For index = 0 To itemResults.Length - 1

    If (itemResults(index).ResultID.Succeeded = True) Then
        listViewItem = _
            Me.ListView_ItemData.Items.Add(opcItems(index).ItemName)
        listViewItem.SubItems.Add("") ' sub item 1
        listViewItem.SubItems.Add("") ' sub item 2
    End If

Next

```

Fig. 6-16: Adding items to a subscription

Removing Items - RemoveItems

To remove items from the OPC subscription, select the items from the list and click on **"Remove"**. In the subsequently shown **"Button_Remove_Item_Click"** form event, the function **"Me.m_OpcServer.Subscriptions(0).RemoveItems()"** is called removing the OPC item applied as parameter (**"Step: 10"**).

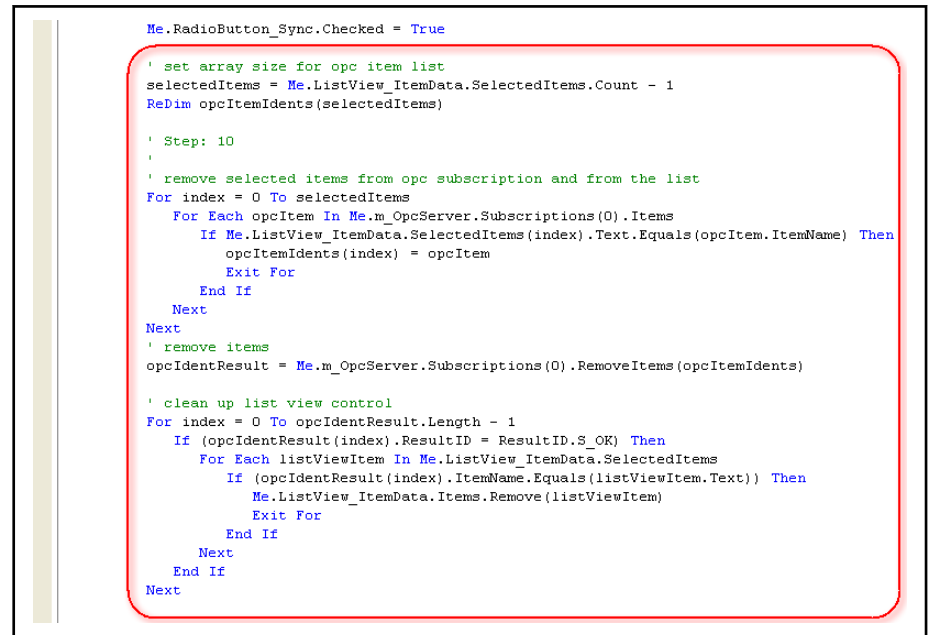


Fig. 6-17: Removing items from a subscription



When removing items from a group, remove the complete list of items. Thus, the administrative tasks for login and logout of the group only have to be executed once in the OPC server.

Reading Control Data

General Information

This section describes read access to control data. A distinction is made between

- "synchronous" and
- "asynchronous" access.

The control data is read in the **"Button_Read_Click"** event handler of the **"Read"** button.

The relevant code segments are:

- **"Step: 11"** - Synchronous reading
- **"Step: 12"** - Asynchronous reading
- **"Step: 13"** - Analyzing data

Synchronous Reading

In case of a "synchronous" access, the client waits until the server provided the data. Depending on the behavior defined in **"Opc.Da.Item"**, the data is either read from the internal "cache" of the OPC server or from the control "Device". The client does not accept **any further operating actions** during this period, i.e. the "synchronous" call is blocking.



Data is generally read more quickly from the "cache" than from the "Device". Always consider this behavior when designing an OPC Client.

"Synchronous" reading ("**Step: 11**") of subscription data is executed via the "opcValueResults = Me.m_OpcServer.Subscriptions(0).Read()" function call and the transfer of the items to be read "Me.m_OpcServer.Subscriptions(0).Items" as parameters.

In this example, the entire group is read and the result is forwarded to the "OnReadComplete" function for processing.

The "OnReadComplete" function is described in [chapter "Analyzing Data" on page 189](#).

Asynchronous Reading

In case of an "asynchronous" access, the task is transferred to the server and the function call returns immediately. The server processes the task and reports the result of the processed task to the client via a predefined "Delegate" ("OnReadComplete" event handler). Thus, the client can **accept and process further commands** via the server during task processing.

"Asynchronous" reading ("**Step: 12**") is performed as for "synchronous" reading, via the "opcValueResults = Me.m_OpcServer.Subscriptions(0).Read()" function call.

The call differs only in the number of parameters to be applied. The transfer parameters and their description are listed in the following table.

Transfer parameters	Description
Opc.Da.Item()	Item collection of all items to be read. As for synchronous reading.
Object	ID transferred from the client. This ID is given to the "Delegate" when processing the results (optionally "Nothing").
ReadCompleteEventHandler	"Delegate" to process the results of the request. The "Delegate" has to be transferred to the function call with the instruction "New ReadCompleteEventHandler(AddressOf OnReadComplete)".
IRequest	Handle to interrupt the request via the "Subscription.Cancel()" function call (optionally "Nothing").

Tab. 6-5: Transfer parameter for asynchronous reading

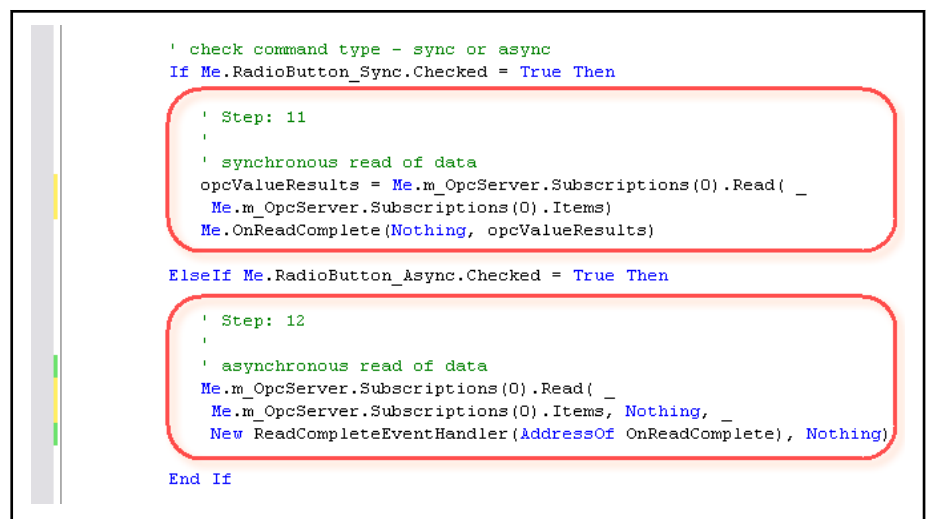


Fig. 6-18: Synchronous and asynchronous reading of control data

Analyzing Data

The "OnReadComplete" function to analyze data returned by the OPC server is called after the "synchronous" reading by the OPC Client and as an "event handler" ("Delegate") by the OPC server when providing the data after the "asynchronous" request.

As control elements are not "thread-safe", i.e. these objects can only be accessed by one thread. Verify whether the "OnReadComplete" function call is running in the same thread as form thread. This is verified in "Step: 13.1".

If the function is not called in the form thread, the call is synchronized by the "BeginInvoke()" function.

In "Step: 13.2" the results are analyzed and displayed in the list.

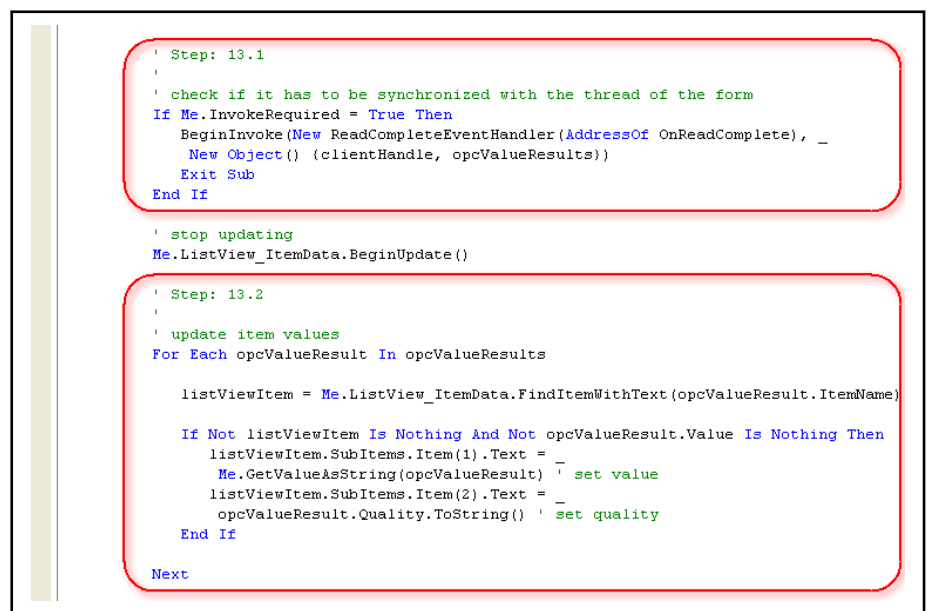


Fig. 6-19: Analyzing data - Reading

Client Development

Writing Control Data

General Information

This section describes the write access to control data. A distinction is made between

- "synchronous" and
- "asynchronous" access.

Control data is written in the "Button_Write_Click" event handler of the **"Write"** button.

The relevant code segments in the OPC Client example project are:

- **"Step: 14"** - Synchronous writing
- **"Step: 15"** - Asynchronous writing
- **"Step: 16"** - Analyzing data

Synchronous Writing

With a "synchronous" access, the client waits until the server has completed the writing task.

The data is written in the "cache" and in the "Device" of the OPC server if the subscription and the item are active. The client does not accept **any further operating actions** during this period, i.e. the "synchronous" call is blocking.

"Synchronous" writing (**"Step: 14"**) of subscription data is performed via the "opcResults = Me.m_OpcServer.Subscriptions(0).Write()" function call and the transfer of the "Opc.Da.ItemValue" ("opcValues") to be written as parameters.

In this example, only one item from the entire subscription is written and the write task result is forwarded to the "OnWriteComplete" function for processing.

The "OnWriteComplete" function is described in [chapter "Analyzing Data" on page 191](#).

Asynchronous Writing

In case of an "asynchronous" access, the task is transferred to the server and the function call returns immediately.

The server subsequently processes the task and reports the task processing result back to the client using a predefined "Delegate" ("OnWriteComplete" event handler). That means the client can **accept and process further operating commands** via the server during task processing.

The "asynchronous" writing (**"Step: 15"**) is executed via the function call as for "synchronous" writing

"opcResults = Me.m_OpcServer.Subscriptions(0).Write()".

The call differs only in the number of parameters to be applied. The transfer parameters and their description are listed in the following table.

Transfer parameters	Description
Opc.Da.ItemValue()	Item collection of all items to be written with values. As for synchronous writing.
Object	ID transferred from the client. This ID is given to the "Delegate" when processing the results (optionally "Nothing").

Transfer parameters	Description
WriteCompleteEventHandler	"Delegate" to process the task results. "Delegate" has to be transferred to the function call with the instruction "New WriteCompleteEventHandler(AddressOf OnWriteComplete)".
IRequest	Handle to interrupt the task via the "Subscription.Cancel()" function call (optionally "Nothing").

Tab. 6-6: Transfer parameters for asynchronous writing

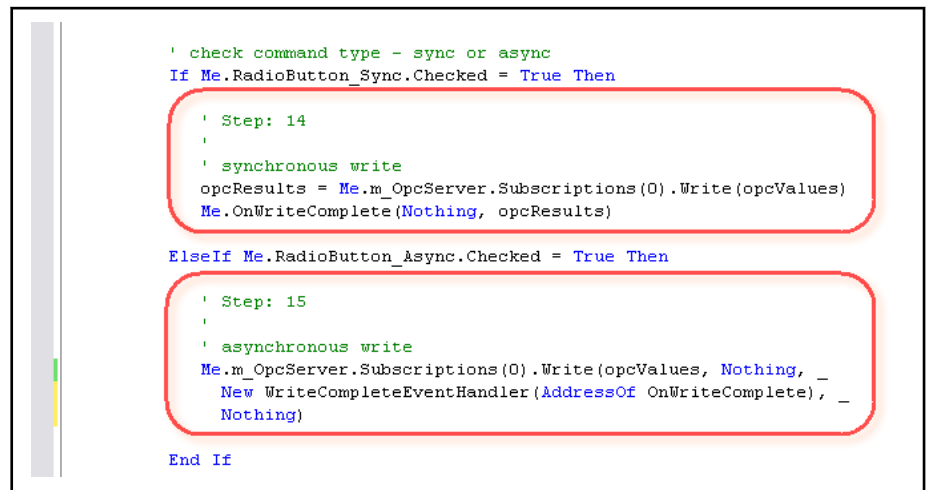


Fig. 6-20: Synchronous and asynchronous writing of control data

Analyzing Data

The "OnWriteComplete" function analyzes the results retrieved from the OPC server.

It is called by the OPC Client after "synchronous" writing and as "event handler" ("Delegate") by the OPC server when completing the task processing after the "asynchronous" command.

As control elements are not "thread-safe", i.e. these objects can only be accessed by one thread. Verify whether the "OnWriteComplete" function thread is running in the same thread as form thread. This is verified in **"Step: 16.1"**. If the function is not called in the form thread, the "BeginInvoke()" function synchronizes the call.

In **"Step: 16.2"** the results of the writing task can be analyzed. It is analyzed in a "log file" or e.g. in a "StatusStrip" control element of the application. The example client does not contain any analysis of the writing task result.

Client Development

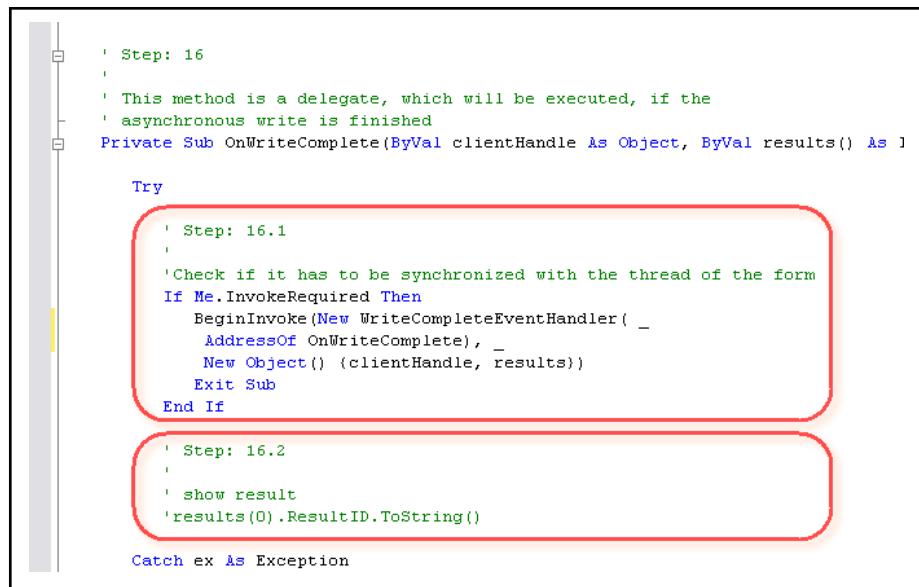


Fig. 6-21: Analyzing Data - Writing

Cyclic Reading of Control Data

General Information

This section describes cyclic reading of control data.

The prerequisite for cyclic reading of values is that

- the subscription and
- the item are active and that
- an event handler is logged on to the subscription.

Activation and deactivation of cyclic reading are controlled via the "RadioButton_CheckedChanged" of the "Radio Buttons" "Synchronous", "Asynchronous" and "Cyclic".

The relevant code segments in the OPC Client example project are:

- **"Step: 17"** - Activating cyclic reading
- **"Step: 18"** - Deactivating cyclic reading
- **"Step: 19"** - Analyzing data

Activating Cyclic Reading

As the prerequisites of the active subscription and active items are met, the "DataChanged" event handler only has to be added to the "iSubscription.DataChanged" subscription list (**"Step: 17"**). Adding the event handler meets all prerequisites and the client receives the "DataChanged" events.

Deactivating Cyclic Reading

To deactivate cyclic reading, the event handler

"DataChanged" is removed from the subscription list

"iSubscription.DataChanged" again. Once removed, no further value changes can be reported to the client.

```

If radioBtn.Text = Me.RadioButton_Cyclic.Text Then

    If Me.RadioButton_Cyclic.Checked = True Then

        ' Step: 17
        '
        ' add event handler for cyclic communication
        iSubscription = Me.m_OpcServer.Subscriptions(0)
        AddHandler iSubscription.DataChanged, AddressOf DataChanged

    Else

        ' Step: 18
        '
        ' remove event handler for cyclic communication
        iSubscription = Me.m_OpcServer.Subscriptions(0)
        RemoveHandler iSubscription.DataChanged, AddressOf DataChanged

    End If

End If
End If

```

Fig. 6-22: Adding and removing event handling

Analyzing Data

The "DataChanged" function is requested cyclically by the OPC server at the time intervals specified in the "UpdateRate" element of the "Opc.Da.SubscriptionState" class.

All items are transferred to the "DataChanged" function with the corresponding values if

- the value has changed or
- the maximum "age" of the value was exceeded.

This means, it depends on the configuration of the "MaxAge" and "MaxAgeSpecified" elements of the "Opc.Da.Item" class.

As control elements are not "thread-safe", i.e. these objects can only be accessed by one thread. Verify whether the "DataChanged" function thread is running in the same thread as form thread. This is verified in **"Step: 19.1"**.

If the function is not called in the form thread, the call is synchronized by the "BeginInvoke()" function.

In **"Step: 19.2"** the results are analyzed and displayed in the list view as with synchronous and asynchronous reading.

```

Try

    ' Step: 19.1
    '
    ' check if it has to be synchronized with the form thread
    If (Me.InvokeRequired = True) Then
        Me.BeginInvoke(New DataChangedEventHandler(AddressOf DataChanged), _
            New Object() {subscriptionHandle, requestHandle, opcValueResults})
        Exit Sub
    End If

    ' stop updating
    Me.ListView_ItemData.BeginUpdate()

    ' Step: 19.2
    '
    ' update item values
    Me.OnReadComplete(Nothing, opcValueResults)

```

Fig. 6-23: Evaluating data - Cyclic reading

Error Handling

Only a minimum of error handling was implemented with the example OPC Client. Analyze and record errors occurring with an OPC Client used on production machines more in detail.

The OPC Client is therefore restricted to verify variables and the simple exception handling as

- "If" queries and
- exception handling with "Try...Catch..." instructions

"If" queries "If" queries verify the parameters required to execute the function. If the specified definitions cannot be fulfilled, the function is canceled.

Exception handling with "Try...Catch..." instructions As the majority of components report an error to the caller during execution of a function by "throwing" an exception, at least one "Try...Catch..." statement should be included in every function/method. In the example OPC Client, only one "Try...Catch..." instruction was programmed due to the scope of the individual functions.

6.2 OPC Client Visual C# .NET

6.2.1 General Information

This section deals with the development of the Visual C# .NET OPC Client using "Microsoft Visual Studio .NET 2005" and covers the following topics:

This section deals with the following topics:

- Creating projects
- Adding References
- Adding forms and control elements
- OPC Client implementation.

The following diagram shows the OPC Client as active Windows application.

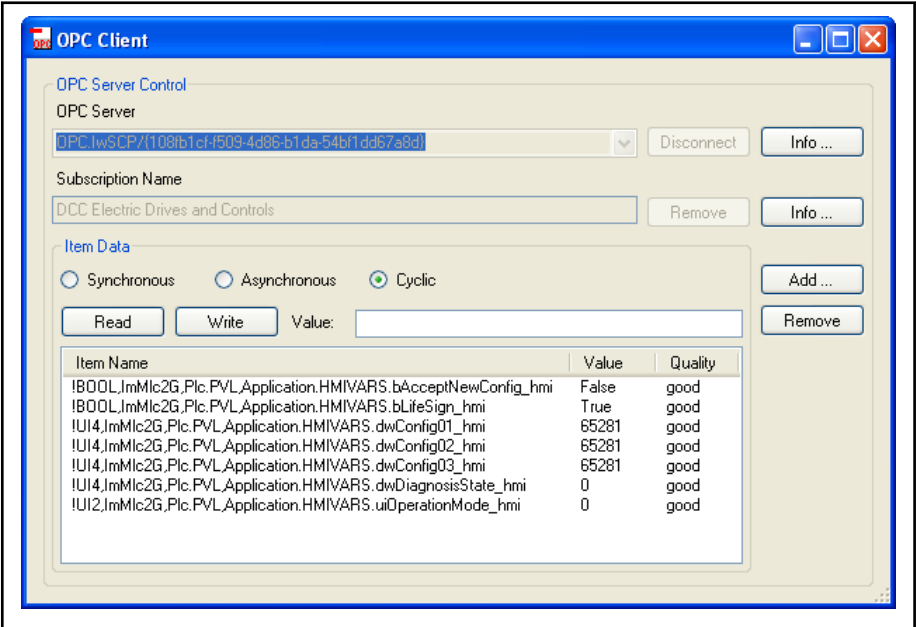


Fig. 6-24: OPC Client interface

6.2.2 Visual C# Project "OPC Client "

New Project

To create a new project in the "Microsoft Visual Studio .NET 2005" development environment, click on the link "New Project" on the starting page or on **File ► New and Project....**

The "New Project" dialog is displayed as shown below.

Select project type and template:

- "Other languages \ Visual C# \ Windows" or
- "Visual C# \ Windows" and
- "Windows Application as template"

Enter "OPCClient" as project name and adapt the suggested storage location.

Create the with **OK**.

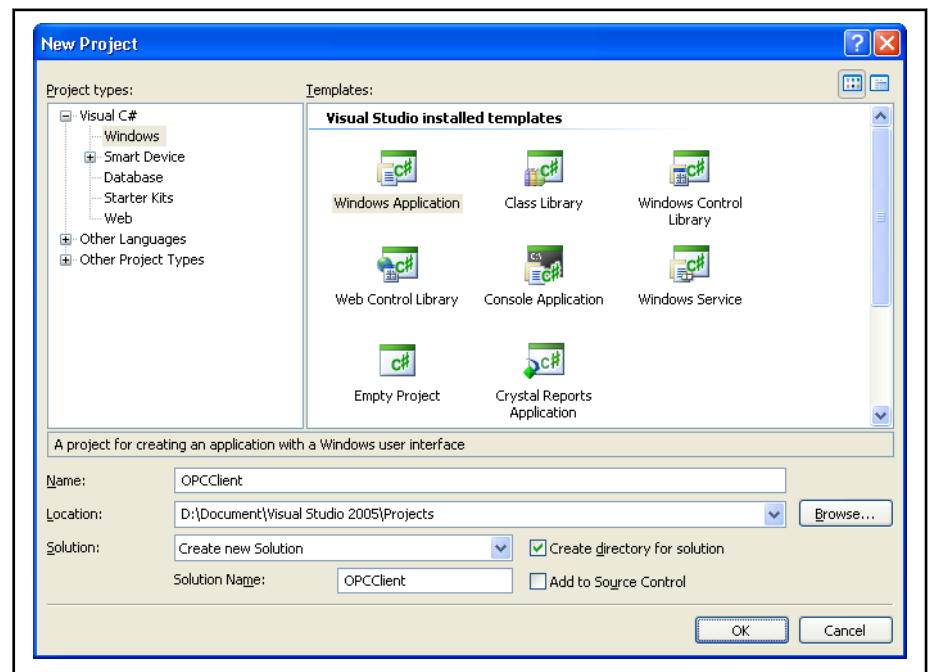


Fig. 6-25: Create new project

Adding References

The project has to contain a reference to "OpcNetApi.dll" and "OpcNetApi.Com.dll" to allow access to the OPC .NET API functionalities.

The "project references" are managed in the project folder explorer on the "References" node of the "OPCClient" project or using the references already added.

Add references via the menu item **Add References ...** of the corresponding context menu or via the **Project** menu. In the "Add Reference" dialog, the two files "OpcNetApi.dll" and "OpcNetApi.Com.dll" can be added to the project on the "Browse" tab.

Both files are provided with the "IndraWorks Suite" installation and saved in the installation folder (e.g. "C:\Program Files\Rexroth\IndraWorks").

Open the files to apply them to the node list "References:" .

Client Development

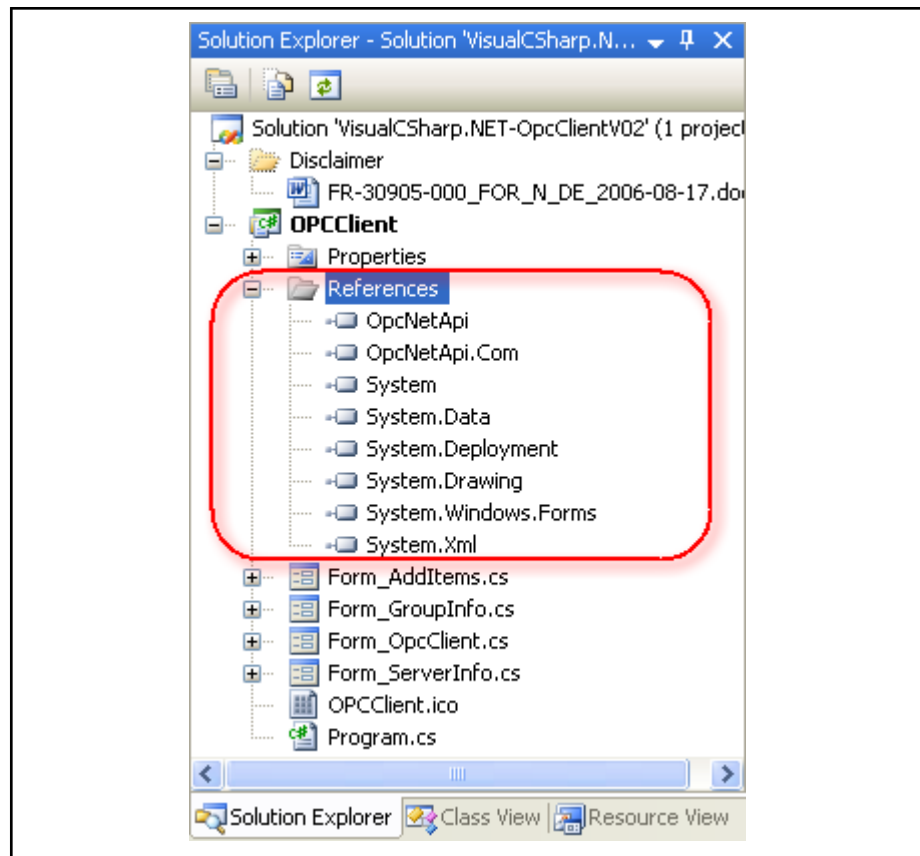


Fig. 6-26: Project references

Adding Windows Form and Control Elements

Add a Windows to the starting project via **Project ► Add Windows Form....**

In the "Add New Element - OPCClient" dialog, the "Windows Form" template is already selected and only the form name has to be entered.

All Windows forms created in the OPC Client example have the identifier name "Form_" as prefix.

The following figure shows an overview on the "OPC Client" project in the "Solution Explorer".

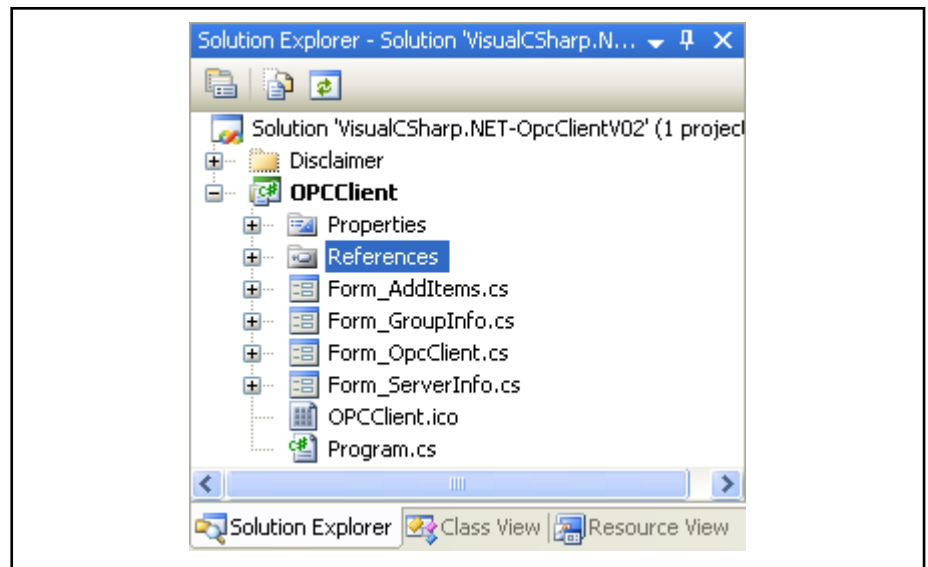


Fig. 6-27: Project overview

"Windows forms" are designed with the "Windows Forms Designer" and the control elements in the "Toolbox". Select the control elements and place them on the form.

The control elements for the selected object are configured in the "Properties" window.

Size and position of the control elements can also be changed using the mouse in the Windows Forms Designer.

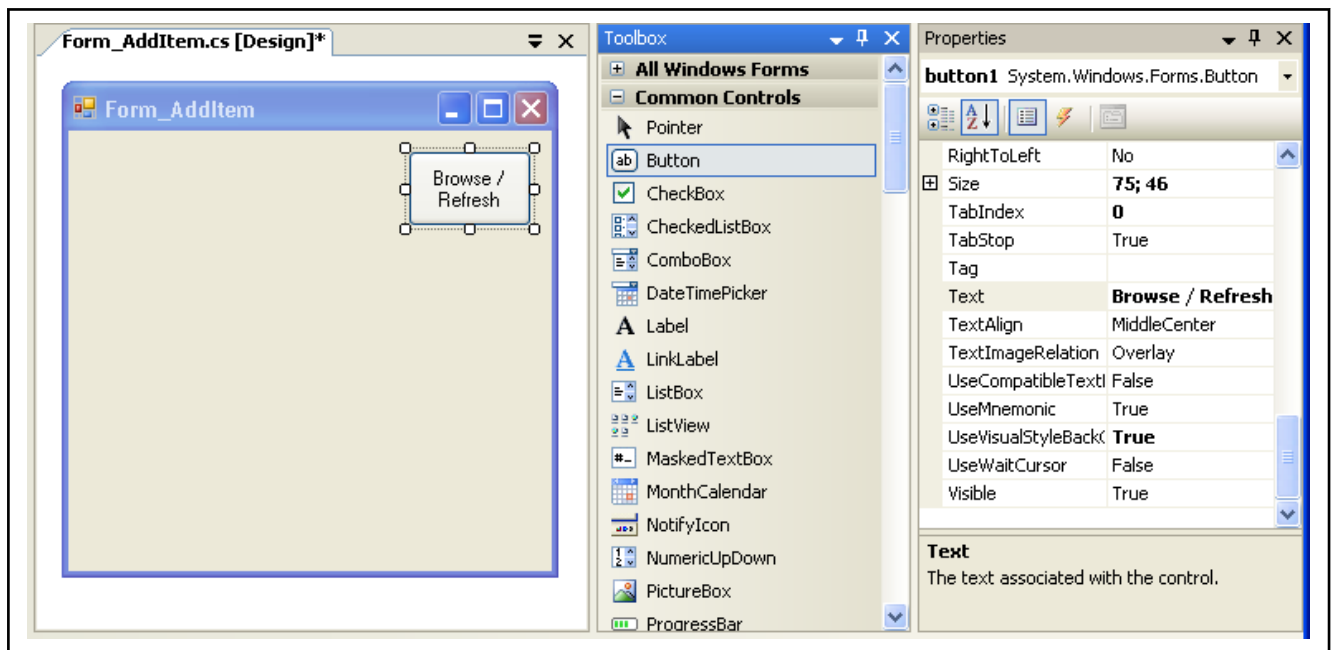


Fig. 6-28: Adding and configuring control elements

The following figure shows the "OPC Client" form in the "Windows Forms Designer".

To maintain clarity, a detailed description of the form and the individual control elements is not provided. All configurable properties can be taken from the example project.

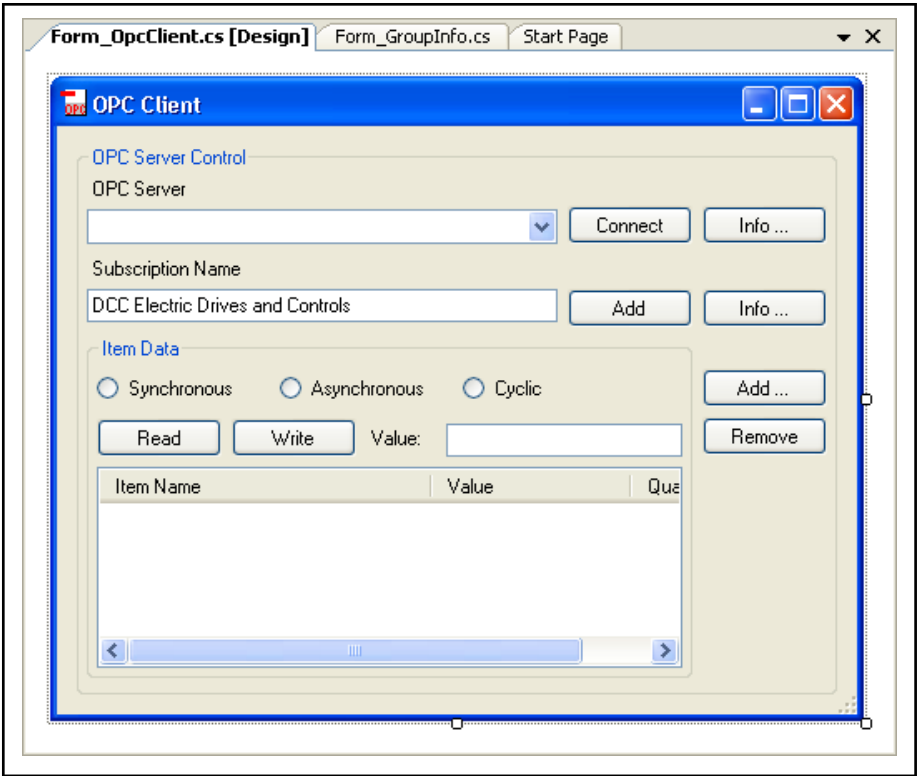


Fig. 6-29: "OPC Clients" Windows Forms Designer

Implementing OPC Client

General Information

The following section describes the procedure when programming the OPC Client.

For a better program overview, individual regions of the class are summarized with the "#region" directive. These "regions" can be expanded or collapsed when creating the program.

The figure below shows the defined areas using the example of the "Form_OpcClient" form.

Area	Description
"#region" -"#endregion"	
"Declaration"	Contains class declaration (member variables and constants).
"Form Events"	Procedures for handling the events triggered by the form and the included control elements.
"OPC Events"	Procedures for handling the events triggered by the OPC server.
"Additional Internal Functions"	Additional internal definitions, structures and functions to optimize user guidance.

Tab. 6-7: Program structuring using "#region" directives

The relevant program lines in the example project are also identified by the keyword "Step:" and numbered consecutively. This means that using the search "Search in Files" (<Ctrl>+<Shift>+<F>) in Visual Studio allows direct branching to the implementation described by entering the search word e.g. "Step: 2".

The implementation described in the following was documented as the operating order of the "OPC Client" application.

The structure is shown in the table below.

Chapter heading	Description
chapter "Declaring Variables" on page 199	Declaring variables to manage OPC server and Windows form.
chapter "Listing OPC Server" on page 200	Listing of all OPC servers installed on the local PC.
chapter "Creating and Deleting OPC Server Instance" on page 200	Creating and canceling the OPC server instance and displaying the OPC server information.
chapter "Creating and Canceling OPC Server Subscription" on page 203	Creating and canceling an OPC subscription and displaying the subscription information.
chapter "Listing Control Items" on page 205	Reading and displaying all control items (browsing).
chapter "Adding and Removing OPC Items" on page 206	Adding and deleting OPC items.
chapter "Reading Control Data" on page 209	Reading "synchronous" and "asynchronous" control data.
chapter "Writing Control Data" on page 211	Writing "synchronous" and "asynchronous" control data.
chapter "Cyclic Reading of Control Data" on page 213	Cyclic control data reading.

Tab. 6-8: Overview on the implementation structure

Declaring Variables

In the **"Step: 1"** declaration section, the "Form_OpcClient" variables are declared.

The syntax chosen for the names ensures that all class variables can be written with the prefix "m_" ("member variables") and that the constant definitions are "capitalized".

The variables are

- the object for the OPC server instance and
- variables to manage the form.

The line **"private Opc.Da.Server m_OpcServer = null;"** is the declaration relevant to the example. The "m_OpcServer" variable contains the created OPC server object.

Client Development

```

// Step: 1
//
// declaration of members used for opc administration
#region " Declaration "

// Step: 1
//
// declaration of members used for opc server administration
private Opc.Da.Server m_OpcServer = null;

//
// members to store configuration
private OPCClient.Form_AddItems m_DialogAddItems = null;
private InternalStateControl m_InternalStateControl;

//
// constant values
private readonly String BUTTON_DESCRIPTION_SERVER_CONNECT = "Connect";
private readonly String BUTTON_DESCRIPTION_SERVER_DISCONNECT = "Disconnect";
private readonly String BUTTON_DESCRIPTION_GROUP_ADD = "Add";
private readonly String BUTTON_DESCRIPTION_GROUP_REMOVE = "Remove";
private readonly Int32 MAX_ITEMS = 10;

#endregion

```

Fig. 6-30: Declaring class variables

Listing OPC Server

All OPC servers installed locally on the PC are listed in the "Load Event" "Step: 2" of the "Form_OpcClient" form.

All OPC servers are requested via the "opcDiscovery.GetAvailableServers()" method of the Discovery interface "Opc.IDiscovery" of the "OpcEnum.exe" service supporting the specification "Data Access 2.0". The available OPC specifications are defined in the "Opc.Specification" structure.

The list provided by the "Discovery Interface" includes the complete URL path with "ProgID" (without version index) and "GUID" to clearly identify the server.

```

// Step: 2
//
// browse for available opc servers on this system
private void Form_OpcClient_Load ( object sender, EventArgs e )
{
    try
    {
        // Step: 2
        //
        // browse for available opc servers on this system
        Opc.IDiscovery opcDiscovery = new OpcCom.ServerEnumerator ();
        Opc.Server[] opcServers = opcDiscovery.GetAvailableServers (
            Opc.Specification.COM_DA_20);

        // copy opc server to ...
        this.ComboBox_Server.BeginUpdate ();

        foreach ( Opc.Server opcServer in opcServers )
        {
            this.ComboBox_Server.Items.Add ( opcServer.Url.Path );
        }
    }
    catch ( Exception except )
    {
    }
}

```

Fig. 6-31: Listing installed OPC servers

Creating and Deleting OPC Server Instance**General Information**

This section describes how to create and cancel the OPC server instance and how to read the server information. The OPC Server instance is managed in the event handler "Button_Connect_Disconnect_Click" either of the

button **"Connect"** or **"Disconnect"**. Server information is ready and displayed via the function `"Button_ServerInfor_Click"` of the **"Info ..."** button.

The relevant code segments in the OPC Client example project are:

- **"Step: 3"** - Create OPC server instance
- **"Step: 4"** - Delete OPC server instance
- **"Step: 5"** and **"Step: 21"** - Read server information

Creating OPC Server Instance - Connect

Creating an OPC server instance and establishing the connection is described in **"Step: 3"**.

The first step creates an OPC server instance `"New Opc.Da.Server"` and transfer of the parameters `"OpcCom.Factory"` and the `"Opc.URL"`.

The `"Opc.URL"` includes information on the OPC specification `"Opc.UrlScheme.DA"` used, the PC name `"localhost"` or the IP address the server is started on as well as the server ID. In the example project, the server ID is retrieved from the combination list field `"ComboBox_Server"`.

The following options are possible to specify the server ID:

- **"OPC.lwSCP"**
ProgID without version index
- **"OPC.lwSCP.1"**
ProgID with version index
- **"OPC.lwSCP/{108fb1cf-f509-4d86-b1da-54bf1dd67a8d}"**
ProgID without version index and GUID
- **"OPC.lwSCP.1/{108fb1cf-f509-4d86-b1da-54bf1dd67a8d}"**
ProgID with version index and GUID

The OPC server is instanced by calling the function `"this.m_OpcServer.Connect()"`.

Deleting OPC Server Instance - Disconnect

The deletion of the OPC server instance is shown in **"Step: 4"**.

Calling the function `"this.m_OpcServer.Disconnect()"` causes logout from the OPC server. If no further client is connected to the server at this point, it is closed.

Subsequently, the function `"this.m_OpcServer.Dispose()"` is called to release resources held in the OPC server instance and the instance is deleted by `"this.m_OpcServer = null"`.

Client Development

```

try
{
    // show hour glas
    this.Cursor = Cursors.WaitCursor;

    if ( this.Button_Connect_Disconnect.Text.Equals (BUTTON_DESCRIPTION
    {
        // Step: 3
        //
        // select opc server and
        this.m_OpcServer = new Opc.Da.Server ( new OpcCom.Factory(),
            new Opc.URL(Opc.UriScheme.DA + "://localhost/" +
                this.ComboBox_Server.Text) );

        // connect with ...
        this.m_OpcServer.Connect();

        // Enable or disable controls
        this.ComboBox_Server.Focus();
        this.SetInternalStateControl(InternalStateControl.Connected);
    }
    else if ( this.Button_Connect_Disconnect.Text.Equals (BUTTON_DESCRI
    {
        // Step: 4
        //
        // disconnect from server and clean up
        this.m_OpcServer.Disconnect();
        this.m_OpcServer.Dispose();
        this.m_OpcServer = null;

        // Enable or disable controls
        this.SetInternalStateControl(InternalStateControl.Disconnected);

        // clear browse info
        if ( this.m_DialogAddItems is Form_AddItems )
            this.m_DialogAddItems.Clear();
    }
    }
}

```

Fig. 6-32: Creating and canceling OPC server instance

Reading Server Information

The information provided by the OPC Server is read and displayed in the "Load Event" **"Step: 21"** of the form "Form_ServerInfo".

The form is called in **"Step: 5"** (without figure).

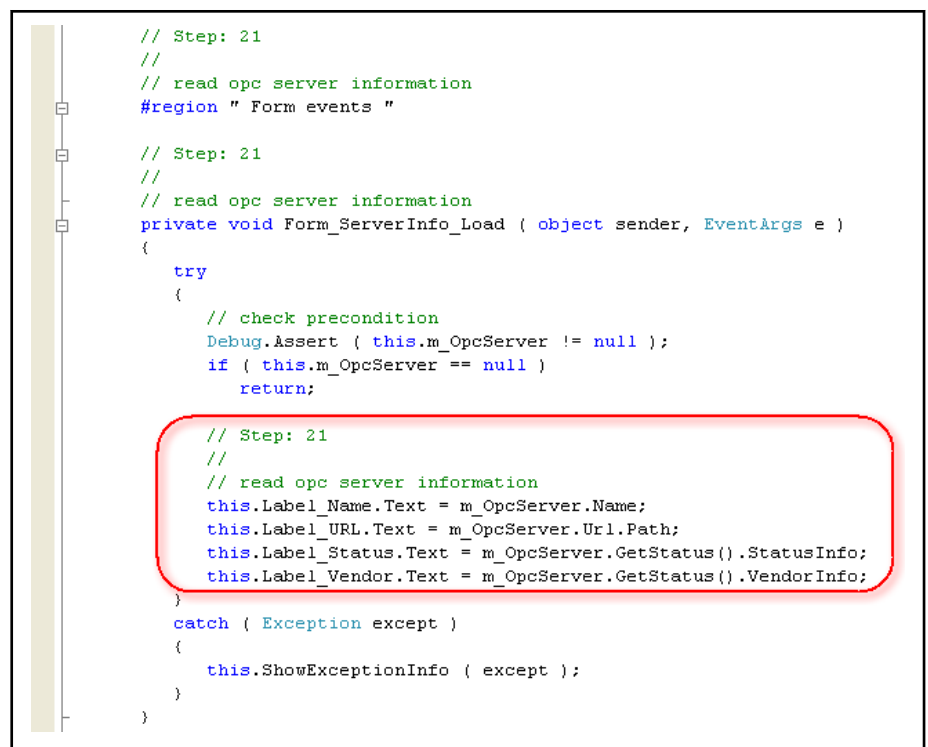


Fig. 6-33: Reading OPC server information

Creating and Canceling OPC Server Subscription

General Information

This section describes login and logout of a "subscription" (group) on the OPC server. The subscription is managed in the "Button_Add_Remove_Group_Click" event handler of the "Add" or "Remove" button and the subscription information is read and displayed in the "Button_GroupInfo_Click" function of the "Info ..." button.

The relevant code segments in the OPC Client example project are:

- "Step: 6" - Create subscription
- "Step: 7" - Cancel subscription
- "Step: 8" and "Step: 22" - Display subscription information

Creating Subscription - CreateSubscription

A subscription is created in the OPC server collection using the function "this.m_OpcServer.CreateSubscription()" and the transfer of the parameter "Opc.Da.SubscriptionState" ("Step: 6", see [fig. 6-34 "Creating OPC subscription" on page 204](#)).

The elements of the "Opc.Da.SubscriptionState" class clearly identify the subscription and configure with regard to behavior or processing by the OPC server.

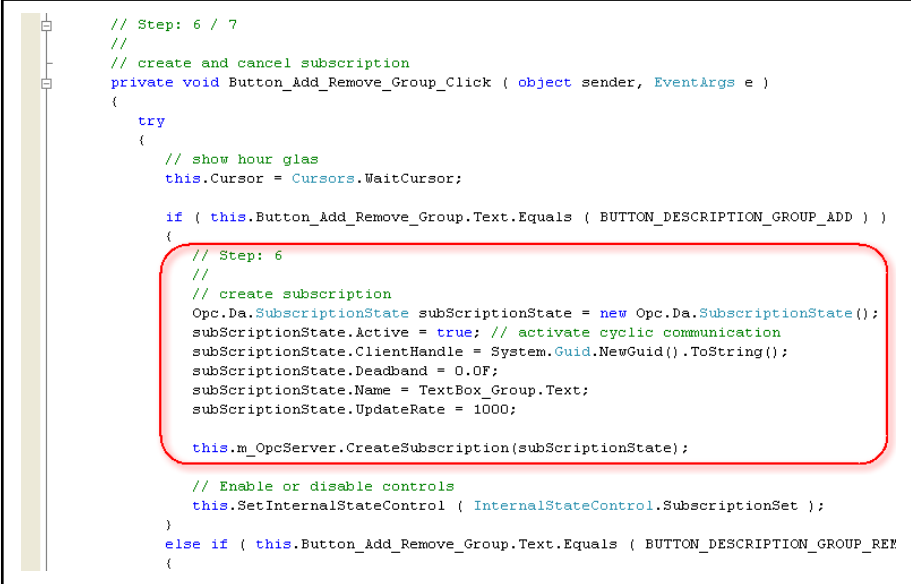
The following table describes the class elements.

Class element	Description
Active	The control cyclically reads the subscription items and saves them in the server cache.
ClientHandle	"Unique Identifier" (specified by the client).

Client Development

Class element	Description
Deadband	Smallest possible change in percentage of a value to trigger an event to the client.
KeepAlive	Largest possible time interval between two events to the client (life sign).
Local	Definition of localization in the format "[language Code]-[country/region code]", e.g. "de-DE", "en-US"
Name	"Unique Name" (specified by client). Specified by the "TextBox_Group.Text" text box.
ServerHandle	"Unique Identifier" (specified by server).
UpdateRate	Time interval the server checks data and sends an event to the client.

Tab. 6-9: Elements of the "Opc.Da.SubscriptionState" class



```

// Step: 6 / 7
//
// create and cancel subscription
private void Button_Add_Remove_Group_Click ( object sender, EventArgs e )
{
    try
    {
        // show hour glas
        this.Cursor = Cursors.WaitCursor;

        if ( this.Button_Add_Remove_Group.Text.Equals ( BUTTON_DESCRIPTION_GROUP_ADD ) )
        {
            // Step: 6
            //
            // create subscription
            Opc.Da.SubscriptionState subSubscriptionState = new Opc.Da.SubscriptionState();
            subSubscriptionState.Active = true; // activate cyclic communication
            subSubscriptionState.ClientHandle = System.Guid.NewGuid().ToString();
            subSubscriptionState.Deadband = 0.0F;
            subSubscriptionState.Name = TextBox_Group.Text;
            subSubscriptionState.UpdateRate = 1000;

            this.m_OpcServer.CreateSubscription(subSubscriptionState);

            // Enable or disable controls
            this.SetInternalStateControl ( InternalStateControl.SubscriptionSet );
        }
        else if ( this.Button_Add_Remove_Group.Text.Equals ( BUTTON_DESCRIPTION_GROUP_REP
    {

```

Fig. 6-34: Creating OPC subscription

Canceling Subscription - CancelSubscription

Cancel a subscription calling the function

"this.m_OpcServer.CancelSubscription()" with transfer of the parameter "Opc.Da.ISubscription" ("Step: 7").

Once the subscription has been canceled from the OPC server collection, call the "iSubscription.Dispose()" function. This enabled the resources used by the subscription. The reference of the interface variables is canceled by assigning "iSubscription = null".

The local allocation "iSubscription = this.m_OpcServer.Subscription()" performed at the start of the code segment is required, as the subscription was removed from the OPC server collection after calling the function "this.m_OpcServer.CancelSubscription()" and it is no longer available.

```

    }
    else if ( this.Button_Add_Remove_Group.Text.Equals ( BUTTON_DESCRIPTION_GROU
    {
        // Step: 7
        //
        // cancel subscription and clean up
        Opc.Da.ISubscription iSubscription = this.m_OpcServer.Subscriptions[0];
        this.m_OpcServer.CancelSubscription(iSubscription);
        iSubscription.Dispose();
        iSubscription = null;

        // Enable or disable controls
        this.SetInternalStateControl ( InternalStateControl.Connected );
    }
}
catch ( Exception except )

```

Fig. 6-35: Canceling OPC subscription

Displaying Subscription Information

The information provided by the subscription is read and displayed in the "Load Event" ("Step: 22") in the form "Form_GroupInfo".

The form is called in "Step: 8" (without figure).

```

// check precondition
Debug.Assert ( this.m_OpcServer != null );
if ( this.m_OpcServer == null )
    return;

// Step: 22
//
// read opc subscription information
Opc.Da.Subscription subSubscription = m_OpcServer.Subscriptions[0];
this.Label_Name.Text = subSubscription.Name;
this.Label_Deadband.Text = subSubscription.GetState ( ).Deadband.ToString ( ) + " %";
this.Label_UpdateRate.Text = subSubscription.GetState ( ).UpdateRate.ToString ( ) + " ms";
}
catch ( Exception except )

```

Fig. 6-36: Reading OPC subscription information

Listing Control Items

General Information

This section describes the reading of all items configured in the control for the access via OPC server.

The relevant code segments in the OPC Client example project are:

- "Step: 20", "Step: 20.1" and "Step: 20.2" - Reading control items

Reading control items

Click on the "Add ..." button on the "Form_OpcClient" form to open the "Form_AddItems" form. As it can take some time to read the items, the items available on the OPC server are not read in the "Load Event" of the form, but by using the "Button_BrowseItems_Click" function. Click on the "Browse / Refresh" button to activate this function.

The two classes "Opc.ItemIdentifier" and "Opc.Da.BrowseFilters" of the "this.m_OPCServer.Browse()" function are transferred as parameters to read the items ("Step: 20.2").

The classes are provided with the following function:

Opc.ItemIdentifier

The "Opc.ItemIdentifier" class specifies the "starting point" of the search in the "tree hierarchy" of the items. This means that if, for example, a search is canceled, it can be continued at this point.

Opc.Da.BrowseFilter

The class "Opc.Da.BrowseFilter" is used as filter criterion for the function call result. There is an option to only read variables beginning with a certain sequence of characters for example.

As there is no default specification for the format in which an OPC server returns the "browsing" information, reading the control items was divided into

Client Development

two functions ("Step: 20.1" and "Step: 20.2"). This division allows reading flat hierarchies (tree structure without sub-nodes) and structured hierarchies (tree structure with sub-nodes) (see following figures).

```
// clear list, stop updating
this.ListBox_Available_Items.BeginUpdate ( );
this.ListBox_Available_Items.Items.Clear ( );

// Step: 20.1
//
// browse items
Opc.Da.BrowseFilters filters = new Opc.Da.BrowseFilters ( );
filters.ElementNameFilter = "";
filters.BrowseFilter = Opc.Da.browseFilter.all;
filters.ReturnAllProperties = false;
filters.ReturnPropertyValues = false;

Opc.ItemIdentifier itemID = new Opc.ItemIdentifier ( );
itemID.ItemName = "";

this.BrowseItems ( itemID, filters );
}
catch ( Exception except )
```

Fig. 6-37: Reading control items - Filter configuration

```
// Step: 20.2
//
// browse items
opcBrowseElements = this.m_OpcServer.Browse ( itemID, filters, out opcBrowsePosition );

// adds each element to the list
foreach (Opc.Da.BrowseElement opcBrowseElement in opcBrowseElements)
{
    //
    // avoid errors while using .NET API 1.30
    if ( opcBrowseElement.ItemName != null )
        itemName = opcBrowseElement.ItemName;
    else
        itemName = opcBrowseElement.Name;

    this.ListBox_Available_Items.Items.Add ( itemName );

    // check on subordinate items
    if (opcBrowseElement.HasChildren == true)
    {
        // set branch to search
        opcItemID.ItemName = itemName; // opcBrowseElement.ItemName;
        this.BrowseItems ( opcItemID, filters );
    }
}
```

Fig. 6-38: Reading control items - Recursion

Adding and Removing OPC Items

General Information

This section describes the addition and deletion of items to/from a subscription.

The relevant code segments in the OPC Client example project are:

- "Step: 9", "Step: 9.1" and "Step: 9.2" - Adding items
- "Step: 10" - Deleting items

Adding items - AddItems

In the form event "Button_Add_Item_Click" ("Step: 9") the form "Form_AddItems" is called and the items selected in the form are added.

The variables can be entered separately in the text field of the "Add Items" form by ";" or they can be selected from the list. When entering the variables, the item syntax of the respective OPC server is to be taken into account.

The following shows the item syntax for the "OPC.lwSCP" server and the "IndraLogic OPC" server.

"OPC.lwSCP" server

- **PLC variables**

"![required data type],[control name or alias],[lwSCP service],[fully qualifying variable name]"

Example:

"!Bool,ImMlc2G,Plc.PVL,Application.HMIVARS.bLifeSign_hmi"

- **sercos parameter**

"![required data type],[control name or alias],[lwSCP service],[drive address], [parameter number], [element coding]"

Example:

"!UI4,ImMlc2G,SPA1,0,C-0-0001,D"

"!UI4,ImMlc2G,SPA1,0,C-0-0001,40"



The "OPC.lwSCP" server allows to access parameters and PLC variables!



For further information on the item syntax of the "OPC.lwSCP" server, refer to [chapter 4.3.3 "lwSCP and OPC.lwSCP Items" on page 93](#).

"IndraLogic OPC" server

- **PLC variables**

"[Control name].[fully qualifying variable name]"

Example:

"PLC_IL2G.Application.HMIVARS.bLifeSign_hmi"



The "IndraLogic OPC" server allows to access PLC variables.

For the description of the "Browse" functionality, refer to [chapter "Listing Control Items" on page 205](#).

Once the items are entered or selected in the "Add Items" form, in **"Step: 9.1"** an "Opc.Da.Item" class is created and initialized for every item. The item name is assigned to the line "opcltems[index].ItemName = items[index]".

The class elements of class "Opc.Da.Item" used in the example are shown in the following table:

Class element	Description
Active	"DataChanged" events for the item are sent to the client.
ActiveSpecified	The "Active" property was defined.
ItemName	Fully qualified item name. The syntax depends on the OPC server.
ItemPath	Fully qualified item path. This property is not analyzed by the "OPC.lwSCP".

Client Development

Class element	Description
MaxAge	When retrieving the item from the OPC server cache, the value may not be older than defined in "MaxAge".
MaxAgeSpecified	The "MaxAge" property was defined. If this property is set to "True", the value is retrieved from the OPC server "cache" or the "Device" depending on "MaxAge". If the property is set to "False", the value is always read from the "Device".

Tab. 6-10: Elements of class "Opc.Da.Item"

After all OPC items have been created, the OPC item array is logged in to the subscription on the OPC server with the function call "this.m_OpcServer.Subscriptions[0].AddItems()" ("Step: 9.2").

```
// Step: 9
//
// add items to the subscription
Opc.Da.Item[] opcItems = new Opc.Da.Item[items.Length];
for (Int32 index = 0; index < items.Length; index++)
{
    // Step: 9.1
    //
    // creating new item
    opcItems[index] = new Opc.Da.Item();
    // set item properties
    opcItems[index].ClientHandle = index;
    opcItems[index].Active = true;
    opcItems[index].ActiveSpecified = true;
    opcItems[index].ItemName = items[index];
    opcItems[index].ItemPath = "";
    opcItems[index].MaxAge = -1;
    //opcItems[index].MaxAgeSpecified = false; // read from device
    opcItems[index].MaxAgeSpecified = true; // read from cache
}

// Step: 9.2
//
// add the item to the subscription and to the list view
ListViewItem listViewItem;
Opc.Da.ItemResult[] itemResults = this.m_OpcServer.Subscriptions[0].AddItems(opcItems);
for (Int32 index = 0; index < itemResults.Length; index++)
{
    if (itemResults[index].ResultID.Succeeded() == true)
    {
        listViewItem = this.ListView_ItemData.Items.Add(opcItems[index].ItemName);
        listViewItem.SubItems.Add ( "" ); // sub item 'Value'
        listViewItem.SubItems.Add ( "" ); // sub item 'Quality'
    }
}
```

Fig. 6-39: Adding items to a subscription

Removing Items - RemoveItems

To remove items from the OPC subscription, select the items from the list and click on "Remove".

In the subsequently triggered "Button_Remove_Item_Click" form event, the function "this.m_OpcServer.Subscriptions[0].RemoveItems()" is called removing the OPC item transferred as parameter ("Step: 10").

```

// Step: 10
//
// remove selected items from opc subscription and from the list
for ( Int32 index = 0; index < selectedItems; index++ )
{
    foreach (Opc.Da.Item opcItem in this.m_OpcServer.Subscriptions[0].Items)
    {
        if ( this.ListView_ItemData.SelectedItems[index].Text.Equals ( opcItem.ItemName ) )
        {
            opcItemIds[index] = opcItem;
            break;
        }
    }
}
// remove items
Opc.IdentifiedResult[] opcIdentResult =
this.m_OpcServer.Subscriptions[0].RemoveItems ( opcItemIds );

// clean up list view control
for ( Int32 index = 0; index < opcIdentResult.Length; index++ )
{
    if ( opcIdentResult[index].ResultID == ResultID.S_OK )
    {
        foreach ( ListViewItem listViewItem in this.ListView_ItemData.SelectedItems )
        {
            if ( opcIdentResult[index].ItemName.Equals ( listViewItem.Text ) )
            {
                this.ListView_ItemData.Items.Remove ( listViewItem );
                break;
            }
        }
    }
}

```

Fig. 6-40: Removing items from a subscription



When removing items from a group, remove the complete list of items. Thus, the administrative tasks for login and logout of the group only have to be executed once in the OPC server.

Reading Control Data

General Information

This section describes read access to control data.

A distinction is made between

- "synchronous" and
- "asynchronous" access.

The control data is read in the "Button_Read_Click" event handler of the "Read" button.

The relevant code segments are:

- "Step: 11" - Synchronous reading
- "Step: 12" - Asynchronous reading
- "Step: 13" - Analyzing data

Synchronous reading

In case of a "synchronous" access, the client waits until the server provided the data. Depending on the behavior defined in "Opc.Da.Item", the data is either read from the internal "cache" of the OPC server or from the control "Device". The client does not accept **any further operating actions** during this period, i.e. the "synchronous" call is blocking.



The data is generally read more quickly from the "cache" than from the "Device". Always consider this behavior when designing an OPC Client.

"Synchronous" reading ("Step: 11") of subscription data is performed via the "opcValueResults = Me.m_OpcServer.Subscriptions[0].Read()" function call and the transfer of the items to be read "Me.m_OpcServer.Subscriptions[0].Items" as parameters.

Client Development

In this example, the entire group is read and the result is forwarded to the "OnReadComplete" function for processing.

The "OnReadComplete" function is described in [chapter "Analyzing data" on page 210](#).

Asynchronous reading

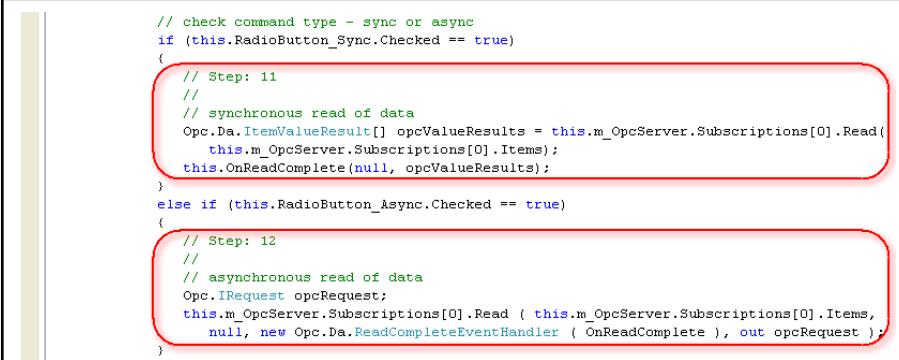
In case of an "asynchronous" access, the task is transferred to the server and the function call returns immediately. The server processes the task and reports the task processing event back to the client via a predefined "Delegate" ("OnReadComplete" event handler). This means that the client can **accept and process further actions** via the server during task processing.

"Asynchronous" reading ("**Step: 12**") is performed as for "synchronous" reading via the "opcValueResults = Me.m_OpcServer.Subscriptions[0].Read()" function call.

The call differs only in the number of parameters to be applied. The transfer parameters and their description are listed in the following table.

Transfer parameters	Description
Opc.Da.Item[]	Item collection of all items to be read. As for synchronous reading.
Object	ID transferred from the client even to the "Delegate" when processing the results (optionally "null").
ReadCompleteEventHandler	"Delegate" to process the results of the request. "Delegate" is to be transferred to the function call using the instruction "new ReadCompleteEventHandler(OnReadComplete)".
IRequest	Handle to interrupt the request via the "Subscription.Cancel()" function call (optionally "null").

Tab. 6-11: Transfer parameter for asynchronous reading



```

// check command type - sync or async
if (this.RadioButton_Sync.Checked == true)
{
    // Step: 11
    //
    // synchronous read of data
    Opc.Da.ItemValueResult[] opcValueResults = this.m_OpcServer.Subscriptions[0].Read(
        this.m_OpcServer.Subscriptions[0].Items);
    this.OnReadComplete(null, opcValueResults);
}
else if (this.RadioButton_Async.Checked == true)
{
    // Step: 12
    //
    // asynchronous read of data
    Opc.IRequest opcRequest;
    this.m_OpcServer.Subscriptions[0].Read ( this.m_OpcServer.Subscriptions[0].Items,
        null, new Opc.Da.ReadCompleteEventHandler ( OnReadComplete ), out opcRequest );
}

```

Fig. 6-41: Synchronous and asynchronous reading of control data

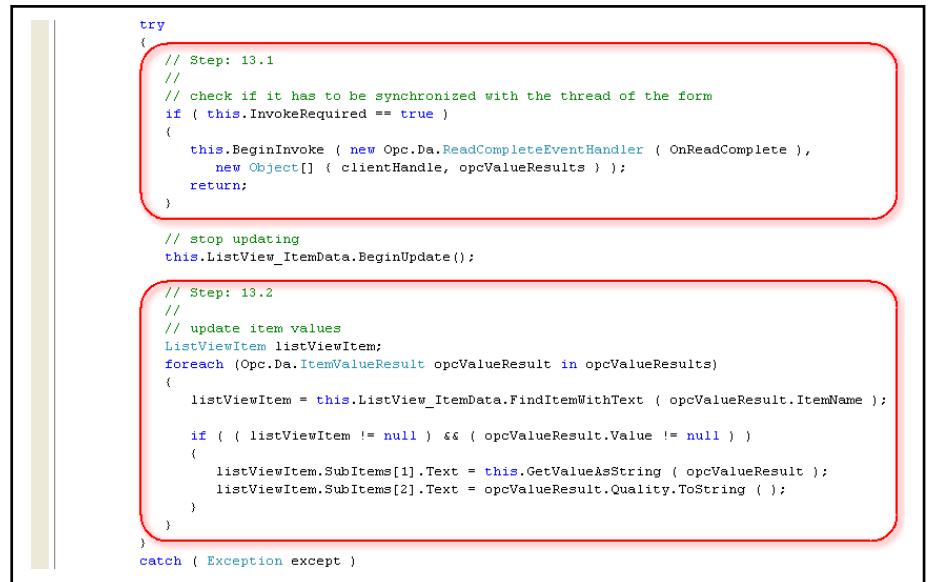
Analyzing data

The "OnReadComplete" function to analyze data returned by the OPC server is called after the "synchronous" reading by the OPC Client and as an "event handler" ("Delegate") by the OPC server when providing the data after the "asynchronous" request.

As control elements are not "thread-safe", i.e. these objects can only be accessed by one thread. Verify whether the "OnReadComplete" function call is

running in the same thread as form thread. This is verified in **"Step: 13.1"**. If the function was not called in the form thread, the request is synchronized by the `"this.BeginInvoke()"` function.

In **"Step: 13.2"** the results are analyzed and displayed in the list.



```

try
{
    // Step: 13.1
    //
    // check if it has to be synchronized with the thread of the form
    if ( this.InvokeRequired == true )
    {
        this.BeginInvoke ( new Opc.Da.ReadCompleteEventHandler ( OnReadComplete ),
            new Object[] { clientHandle, opcValueResults } );
        return;
    }

    // stop updating
    this.ListView_ItemData.BeginUpdate();

    // Step: 13.2
    //
    // update item values
    ListViewItem listViewItem;
    foreach (Opc.Da.ItemValueResult opcValueResult in opcValueResults)
    {
        listViewItem = this.ListView_ItemData.FindItemWithText ( opcValueResult.ItemName );

        if ( ( listViewItem != null ) && ( opcValueResult.Value != null ) )
        {
            listViewItem.SubItems[1].Text = this.GetValueAsString ( opcValueResult );
            listViewItem.SubItems[2].Text = opcValueResult.Quality.ToString ();
        }
    }
}
catch ( Exception except )

```

Fig. 6-42: Analyzing data - Reading

Writing Control Data

General Information

This section describes the write access to control data.

A distinction is made between

- "synchronous" and
- "asynchronous" access.

Control data is written in the `"Button_Write_Click"` event handler of the **"Write"** button.

The relevant code segments in the OPC Client example project are:

- **"Step: 14"** - Synchronous writing
- **"Step: 15"** - Asynchronous writing
- **"Step: 16"** - Analyzing data

Synchronous writing

With a "synchronous" access, the client waits until the server has completed the writing task. The data is written in the "cache" and in the "Device" of the OPC server if the subscription and the item are active. The client does not accept **any further operating actions** during this period, i.e. the "synchronous" call is blocking.

"Synchronous" writing (**"Step: 14"**) of subscription data is performed via the `"opcResults = Me.m_OpcServer.Subscriptions[0].Write()"` function call and the transfer of the `"Opc.Da.ItemValue"` (`"opcValues"`) to be written as parameters.

In this example, only one item from the entire subscription is written and the write task result is forwarded to the `"OnWriteComplete"` function for processing.

Client Development

The "OnWriteComplete" function is described in [chapter "Analyzing data" on page 212](#).

Asynchronous writing

In case of an "asynchronous" access, the task is transferred to the server and the function call returns immediately. The server then processes the task and reports the task processing result back to the client via a predefined "Delegate" ("OnWriteComplete" event handler). This means that the client can **accept and process further actions** via the server during task processing.

The "asynchronous" writing (**"Step: 15"**) is performed as for "synchronous" writing via the "opcResults = this.m_OpcServer.Subscriptions[0].Write()" function call.

The call differs only in the number of parameters to be applied. The transfer parameters and their description are listed in the following table.

Transfer parameters	Description
Opc.Da.ItemValue()	Item collection of all items to be written with values. As for synchronous writing.
Object	ID transferred from the client even to the "Delegate" when processing the results (optionally "null").
WriteCompleteEventHandler	"Delegate" to process the task results. "Delegate" is to be transferred to the function call using the instruction "new WriteCompleteEventHandler(OnWriteComplete)".
IRequest	Handle to interrupt the task via the "Subscription.Cancel()" function call (optionally "null").

Tab. 6-12: Transfer parameters for asynchronous writing

```

// check command type - sync or async
if ( this.RadioButton_Sync.Checked == true )
{
    // Step: 14
    // synchronous write
    Opc.IdentifiedResult[] opcIdentResults =
        this.m_OpcServer.Subscriptions[0].Write ( opcItemValues );
    this.OnWriteComplete(null, opcIdentResults);
}
else if ( this.RadioButton_Async.Checked == true )
{
    // Step: 15
    // asynchronous write
    Opc.IRequest opcRequest;
    this.m_OpcServer.Subscriptions[0].Write ( opcItemValues, null,
        new Opc.Da.WriteCompleteEventHandler ( OnWriteComplete ), out opcRequest );
}

```

Fig. 6-43: Synchronous and asynchronous writing of control data

Analyzing data

The "OnWriteComplete" function analyzes the results retrieved from the OPC server. It is called by the OPC Client after "synchronous" writing and as "event handler" ("Delegate") by the OPC server when completing the task processing after the "asynchronous" command.

As control elements are not "thread-safe", i.e. these objects can only be accessed by one thread. Verify whether the "OnWriteComplete" function call is running in the same thread as form thread. This is verified in **"Step: 16.1"**.

If the function is not called in the form thread, the call is synchronized by the "this.BeginInvoke()" function.

In "Step: 16.2" the results of the writing task can be analyzed. It is analyzed in a "log file" or e.g. in a "StatusStrip" control element of the application. The example client does not contain any analysis of the writing task result.

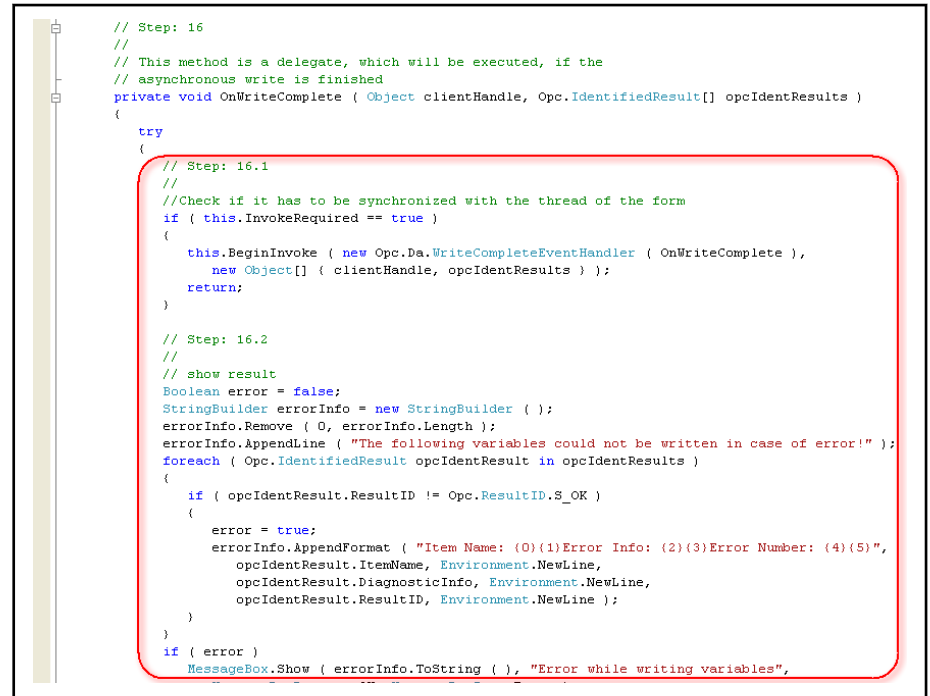


Fig. 6-44: Analyzing Data - Writing

Cyclic Reading of Control Data

General Information

This section describes cyclic reading of control data.

The prerequisite for cyclic reading of values is that

- the subscription and
- the item are active and that
- an event handler is logged on to the subscription.

Activation and deactivation of cyclic reading are controlled via the "RadioButton_CheckedChanged" of the "Radio Buttons" "Synchronous", "Asynchronous" and "Cyclic".

The relevant code segments in the OPC Client example project are:

- "Step: 17" - Activating cyclic reading
- "Step: 18" - Deactivating cyclic reading
- "Step: 19" - Analyzing data

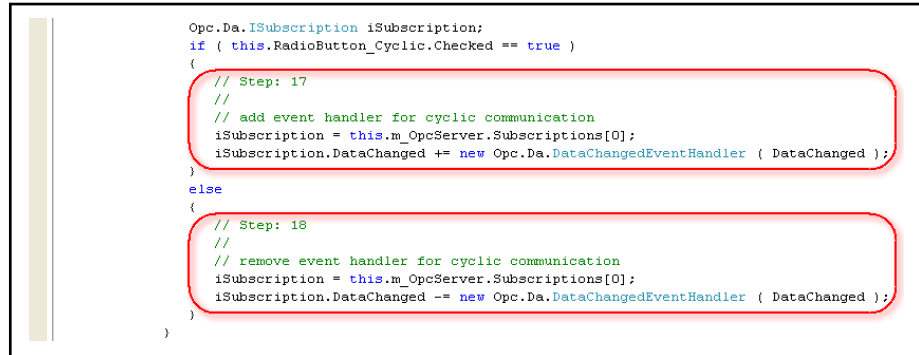
Activating cyclic reading

As the prerequisites of the active subscription and active items are met, the "DataChanged" event handler only has to be added to the "iSubscription.DataChanged" subscription list ("Step: 17"). Adding the event handler meets all prerequisites and the client receives the "DataChanged" events.

Client Development

Deactivating Cyclic Reading

To deactivate cyclic reading, the "DataChanged" event handler is removed again from the "iSubscription.DataChanged" subscription list. Once removed, no further value changes can be reported to the client.



```

Opc.Da.ISubscription iSubscription;
if ( this.RadioButton_Cyclic.Checked == true )
{
    // Step: 17
    //
    // add event handler for cyclic communication
    iSubscription = this.m_OpcServer.Subscriptions[0];
    iSubscription.DataChanged += new Opc.Da.DataChangedEventHandler ( DataChanged );
}
else
{
    // Step: 18
    //
    // remove event handler for cyclic communication
    iSubscription = this.m_OpcServer.Subscriptions[0];
    iSubscription.DataChanged -= new Opc.Da.DataChangedEventHandler ( DataChanged );
}
}

```

Fig. 6-45: Adding and removing event handling

Analyzing data

The "DataChanged" function is requested cyclically by the OPC server at the time intervals specified in the "UpdateRate" element of the "Opc.Da.SubscriptionState" class.

All items are transferred to the "DataChanged" function with the corresponding values if

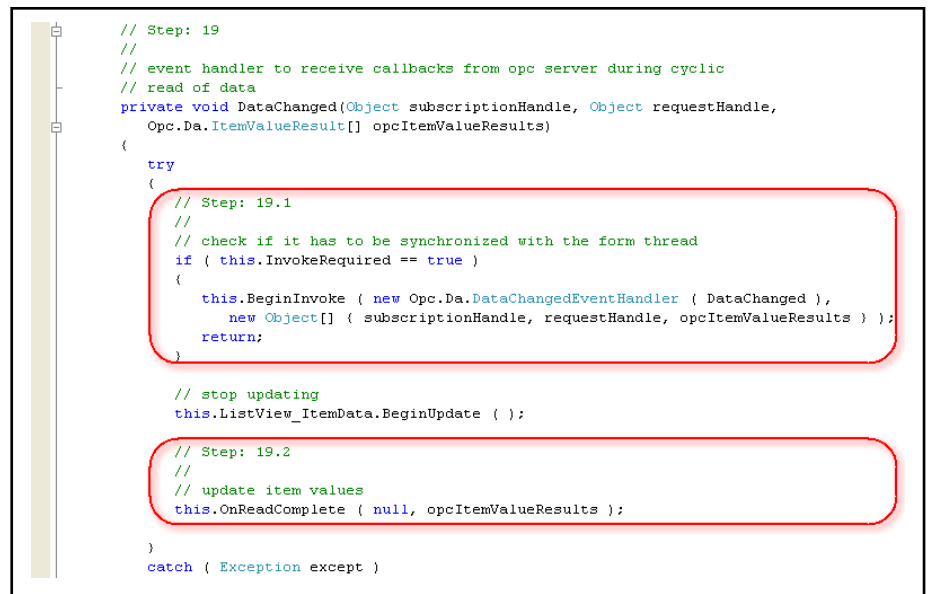
- the value has changed or
- the maximum "age" of the value was exceeded.

This means it depends on the configuration of the "MaxAge" and "MaxAgeSpecified" elements of the "Opc.Da.Item" class.

As control elements are not "thread-safe", i.e. these objects can only be accessed by one thread. Verify whether the "DataChanged" function thread is running in the same thread as form thread. This is verified in **"Step: 19.1"**.

If the function is not called in the form thread, the call is synchronized by the "this.BeginInvoke()" function.

In **"Step: 19.2"** the results are analyzed and displayed in the list view as with synchronous and asynchronous reading.



```

// Step: 19
//
// event handler to receive callbacks from opc server during cyclic
// read of data
private void DataChanged(Object subscriptionHandle, Object requestHandle,
    Opc.Da.ItemValueResult[] opcItemValueResults)
{
    try
    {
        // Step: 19.1
        //
        // check if it has to be synchronized with the form thread
        if ( this.InvokeRequired == true )
        {
            this.BeginInvoke ( new Opc.Da.DataChangedEventHandler ( DataChanged ),
                new Object[] { subscriptionHandle, requestHandle, opcItemValueResults } );
            return;
        }

        // stop updating
        this.ListView_ItemData.BeginUpdate ( );

        // Step: 19.2
        //
        // update item values
        this.OnReadComplete ( null, opcItemValueResults );
    }
    catch ( Exception except )
    {
    }
}

```

Fig. 6-46: Evaluating data - Cyclic reading

Error Handling

Only a minimum of error handling was implemented with the example OPC Client. Analyze and record errors occurring with an OPC Client used on production machines more in detail.

The OPC Client is therefore restricted to verify variables and the simple exception handling as

- "If" queries and
- exception handling with "try...catch..." instructions

"If" queries

"If" enquiries check the parameters required to perform the function. If the specified definitions cannot be fulfilled, the function is canceled.

Exception handling with "try...catch..." instructions

As the majority of components report an error to the caller during execution of a function by "throwing" an exception, at least one "Try...Catch..." instruction should be included in every function/method. In the example OPC Client, only one "Try...Catch..." instruction was programmed due to the scope of the individual functions.

7 Service and Support

Our worldwide service network provides an optimized and efficient support. Our experts offer you advice and assistance should you have any queries. You can contact us **24/7**.

Service Germany Our technology-oriented Competence Center in Lohr, Germany, is responsible for all your service-related queries for electric drive and controls.

Contact the **Service Helpdesk & Hotline** under:

Phone: **+49 9352 40 5060**
Fax: **+49 9352 18 4941**
E-mail: service.svc@boschrexroth.de
Internet: <http://www.boschrexroth.com>

Additional information on service, repair (e.g. delivery addresses) and training can be found on our internet sites.

Service worldwide Outside Germany, please contact your local service office first. For hotline numbers, refer to the sales office addresses on the internet.

Preparing information To be able to help you more quickly and efficiently, please have the following information ready:

- Detailed description of malfunction and circumstances resulting in the malfunction
- Type plate name of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your email address)

Index

A

Abbreviations.....	16
About this documentation.....	9
Validity of the documentation.....	9
Activate device configuration.....	93
Ad Windows form and control element.....	175
Add devices.....	89
Application examples.....	133
HMI of third party manufacturers.....	159
IndraLogic visualization.....	159
Overview.....	133
Symbol configuration.....	133
Symbol configuration via pragma.....	135
VI-Composer.....	136
WinStudio with IndraLogic XLC.....	136

B

BRWS driver.....	79
Areas of application.....	80
Configuration.....	80
Data volume.....	83
Documentation.....	88
Driver sheets.....	81
Error messages.....	87
Message information.....	87
Properties.....	79
Protocol structure.....	83
Support PLC variables and parameters.....	84

C

Change devices OPC alias.....	92
Close applications.....	93
Communication example via OPC UA	
Subscribe and write.....	125
Compact industrial PC VPB 40.3 with the operating device VDP 08.3/16.3/40.3	
Brief description.....	34
Compact industrial PC VPB 40.3 with the operating device VDP xx.3	
Documentation.....	37
Variants.....	35
Compact industrial PC VPB 40.3 with the operating device VDPxx.3	
Accessories.....	36
Field of application.....	34
Technical data.....	34
Wear parts.....	35
Compact industrial PC VPP 16.3/40.3/60.3	
Brief description.....	37
Field of application.....	38
Technical data.....	38
Compact industrial PC VPP xx.3	
Accessories.....	39
Variants.....	39
Wear parts.....	38

Compact industrial PC VPPxx.3	
Documentation.....	40
Configure IndraLogic OPC.DA server.....	96
Configure server.....	97
Item syntax.....	107
OPCServer.ini.....	100
Server configuration.....	97
Storage directory.....	104
Storage directory Windows 7.....	105
Storage directory Windows CE.....	105
Storage directory Windows XP.....	104
Configure IndraLogic OPC.DA Server	
Determine communication address.....	97
Configure OPC server connection with WinStudio.....	107
Configure OPC-UA client connection using WinStudio.....	128
Connector panel.....	28
Create device configuration.....	89

D

DC 24 V voltage supply.....	28
Determine communication address.....	97
Diagnostic displays.....	29
Display device properties.....	90

E

Error messages Bosch Rexroth WinStudio driver.....	87
Ethernet interface.....	29

H

Housing and mounting dimensions of the VR 21xx.....	27
---	----

I

Implement OPC Client.....	177
IndraControl VCH.....	22
IndraControl VCH 08 - Compact hand-held terminal for mobile use	
Brief description.....	22
IndraControl VCH 08 – Compact hand-held terminal for mobile use	
Accessories.....	23
Documentation.....	23
Technical data.....	22
Variants.....	23
IndraControl VCH 08 –Compact hand-held terminal for mobile use	
Field of application.....	22
IndraControl VCP.....	17
IndraControl VEP 30.3/40.3/50.3 / VEP 30.4/40.4/50.4 – Embedded PC-based operating terminals	
Brief description.....	31

Index

Variants.....	32	Analyze data.....	210, 212, 214
IndraControl VEP 30.5/40.5/50.5 / VEP		Asynchronous reading.....	210
30.4/40.4/50.4 – Embedded PC-based operating terminals		Asynchronous writing.....	212
Field of application.....	31	Cancel subscription - CancelSubscription... ..	204
Technical data.....	31	Create and cancel OPC server subscription	203
Wear parts.....	32	Create and delete OPC server instance.....	200
IndraControl VEP 40.5/50.5 / VEP		Create OPC server instance - Connect.....	201
30.4/40.4/50.4 – Embedded PC-based operating terminals		Create subscription - CreateSubscription....	203
Accessories.....	33	Cyclic reading of control data.....	213
Documentation.....	33	Deactivate cyclic reading.....	214
IndraControl VPB and VDP.....	34	Declare variables.....	199
IndraControl VPP.....	37	Delete OPC server instance - Disconnect... ..	201
IndraControl VR21.....	23	Display subscription information.....	205
IndraLogic items.....	95	Error handling.....	215
IndraLogic OPC.DA.....	96	Exception handling with try...catch... instructions.....	215
Exclusion of liability.....	107	Implement OPC Client.....	198
Properties.....	96	IndraLogic OPC server.....	207
Information representation		List control items.....	205
Names and abbreviations.....	16	List OPC server.....	200
Interfaces.....	28	New project.....	195
Ethernet interface.....	29	Opc.Da.BrowseFilter.....	205
X9, X10 USB interfaces.....	29	Opc.ItemIdentifier.....	205
Introduce OPC Unified Architecture		OPC.lwSCP server.....	207
General information.....	111	Read control data.....	209
OPC UA information model.....	112	Read control items.....	205
Specification overview.....	111	Read server information.....	202
Introduction into the OPC Unified Architecture		Remove items - RemoveItems.....	208
Service-oriented architecture.....	112	Synchronous reading.....	209
Item syntax.....	107	Synchronous writing.....	211
lwSCP items.....	93	Write control data.....	211
lwSCP server.....	93	OPC Client Visual C#.NET	
M		If queries.....	215
Memory card.....	29	OPC UA.....	110
Modbus TCP.....	130	Bosch Rexroth OPC UA server.....	116
Introduction and overview.....	130	Introduce OPC Unified Architecture.....	111
ModbusTCP.....	160	OPC UA communication example.....	121
Configuration with Pro-face GP-Pro EX 2.2.	166	Overview.....	110
Configuration with Siemens WinCC-flexible 2007.....	162	OPC UA communication example	
General.....	160	Browse.....	124
O		Introduction.....	121
OPC Client Visual Basic .NET.....	173	Prerequisites.....	121
Exception handling with Try...Catch... instructions.....	194	UaExpert.....	121
If queries.....	194	OPC UA Communication example	
Visual Basic project OPC Client	173	Establish connection.....	123
Visual C# Project OPC Client.....	195	OPC.DA Server.....	159
OPC Client Visual C# .NET.....	194	Configuration.....	159
Activate cyclic reading.....	213	Transfer configuration on a visualization device VEPxx.4 with WindowsCE.....	160
Add and remove OPC items.....	206	OPC.lwSCP.....	88
Add items - AddItems.....	206	Exclusion of liability	96
Add References.....	195	Properties.....	88
Add Windows form and control elements....	196	OPC.lwSCP items.....	93
		OPC.lwSCP server.....	93
		OPCServer.ini.....	100
		Ordering information.....	30
		Overview.....	17

P

PROFIBUS DP 20

R

Remove devices 92

S

Safety alert symbol 10

Safety instructions 9, 10

Save device configuration 92

Save device configuration as 92

SCP OPC configurator

General 88

Operation 89

SCP-OPC configurator

Activate device configuration 93

Change devices OPC alias 92

Create device configuration 89

Remove devices 92

Save device configuration 92

Save device configuration as 92

SCP-OPC Configurator 88

Add devices 89

Close applications 93

Display device properties 90

Open device configuration 89

Screen size 59

SD memory card 29

sercos items 95

Signal words 10

Slot for memory card 29

Small control panel VCP

Accessories 20

Brief description 17

Documentation 21, 31

Technical data 19

Variants 19

Small Control Panel VR21xx

Brief description 23

Status displays 29

Storage directory - OPC.DA server 104

Windows 7 105

Windows CE 105

Windows XP 104

Support

See service hotline 217

Symbol configuration

General information 133

Symbol configuration with Wizard 134

Symbols 10

T

Technical data 26

Voltage supply 28

Type code 30

U

USB interfaces 29

V

Variants of VR 21 25

VDP 16.3 34

VDP 40.3 34

VDP 60.3 34

VI-Composer

Communication protocol BRC-symbolic 44

Configurable small operating terminals 43

Connection settings 44

Documentation 45

Field of application 43

VI-Logic

Documentation 43

Field of application 41

Properties of the integrated visualization 41

Protocol 43

Supported visualization devices 43

Visualization elements 42

Visual Basic project OPC Client 173

Activate cyclic reading 192

Add items - AddItems 185

Add references 174

Analyze data 189, 191, 193

Asynchronous reading 188

Asynchronous writing 190

 Cancel OPC subscription - CancelSub-
 scription 182

Create OPC server instance - Connect 179

 Create OPC subscription - CreateSub-
 scription 181

Cyclic reading of control data 192

Deactivate cyclic reading 192

Declare variables 178

Delete OPC server instance - Disconnect 180

Display subscription information 183

Error handling 194

Implement OPC Client 177

List control items 183

List OPC server 179

New project 173

Opc.Da.BrowseFilter 184

Opc.ItemIdentifier 184

Read control data 187

Read control items 183

Read server information 181

Remove items - RemoveItems 187

Synchronous reading 187

Synchronous Writing 190

Write control data 190

Visual Basic project OPC Client

Add Windows form and control element 175

IndraLogic OPC server 185

OPC.lwSCP Server 185

Voltage supply 28

Index

Voltage supply (DC 24 V).....	28
VPB 40.....	34

W

WinStudio.....	45
Activate a WinStudio license.....	74
Alarms.....	64
Consequences of a missing or a lower class license.....	74
Create and edit WinStudio screens.....	59
Create WinStudio application.....	75
Cross reference search.....	71
Database spy.....	70
Debugger.....	71
Export/import.....	72
External databases.....	66
General WinStudio settings and functions....	47
Graphic script.....	67
Linked symbols.....	61
LogWin.....	70
Mathematic sheets.....	63
More information.....	77
Multilingualism.....	75
Object properties and variable selection.....	60
Overview.....	45, 75
Recipes.....	64
Reports.....	65
Schedules.....	66
Screen properties.....	59
Screen scripts.....	68
Scripts.....	66
Start application.....	69
Start script.....	67
Transfer application and activation on tar- get device.....	76
Trends.....	66
User management.....	75
User scripts.....	68
User Symbols.....	62
Variable import wizard.....	72
Verify application.....	69
Web functionality.....	75
WinStudio applications.....	75
WinStudio elements in the Project Explorer...	46
WinStudio library.....	61
WinStudio license overview.....	73
WinStudio licensing.....	73
WinStudio screens.....	59
WinStudio tasks.....	63
WinStudio tools.....	69
WinStudio variables.....	60
WinStudio with IndraLogic	
Creating the IndraWorks project.....	137
Creating the WinStudio visualization.....	150
Integrating the WinStudio visualization in IndraWorks.....	143
Starting the interface.....	156

Technical requirements.....	137
-----------------------------	-----

X

X5.....	29
X9.....	29
X10.....	29

Notes

Bosch Rexroth AG
Electric Drives and Controls
P.O. Box 13 57
97803 Lohr, Germany
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr, Germany
Tel. +49 9352 18 0
Fax +49 9352 18 8400
www.boschrexroth.com/electrics



R911336362

DOK-XLCMLC-HMI*****V13-AP02-EN-P